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 $\boldsymbol{M}_{\lambda,j}^{\Omega\times A}$ and by specifying the integration domain as superscript to show dimensionality of integration:

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

measurement equation

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

rendering equation

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

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

Motivation

Computergraphik und Visualisierung

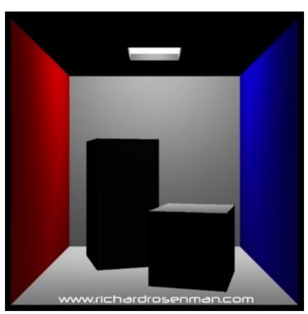
- Formal solution of rendering equation:

$$L_\lambda^{\mathrm{out}} = (\boldsymbol{I} - \boldsymbol{R} \cdot \boldsymbol{T})^{-1} L_\lambda^{\mathrm{emit}}$$
$$= L_\lambda^{\mathrm{emit}} + \boldsymbol{RT} \cdot L_\lambda^{\mathrm{emit}} + (\boldsymbol{RT})^2 \cdot L_\lambda^{\mathrm{emit}} + \cdots$$

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

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

- This is a $2 \times 2 + 2 + 2 + 2 + \cdots = \infty$ 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

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

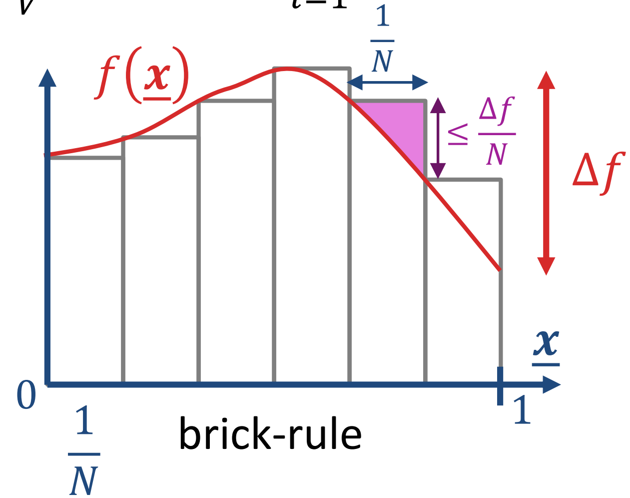
- in 2D we get with $n = \sqrt{N}$

$$\epsilon_2 = n^2 \cdot \frac{1}{2}\frac{\Delta f}{n}\frac{1}{n^2} = \frac{\Delta f}{2\sqrt{N}}$$

- and in d dimensions

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

$$F = \int_V f(\underline{x})d\underline{x} \approx \sum_{i=1}^{N} f(\underline{x}_i)\cdot\omega(\underline{x}_i)$$



- 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

- random variable X distributed on domain Ω with $p(x)$
- sample random variable: $x \sim X$
- given function $f(x)$ on Ω expectation value is defined as

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

- now let us take N samples $x_i \sim X$ and compute expectation by integrating once per sample:

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

- 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.

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

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

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

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:

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

- 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.

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

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

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

Expectation Value Estimator Variance

Computergraphik und Visualisierung

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

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

  since the x_i are independent samples
- 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:

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

Variance Proof

Computergraphik und Visualisierung

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

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

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

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

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

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

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

Monte Carlo Integration

Computergraphik und Visualisierung

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

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

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

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

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:

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

  independent of $\underline{x}_0$ resulting in no variance at all
- 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)

