

Monte Carlo Techniques

Prof. Dr. Stefan Gumhold
Chair of Computer Graphics and
Visualization, TU Dresden



Overview

Monte Carlo Techniques

- ◆ Monte Carlo integration
- ◆ Monte Carlo Global Illumination
- ◆ Importance Sampling



Monte Carlo Integration



Motivation

- To solve the measuring / rendering equation we need to solve multi-dimensional nested integrals
- Operator notation is extended with measurement operator $\mathbf{M}_{\lambda,j}^{\Omega \times A}$ and by specifying the integration domain as superscript to show dimensionality of integration:

$$\Phi_{\lambda,j} = \iint_A \iint_{\Omega_j^{\text{in}}(\mathbf{x})} W_{\lambda}^e(\mathbf{x}, \boldsymbol{\omega}^{\text{in}}) L_{\lambda}^{\text{in}}(\mathbf{x}, \boldsymbol{\omega}^{\text{in}}) \cos \theta^{\text{in}} d\Omega_j^{\text{in}} dA$$

measurement equation



$$\Phi_{\lambda,j} = \mathbf{M}_{\lambda,j}^{\Omega \times A} L_{\lambda}^{\text{in}}$$

$$L_{\lambda}^{\text{out}}(\mathbf{x}, \boldsymbol{\omega}^{\text{out}}) = L_{\lambda}^{\text{emit}}(\mathbf{x}, \boldsymbol{\omega}^{\text{out}}) + \iint_{\Omega^{\text{in}}} \rho(\mathbf{x}, \boldsymbol{\omega}^{\text{in}}, \boldsymbol{\omega}^{\text{out}}) L_{\lambda}^{\text{in}}(\mathbf{x}, \boldsymbol{\omega}^{\text{in}}) \cos \theta^{\text{in}} d\Omega^{\text{in}}$$

rendering equation



$$L_{\lambda}^{\text{out}} = L_{\lambda}^{\text{emit}} + \mathbf{R}_{\lambda}^{\Omega} L_{\lambda}^{\text{in}}$$

Motivation

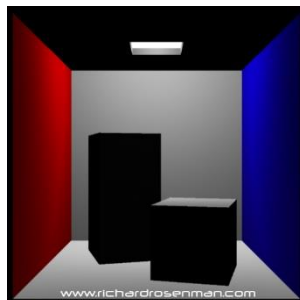
- Formal solution of rendering equation:

$$L_{\lambda}^{\text{out}} = (\mathbf{I} - \mathbf{R} \cdot \mathbf{T})^{-1} L_{\lambda}^{\text{emit}} \\ = L_{\lambda}^{\text{emit}} + \mathbf{RT} \cdot L_{\lambda}^{\text{emit}} + (\mathbf{RT})^2 \cdot L_{\lambda}^{\text{emit}} + \dots$$

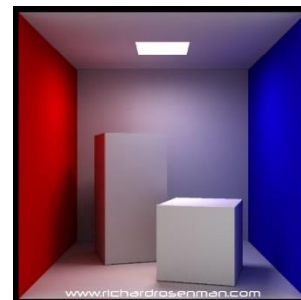
- can also be written as nested integrals and plugged into measuring equation with shortcut L_{λ}^e for $L_{\lambda}^{\text{emit}}$:

$$\Phi_{\lambda,j} = \mathbf{M}_{\lambda,j}^{\Omega \times A} \mathbf{T} \left(L_{\lambda}^e + \mathbf{R}_{\lambda}^{\Omega} \mathbf{T} \left(L_{\lambda}^e + \mathbf{R}_{\lambda}^{\Omega} \mathbf{T} \left(L_{\lambda}^e + \mathbf{R}_{\lambda}^{\Omega} \mathbf{T} (\dots) \right) \right) \right)$$

- This is a $2 \times 2 + 2 + 2 + 2 + \dots = \infty$ dimensional integral and it is important to evaluate nested integrals:



VS



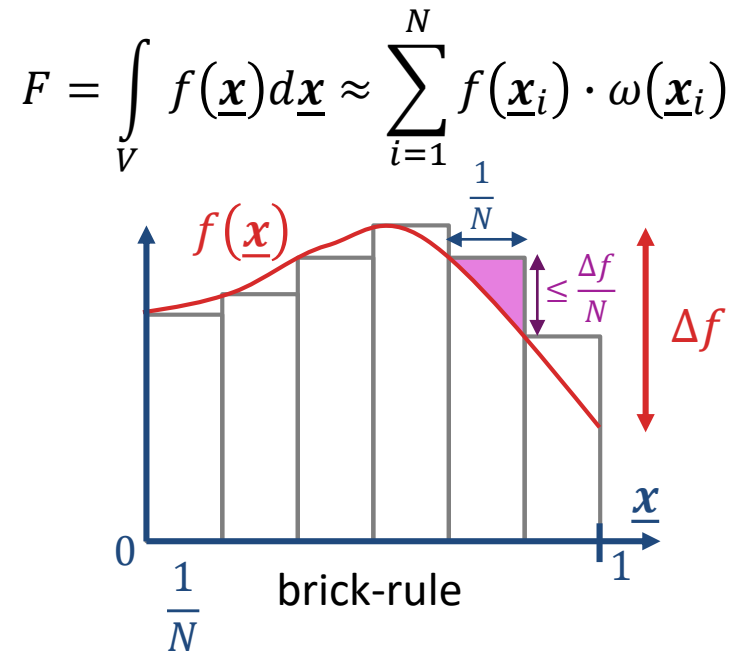
Classic Quadrature

- Numerical Integration is based on quadrature rules that weight function value samples $f(\underline{x}_i)$ with a weight $\omega(\underline{x}_i)$ that depends on the quadrature rule (i.e. brick-rule, [trapezoidal-rule](#), [Simpson-rule](#), ...)
- The integration error can be estimated from the **total variation** Δf of f in 1D to

$$\epsilon_1 = N \cdot \frac{1}{2} \frac{\Delta f}{N} \frac{1}{N} = \frac{\Delta f}{2N}$$

- in 2D we get with $n = \sqrt{N}$
- $$\epsilon_2 = n^2 \cdot \frac{1}{2} \frac{\Delta f}{n} \frac{1}{n^2} = \frac{\Delta f}{2\sqrt{N}}$$

- and in d dimensions
- $$\epsilon_d = n^d \cdot \frac{1}{2} \frac{\Delta f}{n} \frac{1}{n^d} = \frac{\Delta f}{2\sqrt[d]{N}}$$



- The required number of samples to get the error below ϵ is then $N_\epsilon \propto \left(\frac{\Delta f}{\epsilon}\right)^d$, i.e. exponential in d
- This phenomenon is called the **dimensional explosion** or the **curse of dimensionality**



Expectation Value

Monte Carlo integration exploits expected value problem

- given a random variable X distributed on domain Ω according to probability density $p(x)$ with

$$\int_{x \in \Omega} p(x) dx = 1$$

we denote the sampling of a specific value x from X as $x \sim X$.

