Monte Carlo Techniques for Realistic Computer Graphics

Advanced Image Synthesis — Spring 2006

Outline

1. Introduction
2. Mathematical Background
3. Solving the Rendering Equation
5. Path Tracing
6. Metropolis Light Transport
7. Biased Techniques
8. Conclusions

Introduction

Measurement Equation

- The measurement equation gives the image color at a pixel

$$M = \int_{A_{pixel}} \int_{T} \int_{\Omega} R(x,\omega,t) L(x,\omega,t) dt\, d\omega\, dA$$

  - Includes pixel filter, aperture and shutter
  - Other effects such as lens system transformation, spectral sensitivity, ... could be included as well

Properties Measurement Equation

- Very high dimensional integral
  - In general, can not be solved analytically
- Rendering Equation is Fredholm integral equation of the 2nd kind
- Integrand has usually many discontinuities
- Point sampling the equation can be performed easily
- Required accuracy is not very high compared to many other applications

Mathematical Background

Monte Carlo Integration

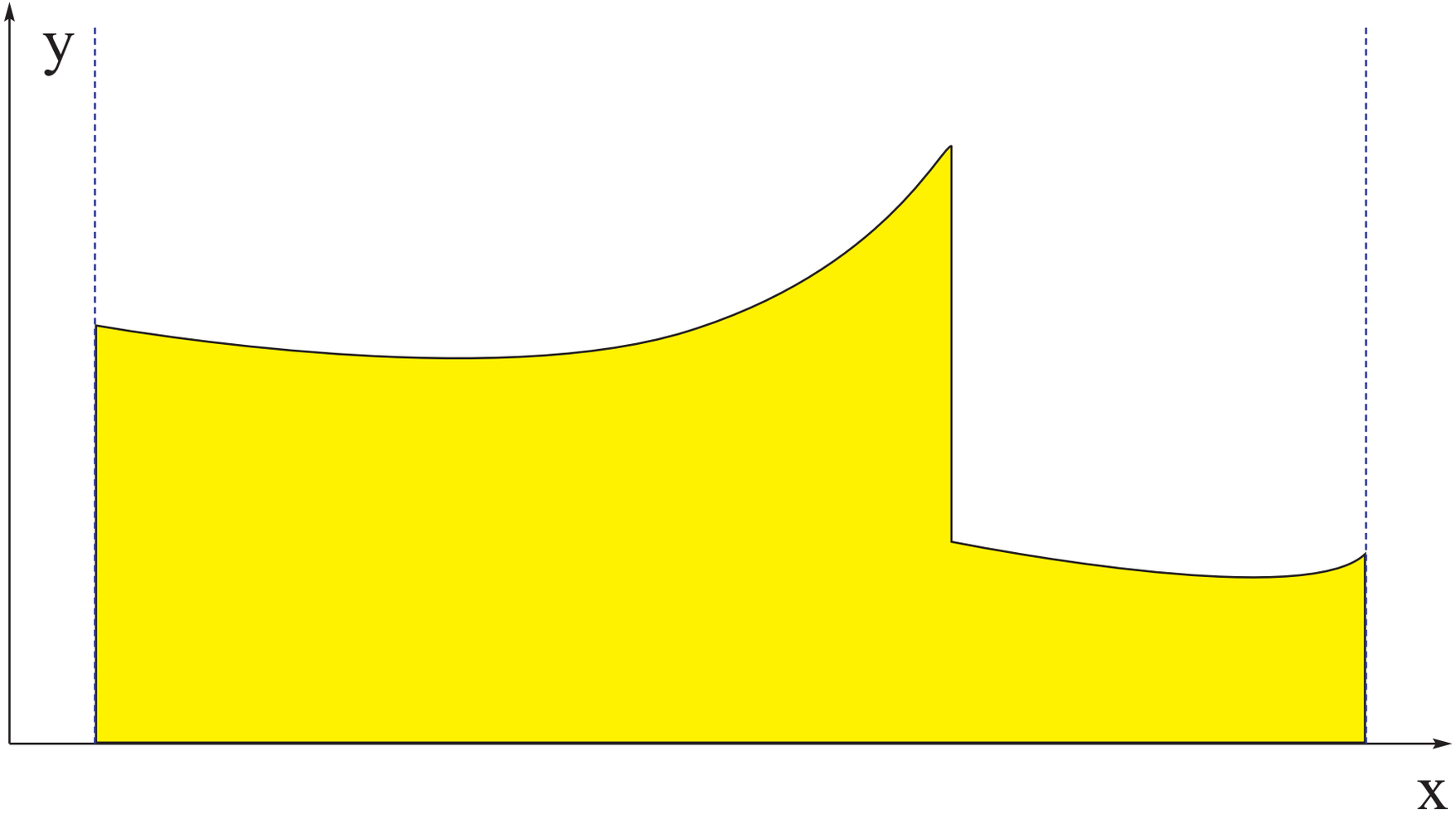
- Basic Monte Carlo method (1D)

$$I = \int_a^b f(x)dx \approx \frac{1}{N}\sum_{i=1}^{N}\frac{f(x_i)}{p(x_i)} = \hat{I}$$

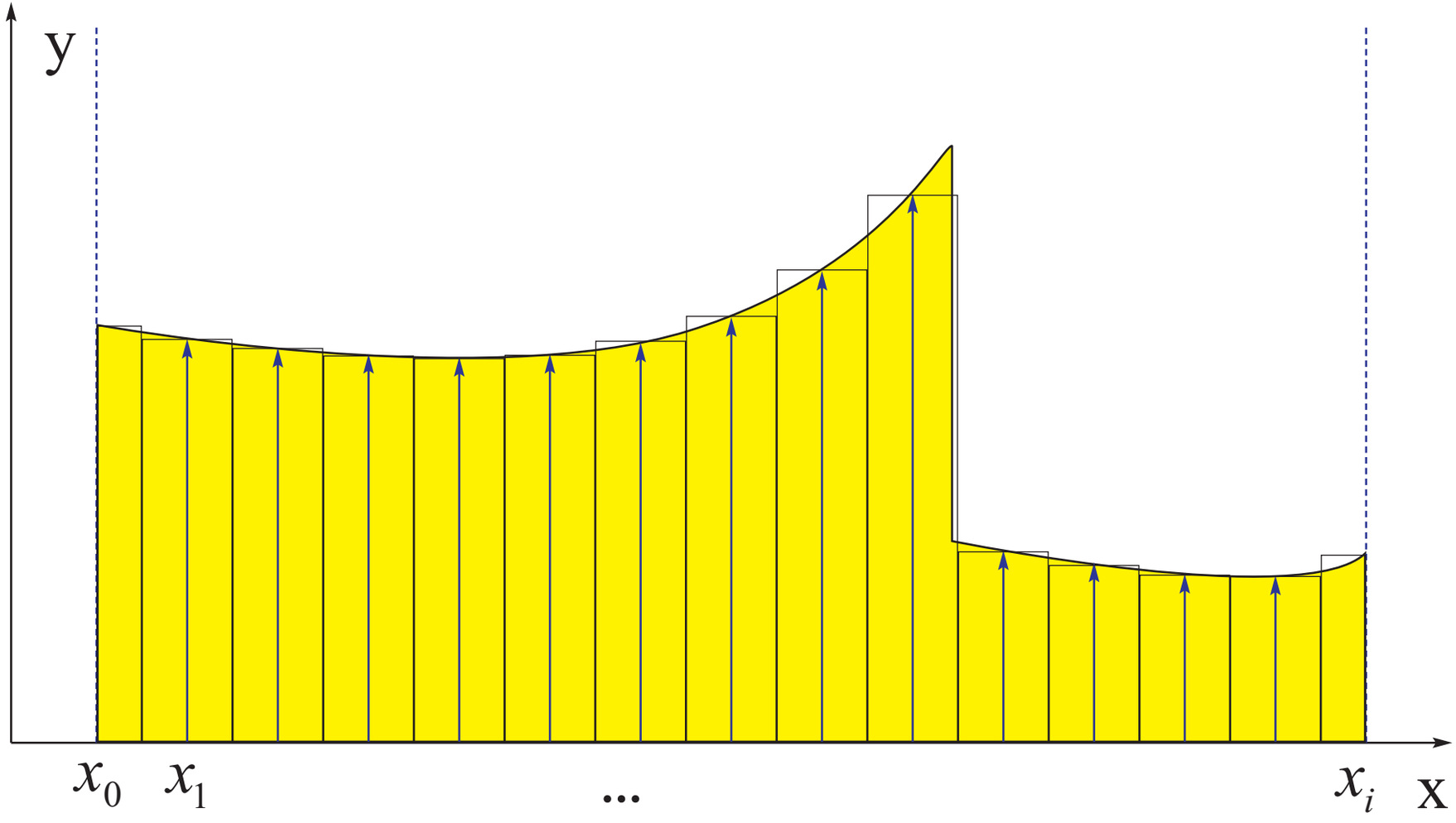
where x_i are samples drawn from the probability distribution p

- Convergence of $O\left(\frac{1}{\sqrt{N}}\right)$ independent of
  - Dimensionality of the integral
  - Smoothness of the integrand

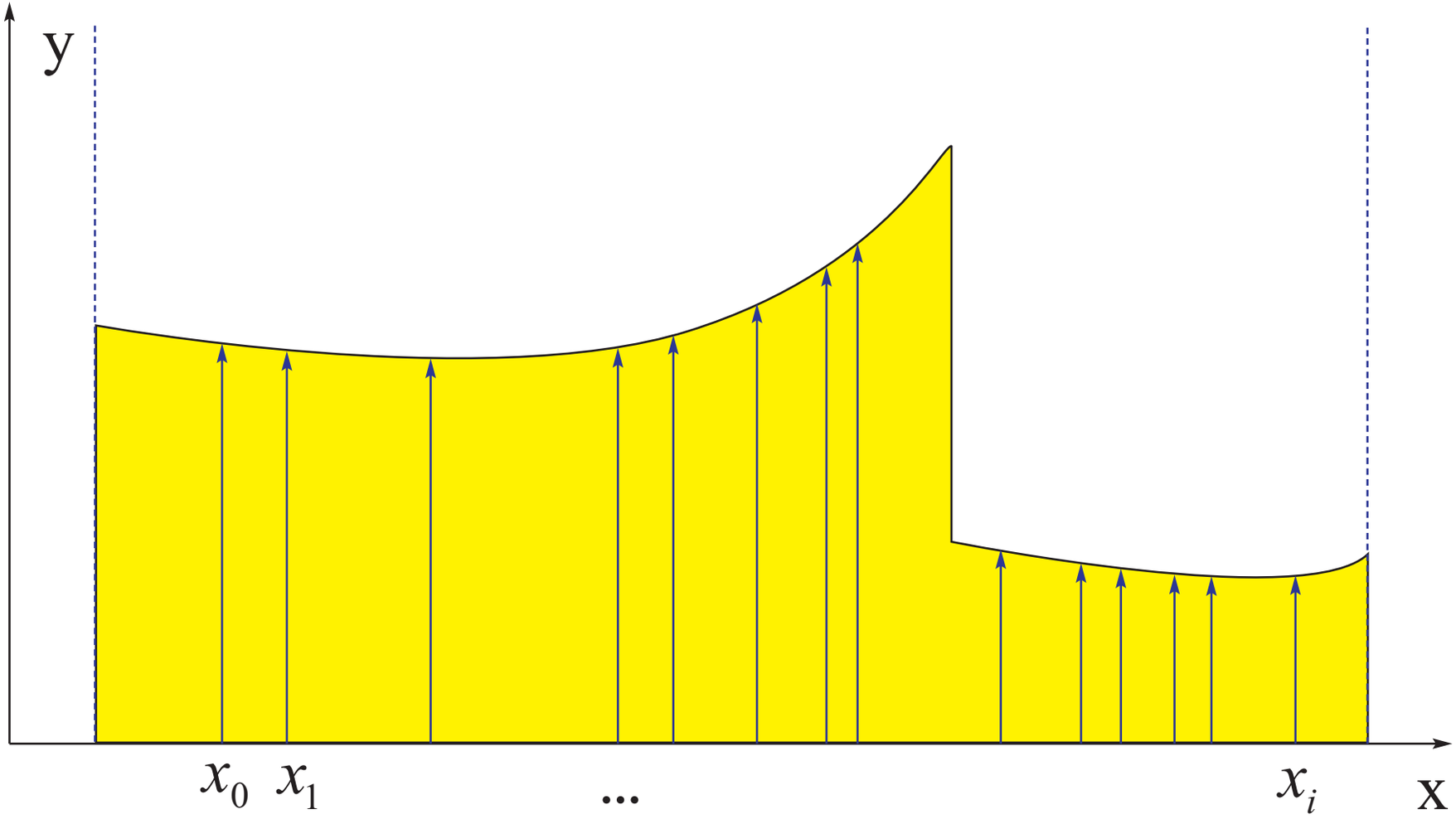
Midpoint Rule



Midpoint Rule



Monte Carlo Integration



Do we get the correct solution?

$$I = \int_a^b f(x)dx \approx \frac{1}{N}\sum_{i=1}^{N}\frac{f(x_i)}{p(x_i)}$$

x_i from $\mathrm{P}[a,b]$, i.e. an arbitrary probability distribution on $[a,b]$

$$E[\hat{I}] = E\left[\frac{1}{N}\sum_{i=1}^{N}\frac{f(x_i)}{p(x_i)}\right]$$

$$E[\hat{I}] = \frac{1}{N}\sum_{i=1}^{N}E\left[\frac{f(x_i)}{p(x_i)}\right]$$

$$E[\hat{I}] = \frac{1}{N}\sum_{i=1}^{N}\int_{-\infty}^{\infty}\frac{f(x)}{p(x)}p(x)dx$$

$$E[\hat{I}] = \frac{1}{N}\sum_{i=1}^{N}\int_{b}^{a}f(x)dx$$

$$E[\hat{I}] = \int_{-a}^{b}f(x)dx = I$$

Biased vs. Unbiased

- Unbiased: $E[\hat{I}] = I$
- Biased: $\beta(\hat{I}) = E[\hat{I}] - I, \qquad \beta(\hat{I}) \neq 0$
  - Consistent: $\lim_{n \to \infty} \beta(\hat{I}) = 0$

