Mathematics for Global Illumination

Massimo Picardello

Mathematics Department, University of Roma "Tor Vergata"

Moscow State University, September 2012

Abstract and disclaimer

This is a simple, almost naif approach to the mathematics of global illumination in Computer Graphics. The speaker is by no way a leading expert in this subject.

Classical Computer Graphics lead to two prominent iterative methods for realistic rendering: Recursive Ray Tracing (where the observer looks at any point x of the scene and gets its diffusive and glossy contributions to illumination, but then his visual ray bounces according to the law of reflection generating a reflected ray, whose hit point y yields an additional purely specular contribution to the illumination of the originating point x, and we trace the rays bounce after bounce, adding up their contributions recursively), and Radiosity (a finite element method for global light interchange). $RRT \Rightarrow$ approximate glossy and diffuse illumination with the addition of enhanced precise highlights. $Rad \Rightarrow$ fine light gradation for purely diffusive environments without highlights or directional effects.

Modern Computer Graphics rediscovers these two methods in one unified photorealistic approach, Global Illumination, based on recursive high-dimensional integral equations solved by tracing rays with suitably chosen probability distribution and computing the

Flux and radiance

We have a scene, made up of surfaces (usually polygons), with one or more light sources.

Definition

- Flux: radiant power, energy / time
- Radiosity: exitant flux / area
- Radiance: flux / (solid angle $\times$ projected area)

Notation

- $L(x \to \vec{\theta})$ *radiance emitted at point* x *in direction* $\vec{\theta}$
- $L(x \leftarrow \vec{\theta})$ *radiance arriving at point* x *from direction* $\vec{\theta}$
- $L(x \to y)$ *radiance emitted at point* x *and arriving at point* y
- $L_e(x \to \vec{\theta})$ *radiance created at point* x *and emitted in direction* $\vec{\theta}$

Invariance: if there are no participating media, then conservation of energy yields $L(x \to y) = L(y \leftarrow x)$.

BRDF

Each surface is assigned appropriate material parameters (typically, the BRDF: *bidirectional reflectance distribution function*).

Definition

BRDF $f_r(x, \vec{\psi} \leftrightarrow \vec{\theta}) = \%$ of radiance arriving at x from direction $\vec{\psi}$ that exits in direction $\vec{\theta}$

Example

- Diffuse reflector: Lambert model, $f_r(x, \vec{\psi} \leftrightarrow \vec{\theta}) = \rho_d/\pi \equiv k_d$, isotropic in the exit angle
- Ideal specular reflector: incident angle $\vec{\psi} \Rightarrow f_r(x, \vec{\psi} \leftrightarrow \cdot) = k_s \delta_{\vec{R}}$, where $\vec{R} = 2\langle \vec{n}, \vec{\psi} \rangle \vec{n} - \vec{\psi}$ is the reflected direction ($\vec{n}$ is the normal versor of the surface at x)
- Approximate specular reflector: Phong model,

$$f_r(x, \vec{\psi} \leftrightarrow \vec{\theta}) = k_s \frac{\langle \vec{R}, \vec{\theta} \rangle^{\text{Phong exponent}}}{\langle \vec{n}, \vec{\psi} \rangle} + k_d$$

The rendering equation: hemispherical formulation

Example (continued)

- Refractive surface: Snell law
- Physically based models: Fresnel equations for amount of reflection and refraction at a smooth surface for the two components of polarized light, and Cook–Torrance microfacets probability distribution to physically approximate any surface with locally smooth surfaces

The rendering equation (hemispherical formulation)

$$L(x \to \vec{\theta}) = L_e(x \to \vec{\theta}) + \int_{\Omega_x} f_r(x, \vec{\psi} \leftrightarrow \vec{\theta})\, L(x \leftarrow \vec{\psi})\, \langle \vec{n}, \vec{\psi} \rangle \, d\vec{\psi}$$

where Ω_x is the front hemisphere at x (we assume that x lies in a non-transparent surface with tangent plane).

The rendering equation: hemispherical formulation

Example (continued)

- Refractive surface: Snell law
- Physically based models: Fresnel equations for amount of reflection and refraction at a smooth surface for the two components of polarized light, and Cook–Torrance microfacets probability distribution to physically approximate any surface with locally smooth surfaces

The rendering equation (hemispherical formulation)

$$L(x \to \vec{\theta}) = L_e(x \to \vec{\theta}) + \int_{\Omega_x} f_r(x, \vec{\psi} \leftrightarrow \vec{\theta})\, L(x \leftarrow \vec{\psi})\, \langle \vec{n}, \vec{\psi} \rangle \, d\vec{\psi}$$

where Ω_x is the front hemisphere at x (we assume that x lies in a non-transparent surface with tangent plane).

The rendering equation: area formulation

Notation

- *A: the union of all the surfaces in the scene*
- $r(x, \vec{\psi})$*: the point of the scene facing* x *in direction* $\vec{\psi}$
- $V(x, y)$*: visibility factor, 1 if* x *and* y *are directly visible (that is* $y = r(x, \vec{\psi})$*), 0 otherwise*
- $G(x, y)$*: geometric factor,*

$$G(x, y) = \frac{\langle \vec{n}_x, \vec{\psi} \rangle \langle \vec{n}_y, -\vec{\psi} \rangle}{\pi \ \mathrm{dist}(x, y)^2}$$

where $y = r(x, \vec{\psi})$

The rendering equation (area formulation)

$$L(x \to \vec{\theta}) = L_e(x \to \vec{\theta}) + \int_A f_r(x, \vec{\psi} \leftrightarrow \vec{\theta})\, L(y \leftarrow -\vec{\psi})\, V(x, y)\, G(x, y)\, d\sigma(y)$$

Recursive Ray Tracing

So we must solve a recursive integral equation. We could try to compute each integral numerically by sufficiently fine sampling of the hemisphere (or the scene area) for each integral: that is, by shooting an equidistributed mesh of rays. Needless to say, this is interminably slow.

Recursive Ray Tracing: approximate each integral

$$\int_A f_r(x, \vec{\psi} \leftrightarrow \vec{\theta})\, L(y \leftarrow -\vec{\psi})\, V(x, y)\, G(x, y)\, d\sigma(y)$$

with the value of the integrand in the direction $\vec{\psi}$ of the light (or of the lights), plus the contribution of just a single a ray shot in the specular direction obtained by reflecting $\vec{\theta}$ at x. This gives the illumination given by the BRDF plus the highlights.

Instead, we shall generate samples with appropriate probability distribution p in the domain of integration of the integral $I = \int f(x)\, dx$

Recursive Ray Tracing

So we must solve a recursive integral equation. We could try to compute each integral numerically by sufficiently fine sampling of the hemisphere (or the scene area) for each integral: that is, by shooting an equidistributed mesh of rays. Needless to say, this is interminably slow.

Recursive Ray Tracing: approximate each integral

$$\int_A f_r(x, \vec{\psi} \leftrightarrow \vec{\theta})\, L(y \leftarrow -\vec{\psi})\, V(x,y)\, G(x,y)\, d\sigma(y)$$

with the value of the integrand in the direction $\vec{\psi}$ of the light (or of the lights), plus the contribution of just a single a ray shot in the specular direction obtained by reflecting $\vec{\theta}$ at x. This gives the illumination given by the BRDF plus the highlights.

Instead, we shall generate samples with appropriate probability distribution p in the domain of integration of the integral $I = \int f(x)\, dx$

Recursive Ray Tracing

So we must solve a recursive integral equation. We could try to compute each integral numerically by sufficiently fine sampling of the hemisphere (or the scene area) for each integral: that is, by shooting an equidistributed mesh of rays. Needless to say, this is interminably slow.

Recursive Ray Tracing: approximate each integral

$$\int_A f_r(x, \vec{\psi} \leftrightarrow \vec{\theta})\, L(y \leftarrow -\vec{\psi})\, V(x,y)\, G(x,y)\, d\sigma(y)$$

with the value of the integrand in the direction $\vec{\psi}$ of the light (or of the lights), plus the contribution of just a single a ray shot in the specular direction obtained by reflecting $\vec{\theta}$ at x. This gives the illumination given by the BRDF plus the highlights.

Instead, we shall generate samples with appropriate probability distribution p in the domain of integration of the integral $I = \int f(x)\, dx$.

Numerical integration with random sampling

Lemma (Monte Carlo estimator)

With N samples x_i:

- *estimator:*

$$\langle I \rangle = \frac{1}{N} \sum_{i=1}^{N} \frac{f(x_i)}{p(x_i)}$$

- *variance*

$$\sigma^2 = \frac{1}{N} \int \left(\frac{f(x)}{p(x)} - I \right)^2 p(x)\, dx$$

Remark

In dimension 1, Monte Carlo integration does not offer great advantages over deterministic integration (quadrature formulas): variance $= 1/\sqrt{N} \leftrightarrow N$ samples $\leftrightarrow$ quadrature error $\approx \|f'\|_\infty / N$. But to achieve the same accuracy in dimension $d > 1$, the quadrature formulas need a mesh of N^d points, whereas the Monte Carlo method still needs N samples. The rendering integrals are in dim 2 (actually, in dim 4 for radiosity).

Reduction of variance

Warning

Variance means noise in the rendered image!

Methods of reduction of variance:

- Importance sampling: variational methods and the Lagrange multipliers yield the following:

Proposition (Distribution with minimal variance)

The variance is zero if p is proportional to f:

$$p(x) = \frac{f(x)}{\int f(t)\,dt}$$

Of course one does not know $\int f(t)\,dt$: this is what we want to estimate. But if one has an idea of how f looks like, this proposition tells us which type of probability distributions are more suitable.

- Stratified sampling: subdivide the domain of integration into m disjoint subdomain (*strata*), and perform Monte Carlo integration separately in each stratum D_j with n_j samples.

Proposition

If all strata have equal size then the variance is smallest if $n_j \equiv 1$ (one sample per stratum)

- How to perform stratified sampling in dimension 2 without using N^2 samples? N-rook algorithm: the strata are the cells of a 2-dimensional grid, choose only one sample in each row and column (just N samples!). Also possible for d-dimensional grids with $d > 2$.
- Several other variance reduction schemes: combining importance and stratified sampling and quasi Monte Carlo, combining several estimators and applying multiple importance, control variates. More difficult... our rendering integrand has many factors: often one chooses a probability distribution proportional to the most relevant factor.

Stochastic path tracing

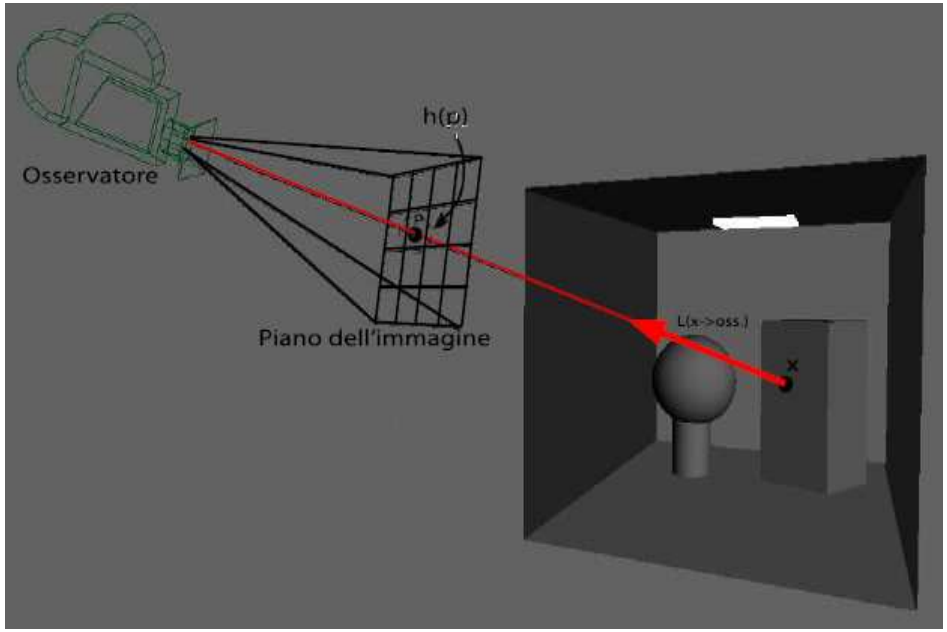


Figure : First step of path tracing

Rendering equation:

$$L(x \to \vec{\theta}) = L_e(x \to \vec{\theta}) + \int_{\Omega_x} f_r(x, \vec{\psi} \leftrightarrow \vec{\theta})\, L(x \leftarrow \vec{\psi})\, \langle \vec{n}, \vec{\psi} \rangle \, d\vec{\psi}$$

Call the integral $L_r(x \to \vec{\theta})$ (r stands for *reflected*).

Estimator of the integral:

$$\langle L_r(x \to \vec{\theta}) \rangle = \frac{1}{N} \sum_{i=1}^{N} \frac{f_r(x, \vec{\psi}_i \leftrightarrow \vec{\theta})\, L(x \leftarrow \vec{\psi}_i)\, \langle \vec{n}, \vec{\psi}_i \rangle}{p(\vec{\psi}_i)}$$

But $L(x \leftarrow \vec{\psi}_i) = L(r(x, \vec{\psi}_i) \to \vec{\psi}_i)$. So we must trace the ray from x in direction $\vec{\psi}_i$ to find $y = r(x, \vec{\psi}_i)$ and calculate the same integral at y, and so on recursively: we trace a tree of rays. But how do we stop the recursion?

- Trivial way: stop after a prescribed number of steps, even adaptively based on predetermined bounds on the residual radiance (everything decreases as k_s^n, exponentially in the BDRF maximum reflectance). But deterministic stopping of a stochastic estimator introduces bias: inaccurate rendering.
- Stochastic stopping time: Russian Roulette.

Estimator of the integral:

$$\langle L_r(x \to \vec{\theta}) \rangle = \frac{1}{N} \sum_{i=1}^{N} \frac{f_r(x, \vec{\psi}_i \leftrightarrow \vec{\theta})\, L(x \leftarrow \vec{\psi}_i)\, \langle \vec{n}, \vec{\psi}_i \rangle}{p(\vec{\psi}_i)}$$

But $L(x \leftarrow \vec{\psi}_i) = L(r(x, \vec{\psi}_i) \to \vec{\psi}_i)$. So we must trace the ray from x in direction $\vec{\psi}_i$ to find $y = r(x, \vec{\psi}_i)$ and calculate the same integral at y, and so on recursively: we trace a tree of rays. But how do we stop the recursion?

- Trivial way: stop after a prescribed number of steps, even adaptively based on predetermined bounds on the residual radiance (everything decreases as k_s^n, exponentially in the BDRF maximum reflectance). But deterministic stopping of a stochastic estimator introduces bias: inaccurate rendering.
- Stochastic stopping time: Russian Roulette.

