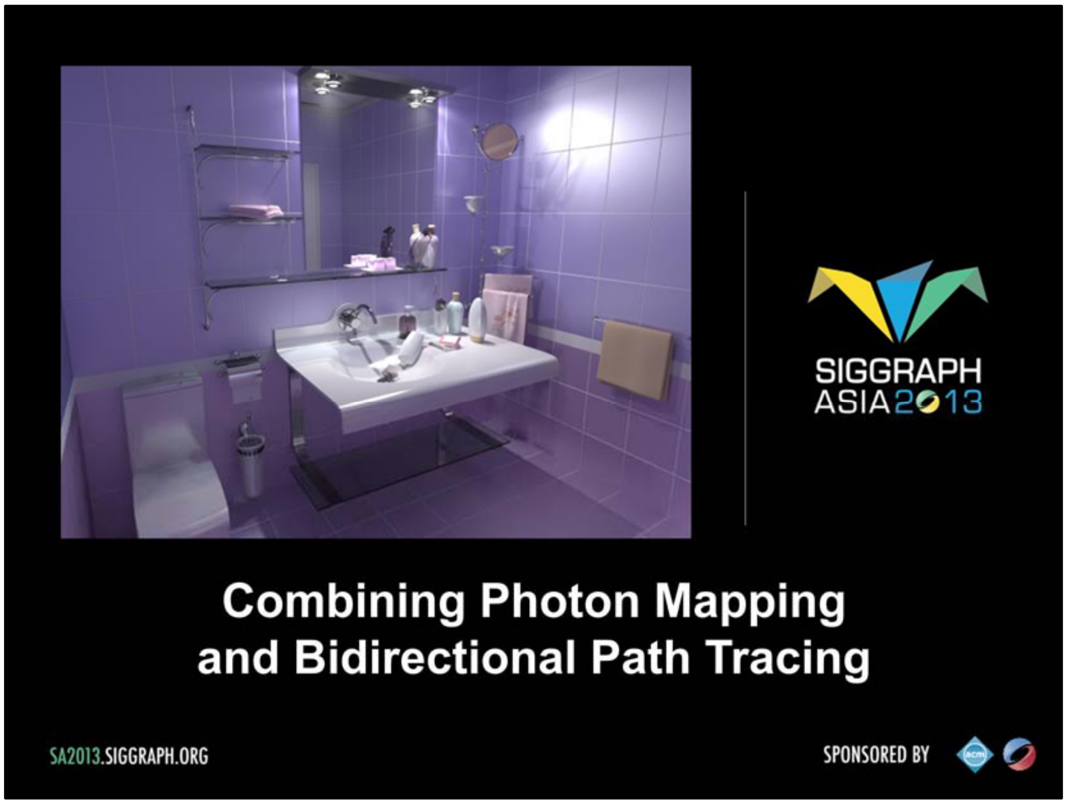


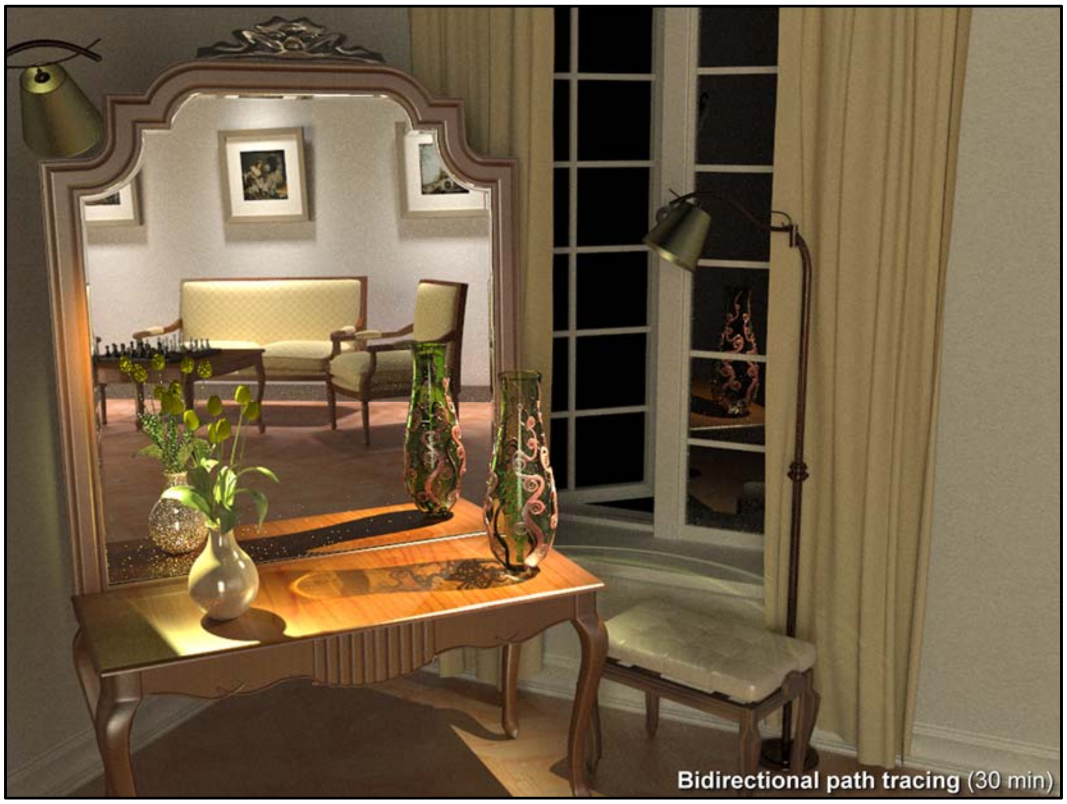
Combining Photon Mapping and Bidirectional Path Tracing

SIGGRAPH ASIA 2013

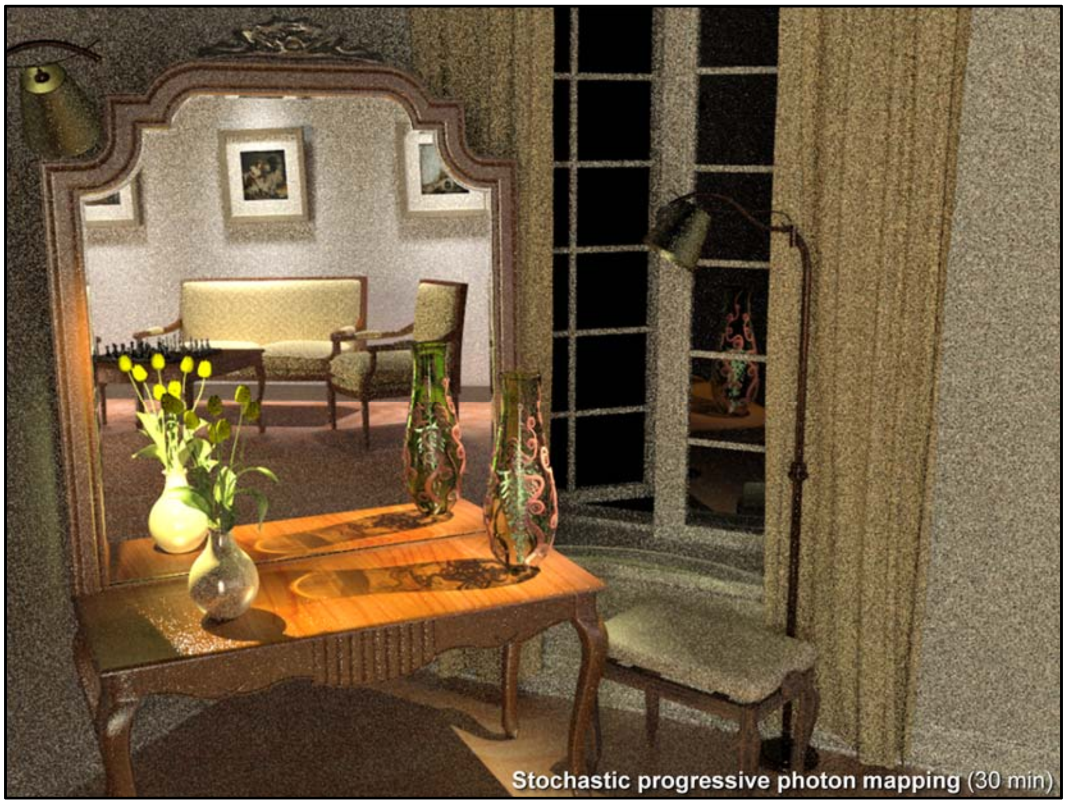
SA2013.SIGGRAPH.ORG

SPONSORED BY

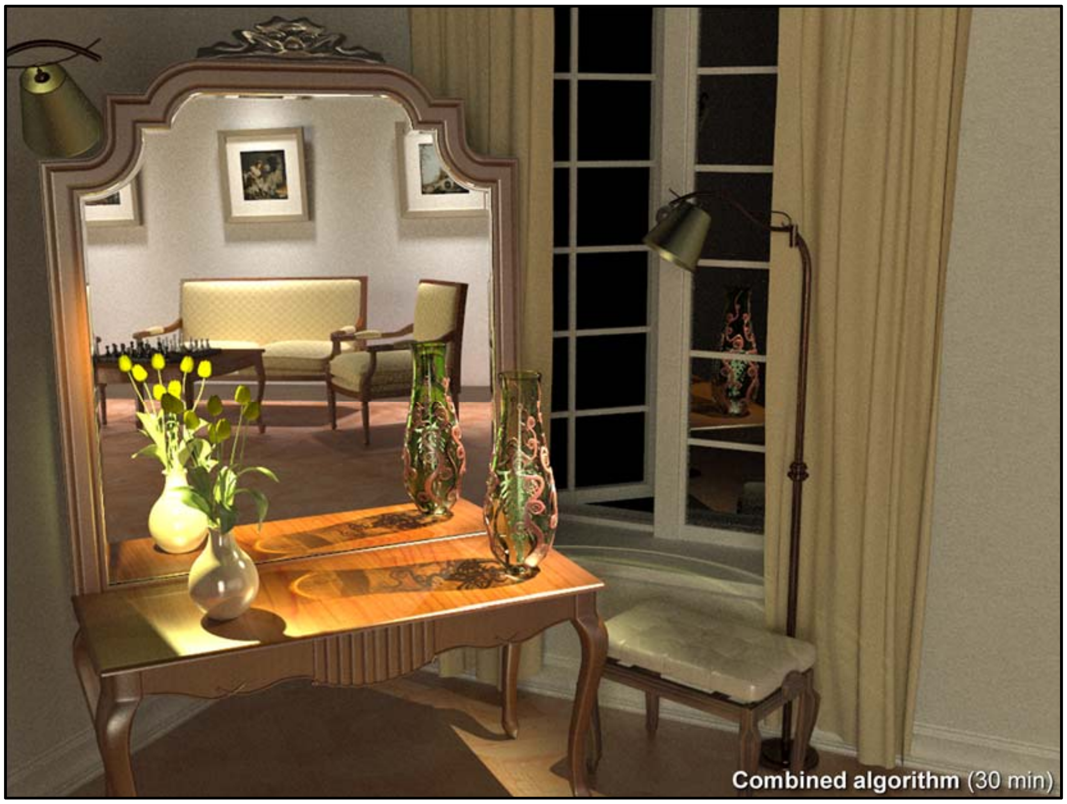
The previous presentations in this course demonstrated the robustness of photon mapping under various difficult lighting conditions. In this talk, I will show how photon mapping can be combined with bidirectional path tracing – another very robust rendering algorithm – to handle complex light transport even more efficiently.



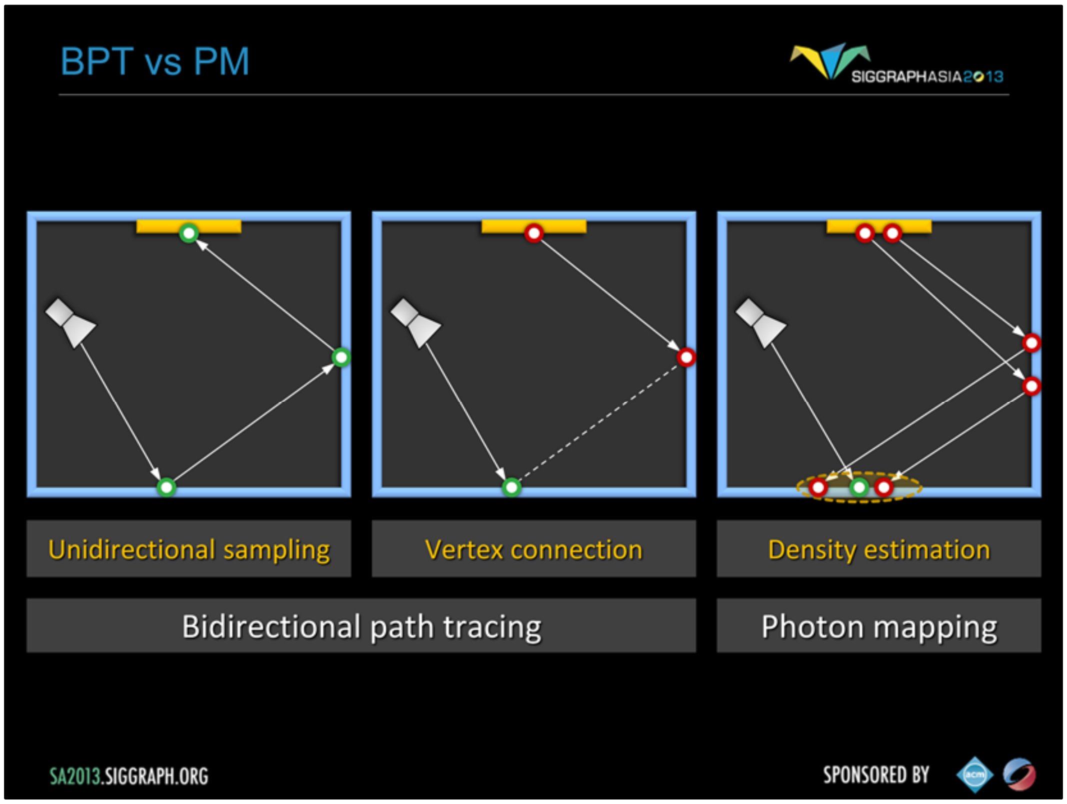
Bidirectional path tracing is one of the most versatile light transport simulation algorithms available today. It can robustly handle a wide range of illumination and scene configurations, but is notoriously inefficient for specular-diffuse-specular light interactions, which occur e.g. when a caustic is seen through a reflection/refraction.



On the other hand, photon mapping (PM) is well known for its efficient handling of caustics. Recently, Hachisuka and Jensen [2009] showed a progressive variant of PM that converges to the correct solution with a fixed memory footprint. Their stochastic progressive photon mapping (PPM) algorithm captures the reflected caustics in our scene quite well. However, it has hard time handling the strong distant indirect illumination coming from the part of the scene behind the camera.



By using multiple importance sampling to combine estimators from bidirectional path tracing and photon mapping, the algorithm I will talk about today automatically finds a good mixture of techniques for each individual light transport path, and produces a clean image in the same amount of time.