- The expectation value $\bar{f}$ of function $f(x)$ at $x \sim X$ is defined as

$$\bar{f} = E[f] = \int_{x \in \Omega} f(x) p(x) dx$$

and depends on domain and probability density.



Expectation Value Estimator

- Let $x_1, x_2, \dots, x_N$ be random samples distributed according to $p(x)$ then an estimator $\hat{f}_N$ of $E[f(x)]$ is

$$E[f(x)] \approx \hat{f}_N = \frac{1}{N} \sum_{i=1}^N f(x_i)$$

- The estimator has the same expected value:

$$E[\hat{f}_N] = \frac{1}{N} \sum_{i=1}^N E[f(x_i)] = E[f(x)] = \bar{f}$$

Proof:

$$E\left[\frac{1}{N} \sum_i f(x_i)\right] = \int_{x_1 \in \Omega} \int_{x_2 \in \Omega} \dots \int_{x_N \in \Omega} \left(\frac{1}{N} \sum_i f(x_i)\right) p(x_N) \cdot \dots \cdot p(x_2) \cdot p(x_1) dx_N \dots dx_2 dx_1$$

- Integration of $f(x_N)$ over x_N yields $\bar{f}$. All other $f(x_{i < N})$ are preserved as they are constant in x_N .

$$E\left[\frac{1}{N} \sum_i f(x_i)\right] = \int_{x_1 \in \Omega} \dots \int_{x_{N-1} \in \Omega} \frac{1}{N} \left(\bar{f} + \sum_{i=1}^{N-1} f(x_i)\right) p(x_{N-1}) \cdot \dots \cdot p(x_1) dx_{N-1} \dots dx_1$$

- Integrating over other $x_{i < N}$ yields another $N - 1$ times $\bar{f}$

$$E\left[\frac{1}{N} \sum_i f(x_i)\right] = \bar{f}$$



- If the x_i are independent samples and $f(x)$ has the variance σ^2 , $\hat{f}_N = \frac{1}{N} \sum_{i=1}^N f(x_i)$ has variance

$$V[\hat{f}_N] = E[(\hat{f}_N - \bar{f})^2] = \frac{1}{N^2} \sum_{i=1}^N \sigma^2 = \frac{\sigma^2}{N}$$

- Independent of the dimension of integration, the standard deviation (measure for statistical error) of the Monte-Carlo estimator decreases with the square root of number of samples:

$$\hat{f}_N = \frac{1}{N} \sum_{i=1}^N f(\underline{x}_i), \quad E[\hat{f}_N] = \bar{f}, \quad \sigma[\hat{f}_N] = \frac{\sigma[f]}{\sqrt{N}}$$

$$V[\hat{f}_N] = E[(\hat{f}_N - \bar{f})^2] = E\left[\left(\frac{1}{N} \sum_{i=1}^N f(x_i) - \bar{f}\right)^2\right]$$

$$= E\left[\left(\frac{1}{N} \sum_{i=1}^N (f(x_i) - \bar{f})\right)^2\right] = \frac{1}{N^2} E\left[\left(\sum_{i=1}^N (f(x_i) - \bar{f})\right)^2\right]$$

$$= \frac{1}{N^2} \sum_{i,j=1}^N E[(f(x_i) - \bar{f})(f(x_j) - \bar{f})] = \frac{1}{N^2} \sum_{i=1}^N \sigma^2 = \frac{\sigma^2}{N}$$

$$E[(f(x_i) - \bar{f})(f(x_{j \neq i}) - \bar{f})] = \int_{x_1 \in \Omega} \cdots \int_{x_N \in \Omega} ((f(x_i) - \bar{f})(f(x_{j \neq i}) - \bar{f})) p(x_N) \cdots p(x_1) dx_N \cdots dx_1$$

$$= \int_{x_i} \int_{x_j} ((f(x_i) - \bar{f})(f(x_{j \neq i}) - \bar{f})) p(x_j) p(x_i) dx_j dx_i$$

$$E[(f(x_i) - \bar{f})(f(x_i) - \bar{f})] = \sigma^2$$

$$= \int_{x_i} (f(x_i) - \bar{f}) \underbrace{\int_{x_j} (f(x_{j \neq i}) - \bar{f}) p(x_j) dx_j}_{=0} \cdot p(x_i) dx_i$$



Monte Carlo Integration

- ◆ To compute the actual integral of the function $f(\underline{x})$ independent of the sampling distribution $p(\underline{x})$, one uses the following modification: $E \left[\frac{f(\underline{x})}{p(\underline{x})} \right] = \int_V f(\underline{x}) d\underline{x} = F$
- ◆ resulting in the Monte Carlo integration technique

$$\hat{f}_N = \frac{1}{N} \sum_{i=1}^N \frac{f(\underline{x}_i)}{p(\underline{x}_i)}, \quad E[\hat{F}_N] = F, \quad \sigma[\hat{F}_N] = \frac{\sigma[f/p]}{\sqrt{N}}$$

- ◆ For nested integrals along light transport paths often $N = 1$ is chosen with one path $\underline{x}_0$:

$$F = \int_V f(\underline{x}) d\underline{x} \approx \hat{F}_1 = \frac{f(\underline{x}_0)}{p(\underline{x}_0)}$$



Importance Sampling

- The choice of the sampling distribution $p(\underline{x})$ can significantly influence the variance of the estimate
- In case $p(\underline{x}) \propto f(\underline{x})$ the normalization constraint for $p(\underline{x})$ gives

$$p(\underline{x}) = c \cdot f(\underline{x}), \int p(\underline{x}) d\underline{x} = 1 \Rightarrow c = \frac{1}{F}$$

- Then the single sample estimator gives for any $\underline{x}_0$:

$$\frac{f(\underline{x}_0)}{p(\underline{x}_0)} \equiv F$$

resulting in no variance as $\sigma[f/p] = \sigma[F] = 0$.

- **Importance sampling** is the strategy to choose $p(\underline{x})$ as proportional to $f(\underline{x})$ as possible to reduce variance.



