Fundamentals of Photorealistic Rendering (Path Tracing and Photon Mapping)

Lecturer: Bochang Moon
GIST

Computer Graphics Laboratory

Image Synthesis

- Pixel intensity at (x, y)
  - $I(x, y) = \int \int \int \int \int f(x - x', y - y')\, L(x', y', u', v', t')\, dt'du'dv'dx'dy'$
- Notes
  - Pixel is a point (not an area): requires a pixel reconstruction filter
- $f(x - x', y - y')$
  - Pixel reconstruction filter with a small width (e.g., box filter)
- x', y': 2D random samples on the image plane
- u', v': 2D random samples on the lens (for depth-of-fields)
- t': 1D time sample (for motion blur)
- The 5D samples define a 5D camera sample, and thus a primary ray

Image Synthesis

- Pixel intensity at (x, y) can be represented into:
  - $\mathrm{I(x,y)} = \int \int \int \int \int \mathrm{f(x-x', y-y')\, L(x', y', u', v', t')\, dt'du'dv'dx'dy'}$

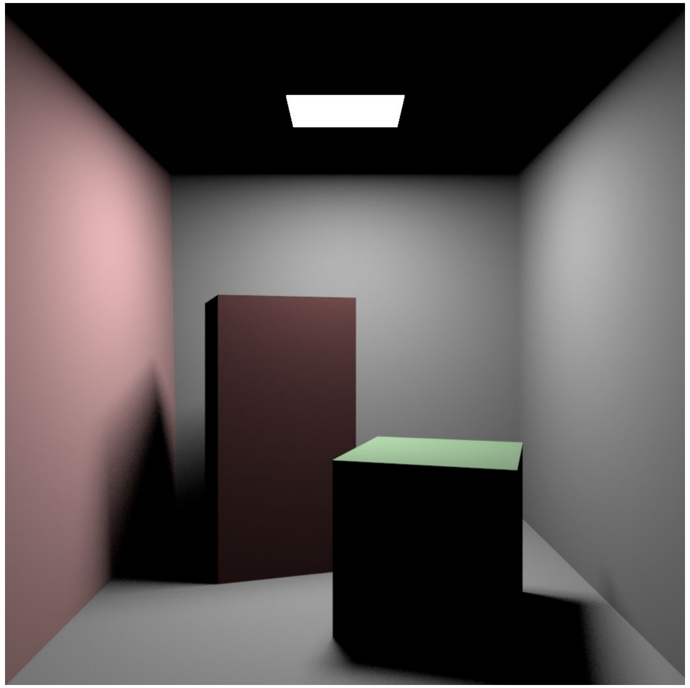
- The 5D samples define a 5D camera sample and thus a primary ray
  - 1) Find an surface point x via ray tracing
    - Determine the x, k_o
  - 2) Solve the rendering equation at x
    - $L_s(k_o) = L_e(k_o) + \int_{all\ k_i} \rho(k_i, k_o) L_f(k_i) cos\theta_i d\sigma_i$

Outline

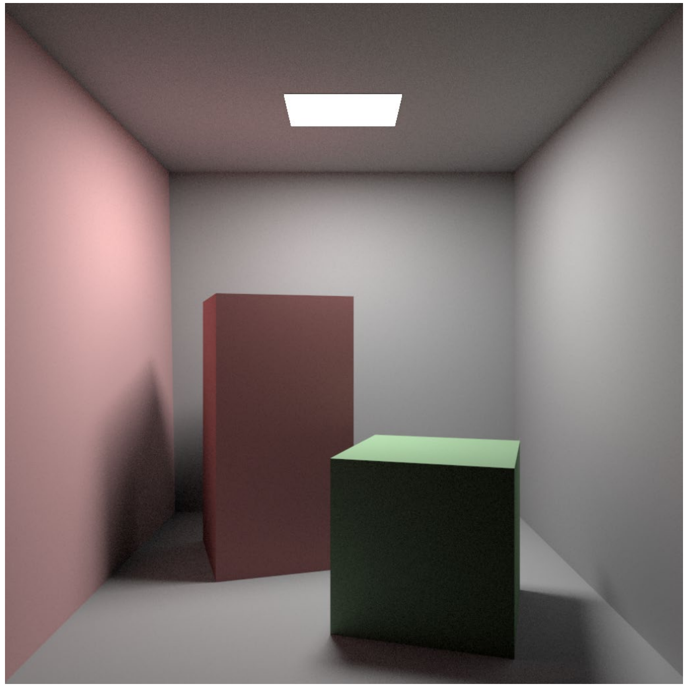
- Path Tracing
- Path Tracing with Next Event Estimation
- Background: Density Estimation
- Photon Mapping
- Progressive Photon Mapping

Global Illumination

- Global illumination methods consider both direct and indirect lighting



Without indirect lighting



With indirect lighting

Path Tracing

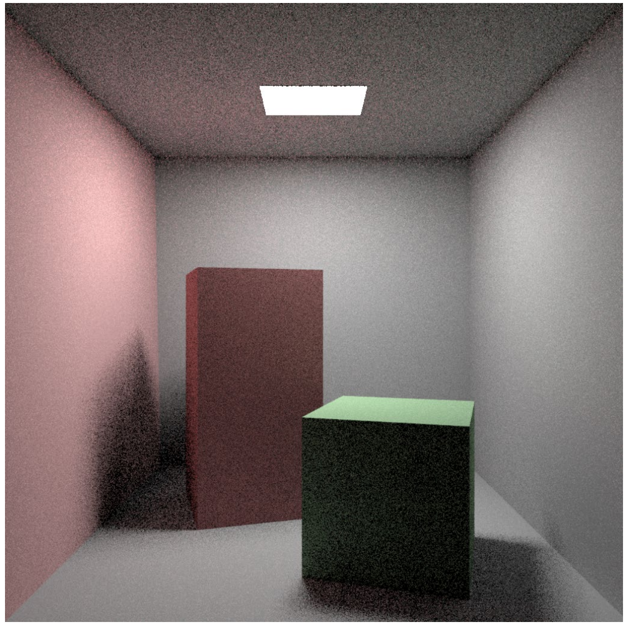
- $L_s(k_o) = L_e(k_o) + \int_{all\ k_i} \rho(k_i, k_o) L_f(k_i) cos\theta_i d\sigma_i$
- Monte Carlo integration
  - $\int_{x \in S} g(x) d\mu \approx \frac{1}{N} \sum_{i=1}^{N} \frac{g(x_i)}{p(x_i)}$
  - When N=1,
  - $L_s(k_o) \approx L_e(k_o) + \frac{\rho(k_i, k_o) L_f(k_i) cos\theta_i}{p(k_i)}$
  - Need to do:
    - Select a random direction k_i
    - Evaluate $L_f(k_i)$

Path Tracing

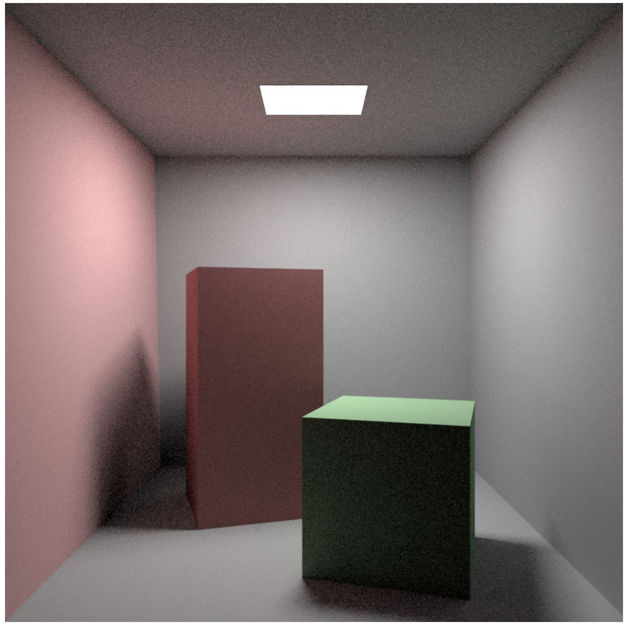
- $L_s(k_o) \approx L_e(k_o) + \frac{\rho(k_i,k_o)L_f(k_i)cos\theta_i d\sigma_i}{p(k_i)}$
- In case of the ideal diffuse surface:
    - $\rho = \frac{R}{\pi}$
    - When we choose a density function $p(k_i) = \frac{cos\theta_i}{\pi}$
        - $L_s(k_o) \approx L_e(k_o) + RL_f(k_i)$
        - Note that we can cancel out the cosign terms

Path Tracing

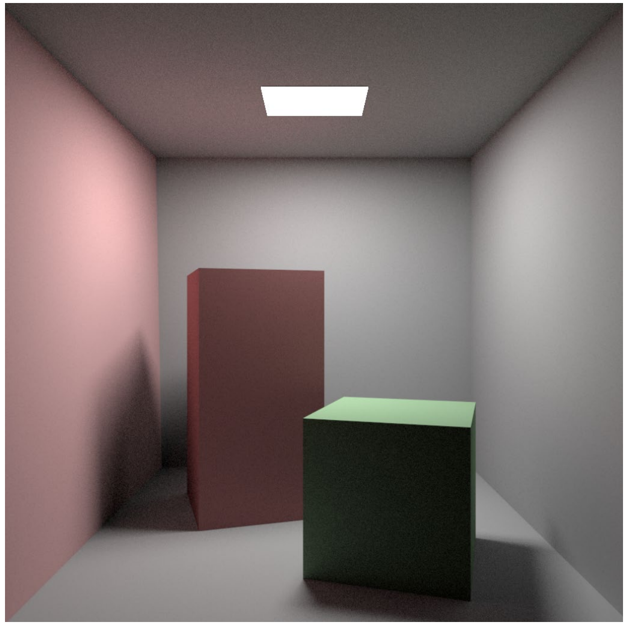
- A general rendering method that solves the full light transport equation (i.e., rendering equation)
- For each pixel color, it makes multiple ray paths, then averages the colors from the ray paths



4 samples / pixel (1.25 secs)



16 samples / pixel (5 secs)



64 samples / pixel (20 secs)

Outline

- Path Tracing
- Path Tracing with Next Event Estimation
- Background: Density Estimation
- Photon Mapping
- Progressive Photon Mapping

Problems in Naïve Path Tracing

- Issues: Hard to find a light path (very high variance)
  - i.e., the probability of hitting luminaries with small sizes is very low.