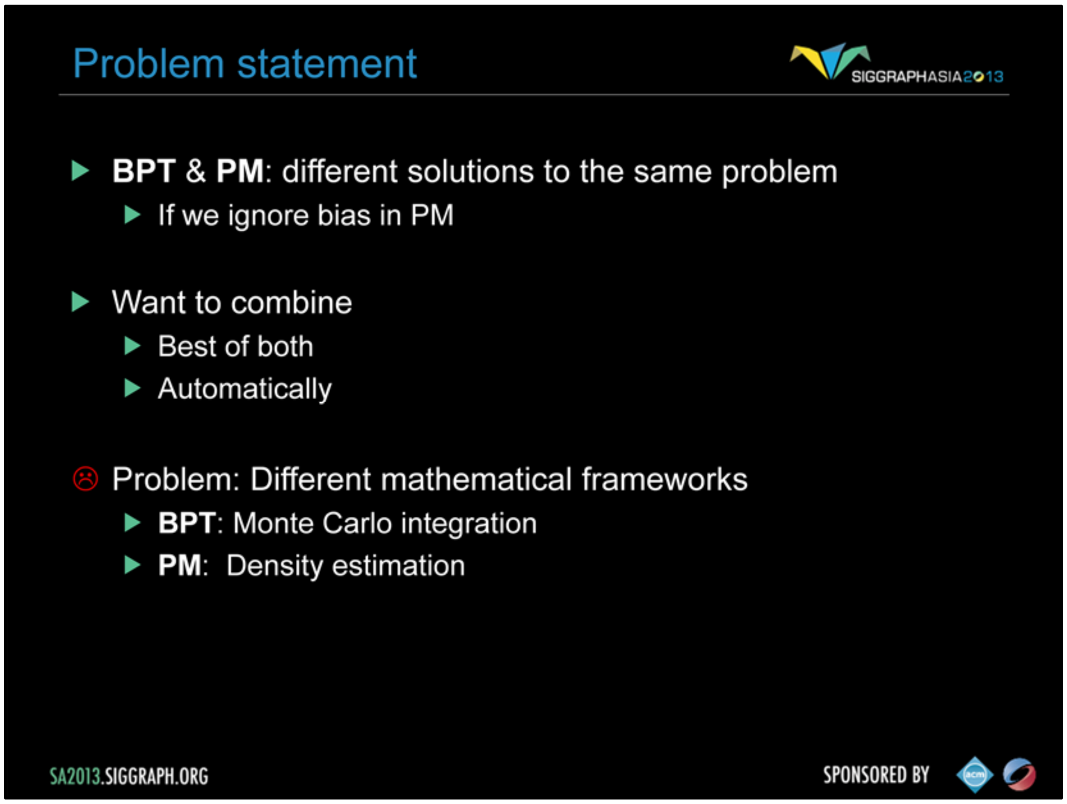


BPT vs PM

Let us start by reviewing how bidirectional path tracing (BPT) and photon mapping (PM) sample light transport paths that connect the light sources to the camera:

[CLICK] The techniques BPT employs can be roughly categorized to *unidirectional sampling* (US) and *vertex connection* (VC). [CLICK] US constructs a path by starting either from a light source or the camera and tracing a random walk in the scene until termination. [CLICK] On the other hand, VC traces one subpath from a light source and another one from the camera, [CLICK] and then completes a full path by connecting their endpoints.

[CLICK] In contrast, PM first traces a number of light subpaths and stores their vertices, a.k.a. photons. [CLICK] It then traces subpaths from the camera and computes the outgoing radiance at the hit points using density estimation by looking up nearby photons.



Problem statement

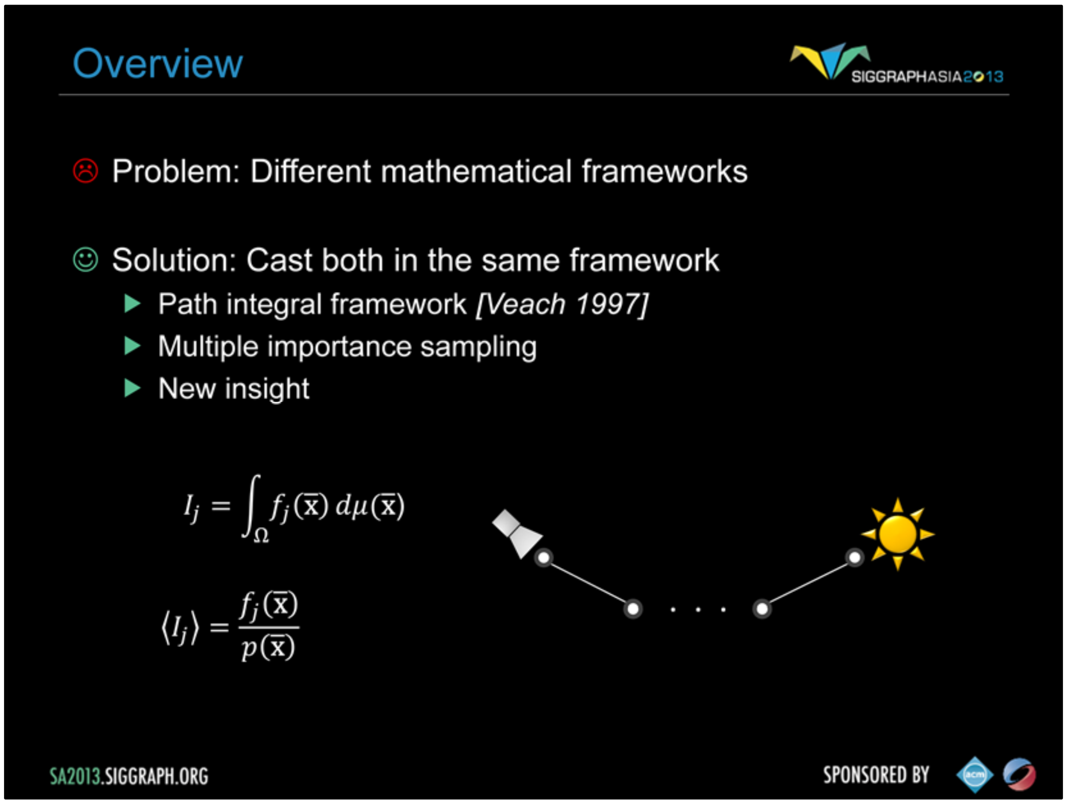
- **BPT** & **PM**: different solutions to the same problem
  - If we ignore bias in PM
- Want to combine
  - Best of both
  - Automatically
- ☹ Problem: Different mathematical frameworks
  - **BPT**: Monte Carlo integration
  - **PM**: Density estimation

SA2013.SIGGRAPH.ORG

SPONSORED BY

We see that bidirectional path tracing (BPT) and photon mapping (PM) are different solutions to the same problem. That is, if we ignore the bias introduced by PM. Furthermore, the previous comparison shows that BPT and PM complement each other in terms of the light transport effects they can efficiently handle.

It is thus logical to want to combine these two methods. Ideally, we want an automatic combination that preserves the qualities of each. Interestingly, even though both methods have been published over 15 years ago, neither a rigorous analysis of their relative performance nor an efficient combination had been shown until very recently. The reason for this is that BPT and PM have originally been defined in different theoretical frameworks – BPT as a standard Monte Carlo estimator to the path integral, and PM as an outgoing radiance estimator based on photon density estimation.



Overview

☹ Problem: Different mathematical frameworks

☺ Solution: Cast both in the same framework

- Path integral framework *[Veach 1997]*
- Multiple importance sampling
- New insight

$$I_j = \int_\Omega f_j(\bar{\mathrm{x}})\, d\mu(\bar{\mathrm{x}})$$

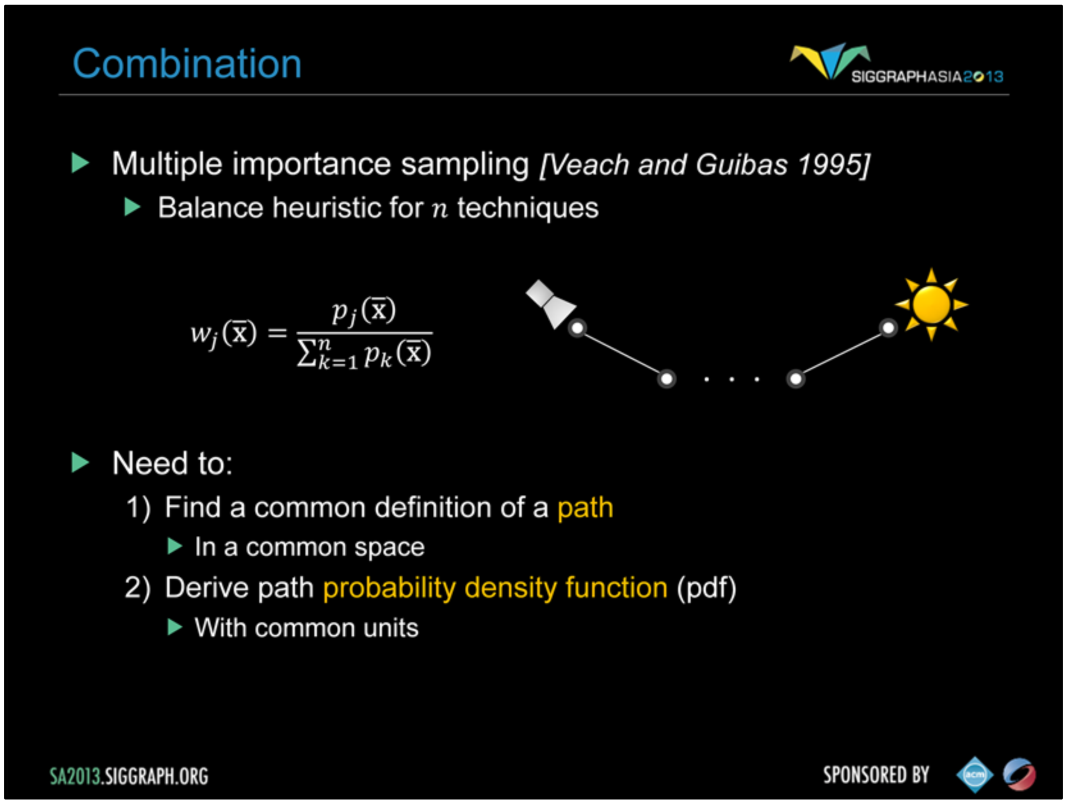
$$\langle I_j \rangle = \frac{f_j(\bar{\mathrm{x}})}{p(\bar{\mathrm{x}})}$$

SA2013.SIGGRAPH.ORG

SPONSORED BY

The first step toward combining these two methods is to cast them in the same mathematical framework. We choose Veach's path integral framework where BPT is already naturally defined. This framework formulates the problem of computing the value of a pixel 'j' as an integral over the energy contribution of all light transport paths from the light sources to the camera. An estimator for this value is obtained by sampling one such random path and dividing its contribution by the path pdf. Different path sampling techniques result in different path pdfs, and BPT uses multiple importance sampling to efficiently combine the resulting estimators.

Later on, we will see that casting both methods in the same path integral framework will also provide a basis for reasoning about the relative efficiency of BPT and PM.



Combination

- Multiple importance sampling *[Veach and Guibas 1995]*
  - Balance heuristic for n techniques

$$w_j(\bar{\mathrm{x}}) = \frac{p_j(\bar{\mathrm{x}})}{\sum_{k=1}^{n} p_k(\bar{\mathrm{x}})}$$

- Need to:
  1) Find a common definition of a path
     - In a common space
  2) Derive path probability density function (pdf)
     - With common units

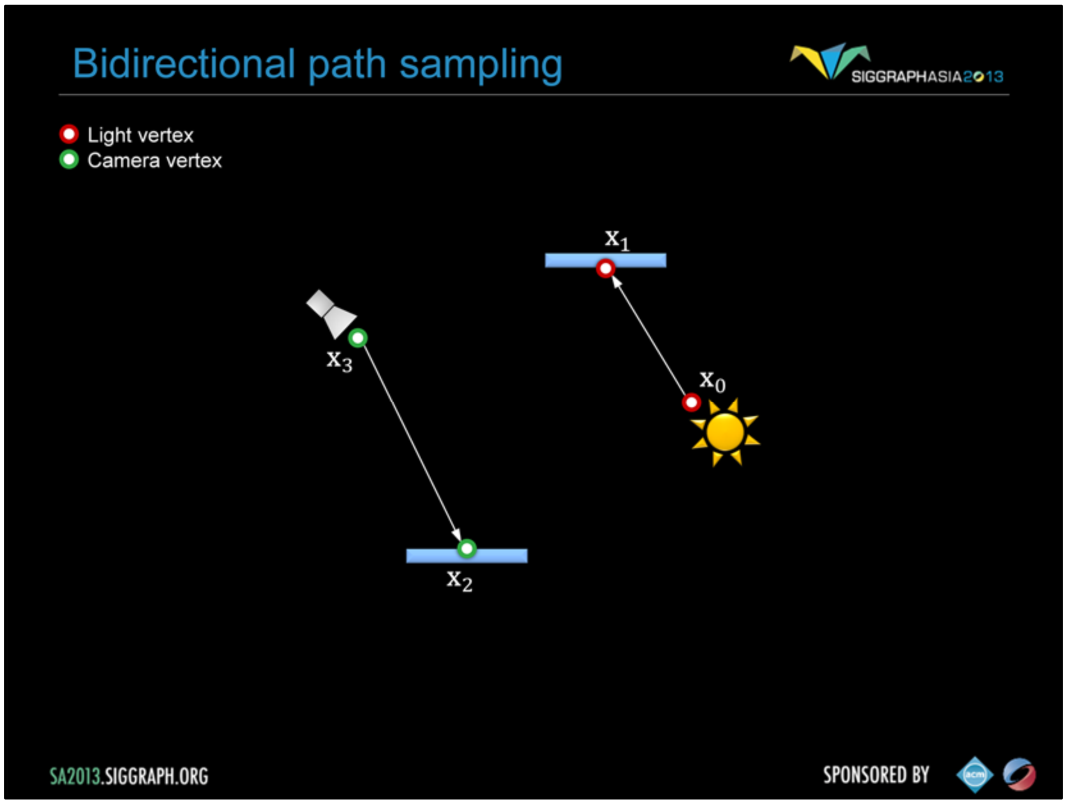
SA2013.SIGGRAPH.ORG

SPONSORED BY

Multiple importance sampling (MIS) combines the estimators corresponding to different path sampling techniques by assigning each estimator a weight that is a function of the actual sampled path. The balance heuristic makes this weight proportional to the path pdf.

In order to combine BPT and PM using MIS, we need two things. First, we have to find a common definition of the light transport paths sampled by both methods. Recall that photon mapping has been originally defined as an outgoing radiance estimator, without explicit distinction between individual full light transport paths constructed with each photon subpath.

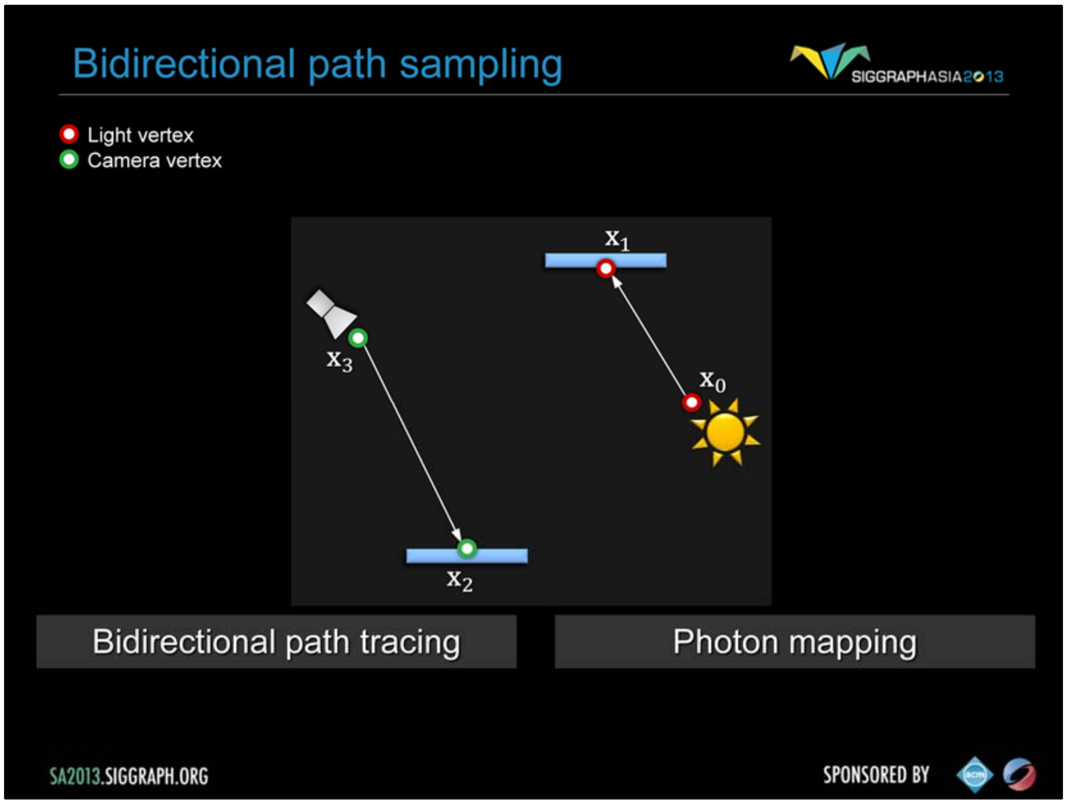
Second, we need to derive the probability density function (pdf) for sampling these paths. In order to apply the balance heuristic in a meaningful way, the pdfs corresponding to the path sampling techniques in both methods should have the same units. This means that the paths need to reside in the same space, i.e. have the same number of vertices.