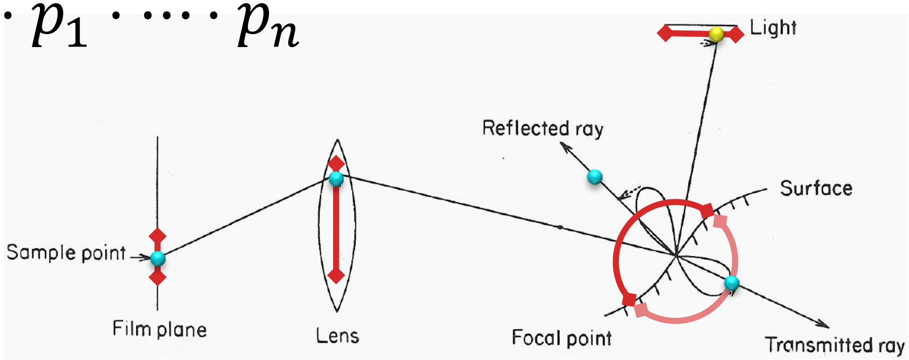
MONTE CARLO
GLOBAL ILLUMINATION

Distribution Raytracing

Computergraphik und Visualisierung

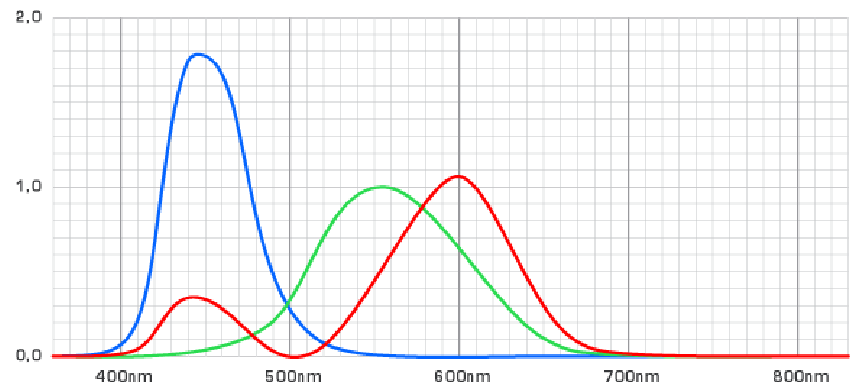
- The contributions to a pixel depend on parameters for light measurement and for the light path
- In distribution and Monte Carlo raytracing all parameters are sampled independently such that the probability densities can be multiplied
- typically the resulting pixel contribution from light transport 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

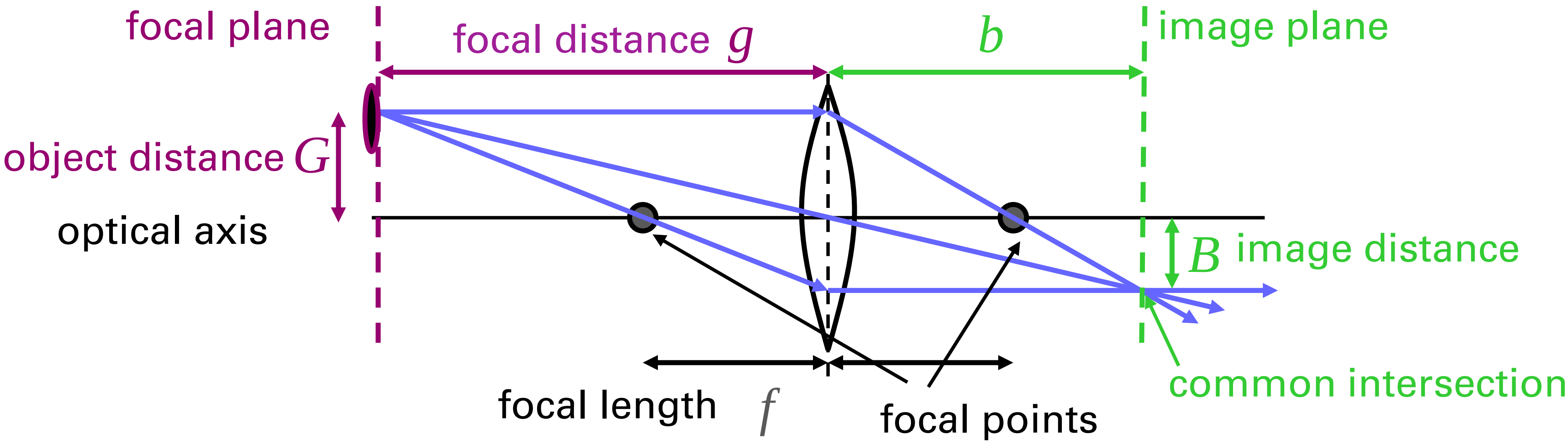
$$X_j = \int_\lambda \bar{x}(\lambda) \cdot \Phi_{\lambda,j} d\lambda \quad Y_j = \int_\lambda \bar{y}(\lambda) \cdot \Phi_{\lambda,j} d\lambda \quad Z_j = \int_\lambda \bar{z}(\lambda) \cdot \Phi_{\lambda,j} d\lambda$$