Error of Monte Carlo Integration

- Mean Squared Error: $MSE(\hat{I}) = E[(\hat{I} - I)^2]$

$$MSE(\hat{I}) = E\left[(\hat{I} - I)^2\right]$$

$$MSE(\hat{I}) = E\left[(\hat{I} - E(\hat{I}) + E(\hat{I}) - I)^2\right]$$

$$MSE(\hat{I}) = E\left[(\hat{I} - E(\hat{I}) + E(\hat{I}) - I)^2\right]$$

...

$$MSE(\hat{I}) = V(\hat{I}) + \beta(\hat{I})^2$$

Error of Monte Carlo Integration

- Mean Squared Error: $MSE(\hat{I}) = V(\hat{I}) + \beta(\hat{I})^2$
- Unbiased estimator
  - $\beta(\hat{I})^2 = 0$
  - MSE can be estimated by variance of independent samples x_i
  - Variance exposes as high-frequency noise in the image

-> Very likely that a noise free image shows the "correct" solution

Error of Monte Carlo Integration

- Mean Squared Error: $MSE(\hat{I}) = V(\hat{I}) + \beta(\hat{I})^2$
- Biased estimator
  - $\beta(\hat{I})^2 \neq 0$
  - Very hard to determine error bounds
  - Error can expose arbitrarily in the image

Efficiency (unbiased estimator)

- Efficiency measure

$$\varepsilon(\hat{I}) = \frac{1}{V(\hat{I}) \cdot T(\hat{I})}$$

$V(\hat{I})$ Variance of $\hat{I}$

$T(\hat{I})$ Time to compute the estimate

Importance Sampling

- Efficiency measure

$$\varepsilon(\hat{I}) = \frac{1}{V(\hat{I}) \cdot T(\hat{I})}$$

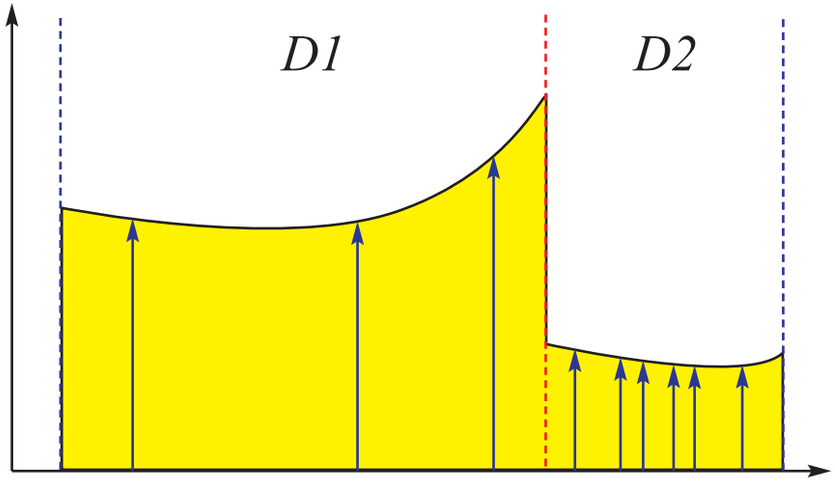
$V(\hat{I})$ Variance of $\hat{I}$

$T(\hat{I})$ Time to compute the estimate

- Idea: "Use minimal number of samples and take them at the right places."

Importance Sampling

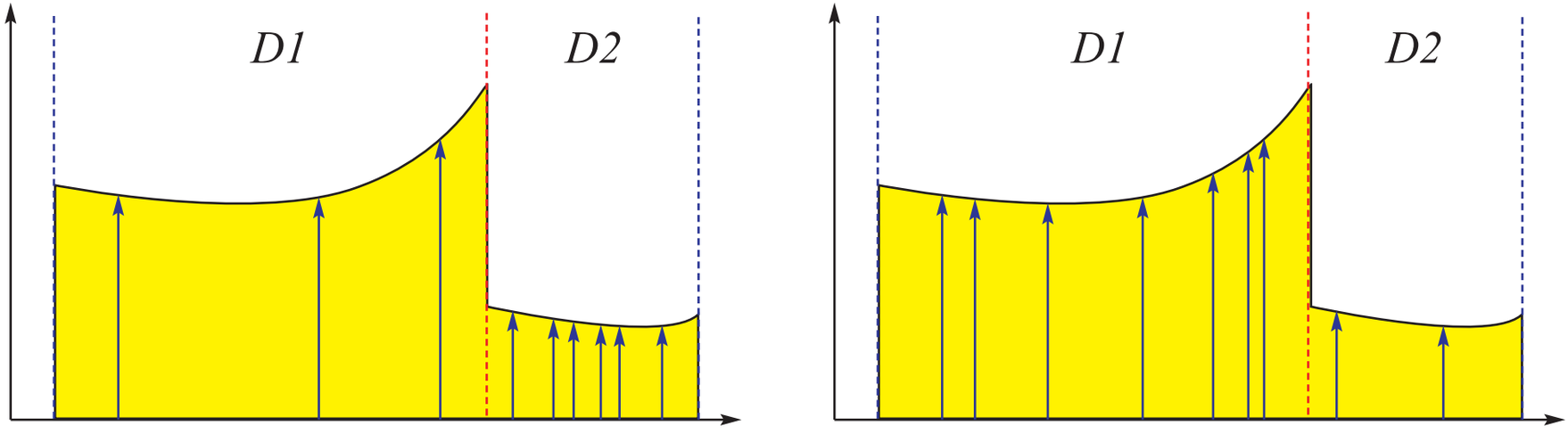
- What are the "right" places?
- Sub-domains where the function value is high, e.g. *D1* have a bigger influence on the value of the integrand



Importance Sampling

- What are the "right" places?
- Sub-domains where the function value is high, e.g. $D1$ have a bigger influence on the value of the integrand

=> Minimize the error in regions where $f(x)$ is high

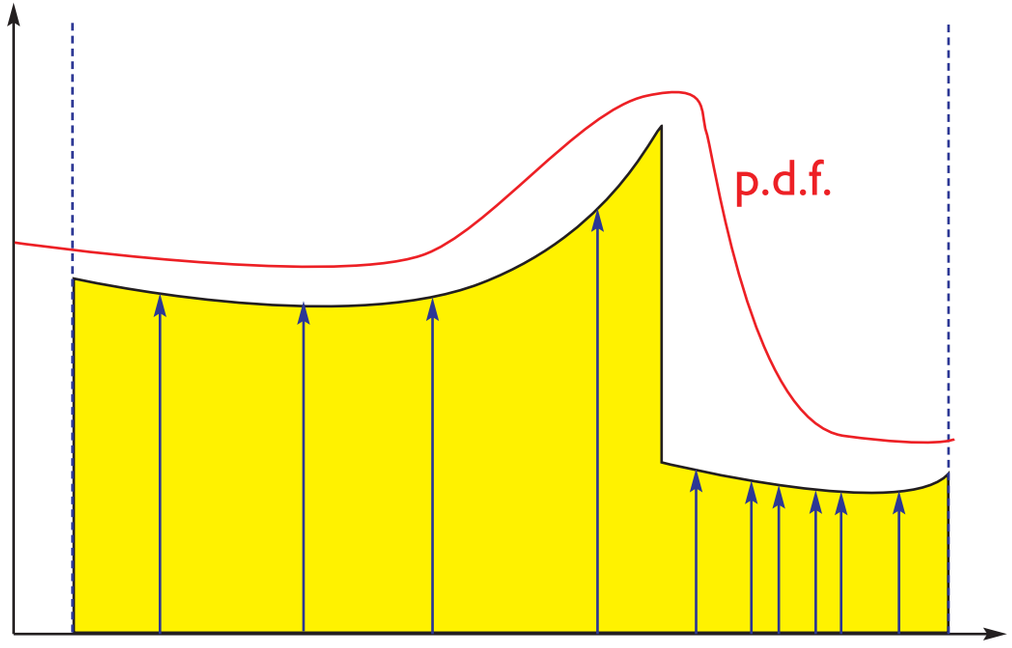


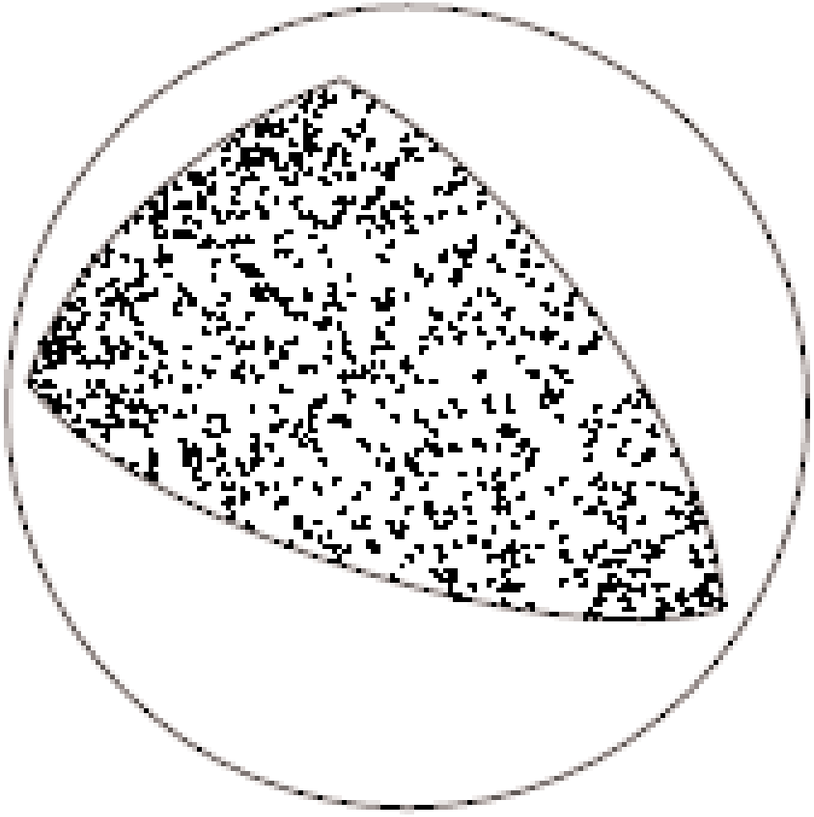
Importance Sampling

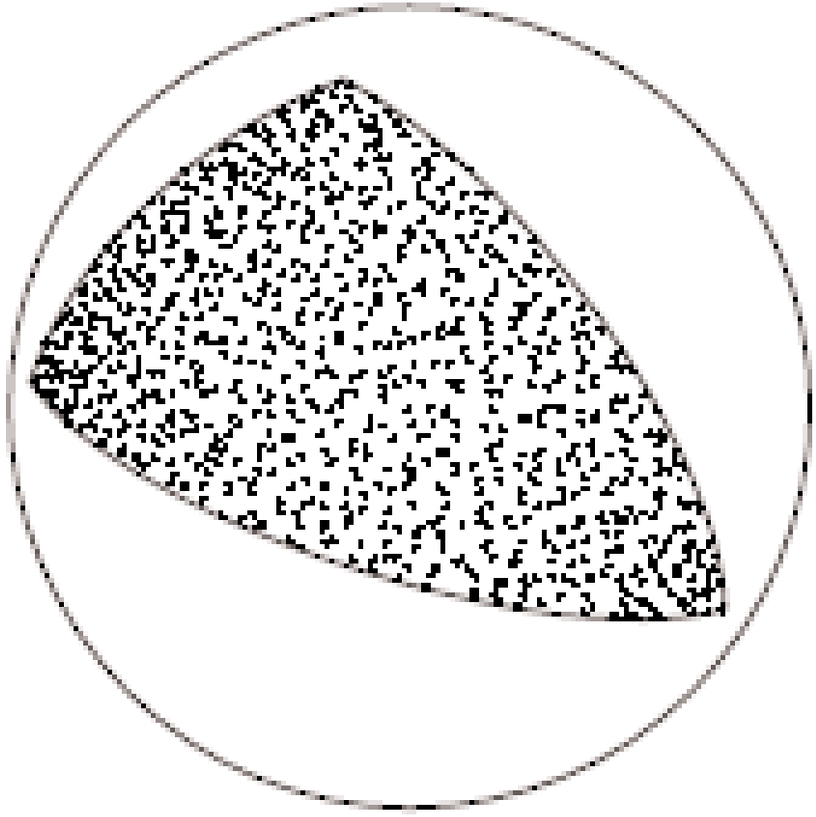
- The probability distribution function (p.d.f.) used to generate the samples and the function to integrate should have a similar shape
  - The p.d.f. can be an arbitrary function as long as the samples are weighted correctly
- Russian Roulette
- Splitting
- Control Variates is another, similar technique for variance reduction

Stratified Sampling

- Even an appropriate p.d.f. can lead to poor sampling
  - The p.d.f. only increases the probability to have a sampling density proportional to the integrand







Stratified Sampling of Spherical Triangles, James Arvo

random sampling

stratified random sampling

Stratified Sampling

- Subdivide domain Δ into non-overlapping sub-domains Δ_1, Δ_2, ...
  - Δ_i is called a stratum
  - The union of all strata covers the whole original integration domain
- Samples are drawn from each stratum Δ_i according to the density p_i inside each stratum
- The variance is guaranteed to be equal or less the variance without stratified sampling

Sampling Random Variables

- Inversion method
- Rejection method
- How to sample specific geometry is covered in all standard texts
  - triangles
  - disks
  - hemispheres
  - see [1], [2], [3], ...