Bidirectional path sampling

- 🔴 Light vertex
- 🟢 Camera vertex

To find a common definition of a path, let us follow the sampling procedures used in BPT and PM. We start by tracing two independent subpaths, one from a light source and another one from the camera.



Bidirectional path sampling

SIGGRAPHASIA2013

● Light vertex
● Camera vertex

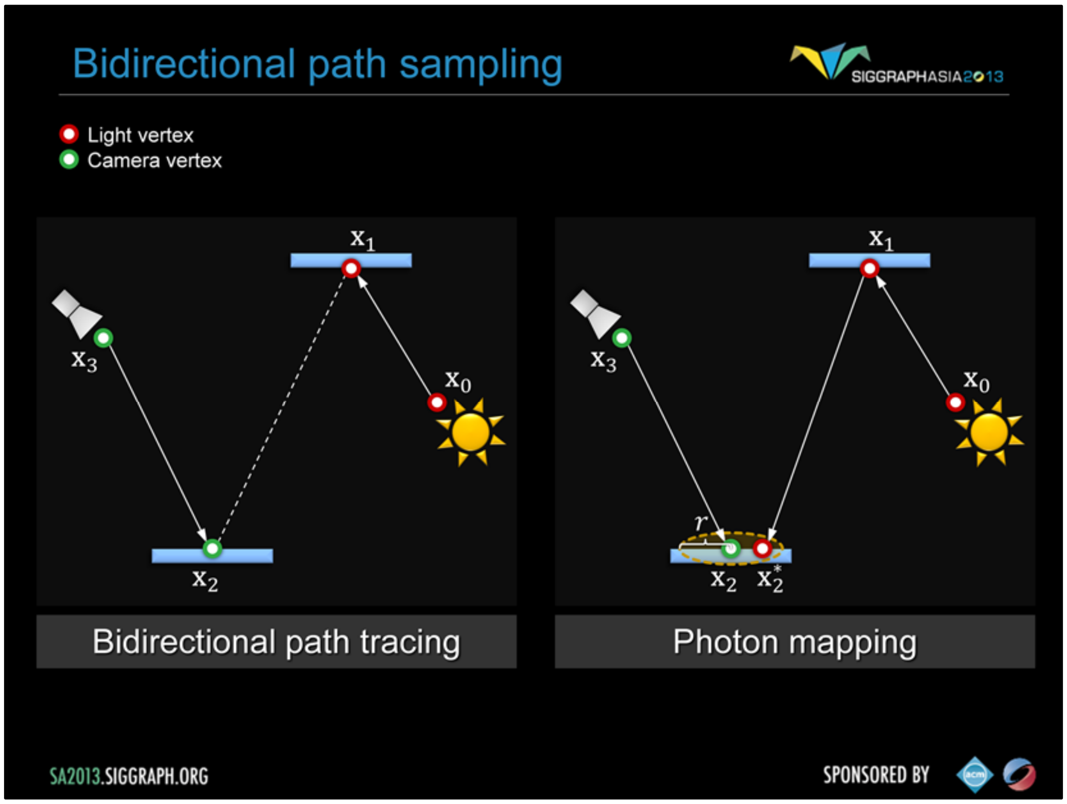
Bidirectional path tracing

Photon mapping

SA2013.SIGGRAPH.ORG

SPONSORED BY

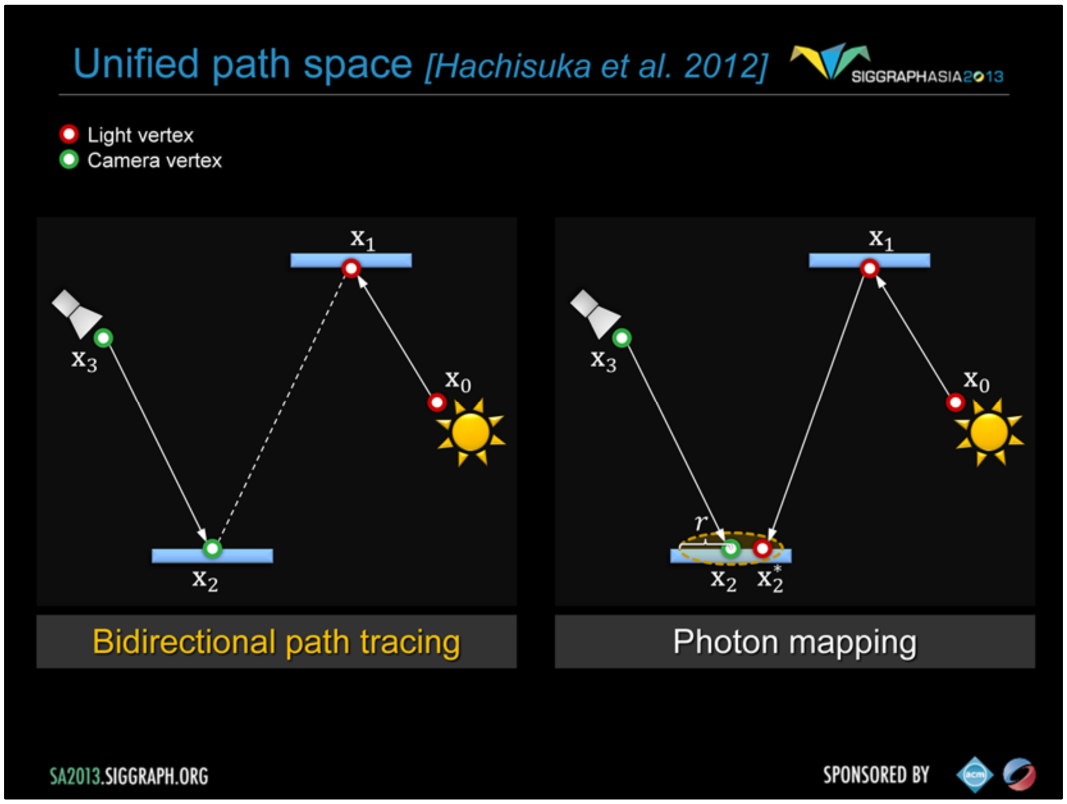
Now let us see how BPT and PM complete a full path.



Bidirectional path sampling

Given these two subpaths, BPT places a shadow connection between their endpoints. [CLICK] Photon mapping, on the other hand, extends the light subpath by sampling one more vertex from $\mathbf{x}_1$, and then concatenates the subpaths only if the photon hit-point $\mathbf{x}_2^*$ lies within a distance r from $\mathbf{x}_2$.

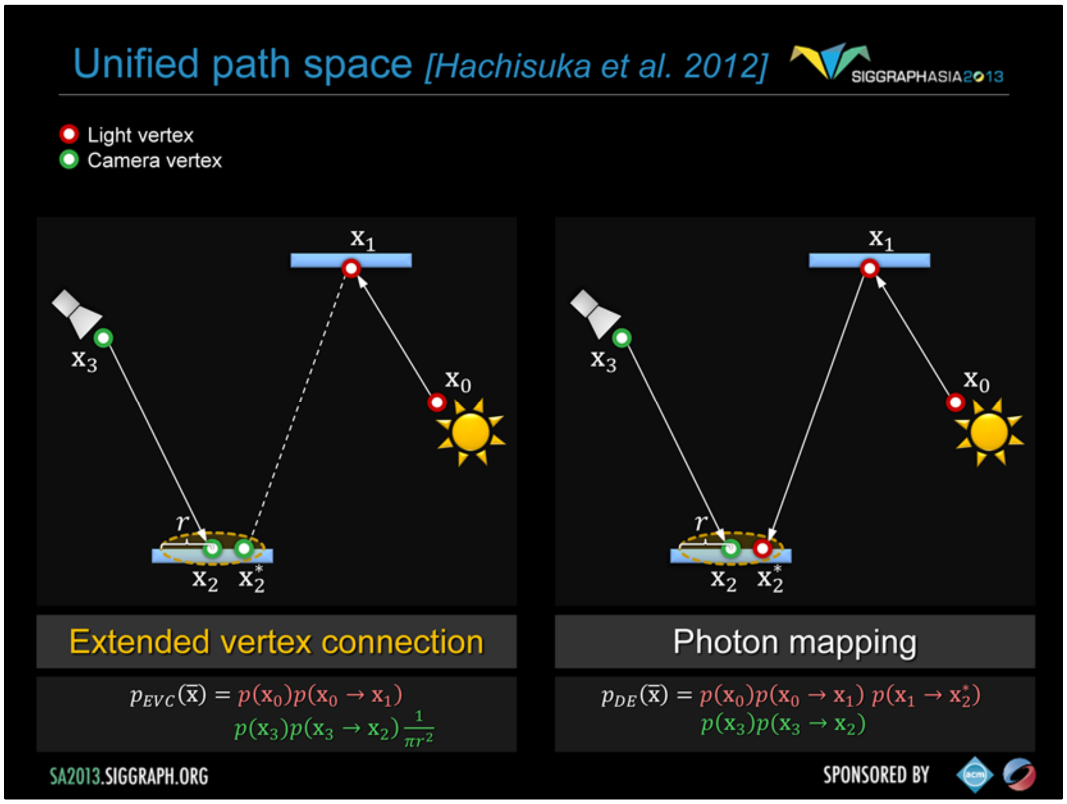
The resulting full path in PM has one more vertex than the corresponding BPT path. Thus, they reside in different path spaces and their pdfs also have different units. Two recent works on combining PM and BPT have approached this problem in different ways.



Unified path space *[Hachisuka et al. 2012]*

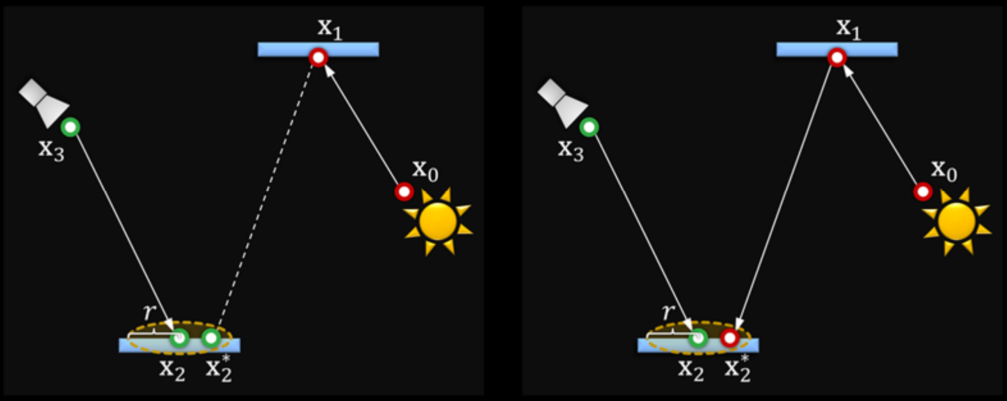
● Light vertex
● Camera vertex

Hachisuka et al. [2012] consider an extension of BPT that samples paths in the higher-dimensional space of PM.



Unified path space [Hachisuka et al. 2012]

● Light vertex
● Camera vertex



Extended vertex connection

$$p_{EVC}(\bar{x}) = p(x_0)p(x_0 \to x_1) \; p(x_3)p(x_3 \to x_2)\frac{1}{\pi r^2}$$

Photon mapping

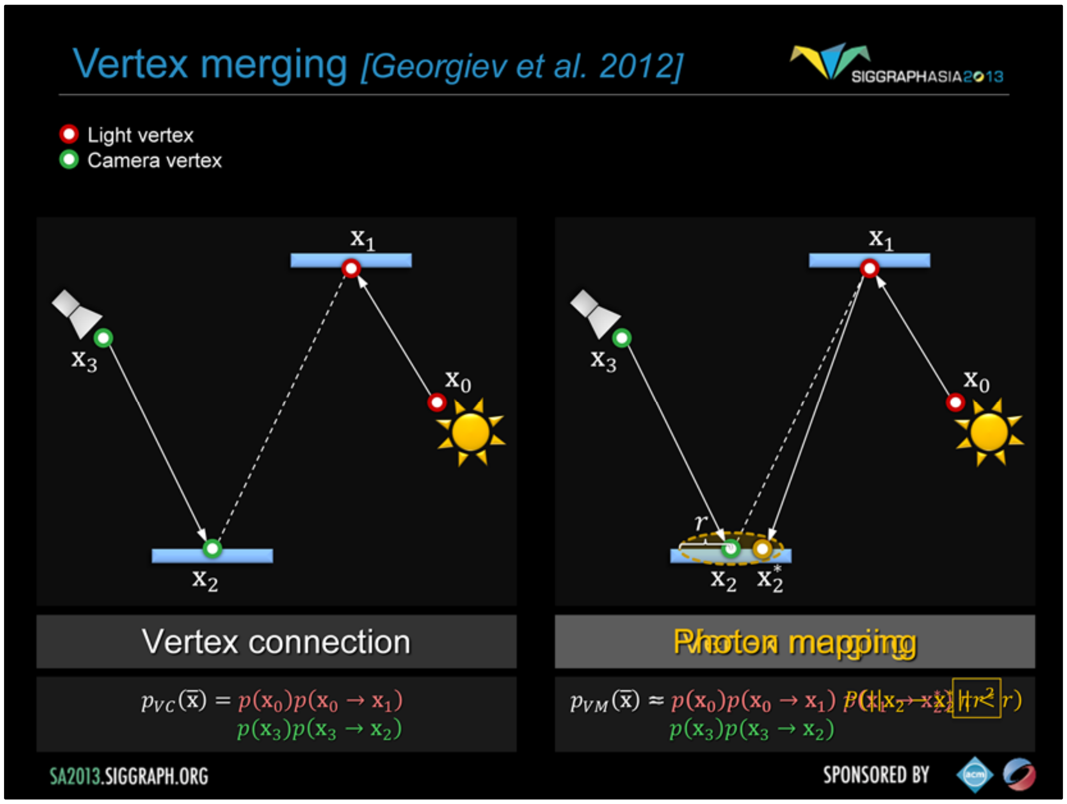
$$p_{DE}(\bar{x}) = p(x_0)p(x_0 \to x_1)\, p(x_1 \to x_2^*) \; p(x_3)p(x_3 \to x_2)$$

SA2013.SIGGRAPH.ORG   SPONSORED BY

This is done by considering a random perturbation of vertex $\mathbf{x}_2$ in an r-neighborhood that yields a new vertex $\mathbf{x}_2^*$. This vertex is then connected to $\mathbf{x}_1$. With this modification, both algorithms sample paths in the same higher-dimensional space. In the case of unidirectional sampling, the camera subpath tracing continues from $\mathbf{x}_2^*$.

[CLICK] The path pdf of the extended vertex connection technique is the product of the pdfs of the individual vertices, where the pdf of $\mathbf{x}_2^*$ is $\frac{1}{\pi r^2}$, as it is sampled uniformly in a circle with a radius r. [CLICK] The pdf of the corresponding photon mapping technique is the product of the vertex pdfs, which are given by the random walk sampling procedure.