Russian Roulette

We want to estimate an integral: for simplicity, take $I = \int_0^1 f(x)\,dx$. The Monte Carlo method selects random points $x_i \in [0,1]$ and computes a weighted average of the values $f(x_i)$. Instead, compress the domain by a factor $P < 1$ and dilate the values of f by $1/P$: we have

$$I = I_{RR} := \int_0^P \frac{1}{P} f\left(\frac{x}{P}\right)\,dx$$

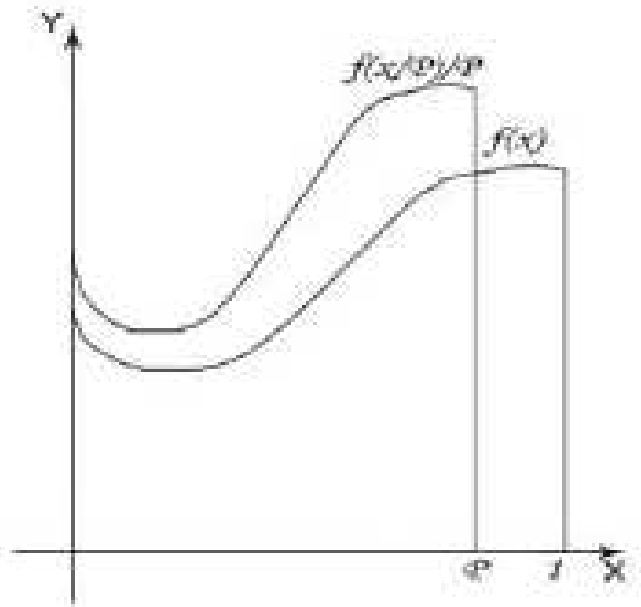


Figure : Russian Roulette estimator

Hence, by sampling I_{RR} with the uniform probability distribution in $[0, 1]$:

Corollary

The expectation of the estimator

$$\langle I_{RR} \rangle = \begin{cases} \frac{1}{P} f\left(\frac{x}{P}\right) & if \quad 0 \leqslant x \leqslant P \\ 0 & otherwise \end{cases}$$

is $\langle I_{RR} \rangle = I$.

Remark

With probability $P < 1$*, the sample points generated in the Russian Roulette are smaller than* P*: their contributions yield the unbiased estimate for* I*. The remaining samples are larger than* P *and yield 0. We say that* P *is the survival probability, and* $1 - P$ *is the absorption probability.*

If we use Russian Roulette to compute the integral of the rendering equation, the absorption probability gives a random absorption for the traced rays, hence a random stopping time.

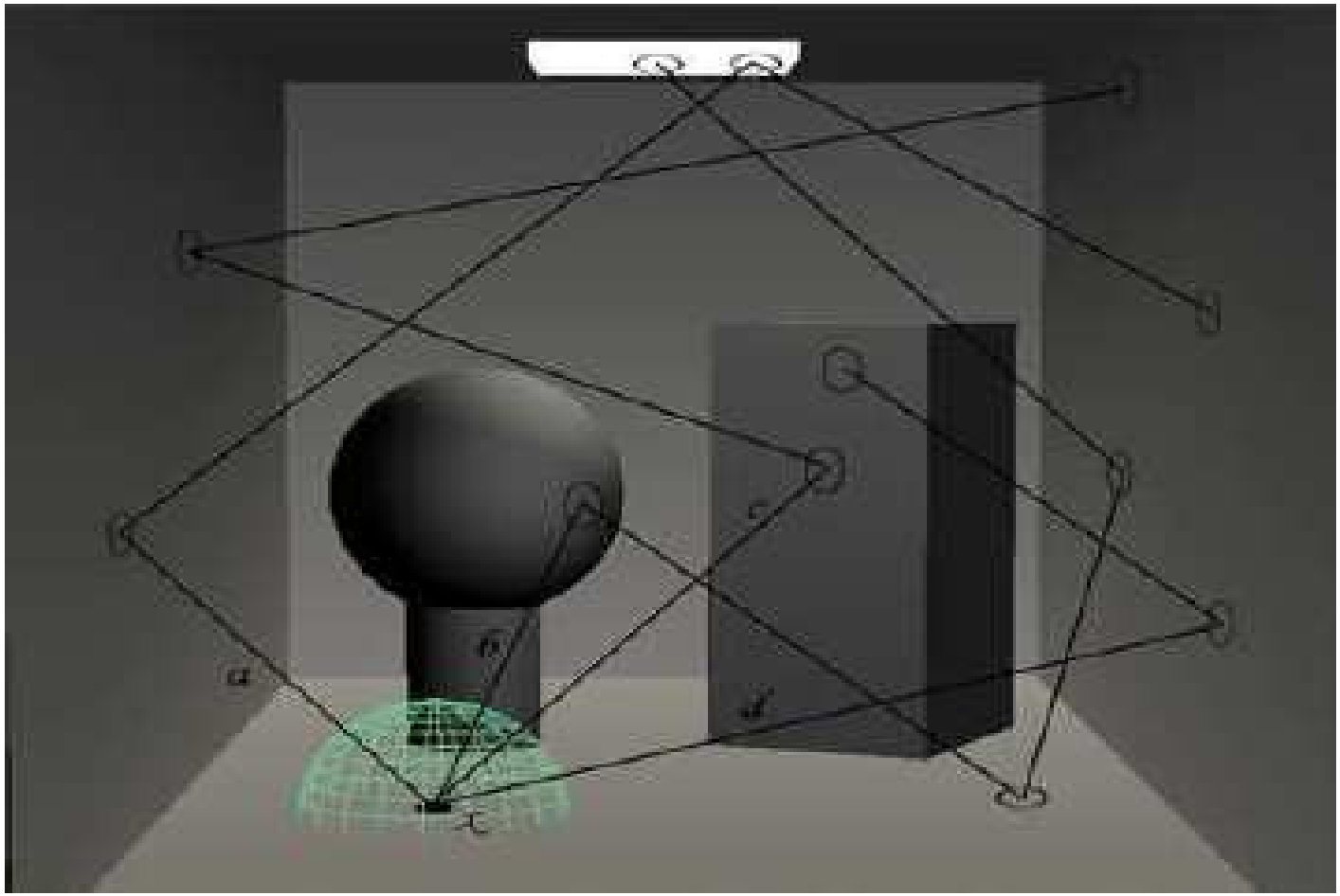


Figure : Simple stochastic ray tracing, with stochastic absorption

Direct and indirect illumination in stochastic path tracing

We have already rewritten the rendering equation for the reflected radiance as

$$L_r(x \to \vec{\theta}) = \int_{\Omega_x} f_r(x, \vec{\psi} \leftrightarrow \vec{\theta})\, L(r(x, \vec{\psi}) \to -\vec{\psi})\, \langle \vec{n}, \vec{\psi} \rangle \, d\vec{\psi}$$

Let $y = r(x, \vec{\psi})$ and decompose $L(y \to -\vec{\psi})$ as the sum of self-emitted and reflected radiance, $L_e(y \to -\vec{\psi}) + L_r(y \to -\vec{\psi})$. This yields

$$L_r(x \to \vec{\theta}) = L_{\mathrm{dir}}(x \to \vec{\theta}) + L_{\mathrm{indir}}(x \to \vec{\theta})$$

The first term on the right is the direct contribution at x from light sources (in classical Computer Graphics this is the Lambert and Phong contribution); the second term is the indirect contribution at x due to reflections from the rest of the scene (the classical Recursive Ray Tracing term). We now study the direct illumination term.

Direct illumination

Notation

The direction versor $\vec{\psi}$ from x to y is denoted by $x\vec{y}$.

Let N_L be the number of light sources in the scene: each corresponds to a light-emitting surface A_k. Let us rewrite the direct illumination term in area formulation:

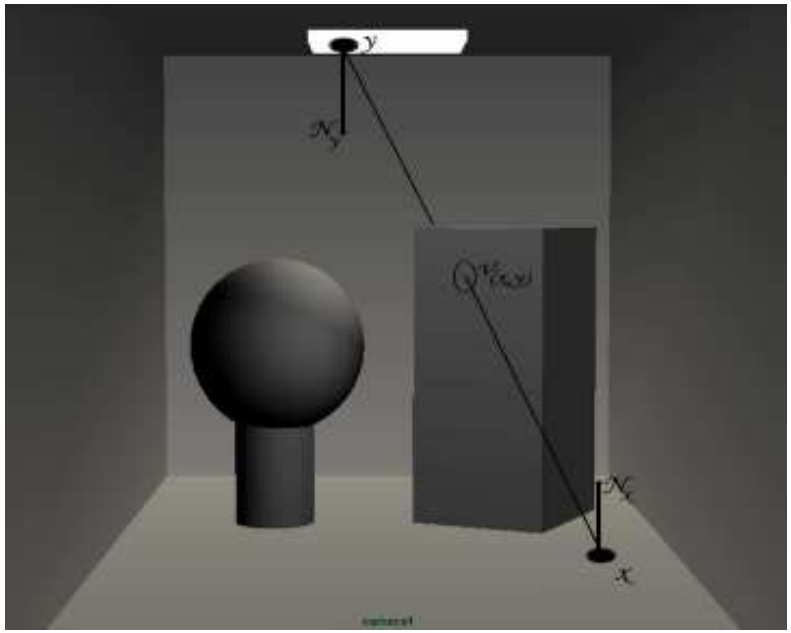
$$L_{\mathrm{dir}}(x \to \vec{\theta}) = \sum_{k=1}^{N_L} \int_{A_k} f_r(x, x\vec{y} \leftrightarrow \vec{\theta})\, L_e(y \to -x\vec{y})\, V(x,y)\, G(x,y)\, d\sigma(y)$$

So we need to generate sample points y on the light sources.

Shadow rays

The term $V(x, y)$ must be calculated by tracing a ray towards y to check if x is in shadow: this is why the rays towards the sampling points on the lights are called a shadow rays. To decrease variance:

- increase the total number of shadow rays;
- distribute shadow rays per light proportionally to respective powers (importance sampling);
- distribute shadow rays to privilege the areas of most impact (for instance the closest parts of the lights).



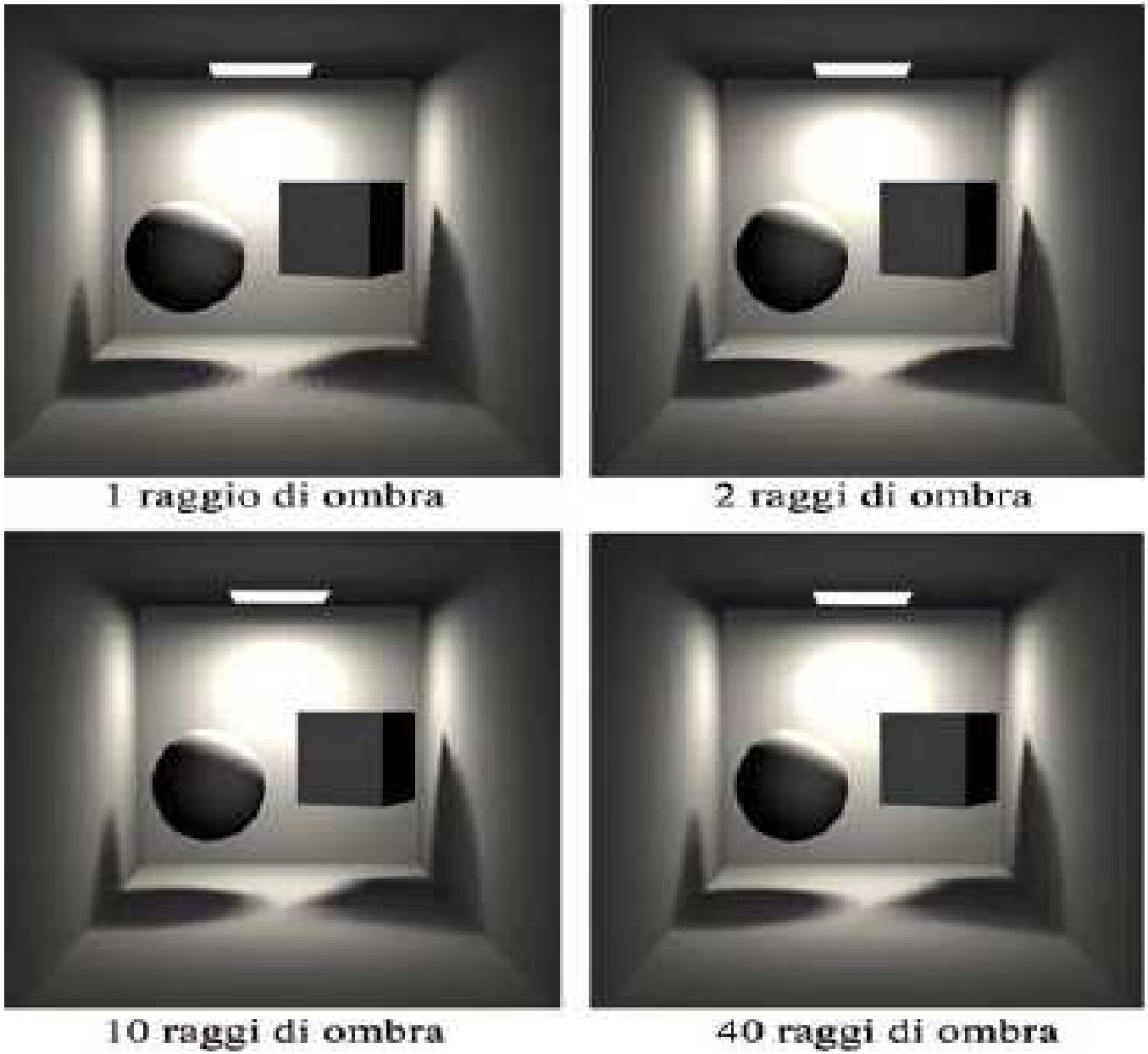


Figure : Sampling light areas with shadow rays: uniform sampling of one light source, 1, 2, 10, 40 shadow rays per pixel respectively

Single light source

Assume only one light. With N_S shadow rays, the estimator for direct illumination is

$$\langle L_{\text{dir}}(x \to \vec{\theta}) \rangle = \frac{1}{N_S} \sum_{k=1}^{N_L} \frac{f_r(x, x\vec{y}_i \leftrightarrow \vec{\theta})\, L_e(y_i \to y_i\vec{x})\, V(x, y_i)\, G(x, y_i)}{p(y_i)}$$

How to choose p to reduce variance? We see that there are various factors at the numerator, and we should choose p to cancel one of them (importance sampling) One factor is the visibility factor; another is the radiance-emission term: for isotropic (=diffuse) lights this is constant, and so it does not introduce variance. Then there are the occlusion and geometric factors. In the hemispherical formulation, some of these are replaced by another factor: the cosine term.

Variance reduction for direct illumination

Typical choices for p:

- uniformly distributed sampling over the light source: for the x in penumbra, not all shadow rays yield a contribution > 0, so there the variance (=noise) increases; for the x in full light, the variance comes from the $\mathrm{dist}(x, y)^2$ at the denominator of the geometric factor $G(x, y)$, particularly when the lights are widespread and some parts of are much closer to x than others;
- uniform sampling of solid angle subtended by the light source: by returning to the hemispheric formulation, we can choose p proportional to the cosine term. This would be better for sources that, seen from x, are foreshortened and cover a small solid angle. However, it is not simple to uniformly sample a general solid angle.
- if light source are not isotropic: it would be useful to sample proportionally to their emission distribution, but very difficult because we must restrict to directions towards x.

Multiple lights

Two strategies:

- treat each light source separately (as if there was only one light);
- consider their union as a unique light source.

In the second case, two steps: in the first step we choose the light to be sampled by means of a discrete probability distribution; in the second step, we sample the selected light source area with the methods seen before, by using a conditional probability distribution function.

Two overall strategies:

- uniform source selection + uniform sampling of light source area;
- power-proportional source selection + uniform sampling of light source area.

Clearly, the second strategy is more efficient (importance sampling). But, two problems:

- for points x that do not see the brightest lights, a lot of variance because most shadow rays give no contribution, and slower convergence because all the rendering comes from the dim lights that are scarcely sampled;

- to generate a shadow ray, three random numbers: two for the choice of the point y on the light source, and one to select which source; adding random choices makes stratified sampling more difficult.