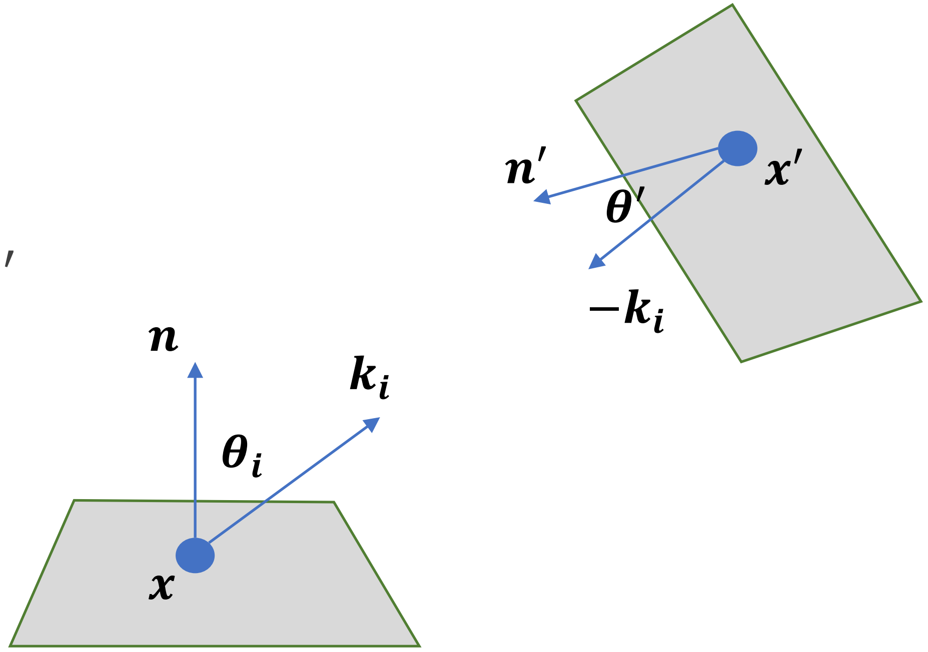
- A common practice for path tracing
  - Path tracing with direct lighting (sometimes referred to as path tracing with next event estimation)

Path Tracing with Direct Lighting

- $\mathrm{L_s}(x, k_o) = \mathrm{L_s^D}(x, k_o) + \mathrm{L_s^I}(x, k_o)$
  - direct lighting $\mathrm{L_s^D}(x, k_o)$ + indirect lighting $\mathrm{L_s^I}(x, k_o)$

- $L_s^{\mathrm{D}}(k_o) = \int_{all\ x' \text{ in luminaries}} \frac{\rho(k_i,k_o)L_{\mathrm{e}}(x',x-x')v(x,x')cos\theta_i cos\theta\prime}{||x-x'||^2} dA'$
  - L_{e}: emitted radiance

- $L_s^{\mathrm{I}}(k_o) = \int_{all\ k_i} \rho(k_i, k_o) L_f^{\mathrm{R}}(k_i) cos\theta_i d\sigma_i$
  - $L_f^{\mathrm{R}}(k_i)$: field radiance (only reflected, i.e., not from luminaries)



Path Tracing with Direct Lighting

- $L_s^{\mathrm{D}}(x, k_o) = \int_{all\ x'} \frac{\rho(k_i,k_o)L_e(x',-k_i)v(x,x')cos\theta_i cos\theta\prime}{||x-x'||^2} dA'$

- Sample a point x' on a luminaire with density function p ($x' \sim p$)

- $L_s^{\mathrm{D}}(x, k_o) \approx \frac{\rho(k_i,k_o)L_e(x',-k_i)v(x,x')cos\theta_i cos\theta\prime}{p(x')\ \|\mathrm{x} - \mathrm{x}'\|^2}$

- Pick a uniform random point x' from the luminaire
  - $p = \frac{1}{A}$ (A is the area of the luminaire)
  - $L_s^{\mathrm{D}}(x, k_o) \approx \frac{\rho(k_i,k_o)L_e(x',-k_i)v(x,x')cos\theta_i cos\theta' A}{\|\mathrm{x} - \mathrm{x}'\|^2}$

Path Tracing with Direct Lighting

- Other details:
  - Only applied when the surface point is non-specular.
  - e.g., narrow BRDFs
    - Sampling a point on luminaries is not effective
  - Uses shadow rays for checking occlusions between the surface point and sampled light point Shadow rays
    - Usually faster than finding the first intersection
  - Samples a point on luminaries can be done via inverse transform sampling when the shapes of luminaries are simple (e.g., spheres, triangles)

Path Tracing with Direct Lighting

- $L_s^{\mathrm{D}}(x, k_o) \approx \dfrac{\rho(k_i,k_o)L_e(x',-k_i)v(x,x')cos\theta_i cos\theta\prime}{p(x')\ \|\mathrm{x} - \mathrm{x}'\|^2}$

- Multiple light sources are given:
  - Generate one shadow ray per each light source, but this is not practical with many light sources
  - A common choice is to pick only a point x' and generate a shadow ray towards x'

Path Tracing with Direct Lighting

- $L_s^{\mathrm{D}}(x, k_o) \approx \dfrac{\rho(k_i, k_o) L_e(x', -k_i) v(x, x') cos\theta_i cos\theta\prime}{p(x') \, \|\mathrm{x} - \mathrm{x}'\|^2}$

- Determining $p(x')$ requires the following:
  - $p(\mathrm{x}') = \mathrm{p(l)p(x'|l)}$
    - Probability of selecting a luminary l: $\mathrm{p(l)}$
    - Probability of sampling a point on the chosen light: $\mathrm{p(x'|l)}$
  - How to select a light source l:
    - Uniform: the probability of selecting a light is equal.
    - Spatial: set the probability proportional to the light power (assume that all lights are visible against a point)
    - Visibility-aware selection: requires an estimation of visibility across surface points
    - Light clustering is a well-known approach for many lights, e.g., thousands of lights

Path Tracing with Direct Lighting

- Additional discussion:
  - Sampling a point on luminaries is often effective for Lambertian surfaces but not very effective on highly glossy surfaces
    - BRDF sampling for direct lighting can be better for such cases
    - How to combine Light sampling and BRDF sampling?
      - Multiple importance sampling (MIS) provides a solution; this will be covered later
  - When is direct lighting effective?
    - Intuitively, it is effective when lights are visible from most surfaces
    - A counter example

Image from https://benedikt-bitterli.me/resources/

Outline

- Path Tracing
- Path Tracing with Next Event Estimation
- Background: Density Estimation
- Photon Mapping
- Progressive Photon Mapping

Density Estimation

- $P(a < X < b) = \int_a^b f(x)dx$ for all a < b

- Problem:
  - Input: a set of observed data points generated from an unknown probability density function (pdf)
  - Output: an estimate of the pdf

- A parametric approach:
  - We assume the pdf is a known parametric family of distributions
  - e.g., if we assume the samples are drawn from a normal distribution, we can estimate the mean μ and variance σ^2 with a sample mean and sample variance

Density Estimation

- In practice,
  - It is hard to employ the parametric approach as our target function (e.g., radiance) may be very complex

- Nonparametric approaches:
  - Histograms
  - Kernel estimator
  - Nearest neighbor method

Histograms

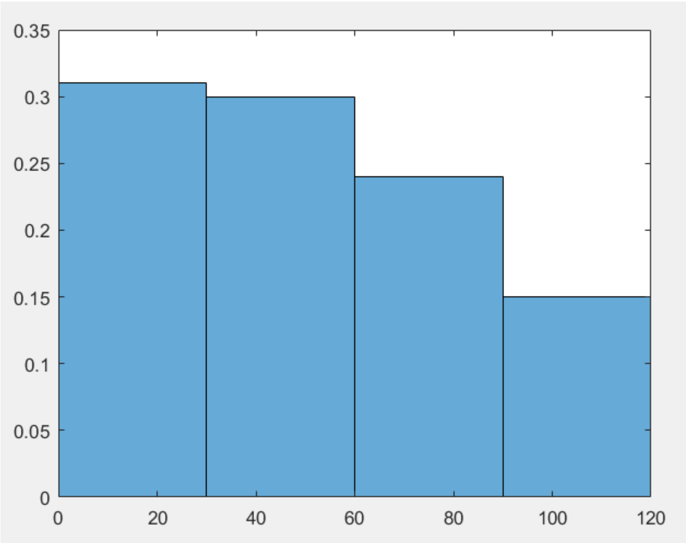
- Input: *n* real data $X_1, \ldots, X_n$
- Given an origin x_0 and a bin width *h,* the bins of histograms are defined by the intervals $[x_0 + mh, x_0 + (m+1)h)$, where m is an integer
- $\hat{f}(x) = \frac{\#\ of\ X_i\ in\ same\ bin\ as\ x}{nh}$

Data

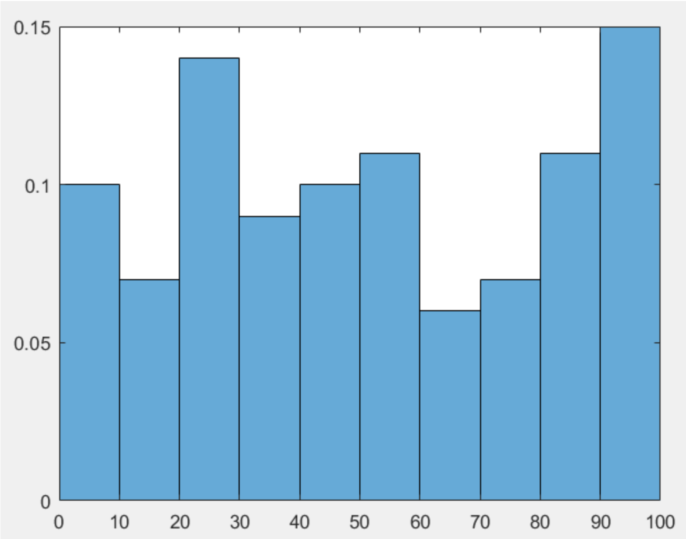
- Example: 100 random samples from a unknown pdf