Vertex merging *[Georgiev et al. 2012]*



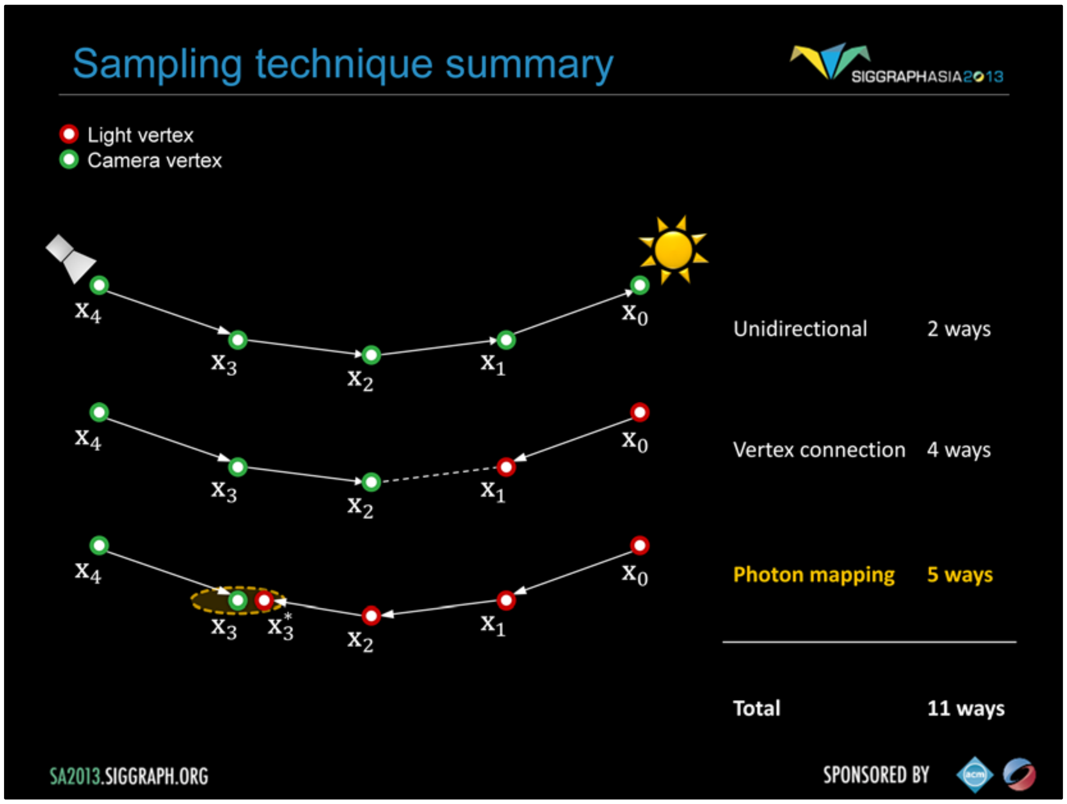
$$p_{VC}(\bar{x}) = p(x_0)p(x_0 \to x_1)\, p(x_3)p(x_3 \to x_2)$$

$$p_{VM}(\bar{x}) \approx p(x_0)p(x_0 \to x_1)\, P(\|x_2 - x_2^*\| < r)\, p(x_3)p(x_3 \to x_2)$$

In contrast, the vertex merging interpretation of Georgiev et al. [2012] considers [CLICK] the last step in photon mapping as establishing a regular vertex connection between $\mathbf{x}_1$ and $\mathbf{x}_2$, but conditioning its acceptance on the random event that a vertex $\mathbf{x}_2^*$ sampled from $\mathbf{x}_1$ lands within a distance r to $\mathbf{x}_2$. This probabilistic acceptance is simply a Russian roulette decision. The full path pdf is then the product of the subpath pdf multiplied by the probability of sampling the point $\mathbf{x}_2^*$ within a distance r of $\mathbf{x}_2$. This acceptance probability is equal to the integral of the PDF of $\mathbf{x}_2^*$ over the r-neighborhood of $\mathbf{x}_1$.

[CLICK] Under the assumptions that the surface around $\mathbf{x}_1$ is locally flat, i.e. that the r-neighborhood is a disk, and that the density of $\mathbf{x}_2^*$ is constant inside this disc, the integral can be well approximated by the PDF of the actual point $\mathbf{x}_2^*$ we have sampled, multiplied by the disc area πr^2. [CLICK] This technique is called *vertex merging*, as it can be intuitively thought to weld the endpoints of the two subpaths if they lie close to each other.

Now that we have common definitions of the paths and their corresponding pdfs in both BPT and PM, we can use the balance heuristic to compute path weights that account for all possible ways to sample the same path in both algorithms. [CLICK] Note that the only difference in the path pdfs given by the two different interpretations is the πr^2 factor in the vertex merging, which appears as a denominator in the extended vertex connection of Hachisuka et al. [2012]. Interestingly, both interpretations result in the same path weights when the path pdfs are plugged into the balance heuristic formula.



Sampling technique summary

SIGGRAPHASIA2013

- 🔴 Light vertex
- 🟢 Camera vertex

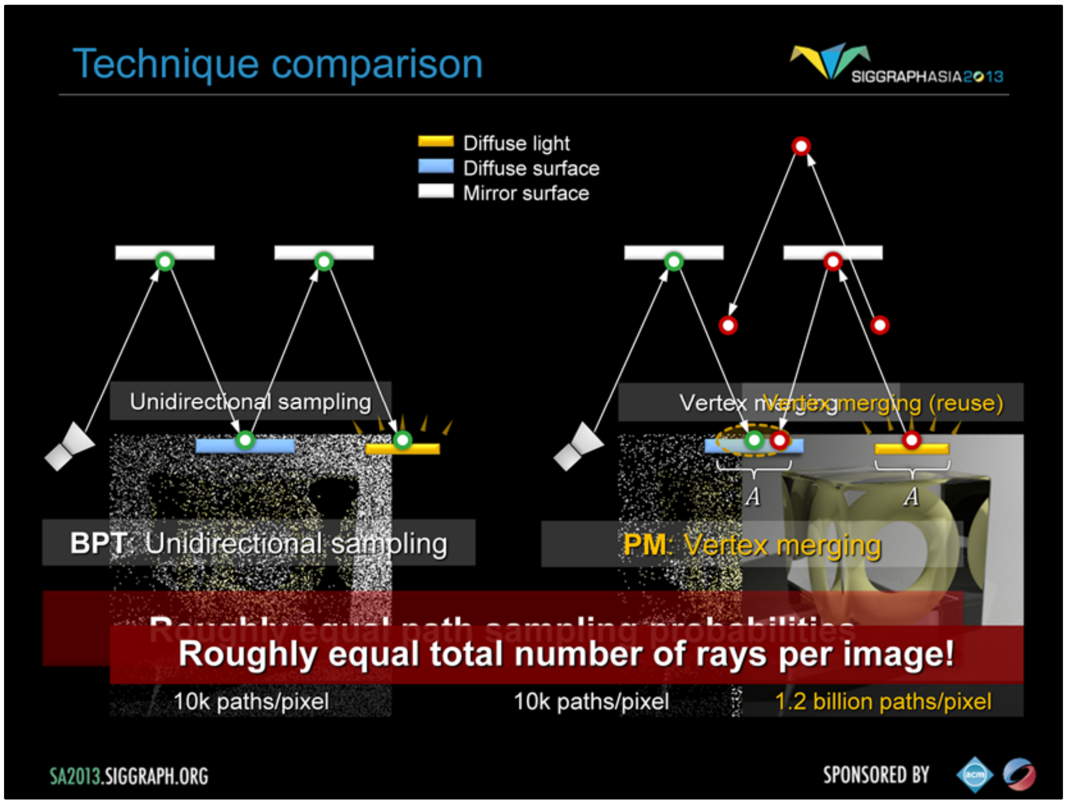
Unidirectional	2 ways
Vertex connection	4 ways
Photon mapping	**5 ways**
Total	**11 ways**

SA2013.SIGGRAPH.ORG

SPONSORED BY

Having formulated photon mapping as a path sampling technique, we can put it side by side with the techniques in BPT. There are two ways to sample a length-4 path unidirectionally, and four ways to sample it via vertex connection. Photon mapping adds five new ways to sample the path, corresponding to merging at the five individual path vertices. In practice, we can avoid merging at the light source and the camera, as directly evaluating emission and sensitivity is usually cheap.

With so many ways to sample the same light transport path, an interesting question arises: which technique is the most efficient for which types of paths? In the following discussion, we will use the vertex merging interpretation to argue about the relative efficiency of BPT and PM.



Technique comparison

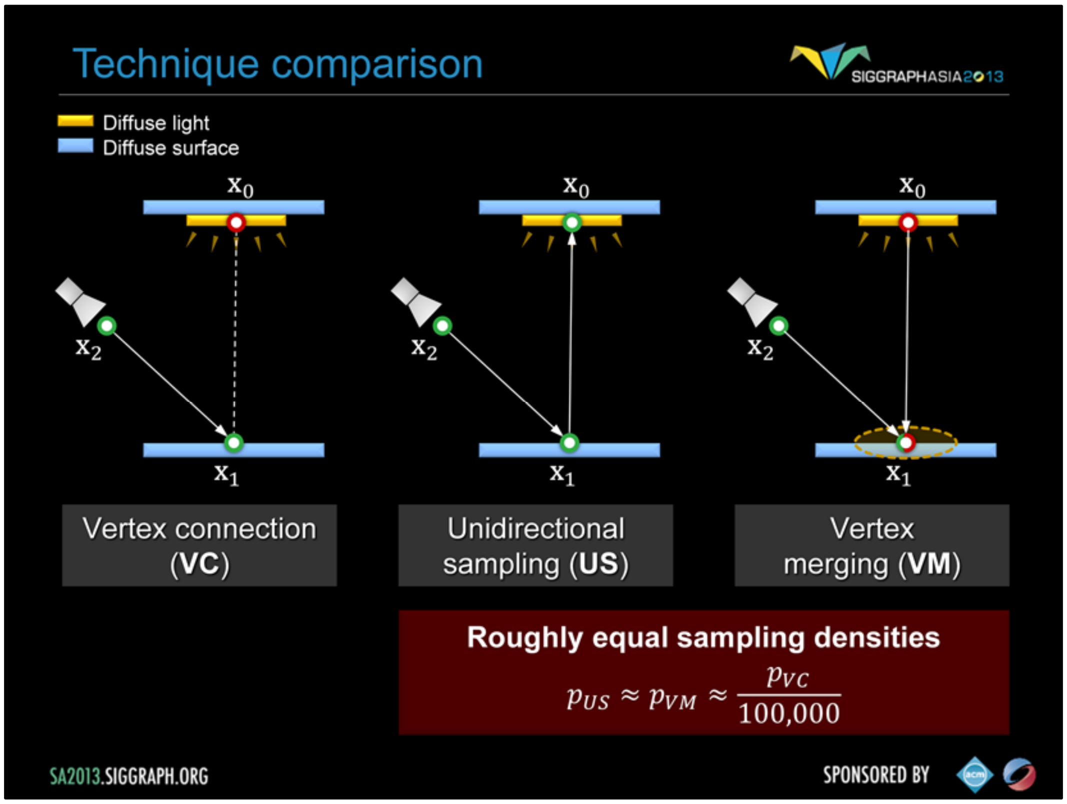
Let us first take a look at specular-diffuse-specular (SDS) paths. Here, BPT can only rely on unidirectional sampling: it traces a path from the camera hoping to randomly hit the light source. With vertex merging, we can trace one light and one camera subpath, and merge their endpoints on the diffuse surface.

[CLICK] It can be shown that if the light source and the merging disk have the same area A, then unidirectional sampling and vertex merging sample paths with roughly the same probability density. This means that we should expect the two techniques to perform similarly in terms of rendering quality.

[CLICK] We render these two images progressively, sampling one full path per pixel per iteration. For the left image we trace paths from the camera until termination or hitting the light. For image on the right, we trace subpaths from both ends, and concatenate them if their endpoints if they lie within a distance $r = \sqrt{A/\pi}$ from each other. Both images look equally noisy, even with 10,000 paths per pixel. This confirms that the path sampling technique used in photon mapping is *not* intrinsically more robust for SDS paths than unidirectional sampling.

[CLICK] However, the strength of photon mapping is computational efficiency – we can very efficiently reuse the light subpaths traced for *all* pixels at the cost of a single range search query. This allows us to quickly construct orders of magnitude more light transport estimators from the same sampling data, with a minimal computational overhead, resulting in a substantial quality improvement.

[CLICK] For all these three images we have traced roughly the same number of rays, and the only difference between the one in the center and the one on the right is that the for right image we have enabled path reuse, by storing, and looking up, the light subpath vertices in a photon map at every rendering iteration.



Technique comparison

Roughly equal sampling densities

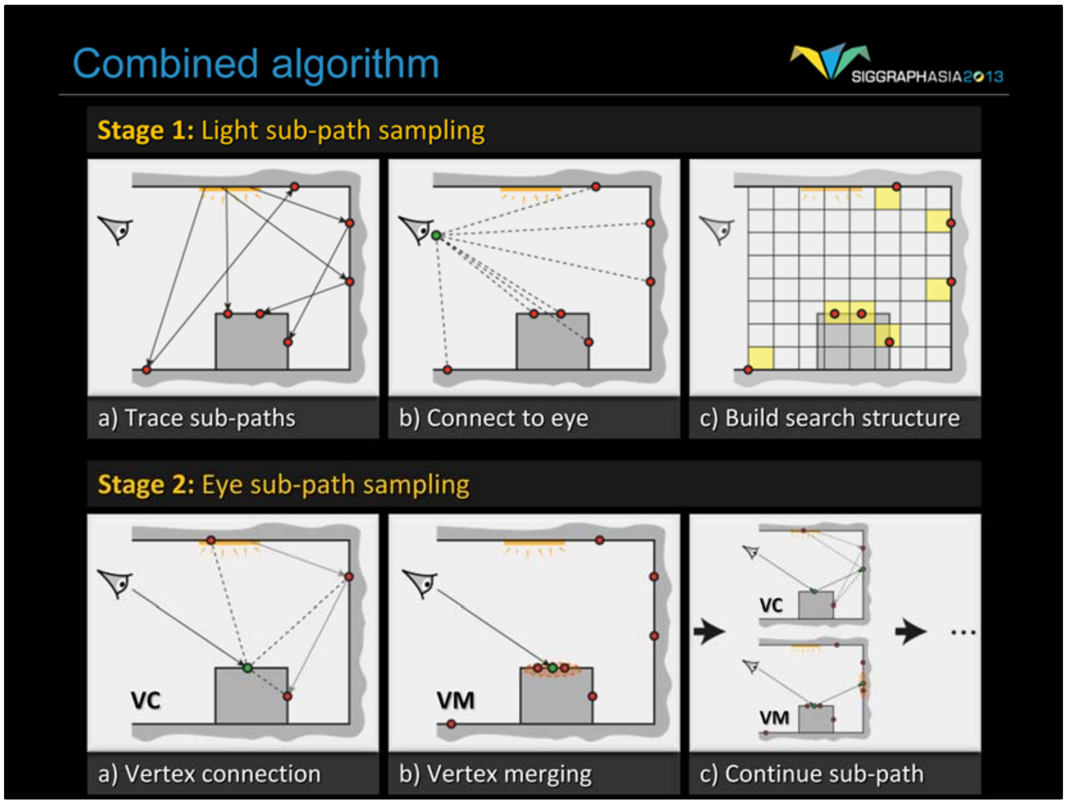
$$p_{US} \approx p_{VM} \approx \frac{p_{VC}}{100{,}000}$$

Now let us look at another extreme example – diffuse illumination. Note that vertex connection (VC) constructs the edge between $\mathbf{x}_1$ and $\mathbf{x}_2$ deterministically, while unidirectional sampling (US) and vertex merging (VM) both rely on random sampling.

Once again, it can be shown that if the light source and the merging disk have the same area, then US and VM sample this path with roughly the same probability density.

[CLICK] For the specific case shown on this slide, this density is about 100,000 lower than that of VC. This demonstrates that VM is not an intrinsically more robust sampling technique than VC either. This is not surprising – if we recall the expression for the VM path PDF, we see that it can only be lower than that of the corresponding VC technique, as their only difference is the probability factor in the VM PDF, which is necessarily in the range $[0; 1]$. Still, by reusing paths across pixels, photon mapping gains a lot of efficiency over unidirectional sampling.

All these insights emerge from the reformulation of photon mapping as a path sampling technique.