Quasi Monte Carlo

- Certain restrictions on the randomness of the number improve the efficiency of the integration (e.g. stratified sampling)
- Only pseudo random numbers can be generated on a computer

Quasi Monte Carlo

- Certain restrictions on the randomness of the number improve the efficiency of the integration (e.g. stratified sampling)
- Only pseudo random numbers can be generated on a computer

=> Use deterministic numbers with certain properties instead of random numbers

- Numbers should be well distributed on the domain

Quasi Monte Carlo

- Convergence for smooth functions

$$O\left(\frac{(\log N)^{s-1}}{N}\right)$$

- s is the dimensionality of the integration

Quasi Monte Carlo

- Convergence for smooth functions

$$O\left(\frac{(\log N)^{s-1}}{N}\right)$$

  - s is the dimensionality of the integration

- Rendering Equation is not smooth
  - Performance in practice shows usually slight advantages for Quasi Monte Carlo techniques

Quasi Monte Carlo

- Stratified sampling and Latin hypercube sampling are intrinsic properties of Quasi Monte Carlo methods
  - Halton Sequence can be used to generate pseudo-stratified samples even if the number of samples is not known in advance
- Implementation is simple compared to random Monte Carlo techniques
  - Especially on parallel computers, where the generation of non-correlated random variables is non-trivial.

Solving the Rendering Equation

Solving Integral Equations

- Fredholm integral equation of the second kind

$$f(x) = g(x) + \int_0^1 K(x,y) \cdot f(y)\, dy$$

- Solution with Finite Element
  - Function might be too complex to reconstruct over the entire domain
  - Might not be necessary to reconstruct entire function
- Solve with Monte Carlo Integration and treat as regular integral

Solving Integral Equations

$$f(x) = g(x) + \frac{K(x,\xi_1)}{p_1(\xi_1)} f_r(\xi_1)$$

Solving Integral Equations

$$f(x) = g(x) + \frac{K(x,\xi_1)}{p_1(\xi_1)} f_r(\xi_1)$$

$$f(x) = g(x) + \frac{K(x,\xi_1)}{p_1(\xi_1)} \left[g(\xi_1) + \frac{K(\xi_1,\xi_2)}{p_1(\xi_2)} f_r(\xi_2) \right]$$

Solving Integral Equations

$$f(x) = g(x) + \frac{K(x,\xi_1)}{p_1(\xi_1)} f_r(\xi_1)$$

$$f(x) = g(x) + \frac{K(x,\xi_1)}{p_1(\xi_1)} \left[g(\xi_1) + \frac{K(\xi_1,\xi_2)}{p_1(\xi_2)} f_r(\xi_2) \right]$$

$$f(x) = g(x) + \frac{K(x,\xi_1)}{p_1(\xi_1)} g(\xi_1) + g(\xi_1) \frac{K(x,\xi_1)}{p_1(\xi_1)} \frac{K(\xi_1,\xi_2)}{p_1(\xi_2)} + \ldots$$

Solving Integral Equations

$$f(x) = g(x) + \frac{K(x,\xi_1)}{p_1(\xi_1)} f_r(\xi_1)$$

$$f(x) = g(x) + \frac{K(x,\xi_1)}{p_1(\xi_1)} \left[g(\xi_1) + \frac{K(\xi_1,\xi_2)}{p_1(\xi_2)} f_r(\xi_2) \right]$$

$$f(x) = g(x) + \frac{K(x,\xi_1)}{p_1(\xi_1)} g(\xi_1) + g(\xi_1) \frac{K(x,\xi_1)}{p_1(\xi_1)} \frac{K(\xi_1,\xi_2)}{p_1(\xi_2)} + \ldots$$

$$f(x) = \sum_{i=0}^{\infty} \left[\prod_{j=1}^{i} \frac{K(\xi_{j-1},\xi_j)}{p_1(\xi_j)} \right] \cdot g(\xi_i)$$

Solving Integral Equations

- Operator form of the Integral Equation

$$f = g + T \circ f$$

- Solution is formally given by the Neumann Series

$$f = g + T \circ g + T^2 \circ g + T^3 \circ g + \ldots$$

$$f = \sum_{k=0}^{\infty} T^k \circ g$$

- Discarding all higher-order terms after some T^k will introduce bias

Solving Integral Equations

- Russian Roulette
  - Stop evaluation of the k-th term in a probabilistic way
  - Compensate for truncation

$$f = g + \frac{1}{1-q_1}(T \circ g) + \frac{1}{1-q_2}(T^2 \circ g) + \frac{1}{1-q_3}(T^3 \circ g) + \ldots$$

  - Termination probability q can be chosen arbitrarily, e.g. based on an estimate of the term value

Solving Integral Equations

- Russian Roulette
  - Stop evaluation of the k-th term in a probabilistic way
  - Compensate for truncation

$$f = g + \frac{1}{1-q_1}(T \circ g) + \frac{1}{1-q_2}(T^2 \circ g) + \frac{1}{1-q_3}(T^3 \circ g) + \ldots$$

  - Termination probability q can be chosen arbitrarily, e.g. based on an estimate of the term value
- Unbiased estimator with increased variance

Solving the Rendering Equation

- Rendering equation is Fredholm integral equation of the second kind

$$L(x,\omega) = L_e(x,\omega) + \int_{\Omega} f_r(x,\omega_i,\omega) L(x,\omega_i)(n \cdot \omega_i) d\omega_i$$

- Operator form of Rendering Equation

$$L(x,\omega) = I \circ L_e + K^1 \circ L_e + K^1 \circ L_e + K^2 \circ L_e + \ldots$$

$$L(x,\omega) = \sum_{i=0}^{\infty} K^i \circ L_e$$

- Monte Carlo integration with Russian Roulette can be used to solve this equation

Solving the Rendering Equation

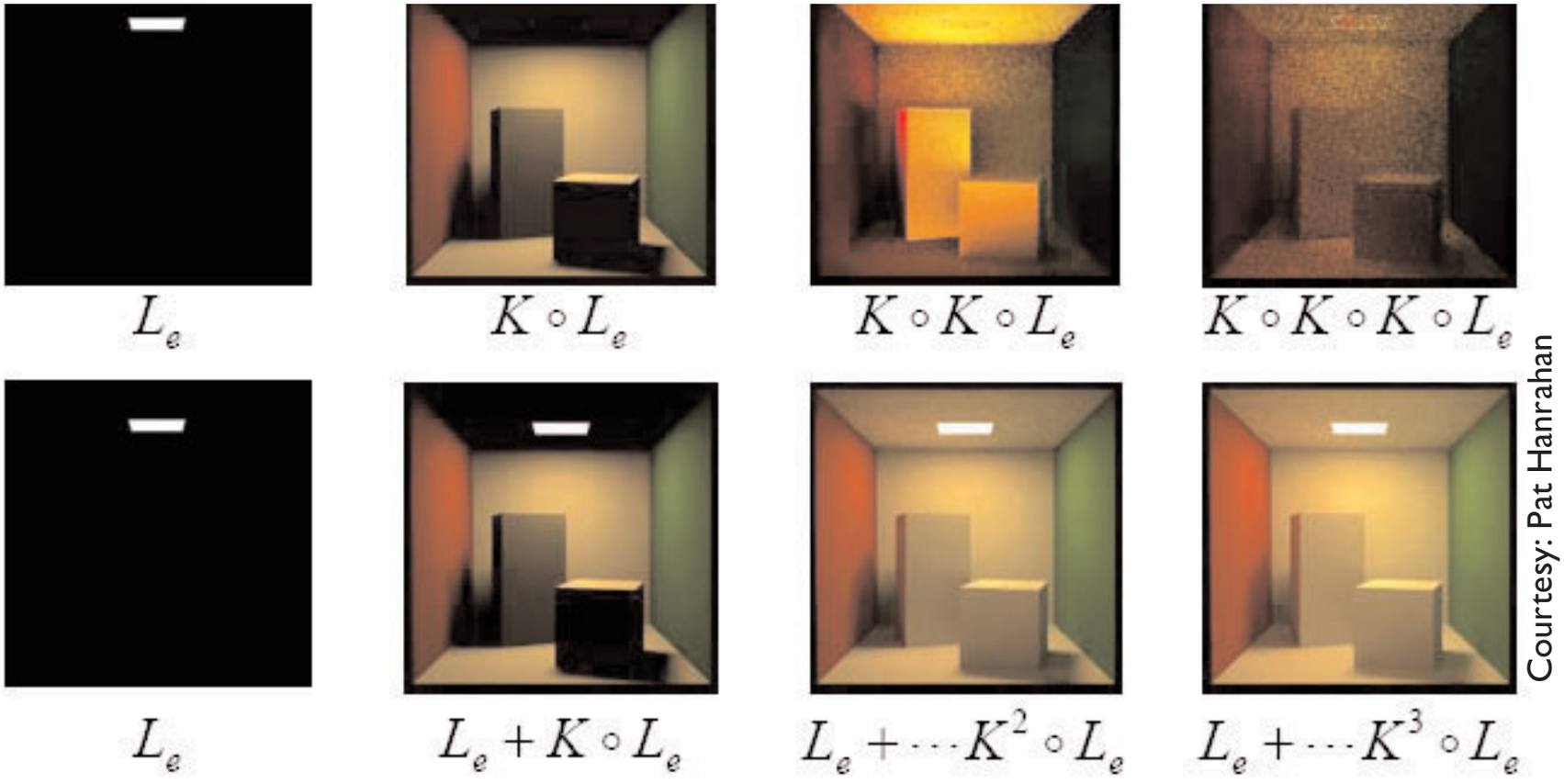
- General solution for rendering equation

$$f = g + \frac{1}{1-q_1}(T \circ g) + \frac{1}{1-q_2}(T^2 \circ g) + \frac{1}{1-q_3}(T^3 \circ g) + \ldots$$

- Random Walk
  - Each K^i corresponds to the light flow between i surface points
- Russian Roulette
  - Light path are randomly terminated after i steps

Solving the Rendering Equation

- Successive approximation of the rendering equation



Courtesy: Pat Hanrahan

Path Tracing

Path Tracing

- Most "fully" random walks have zero or a very small contribution
  - Usually, at least one pair of consecutive vertices of the paths will be not mutually visible
  - For glossy and specular surfaces most paths will have a very small contribution because they are far off the peak direction of the BRDF
- Unable to sample delta distributions, e.g. point lights

Path Tracing

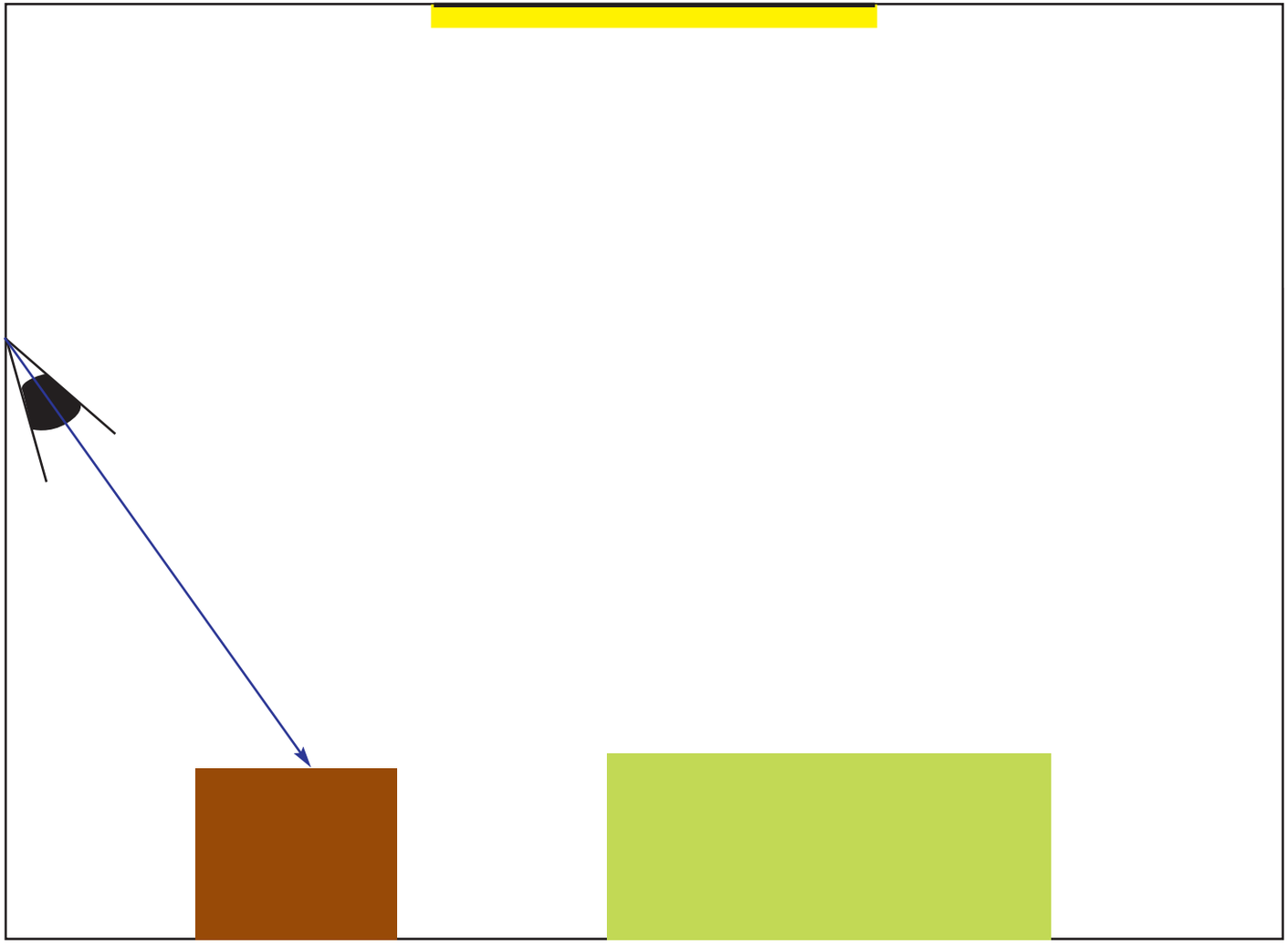
- Most “fully” random walks have zero or a very small contribution
  - Usually, at least one pair of consecutive vertices of the paths will be not mutually visible
  - For glossy and specular surfaces most paths will have a very small contribution because they are far off the peak direction of the BRDF
- Unable to sample delta distributions, e.g. point lights