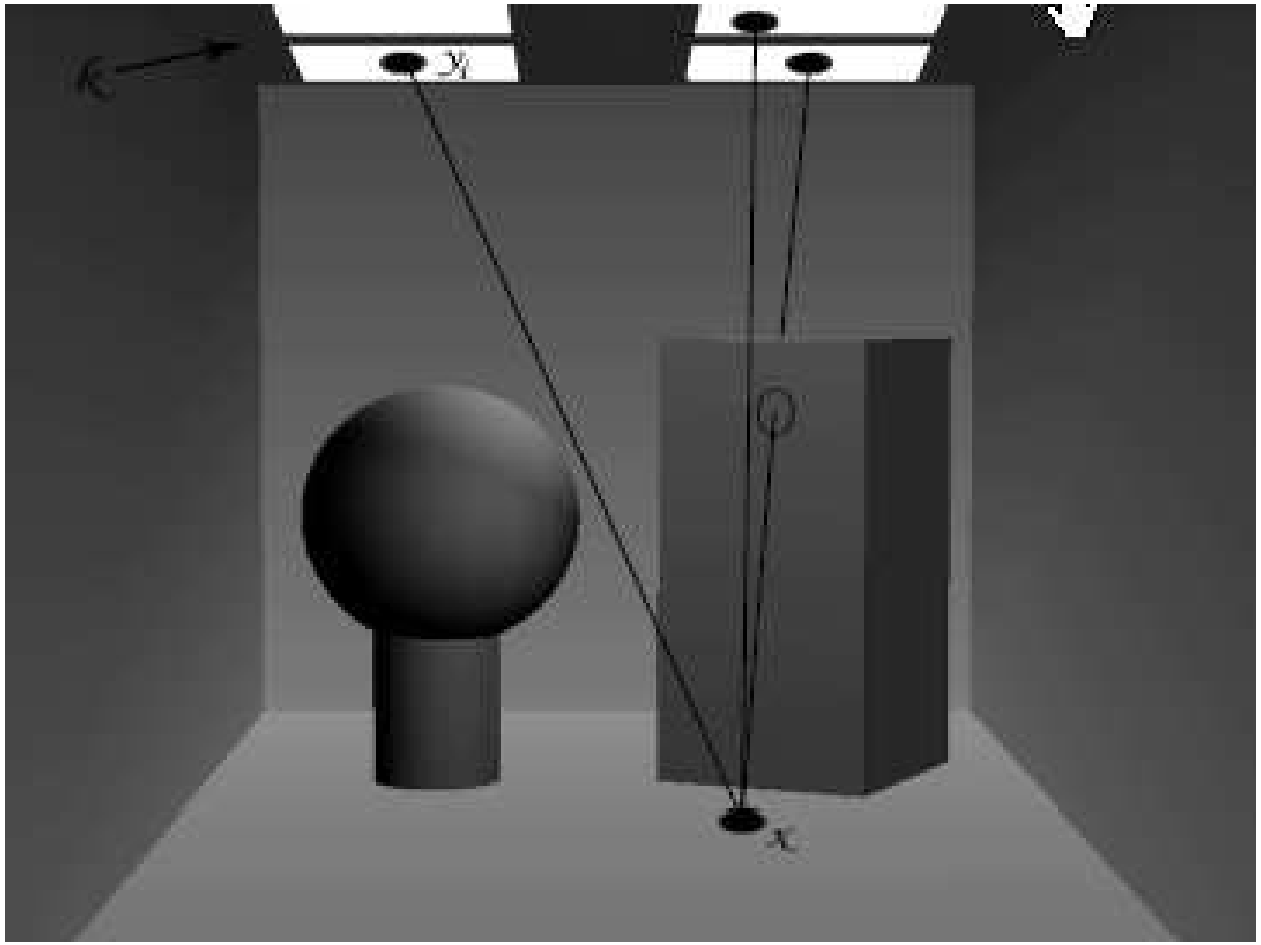


Figure : Sampling two light areas with shadow rays

Indirect illumination

This is the term where we have recurrence:

$$L_r(x \to \vec{\theta}) = \int_{\Omega_x} f_r(x, \vec{\psi} \leftrightarrow \vec{\theta})\, L_r(r(x, \vec{\psi}) \to -\vec{\psi})\, \langle \vec{n}, \vec{\psi} \rangle \, d\vec{\psi}$$

Estimator with N samples:

$$\langle L_r(x \to \vec{\theta}) \rangle = \frac{1}{N} \sum_{i=1}^{N} \frac{f_r(x, \vec{\psi}_i \leftrightarrow \vec{\theta})\, L_r(r(x, \vec{\psi}_i) \to -\vec{\psi}_i)\, \langle \vec{n}, \vec{\psi}_i \rangle}{p(\vec{\psi}_i)}$$

In addition, there is a self-emitted term. We trace paths (and for each hit a shadow ray towards the light sources, to obtain a sample value of the integrand) until absorption. At each hit on a light source, the self-emitted light contribution is accumulated. So at each bounce this recursive integral stores the effect of indirectly visible light sources.

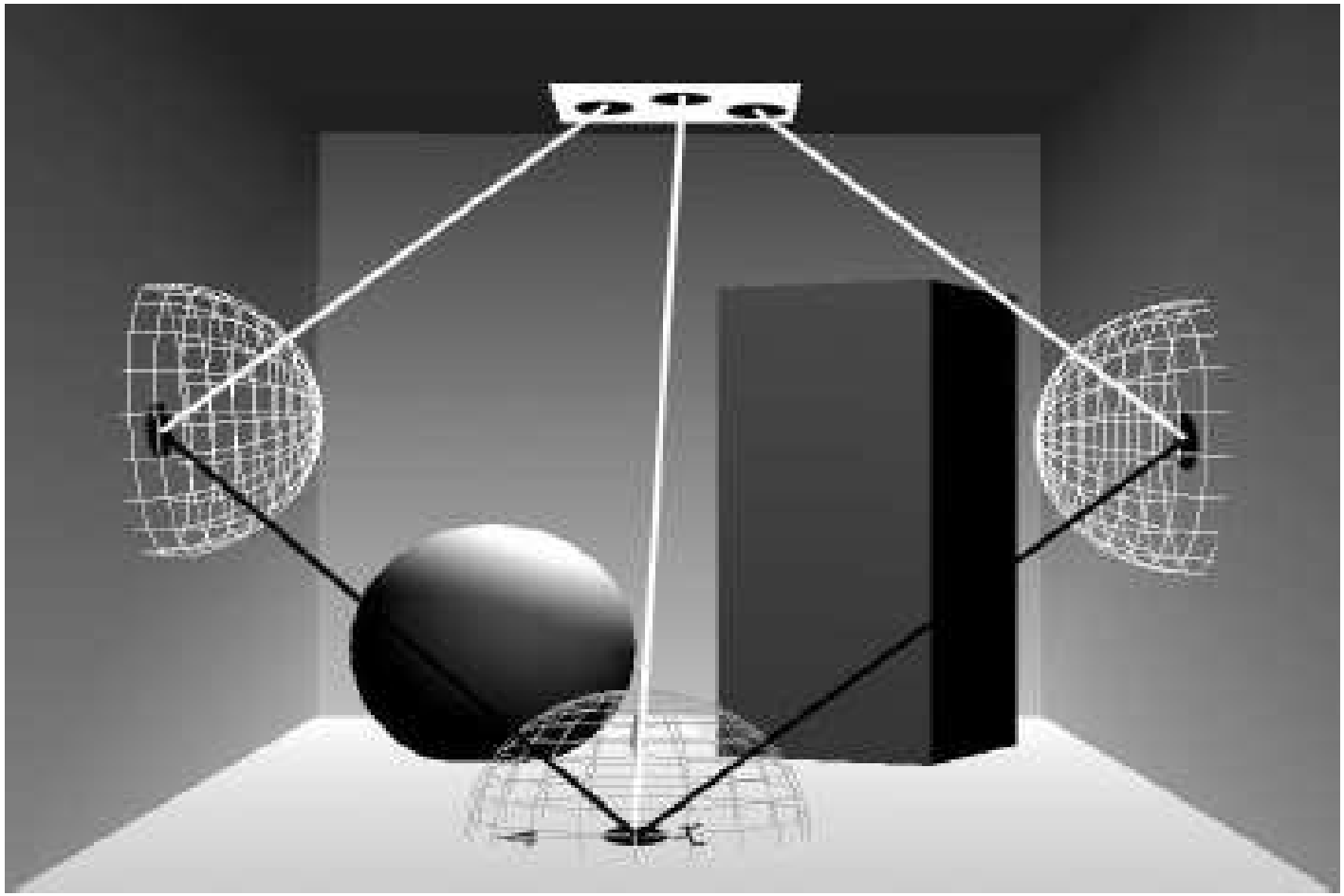


Figure : Sampling indirect illumination: bouncing paths are black, shadow rays white

Strategies for variance reduction

For Russian roulette: survival probabilities proportional to the reflectance of the point where reflection occurs (or the angular average of the BRDF).
For the integrand: choose the probability distribution p proportional to one of the following:

- the cosine term;
- the BRDF: for instance, if the BRDF comes from the Lambert+Phong model, we randomly generate three events according to which we choose p proportional to the diffuse or glossy term of the BRDF, or else absorption;
- the incident radiance field L_r: but this is unknown, we can only make an adaptive choice at each hitting point by approximation based on prediction from the previous steps of the recurrence process.
- better: the product of all three... but... sampling according to a function of more variables is very time consuming and yields variance.

Light tracing

The light tracing method traces rays not from the camera but from the light source (or sources if there are more than one - a discrete probability distribution, that may be weighted with importance, selects from which light area we start the tracing for each bouncing process). Instead of shadow rays now we trace observer rays, or camera rays to check if the bounce point is visible through each pixel.
Since there are few lights but many pixels, this method is less efficient. However, it is important in two-pass algorithms that combine path tracing and light tracing and interpolate and add the light contribution (after storing the partial result in an appropriately efficient data storage at the end of the first pass and retrieving quickly the data after the second pass).

The need for the light tracing pass is due to the fact that it lets light rays naturally accumulate and become sampled at high density in the areas of the scene where the light is concentrated by the optics (glass spheres, lenses, the effect of waves on the shading at the bottom of a shallow sea or pool). These areas, called caustics, have extremely high contrast and require high density sampling, that would not be possible with path tracing from the camera unless the sample count per pixel is chosen to be prohibitively high.

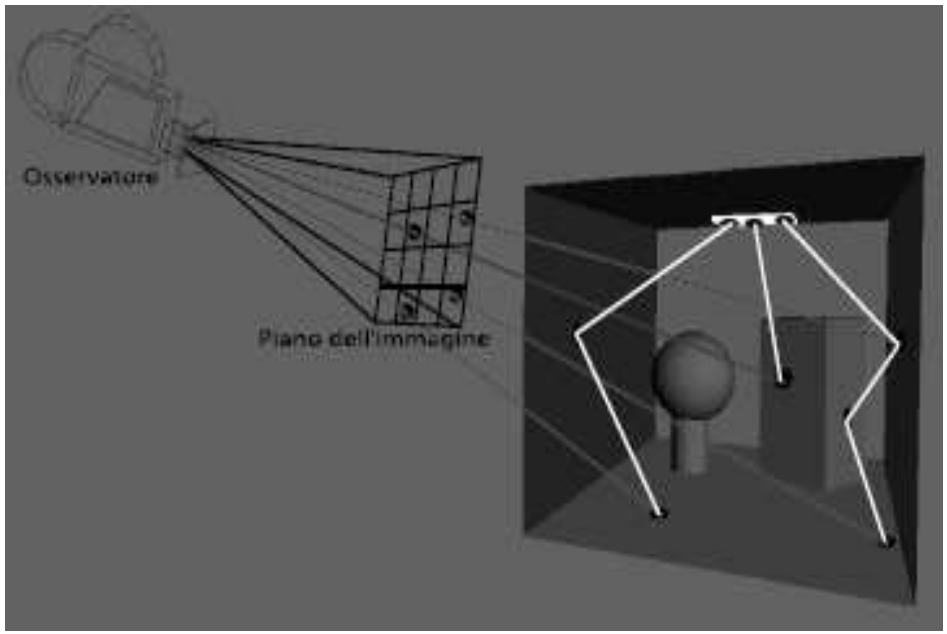


Figure : Sampling light tracing: the light rays are drawn in white, the camera rays in red

Radiosity

Here is the classical finite element method for light transport, revised in terms of the rendering equation. The scene is subdivided into N patches S_i of area A_i. We now suppose that the scene is purely diffusive, so there is no directional distribution of reflected light. We may as well integrate over exit angles in the rendering equation for radiance: note that the incoming radiance still depends on the incoming angle, but the outgoing radiance now becomes a function L only of the point x, without angles, since diffuse materials emit isotropically (Lambert model). This way we obtain the following 2-dimensional integral for the outgoing radiance $L(x)$ and 4-dimensional for the *average* radiosity b_i of S_i:

$$L(x) = L_e(x) + \int_{\Omega_x} f_r(x)\, L(x \leftarrow \vec{\psi})\, \langle \vec{n}_x,\, \vec{\psi} \rangle\, d\vec{\psi}$$

$$b_i = \frac{1}{A_i} \int_{S_i} \int_{\Omega_x} L(x)\, \langle \vec{n}_x,\, \vec{\theta} \rangle\, d\vec{\theta}\, d\sigma(x) \qquad (1)$$

Radiosity from the rendering equation

Since for the facing point $y = r(x, -\vec{\psi})$ one has $L(x \leftarrow \vec{\psi}) = L(y \rightarrow -\vec{\psi}) = L(y)$, the area formulation of the rendering equation yields

$$L(x) = L_e(x) + f_r(x) \int_S L(y)\, K(x,y)\, d\sigma(y) \tag{2}$$

where $K(x,y) = G(x,y)\, V(x,y)$. It is trivial to check that the integral of the cosine of the latitude over an hemisphere is π. So $b(x) = \int_{\Omega_x} L(x) \langle \vec{n}_x, \vec{\theta} \rangle \, d\vec{\theta} = \pi L(x)$ and by (1)

$$b_i = \frac{1}{A_i} \int_{S_i} L(x) \int_{\Omega_x} \langle \vec{n}_x, \vec{\theta} \rangle \, d\vec{\theta}\, d\sigma(x) = \frac{1}{A_i} \int_{S_i} b(x)\, d\sigma(x)$$

Part of the radiosity is self-emitted and part reflected. If the patches are small enough that we can approximate $b(x) = b_i$ and $f_r(x) = \rho_i$ with constants on each patch, this and (2) yield

$$b_i = b_i^{(e)} + \rho_i \sum_{j=1}^{N} F_{ij} b_j$$

where $F_{ij} = \frac{1}{A_i} \int_{S_i} \int_{S_j} K(x,y)\, d\sigma(y)\, d\sigma(x)$ is the form factor. Note that the integral is 4-dimensional, and that the linear system has the recursive form of type $\vec{b} = \vec{e} + A\vec{b}$, where $A = \mathbb{I} - \{\rho_i F_{ij}\}$.
Observe that the kernel K is invariant by swapping S_i and S_j: so, because of the division by the area, we have the *reciprocity relation*

$$A_i\, F_{ij} = A_j\, F_{ji}\,. \tag{3}$$

The radiosity linear system

Now let $M_{ij} = \delta_{ij} - \rho_i F_{ij}$. Then the vector of radiosities of the patches is the solution of the following *radiosity linear system*:

$$M\vec{b} = \vec{e}$$

or, in recursive form,

$$\vec{b} = \vec{e} + A\vec{b}\,.$$

The classical Radiosity method

- split the scene into patches
- to each path assign constant reflectivity ρ_i
- compute the form factors from their 4-dimensional integrals and store them as necessary
- solve the radiosity linear system numerically via iterative methods like Jacobi, Gauss–Seidel or Southwell
- visualize the results (normally this is done by transferring radiosities of patches to luminosities of vertices, and then computing the shading per pixel by bilinear interpolation of the luminosities of the projected vertices)

The method converges because the radiosity matrix M is strictly diagonally dominant. But the computation of the form factors makes it very costly time-wise, and their storage very costly in terms of memory.