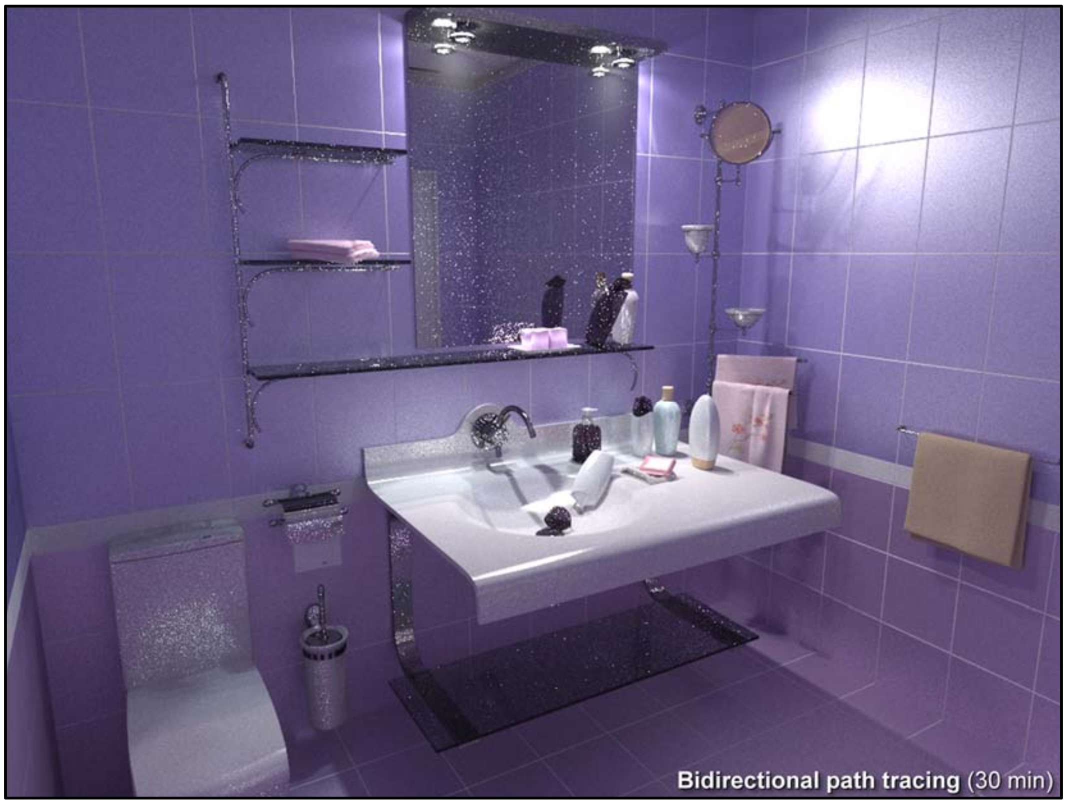
Even more importantly, we now have the necessary ingredients for combining photon mapping and bidirectional path tracing into one unified algorithm. The vertex merging path PDFs tell us how to weight all sampling techniques in multiple importance sampling, and the insights from the previous two slides command to strive for path reuse.

The combined algorithm operates in two stages.

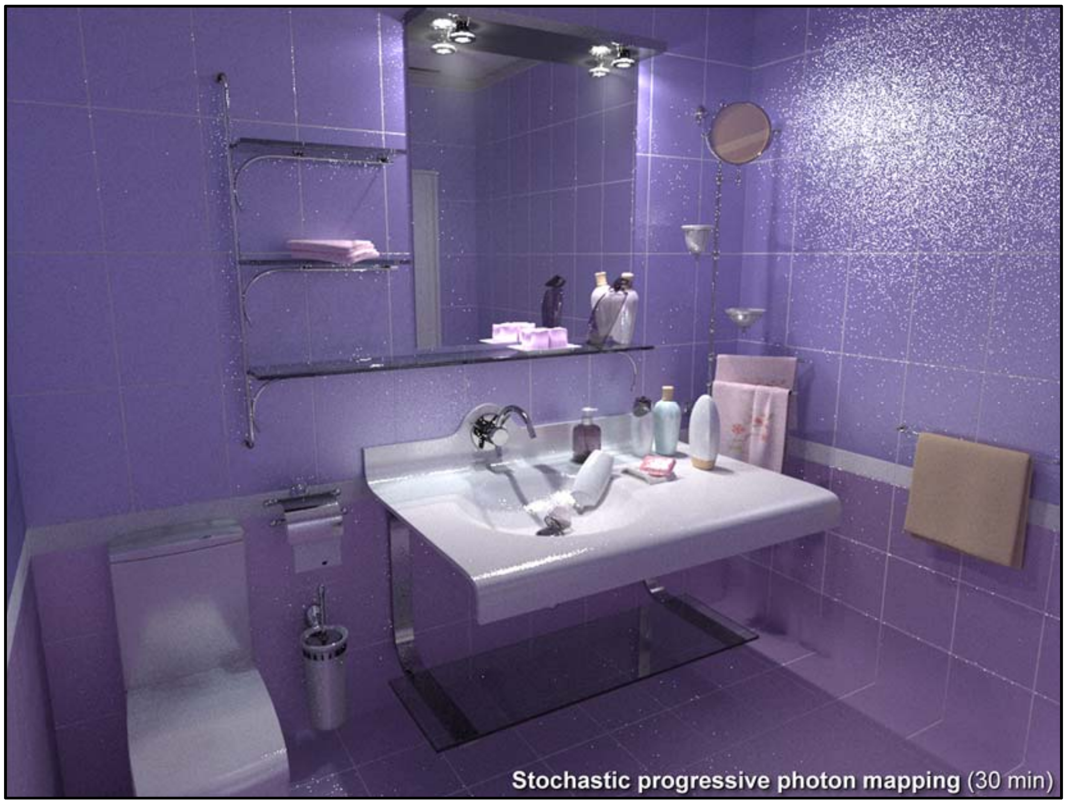
1. In the first stage, we
   a) [CLICK] trace the light subpaths for all pixels,
   b) [CLICK] connect them to the camera, and
   c) [CLICK] store them in a range search acceleration data structure (e.g. a kd-tree or a hashed grid).

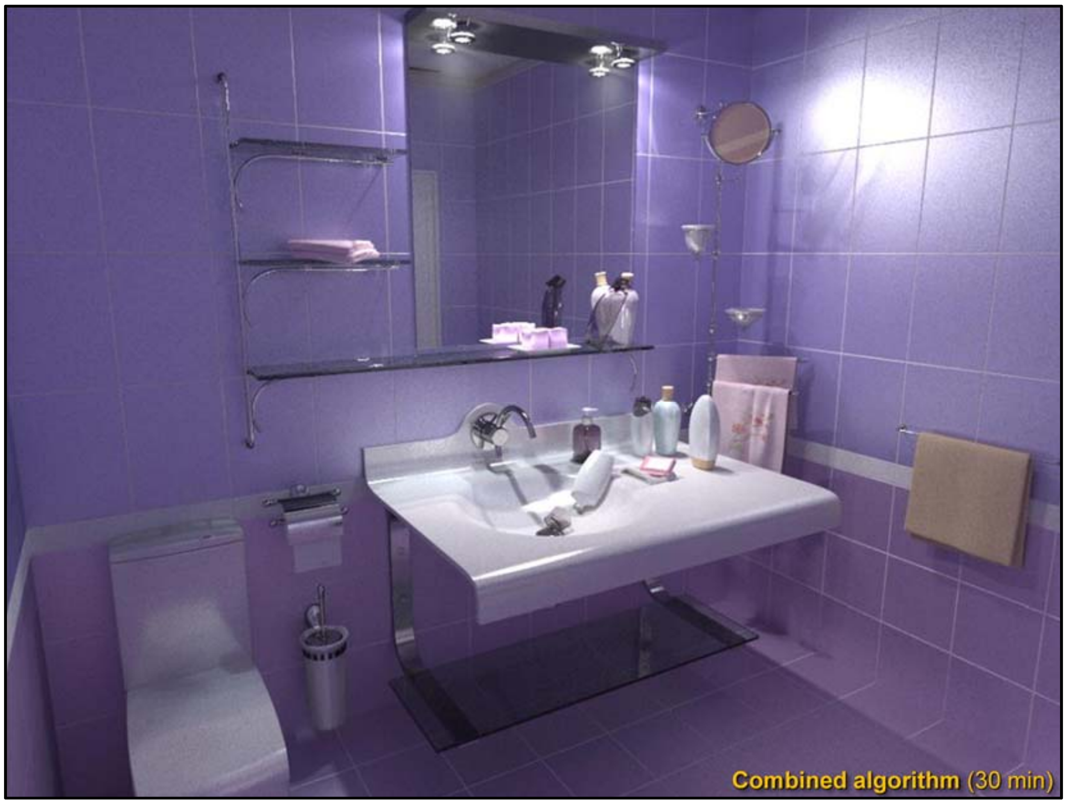
2. [CLICK] In the second stage, we trace a camera subpath for every pixel.
   a) [CLICK] Each sampled vertex on this path is connected to a light source (a.k.a. next event estimation), connected to the vertices of the light subpath corresponding to that pixel, and
   b) [CLICK] merged with the vertices of *all* light subpaths.
   c) [CLICK] We then sample the next vertex and do the same.

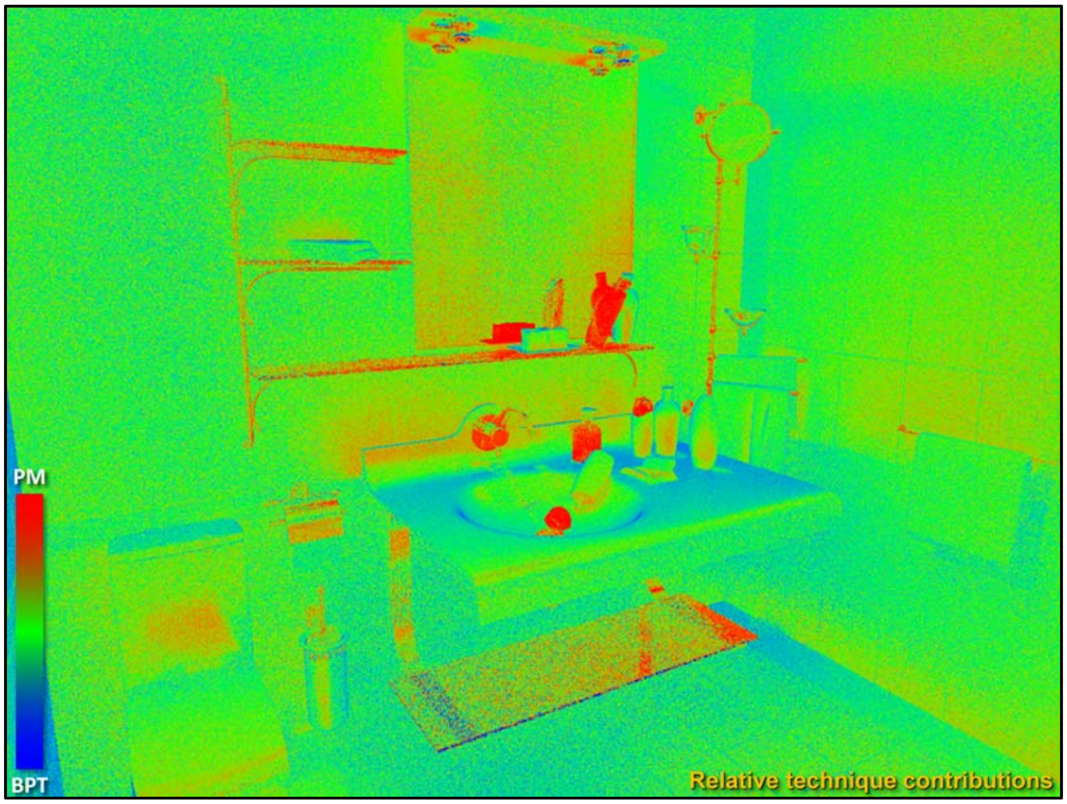
In a progressive rendering setup, we perform these steps at each rendering iteration, reducing the vertex merging radius.



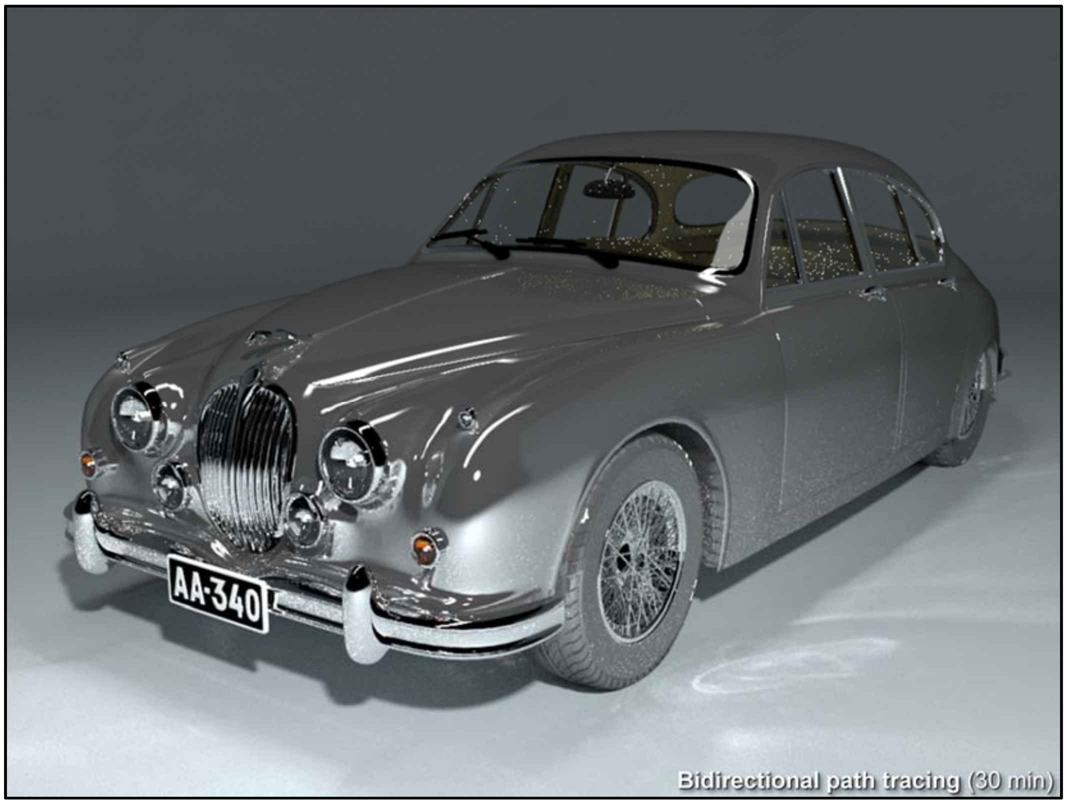
Let us now see how this combined algorithm stacks up against bidirectional path tracing and stochastic progressive photon mapping on a number of scenes with complex illumination.