- To sample only once per computed color value, one can sample λ from the sum of the efficiency curves, which correspond to the photopic luminousity function:

$$p(\lambda) \propto x(\lambda) + y(\lambda) + z(\lambda)$$

Thin Lense

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



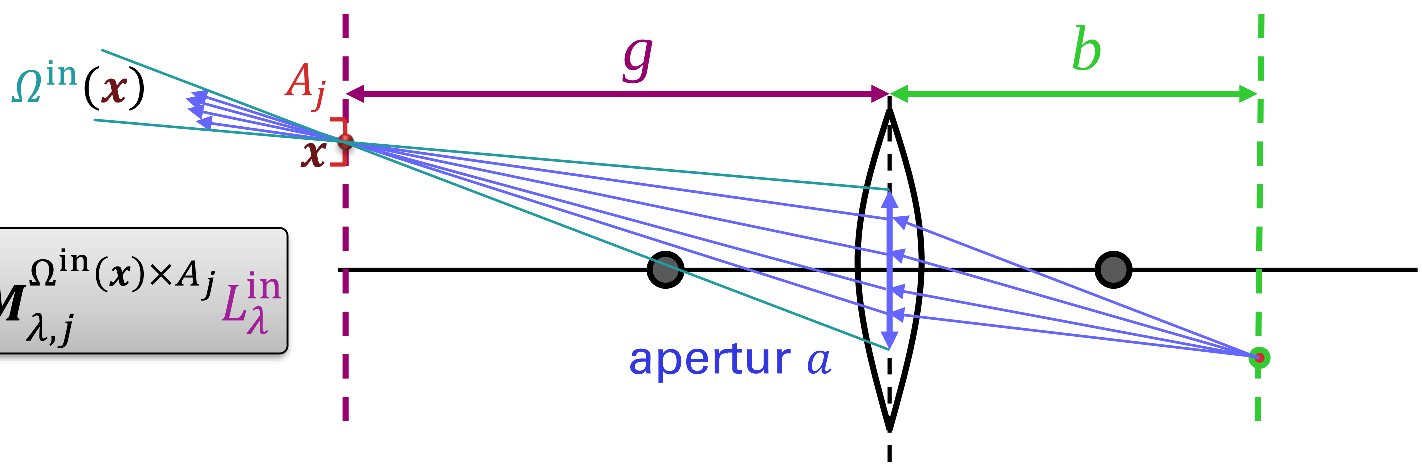
lense equation

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

Depth of Field

- To simulate depth of field, we need to know the focal plane (defined through focal distance g) and the aperture size a (measured as area of circle)
- The pixel area A_j over which we integrate $\boldsymbol{x}$ is defined on the focal plane
- The directions $\Omega_j^{\text{in}}(\boldsymbol{x})$ over which we integrate are spanned by $\boldsymbol{x}$ and the aperture a.

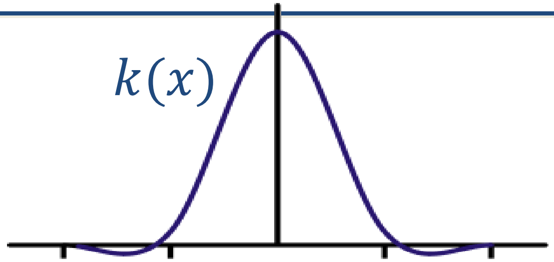
$$\Phi_{\lambda,j} = \boldsymbol{M}_{\lambda,j}^{\Omega^{\text{in}}(\boldsymbol{x})\times A_j} L_\lambda^{\text{in}}$$