=> Naive path tracing very inefficient

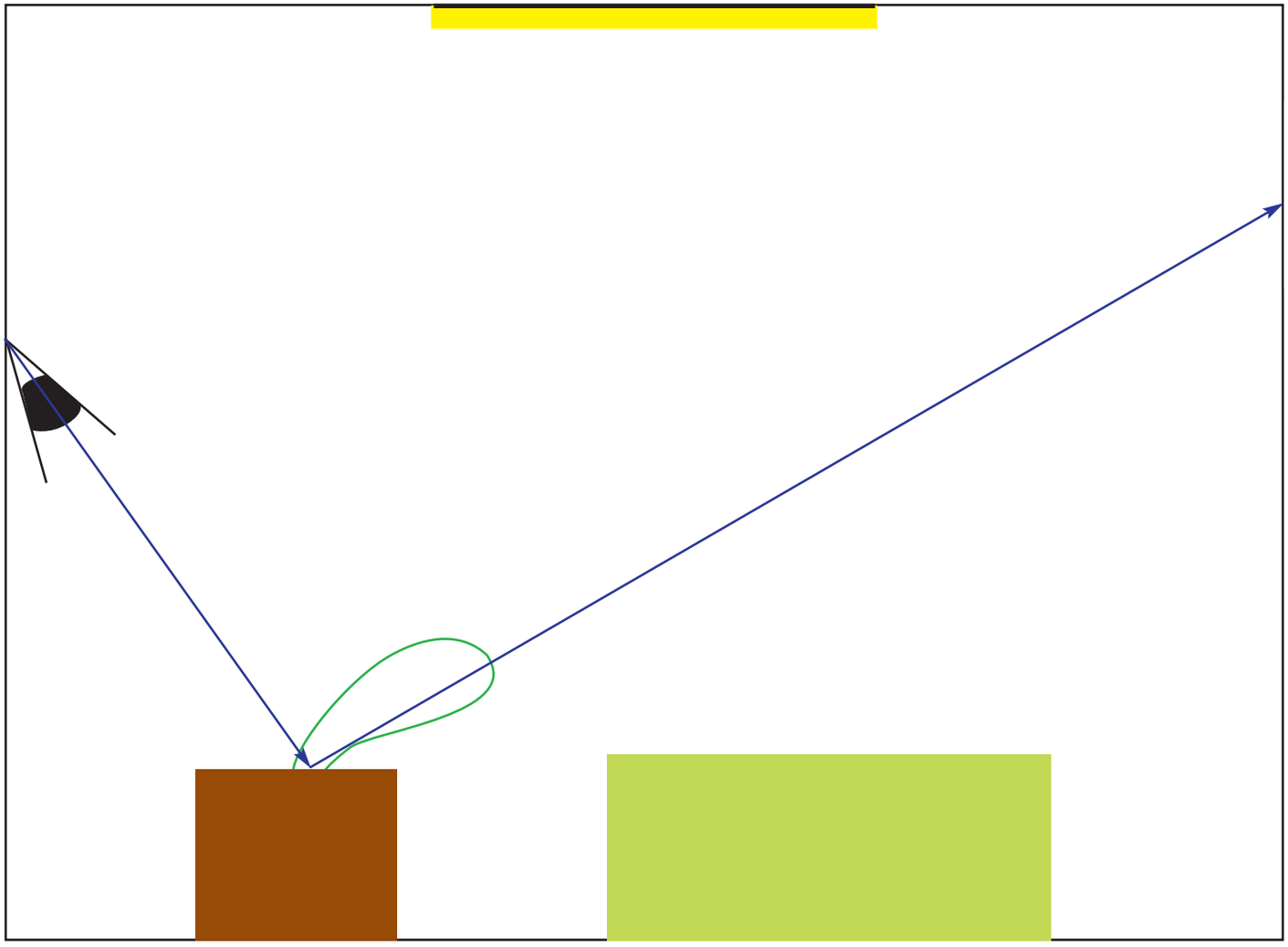
Path Tracing

1. Generate a random ray
   - Random w.r.t. to all domains (time, pixel area, ...)
2. Find intersection point closest to the ray origin
3. Russian Roulette: randomly decide either
   - Terminate path: Store (weight * incident_radiance)
   - Extend path:
     - Scatter new ray according to the BRDF
     - Update probability of path
     - Go back to 2.