Geometric meaning of the form factors

The form factor from patch i (S_i) to patch j (S_j) is expressed by $F_{ij} = \frac{1}{A_i} \int_{S_i} \int_{S_j} K(x,y)\, d\sigma(y)\, d\sigma(x)$. For any point $x \in S_i$ let us consider the inner integral, $\int_{S_j} K(x,y)\, d\sigma(y)$, where $K(x,y) = V(x,y)\, G(x,y)$, the product of the visibility factor V and the geometric factor

$$G(x,y) = \frac{\langle \vec{n}_x, \vec{\psi} \rangle \langle \vec{n}_y, -\vec{\psi} \rangle}{\pi \ \mathrm{dist}(x,y)^2}\ .$$

This geometric factor measures the projected area of S_j onto the unit frontal hemisphere centered at x, weighted with the cosine of the incident direction. So, since the integral of the cosine over the hemisphere cancels out the π at the denominator, the integral of GV over $y \in S_j$ measures the solid angle of S_j when seen from point x. Thus, the form factor is the *average* solid angle covered by S_j when seen from points in S_i.

Computing form factors: local lines

Two methods to compute form factors.
Local lines: for Monte Carlo computation of the integral of the form factor,

- select equidistributed random points $x_i \in S_i$, that is with probability distribution $p_{x_i} = 1/A_i$
- select random directions $\vec{\theta} \in \Omega_{x_i}$ with conditional probability given by the cosine, $p_{\vec{\theta}|x_i} = \frac{1}{\pi}\langle \vec{\theta}, \vec{n}_{x_i} \rangle$
- form the joint probability $p(x, \vec{\theta}) = p_{x_i} p_{\vec{\theta}|x_i}$
- let $\chi_j(x_i, \vec{\theta}) := 1$ if the ray from $x_i \in S_i$ with direction $\vec{\theta}$ hits S_j, and $\chi_j = 0$ otherwise

Then

Proposition (sampling the set of form factors from S_i)

The expectation of χ_j w.r. to the joint probability $p_{x_i} p_{\vec{\theta}|x_i}$ is exactly F_{ij}.

More precisely:

Proposition (form factors by local line sampling)

The Monte Carlo estimator with n_i samples

$$\langle F_{ij} \rangle = \frac{1}{n_i} \sum_{k=1}^{n_i} \frac{\chi_j \left(y_k^{(i)} \right)}{p \left(y_k^{(i)} \right)}$$

is unbiased and has expectation $E(\langle F_{ij} \rangle) = F_{ij}$. Its variance is $\sigma^2 = \frac{1}{n_j} F_{ij}(1 - F_{ij})$.

Remark

The probability that a particle shoot from S_i with distribution probability p hits S_j is the form factor F_{ij}.

Computing form factors: global lines

Encompass all the scene in a large sphere and draw N lines with both ends equidistributed on this sphere. Integral geometry shows that

Proposition (form factors by global line sampling)

- *these lines hit each patch in an equidistributed way (note: any two consecutive intersections of a ray with the patches yields two local lines, one starting at each of the two intersections and ending at the other, in opposite orientations)*
- *if A_T is the total area of the scene and N_i is the number of hits on patch S_i, then $N_i \approx NA_i/A_T$*
- *if N_{ij} is the number of lines with consecutive intersections in S_i and S_j, then $N_{ij}/N_i \approx F_{ij}$*

Form factors need no longer to be computed: we only need to know that they are expectations of hits of rays shot with a given probability, then we sample with respect to that probability to cancel the form factors out of Monte Carlo estimators!

Computing form factors: global lines

Encompass all the scene in a large sphere and draw N lines with both ends equidistributed on this sphere. Integral geometry shows that

Proposition (form factors by global line sampling)

- *these lines hit each patch in an equidistributed way (note: any two consecutive intersections of a ray with the patches yields two local lines, one starting at each of the two intersections and ending at the other, in opposite orientations)*
- *if A_T is the total area of the scene and N_i is the number of hits on patch S_i, then $N_i \approx NA_i/A_T$*
- *if N_{ij} is the number of lines with consecutive intersections in S_i and S_j, then $N_{ij}/N_i \approx F_{ij}$*

Form factors need no longer to be computed: we only need to know that they are expectations of hits of rays shot with a given probability, then we sample with respect to that probability to cancel the form factors out of Monte Carlo estimators!

Stochastic Relaxation methods for the radiosity linear system: Jacobi iteration

The radiosity linear system is expressed in recursive form as $\vec{b} = \vec{e} + A\vec{b}$: for simplicity, let us write $\vec{b} = \vec{e} + A\vec{b}$.

Jacobi iterations:

$$\vec{b}^{(k+1)} = \vec{e} + A\vec{b}^{(k)}$$

If A is a contraction (that is, it decreases distance between pairs of points), the iterative method obviously converges to one and the same limit vector (the *equilibrium radiosity distribution*) whichever starting vector $\vec{b}^{(k)}$ is chosen. So, the convergence condition is spectral radius $\rho_A < 1$ (*Theorem: this is automatically satisfied since the radiosity matrix is strongly diagonally dominant*).

Stochastic Relaxation methods for the radiosity linear system: Jacobi iteration

The radiosity linear system is expressed in recursive form as $\vec{b} = \vec{e} + A\vec{b}$: for simplicity, let us write $\vec{b} = \vec{e} + A\vec{b}$.

Jacobi iterations:

$$\vec{b}^{(k+1)} = \vec{e} + A\vec{b}^{(k)}$$

If A is a contraction (that is, it decreases distance between pairs of points), the iterative method obviously converges to one and the same limit vector (the *equilibrium radiosity distribution*) whichever starting vector $\vec{b}^{(k)}$ is chosen. So, the convergence condition is spectral radius $\rho_A < 1$ (*Theorem: this is automatically satisfied since the radiosity matrix is strongly diagonally dominant*).

Three iteration schemes:

(*i*) Gathering: A natural interpretation of the equations: at each iteration, each patch in turn gathers radiance from all others: $b_i^{(k+1)} = e_i + \rho_i \sum_j F_{ij} b_j^{(k)}$, where $b_j^{(k)}$ represents the radiance collected by the patch i from the emission of the patch j.

(*ii*) Shooting: Transform the linear sytem of radiance into the dual system of energy (by multiplication by the patch areas): because of the reciprocity relation (3) this yields $p_i^{(k+1)} = \epsilon_i + \rho_i \sum_j F_{ji} p_j^{(k)}$, where $p_j^{(k)}$ represents the power shot into the scene by the patch i from the emission of the patch j.

(*iii*) Incremental shooting:

$$\triangle p_i^{(0)} = \epsilon_i$$

$$\triangle p_i^{(k+1)} = \rho_i \sum_j \triangle p_j^{(k)} F_{ji}$$

$$p_i^{(k)} = \sum_{m=0}^{k} \triangle p_i^{(m)}$$

Three iteration schemes:

(*i*) Gathering: A natural interpretation of the equations: at each iteration, each patch in turn gathers radiance from all others: $b_i^{(k+1)} = e_i + \rho_i \sum_j F_{ij} b_j^{(k)}$, where $b_j^{(k)}$ represents the radiance collected by the patch i from the emission of the patch j.

(*ii*) Shooting: Transform the linear sytem of radiance into the dual system of energy (by multiplication by the patch areas): because of the reciprocity relation (3) this yields $p_i^{(k+1)} = \epsilon_i + \rho_i \sum_j F_{ji} p_j^{(k)}$, where $p_j^{(k)}$ represents the power shot into the scene by the patch i from the emission of the patch j.

(*iii*) Incremental shooting:

$$\triangle p_i^{(0)} = \epsilon_i$$

$$\triangle p_i^{(k+1)} = \rho_i \sum_j \triangle p_j^{(k)} F_{ji}$$

$$p_i^{(k)} = \sum_{m=0}^{k} \triangle p_i^{(m)}$$

Three iteration schemes:

(*i*) Gathering: A natural interpretation of the equations: at each iteration, each patch in turn gathers radiance from all others: $b_i^{(k+1)} = e_i + \rho_i \sum_j F_{ij} b_j^{(k)}$, where $b_j^{(k)}$ represents the radiance collected by the patch i from the emission of the patch j.

(*ii*) Shooting: Transform the linear sytem of radiance into the dual system of energy (by multiplication by the patch areas): because of the reciprocity relation (3) this yields $p_i^{(k+1)} = \epsilon_i + \rho_i \sum_j F_{ji} p_j^{(k)}$, where $p_j^{(k)}$ represents the power shot into the scene by the patch i from the emission of the patch j.

(*iii*) Incremental shooting:

$$\triangle p_i^{(0)} = \epsilon_i$$

$$\triangle p_i^{(k+1)} = \rho_i \sum_j \triangle p_j^{(k)} F_{ji}$$

$$p_i^{(k)} = \sum_{m=0}^{k} \triangle p_i^{(m)}$$

Here $\triangle p_i^{(k+1)}$ represents the incremental power collected during iteration $k+1$ from the other patches, and the interpretation is that each patch collects power at each stage, gathering a potential that will be shot at the next shooting: this scheme iterates the redistribution among the patches of their potential energy.
If we would proceed deterministically, now this would suggest to choose an iteration method like Southwell relaxation, where at each step the shooting occurs at the patch that has more potential energy $\triangle p_i^{(k+1)}$ to release. Instead, we proceed probabilistically.

Stochastic incremental shooting iteration scheme

Each iteration method has a stochastic counterpart. No more computing and storing the form factors! The samples automatically yield this required information on the spot.

For instance: stochastic incremental shooting iteration scheme,

$$\triangle p_i^{(k+1)} = \rho_i \sum_j \triangle p_j^{(k)} F_{ji} = \sum_{j,l} \triangle p_j^{(k)} F_{ji} \rho_l \delta_{li} . \qquad (4)$$

- Pick pairs of patched (j, l), say, by local line sampling, as follows:
  1. select source patch j with probability π_j proportional to its unshot power $\triangle p_j^{(k)} / \triangle p_{Tot}^{(k)}$, where $\triangle p_{Tot}^{(k)} := \sum_j \triangle p_j^{(k)}$
  2. select destination patch l with conditional probability $\pi_{l|j} = F_{jl}$, the solid angle covered by l, but without really computing F_{jl}: just trace a local line as sample (and *if visibility* $= 0$, *discard*).
- Now the combined probability of picking (j, l) is $\pi_{lj} = \pi_j \, \pi_{l|j} = \triangle p_j^{(k)} F_{jl} / \triangle p_{Tot}^{(k)}$
- In the estimator, these probabilities go to the denominator of (4) and cancel out the terms $\triangle p_j^{(k)} F_{ji}$

Stochastic incremental shooting iteration scheme

Each iteration method has a stochastic counterpart. No more computing and storing the form factors! The samples automatically yield this required information on the spot.

For instance: stochastic incremental shooting iteration scheme,

$$\triangle p_i^{(k+1)} = \rho_i \sum_j \triangle p_j^{(k)} F_{ji} = \sum_{j,l} \triangle p_j^{(k)} F_{ji} \rho_l \delta_{li} . \tag{4}$$

- Pick pairs of patched (j, l), say, by local line sampling, as follows:
  1. select source patch j with probability π_j proportional to its unshot power $\triangle p_j^{(k)} / \triangle p_{Tot}^{(k)}$, where $\triangle p_{Tot}^{(k)} := \sum_j \triangle p_j^{(k)}$
  2. select destination patch l with conditional probability $\pi_{l|j} = F_{jl}$, the solid angle covered by l, but without really computing F_{jl}: just trace a local line as sample (and *if visibility* $= 0$, *discard*).
- Now the combined probability of picking (j, l) is $\pi_{lj} = \pi_j \, \pi_{l|j} = \triangle p_j^{(k)} F_{jl} / \triangle p_{Tot}^{(k)}$
- In the estimator, these probabilities go to the denominator of (4) and cancel out the terms $\triangle p_j^{(k)} F_{ji}$

Stochastic incremental shooting iteration scheme

Each iteration method has a stochastic counterpart. No more computing and storing the form factors! The samples automatically yield this required information on the spot.

For instance: stochastic incremental shooting iteration scheme,

$$\triangle p_i^{(k+1)} = \rho_i \sum_j \triangle p_j^{(k)} F_{ji} = \sum_{j,l} \triangle p_j^{(k)} F_{ji} \rho_l \delta_{li} . \tag{4}$$

- Pick pairs of patched (j, l), say, by local line sampling, as follows:
  1. select source patch j with probability π_j proportional to its unshot power $\triangle p_j^{(k)} / \triangle p_{Tot}^{(k)}$, where $\triangle p_{Tot}^{(k)} := \sum_j \triangle p_j^{(k)}$
  2. select destination patch l with conditional probability $\pi_{l|j} = F_{jl}$, the solid angle covered by l, but without really computing F_{jl}: just trace a local line as sample (and *if visibility* $= 0$, *discard*).
- Now the combined probability of picking (j, l) is $\pi_{lj} = \pi_j \, \pi_{l|j} = \triangle p_j^{(k)} F_{jl} / \triangle p_{Tot}^{(k)}$
- In the estimator, these probabilities go to the denominator of (4) and cancel out the terms $\triangle p_j^{(k)} F_{ji}$

Stochastic incremental shooting iteration scheme

Each iteration method has a stochastic counterpart. No more computing and storing the form factors! The samples automatically yield this required information on the spot.

For instance: stochastic incremental shooting iteration scheme,

$$\triangle p_i^{(k+1)} = \rho_i \sum_j \triangle p_j^{(k)} F_{ji} = \sum_{j,l} \triangle p_j^{(k)} F_{ji} \rho_l \delta_{li} . \tag{4}$$

- Pick pairs of patched (j, l), say, by local line sampling, as follows:
  1. select source patch j with probability π_j proportional to its unshot power $\triangle p_j^{(k)} / \triangle p_{Tot}^{(k)}$, where $\triangle p_{Tot}^{(k)} := \sum_j \triangle p_j^{(k)}$
  2. select destination patch l with conditional probability $\pi_{l|j} = F_{jl}$, the solid angle covered by l, but without really computing F_{jl}: just trace a local line as sample (and *if visibility* $= 0$, *discard*).
- Now the combined probability of picking (j, l) is $\pi_{lj} = \pi_j \, \pi_{l|j} = \triangle p_j^{(k)} F_{jl} / \triangle p_{Tot}^{(k)}$
- In the estimator, these probabilities go to the denominator of (4) and cancel out the terms $\triangle p_j^{(k)} F_{ji}$

- So, the estimate for $\triangle p_i^{(k+1)}$ turns out to be simply

$$\triangle p_{Tot}^{(k)} \sum_{j,l} \rho_l \, \delta_{li} = \rho_i \triangle p_{Tot}^{(k)} N_i \, ,$$

where N_i is the total number of hits on patch i of sampling local lines (starting at all patches j). Similar procedure with global line.

Incremental versus regular shooting

Unlike the deterministic method, in the stochastic shooting the results of subsequent iterations are averaged: new iterations do not replace old ones. So we use more samples: no discarding. But higher order interreflections become bright slowly because of the averaging (they are not present at the early iterations). To make up, two steps:

1. First a complete incremental shooting iteration sequence until convergence: interreflections ok, but noisy;
2. after, using this equilibrium power distribution as if it would define initial radiance conditions (=lamps), one regular shooting, and average of the two results. The outcome is equivalent to one iteration with twice as many samples.