75	95	32	96	97	25	5	20	47	95
34	10	99	39	39	21	20	49	26	58
48	93	15	55	97	80	14	26	70	85
86	84	68	8	78	17	86	85	55	74
82	52	88	90	4	46	46	46	11	92
56	42	27	70	31	29	26	50	25	7
86	83	39	21	30	61	76	87	90	50
67	17	1	23	50	93	38	42	8	32
9	42	23	8	67	62	53	94	15	90
4	55	4	49	99	99	55	68	25	73

Histograms



h=3
0



h=1
0

Histograms

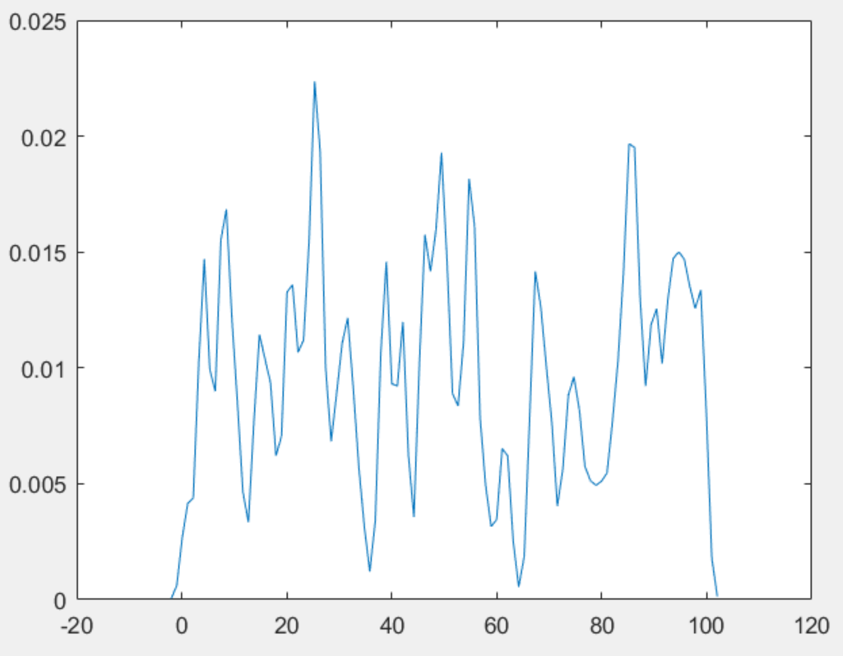
- It discretizes the density function
- The parameter h controls the amount of smoothing
- The number of bins grows exponentially as the dimensionality of the data increases
- The density function has discontinuities at the bin boundaries

Kernel Estimator

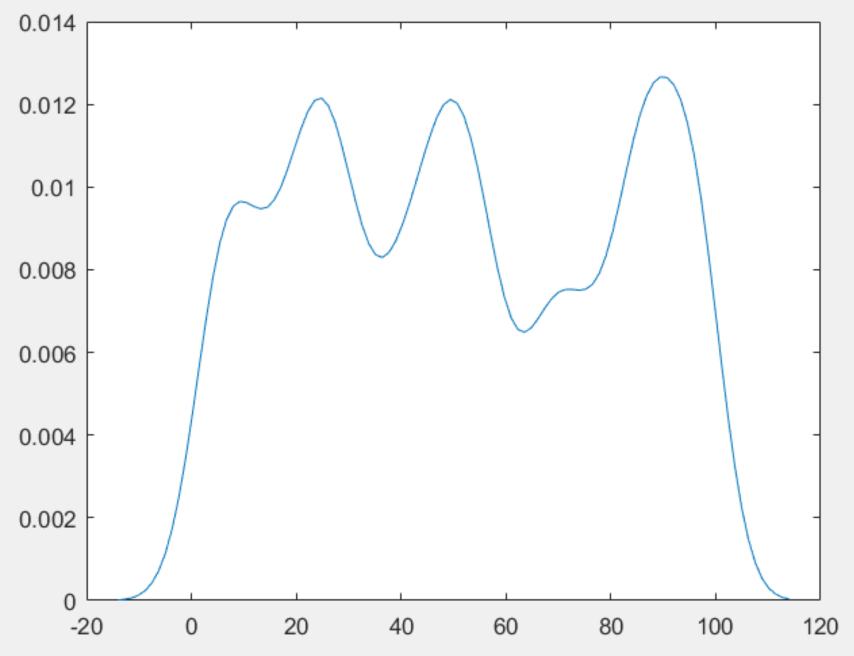
- $\hat{f}(x) = \frac{1}{nh}\sum_{i=1}^{n} K\left(\frac{x-X_i}{h}\right)$
  - h: window width (smoothing parameter, bandwidth)
  - *K is a kernel function*
    - $\int_{-\infty}^{\infty} K(x)dx \ = 1$

  - Intuitively,
    - Place a bump at each observation
    - Kernel estimator is a sum of bumps

Kernel Estimator



h=1



h=5

Kernel Estimator

- $\hat{f}(x)$ inherits all the continuity and differentiability properties of K
- e.g., when K is the normal density function, $\hat{f}(x)$ will be a smooth curve
- The parameter h controls the amount of smoothing

Nearest Neighbor Method

- Adapt the amount of smoothing to the local density of data
- The generalized kth nearest neighbor density estimate:
- $\hat{f}(t) = \frac{1}{nd_k(t)} \sum_{i=1}^{n} K\left(\frac{t-X_i}{d_k(t)}\right)$
  - $d_k(t)$: kth nearest distance from the query point, t

Variable Kernel Method

- $\hat{f}(x) = \frac{1}{n}\sum_{i=1}^{n} \frac{1}{hd_{i,k}} K\left(\frac{x - X_i}{hd_{i,k}}\right)$
- $d_{i,k}$: distance from X_i to the kth nearest point
    - Note that the window width is independent of query points
- The window width of the kernel placed on the point X_i is proportional to $d_{i,k}$
- It considers the local density of observed data

General Weight Function

- $\hat{f}(x) = \frac{1}{n}\sum_{i=1}^{n} w(X_i, t)$
  - $\int_{-\infty}^{\infty} w(x, y)dy \ = 1$
  - $w(x, y) \geq 0$ for all x and y
  - e.g. histogram
    - $w(x, y) = \frac{1}{h(x)}$ if x and y are in the same bin
    - $w(x, y) = 0$ otherwise

Discussion

- Choice of Kernels
  - Epanechnikov
    - $K(t) = \frac{0.75\left(1-\frac{1}{5t^2}\right)}{\sqrt{5}}$ for $|t| < \sqrt{5}$
    - $K(t) = 0$ otherwise
  - Biweight
    - $K(t) = \frac{15}{16}(1 - t^2)^2$ for $|t| < 1$
    - $K(t) = 0$ otherwise
  - Gaussian
    - $K(t) = \frac{1}{\sqrt{2\pi}} e^{-0.5t^2}$

Discussion

- Choice of Smoothing Parameters
  - Manual choices: visualize density estimation results with different parameters and select a proper one
  - Automatic choices: select optimal parameters that minimize errors

Error Estimation for Density Estimation

- Mean Squared Error (MSE) at a single point, t
  - $MSE_t(\hat{f}) = E\left(\hat{f}(t) - f(t)\right)^2 = \left(E\left(\hat{f}(t)\right) - f(t)\right)^2 + V\left(\hat{f}(t)\right)$
  - $E\left(\hat{f}(t)\right) = \frac{1}{n}\sum E\big(w(X_i, t)\big) = \int w(x,t)f(x)dx$ (Silverman 1986, 36pp)
  - $V\left(\hat{f}(t)\right) = \frac{1}{n}V\big(w(X_i, t)\big) = \frac{1}{n}\left[\int w(x,t)^2 f(x)dx - \{\int w(x,t)f(x)dx\}^2\right]$
  - The bias, $E\left(\hat{f}(t)\right) - f(t)$, does not depend on the sample size, *n*
  - The variance, $V\left(\hat{f}(t)\right)$, decreases as we use more samples

Error Estimation for Density Estimation

- Asymptotic version (Silverman 1986, 39pp) for univariate data
  - $bias_h(x) = E\left(\hat{f}(x)\right) - f(x) \approx \frac{1}{2}h^2 f''(x)k_2$ *(derived using Taylor expansion)*
  - $V\left(\hat{f}(x)\right) \approx n^{-1}h^{-1}f(x)\int K(t)^2\,dt$

- A trade-off between bias and variance

- In practice,
  - We need a way to estimate unknown terms $f(x)$, $f''(x)$

Discussion

- Radiance estimation in (progressive) photon mapping is an application of the density estimation

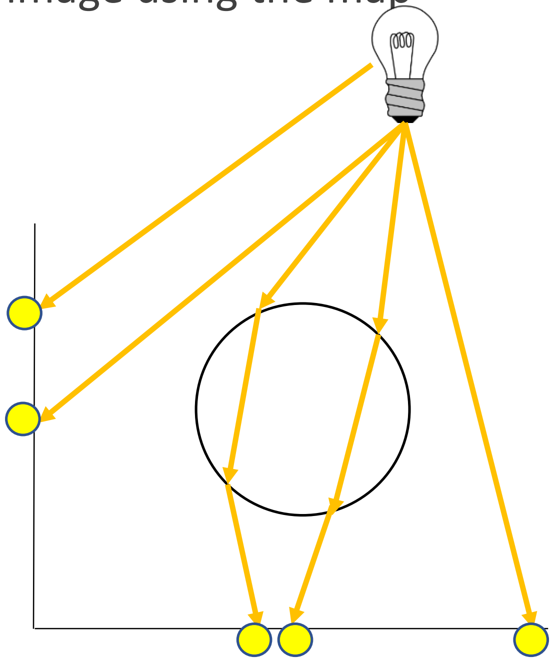
- The radius of the radiance estimation is related to the error terms
- Q. Can we choose an optimal radius in a data-driven way?