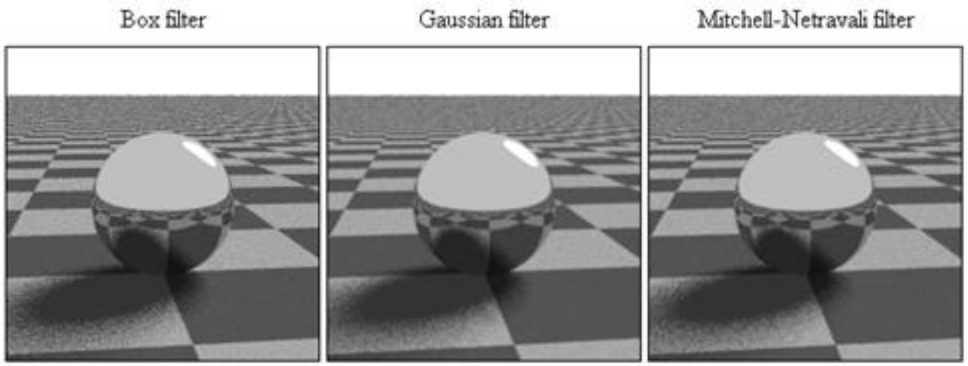
Spatial and Temporal Filtering

Computergraphik und Visualisierung

- uniform sampling of the pixel area corresponds to a box filter
- A better approximation to the theoretically optimal sinc-filter is the polynomial Mitchell-Netravali-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-Netravali-Filter



Approximating the Reflection Integral

Computergraphik und Visualisierung

- To approximate the directional form

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

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

- Given N samples $\boldsymbol{\omega}_i^{\text{in}}$ the integral is approximated by

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

- Two questions arise here:
  - To compute $L_\lambda^{\text{in}}(\boldsymbol{\omega}_i^{\text{in}})$ we need a recursion, but when to terminate this?
  - How to choose $p(\boldsymbol{\omega}_i^{\text{in}})$ for efficient importance sampling?

Termination of Recursion through Russian Roulette

Computergraphik und Visualisierung

- Given a function $f(x)$ over V and an estimator $\hat{f}$ for the integral F of $f(x)$ over V: $\mathrm{E}[\hat{f}] = F$
- 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
$$\widehat{RR}_{\hat{f}}(s) = \begin{cases} \hat{f}/s & \text{if} \quad b = 1 \\ 0 & \text{if} \quad b = 0 \end{cases}$$
- The expectation value of $\widehat{RR}_{\hat{f}}(s)$ is the same and still F:
$\mathrm{E}[\widehat{RR}_{\hat{f}}(s)] = s \cdot \mathrm{E}[\hat{f}/s] + (1-s) \cdot 0 = \mathrm{E}[\hat{f}]$
- The variance is increased, though.
- With the Russian Roulette estimator $\widehat{RR}_{\hat{L}_\lambda^{\mathrm{reflect}}}(s)$ 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 aperture, 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_shuffel(l, p_λ, N);
                  // sample time values for motion blurr
                  std::vector<float> t, p_t; sample_MN_filter_statified_and_shuffel(t, p_t, dt, N);
                  // sample pixel for antialiasing
                  std::vector<float> x, p_x; sample_MN_filter_statified_and_shuffel(x, p_x, dx, N);
                  std::vector<float> y, p_y; sample_MN_filter_statified_and_shuffel(y, p_y, dy, N);
                  // sample aperture for depth of field
                  std::vector<P2D>   a, p_a; sample_circle_statified_and_shuffel(a, p_a, sqrt(aperture/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]);
                        // set potential from sensor efficiency
                        float potential = spectral_sensitivity(λ[k]);
                        // compute incoming radiance
                        float L_in_λ = incoming_radiance(scene, ray, λ[k], potential);
                        // add contributions to color channel integrators
                        X += L_in_λ*x_sensitivity(λ[k])/(p_x[k]*p_y[k]*p_a[k]*p_t[k]*p_λ[k]);
                        Y += L_in_λ*y_sensitivity(λ[k])/(p_x[k]*p_y[k]*p_a[k]*p_t[k]*p_λ[k]);
                        Z += L_in_λ*z_sensitivity(λ[k])/(p_x[k]*p_y[k]*p_a[k]*p_t[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