- So, the estimate for $\triangle p_i^{(k+1)}$ turns out to be simply

$$\triangle p_{Tot}^{(k)} \sum_{j,l} \rho_l \, \delta_{li} = \rho_i \triangle p_{Tot}^{(k)} N_i \, ,$$

where N_i is the total number of hits on patch i of sampling local lines (starting at all patches j). Similar procedure with global line.

Incremental versus regular shooting

Unlike the deterministic method, in the stochastic shooting the results of subsequent iterations are averaged: new iterations do not replace old ones. So we use more samples: no discarding. But higher order interreflections become bright slowly because of the averaging (they are not present at the early iterations). To make up, two steps:

1. First a complete incremental shooting iteration sequence until convergence: interreflections ok, but noisy;
2. after, using this equilibrium power distribution as if it would define initial radiance conditions (=lamps), one regular shooting, and average of the two results. The outcome is equivalent to one iteration with twice as many samples.

- So, the estimate for $\triangle p_i^{(k+1)}$ turns out to be simply

$$\triangle p_{Tot}^{(k)} \sum_{j,l} \rho_l \, \delta_{li} = \rho_i \triangle p_{Tot}^{(k)} N_i \, ,$$

where N_i is the total number of hits on patch i of sampling local lines (starting at all patches j). Similar procedure with global line.

Incremental versus regular shooting

Unlike the deterministic method, in the stochastic shooting the results of subsequent iterations are averaged: new iterations do not replace old ones. So we use more samples: no discarding. But higher order interreflections become bright slowly because of the averaging (they are not present at the early iterations). To make up, two steps:

1. First a complete incremental shooting iteration sequence until convergence: interreflections ok, but noisy;
2. after, using this equilibrium power distribution as if it would define initial radiance conditions (=lamps), one regular shooting, and average of the two results. The outcome is equivalent to one iteration with twice as many samples.

Gathering

Instead, gathering collects samples shot from each patch. So it is useful to clean noise on small patches, that have low probability od being hit at random. So we perform incremental + regular shooting and take the average of the two as initial radiosity distribution, then we perform one iteration of gathering and we finally average.

Discrete random walks

Let p_{ei} be the self-emitted power from patch i, and $p_{eT} = \sum_i p_{ei}$ be the total self-emitted power. After normalization, the radiosity power system becomes

$$\frac{p_i}{p_{eT}} = \frac{p_{ei}}{p_{eT}} + \sum_j \frac{p_j}{p_{eT}} F_{ji}\, \rho_i \,. \tag{5}$$

We may regard the unknowns in this linear system as numbers of visits of a random walk according to the following model. A particle jumps from an urn to another. There are n urns. The "transition probability" from urn i to urn j are p_{ij}, but they do not need to add up to 1, they can be smaller, and $\alpha_i = 1 - \sum_j p_{ij}$ is the absorption probability at urn j. The starting urn is i with probability π_i (*birth probability*). There are counters for the number of visits of the particle at each urn. Suppose that there are N particles jumping. Since a particle visiting urn i is either born there or it comes from urn j with probability p_{ji}, the expected number of visits at urn i clearly is

$$C_i = N\pi_i + \sum_j C_j p_{ji} \,.$$

Averaging with respect to the samples, i.e. dividing by the number N of particles, we get the *collision densities* at the urns:

$$\chi_i = \pi_i + \sum_j \chi_j p_{ji} \, .$$

This is the same as (5) with birth probabilities $\pi_i = p_i / p_{eT}$ and transition probabilities $p_{ji} = F_{ji} \, \rho_i$.

Shooting random walk method: survival estimation

1. A particle is generated at some patch (=urn) i with birth probability π_i.
2. A jump $i \to j$ is selected by sampling a local line.
3. An acceptance/rejection test is performed with survival probability ρ_j. If the test fails, the particle is *absorbed*. If the particle is absorbed, it is discarded, otherwise it increments the counter $C_i = C_i^S$. Particles' births at the light sources are not counted, because we do not need to estimate the lamps' intensity, it is known a priori (*source term estimation suppression*).

Shooting or gathering random walk estimators

The previous random walk estimator is of type *shooting survival*: "shooting" because the particles originate at the light sources (the generators of self-emitted power), and light particles are traced henceforth as reflected by surfaces; "survival" because they are counted only in case of survival. The shooting estimators have dual estimators: *gathering*. In shooting, each particle produces a contribution where it hits, while in gathering it produces a contribution where it is shot. Gathering traces light backward, that is, it does not simulate light propagation: it traces rays, not light paths. These rays are multiply reflected until they hit the light sources or are absorbed. Gathering estimators are less efficient, but may help to clean noise in small patches $\Rightarrow$ do shooting + gathering, and average the results with appropriate weights to minimize the total variance. Here we limit attention to shooting random walks.

Shooting survival estimator of power

$$\frac{p_i}{p_{eT}} \approx \frac{C_i^S}{N}.$$

Shooting or gathering random walk estimators

The previous random walk estimator is of type *shooting survival*: "shooting" because the particles originate at the light sources (the generators of self-emitted power), and light particles are traced henceforth as reflected by surfaces; "survival" because they are counted only in case of survival. The shooting estimators have dual estimators: *gathering*. In shooting, each particle produces a contribution where it hits, while in gathering it produces a contribution where it is shot. Gathering traces light backward, that is, it does not simulate light propagation: it traces rays, not light paths. These rays are multiply reflected until they hit the light sources or are absorbed. Gathering estimators are less efficient, but may help to clean noise in small patches $\Rightarrow$ do shooting + gathering, and average the results with appropriate weights to minimize the total variance. Here we limit attention to shooting random walks.

Shooting survival estimator of power

$$\frac{p_i}{p_{eT}} \approx \frac{C_i^S}{N}.$$

Shooting or gathering random walk estimators

The previous random walk estimator is of type *shooting survival*: "shooting" because the particles originate at the light sources (the generators of self-emitted power), and light particles are traced henceforth as reflected by surfaces; "survival" because they are counted only in case of survival. The shooting estimators have dual estimators: *gathering*. In shooting, each particle produces a contribution where it hits, while in gathering it produces a contribution where it is shot. Gathering traces light backward, that is, it does not simulate light propagation: it traces rays, not light paths. These rays are multiply reflected until they hit the light sources or are absorbed. Gathering estimators are less efficient, but may help to clean noise in small patches $\Rightarrow$ do shooting + gathering, and average the results with appropriate weights to minimize the total variance. Here we limit attention to shooting random walks.

Shooting survival estimator of power

$$\frac{p_i}{p_{eT}} \approx \frac{C_i^S}{N}.$$

Collision estimation

Ray tracing is time consuming. If a particle is absorbed after hitting a patch, its contribution is discarded, but we have traced the ray nevertheless. Better to use another counter: count particles visiting a patch, whether absorbed or not. Total number of visits at i: counter C_i. Of course, survival expectation is $C_i^S \approx \rho_i \, C_i$. Therefore:

Collision estimator of power

$$\frac{p_i}{p_{eT}} \approx \rho_i \, \frac{C_i}{N} \, .$$

Collision estimation

Ray tracing is time consuming. If a particle is absorbed after hitting a patch, its contribution is discarded, but we have traced the ray nevertheless. Better to use another counter: count particles visiting a patch, whether absorbed or not. Total number of visits at i: counter C_i. Of course, survival expectation is $C_i^S \approx \rho_i \, C_i$. Therefore:

Collision estimator of power

$$\frac{p_i}{p_{eT}} \approx \rho_i \, \frac{C_i}{N} \, .$$

Absorption estimation

This counter only counts particles absorbed at a patch: it measures energy left where the photons are absorbed. It is useful for dark scenes, where absorption is the most likely event. Total number of visits at i: counter C_i^A. Of course, $C_i = C_i^S + C_i^A$, and absorption expectations verify $C_i^A \approx (1 - \rho_i)\, C_i$. But $C_i^S \approx \rho_i\, C_i$, and so $C_i^A = (1 - \rho_i)\, \frac{C_i^S}{\rho_i}$. Hence:

Absorption estimator of power

$$\frac{p_i}{p_{eT}} \approx \frac{\rho_i}{1 - \rho_i}\, \frac{C_i^A}{N}\,.$$

Since not all particles are absorbed, some (the surviving ones) are discarded by this estimator, and this increases variance. Thus, the collision estimator is usually more efficient (less variance) than the absorption (and the survival) estimator. At a very reflective area, the survival estimator has lower variance (less noise) than the absorption one, and vice-versa at a dark or black area.

Absorption estimation

This counter only counts particles absorbed at a patch: it measures energy left where the photons are absorbed. It is useful for dark scenes, where absorption is the most likely event. Total number of visits at i: counter C_i^A. Of course, $C_i = C_i^S + C_i^A$, and absorption expectations verify $C_i^A \approx (1 - \rho_i)\, C_i$. But $C_i^S \approx \rho_i\, C_i$, and so $C_i^A = (1 - \rho_i)\, \frac{C_i^S}{\rho_i}$. Hence:

Absorption estimator of power

$$\frac{p_i}{p_{eT}} \approx \frac{\rho_i}{1 - \rho_i}\, \frac{C_i^A}{N}\,.$$

Since not all particles are absorbed, some (the surviving ones) are discarded by this estimator, and this increases variance. Thus, the collision estimator is usually more efficient (less variance) than the absorption (and the survival) estimator. At a very reflective area, the survival estimator has lower variance (less noise) than the absorption one, and vice-versa at a dark or black area.

Continuous random processes: photon density estimation

This is a continuous state space random process, that simulates randomly the photons' trajectories. Let us apply it to radiosity, that is, to diffusive scenes.

- Birth probability proportional to self-emitted radiosity: for diffusive patches, birth density at point x_0 is $S(x_0) = b_e(x_0)/p_{eT}$.
- Initial particle direction $\vec{\theta}_0$ with conditional density distribution proportional to the angular light emission distribution of the light source at x_0 times the cosine of the deviation from the normal direction $\vec{n}_{x_0}$: for a diffusive light source (isotropic), this transport term is just this cosine:

$$T(\vec{\theta}_0|x_0) = \frac{1}{\pi}\langle\vec{\theta}_0,\, \vec{n}_{x_0}\rangle\,.$$

- First hit at point x_1 with conditional probability density function

$$T(x_1|\, x_0,\, \vec{\theta}_0) = \frac{1}{\mathrm{dist}^2(x_0,\, x_1)}\langle -\vec{\theta}_0,\, \vec{n}_{x_1}\rangle\, V(x_0,\, x_1)$$

where V is the visibility factor.

Continuous random processes: photon density estimation

This is a continuous state space random process, that simulates randomly the photons' trajectories. Let us apply it to radiosity, that is, to diffusive scenes.

- Birth probability proportional to self-emitted radiosity: for diffusive patches, birth density at point x_0 is $S(x_0) = b_e(x_0)/p_{eT}$.
- Initial particle direction $\vec{\theta}_0$ with conditional density distribution proportional to the angular light emission distribution of the light source at x_0 times the cosine of the deviation from the normal direction $\vec{n}_{x_0}$: for a diffusive light source (isotropic), this transport term is just this cosine:

$$T(\vec{\theta}_0|x_0) = \frac{1}{\pi}\langle\vec{\theta}_0,\, \vec{n}_{x_0}\rangle\,.$$

- First hit at point x_1 with conditional probability density function

$$T(x_1|\,x_0,\,\vec{\theta}_0) = \frac{1}{\mathrm{dist}^2(x_0,\,x_1)}\langle-\vec{\theta}_0,\,\vec{n}_{x_1}\rangle\,V(x_0,\,x_1)$$

where V is the visibility factor.

Continuous random processes: photon density estimation

This is a continuous state space random process, that simulates randomly the photons' trajectories. Let us apply it to radiosity, that is, to diffusive scenes.

- Birth probability proportional to self-emitted radiosity: for diffusive patches, birth density at point x_0 is $S(x_0) = b_e(x_0)/p_{eT}$.
- Initial particle direction $\vec{\theta}_0$ with conditional density distribution proportional to the angular light emission distribution of the light source at x_0 times the cosine of the deviation from the normal direction $\vec{n}_{x_0}$: for a diffusive light source (isotropic), this transport term is just this cosine:

$$T(\vec{\theta}_0|x_0) = \frac{1}{\pi}\langle\vec{\theta}_0,\, \vec{n}_{x_0}\rangle\,.$$

- First hit at point x_1 with conditional probability density function

$$T(x_1|\, x_0,\, \vec{\theta}_0) = \frac{1}{\mathrm{dist}^2(x_0,\, x_1)}\langle -\vec{\theta}_0,\, \vec{n}_{x_1}\rangle\, V(x_0,\, x_1)$$

where V is the visibility factor.

- Combining conditional probabilities, we get the transition density $x_0 \to x_1$, that samples particles reflecting at x_1 coming from x_0 :

$$T(x_1|\, x_0) = \rho(x_1)\, T(\vec{\theta}_0|x_0)\, T(x_1|\, x_0,\, \vec{\theta}_0)$$

(this because x_1 is in a purely diffusive patch, otherwise we would replace the reflectance $\rho(x_1)$ with the *albedo* $\rho(x_1\,,\, -\vec{\theta}_0)$ that measures the fraction of the power incoming to x_1 from x_0 that gets reflected). Observe that $T(\vec{\theta}_0|x_0)\, T(x_1|\, x_0,\, \vec{\theta}_0) = T(x_1|\, x_0)/\rho(x_1)$ is precisely $G(x_0,\, x_1)\, V(x_0,\, x_1) = K(x_0,\, x_1)$. That is,

$$T(x_1|\, x_0) = \rho(x_1)\, K(x_0,\, x_1)\,. \tag{6}$$

- Continue this way, each time by sampling reflected particles (if survival is sampled): the reflected direction is chosen with distribution given by the BRDF times the outgoing cosine (the BRDF disappears for purely diffusive patches), and then performing the survival test. This way we get a transition density function $T(x|\, y)$ for any pair $(x,\, y)$ of points in the scene.

- Combining conditional probabilities, we get the transition density $x_0 \to x_1$, that samples particles reflecting at x_1 coming from x_0 :

$$T(x_1|\, x_0) = \rho(x_1)\, T(\vec{\theta}_0|x_0)\, T(x_1|\, x_0,\, \vec{\theta}_0)$$

(this because x_1 is in a purely diffusive patch, otherwise we would replace the reflectance $\rho(x_1)$ with the *albedo* $\rho(x_1\,,\, -\vec{\theta}_0)$ that measures the fraction of the power incoming to x_1 from x_0 that gets reflected). Observe that $T(\vec{\theta}_0|x_0)\, T(x_1|\, x_0,\, \vec{\theta}_0) = T(x_1|\, x_0)/\rho(x_1)$ is precisely $G(x_0,\, x_1)\, V(x_0,\, x_1) = K(x_0,\, x_1)$. That is,

$$T(x_1|\, x_0) = \rho(x_1)\, K(x_0,\, x_1)\,. \tag{6}$$

- Continue this way, each time by sampling reflected particles (if survival is sampled): the reflected direction is chosen with distribution given by the BRDF times the outgoing cosine (the BRDF disappears for purely diffusive patches), and then performing the survival test. This way we get a transition density function $T(x|\, y)$ for any pair $(x,\, y)$ of points in the scene.

- Obviously, in general the expected density $\chi(x)$ of particle hits at x satisfies the equation

$$\chi(x) = S(x) + \int \chi(y)\, T(x|\, y)\, d\sigma(y)$$

where the integral is taken over the union of all surfaces of the scene.

- But for a diffuse environment, by (6) this becomes

$$\chi(x) = \frac{b_e(x)}{p_{eT}} + \int \chi(y)\, K(y,\, x)\, \rho(x)\, d\sigma(y) \equiv \frac{b(x)}{p_{eT}} \ :$$

the density of particle hits is proportional to the radiosity!