Summary

- ◆ The solution to the rendering equation is an infinite dimensional integral
- ◆ Numeric approximation of this integral with standard quadrature approaches suffers from the curse of dimensionality
- ◆ Monte Carlo techniques pose the numeric integration problem as an expected value estimation problem
- ◆ The resulting estimators are based on averaging estimates from samples drawn from a distribution $p(x)$
- ◆ Standard deviation (sqrt of variance) of the estimate corresponds to the approximation error and does not suffer from the curse of dimensionality
- ◆ A good choice of $p(x)$ can significantly reduce variance (importance sampling)

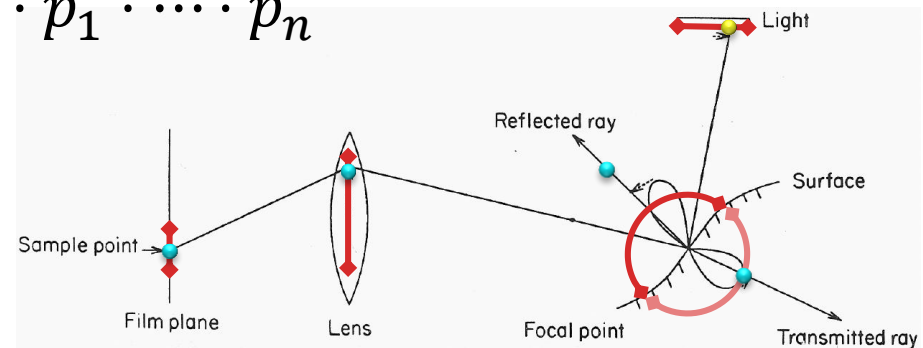


MONTE CARLO GLOBAL ILLUMINATION

Distribution Raytracing

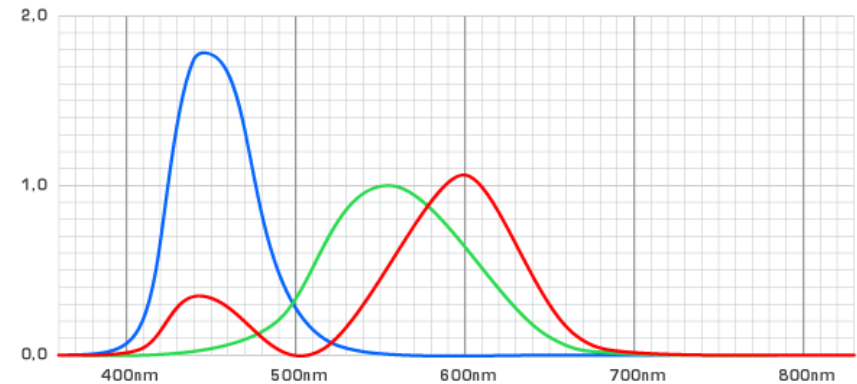
- The contributions to a pixel depend on **parameters** for light measurement and for the light path
- In distribution and Monte Carlo raytracing all parameters are **sampled independently** such that the probability densities can be multiplied
- typically the resulting pixel contribution from light transport yields product of individual terms such that a single path **Monte Carlo estimator** looks like

$$\hat{j} = \frac{f_0 \cdot f_1 \cdot \dots \cdot f_n}{p_0 \cdot p_1 \cdot \dots \cdot p_n}$$



Sampling the spectrum

- In spectral global illumination one samples the wavelength
- This can be done per color channel based on the spectral efficiency curves
- In the color space XYZ, the spectral efficiency curves $\bar{x}(\lambda)$, $\bar{y}(\lambda)$ and $\bar{z}(\lambda)$ correspond to probability distributions when computing the integrals



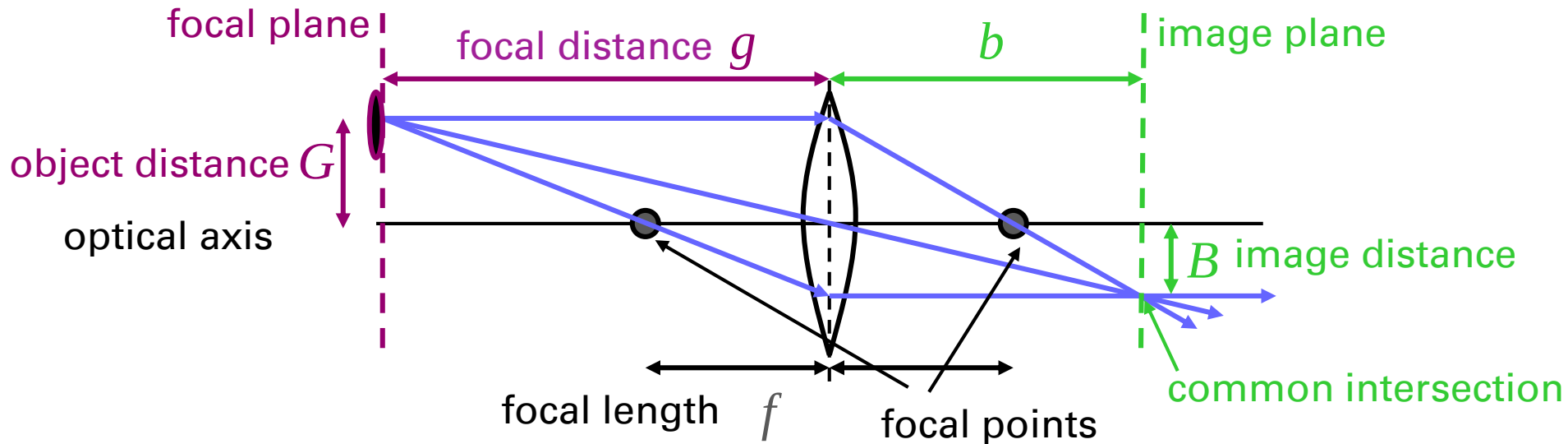
plots of $\bar{x}(\lambda)$, $\bar{y}(\lambda)$ and $\bar{z}(\lambda)$ from wikipedia

$$X_j = \int_{\lambda} \bar{x}(\lambda) \cdot \Phi_{\lambda,j} d\lambda \quad Y_j = \int_{\lambda} \bar{y}(\lambda) \cdot \Phi_{\lambda,j} d\lambda \quad Z_j = \int_{\lambda} \bar{z}(\lambda) \cdot \Phi_{\lambda,j} d\lambda$$

- To sample only once per computed color value, one can sample λ from photopic luminosity function: $p(\lambda) \propto x(\lambda) + y(\lambda) + z(\lambda)$
- resulting estimators: $\hat{X}_j = \frac{1}{N} \sum_{i=1}^N \frac{\bar{x}(\lambda_i)}{p(\lambda_i)} \cdot \hat{\Phi}_{\lambda_i,j}, \hat{Y}_j \dots$

Thin Lens

- thin lenses map all rays through points on the focal plane onto points on the image plane (common intersection)