Computergraphik und Visualisierung

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

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


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

Reference Implementation reflected radiance

Computergraphik und Visualisierung

```
float reflected_radiance(const Scene& scene, const HitInfo& hit,
                         const V3D& omega_out,
                         float λ, 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), λ, 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

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

$$= L_\lambda^{\text{out}}\left(\boldsymbol{y}, -\boldsymbol{\omega}^{\text{in}}\right), \qquad \boldsymbol{y} = \text{trace}\left(\boldsymbol{x}, \boldsymbol{\omega}^{\text{in}}\right)$$

$$= L_\lambda^{\text{emit}}\left(\boldsymbol{y}, -\boldsymbol{\omega}^{\text{in}}\right) + L_\lambda^{\text{reflect}}\left(\boldsymbol{y}, -\boldsymbol{\omega}^{\text{in}}\right)$$

split incoming radiance in direct and indirect illumination to support direct light sampling

$$\rho_{\text{diff}} + \rho_{\text{spec}} + \rho_{\text{mirr}}$$

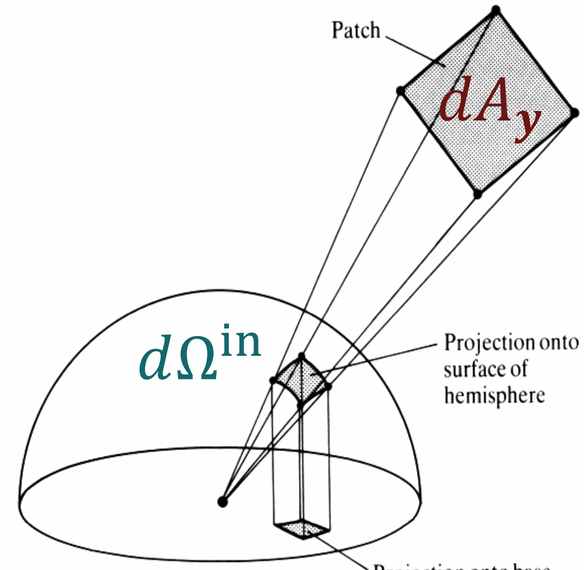
split BRDF into diffuse and specular / mirror reflection parts and adapt sampling

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:

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

- As a result we sampled according to $p_{\text{Nusselt}}(\boldsymbol{\omega}^{\text{in}}) = \cos\theta^{\text{in}}/\pi$.



$$\cos\theta^{\text{in}} d\Omega^{\text{in}}$$

$$\|$$

$$\sout{V(\boldsymbol{x},\boldsymbol{y})} \frac{\cos\theta^{\boldsymbol{x}\to\boldsymbol{y}} \cos\theta^{\boldsymbol{y}\to\boldsymbol{x}}}{\|\boldsymbol{x}-\boldsymbol{y}\|^2} dA_{\boldsymbol{y}}$$

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

Direct Light Sampling

- splits the reflection integral into direct & indirect part:

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

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

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

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

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

- Next one rewrites direct illumination in area formulation

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

Direct Light Sampling

Computergraphik und Visualisierung

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

$$L_\lambda^{\text{direct}}(\boldsymbol{x},\boldsymbol{\omega}^{\text{out}}) = \sum_{l=1}^{N_l^{\text{area}}} \iint_{\boldsymbol{y}\in A_l} \rho(\boldsymbol{x},\boldsymbol{\omega}^{\boldsymbol{x}\to\boldsymbol{y}},\boldsymbol{\omega}^{\text{out}}) L_{\lambda,l}^{\text{emit,area}}(\boldsymbol{y},\boldsymbol{\omega}^{\boldsymbol{y}\to\boldsymbol{x}}) G(\boldsymbol{x},\boldsymbol{y})\, dA_{\boldsymbol{y}}$$

$$+ \sum_{l=1}^{N_l^{\text{pnt}}} \rho\left(\boldsymbol{x},\boldsymbol{\omega}^{\boldsymbol{x}\to\boldsymbol{y}_l^{\text{pnt}}},\boldsymbol{\omega}^{\text{out}}\right) I_{\lambda,l}^{\text{emit,pnt}}\left(\boldsymbol{y}_l^{\text{pnt}},\boldsymbol{\omega}^{\boldsymbol{y}_l^{\text{pnt}}\to\boldsymbol{x}}\right) V\left(\boldsymbol{x},\boldsymbol{y}_l^{\text{pnt}}\right) \frac{\cos\theta^{\boldsymbol{x}\to\boldsymbol{y}_l}}{\left\|\boldsymbol{x}-\boldsymbol{y}_l^{\text{pnt}}\right\|^2}$$

$$+ \sum_{l=1}^{N_l^{\text{dir}}} \rho\left(\boldsymbol{x},\boldsymbol{\omega}_l^{\text{dir}},\boldsymbol{\omega}^{\text{out}}\right) H_{\lambda,l}^{\text{emit,dir}}\left(-\boldsymbol{\omega}_l^{\text{dir}}\right) \cos\theta_l^{\text{dir}} V^{\text{dir}}\left(\boldsymbol{x},\boldsymbol{\omega}_l^{\text{dir}}\right)$$

- 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.
- $V^{\text{dir}}(\boldsymbol{x},\boldsymbol{\omega})$ checks if the ray $(\boldsymbol{x},\boldsymbol{\omega})$ hits the background

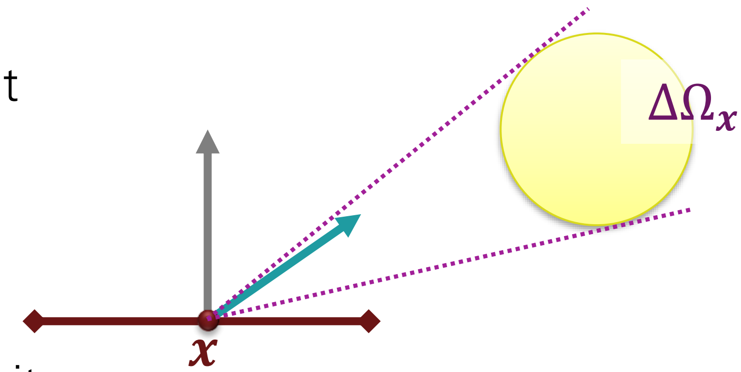
Direct Light Sampling of Area Lights

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

- Per area light l we need to sample a number N_l of points $\boldsymbol{y}_i$ on the light source and estimate the direct light contribution

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

- 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.



Indirect Light Sampling

Computergraphik und Visualisierung

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

- 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

Computergraphik und Visualisierung