- Obviously, in general the expected density $\chi(x)$ of particle hits at x satisfies the equation

$$\chi(x) = S(x) + \int \chi(y)\, T(x|\, y)\, d\sigma(y)$$

where the integral is taken over the union of all surfaces of the scene.

- But for a diffuse environment, by (6) this becomes

$$\chi(x) = \frac{b_e(x)}{p_{eT}} + \int \chi(y)\, K(y,\, x)\, \rho(x)\, d\sigma(y) \equiv \frac{b(x)}{p_{eT}} \ :$$

the density of particle hits is proportional to the radiosity!

- Obviously, in general the expected density $\chi(x)$ of particle hits at x satisfies the equation

$$\chi(x) = S(x) + \int \chi(y)\, T(x|\, y)\, d\sigma(y)$$

where the integral is taken over the union of all surfaces of the scene.

- But for a diffuse environment, by (6) this becomes

$$\chi(x) = \frac{b_e(x)}{p_{eT}} + \int \chi(y)\, K(y,\, x)\, \rho(x)\, d\sigma(y) \equiv \frac{b(x)}{p_{eT}} \ :$$

the density of particle hits is proportional to the radiosity!

Histogram method for evaluation of radiosity

The hit density is $\chi(x) = b(x)/p_{eT}$, and, with an N-particle random process, for each patch S_i, $\int_{S_i} \chi(x)\, d\sigma(x) \approx N_i/N$, where N_i is the number of hits on S_i. Remember that A_i is the area of S_i. Then, using patches as bins for counting, we can estimate the histogram of radiosity, that is the patch radiosities $b_i = \frac{1}{A_i} \int_{S_i} b(x)\, d\sigma(x)$, as follows:

$$b_i = \frac{1}{A_i} \int_{S_i} b(x)\, d\sigma(x) \approx \frac{p_{eT}}{A_i} \frac{N_i}{N}\,.$$

This method gives high variance (high level of noise) to small patches, unless many photons are traced.

Orthogonal basis expansion for evaluation of radiosity

This is a higher order approximation for patch radiosities: instead than considering them constant on each patch by taking an average, they are interpolated as polynomials in the two variables that parameterize the patch, or maybe by means of an expansion into suitable orthogonal functions. Every part of the computation remains as it was, but the integral over the patch is replaced by the set of weighted integrals where the weights are these polynomials. In this way, we can produce a smoother transition between adjoining patches: instead, the histogram method gives a sharp break, that is a jagged look, that usually forces to worsen the precision by transfering patch radiosities to the vertices of the patches and then applying bilinear (actually, bicubic) interpolation, a procedure called *Gouraud shading*. Gouraud's shading degrades the image: it can produce shadow leaks or light leaks, it is inaccurate for instance with sharp shadow boundaries, it can produce Mach bands.

Orthogonal basis expansion, as well as the next methods, kernel methods and nearest neighbor methods, attenuate jagging, but do not eliminate it altogether.

Orthogonal basis expansion for evaluation of radiosity

This is a higher order approximation for patch radiosities: instead than considering them constant on each patch by taking an average, they are interpolated as polynomials in the two variables that parameterize the patch, or maybe by means of an expansion into suitable orthogonal functions. Every part of the computation remains as it was, but the integral over the patch is replaced by the set of weighted integrals where the weights are these polynomials. In this way, we can produce a smoother transition between adjoining patches: instead, the histogram method gives a sharp break, that is a jagged look, that usually forces to worsen the precision by transfering patch radiosities to the vertices of the patches and then applying bilinear (actually, bicubic) interpolation, a procedure called *Gouraud shading*. Gouraud's shading degrades the image: it can produce shadow leaks or light leaks, it is inaccurate for instance with sharp shadow boundaries, it can produce Mach bands.

Orthogonal basis expansion, as well as the next methods, kernel methods and nearest neighbor methods, attenuate jagging, but do not eliminate it altogether.

Kernel methods for evaluation of radiosity

Consider the trivial identity $b(z) = \int b(x)\,\delta(x - z)\,d\sigma(z)$ and approximate the delta distribution with an approximation of the identity, that is a kernel $H(x, z)$ centered at x, non-negative, of integral 1 and small far from x: normally one chooses a bell shape, narrow and high.
This approximation and subsequent discretization of the integral as a sum $\longrightarrow$ the radiosity function of the patch $S_i \approx$ sum of translates of this kernel, centered at the N hit points x_j.

Estimator of the kernel method

$$b(z) = \frac{P_{eT}}{N} \sum_{j=1}^{N} H(x_j, z)\,.$$

If the kernel function H is a narrow bell, this method is good near the edges of patches, where the previous methods of histogram or function expansion of the integrals would yield bleeding of color from adjoining patches.

Kernel methods for evaluation of radiosity

Consider the trivial identity $b(z) = \int b(x)\,\delta(x - z)\,d\sigma(z)$ and approximate the delta distribution with an approximation of the identity, that is a kernel $H(x, z)$ centered at x, non-negative, of integral 1 and small far from x: normally one chooses a bell shape, narrow and high.
This approximation and subsequent discretization of the integral as a sum $\longrightarrow$ the radiosity function of the patch $S_i \approx$ sum of translates of this kernel, centered at the N hit points x_j.

Estimator of the kernel method

$$b(z) = \frac{P_{eT}}{N} \sum_{j=1}^{N} H(x_j, z)\,.$$

If the kernel function H is a narrow bell, this method is good near the edges of patches, where the previous methods of histogram or function expansion of the integrals would yield bleeding of color from adjoining patches.

Nearest neighbor estimation of radiosity

Instead than fixing a patch and counting the number N of hits there, one fixes N *neighboring particles* hitting a patch and looks for the smallest area A in the patch that contains the hit points. The ratio N/A is the hit density.

A great advantage is that this method does not depend on the mesh that models the scene, it does not even require a mesh. The disadvantage is that the estimator is more difficult to write, and that we need to store the hit point coordinates in such a database structure that allows us to recover quickly which stored points are neighbors, that is, are the closest to any chosen point, without checking distances serially, that is one by one consecutively. A structure suitable for this task is a tree of binary partitions of 3D-space called a *kd-tree*.

These ideas are central to a more advanced multi-pass method, the *photon mapping*, to be outlined later.

Nearest neighbor estimation of radiosity

Instead than fixing a patch and counting the number N of hits there, one fixes N *neighboring particles* hitting a patch and looks for the smallest area A in the patch that contains the hit points. The ratio N/A is the hit density.

A great advantage is that this method does not depend on the mesh that models the scene, it does not even require a mesh. The disadvantage is that the estimator is more difficult to write, and that we need to store the hit point coordinates in such a database structure that allows us to recover quickly which stored points are neighbors, that is, are the closest to any chosen point, without checking distances serially, that is one by one consecutively. A structure suitable for this task is a tree of binary partitions of 3D-space called a *kd-tree*.

These ideas are central to a more advanced multi-pass method, the *photon mapping*, to be outlined later.

Nearest neighbor estimation of radiosity

Instead than fixing a patch and counting the number N of hits there, one fixes N *neighboring particles* hitting a patch and looks for the smallest area A in the patch that contains the hit points. The ratio N/A is the hit density.

A great advantage is that this method does not depend on the mesh that models the scene, it does not even require a mesh. The disadvantage is that the estimator is more difficult to write, and that we need to store the hit point coordinates in such a database structure that allows us to recover quickly which stored points are neighbors, that is, are the closest to any chosen point, without checking distances serially, that is one by one consecutively. A structure suitable for this task is a tree of binary partitions of 3D-space called a *kd-tree*.

These ideas are central to a more advanced multi-pass method, the *photon mapping*, to be outlined later.

Final gathering

Hybrid methods mix a phase of radiance solution via the rendering equation with a second phase, for instance with direct ray tracing, or with selected types of light transport. We start this presentation with final gathering.

We have already noted that the complete radiosity solutions are jagged: each patch has sharp boundary edges. This problem is reduced by orthogonal basis expansion, by kernel methods and by nearest neighbor methods, but not completely suppressed.

Better refinement: a two-phase method, *final gathering*. Here we outline it for radiosity, but it works also with non-diffusive scenes if we use the appropriate BRDF's.

First phase: one of the previous methods provides a complete radiosity solution, but jagged and noisy. Second phase: the previous solution is regarded as a pre-computed radiance distribution $\tilde{L}(y)$ of the scene. If the scene is purely diffusive, this solution depends only on the point y that varies on the surfaces of the scene, but not on directions. This pre-computed distribution is now refined via ray tracing, as follows.

Final gathering

Hybrid methods mix a phase of radiance solution via the rendering equation with a second phase, for instance with direct ray tracing, or with selected types of light transport. We start this presentation with final gathering.

We have already noted that the complete radiosity solutions are jagged: each patch has sharp boundary edges. This problem is reduced by orthogonal basis expansion, by kernel methods and by nearest neighbor methods, but not completely suppressed.

Better refinement: a two-phase method, *final gathering*. Here we outline it for radiosity, but it works also with non-diffusive scenes if we use the appropriate BRDF's.

First phase: one of the previous methods provides a complete radiosity solution, but jagged and noisy. Second phase: the previous solution is regarded as a pre-computed radiance distribution $\tilde{L}(y)$ of the scene. If the scene is purely diffusive, this solution depends only on the point y that varies on the surfaces of the scene, but not on directions. This pre-computed distribution is now refined via ray tracing, as follows.

Final gathering

Hybrid methods mix a phase of radiance solution via the rendering equation with a second phase, for instance with direct ray tracing, or with selected types of light transport. We start this presentation with final gathering.

We have already noted that the complete radiosity solutions are jagged: each patch has sharp boundary edges. This problem is reduced by orthogonal basis expansion, by kernel methods and by nearest neighbor methods, but not completely suppressed.

Better refinement: a two-phase method, *final gathering*. Here we outline it for radiosity, but it works also with non-diffusive scenes if we use the appropriate BRDF's.

First phase: one of the previous methods provides a complete radiosity solution, but jagged and noisy. Second phase: the previous solution is regarded as a pre-computed radiance distribution $\tilde{L}(y)$ of the scene. If the scene is purely diffusive, this solution depends only on the point y that varies on the surfaces of the scene, but not on directions. This pre-computed distribution is now refined via ray tracing, as follows.

Remember Figure 1: the first step to draw the image at a given pixel is

$$L_{\text{pixel}} = \int_{\text{imageplane}} L(p \to \text{eye})\, h(p)\, dp$$

where $h(p)$ is a filter function for the pixel (maybe its characteristic function), and $L(p \to \text{eye}) = L(x \to \vec{\theta})$ with x the point of the scene visible through the center of the pixel and $\vec{\theta}$ the direction from x to the observer.

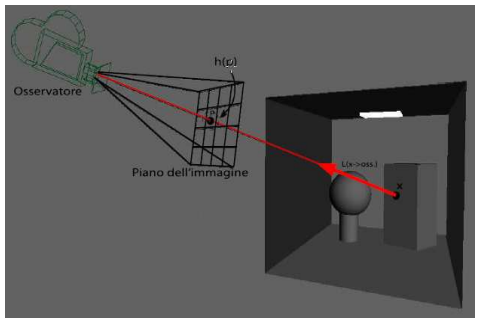


Figure : First step of path tracing

Then we computed $L(x \to \vec{\theta})$ by recurrence via the rendering equation. Instead, now at the right hand side of the rendering equation we make use of the pre-computed solution $\tilde{L}$, so no recurrence.

In area formulation

$$L(x \to \vec{\theta}) \equiv L(x) = L_e(x) + f_r(x) \int_{\text{scene}} \tilde{L}(y)\, G(x,y)\, V(x,y)\, d\sigma(y)\,,$$

and in hemispherical formulation

$$L(x) = L_e(x) + f_r(x) \int_{\Omega_x} \tilde{L}(r(x,\vec{\psi})\, \langle \vec{n}_x, \vec{\psi} \rangle\, d\vec{\psi}\,.$$

No recurrence: simply integrate by the Monte Carlo method.
If we sample $\vec{\psi}$ with uniform (cosine) distribution, or y uniformly over the scene, then huge variance, because we shall often miss the important light sources. Importance sampling must be used, privileging the areas or patches where $\tilde{L}$ is high, or the patches that are closer, or the y's where the geometric factor is big (but using all 3 together yields a multidimensional pdf and to produce samples with this pdf gives high variance).

General multi-pass methods

A multi-pass method combines various algorithms to produce the image, by separating different modes of light transport and applying different methods to each. It is important not to include the same mode of light transport in more than one pass, or if we do so then it is necessary to appropriately weight the contribution from each pass with weights that add up to 1.

Regular expressions for different light transport modes: The expressions are made up with letters that denote which parts of the scenes are hit by the transport:

- L one of the light sources
- D a diffuse reflection
- G a glossy or partially specular reflection
- S a perfectly specular reflection
- E the observer (camera position, or *eye*)

General multi-pass methods

A multi-pass method combines various algorithms to produce the image, by separating different modes of light transport and applying different methods to each. It is important not to include the same mode of light transport in more than one pass, or if we do so then it is necessary to appropriately weight the contribution from each pass with weights that add up to 1.

Regular expressions for different light transport modes: The expressions are made up with letters that denote which parts of the scenes are hit by the transport:

- L one of the light sources
- D a diffuse reflection
- G a glossy or partially specular reflection
- S a perfectly specular reflection
- E the observer (camera position, or *eye*)

All possible light paths are of type $L(D|G|S)^*E$, where $|$ means "or" and * means "repeatedly". Radiosity algorithms cover paths of type LD^*E; recursive ray tracing, $G(G|S)^*\tilde{D}E$, where $\tilde{D}$ means either 0 or 1 hit on a diffusive surface (where the path is stopped and the illumination from each light source is computed by means of a shadow ray (geometric + visibility factor)). Caustics are bright spots of light produced by specular reflections (lenses or concave mirrors). They are generated by photons that undergo one or more specular (including lenses!) or glossy reflections and then hit a diffusive surface: $L(G|S)^*D$. Really, most algorithms are multi-path, because they require an object pass to compute luminosity of the scene, followed by a visualization pass to lighten and color the pixels in the image plane, as the next examples shows.

All possible light paths are of type $L(D|G|S)^*E$, where $|$ means "or" and * means "repeatedly". Radiosity algorithms cover paths of type LD^*E; recursive ray tracing, $G(G|S)^*\tilde{D}E$, where $\tilde{D}$ means either 0 or 1 hit on a diffusive surface (where the path is stopped and the illumination from each light source is computed by means of a shadow ray (geometric + visibility factor)). Caustics are bright spots of light produced by specular reflections (lenses or concave mirrors). They are generated by photons that undergo one or more specular (including lenses!) or glossy reflections and then hit a diffusive surface: $L(G|S)^*D$. Really, most algorithms are multi-path,

because they require an object pass to compute luminosity of the scene, followed by a visualization pass to lighten and color the pixels in the image plane, as the next examples shows.

Examples

- Direct visualization: the light transport radiance solution stored in the first pass is accessed directly by tracing a ray through the pixel p and the result is attributed to p. This is the case for radiosity.
- Final gathering: for each x visible through p, the incoming radiance over Ω_x is not reconstructed recursively, but read from the solution stored at the first pass. In the case of a radiosity stored solution, its transport modes are LD^*. But the hemisphere reading at the second pass takes into account the BRDF at point x, that may not be diffusive. So, overall, the transport modes are $LD^*(D|G|S)E$. If a light source is directly visible through p (i.e., $x \in$ a lamp) then we should add LE.