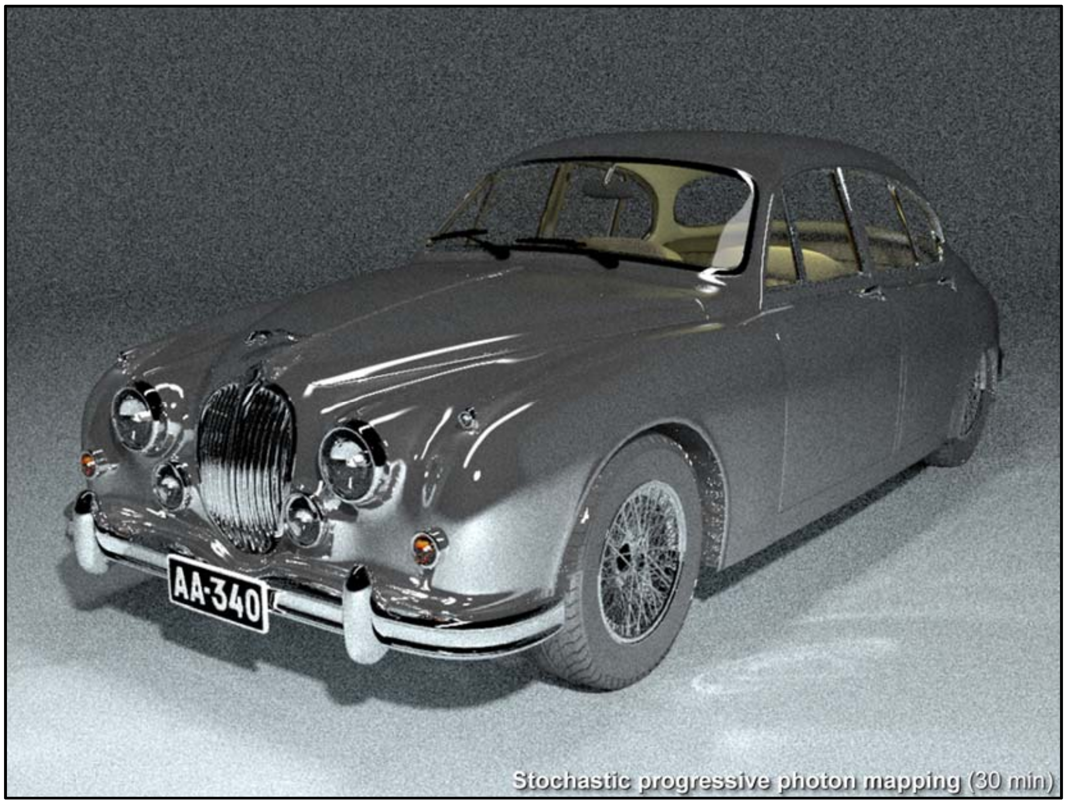

Stochastic progressive photon mapping (30 min)

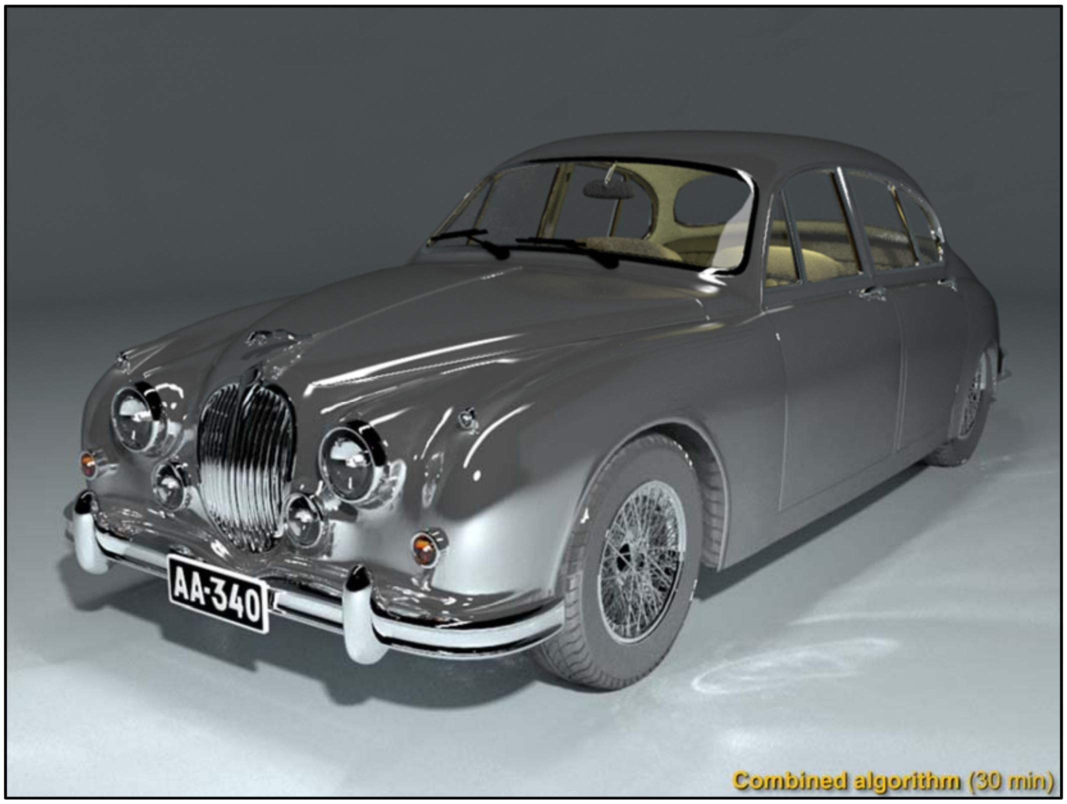

Combined algorithm (30 min)

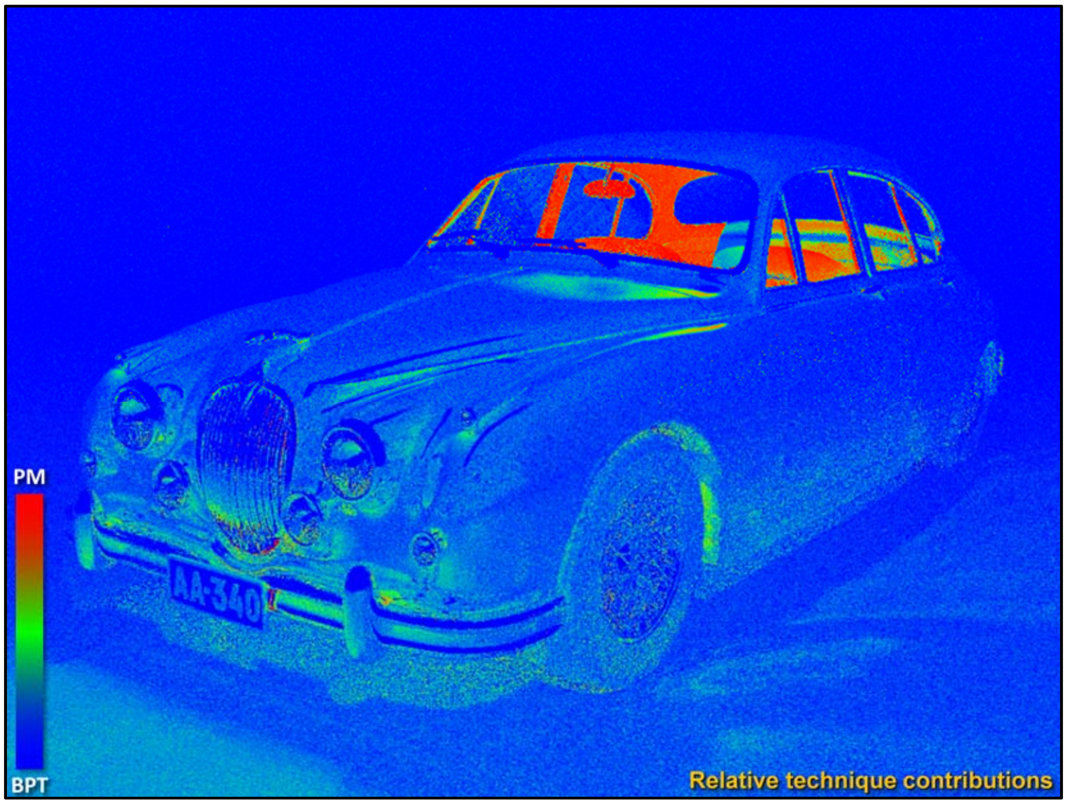


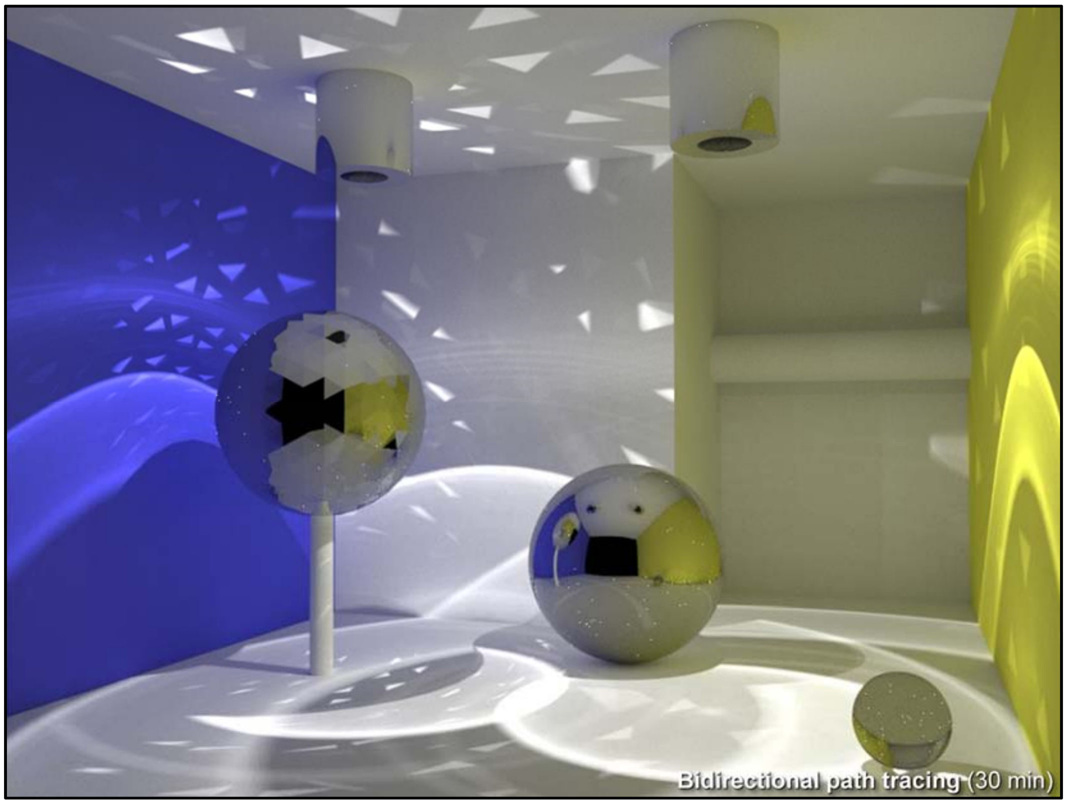
Here, we visualize the relative contributions of BPT and PM techniques to the combined image from the previous slide. This directly corresponds to the weights that the combined algorithm assigned to these techniques.

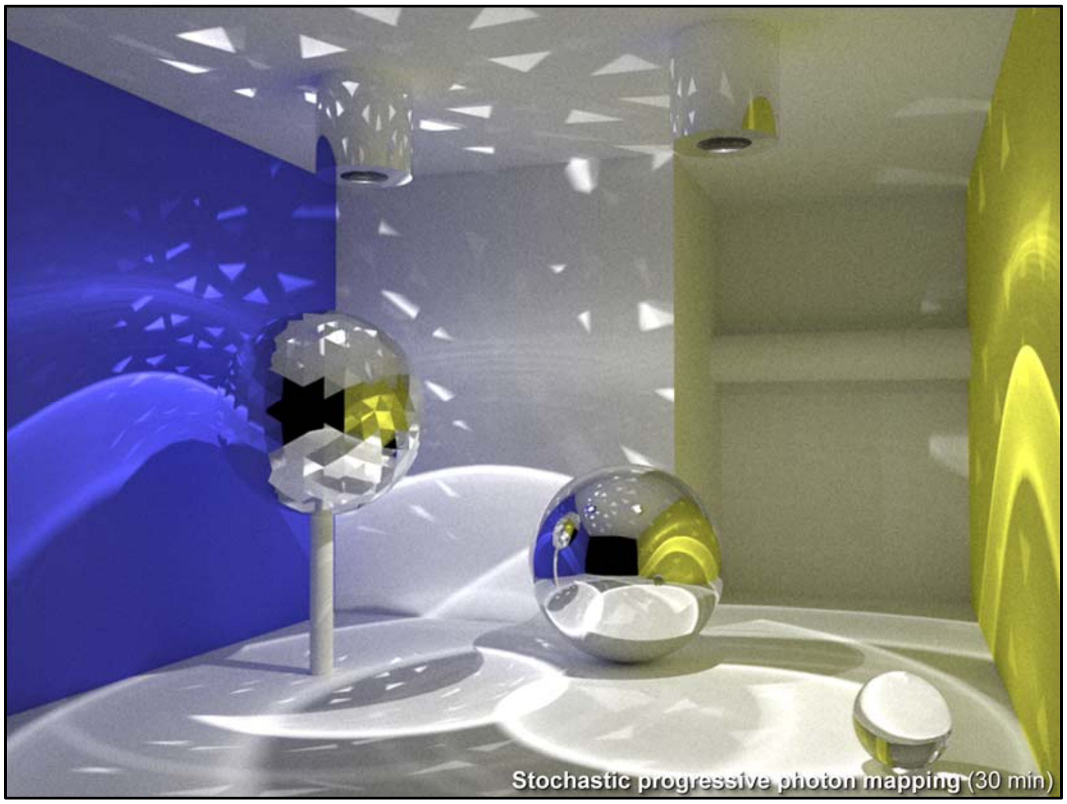

Bidirectional path tracing (30 min)

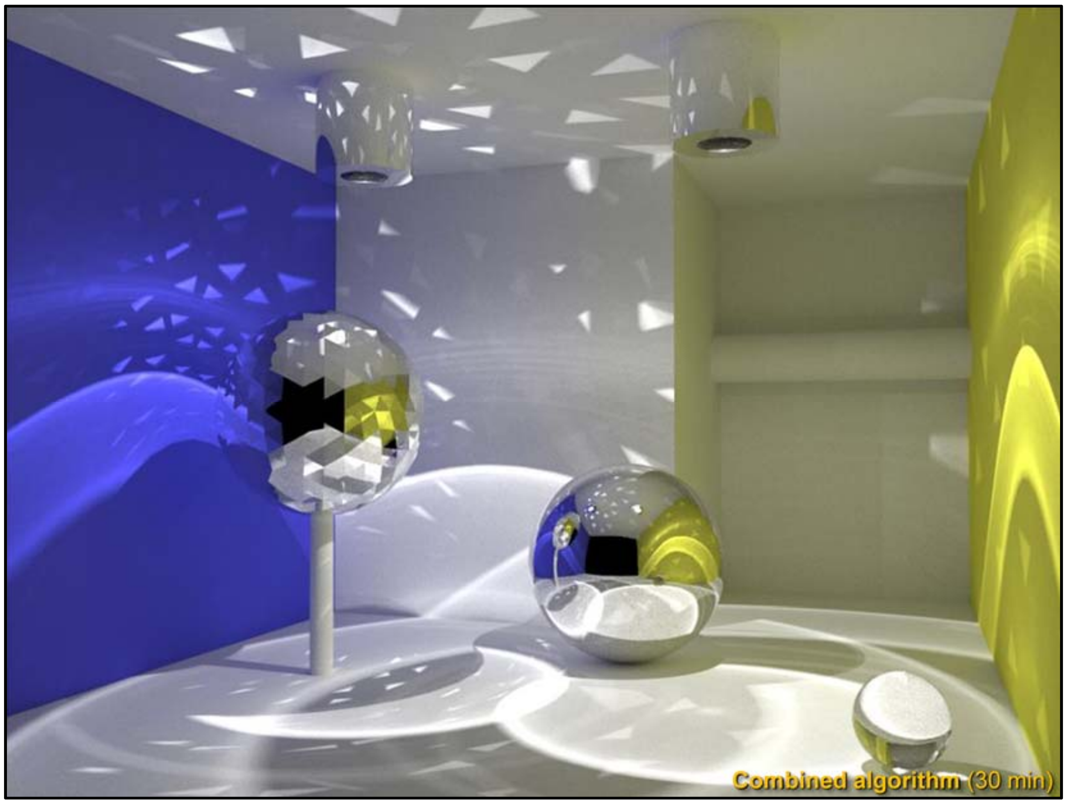

AA-340
Stochastic progressive photon mapping (30 min)