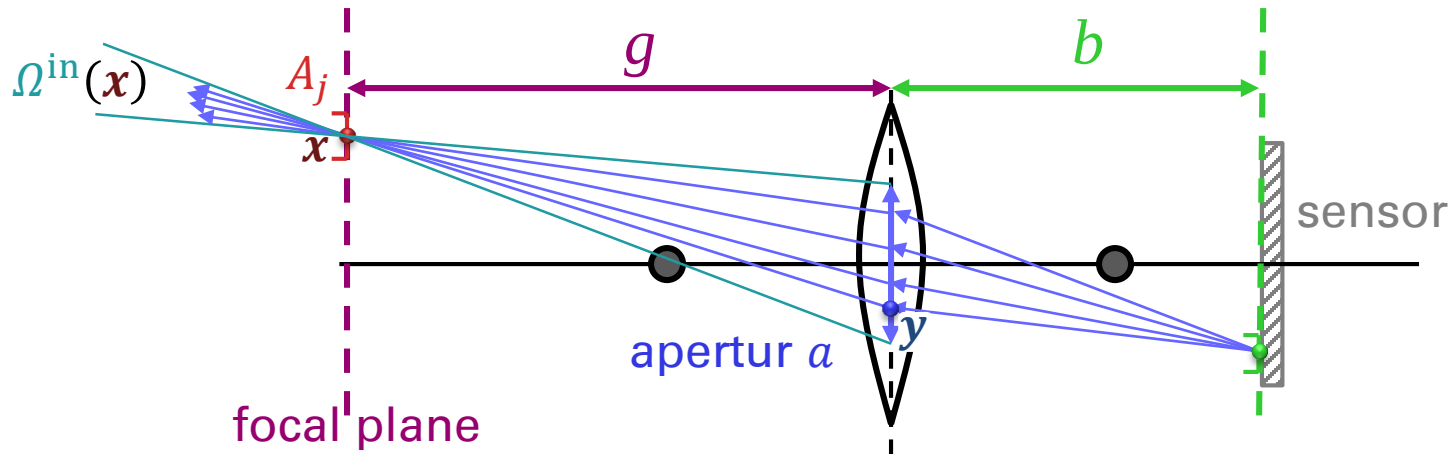
lense equation

$$\frac{1}{g} + \frac{1}{b} = \frac{1}{G} + \frac{1}{B} = \frac{1}{f}$$

$$\Phi_{\lambda,j} = \iint_A \iint_{\Omega_j^{\text{in}}(\mathbf{x})} W_{\lambda}^e(\mathbf{x}, \boldsymbol{\omega}^{\text{in}}) L_{\lambda}^{\text{in}}(\mathbf{x}, \boldsymbol{\omega}^{\text{in}}) \cos \theta^{\text{in}} d\Omega_j^{\text{in}} dA$$



Depth of Field from Measurement Eq



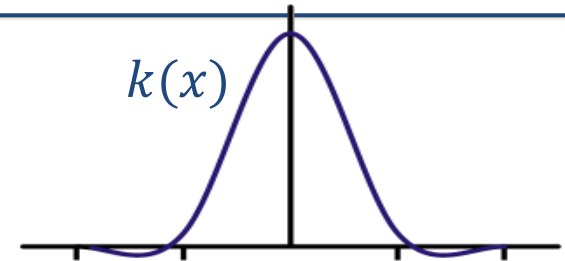
- Evaluation of the measurement equation supports depth of field simulation. For thin lense we map sensor to focal plane, where we need area A_j together with aperture a on lense (measured as area of circle)
- We can sample uniformly $\mathbf{x}$ over A_j and $\mathbf{y}$ over a and compute $\boldsymbol{\omega}^{\text{in}} = \widehat{\mathbf{x} - \mathbf{y}}$ and $d\Omega_j^{\text{in}} = \frac{\cos \theta^{\text{in}} da}{g^2}$ resulting in the estimator:

$$\hat{\Phi}_{\lambda,j} = \frac{1}{N} \sum_{i=1}^N \frac{W_{\lambda}^e \cos^2 \theta_i^{\text{in}}}{g^2 \frac{1}{A_j} \frac{1}{a}} \cdot \hat{L}_{\lambda}^{\text{in}}(\mathbf{x}_i, \boldsymbol{\omega}_i^{\text{in}}) = \frac{a}{g^2} A_j W_{\lambda}^e \cos^2 \theta_i^{\text{in}} \cdot \hat{L}_{\lambda}^{\text{in}}(\mathbf{x}_i, \boldsymbol{\omega}_i^{\text{in}})$$

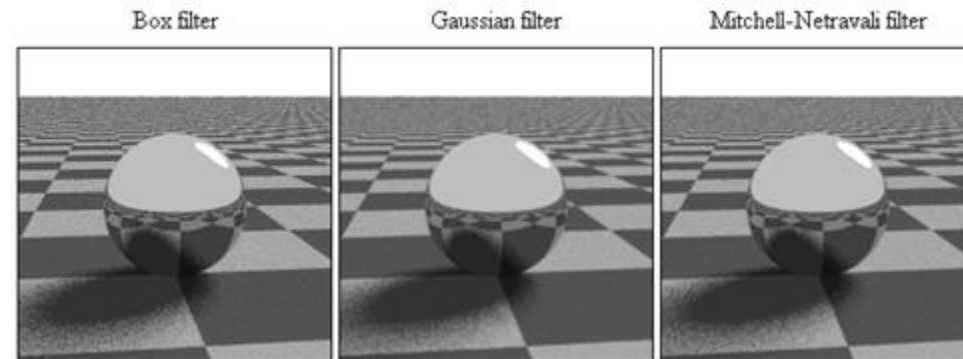
Spatial and Temporal Filtering

- uniform sampling of the pixel area corresponds to a box filter
- A better approximation to the theoretically optimal sinc-filter is the polynomial Mitchell-Netravalí-Filter $k(x)$
- This can be easily extended to 2D with the tensor product $k(x) \cdot k(y)$
- The convolution integral is solved by drawing samples according to

$$p(x, y) \propto |k(x)k(y)|$$
- Temporal filtering gives motion blurr and can be done in the same way for a time interval.



Mitchell-Netravalí-Filter



- ◆ To approximate the directional form

$$L_{\lambda}^{\text{reflect}}(\omega^{\text{out}}) = \iint_{\Omega^{\text{in}}} \rho(\omega^{\text{in}}, \omega^{\text{out}}) L_{\lambda}^{\text{in}}(\omega^{\text{in}}) \cos\theta^{\text{in}} d\Omega^{\text{in}}$$

one needs a sampling of the hemispherical directions ω^{in} according to some distribution $p(\omega)$.