Outline

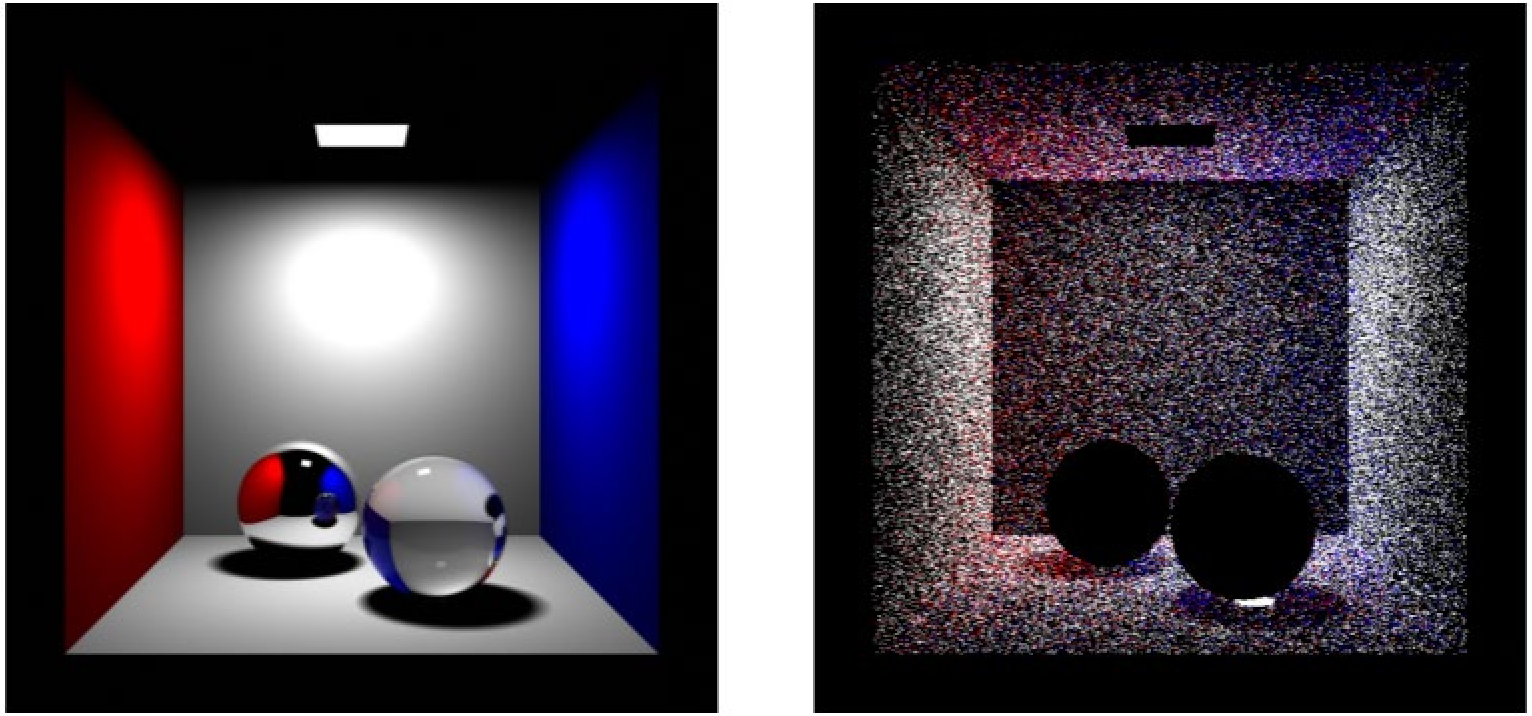
- Path Tracing
- Path Tracing with Next Event Estimation
- Background: Density Estimation
- Photon Mapping
- Progressive Photon Mapping

Photon Mapping

- [Jensen 1996]
- A two-pass rendering method
  - 1. build a photon map
  - 2. render an image using the map



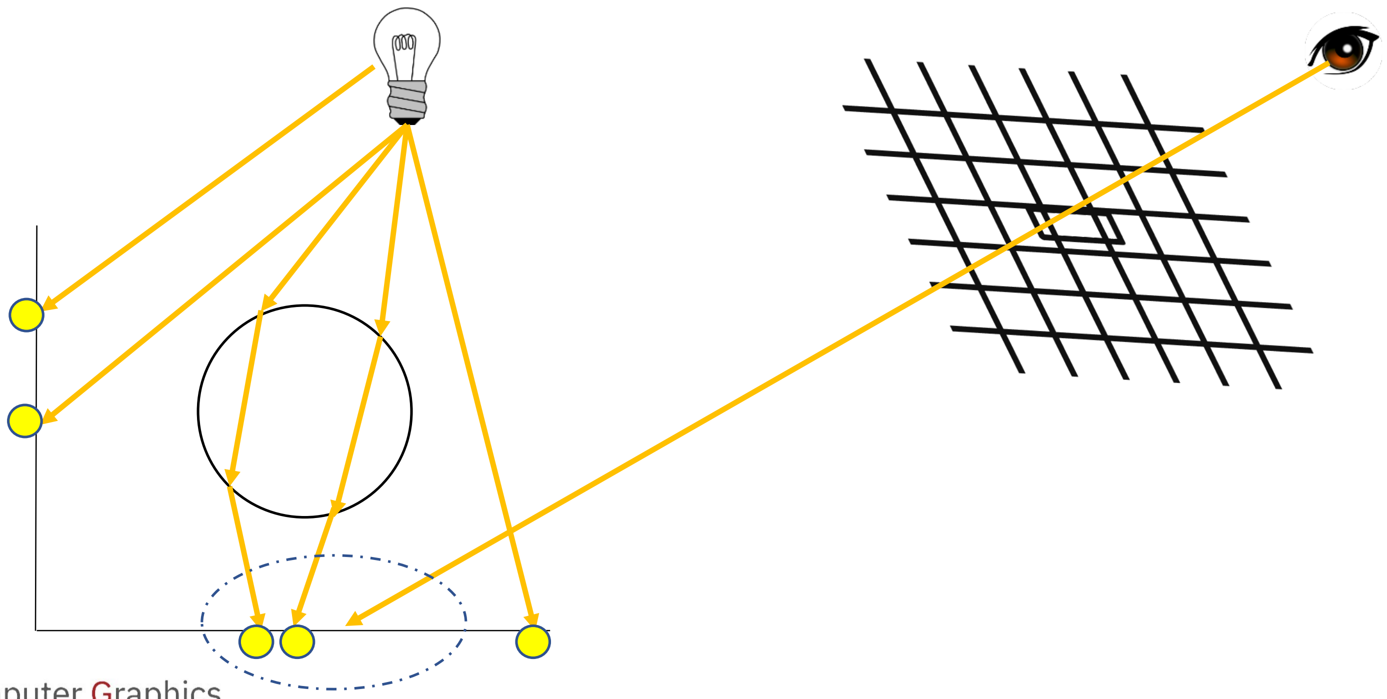
Photon Mapping



[Christensen and Jensen, SIGGRAPH 2000 course]

Photon Mapping

- A two-pass rendering method
  - 1. (photon tracing) Build a photon map
  - 2. (radiance estimation) Render an image using the map



Photon Mapping

- Photon emission
  - Photons are generated from light sources
  - Each photon carries a fraction of the power of a light source
    - Each photon carries a color: $\Delta\Phi_{\mathrm{j}} = \frac{\Phi_{light}}{n_e}$ where n_e is the number of emitted photons

- Structures of a photon
  - 3D position
  - A fraction of the power
  - Incident direction

- Multiple light sources?
  - Each light source emits photons
  - Brighter light sources can emit more photons than the others

Photon Scattering

- Photon tracing employs a ray tracing procedure:
  - If a photon hits a surface, it will be reflected or absorbed

- In practice, a Russian Roulette (RR) is used
  - RR is a stochastic technique to determine whether a photon is reflected or not
  - RR is widely used in Monte Carlo ray tracing methods
  - It improves efficiency by increasing the likelihood that samples can have high contributions
  - Technical details of RR will be given later

Photon Scattering

- Photon tracing employs a ray tracing procedure:
  - If a photon hits,
    - $p = d$ $(probability\ of\ reflection = reflectivity\ of\ the\ surface)$
    - $\xi \in [0,1]$ $(uniformly\ distributed\ random\ number)$
    - $if\ (\xi < p\ )$
      - $reflect\ photon\ with\ power\ \Phi_p$
    - $else$
      - $photon\ is\ absorbed$