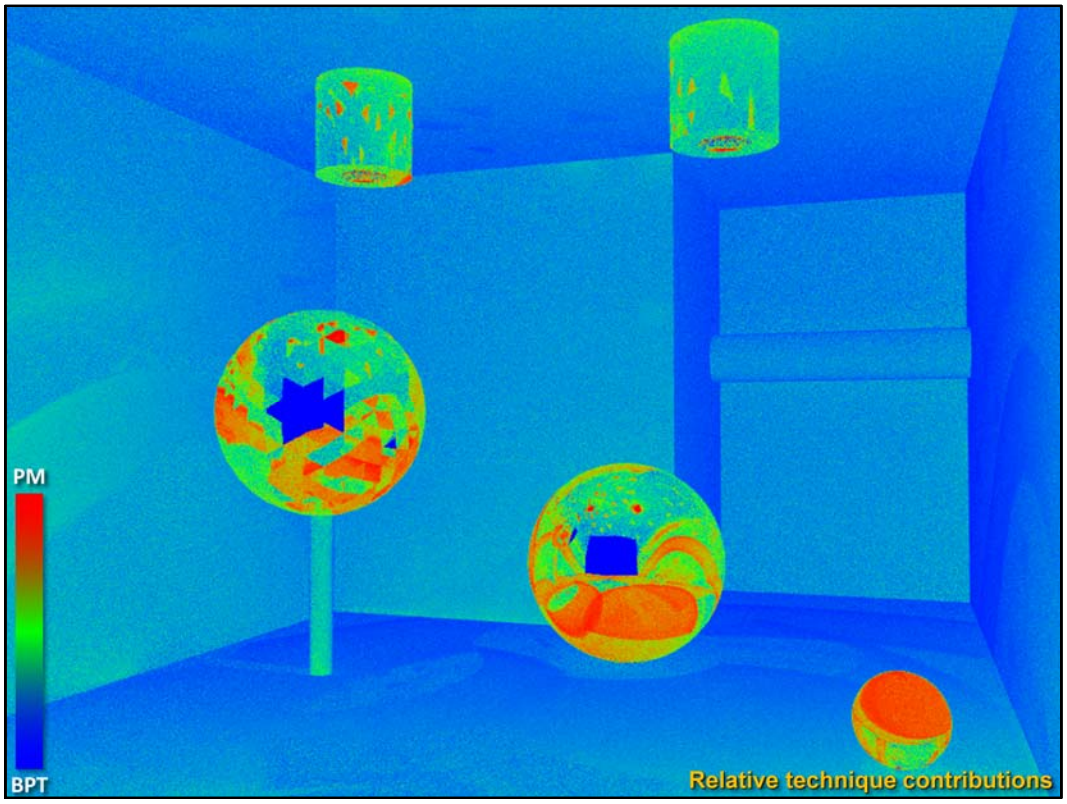

AA-340
Combined algorithm (30 min)

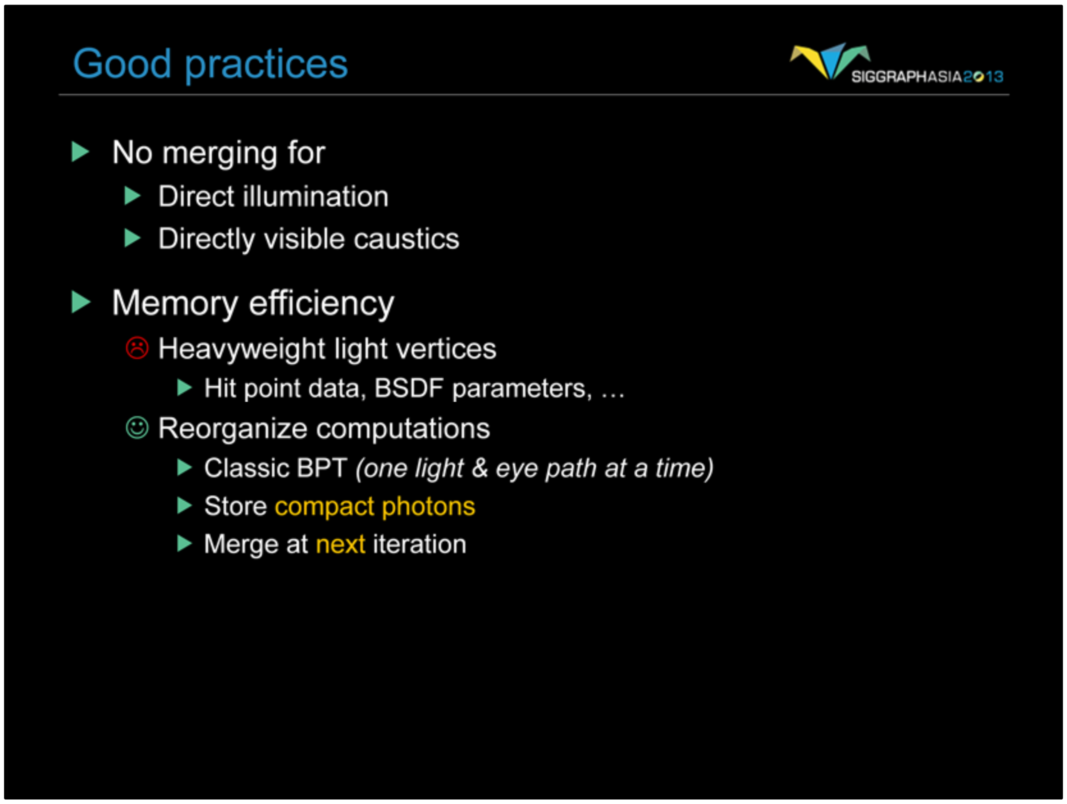

PM
BPT
AA-340
Relative technique contributions


Bidirectional path tracing (30 min)


Stochastic progressive photon mapping (30 min)


Combined algorithm (30 min)


PM
BPT
Relative technique contributions



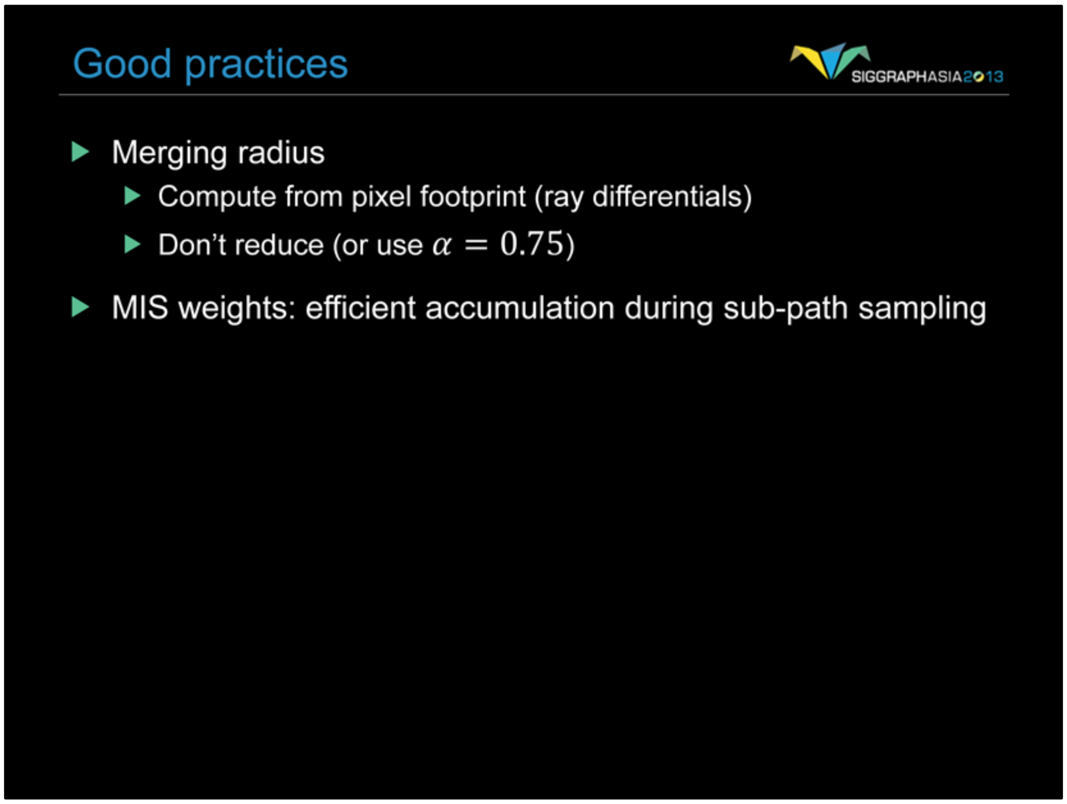
Good practices

- No merging for
  - Direct illumination
  - Directly visible caustics
- Memory efficiency
  - ☹ Heavyweight light vertices
    - Hit point data, BSDF parameters, …
  - ☺ Reorganize computations
    - Classic BPT *(one light & eye path at a time)*
    - Store compact photons
    - Merge at next iteration

I will now discuss some good practices for the practical implementation of the combined algorithm in a production renderer.

We can practically always skip vertex merging (VM) for direct illumination and directly visible caustics, as vertex connection usually performs better. Doing this also avoids the correlated noise and bias inherent to VM.

The combined algorithm has the problem that its memory footprint can be much larger than that of (progressive) photon mapping. This comes from using the stored light vertices not only for merging, but for connection as well. We need to store hit point data with these vertices, which includes coordinate frame, BSDF structure, and possibly other data, making the vertex footprint as large as 1KB in some cases. This results in 1GB of storage for 1 million vertices. For the progressive variant of the combined algorithm, we can dramatically reduce this footprint by rearranging its computations. We follow the classical BPT implementation, where for each pixel we trace one light subpath and one camera subpath. After performing the vertex connections, we store the light subpath vertices in the acceleration structure. These vertices will then be used for merging at the *next* rendering iteration, while the camera subpath vertices for the current iteration are merged with the light vertices from the *previous* iteration. This allows us to safely strip the hit point data off the light vertices before storing them in the structure, as they will be only used for merging. And just like in photon mapping, for merging we only need to keep around a small amount of vertex data, i.e. position, direction, and throughput.

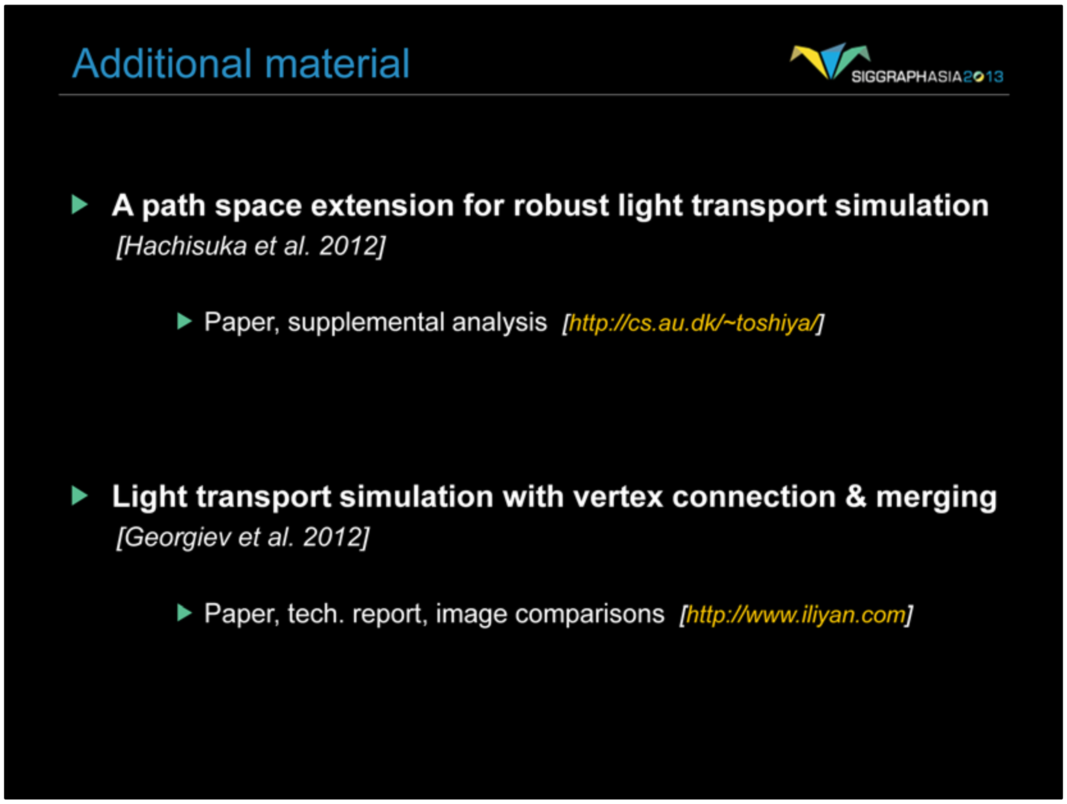


Good practices

- Merging radius
  - Compute from pixel footprint (ray differentials)
  - Don't reduce (or use $\alpha = 0.75$)
- MIS weights: efficient accumulation during sub-path sampling

The progressive combined algorithm has two parameters: the initial merging radius r and its reduction rate α (see "Progressive Photon Mapping" [Hachisuka et al. 2008]). We can often afford to use a much smaller radius than in PPM, as VCM mixes a large number of path sampling techniques, and VM is mostly used for caustics. An automatic and robust way to compute a good merging radius for each camera path is to derive it from the pixel footprint, which is given by the ray differentials that most renderers already implement and use for texture filtering. As for the reduction parameter α, I recommend using $\alpha = 0.75$, which is a provably good value (see VCM paper [Georgiev et al. 2012]), Alternatively, we can also opt to not reduce the radius altogether (i.e. setting $\alpha = 1$), especially when using ray differentials which give a small enough radius to mostly avoid noticeable correlated noise and bias.

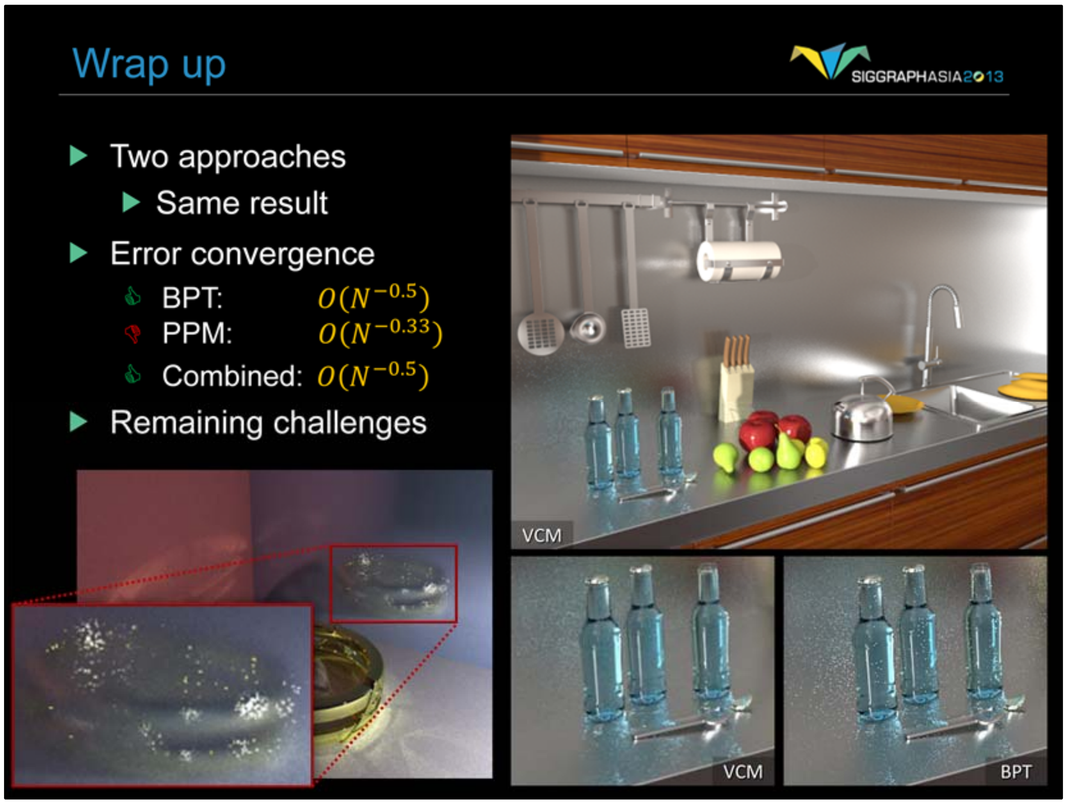
Efficient MIS weight computation is another important practical aspect of the combined algorithms, and fortunately there exists a scheme for cumulative computation of weight data during the subpath random walks. This scheme is discussed in length in the technical report cited on the next slide.