- ◆ Given N samples ω_i^{in} the integral is approximated by

$$L_{\lambda}^{\text{reflect}}(\omega^{\text{out}}) \approx \frac{1}{N} \sum_i \frac{\rho(\omega_i^{\text{in}}, \omega^{\text{out}}) L_{\lambda}^{\text{in}}(\omega_i^{\text{in}}) \cos\theta_i^{\text{in}}}{p(\omega_i^{\text{in}})}$$

- ◆ Two questions arise here:

- ◆ To compute $L_{\lambda}^{\text{in}}(\omega_i^{\text{in}})$ we need a recursion, but when to terminate this?
- ◆ How to choose $p(\omega_i^{\text{in}})$ for efficient importance sampling?

Termination of Recursion through Russian Roulette



- Given a function $f(x)$ over V and an estimator $\hat{F}$ for the integral F of $f(x)$ over V :
 $E[\hat{F}] = F$
- The Russian Roulette estimator of $\widehat{RR}_{\hat{F}}(s)$ samples a binary random variable $b \in \{0,1\}$ with the success probability $p(1) = s$ and returns the estimate
$$\widehat{RR}_{\hat{F}}(s) = \begin{cases} \hat{F}/s & \text{if } b = 1 \\ 0 & \text{if } b = 0 \end{cases}$$
- The expectation value of $\widehat{RR}_{\hat{F}}(s)$ is the same and still F :
 $E[\widehat{RR}_{\hat{F}}(s)] = s \cdot E[\hat{F}/s] + (1 - s) \cdot 0 = E[\hat{F}]$
- The variance is increased, though.
- With the Russian Roulette estimator $\widehat{RR}_{\hat{L}_{\lambda}^{\text{reflect}}}(s)$ we can stochastically terminate the recursion in the rendering equation without changing the expectation value.
- A fixed termination depth would introduce **bias** to the global illumination approach.
- The success probability should be large as long as the influence on the current pixel is high.

Reference Implementation

primary rays – stratified sampling



```
void global_illumination(image& I, // to be computed image
                        const Scene& scene, const PinholeCamera& camera,
                        float ap_radius, float focal_distance,
                        float t, float dt, // current time and frame length
                        int N // number primary rays)
{
    for (int i=0; i<I.width(); ++i) {
        for (int j=0; j<I.height(); ++j) {
            // sample wavelengths and store sample probabilities
            std::vector<float> λ, p_λ; sample_spectral_sensitivity_stratified_and_shuffle(1, p_λ, N);
            // sample time values for motion blurr
            std::vector<float> t, p_t; sample_MN_filter_statified_and_shuffle(t, p_t, dt, N);
            // sample pixel for antialiasing
            std::vector<float> x, p_x; sample_MN_filter_statified_and_shuffle(x, p_x, dx, N);
            std::vector<float> y, p_y; sample_MN_filter_statified_and_shuffle(y, p_y, dy, N);
            // sample aperture for depth of field
            std::vector<P2D> a, p_a; sample_circle_statified_and_shuffle(a, p_a, sqrt(ap_radius/PI), N);
            // prepare color channel values
            float X = 0, Y = 0, Z = 0;
            // N-rook sampling of primary rays
            for (int k=0; k<N; ++k) {
                // select time value for motion blurr
                scene.select_time(t[k]);
                // construct ray from aperture sample through pixel sample
                Ray ray = camera.construct_ray(focal_distance, i, j, x[k], y[k], a[k]);
                // compute potential from sensor efficiency and sampling probs
                float potential = sensor_efficiency(λ[k], ray)*sqr(cos(ray.theta))/
                    (sqr(focal_distance)*p_x[k]*p_y[k]*p_a[k]*p_t[k]);
                // compute incoming radiance
                float L_in_λ = incoming_radiance(scene, ray, λ[k], potential);
                // add contributions to color channel integrators
                X += potential*L_in_λ*x_sensitivity(λ[k])/p_λ[k];
                Y += potential*L_in_λ*y_sensitivity(λ[k])/p_λ[k];
                Z += potential*L_in_λ*z_sensitivity(λ[k])/p_λ[k];
            }
            // normalize Monte Carlo estimates
            I.set_pixel_XYZ(i, j, X/N, Y/N, Z/N);
        }
    }
}
```

Reference Implementation

incoming and outgoing radiance



```
float incoming_radiance(const Scene& scene, const Ray& ray,
                        float  $\lambda$ , float potential)
{
    // trace ray and check for background case
    HitInfo info;
    if (!scene.trace(ray, info))
        return S.background_radiance(ray,  $\lambda$ );

    // otherwise transport outgoing radiance from hit point
    return outgoing_radiance(scene, info, -ray.omega,  $\lambda$ , potential);
}

float outgoing_radiance(const Scene& scene, const HitInfo& hit, V3D omega_out,
                        float  $\lambda$ , float potential)
{
    // combine emission and reflected radiance
    return info.L_emit(omega_out) +
        reflected_radiance(scene, info, omega_out,  $\lambda$ , potential);
}
```

Reference Implementation

reflected radiance



```
float reflected_radiance(const Scene& scene, const HitInfo& hit,
                        const V3D& omega_out,
                        float  $\lambda$ , float potential) {
    // configure Russian Roulette
    float s = choose_s(potential);
    // do Russian Roulette
    if (rand(0,1) > s) return 0.0f;
    // Monte Carlo estimator of L_reflect
    float L_reflect = 0;
    int N = choose_N(potential);
    for (int i=0; i<N; ++i) {
        // choose sample on hemisphere
        P2D omega_in;
        float p_omega_in;
        sample_hemisphere(info.get_normal(), omega_in, p_omega_in);
        // compute potential
        float weight = info.brdf(omega_in, omega_out) * cos(omega_in.theta);
        // compute contribution to integral by recursion and division through p_omega_in
        L_reflect += weight * incoming_radiance(S, Ray(info.x, omega_in),  $\lambda$ , weight*potential) /
            p_omega_in;
    }
    // normalize Monte Carlo estimates and account for Russian Roulette
    return L_reflect / N / s;
}
```



IMPORTANCE SAMPLING IN GLOBAL ILLUMINATION

Importance Sampling of the Reflection Integral

use **Nusselt's analogon** to sample based on cosine

$$\iint_{\Omega^{\text{in}}} \rho(\omega^{\text{in}}, \omega^{\text{out}}) L_{\lambda}^{\text{in}}(\omega^{\text{in}}) \cos \theta^{\text{in}} d\Omega^{\text{in}}$$

$\rho_{\text{diff}} + \rho_{\text{spec}} + \rho_{\text{mirr}}$
split BRDF into diffuse and specular / mirror reflection parts and adapt sampling