Photon Storing

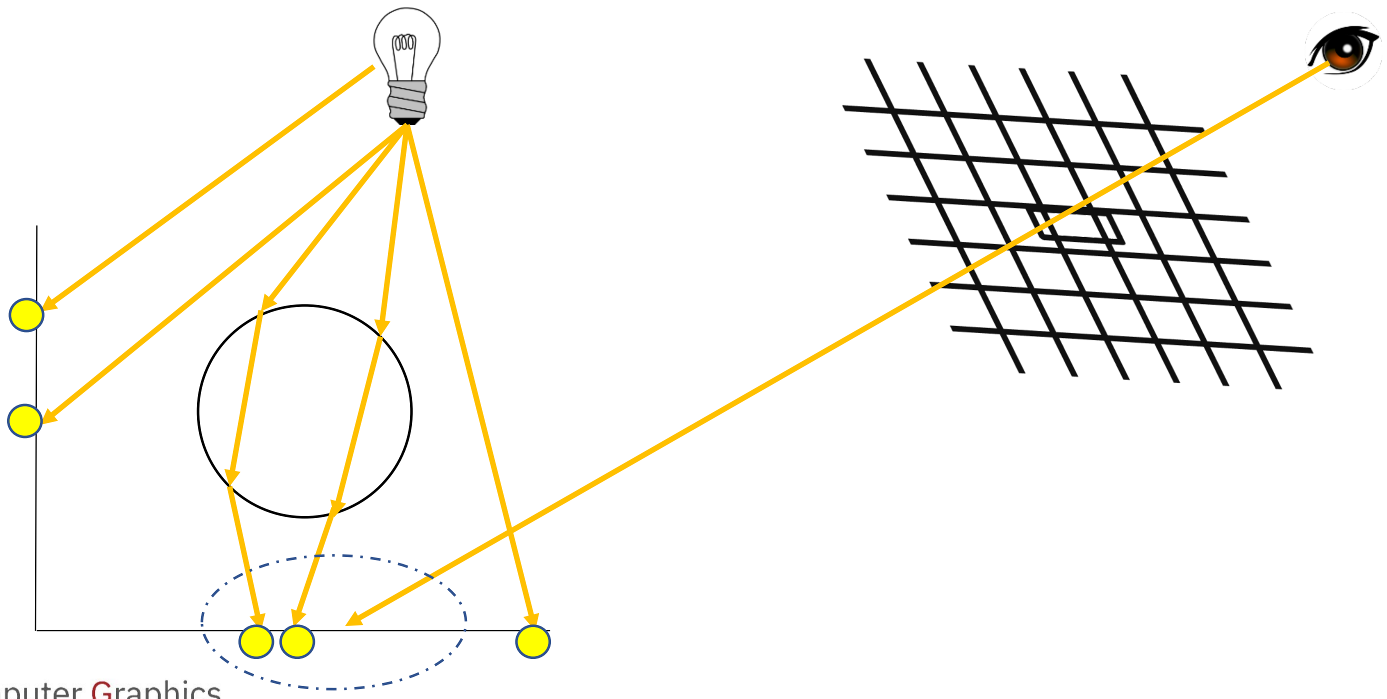
- Store photons when they hit non-specular surfaces
  - Do not need to store them for specular reflection (e.g., reflection on mirrors)

- An emitted photon can be stored several times along its path

- A tree structure (kd-trees) is used to maintain the photons
  - This will be utilized for searching neighboring photons in the second step

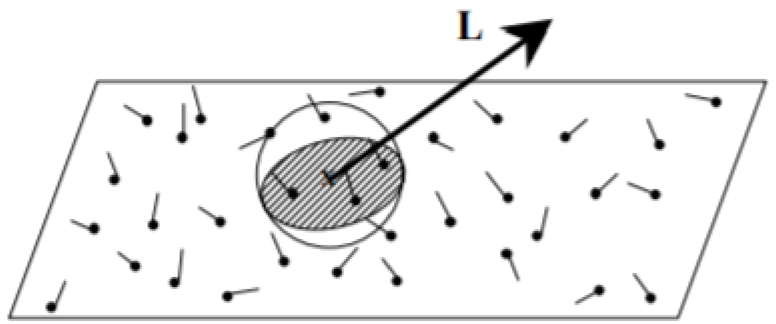
Photon Mapping

- A two-pass rendering method
  - 1. (photon tracing) build a photon map
  - 2. (radiance estimation) render an image using the map



Radiance Estimation

- Need to estimate radiance at query points
  - Relation: $L_f(k_i) = \frac{\Delta\Phi}{\Delta A \cos\theta_\mathrm{i} \Delta\sigma}$
- $L_s(k_o) = \int_{all\ k_i} \rho(k_i, k_o) L_f(k_i) cos\theta_i d\sigma_i$
- $\Delta A = \pi r^2$
  - Assume a locally flat surface around x
  - The radius r can be set using the k-nearest neighbor search, i.e., the k-th largest distance between a query point and the positions of photons
- $L_s(k_o) \approx \sum_{\mathrm{i}=1}^{N} \rho(k_\mathrm{i}, k_o) \frac{\Delta\Phi_\mathrm{i}}{\pi r^2 \cos\theta_\mathrm{i} \Delta\sigma_\mathrm{i}} \cos\theta_\mathrm{i} \Delta\sigma_\mathrm{i} = \sum_{\mathrm{i}=1}^{N} \rho(k_\mathrm{i}, k_o) \frac{\Delta\Phi_\mathrm{i}}{\pi r^2}$



[Christensen and Jensen, SIGGRAPH 2000 course]

Radiance Estimation

- $L_s(k_o) \approx \frac{1}{\pi r^2} \sum_{\mathrm{i}=1}^{N} \rho(k_\mathrm{i}, k_o)\, \Delta\Phi_\mathrm{i}$
- When assumptions (e.g., locally flat surfaces) are valid, and the number of photons is infinite, the approximation error will be zero

- Discussion:
  - Photons are view-independent, and thus one can reuse the photons when the camera animates
  - Q. Can we store an infinite number of photons?
    - If not, is there any way to accomplish the consistency?

Outline

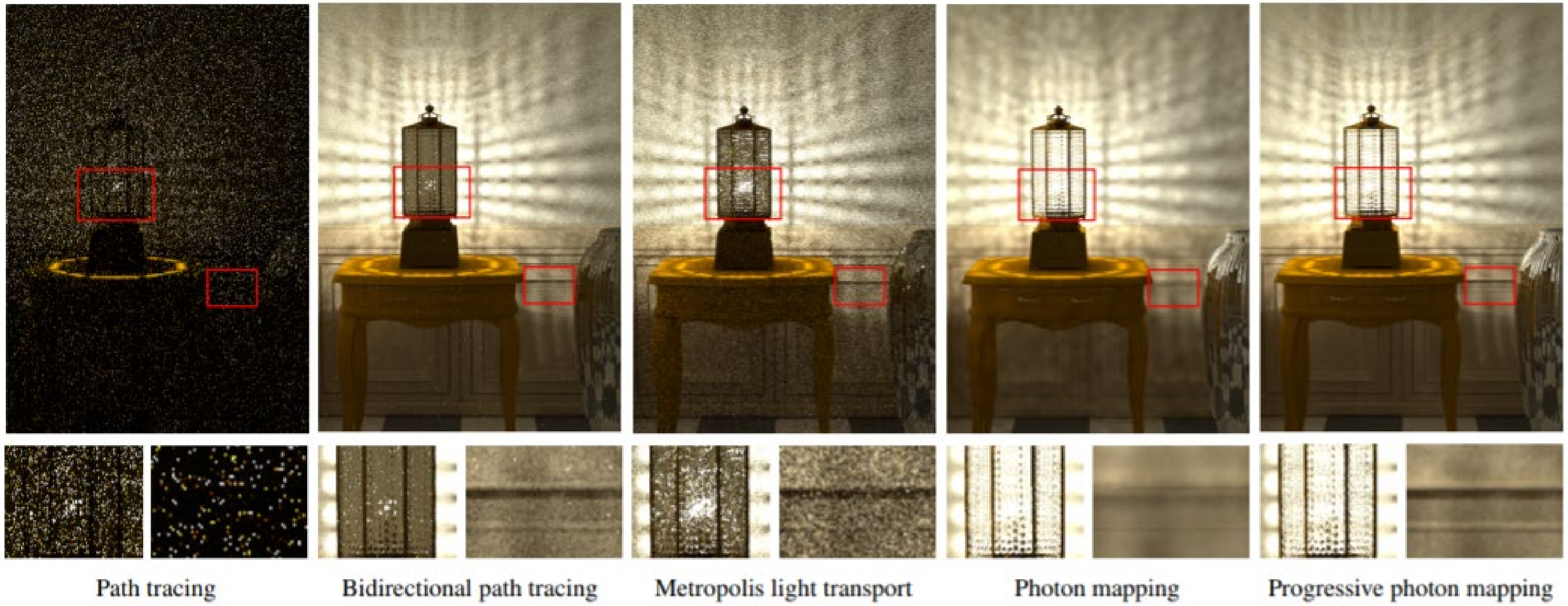
- Path Tracing
- Path Tracing with Next Event Estimation
- Background: Density Estimation
- Photon Mapping
- Progressive Photon Mapping

Radiance Estimation

- $L_s(k_o) \approx \frac{1}{\pi r^2} \sum_{\mathrm{i}=1}^{N} \rho(k_{\mathrm{i}}, k_o)\, \Delta\Phi_{\mathrm{i}}$
- When the number of photons goes infinite (and thus $r \to 0$), it converges to the correct solution (consistent)
  - In practice, we cannot store an infinite number of photons due to a finite memory space

Progressive Photon Mapping

- SIGA 2008, Hachisuka et al.