Examples, continued

- Recursive stochastic ray tracing: the read-out pass consists of recursively tracing specular or partially specular reflections: transport modes $(G|S)^*E$. In classic recursive ray tracing, the first pass just stores shadow ray contribution, that is, direct illumination.
  To improve this, the first pass might store a radiosity solution, that is type LD^* (similar to assume that the shadow rays stored before were only diffusive (Lambert model) and not semi-glossy (Phong)). This yields precisely the final gathering method. Then the global modes for transports covered by this recursive ray tracing are $LD^*(G|S)^*E$. Here it is crucial that the same modes are not used in both passes. This is why we assumed that the first pass was a purely diffusive solution, that is radiosity. If we want to include semi-glossy reflections into the stored solution, the first pass becomes $L(D|G)^*$, but then the second pass must use only purely specular reflections, S^*E.

Sharing the same transport modes over multiple passes

Nevertheless, sometimes the same light transports are used in more than one pass. If so, different types of transports that appear in more than one pass should be weighted so that their overall contribution has total weight 100%, otherwise the estimator is biased. The choice of weight for a transport mode should optimize those passes that are best suited to that transport mode: this optimizes variance. For instance, caustics are better rendered in a bidirectional tracing pass (see later). *Warning:* this may be difficult if the two passes use different number of samples for the shared transport mode.
This allows to save time and reduce variance, and it is sometimes done automatically (and inaccurately) in most commercial renderers.

Bidirectional path tracing

Bidirectional path tracing makes use of the GRDF (Global Reflection Distribution Function) $G_r(x \to \vec{\theta},\, y \leftarrow \vec{\psi})$, that measure the contribution, to the flux $S \equiv S_p$ through the points visible through a pixel, of an elementary self-emitted light quantum (an element of a light source) $L_e(x \to \vec{\theta})$ at x in direction $\vec{\theta}$ and of a self-emitted importance quantum $W_e(y \leftarrow \vec{\psi})$ at y in direction $\vec{\psi}$ (that measures the importance of $(y, \vec{\psi})$ to the observer):

$$\Phi(S) = \int_{\text{scene}} dx \int_{\Omega_x} d\vec{\theta} \int_{\text{scene}} dy \int_{\Omega_y} d\vec{\psi}$$
$$L_e(x \to \vec{\theta})\, G_r(x \to \vec{\theta},\, y \leftarrow \vec{\psi})\, W_e(y \leftarrow \vec{\psi})\, \langle \vec{n}_x, \vec{\theta} \rangle\, \langle \vec{n}_y, \vec{\psi} \rangle\,.$$

Two different path generators are needed:

1. an eye-path $y_0, y_1, \ldots, y_k$ starting at y_0 visible through the pixel, with conditional probabilities for each next point proportional to the BRDF, and termination by Russian roulette absorption;
2. a light path $x_0, x_1, \ldots, x_l$ starting at x_0 belonging to some lamp and generated analogously.

The two path endpoints x_l and y_k are then joined by a straight segment, thereby giving rise to a unique path of length $k + l + 1$. An estimator for the flux using this single path is obtained by dividing by its probability density function:

$$\Phi(S) = \frac{\text{numerator}}{\text{pdf}(x_0, \ldots, x_l, y_k, \ldots, y_0)},$$

where the numerator is

$$L_e(x_0 \to x_0\vec{x}_1)\, G(x_0, x_1)\, V(x_0, x_1)\, f_r(x_1, x_1\vec{x}_0) \leftrightarrow x_0\vec{x}_2 \ldots$$
$$G(x_l, y_k)\, V(x_l, y_k)\, f_r(y_k, y_k\vec{x}_l) \leftrightarrow y_k\vec{y}_{k-1} \ldots$$
$$f_r(y_1, y_1\vec{y}_0) \leftrightarrow y_1\vec{y}_2)\, G(y_1, y_0)\, V(y_1, y_0)\, W_e(y_0 \leftarrow y_0\vec{y}_1).$$

Stochastic ray tracing is the particular case of this if we generate rays with $l = 0$ (thereby tracing rays only); light tracing is the particular case for $k = 0$ (light rays only). For instance, in stochastic ray tracing we generate aan eye path until absorption, and then compute its contribution by tracing a shadow ray towards the light sources. This shadow ray is actually a light ray of length $l = 0$.

The two path endpoints x_l and y_k are then joined by a straight segment, thereby giving rise to a unique path of length $k + l + 1$. An estimator for the flux using this single path is obtained by dividing by its probability density function:

$$\Phi(S) = \frac{\text{numerator}}{\text{pdf}(x_0, \ldots, x_l, y_k, \ldots, y_0)},$$

where the numerator is

$$\begin{aligned} L_e(x_0 \to x_0\vec{x}_1)\, G(x_0, x_1)\, V(x_0, x_1)\, f_r(x_1, x_1\vec{x}_0) \leftrightarrow x_0\vec{x}_2 \ldots \\ G(x_l, y_k)\, V(x_l, y_k)\, f_r(y_k, y_k\vec{x}_l) \leftrightarrow y_k\vec{y}_{k-1} \ldots \\ f_r(y_1, y_1\vec{y}_0) \leftrightarrow y_1\vec{y}_2)\, G(y_1, y_0)\, V(y_1, y_0)\, W_e(y_0 \leftarrow y_0\vec{y}_1)\,. \end{aligned}$$

Stochastic ray tracing is the particular case of this if we generate rays with $l = 0$ (thereby tracing rays only); light tracing is the particular case for $k = 0$ (light rays only). For instance, in stochastic ray tracing we generate aan eye path until absorption, and then compute its contribution by tracing a shadow ray towards the light sources. This shadow ray is actually a light ray of length $l = 0$.

Here we connected the endpoints of an eye path and a light path, but instead we could connect any vertex of the first to any vertex of the second. This allows us to sample new paths by reusing their subpaths: so we save time because we do not have to recompute their pdf (that includes tracing shadow rays, an expensive task!). The specular reflections in caustics are better dealt with by including them in the light path portion, because in the light paths the mirrors and lenses naturally focus the photons onto the caustics. Instead, if the same mirror is visible in the image, it is better to deal with the mirrored images by eye rays: indeed, these images are obtained by photons that have hit a diffusive part of the scene and bounce to the mirror, and so are naturally focused there via eye paths (ray tracing).

Here we connected the endpoints of an eye path and a light path, but instead we could connect any vertex of the first to any vertex of the second. This allows us to sample new paths by reusing their subpaths: so we save time because we do not have to recompute their pdf (that includes tracing shadow rays, an expensive task!). The specular reflections in caustics are better dealt with by including them in the light path portion, because in the light paths the mirrors and lenses naturally focus the photons onto the caustics. Instead, if the same mirror is visible in the image, it is better to deal with the mirrored images by eye rays: indeed, these images are obtained by photons that have hit a diffusive part of the scene and bounce to the mirror, and so are naturally focused there via eye paths (ray tracing).

Metropolis light transport: generalities

Here paths are sampled on the basis of the contribution they make to the final image. To each path we apply random mutations that give rise to a sequence of new paths. Each mutation is accepted or rejected according to how large is this contribution: so the method focuses onto important paths. For instance, if the scene models a room lightened by a small window only, or the interior of a pinhole camera, the important paths are hard to find, because they must pass through the window or the pinhole: but when the Metropolis algorithm finds one, then it generates small mutations around it and so has a good chance to generate many more of them. Moreover, the mutations reuse subpaths of the mutating path, and so, just as for bidirectional path tracing, we save time in computing the contributions of the mutated paths.

Metropolis sampling: outline

The Metropolis algorithm works this way: given a state space Ω and a non-negative function p of integral 1 on Ω, the Metropolis algorithm generates a random walk $\{x_n\}$ such that the probability distribution of x_n is eventually p, independently on x_0. This is done by accepting or rejecting mutations while generating the x_n's according to the following *detailed balance*. The probability that the mutation from x to y is accepted is $\alpha(x \to y)$ (to be determined). The probability of the mutation from x to y is known by the mutation technique adopted: call it $T(x \to y)$. To mantain equilibrium in the probability distribution of x or y, we must have

$$p(x)\, T(x \to y)\, \alpha(x \to y) = p(y)\, T(y \to x)\, \alpha(y \to x)\,.$$

Hence the best choice is

$$\alpha(x \to y) = \min\left\{1,\ \frac{p(y)\, T(y \to x)}{p(x)\, T(x \to y)}\right\}\,.$$

To apply this to rendering, we start with a finite number of paths generated by bidirectional path tracing and mutate, discarding paths with low contributions. Mutations are performed by deleting some subpath (this gives rise to two new internal ends), then continuing both these ends by attaching some new vertices and finally joining the new endpoints. We also perform perturbations, by moving one or more vertices of the paths. Small perturbation are useful, for instance, to slightly mutate light paths that reach caustics, so the mutated paths pass through a neighborhood of the caustic and explore its bright and dark areas.

Photon mapping

Photon mapping is a two pass algorithm that - like bidirectional ray tracing - traces both eye rays and light rays, but it stores as much as possible the illumination results of the first pass to reuse them. The first pass consists in tracing photons from light sources into the scene, with multiple bounces. The flux carried by these photons is stored in a data structure that allows to easily recover the n nearest hits at any point in the scene (a *kd-tree*). Through this data structure, based on space partitions and not on parametrization of the scene surface, the data storage is decoupled from surface parametrization, and so the method applies also to non-parameterizable, non-smooth scenes, as fractals for instance, and is impervious to meshing artifacts.

Photon mapping can be easily restricted to specific transport modes, as for instance the caustic transport mode, where it is particularly efficient (also because caustics are best dealt with light rays). Caustics are extremely slow to render and noisy through ray tracing, because of the enormous resolution needed to handle their extremely high variations of light intensity within tiny distances: the first pass of photon mapping could be limited to the caustic map, that is to transport modes of type LS^*D, where the method is perfectly suited (it was created for them). However, certain transport modes, in particular the caustics mode, need to be covered more than once (once together with all other modes at low sampling rate, and the other time by itself alone at high sampling rate), and so the photon mapping algorithm is biased. Therefore photon mapping is usually applied as a refinement method in addition to stochastic ray tracing (or other unbiased methods) to improve the rendering of indirect illumination.

Photon mapping, pass 1: tracing photons from light sources

In rendering, flux is usually computed as an integral, but in photon mapping it is computed as transported by a set of pointwise "photons", that are traced starting from light sources into the scene exactly as in stochastic ray (or light) tracing. Each hit on a non-specular point of the scene (not only at absorption) is recorded and stored (*mapped*) in the kd-tree, and contributes to the flux through that point. This lighting model can be interpreted as storing in non-specular surfaces the energy dropped by photons at each hit. Note that the photon map stores photons that travel via transport modes $L(S|G|D)^*(G|D)$, ending at a non-specular point. A particular case is caustics, which are dealt also with a separate map with many photons, the caustic map, that covers only the caustic transport mode LS^*D. This mode was already covered in the global photon map, but with a lower number of photons. Because light tracing pushes photons into caustics, the caustic map has low variance even if the number of its photons, although high, is much lower than what stochastic ray tracing would need for a similar accuracy.

Reflected radiance and photon density

The photon map at a point x represents incoming flux, so the photon density (flux divided by area) is the irradiance at x: to compute the reflected radiance we multiply this by the BRDF at x. For this we need to compute the smallest disc (or projected sphere) the contains the n closest photon hits. Once the radius ρ of this disc is recovered from the kd-tree, dividing by the area we obtain:

Reflected radiance through photon map

$$L(x \to \vec{\theta}) = \sum_{i=1}^{n} f_r(x, \vec{\theta} \leftrightarrow \vec{\theta}_i) \frac{\triangle \Phi_i(x \leftarrow \vec{\theta}_i)}{\pi \rho^2}$$

where $\triangle \Phi_i$ is the contribution to the flux provided by the i-th photon.

Photon mapping, pass 2: querying for the image

Direct visualization, considered in Example (1), could be used as pass 2, but of course it would produce some blurring (because we would average radiance over each pixel). Instead, once a ray is traced through each pixel to the first hit point x, similarly to final gathering, we now compute direct illumination at x as usual, by a ray tracer, and then we query the previously stored photon map for indirect illumination at x: this means that indirect illumination is probed by eye-paths that have undergone one diffuse or glossy reflection (however, with possibly multiple bounces, not only one as in final gathering). Note that these paths, considered in reverse, are terminal segments of the light paths of the global photon map in the first pass: this is another reason for bias.

But the actual algorithm for pass 1 is slightly more articulated: it factors through different types of hit points after the first glossy or diffuse bounce at point x. Here are the details:

Photon mapping, pass 2: querying for the image

Direct visualization, considered in Example (1), could be used as pass 2, but of course it would produce some blurring (because we would average radiance over each pixel). Instead, once a ray is traced through each pixel to the first hit point x, similarly to final gathering, we now compute direct illumination at x as usual, by a ray tracer, and then we query the previously stored photon map for indirect illumination at x: this means that indirect illumination is probed by eye-paths that have undergone one diffuse or glossy reflection (however, with possibly multiple bounces, not only one as in final gathering). Note that these paths, considered in reverse, are terminal segments of the light paths of the global photon map in the first pass: this is another reason for bias.
But the actual algorithm for pass 1 is slightly more articulated: it factors through different types of hit points after the first glossy or diffuse bounce at point x. Here are the details:

- one part of the contribution is direct illumination, dealt with by Monte Carlo sampling of the rendering integral limited to sampling those rays that are bouncing towards light sources, as seen before;
- if x produces a specular reflection or refraction, the bouncing ray is ray traced, as in the basic, non-recursive ray tracing;
- in the frontal hemisphere at x we know from pass 1 the directions of caustics: they are now queried from the caustic photon map using many samples to ensure high resolution;
- the remaining indirect illumination is obtain by sampling the frontal hemisphere to query the global photon map, hence the global rendering solution stored in pass 1: note that this query involves a complete solution, the global map, that already incorporates multiple bouncing of light, and therefore enhances the level of accuracy of the rendering of indirect illumination.

A physical model for participating media: volume emittance

Radiance is preserved along paths only in a vacuum: if there are atmospheric or other participating media, radiance can be diffused or emitted, absorbed, in-scattered or out-scattered, and in general is not preserved form beginning to end of the propagation.

Volume radiance:

$$L^v(x, \vec{\theta}) := L(x, \vec{\theta})\, \sigma_t(x)$$

where L is surface radiance and $\sigma_t(x)$ the probability that a photon hits the medium within unit distance of its travel starting at x (*extinction coefficient*).

Volume emission

The volume emittance function $e(z)$ is a function of the three space variables that measures the number of photons emitted at point z per unit time. Typical instance: flames.
Let $z_s = x + s\,\vec{\theta}$ be the straight segment from x to, say, y (here $\vec{\theta} = y - x$). Then the increment of the radiance due to volume emission in this segment is

$$L^e(y) - L^e(x) = \int_0^1 \frac{e(z_s)}{4\pi}\, ds$$

(4π at the denominator is the solid angle of the whole sphere, needed to transform the number $e(z)$ of emitted photons into an angular density). In other words,

$$\frac{\partial L^e}{\partial \vec{\theta}}(z) = \frac{e(z)}{4\pi}\,.$$

Absorption

Photons might be absorbed (that is, die) during their straight travel because of collisions with the participating medium.

Absorption coefficient $\sigma_a(x)$:= probability that a photon is absorbed per unit distance traveled starting at x.