$L_{\lambda}^{\text{out}}(\mathbf{y}, -\omega^{\text{in}}), \quad \mathbf{y} = \text{trace}(\mathbf{x}, \omega^{\text{in}})$

$L_{\lambda}^{\text{emit}}(\mathbf{y}, -\omega^{\text{in}}) + L_{\lambda}^{\text{reflect}}(\mathbf{y}, -\omega^{\text{in}})$
split incoming radiance in direct and indirect illumination to support **direct light sampling**

Nusselt's Analogon (1928)

- Nusselt realized that the cosine weighted solid angle corresponds to its projection onto the unit disk located in tangential space of the surface
- Ignoring visibility this also holds for the area formulation, i.e. integration over a distant surface patch corresponds to the projection over unit sphere onto the unit disk.
- Importance sampling of the cosine term is achieved by uniformly sampling the unit disk ($p(x, y) = 1/\pi$) and projecting back onto the unit sphere:

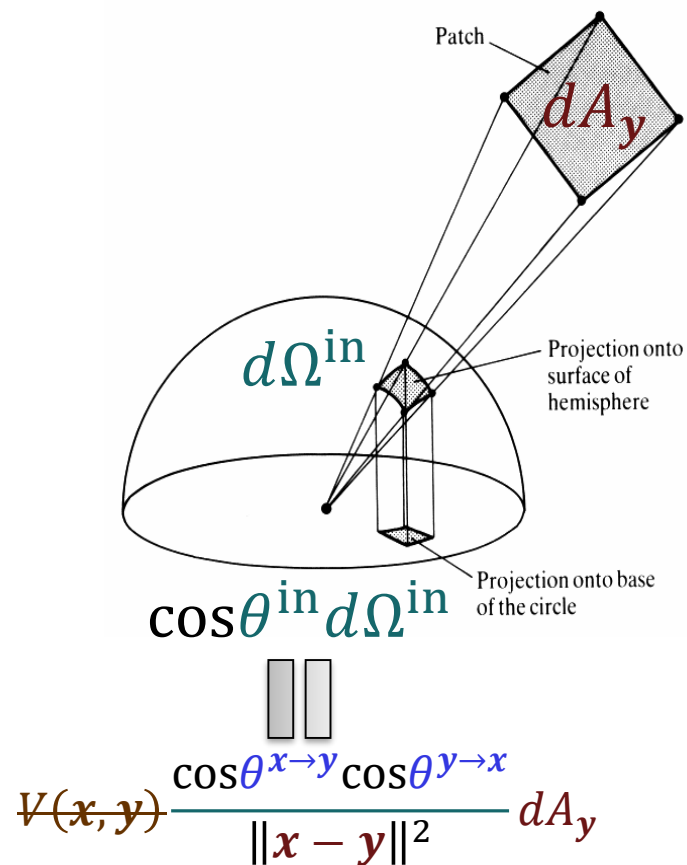
$$\hat{\omega} = (x, y, 1 - \sqrt{x^2 + y^2})$$

- As a result we sampled according to

$$p_{\text{Nusselt}}(\omega^{\text{in}}) = \cos\theta^{\text{in}}/\pi.$$



$$L_{\lambda, \text{Nusselt}}^{\text{reflect}}(\omega^{\text{out}}) \approx \frac{\pi}{N} \sum_i \rho(\omega_i^{\text{in}}, \omega^{\text{out}}) L_{\lambda}^{\text{in}}(\omega_i^{\text{in}})$$





Direct Light Sampling

- splits the reflection integral into direct & indirect part:

$$L_{\lambda}^{\text{reflect}}(\mathbf{x}, \boldsymbol{\omega}^{\text{out}}) = \iint_{\Omega^{\text{in}}} \rho(\mathbf{x}, \boldsymbol{\omega}^{\text{in}}, \boldsymbol{\omega}^{\text{out}}) L_{\lambda}^{\text{in}}(\mathbf{x}, \boldsymbol{\omega}^{\text{in}}) \cos \theta^{\text{in}} d\Omega^{\text{in}}$$

$$L_{\lambda}^{\text{reflect}}(\mathbf{x}, \boldsymbol{\omega}^{\text{out}}) = \iint_{\Omega^{\text{in}}} \rho(\mathbf{x}, \boldsymbol{\omega}^{\text{in}}, \boldsymbol{\omega}^{\text{out}}) [L_{\lambda}^{\text{emit}}(\mathbf{y}, -\boldsymbol{\omega}^{\text{in}}) + L_{\lambda}^{\text{reflect}}(\mathbf{y}, -\boldsymbol{\omega}^{\text{in}})] \cos \theta^{\text{in}} d\Omega^{\text{in}}$$

$$L_{\lambda}^{\text{reflect}}(\mathbf{x}, \boldsymbol{\omega}^{\text{out}}) = L_{\lambda}^{\text{direct}}(\mathbf{x}, \boldsymbol{\omega}^{\text{out}}) + L_{\lambda}^{\text{indirect}}(\mathbf{x}, \boldsymbol{\omega}^{\text{out}})$$

$$L_{\lambda}^{\text{direct}}(\mathbf{x}, \boldsymbol{\omega}^{\text{out}}) = \iint_{\Omega^{\text{in}}} \rho(\mathbf{x}, \boldsymbol{\omega}^{\text{in}}, \boldsymbol{\omega}^{\text{out}}) L_{\lambda}^{\text{emit}}(\text{trace}(\mathbf{x}, \boldsymbol{\omega}^{\text{in}}), -\boldsymbol{\omega}^{\text{in}}) \cos \theta^{\text{in}} d\Omega^{\text{in}}$$

$$L_{\lambda}^{\text{indirect}}(\mathbf{x}, \boldsymbol{\omega}^{\text{out}}) = \iint_{\Omega^{\text{in}}} \rho(\mathbf{x}, \boldsymbol{\omega}^{\text{in}}, \boldsymbol{\omega}^{\text{out}}) L_{\lambda}^{\text{reflect}}(\text{trace}(\mathbf{x}, \boldsymbol{\omega}^{\text{in}}), -\boldsymbol{\omega}^{\text{in}}) \cos \theta^{\text{in}} d\Omega^{\text{in}}$$

- Next one rewrites direct illumination in area formulation

$$L_{\lambda}^{\text{direct}}(\mathbf{x}, \boldsymbol{\omega}^{\text{out}}) = \iint_{\mathbf{y} \in A} \rho(\mathbf{x}, \boldsymbol{\omega}^{\mathbf{x} \rightarrow \mathbf{y}}, \boldsymbol{\omega}^{\text{out}}) L_{\lambda}^{\text{emit}}(\mathbf{y}, \boldsymbol{\omega}^{\mathbf{y} \rightarrow \mathbf{x}}) G(\mathbf{x}, \mathbf{y}) dA_{\mathbf{y}}$$