Images from [Hachisuka et al. 08]

Progressive Photon Mapping

- Multi-pass rendering method
  - 1st pass
    - Query points are generated from the eye
  - Refinement passes:
    - Photon tracing
    - Progressive radiance estimation

Progressive Photon Mapping

- Photon Mapping
  - $L_s(x, k_o) = \frac{1}{\pi r(x)^2} \sum_{p=1}^{N} \rho(x, k_p, k_o) \, \Delta\Phi_p(x, k_p)$

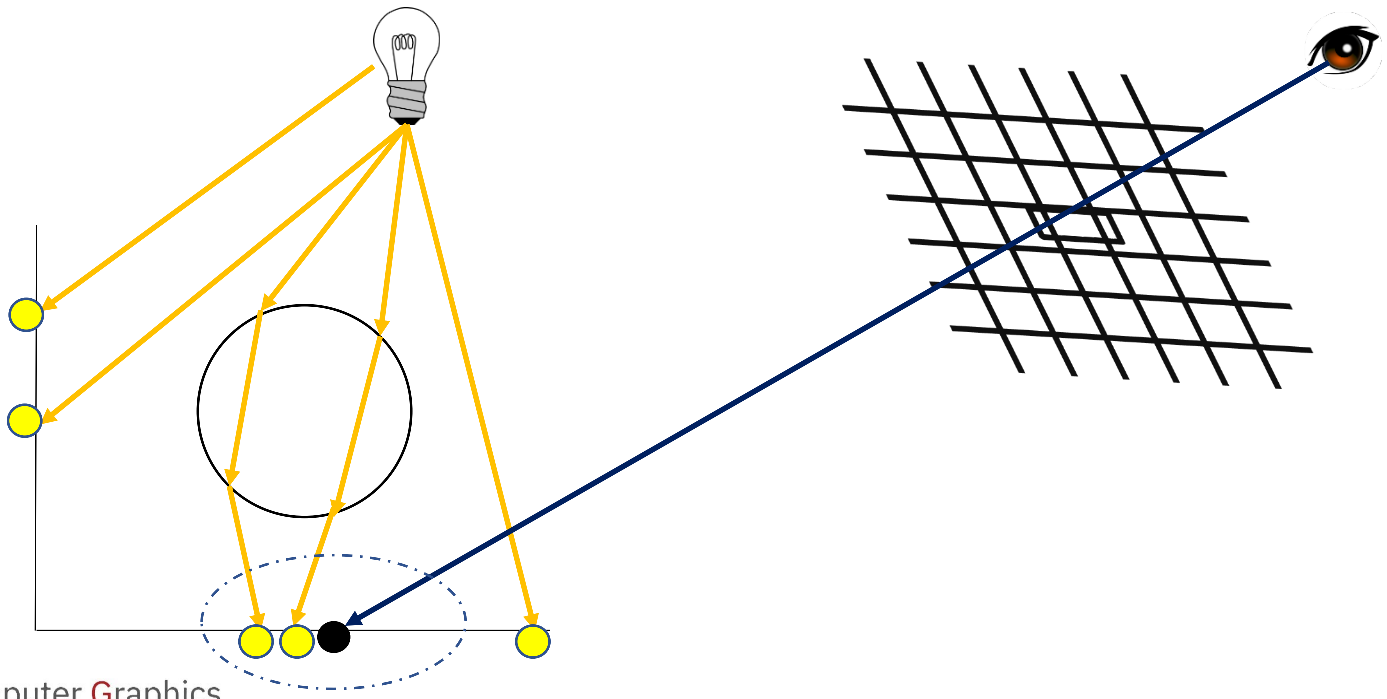
- Progressive Photon Mapping
  - $L_s^0(x, k_o) = \frac{1}{\pi r_0(x)^2} \sum_{p=1}^{N_0} \rho(x, k_p, k_o) \, \Delta\Phi_p(x, k_p)$
  - $L_s^1(x, k_o) = \frac{1}{\pi r_1(x)^2} \sum_{p=1}^{N_1} \rho(x, k_p, k_o) \, \Delta\Phi_p(x, k_p)$
  - …
  - $L_s^i(x, k_o) = \frac{1}{\pi r_i(x)^2} \sum_{p=1}^{N_i} \rho(x, k_p, k_o) \, \Delta\Phi_p(x, k_p)$
  - …

Progressive Photon Mapping

- Progressive Photon Mapping
  - $L_r^i(x, k_o) = \frac{1}{\pi r_i(x)^2} \sum_{p=1}^{N_i} \rho(x, k_p, k_o) \Delta\Phi_p(x, k_p)$
  - $\lim_{i \to \infty} L_r^i(x, k_o) = L(x, k_o)$
  - Key properties for consistency:
    - $r_{i+1}(x) < r_i(x)$
    - $N_{i+1} > N_i$

Progressive Photon Mapping

- PPM assumes the photon density is locally uniform
- Reduce the radius of each hit point while accumulating newly added photons



Progressive Photon Mapping

- PPM assumes the photon density is locally uniform
- Reduce the radius of each hit point while accumulating newly added photons
- $\frac{N_i+M_i}{\pi r_i^2} = \frac{N_{i+1}}{\pi r_{i+1}^2}$
- $N_{i+1} = N_i + \alpha M_i$
  - N_i: # of photons in the previous steps
  - M_i: # of photons in the current step
  - α: a fraction of newly added photons to keep
- $\frac{N_i+M_i}{\pi r_i^2} = \frac{N_i+\alpha M_i}{\pi r_{i+1}^2}$
- $r_{i+1} = r_i \sqrt{\frac{N_i+\alpha M_i}{N_i+M_i}}$

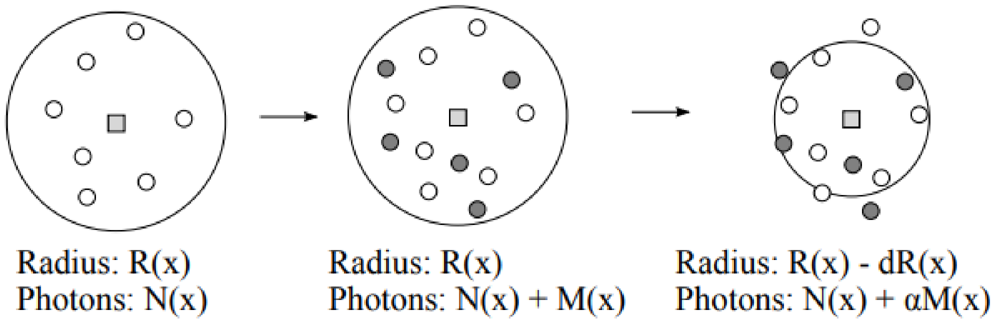


Image from [Hachisuka et al. 2008]

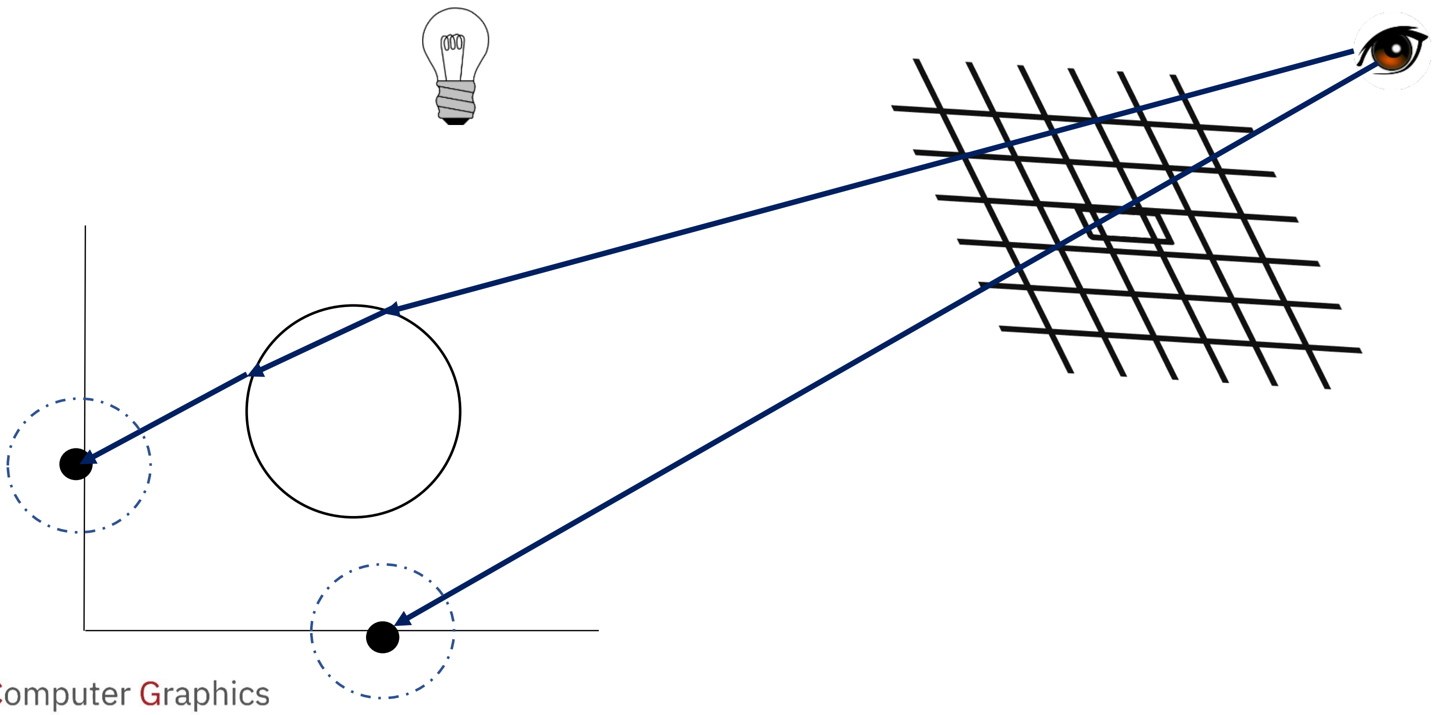
Progressive Photon Mapping

- Flux correction
  - $\tau_{N_i}(x, k_o) = \sum_{p=1}^{N_i} \rho\left(x, k_p, k_o\right) \Delta\Phi_p\left(x, k_p\right)$
  - $\tau_{M_i}(x, k_o) = \sum_{p=1}^{M_i} \rho\left(x, k_p, k_o\right) \Delta\Phi_p\left(x, k_p\right)$
  - $\tau_{N_{i+1}}(x, k_o) = \left(\tau_{N_i}(x, k_o) + \tau_{M_i}(x, k_o)\right) \frac{N_i + \alpha M_i}{N_i + M_i}$

- Radiance
  - $L_r^i(x, k_o) = \frac{1}{\pi r_i^2} \times \frac{\tau_{N_i}}{total\ \#\ of\ emitted\ photons}$