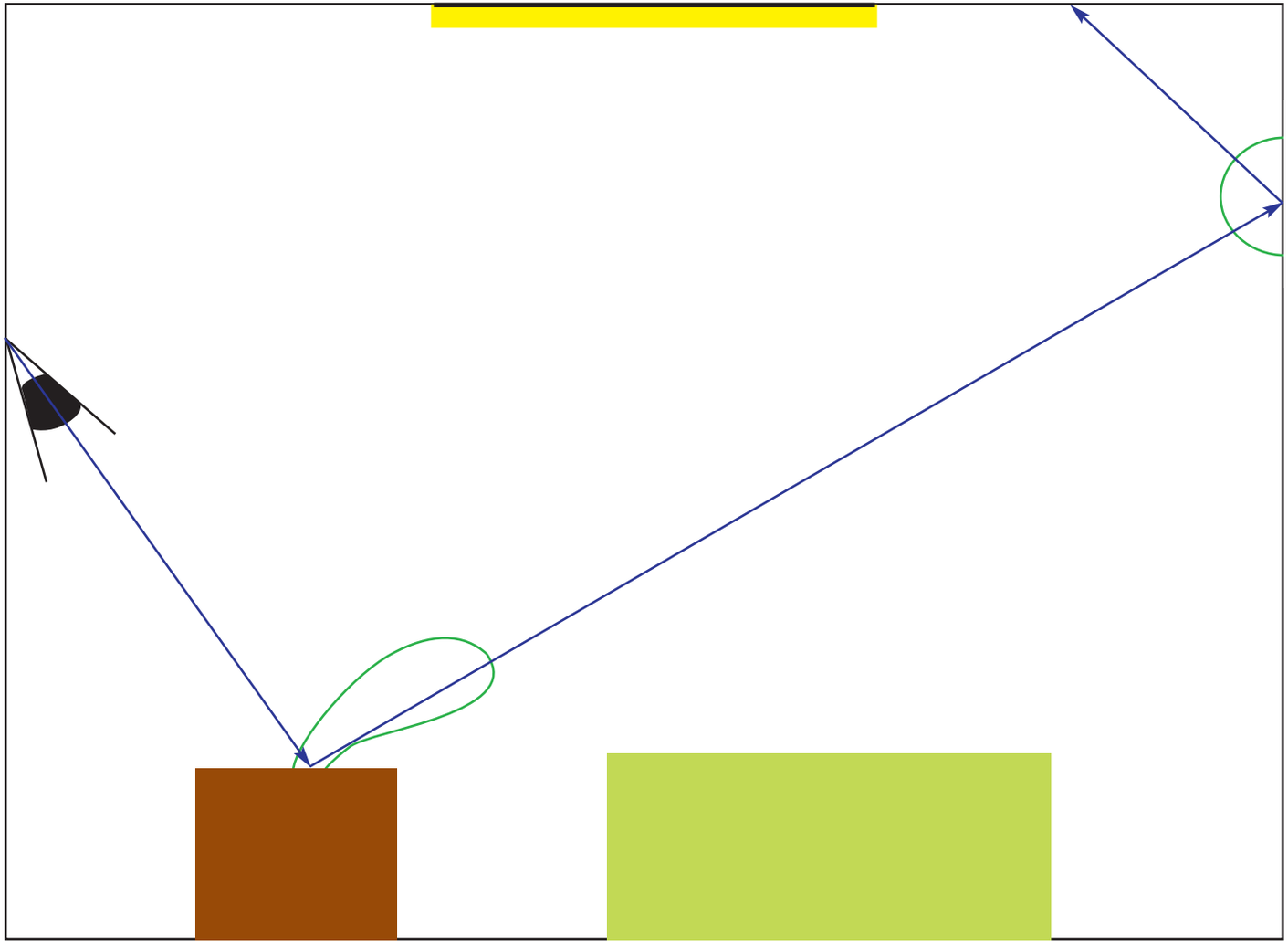
Path Tracing



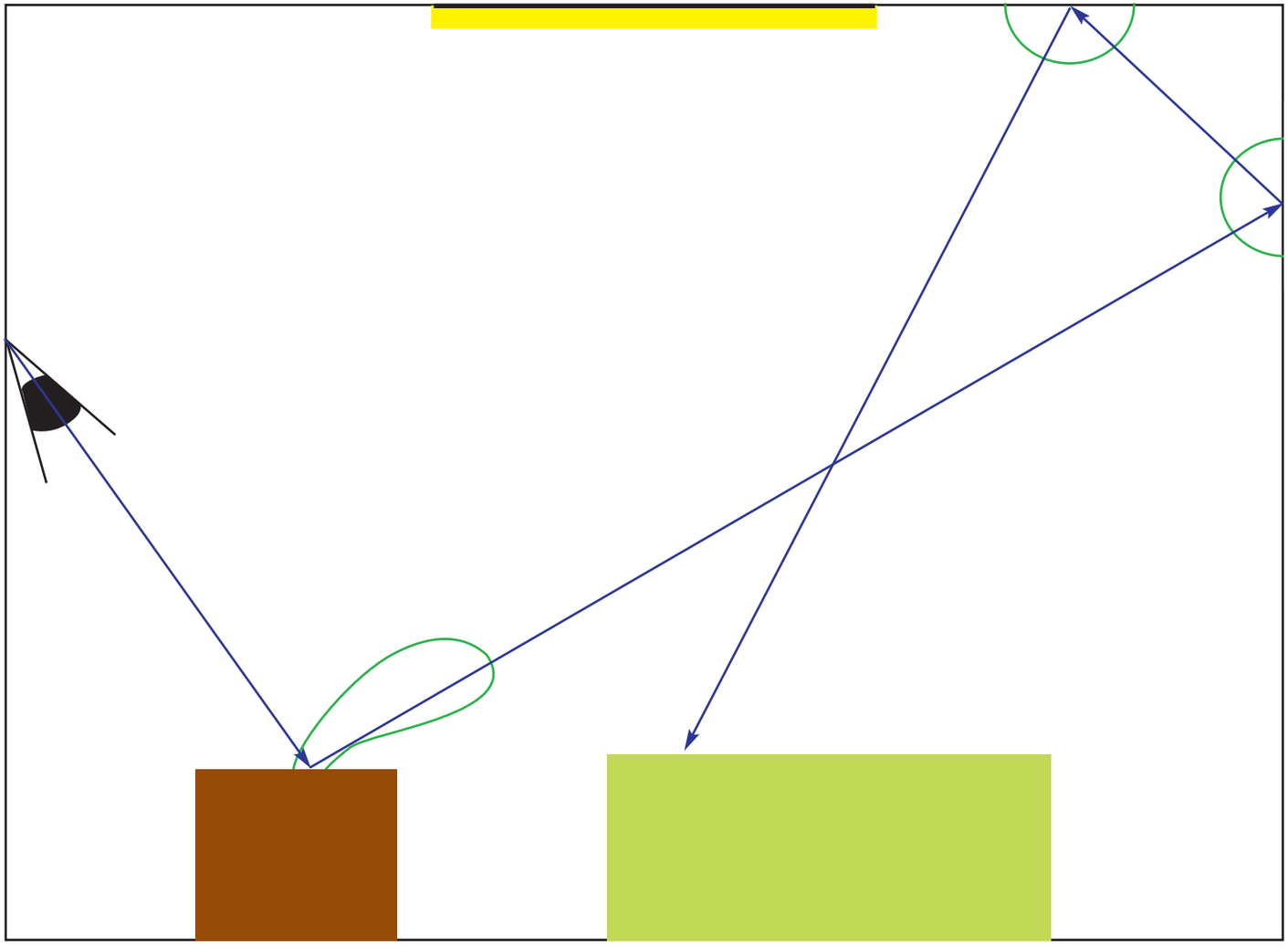
Path Tracing



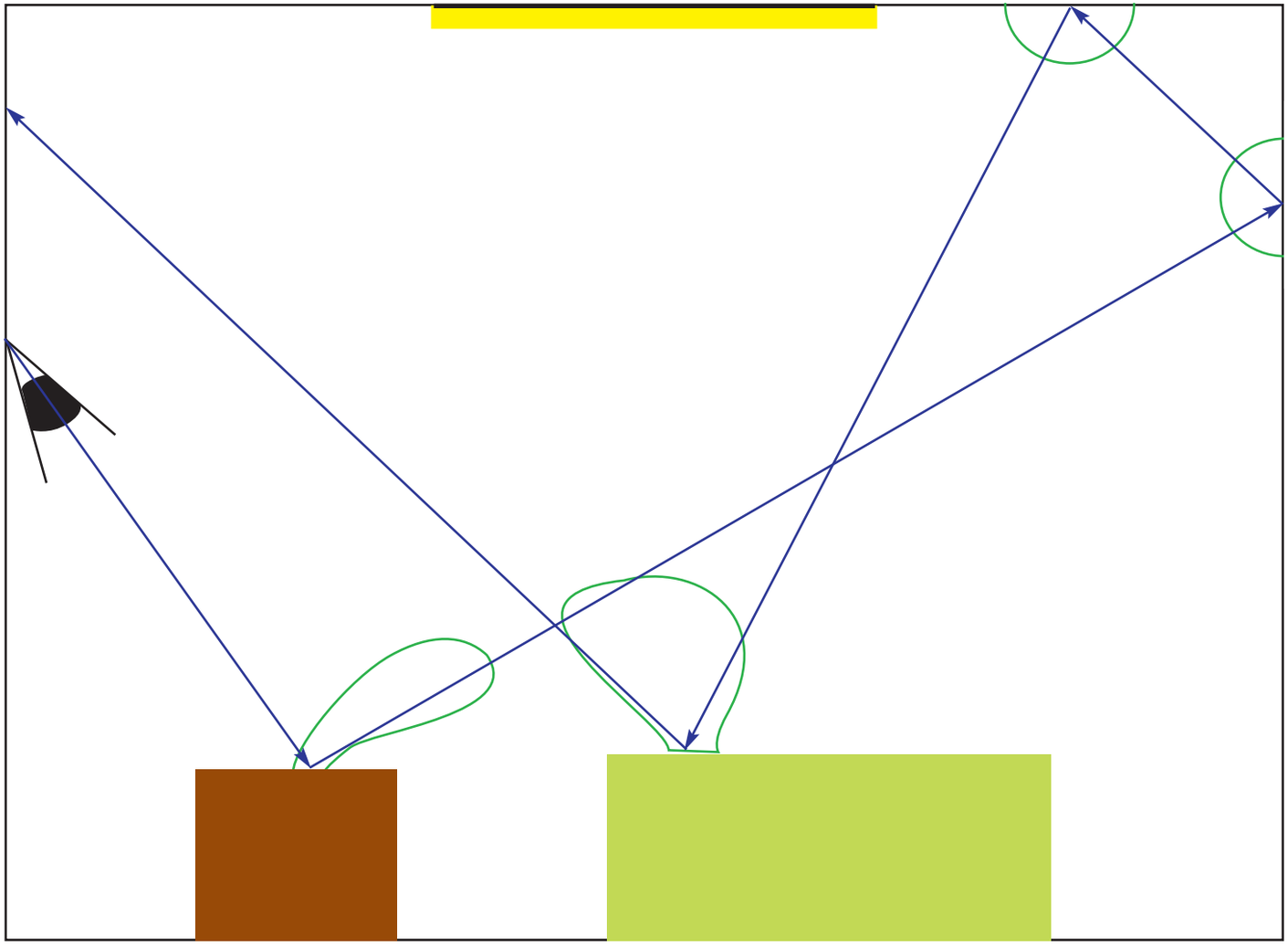
Path Tracing



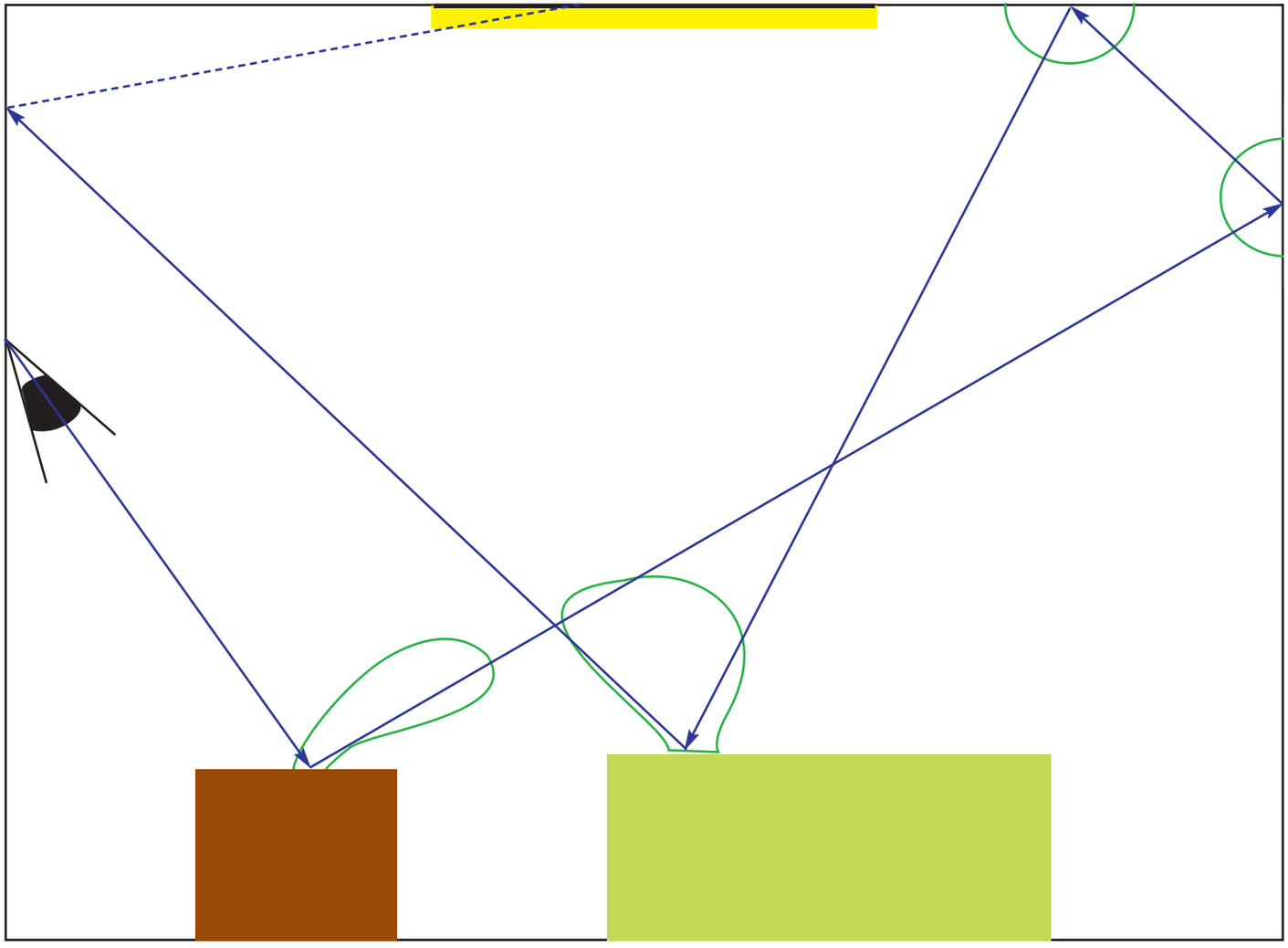
Path Tracing



Path Tracing



Path Tracing



Path Tracing

- Importance sampling can perform poorly
  - Rendering Equation is integral over multiple terms each with its own distributions
  - Mirror-like BRDF and samples distributed w.r.t. light source
  - Diffuse BRDF and small light source and light distributed w.r.t. BRDF

Path Tracing

- Importance sampling can perform poorly
  - Rendering Equation is integral over multiple terms each with its own distributions
  - Mirror-like BRDF and samples distributed w.r.t. light source
  - Diffuse BRDF and small light source and light distributed w.r.t. BRDF
- Combine multiple sampling strategies
  - Naive approach does not work — variance is additive
  - Unbiased estimator desired

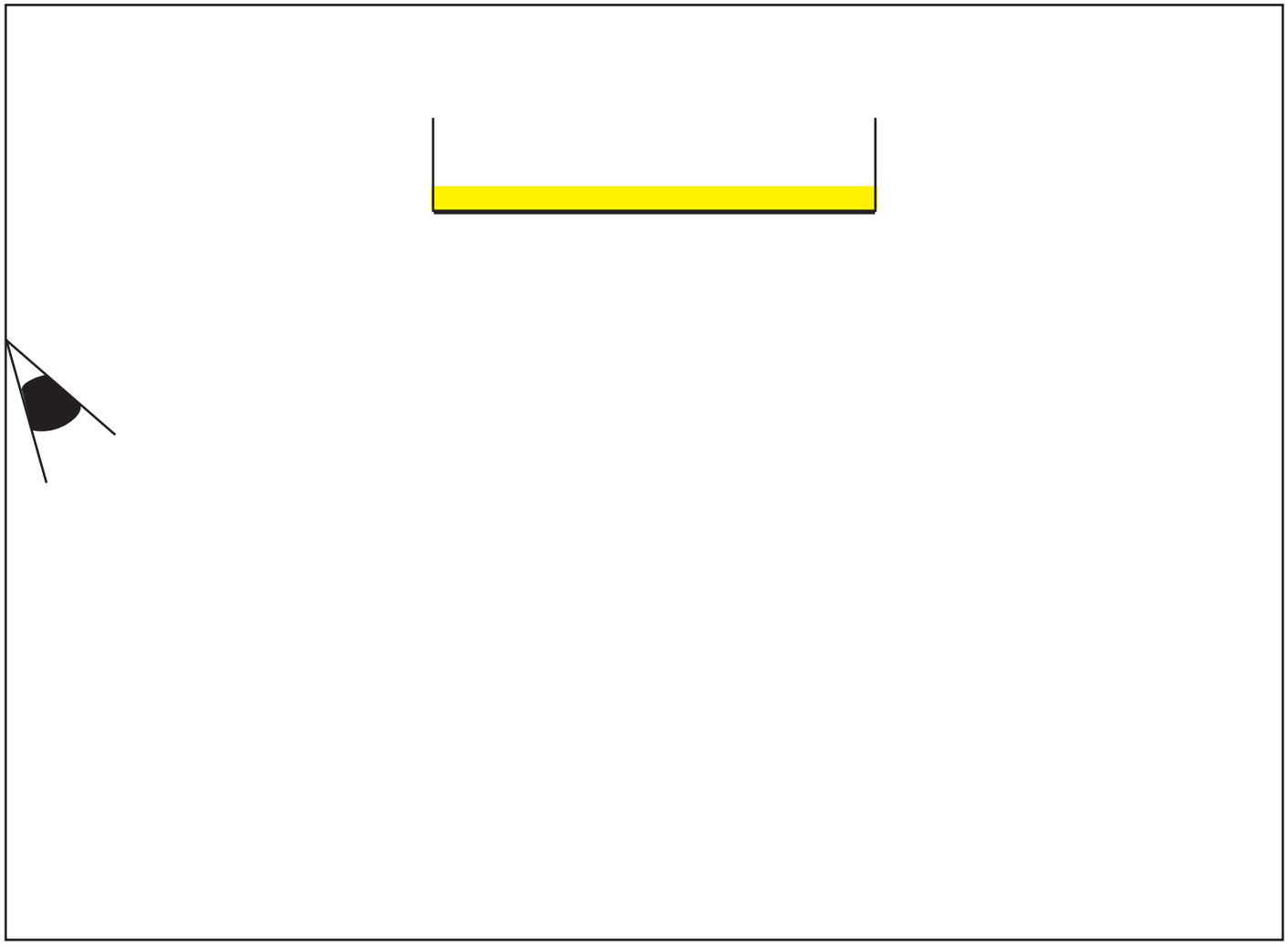
Multiple Importance Sampling

- Idea: Sample from multiple distribution and at least one will provide a good approximation to the integrand in a certain sub-domain

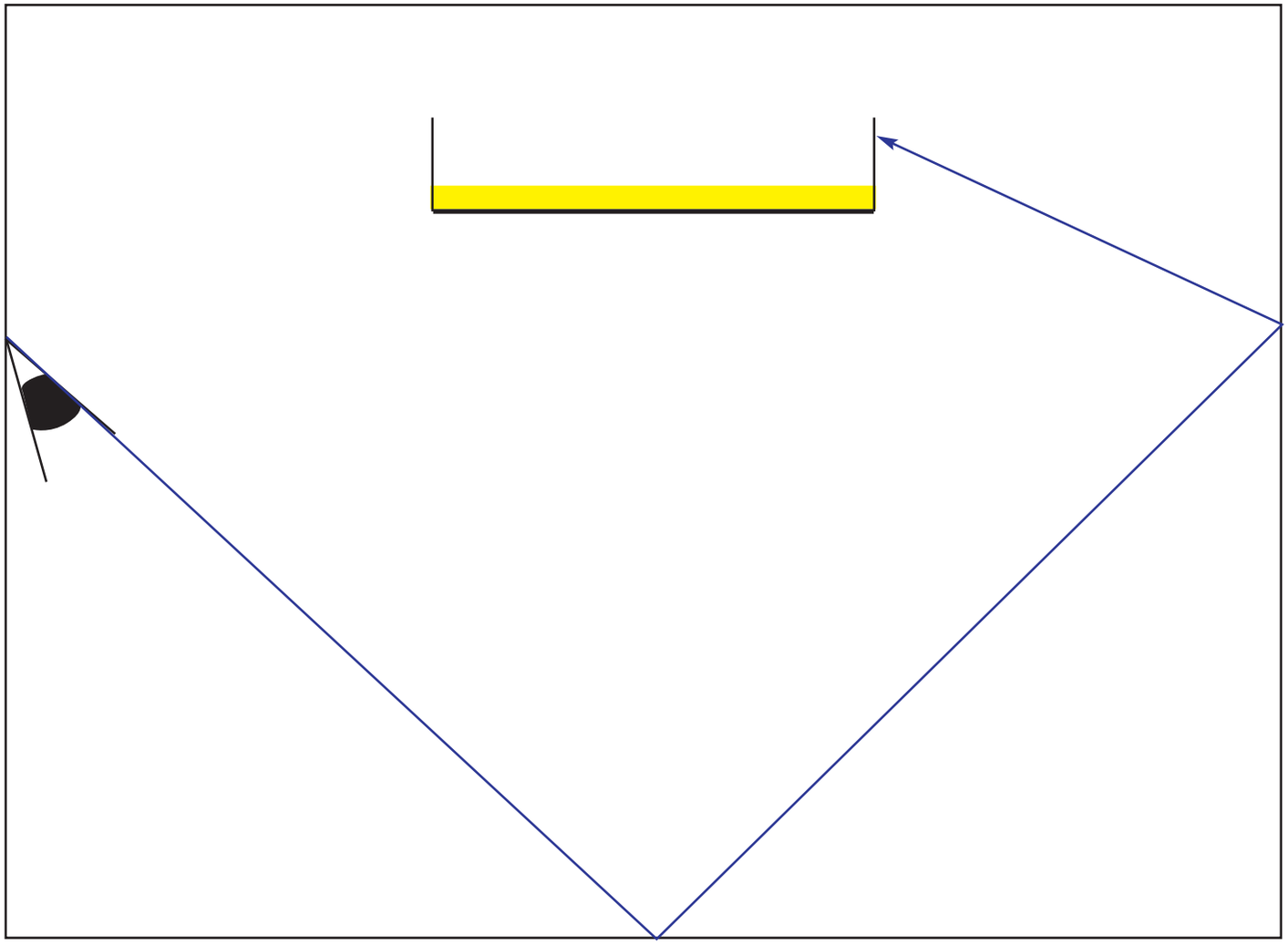
$$\frac{1}{n_f + n_g}\left(\sum_{i=1}^{n_f}\frac{f(X_i)g(X_i)w_f(X_i)}{p_f(X_i)}+\sum_{i=1}^{n_g}\frac{f(Y_i)g(Y_i)w_g(Y_i)}{p_g(Y_i)}\right)$$