Direct Light Sampling

- Finally, one splits integral into sum over light sources and adds point and directional lights

$$\begin{aligned} L_{\lambda}^{\text{direct}}(\mathbf{x}, \boldsymbol{\omega}^{\text{out}}) = & \sum_{l=1}^{N_l^{\text{area}}} \iint_{\mathbf{y} \in A_l} \rho(\mathbf{x}, \boldsymbol{\omega}^{\mathbf{x} \rightarrow \mathbf{y}}, \boldsymbol{\omega}^{\text{out}}) L_{\lambda, l}^{\text{emit, area}}(\mathbf{y}, \boldsymbol{\omega}^{\mathbf{y} \rightarrow \mathbf{x}}) G(\mathbf{x}, \mathbf{y}) dA_{\mathbf{y}} \\ & + \sum_{l=1}^{N_l^{\text{pnt}}} \rho(\mathbf{x}, \boldsymbol{\omega}^{\mathbf{x} \rightarrow \mathbf{y}_l^{\text{pnt}}}, \boldsymbol{\omega}^{\text{out}}) I_{\lambda, l}^{\text{emit, pnt}}(\mathbf{y}_l^{\text{pnt}}, \boldsymbol{\omega}^{\mathbf{y}_l^{\text{pnt}} \rightarrow \mathbf{x}}) V(\mathbf{x}, \mathbf{y}_l^{\text{pnt}}) \frac{\cos \theta^{\mathbf{x} \rightarrow \mathbf{y}_l}}{\|\mathbf{x} - \mathbf{y}_l^{\text{pnt}}\|^2} \\ & + \sum_{l=1}^{N_l^{\text{dir}}} \rho(\mathbf{x}, \boldsymbol{\omega}_l^{\text{dir}}, \boldsymbol{\omega}^{\text{out}}) H_{\lambda, l}^{\text{emit, dir}}(-\boldsymbol{\omega}_l^{\text{dir}}) \cos \theta_l^{\text{dir}} V^{\text{dir}}(\mathbf{x}, \boldsymbol{\omega}_l^{\text{dir}}) \end{aligned}$$

- where point light sources emit spectral intensity which is spectral light power per solid angle
- directional lights must be written in the directional form. They emit power per area.
- $V^{\text{dir}}(\mathbf{x}, \boldsymbol{\omega})$ checks if the ray $(\mathbf{x}, \boldsymbol{\omega})$ hits the background

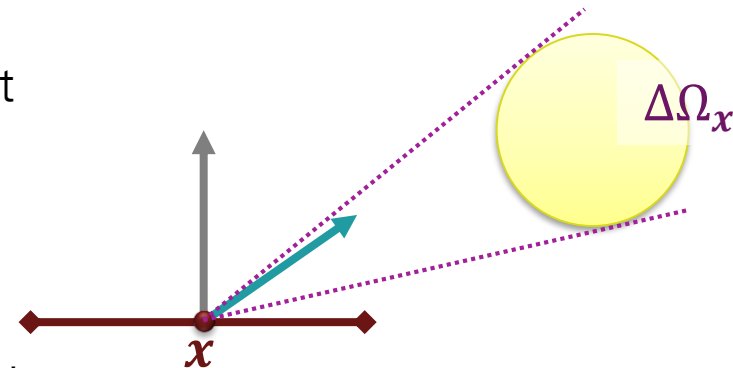
Direct Light Sampling of Area Lights



$$L_{\lambda,l}^{\text{direct,area}}(\mathbf{x}, \omega^{\text{out}}) = \iint_{\mathbf{y} \in A_l} \rho(\mathbf{x}, \omega^{\mathbf{x} \rightarrow \mathbf{y}}, \omega^{\text{out}}) L_{\lambda,l}^{\text{emit,area}}(\mathbf{y}, \omega^{\mathbf{y} \rightarrow \mathbf{x}}) V(\mathbf{x}, \mathbf{y}) \frac{\cos \theta^{\mathbf{x} \rightarrow \mathbf{y}} \cos \theta^{\mathbf{y} \rightarrow \mathbf{x}}}{\|\mathbf{x} - \mathbf{y}\|^2} dA_{\mathbf{y}}$$

- Per area light l we need to sample a number N_l of points $\mathbf{y}_i$ on the light source and estimate the direct light contribution
- Sampling of area light sources can have high variance if the visibility with respect to current point changes (i.e. spherical light source is half invisible to any scene point)
- Then it can help to reduce sampling to the light source part that is front facing with respect to current point (i.e. for a sphere this results in a circle that spans a direction cone)
- Nusselt's analog can be used to sample projected solid angle projected again on the unit disk, but projection is hard to do and hard to sample and we **still need visibility check**.

$$L_{\lambda,l}^{\text{direct,area}}(\mathbf{x}, \omega^{\text{out}}) \approx \frac{1}{N_l} \sum_{i=1}^{N_l} \frac{\rho(\mathbf{x}, \omega^{\mathbf{x} \rightarrow \mathbf{y}_i}, \omega^{\text{out}}) L_{\lambda,l}^{\text{emit,area}}(\mathbf{y}_i, \omega^{\mathbf{y}_i \rightarrow \mathbf{x}}) G(\mathbf{x}, \mathbf{y}_i)}{p(\mathbf{y}_i)}$$





Indirect Light Sampling

$$L_{\lambda}^{\text{indirect}}(\mathbf{x}, \boldsymbol{\omega}^{\text{out}}) = \iint_{\Omega^{\text{in}}} \rho(\mathbf{x}, \boldsymbol{\omega}^{\text{in}}, \boldsymbol{\omega}^{\text{out}}) L_{\lambda}^{\text{reflect}}(\text{trace}(\mathbf{x}, \boldsymbol{\omega}^{\text{in}}), -\boldsymbol{\omega}^{\text{in}}) \cos\theta^{\text{in}} d\Omega^{\text{in}}$$

- Indirect light typically comes from all directions and the integral should be sampled according to cosine term alone or cosine term and BRDF together.

```
float reflected_radiance(const Scene& scene, const HitInfo& hit,
                        const V3D& omega_out,
                        float λ, float potential) {
    // split integral into direct and indirect parts
    return direct_reflected_radiance(scene, hit, omega_out, λ, potential) +
           indirect_reflected_radiance(scene, hit, omega_out, λ, potential);
}

float indirect_reflected_radiance(const Scene& scene, const HitInfo& hit,
                                  const V3D& omega_out,
                                  float λ, float potential) {
    // same implementation as old implementation of reflected_radiance
    :
}
```