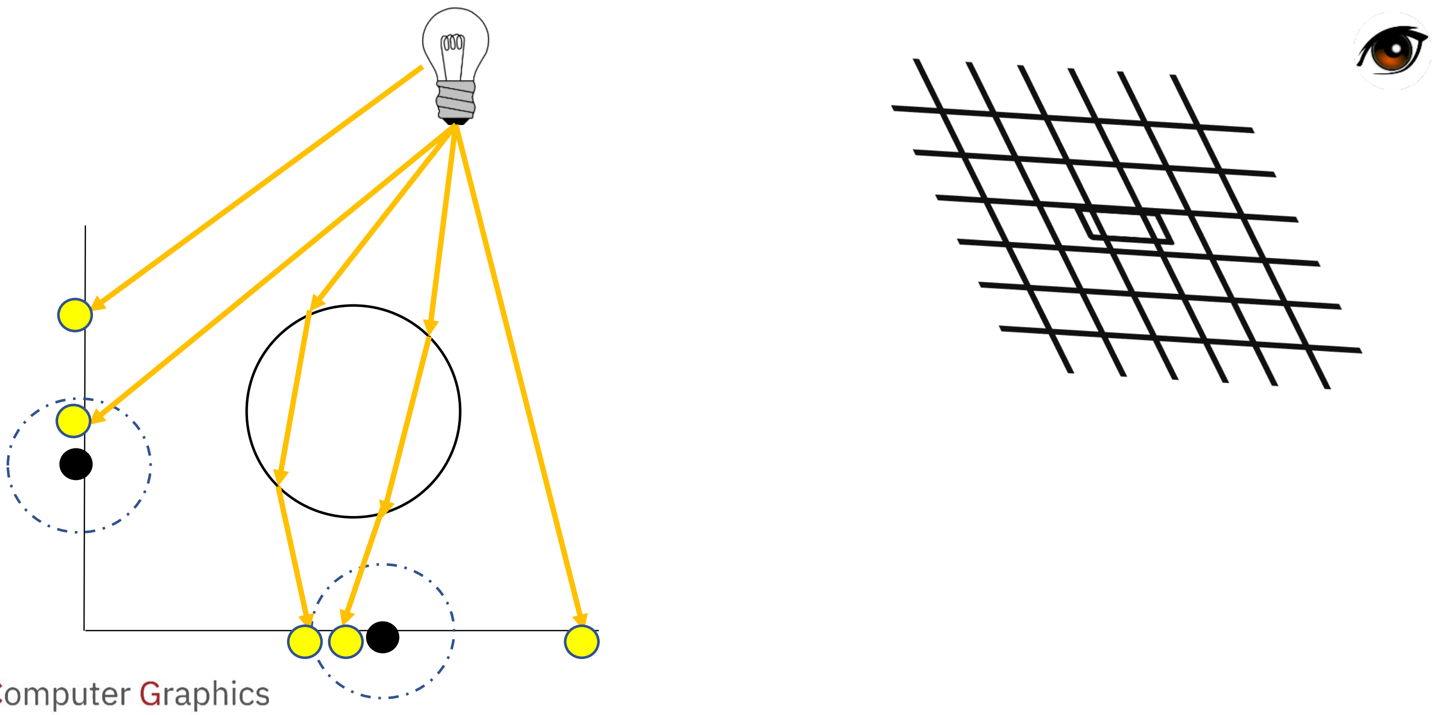
Progressive Photon Mapping

- 1^{st} pass
  - Generate hit points



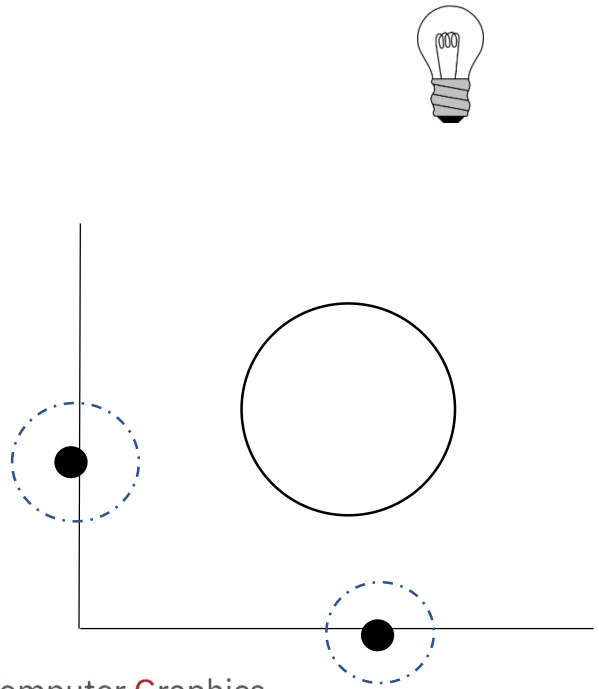
Progressive Photon Mapping

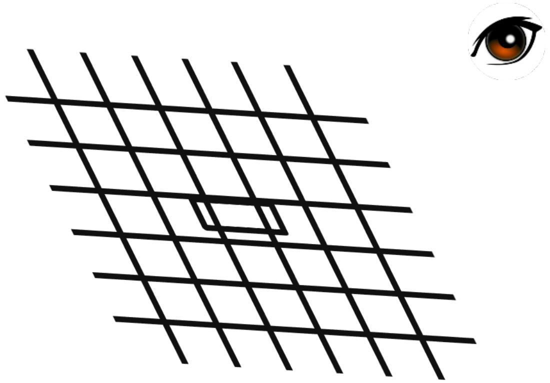
- 1st Refinement pass
    - Generate photons