- Weight samples from each technique to keep unbiased
- Different weighting of integrand possible
  - Balance Heuristic, Power Heuristic

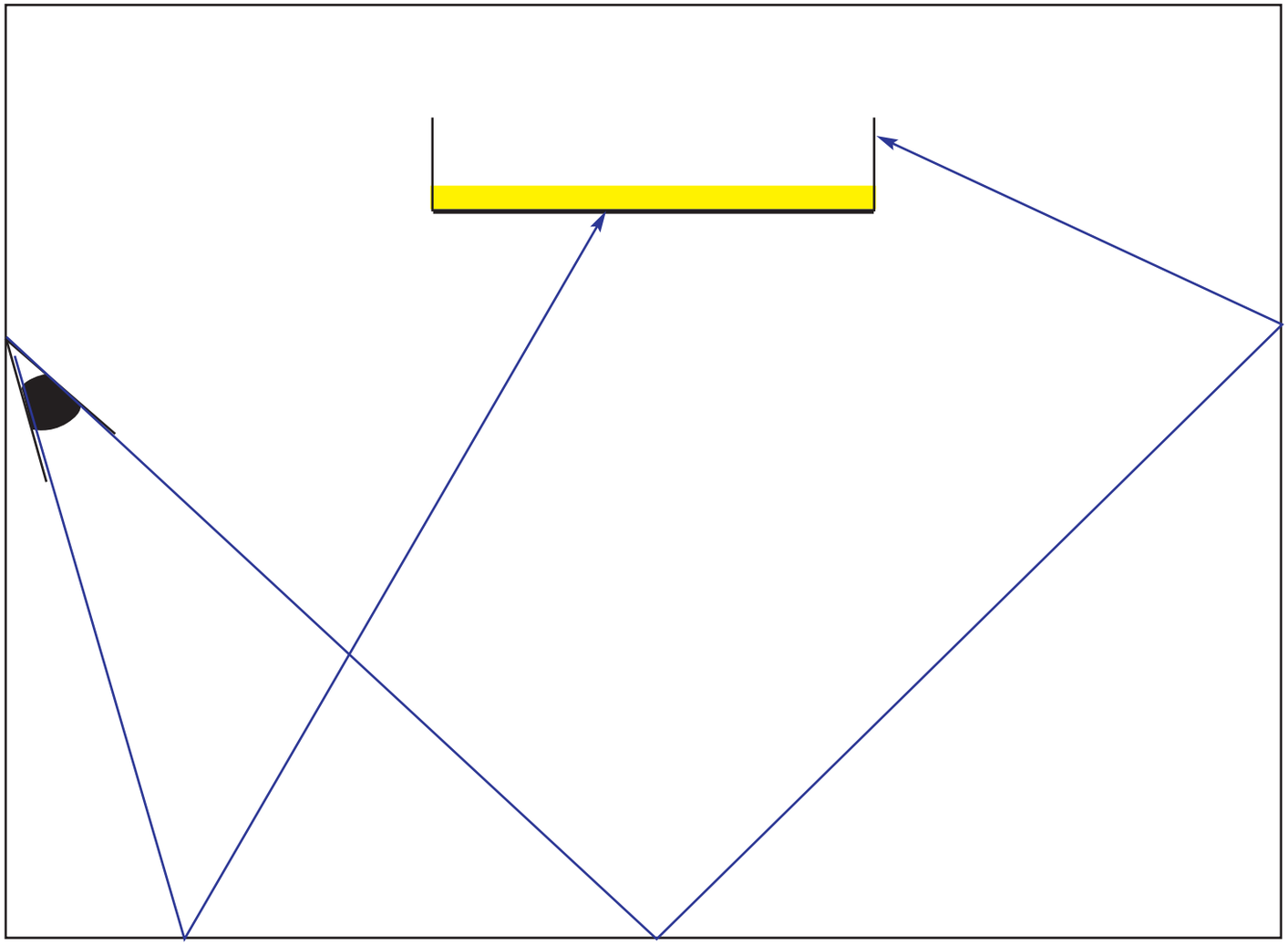
Bi-Directional Path Tracing



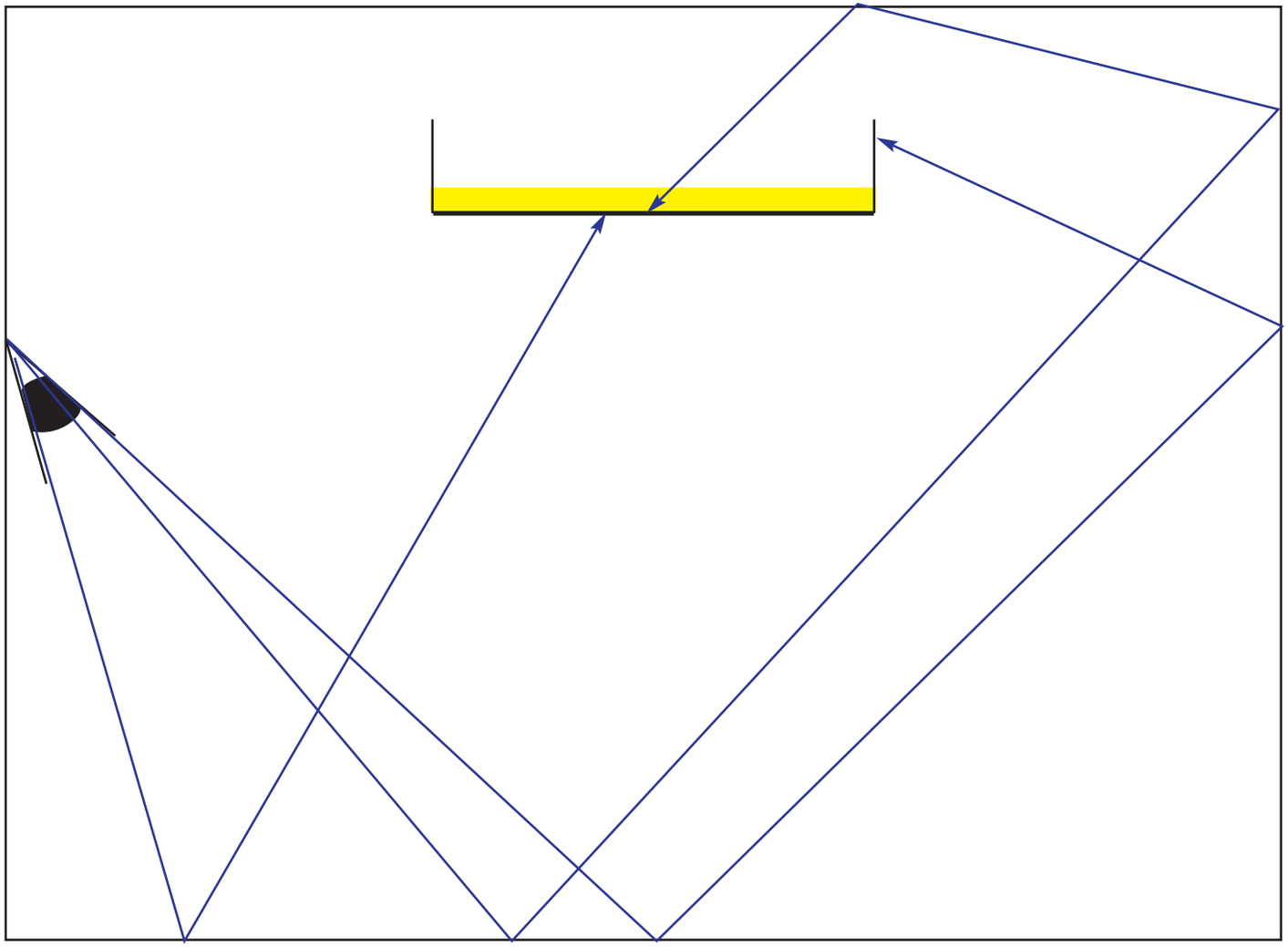
Bi-Directional Path Tracing



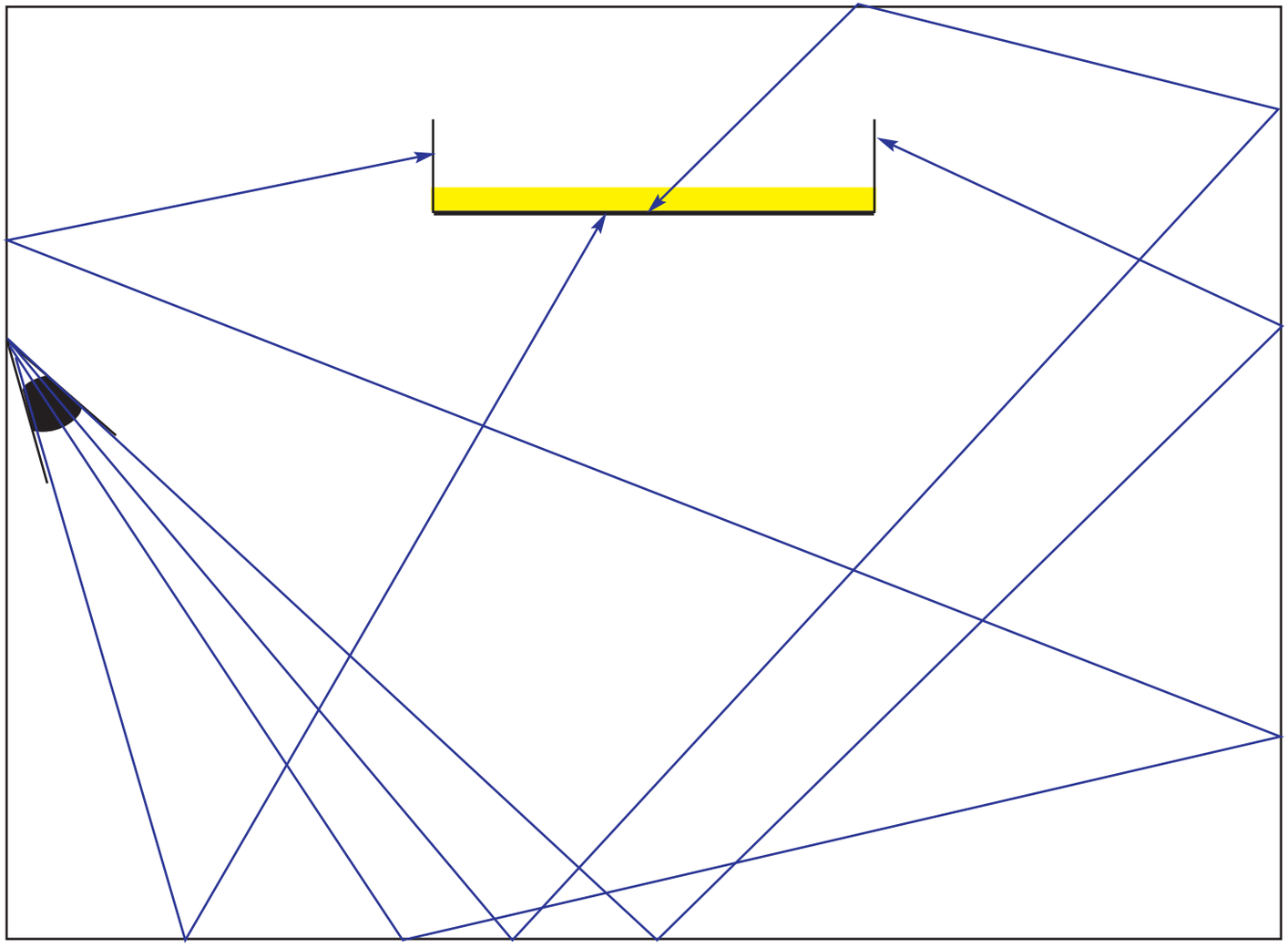
Bi-Directional Path Tracing



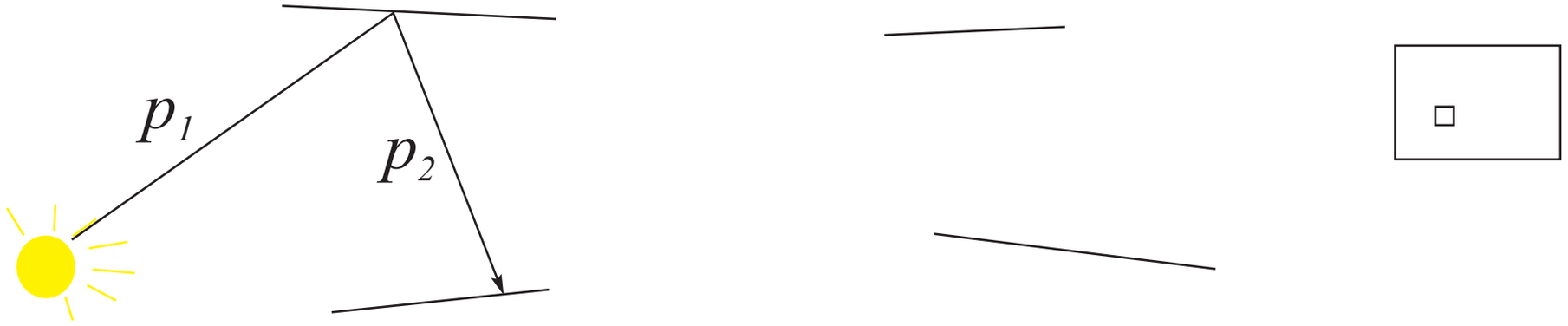
Bi-Directional Path Tracing



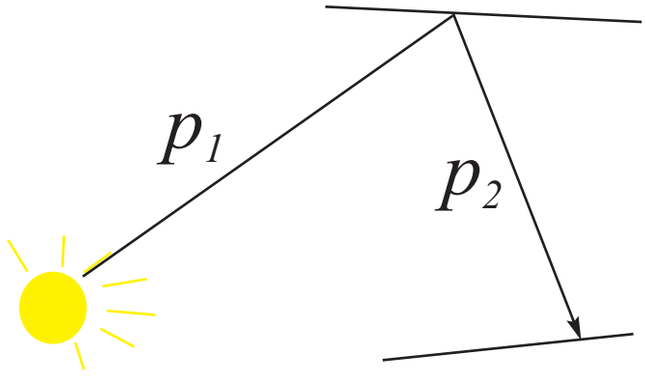
Bi-Directional Path Tracing

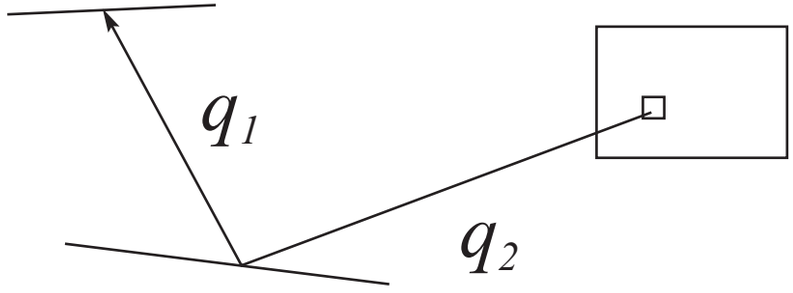


Bi-Directional Path Tracing

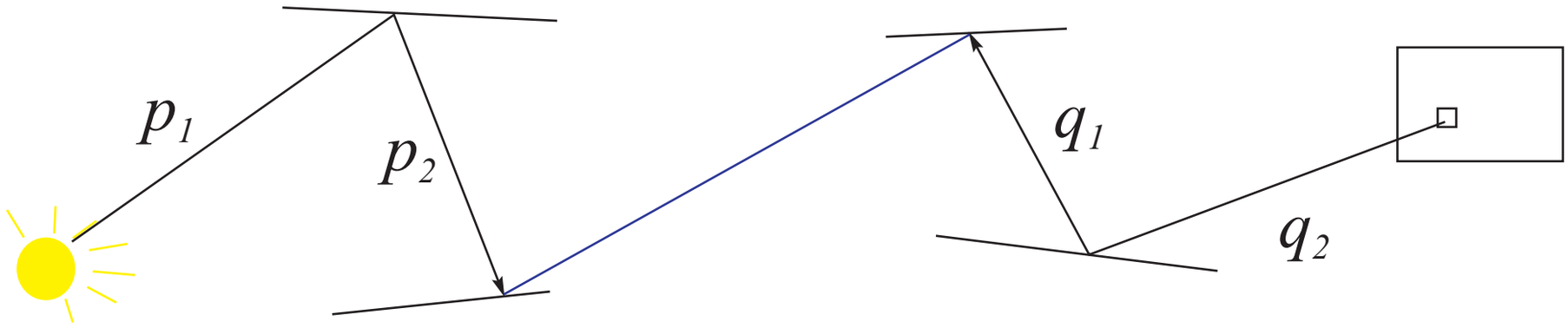


Bi-Directional Path Tracing

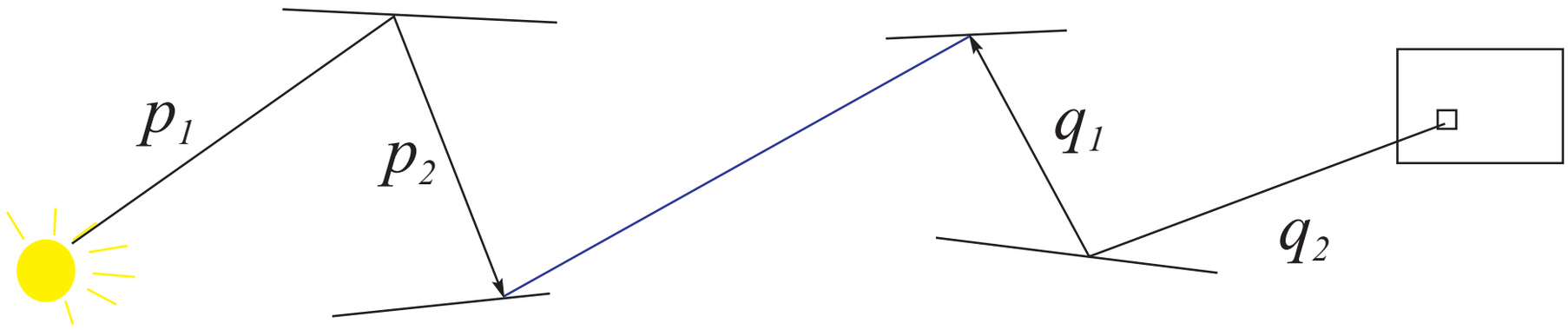




Bi-Directional Path Tracing



Bi-Directional Path Tracing



- Random walk $x = \{x_0, x_1, x_2, \ldots, x_k\}$
  - Generate m samples $p = \{p_1, \ldots, p_{m-1}\}$ from the eye
  - Generate $(k-m)$ samples $q = \{q_1, \ldots, q_{k-m}\}$ from the eye
  - Shoot "shadow" ray to connect sub-paths

Bi-Directional Path Tracing

- Multiple Importance Sampling where each combination of sub-paths is a different sampling strategy
  - All $(k+1)$ possible combination of sub-paths are generated and used
  - Each combination is particularly well suited for specific features
- Optimizations
  - Use optimized direct lighting to sample paths with only one vertex on the "camera path"

Metropolis Light Transport

Metropolis Light Transport

- Based on Metropolis-Hastings algorithm
- Particularly well suited for difficult sampling problems
- Goal: Generate random walk $X_0, X_1, \ldots X_i$ such that $\boldsymbol{X}$ is distributed proportionally to the integrand f
- Estimate of the integrand is given by the number of sample paths per pixel on the image plane
- Costs for a sample are small

Metropolis Light Transport

- X_i is generated by a random change to X_{i-1}
  - X_i is a mutation of X_{i-1}
  - X_i is accepted or rejected with a carefully chosen probability
- Path space is explored locally
  - Regions in path space with a high contribution to the image are sampled densely
  - Ergodicity guarantees that whole path space is sampled

Metropolis Light Transport



Courtesy: [1]

Metropolis Light Transport

- Transition function K determines the probability that $X_i = Y$ given $X_{i-1} = X$

$$\int_{\Omega} K(Y \to X)\, dX = 1, \qquad \forall X$$

- Each X_i is a random variable, its p.d.f. is $p_i(X_i)$

$$p_i(X) = \int_{\Omega} p_{i-1}(Y) \cdot K(Y \to X)\, dY$$

- p_i converge to a unique p.d.f. p^* when i goes to infinity
  - Independent of the starting point of the random walk

Metropolis Light Transport

- Detailed Balance
  - Equilibrium state of a physical system
  - The transition probability between all pairs of states is identical

Detailed Balance

- Detailed Balance