It depends on the medium's density, and so it is not constant. Consider again photons traveling from x to y and, as in the previous page, denote this segment by $z_s = x + s\,\vec{\theta}$ $(0 \leqslant s \leqslant 1)$. Then the rate of decay of the photons by absorption is proportional to the number of photons traveling (i.e., to $L(x \to \vec{\theta})$), and the constant of proportionality is $-\sigma_a$:

$$\frac{\partial L(x \to \vec{\theta})}{\partial \vec{\theta}}(z) = -\sigma_a(x)\, L(x \to \vec{\theta})\,,$$

or equivalently,

$$\frac{dL(z_s \to \vec{\theta})}{ds} = -\sigma_a(z_s)\, L(z_s \to \vec{\theta})\,.$$

Thus, if σ_a is constant from x to y,

$$L(z_s \to \vec{\theta}) = L(x \to \vec{\theta})\, e^{-\sigma_a s}\,,$$

Out-scattering

Instead of being absorbed and dying, photons might be deflected away from their path during collisions with the media.
Mathematical description: same as absorption, but $\sigma_a(z) \longrightarrow \sigma_s(z)$: probability of scattering per unit distance.
Then, collect together absorption and out-scattering:

- *extinction coefficient* $\sigma_t = \sigma_a + \sigma_s$: probability of collision per unit distance;
- *decay coefficient at* $z = z_s$:
  $\tau(x, z) = \exp\left(-\int_0^{\mathrm{dist}(x,z)} \sigma_t(x + t\vec{\theta})\, dt\right)$;
- *reduced radiance* $L^r(z \to \vec{\theta}) = \tau(x, z)\, L(x \to \vec{\theta})$.

Albedo

$\alpha(z) = \sigma_s(z)/\sigma_t(z)$: probability that collision with media at z gives rise to scattering instead of absorption (the volume analogue of reflectance)

In-scattering

Conversely, photon scattered al z from other direction may continue in direction $\vec{\theta}$: *volume density of in-scattering* $L^{vi}(z \to \vec{\theta})$.

In-scattering radiance $L^{vi}(z \to \vec{\theta})$: properties

- it occurs if scattering occurs at z: $\sigma_s(z) \neq 0$;
- it is proportional to the volume irradiance $L^v(z \leftarrow \vec{\psi})$ (reduced photon density incoming at z from directions $\vec{\psi}$ multiplied with collision probability $\sigma_s(z)$):

$$L^v(z \leftarrow \vec{\psi}) := \sigma_t(z)\, L(z \leftarrow \vec{\psi})\,;$$

- it is proportional to scattering probability $p(z, \vec{\psi} \leftrightarrow \vec{\theta})$ from direction $\vec{\psi}$ to direction $\vec{\theta}$ (generally symmetric);
- it is proportional to the albedo $\alpha(z)$.

The expression for in-scattering radiance $L^{vi}(z \to \vec{\theta})$

Summarizing:

$$L^{vi}(z \to \vec{\theta}) = \int \alpha(z)\, p(z, \vec{\psi} \leftrightarrow \vec{\theta})\, L^{v}(z \leftarrow \vec{\psi})\, d\vec{\psi}$$

$$= \int \sigma_s(z)\, p(z, \vec{\psi} \leftrightarrow \vec{\theta})\, L(z \leftarrow \vec{\psi})\, d\vec{\psi}\,.$$

Typical phase functions $p(z, \vec{\psi} \leftrightarrow \vec{\theta})$

- Purely diffuse $\Rightarrow$ $p(z) = 1/4\pi$.
- Henyey–Greenstein:

$$p(z, \vec{\psi} \leftrightarrow \vec{\theta}) = \frac{1}{4\pi} \frac{1 - g^2}{1 + g^2 - 2g \sqrt[\frac{3}{2}]{\langle \vec{\psi}, \vec{\theta} \rangle}} .$$

  The parameter g is the average scattering cosine $\langle \vec{\psi}, \vec{\theta} \rangle$, and measure scattering anisotropy: $g > 0 \Rightarrow$ mainly forwards, $g < 0 \Rightarrow$ mainly backwards.
- Raleygh: diffusion of light in the sky.

Augmentation and reduction of radiance along straight paths

Suppose $x \underset{\vec{\theta}}{\to} y$: then how much of the radiance $L(x \to \vec{\theta})$ reaches y?

- Part of the radiance is reduced by absorption or out-scattering: $L^r(y \leftarrow -\vec{\theta}) = L(x \to \vec{\theta})\,\tau(x, y)$.
- New radiance is created during travel by emission or in-scattering: $L^+(z \to \vec{\theta}) = \frac{e(z)}{4\pi} + L^{vi}(z \to \vec{\theta})$.
- Also the radiance entering the path at $z_t = x + t\,\vec{\theta}$ by in-scattering is subject to reduction: so,

$$L(y \leftarrow -\vec{\theta}) = L(x \to \vec{\theta})\,\tau(x, y) + \int_0^{\mathrm{dist}(x,y)} L^+(z_t \to \vec{\theta})\,\tau(z_t, y)\,dt\,.$$

Rendering equation for participating media

Now the recursive rendering equation becomes:

Rendering equation, hemisphere formulation

With $y_{\vec{\psi}}$ = point of scene facing x in direction $\vec{\psi}$,

$$L(x \to \vec{\theta}) = L_e(x \to \vec{\theta}) + \int_{\Omega_x} L(y_{\vec{\psi}} \to -\vec{\psi})\, \tau(x, y)\, f_r(x, \vec{\theta} \leftrightarrow \vec{\psi})\, \langle \vec{\psi}, \vec{n}_x \rangle \, d\vec{\psi} + \int_{\Omega_x} \left(\int_0^{\mathrm{dist}(x,y)} L^+(z_t \to \vec{\theta})\, \tau(z_t, y)\, dt \right) f_r(x, \vec{\theta} \leftrightarrow \vec{\psi})\, \langle \vec{\psi}, \vec{n}_x \rangle \, d\vec{\psi}\,.$$

Rendering equation, area & volume formulation

With $\vec{\psi}_y = \vec{xy}$ direction from x to y,

$$L(x \to \vec{\theta}) = L_e(x \to \vec{\theta})$$

$$+ \int_{\text{surfaces}} f_r(x, \vec{\theta} \leftrightarrow$$

$$vecxy)\, L(y \to \vec{yx})\, \tau(x,y)\, V(x,y)\, \frac{\langle \vec{xy}, \vec{n}_x \rangle \langle \vec{yx}, \vec{n}_y \rangle}{\pi\ \mathrm{dist}^2(x,y)}\, \sigma(y)$$

$$+ \int_{\text{volume}} f_r(x, \vec{\theta} \leftrightarrow \vec{xy})\ L^+(z \to \vec{yx})\, \tau(z,y)\, V(x,z)\, \frac{\langle \vec{xy}, \vec{n}_x \rangle}{\pi\ \mathrm{dist}^2(x,y)}\, d\,\mathrm{vol}(z)\,,$$

where the integrals are over all the surfaces of the scene and all the volume of the scene, respectively, and $\mathrm{dist}^2(x,z)\, ds\, d\vec{\psi} = G(x,z)\, d\,\mathrm{vol}(z)$.

Overview of numerical algorithms for participating media

- Bidirectional path tracing: accurate but too costly in optically thick media (many collisions).
- Volume photon density estimation: excellent for volume caustics, accurate, but again too costly in optically thick media.
- Diffusion processes: suitable for optically thick media.

Here are the details:

Bidirectional path tracing for participating media

- For light ray, we sample starting point by either surface or volume emission, importance sampling based on relative amount of self-emitted power.
- In case of volume emission, we sample starting point by importance sampling based on $e(z)$: bright volumes are preferred as sources.
- Direction: isotropic distribution (usually so with flames).
- Sample location of next collision: either collision with medium or with next surface. Sample the distance along path: draw an equidistributed random number $\eta \in [0, 1)$, find by Monte Carlo integration (or analytically) r such that

$$\tau(x, z_r) = \exp\left(-\int_0^r \sigma_t(x + t\vec{\theta})\, dt\right) = 1 - \eta$$

(that is, $\tau(x, z)$ is the pdf for sampling r). Then, if $r <$ first hit point $\Rightarrow$ collision with media at distance r, else surface hit.

Bidirectional path tracing, continued

- In case of collision, sample scattering or absorption proportionally to the albedo $\alpha(z)$.
- In case of scattering, sample outgoing direction based on phase function $p(z, \vec{\theta} \leftrightarrow \vec{\psi})$
- In case of surface hit, sample outgoing direction based as usual, based on $f_r(z, \vec{\theta} \leftrightarrow \vec{\psi}) \langle \vec{n}_x, \vec{\psi} \rangle / \rho(z)$
- Connect the terminal vertices of the eye path and light path by a connecting line. The contribution of this line, with no medium, used to be $V(x, y) \langle \vec{n}_x, \vec{\theta} \rangle \langle \vec{n}_y, -\vec{\theta} \rangle$: with a medium, each inner product is replaced by 1 in case the corresponding endpoint is a volume collision instead of a surface hit.

Volume photon density estimation

Here the only variation from before is the need to compute volume density of photons colliding with medium. All methods work: hystogram, for instance, uses volume bins instead of surface bins. A new photon map, the volume map, is generated for this, and uses its kd-tree (being based on space partitions, a kd-tree is perfect for volumes). For the volume map, photon density is recovered by finding the smallest *sphere*, not disc, that contains the n closest records. The second pass of the photon mapping algorithm requires querying pre-computed and stored radiance. Without a medium, radiance is preserved along rays while reading; now we need to integrate along path (small adaptive steps) summing up all volume contributions and decay before the next surface hit.

Volume photon density estimation

Here the only variation from before is the need to compute volume density of photons colliding with medium. All methods work: hystogram, for instance, uses volume bins instead of surface bins. A new photon map, the volume map, is generated for this, and uses its kd-tree (being based on space partitions, a kd-tree is perfect for volumes). For the volume map, photon density is recovered by finding the smallest *sphere*, not disc, that contains the n closest records. The second pass of the photon mapping algorithm requires querying pre-computed and stored radiance. Without a medium, radiance is preserved along rays while reading; now we need to integrate along path (small adaptive steps) summing up all volume contributions and decay before the next surface hit.

Volume light diffusion

In highly scattering media, with a high collision density and long trajectories (for instance, clouds), the previous methods are too costly. Here it is better to statistically handle the many discrete collisions as a continuous diffusion, almost isotropic due to the averaging produced by multiple scattering.

Define *direct radiance* L_r as the radiance that reaches point x directly from a light source or the boundary of the scene, and *diffuse radiance* L_d as the radiance that bounced at least once. With many collisions, the latter is almost isotropic. Just keep a small first order directional term: model it as

$$L_d(x \to \vec{\theta}) = U^d(x) + \frac{3}{4\pi} \langle \vec{F}^d(x), \vec{\theta} \rangle ,$$

with U^d the angular average

$$U^d(x) = \frac{1}{4\pi} \int_\Omega L_d(x \to \vec{\theta}) \, d\vec{\theta}$$

and $\vec{F}^d$ is the average *projected* radiance

$$\langle \vec{F}^d(x), \vec{\theta} \rangle = \int_\Omega L(x \to \vec{\psi}) \langle \vec{\psi}, \vec{\theta} \rangle \, d\vec{\psi} \, .$$

Then U^d satisfies a diffusion equation:

$$\triangle U^d(x) - 3\sigma_a \left(\sigma_a + \sigma_s(1-g)\right) U^d(x) = -Q(x)$$

where $Q(x)$ is a suitable driving term that can be computed explicitly from L_r, and g is the average scattering cosine of the phase function. The consant after the minus sign is denoted σ^2_{tr}.
In simple cases there is an analytic solution, otherwise we proceed by numerical PDE.

Sub-surface scattering

Translucent materials: photons hit their surface at x, penetrate inside, undergo multiple scattering and exit at y (or are absorbed). Some materials are optically thick and therefore only diffusion methods are viable. Here the rendering equation becomes

$$L(y \to \vec{\theta}) = \int_{\text{material surface}} \int_{\Omega_x^+} L(x \leftarrow \vec{\psi})\, S(x, \vec{\psi} \leftrightarrow y, \vec{\psi})\, d\vec{\psi}\, d\sigma(x)$$

and it needs the *Bidirectional Scattering-Surface Reflection Distribution Function* BSSRDF $S(x, \vec{\psi} \leftrightarrow y, \vec{\psi})$: light entering at x may exit at $y \neq x$.

- Hanrahan: analytic solution in a planar slab of homogeneous material, allowing only one scattering.
- Jensen: approximate solution of diffusion equation with unlimited bounces, for a homogeneous material that fills a half-space

Sub-surface scattering

Translucent materials: photons hit their surface at x, penetrate inside, undergo multiple scattering and exit at y (or are absorbed). Some materials are optically thick and therefore only diffusion methods are viable. Here the rendering equation becomes

$$L(y \to \vec{\theta}) = \int_{\text{material surface}} \int_{\Omega_x^+} L(x \leftarrow \vec{\psi})\, S(x, \vec{\psi} \leftrightarrow y, \vec{\psi})\, d\vec{\psi}\, d\sigma(x)$$

and it needs the *Bidirectional Scattering-Surface Reflection Distribution Function* BSSRDF $S(x, \vec{\psi} \leftrightarrow y, \vec{\psi})$: light entering at x may exit at $y \neq x$.

- Hanrahan: analytic solution in a planar slab of homogeneous material, allowing only one scattering.
- Jensen: approximate solution of diffusion equation with unlimited bounces, for a homogeneous material that fills a half-space

Jensen's approximate sub-surface scattering solution

The approximate BSSRDF is

$$S(x, \vec{\psi} \leftrightarrow y, \vec{\psi}) = \frac{1}{\pi} F_t(\eta, \vec{\theta})\, R_d(x, y)\, F_t(\eta, \vec{\psi}),$$

where F_t's are the Fresnel transmission terms incoming at x (direction $\vec{\theta}$) and outgoing at y (direction $\vec{\psi}$), η is the refraction index, and

$$R_d(x, y) = \frac{\alpha'}{4\pi} \left[z_r (1 + \sigma_{\mathrm{tr}} d_r) \frac{e^{-\sigma_{\mathrm{tr}} d_r}}{d_r^3} + z_v (1 + \sigma_{\mathrm{tr}} d_v) \frac{e^{-\sigma_{\mathrm{tr}} d_v}}{d_v^3} \right]$$

with $\sigma_s' = \sigma_s(1 - g)$, $\sigma_t' = \sigma_s' + \sigma_a$, $\alpha' = \sigma_s'/\sigma_t'$, z_r and z_v are two imaginary sources of light placed on the normal line of the entrance point x, respectively inside and outside the material, and d_r and d_v are their respective distance from the exit point y.

Notation

If $\mathrm{dist}(x, y) = r$ *we write* $R_d(x, y) \equiv R_d(r)$.

All these constants, including the position of the two imaginary sources, can be calculated from the physical properties of the material. A distance r is obtained by sampling with pdf $R_d(r)$ (that is, by normalizing R_d and sampling r to compute the expected value $\int r\, R_d(r)\, dr$): then y is chosen on the surface at distance r from x. Because of the refraction index, the diffuse reflectance for sub-surface scattering varies with depth at a different rate for different light wavelengths (i.e., colors).

References

- Ph. Dutré, Ph. Bekaert, K. Bala, *Advanced Global Illumination*, A. K. Peters, 2003
- H. W. Jensen, *Realistic Image Synthesis Using Photon Mapping*, A. K. Peters, 2001
- M. Pharr, G. Humphreys, *Physically Based Rendering: from Theory to Implementation*, Morgan Kaufmann (Elsevier), 2004
- J. D. Foley, A. van Dam, S. K. Feiner, J. F. Hughes, *Computer Graphics: principles and Practice*, Addison-Wesley, 1990
- F. X. Sillion, C. Puech, *Radiosity and Global Illumination*, Morgan Kaufmann, 1994
- M. Picardello, *Algoritmi e metodi matematici in computer graphics*, www.mat.uniroma2.it/ picard/SMC/didattica/materiali_did/Comp.Graph./Note_di_Computer_Grap (for related books see www.mat.uniroma2.it/ picard/SMC/didattica/materiali_did/home_materiali_STM.html).