Refined Implementation for Direct Light Sampling



```
float direct_reflected_radiance(const Scene& scene, const HitInfo& hit,
                                const V3D& omega_out,
                                float  $\lambda$ , float potential) {
    // iterate all light sources
    float L_reflect = 0;
    for (int l=0; l<scene.get_nr_area_lights(); ++l) {
        float L_reflect_l = 0;
        // generate several shadow rays
        int N_l = scene.get_area_light(l).estimate_nr_shadow_rays(hit, potential);
        for (int i=0; i<N_l; ++i) {
            // choose sample on area light source and store p(y) in p_y
            P3D y; float p_y;
            y = scene.get_area_light(l).sample(hit, p_y);
            // compute visibility term
            if (scene.is_visible(hit.x(), y)) {
                V3D omega_in = (y-hit.x()).normalize();
                // compute contribution to integral without recursion in area formulation
                L_reflect_l += hit.brdf(omega_in, omega_out) * cos(omega_in.theta) *
                    dot(-omega_in, scene.get_area_light(l).get_normal(y)) *
                    scene.get_area_light(l).emitted_radiance(y, -omega_in) /
                    ((y-hit.x()).sqr_length() * p_y);
            }
        }
        // normalize Monte Carlo estimates and sum up contributions of light sources
        L_reflect += L_reflect_l/N_l;
    } // add code to account for point and directional light sources here
    return L_reflect; }
```



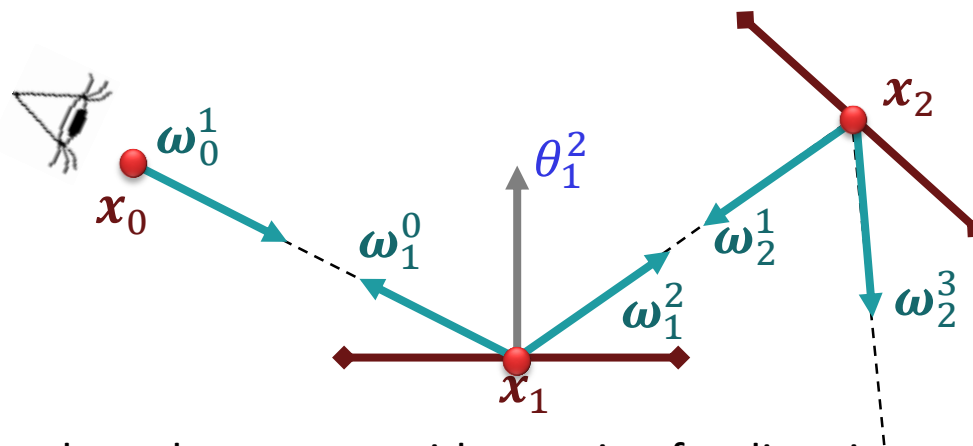
$$L_{\lambda}^{\text{direct}}(\mathbf{x}, \omega^{\text{out}}) = \sum_{l=1}^{N_l^{\text{area}}} L_{\lambda,l}^{\text{direct,area}}(\mathbf{x}, \omega^{\text{out}}) + \dots$$

- When we have a large number of light sources, one wants to avoid sampling all of them potentially with several shadow rays
- Similar to Russian Roulette one can probabilistically select one light source and then just sample this for the current estimate.
- For this one defines a probability $p_l^{\text{area|pnt|dir}}$ for each light source such that summation over probabilities yields one.
- Then one chooses one (or more) light source[s] according to the assigned probabilities, estimates the direct light contribution $L_{\lambda,l}^{\text{direct,*}}$ with one (or several) sample[s] and returns as estimate $L_{\lambda,l}^{\text{direct,*}}/p_l^*$
- One option is to assign the probabilities proportional to the emitted spectral power of the light sources

Strategies to choose the number of samples, i.e. Pathtracing



- ◆ The optimal number of samples to estimate the integrals (indirect sampling directions, area light samples) depends on the scene
- ◆ There is no generally optimal strategy
- ◆ **Path-Tracing** chooses only one indirect light sample and comes in several variants with respect to sampling of the direct lights. Sampling a large number of primary rays is always necessary to reduce variance sufficiently.



example path $x_0, x_1, \dots$ with notation for directions and angles



Monte Carlo Path Tracing

- Formal solution of rendering equation:

$$\begin{aligned} L_{\lambda}^{\text{out}} &= (\mathbf{I} - \mathbf{R} \cdot \mathbf{T})^{-1} L_{\lambda}^{\text{emit}} \\ &= L_{\lambda}^{\text{emit}} + \mathbf{RT} \cdot L_{\lambda}^{\text{emit}} + (\mathbf{RT})^2 \cdot L_{\lambda}^{\text{emit}} + \dots \\ &= L_{\lambda}^{\text{emit}} + \mathbf{RT} \left(L_{\lambda}^{\text{emit}} + \mathbf{RT} (L_{\lambda}^{\text{emit}} + \dots) \right) \end{aligned}$$

- Each application of $\mathbf{R}$ is a nested integral over directions, which can be approximated with one sample ω_j^{in}

$$L_{\lambda,j}^{\text{reflect}}(\omega_j^{j-1}) \approx \frac{\rho(\omega_j^{j+1}, \omega_j^{j-1}) \cos \theta_j^{j+1}}{p(\omega_j^{j+1})} L_{\lambda,j}^{\text{in}} =: \frac{w_{\lambda,j}}{p_j} \cdot L_{\lambda}^{\text{in}}(\omega_j^{j+1})$$

- Plugging in with $L_{\lambda,j}^{\text{e}} := L_{\lambda}^{\text{emit}}(\underline{x}_j, \omega_j^{j-1})$ yields:

$$L_{\lambda}^{\text{out}}(\underline{x}_0, \omega_0^1) = \frac{w_{\lambda,1}}{p_1} \cdot \left(\frac{w_{\lambda,2}}{p_2} \cdot \left(\dots \cdot \left(\frac{w_{\lambda,n}}{p_n} L_{\lambda,n+1}^{\text{e}} \right) \dots \right) \right)$$

Summary

- ◆ The support of colors, spatio temporal filtering, and depth of field yields a large number of parameters to integrate over resulting in a large number of primary rays
- ◆ The reflection integral can be approximated by sampling the Hemisphere and using Monte Carlo estimation.
- ◆ Importance sampling can be done for
 - ◆ Cosine term with Nusselt's analogon sampling the unit disk
 - ◆ By splitting the BRDF into diffuse and specular parts
 - ◆ By splitting incoming radiance into direct and indirect illumination and sampling for the direct part the light sources directly
- ◆ Path Tracing is a Monte Carlo integration technique where reflection from indirect illumination is approximated with one sample per recursion