$$f(x)T(y \mid x)a(y \mid x) = f(y)T(x \mid y)a(x \mid y)$$

- Is $p^* \sim f$ in detailed balance?

$$p_i(x) = \int_\Omega p_{i-1}(x)K(y \mid x)dy$$

$$p_i(x) = \left[1 - \int_\Omega a(y \mid x)T(y \mid x)dy\right] p_{i-1}(x) + \int_\Omega a(x \mid y)T(x \mid y)p_{i-1}(y)dy$$

$$p_i(x) = p_{i-1}(x) + \int_\Omega \left(a(x \mid y)T(x \mid y)p_{i-1}(y) - a(y \mid x)T(y \mid x)(x)\right)dy$$

$$p_i(x) = p_{i-1}(x)$$

State Transition

- Construct transition function K so that it converges to the equilibrium state of the light transport in a scene as fast as possible

State Transition

- Construct transition function K so that it converges to the equilibrium state of the light transport in a scene as fast as possible
- Create mutations according to a tentative transition function $T(x \to y)$ with $x'_i = y$
  - Tentative transition function can be chosen arbitrarily
- The tentative samples y_i are either accepted or rejected according to an acceptance probability

$$a(x \to y) \Rightarrow x_i = \begin{cases} x'_i & \text{with } a(x_{i-1} \to x'_i) \\ x_{i-1} & \text{otherwise} \end{cases}$$

Acceptance Function

- Acceptance probability
  - Make acceptance probabilities as large as possible

$$a(y \mid x) = \min\left\{1, \frac{f(y) \cdot T(x \mid y)}{f(x) \cdot T(y \mid x)}\right\}$$

  - Note: function value has to be computed before a sample is accepted or rejected

Metropolis Light Transport

- Monte Carlo estimate for a pixel value

$$m_j = E\left[\frac{1}{N}\sum_{i=1}^{N}\frac{R_j(X_i)f(X_i)}{p(X_i)}\right]$$

  - R is pixel filter kernel

- Assume $p^* = (1/b)\cdot f$, where b is a normalization constant

$$m_j = E\left[\frac{1}{N}\sum_{i=1}^{N} b\cdot R_j(X_i)\right]$$

Problems

- Metropolis sampling can be used to sample with

$$p^* = (1/b) \cdot f$$

- Startup Bias: The samples X_i have the desired distribution p^* only if i goes to infinity?
- How to choose normalization constant b?

Metropolis Light Transport

- Avoid startup bias
  - Generate a random initial state X_0 with some existing sampling technique, e.g. path tracing
  - Sample has p_0 instead of p^*
  - Unbiased estimator is

$$m_j = E\left[\frac{1}{N}\sum_{i=1}^{N} W_i \cdot R_j(X_i)\right] \quad \text{with} \quad W_i = W_0 = \frac{f(X_0)}{p(X_0)}$$

Metropolis Light Transport

- Normalization constant b
  - Corresponds to image brightness
  - Metropolis algorithm only determines relative pixel intensities
  - Generate n samples with some existing path tracing technique and use average for b

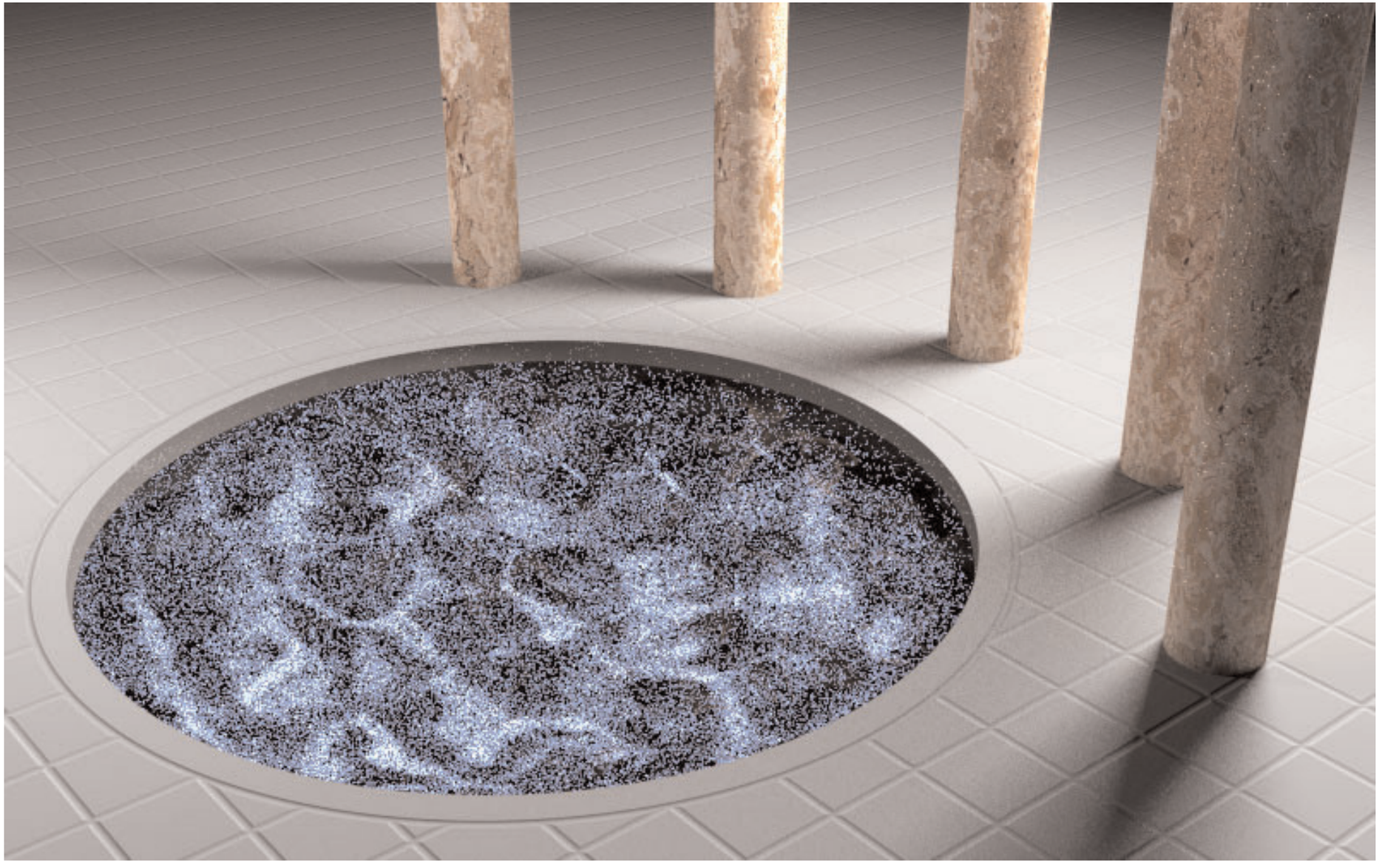
Mutation Strategies

- Can have arbitrary form
- Typical: Move / Add / Delete / Replace small number of vertices of the existing path
- Mutation strategies can be constructed for sampling special phenomena
- In general, all accepted paths are weighted equally