Progressive Photon Mapping

- 1st Refinement pass
  - Generate photons

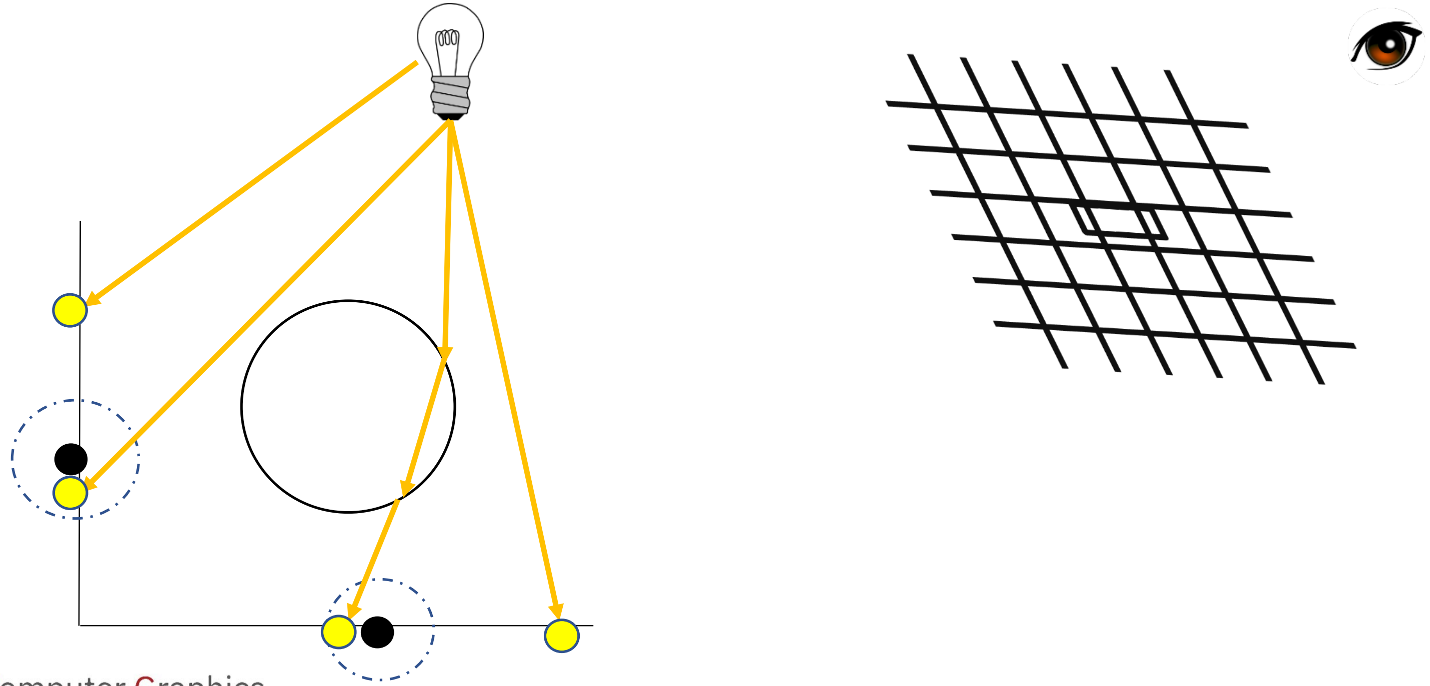




Computer Graphics
Laboratory

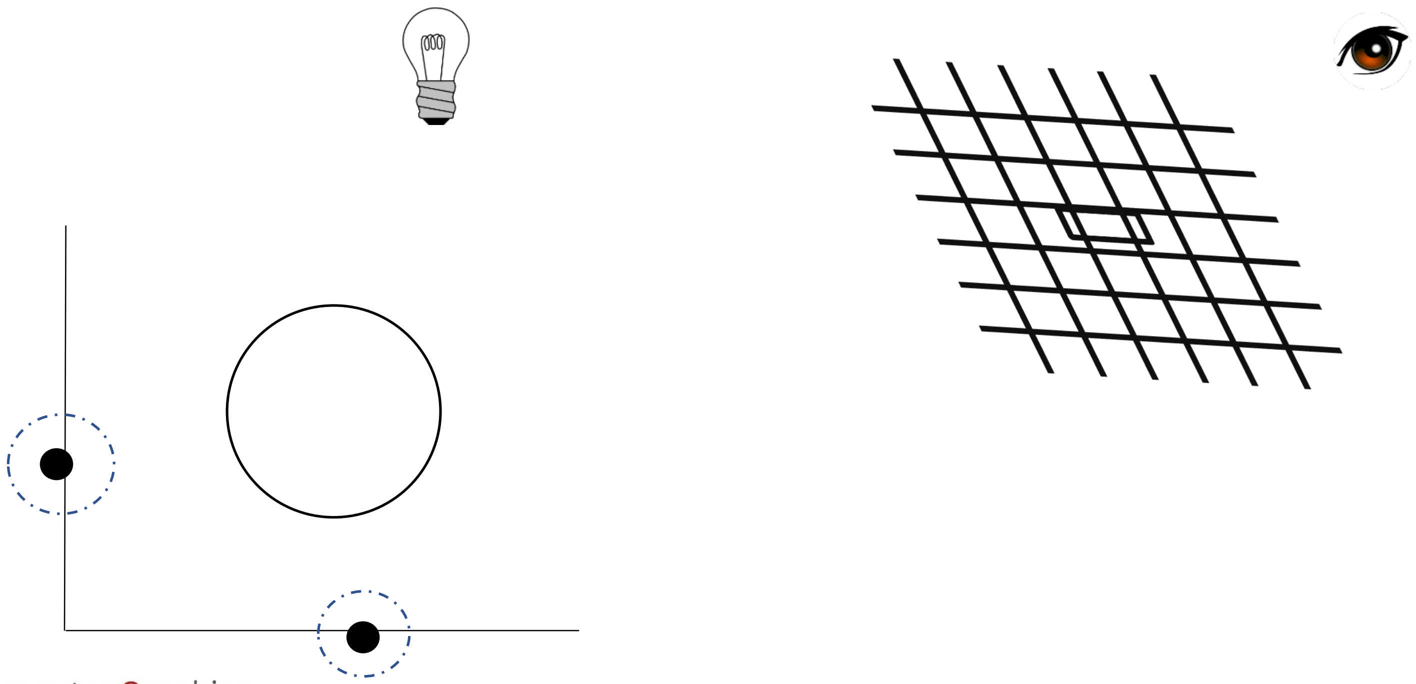
Progressive Photon Mapping

- 2nd Refinement pass
  - Generate photons



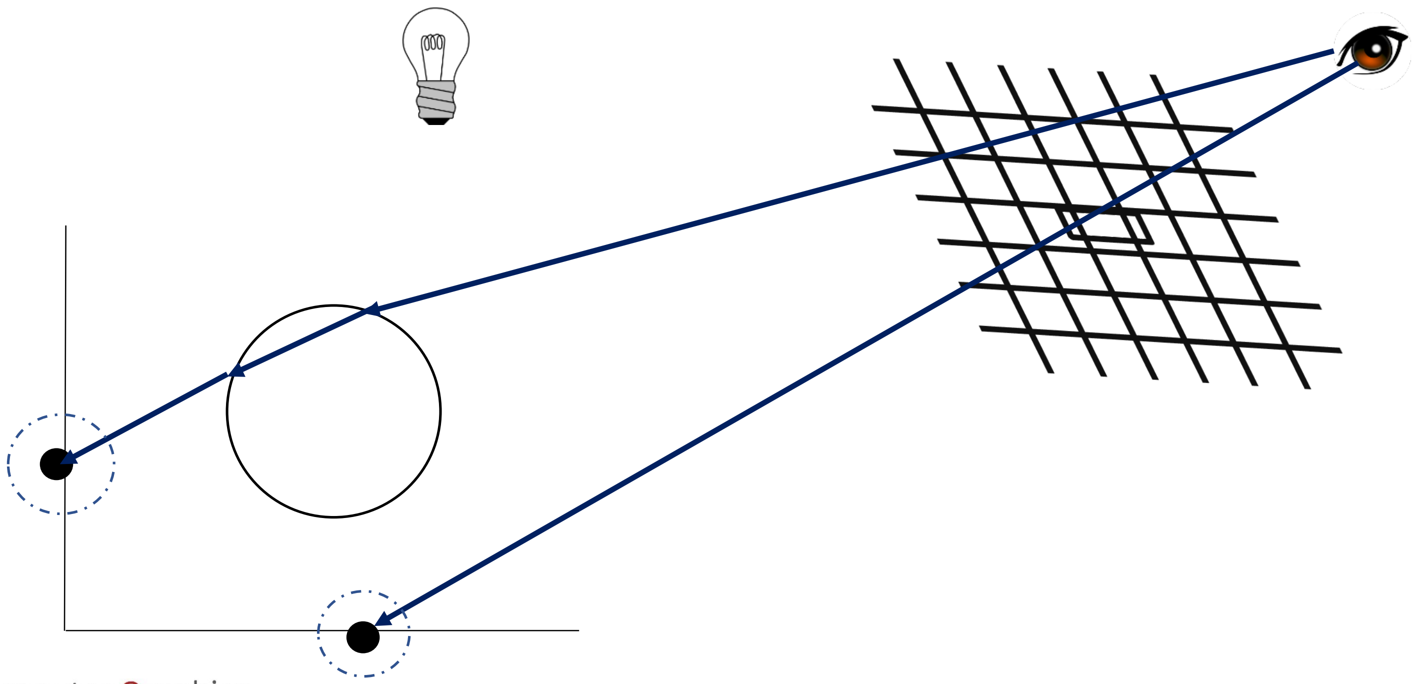
Progressive Photon Mapping

- 2^{nd} Refinement pass
  - Generate photons

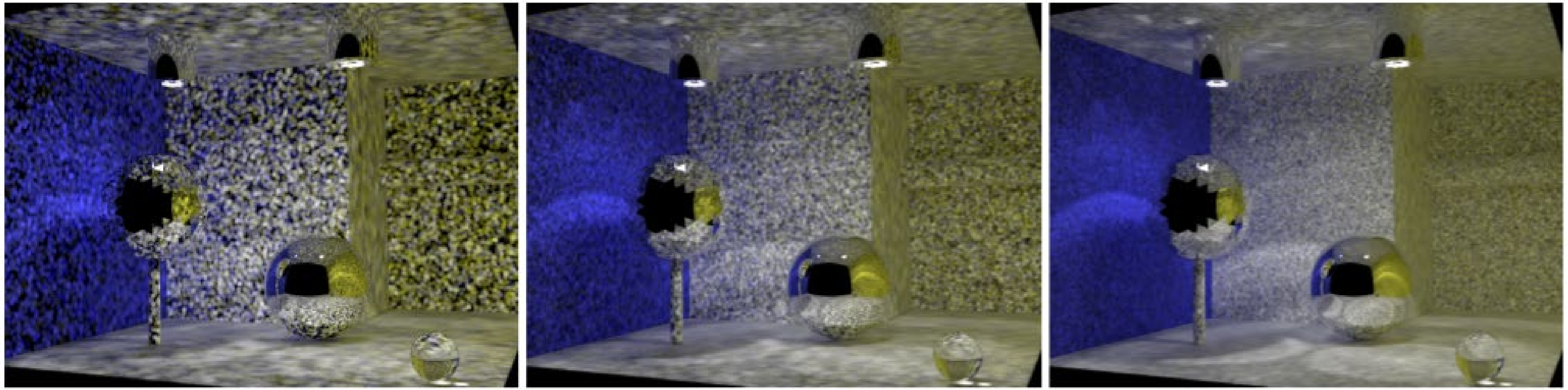


Progressive Photon Mapping

- Rendering



PPM Results



0.1M photons | 0.4M photons | 1.6M photons

Image from [Hachisuka et al. 2008]

PPM Results

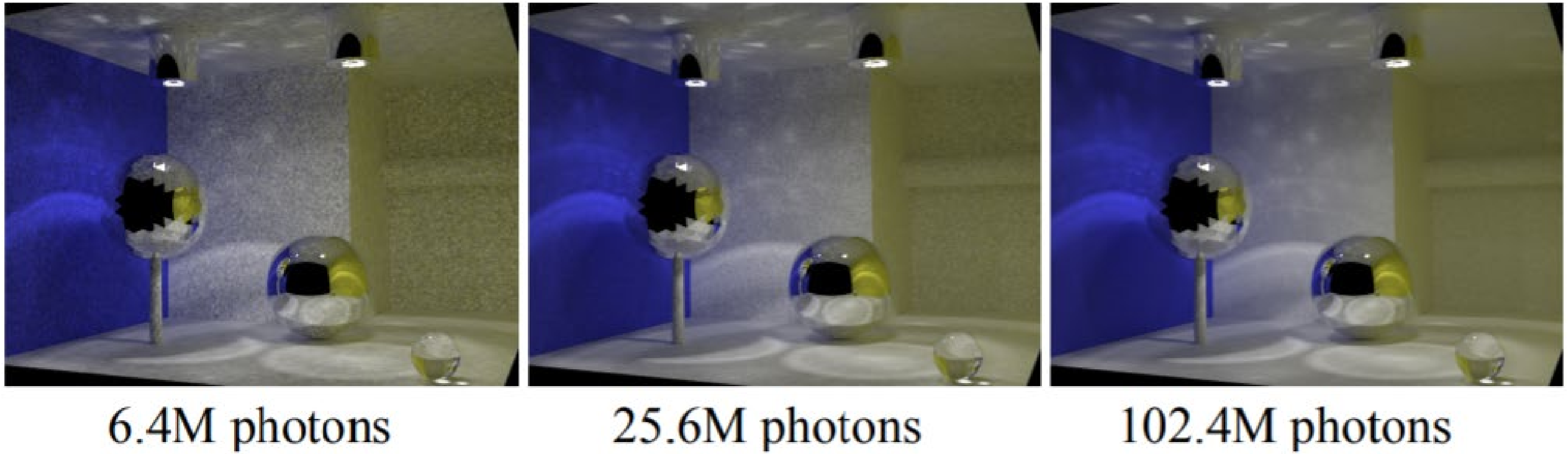


Image from [Hachisuka et al. 2008]

Further Reading

- Discussion:
  - PPM is a view-dependent rendering, so photons should be generated per frame
- Distributed effects?
  - # of hit points can introduce a memory issue, especially for distributed effects
  - Stochastic Progressive Photon Mapping, Hachisuka et al., SIGA09
    - Shared hit points per pixel
- Bandwidth (i.e., kernel radius) optimizations:
  - APPM: Adaptive Progressive Photon Mapping, Kaplanyan and Dachsbacher et al. 2012
  - CPPM: Chi-squared Progressive Photon Mapping, Lin et al. 2020