```
float direct_reflected_radiance(const Scene& scene, const HitInfo& hit,
                                const V3D& omega_out,
                                float λ, 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; }
```

Dealing with Large Number of Lights

Computergraphik und Visualisierung

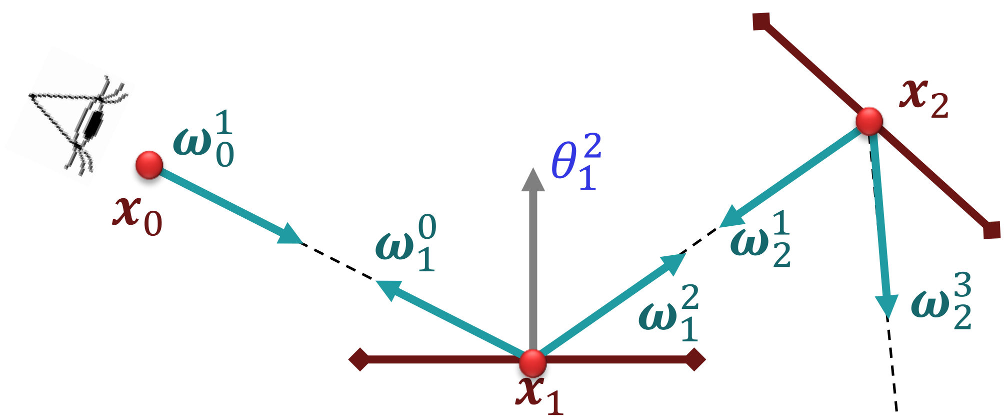
$$L_\lambda^{\text{direct}}(\boldsymbol{x}, \boldsymbol{\omega}^{\text{out}}) = \sum_{l=1}^{N_l^{\text{area}}} L_{\lambda,l}^{\text{direct,area}}(\boldsymbol{x}, \boldsymbol{\omega}^{\text{out}}) + \cdots$$

- 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

Computergraphik und Visualisierung

- 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 $x_0, x_1, \ldots$ with notation for directions and angles

Monte Carlo Path Tracing

- Formal solution of rendering equation:

$$L_\lambda^{\text{out}} = (\boldsymbol{I} - \boldsymbol{R}\cdot\boldsymbol{T})^{-1} L_\lambda^{\text{emit}}$$
$$= L_\lambda^{\text{emit}} + \boldsymbol{RT}\cdot L_\lambda^{\text{emit}} + (\boldsymbol{RT})^2\cdot L_\lambda^{\text{emit}} + \cdots$$
$$= L_\lambda^{\text{emit}} + \boldsymbol{RT}\left(L_\lambda^{\text{emit}} + \boldsymbol{RT}\left(L_\lambda^{\text{emit}} + \cdots\right)\right)$$

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

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

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

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

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