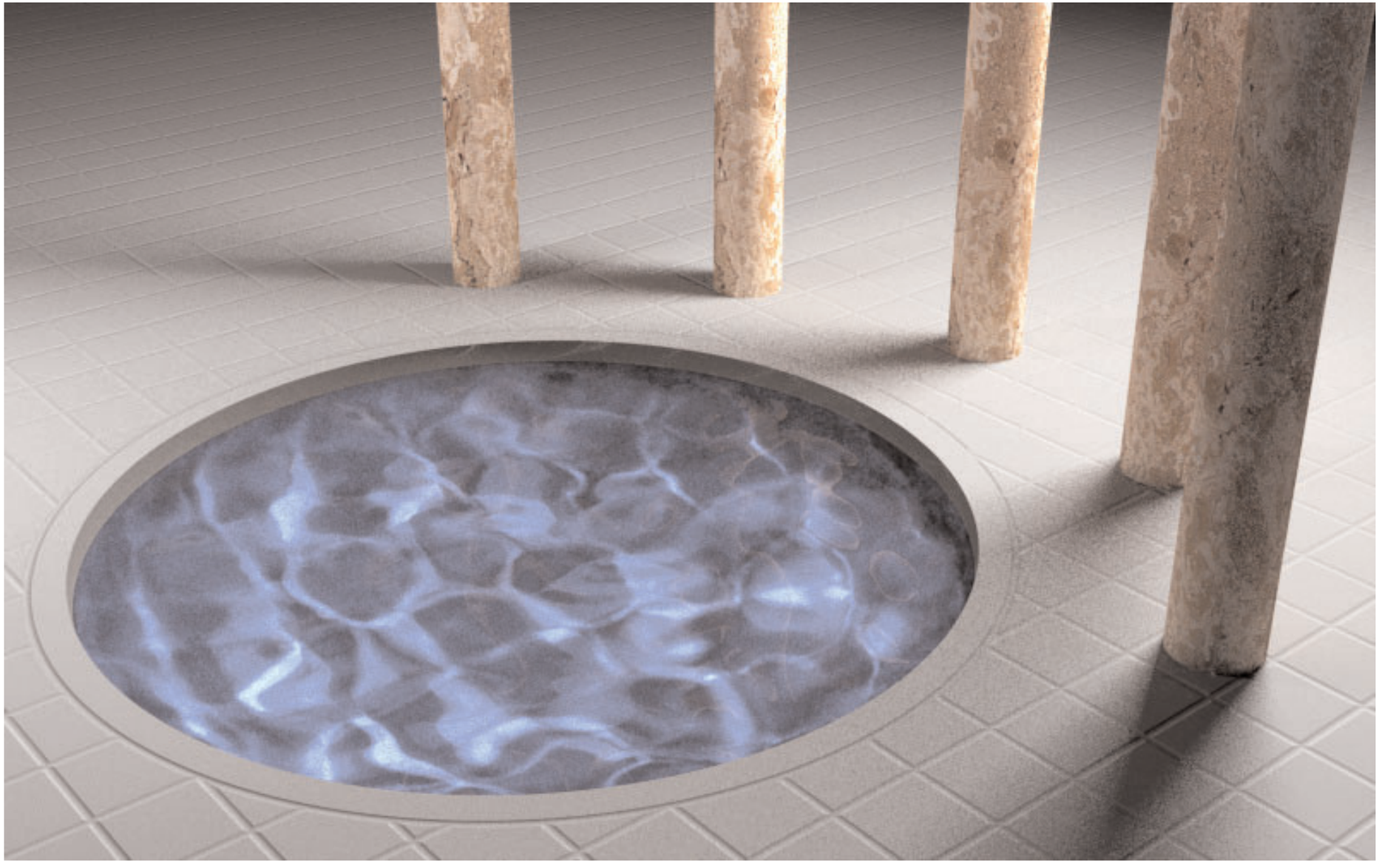
Mutation Strategies

- Bidirectional Mutations
  - Replace sub-path of existing path
  - Similar to bidirectional path tracing
- Perturbations
  - Slight modifications of vertices of existing path
  - Caustic perturbation
  - Lens sub-path mutations
  - ...

Metropolis Light Transport



Metropolis Light Transport



Biased Techniques

Problems Unbiased Techniques

- High frequency noise (high variance) strongly visible for human visual system
- Techniques discussed so far are very computationally expensive
  - Not acceptable even for many offline application
- Slow convergence

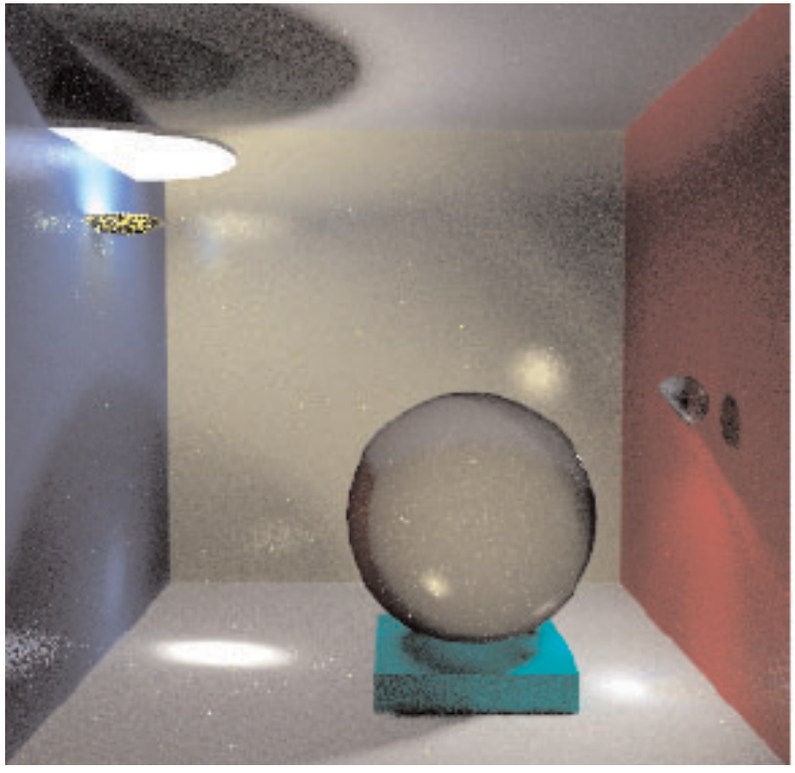
Filtering

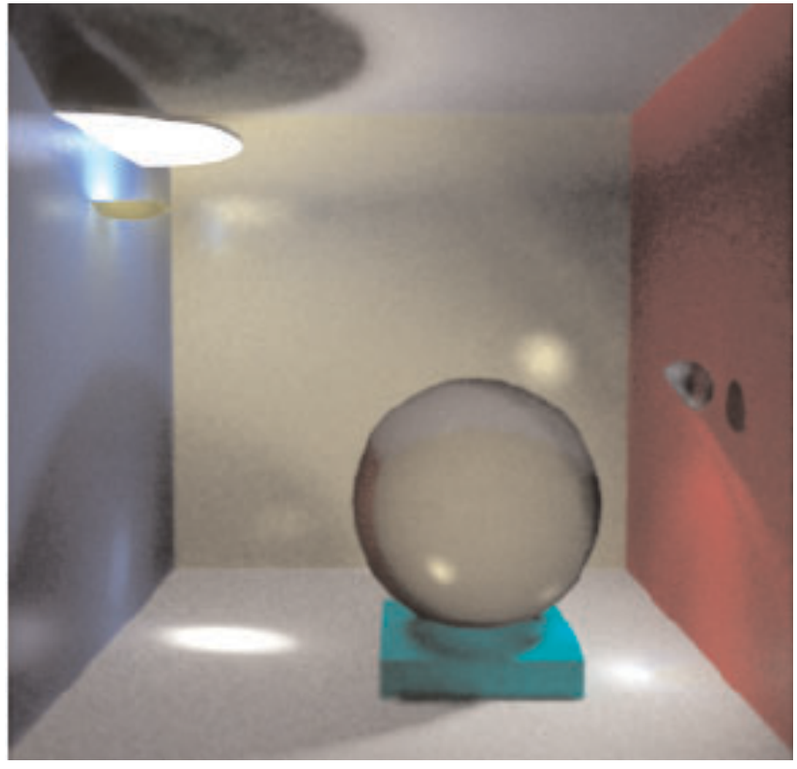
- Remove noise by post-processing the image or the samples
  - Keep high-frequency features, e.g. edges and shadows, in the image
  - Keep energy

Filtering

- Remove noise by post-processing the image or the samples
  - Keep high-frequency features, e.g. edges and shadows, in the image
  - Keep energy
- Re-distribute / diffuse energy over multiple pixels to avoid spikes in the variance
- Re-distribute power of samples over image plane
  - Progressive refinement starting from blurred image

Filtering





Courtesy: Adaptive Filtering for Progressive Monte Carlo Image Rendering, Frank Suykens, Yves D. Willems

Adaptive Sampling

- Use sampling density based on estimated variance

$$V \approx \frac{1}{N-1}\left[\frac{1}{N}\sum_{i=1}^{N} L_i^{\,2} - \left(\frac{1}{N}\sum_{i=1}^{N} L_i\right)^2\right]$$

- Use N_i samples in pixel p_i
  - Increases efficiency by sampling only those pixels densely which require many samples
- Bias introduced usually very small
  - Pixels where an edge of a light source passes through might have significant bias

Irradiance Caching

- Assumption:
  - Irradiance field is smooth
  - Most surfaces in the scene are diffuse or other techniques are used to compute non-diffuse components of the BRDFs
- Visual artifacts if smoothness assumption not true
  - Caustics hardly possible at all
- Global bias which is hard to control

Irradiance Caching

- Cache irradiance values at discrete points
- Compute new irradiance values by interpolating between cached values
  - Use error estimate to determine interpolation error
- Compute and cache values if estimated error is too high
- Distances, normals, gradients are used to estimate the error

Photon Mapping

- Assumptions
  - Irradiance field is mostly smooth but contains peaks, e.g. caustics
  - The eye and the light are the sources of the most important light paths in the scene
- Two-pass algorithm
  1. Building photon map
  2. Rendering with photon map as static data structure used to look up illumination in the scene

Photon Map Building

- Trace photons emitted from the light sources
  - Power distribution of light source determines emittance distribution of the photons
- Photons are stored at diffuse surfaces (or surfaces with a diffuse component)
- Russian Roulette: At each diffuse surface, random decision if photon is scattered at the surface or absorbed

Photon Mapping

- Photon map represents incoming flux
- Exitant radiance $L_r(x,\omega)$ can be estimated using the nearest n photons around x
  - n nearest neighbor density estimate

$$L_r(x,\omega) = \int_\Omega f_r(x,\omega_i,\omega)\frac{d^2\Phi_i(x,\omega_i)}{dA_i}$$

$$L_r(x,\omega) \approx \sum_{k=1}^{N} \int_\Omega f_r\left(x,\omega_i^{(k)},\omega\right)\frac{\Delta\Phi_i^{(k)}\left(x,\omega_i^{(k)}\right)}{\Delta A}$$

Rendering

- Combination of different techniques to render image
- Recursive ray tracing is used for specular components
- Monte Carlo ray tracing with photon map is used for diffuse surfaces
  - Photon Map can be used for important sampling
- Final Gathering can be used to improve the quality
- Direct lighting: Often most efficient to sample light sources directly on the fly

Rendering

- Final Gathering
  - Instead of directly querying the photon map at an intersection point, rays are shot according to the BRDF at the intersection point
  - The photon map is sampled at the intersection points of these rays
  - Generating rays according to the distribution of incoming photons is an alternative, multiple importance sampling can be used to combine the approaches

Photon Mapping: Bias

- Building the photon map unbiased
- Approximating the exitant radiance introduces bias
  - Nearest neighbors only approximate illumination at x
  - Density is estimated using the nearest neighbors
- Bias is consistent, i.e. it vanishes when the number of photons used goes to infinity
- Bias is purely local
- Error hard to estimate

Conclusion

Monte Carlo Techniques for Realistic Image Synthesis

- Physically motivated
- Most accurate algorithm so far
- Can sample (almost) all possible light paths
- Far from real time
  - For complex measurement functions and difficult lighting environments still several days
- Mostly Photon Mapping and Quasi Monte Carlo techniques used in commercial rendering systems

Conclusions



Conclusions



S.T.A.L.K.E.R.



The Courtyard House, Henrik Wann Jensen

Monte Carlo Techniques for Realistic Image Synthesis?

- Where do we really need the physical accuracy?
  - Is it worth to research these ideas further?
- Slow performance through inefficient implementations?
- Unbiased Monte Carlo Algorithms are a straight forward application of algorithms developed for physical simulation.
  - Can the application of more domain knowledge lead to more efficient (unbiased) algorithms?

Literature

[1] *Robust Monte Carlo Methods for Light Transport Simulation*, Eric Veach, Ph.D. dissertation, Stanford University, December, 1997

[2] *State of the art in Monte Carlo global illumination*, Philip Dutré, Henrik Wann Jensen, Jim Arvo, Kavita Bala, Philippe Bekaert, Steve Marschner, Matt Pharr, SIGGRAPH 2004 course notes, August 2004

[3] *State of the Art in Monte Carlo Ray Tracing for Realistic Image Synthesis*, Henrik Wann Jensen, Jim Arvo, Marcos Fajardo, Pat Hanrahan, Don Mitchell, Matt Pharr, and Peter Shirley, SIGGRAPH 2001 course notes, August 2001

[4] *Physically Based Rendering: From Theory to Implementation*, Matt Pharr and Greg Humphreys, Morgan Kaufmann, July 2004

[5] *Mathematical Methods for Computer Graphics*, CS 448 - Topics in Computer Graphics, Lecture Notes, Stanford University, 1997

[6] *Strictly Deterministic Sampling Methods in Computer Graphics,* mental images technical report, 2001 (published in Monte Carlo Ray Tracing, SIGGRAPH 2003 course notes, July 2003)

[7] *The quiet revolution in interactive rendering*, Matt Pharr, Stanford EE Computer Systems Colloquium, November 2005

[8] *Mathematical Models and Monte Carlo Algorithms for Physically Based Rendering*, Eric Lafortune, Ph.D. thesis, Department of Computer Science, Katholieke Universitiet Leuven. Leuven, Belgium, February 1996.

Slides available under

www.dgp.toronto.edu/~lessig/ais/mc.pdf