Additional material

SIGGRAPHASIA2013

- **A path space extension for robust light transport simulation** *[Hachisuka et al. 2012]*
  - Paper, supplemental analysis *[http://cs.au.dk/~toshiya/]*
- **Light transport simulation with vertex connection & merging** *[Georgiev et al. 2012]*
  - Paper, tech. report, image comparisons *[http://www.iliyan.com]*



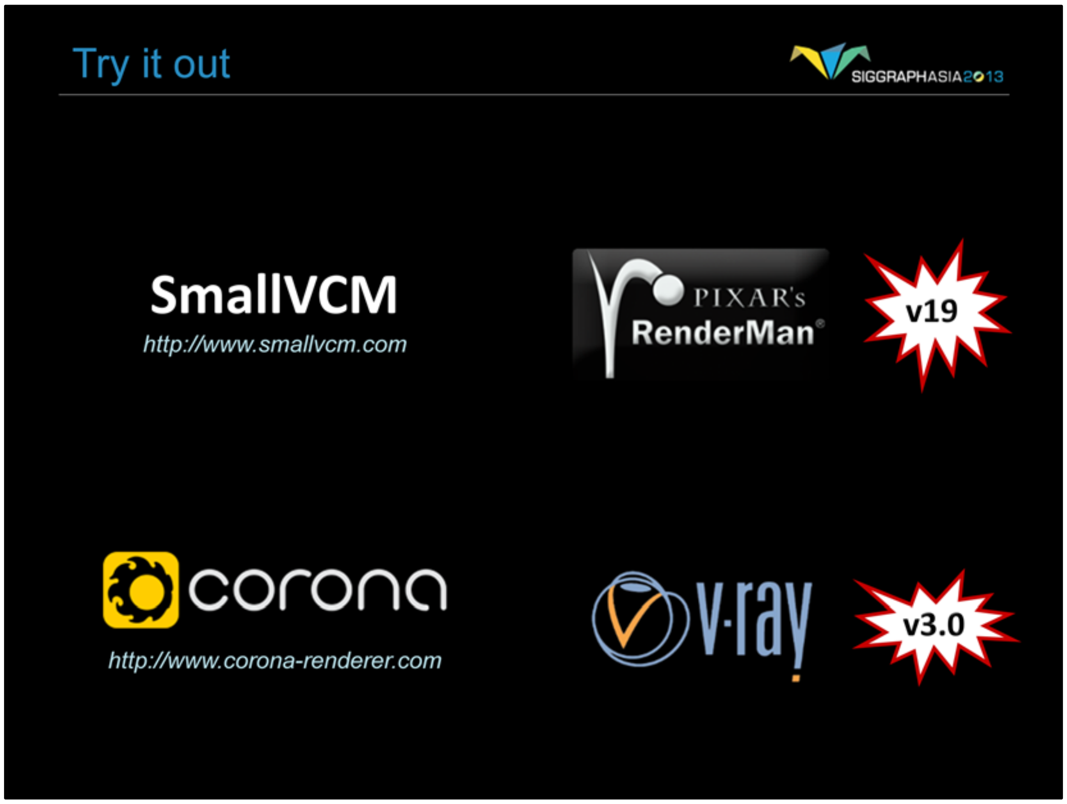
Wrap up

- Two approaches
  - Same result
- Error convergence
  - BPT: $O(N^{-0.5})$
  - PPM: $O(N^{-0.33})$
  - Combined: $O(N^{-0.5})$
- Remaining challenges

SIGGRAPHASIA2013
VCM
VCM
BPT

Two independent groups of researchers have taken different approaches on efficiently combining BPT and PM. The result of both these efforts is the same algorithm, which importantly also inherits the higher asymptotic error convergence rate of BPT, meaning that it approaches the correct solution faster than PPM as we spend more computational effort (i.e. sample more paths). The asymptotic analysis can be found in the VCM paper [Georgiev et al. 2012].

[CLICK] Even though VCM is a step forward in Monte Carlo rendering and has proven very useful in practice, it doesn’t come without limitations. Specifically, it cannot handle more efficiently those types of light transport paths that are difficult for both BPT and PM to sample. [CLICK] A prominent example are caustics falling on a glossy surface. [CLICK] And on this kitchen scene, even though VCM brings practical improvement over BPT, there is still a lot to be desired from the caustics.


Try it out

SIGGRAPHASIA2013

SmallVCM
http://www.smallvcm.com

PIXAR's RenderMan® v19

corona
http://www.corona-renderer.com

V-ray v3.0

If you want to try this algorithm yourself, we have released a reference open source implementation in the SmallVCM renderer.

A number of companies have also announced integration of the algorithm in the upcoming releases of their commercial renderers. One example is Pixar's Photorealistic RenderMan v19, and Chaos Group have already shown first images from V-Ray 3.0. The algorithm is also already implemented in the public Alpha release of Corona renderer.

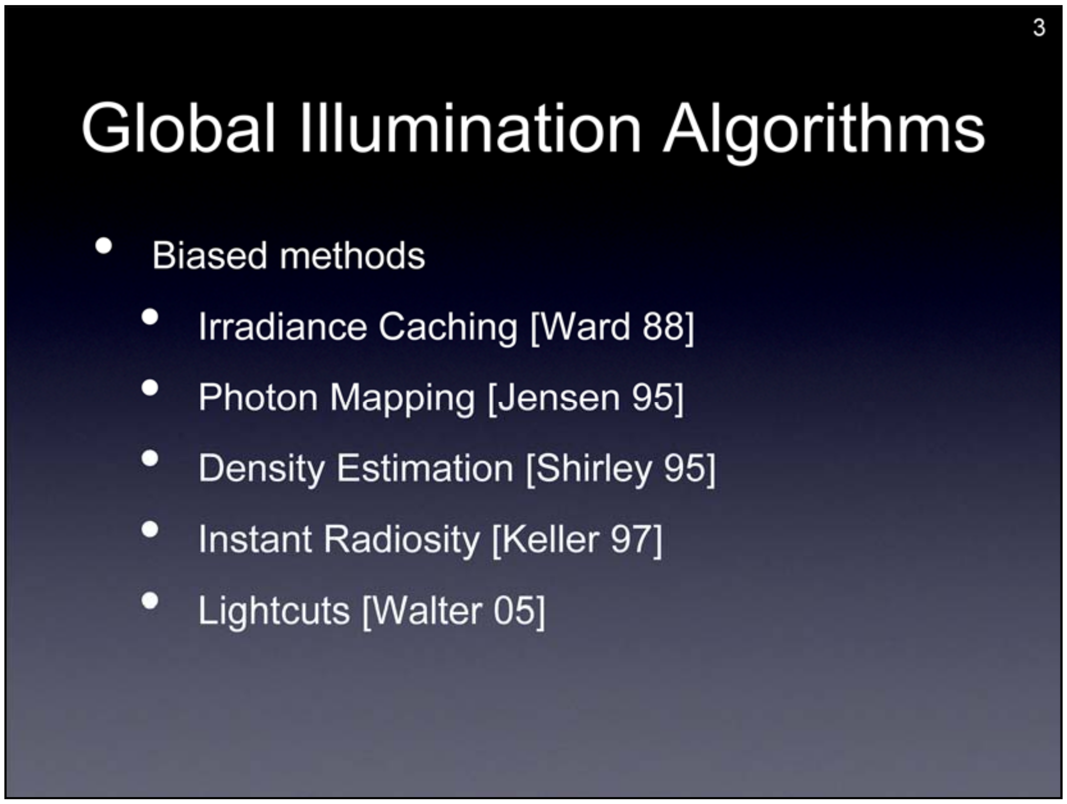
Progressive Photon Mapping

Toshiya Hachisuka[*] Shinji Ogaki[†] Henrik Wann Jensen[*]

[*]University of California, San Diego
[†]University of Nottingham

EXIT
Bunkspeed
Gulfstream interior

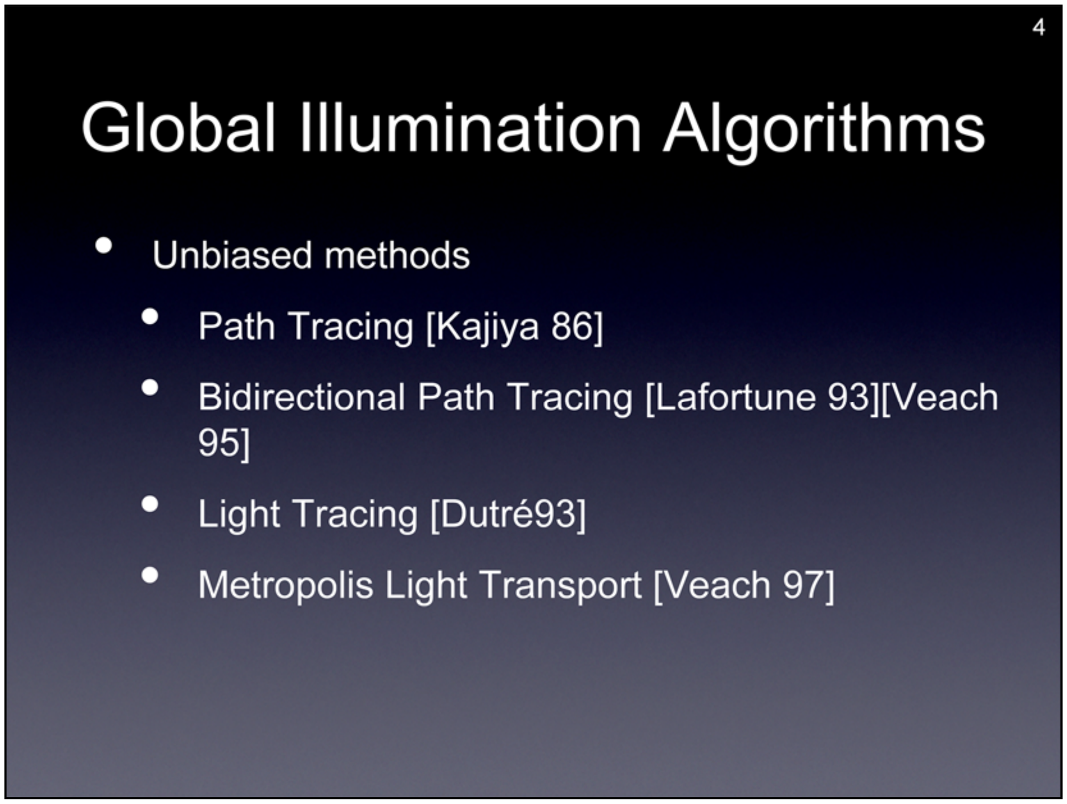
Using recent global illumination techniques, it is possible to render realistic images as shown here.



Global Illumination Algorithms

- Biased methods
  - Irradiance Caching [Ward 88]
  - Photon Mapping [Jensen 95]
  - Density Estimation [Shirley 95]
  - Instant Radiosity [Keller 97]
  - Lightcuts [Walter 05]

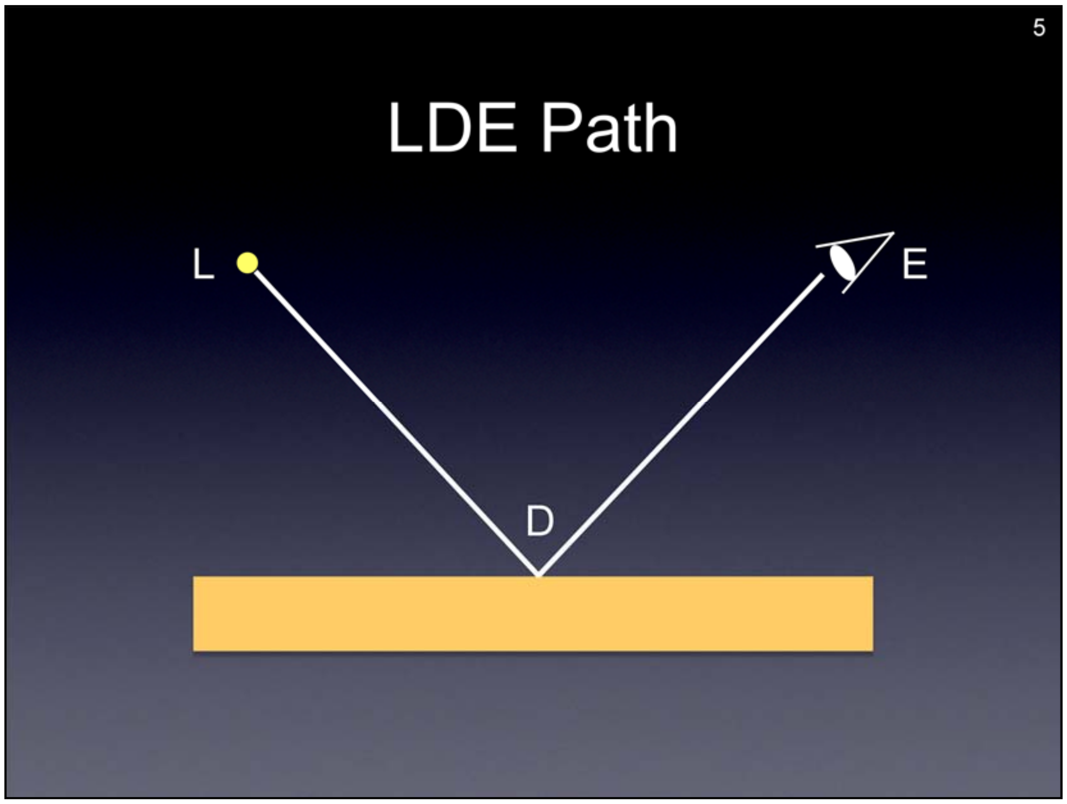
Global illumination techniques are typically based on Monte Carlo method, which are further classified into unbiased methods and biased methods. Biased methods are often faster than unbiased methods, and widely used in many rendering systems. However, images rendered by biased methods contain systematic errors associated with each algorithm.



Global Illumination Algorithms

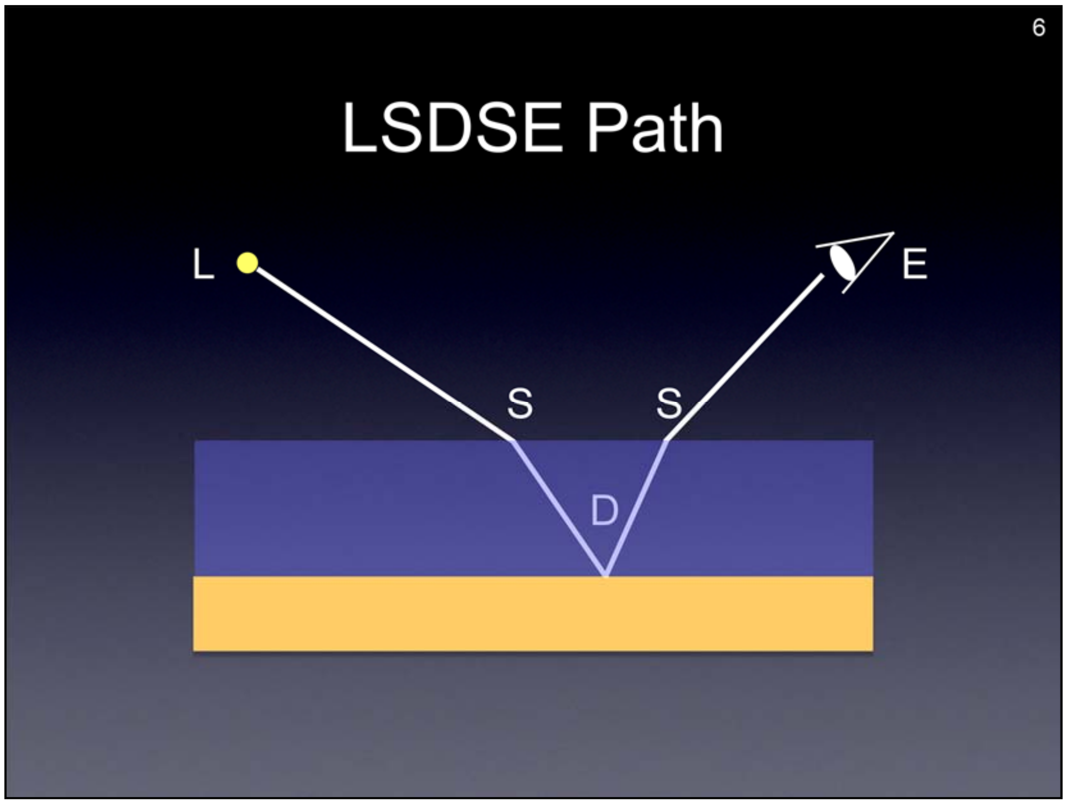
- Unbiased methods
  - Path Tracing [Kajiya 86]
  - Bidirectional Path Tracing [Lafortune 93][Veach 95]
  - Light Tracing [Dutré93]
  - Metropolis Light Transport [Veach 97]

Unbiased methods can compute correct images, in the sense that, it gives the correct solution to the rendering equation on average. If we need a very accurate image, we usually choose an unbiased method because the result can be arbitrarily accurate just by increasing the number of samples. Given all the algorithms, it is natural to think that global illumination is a solved problem. We claim that this is not true.



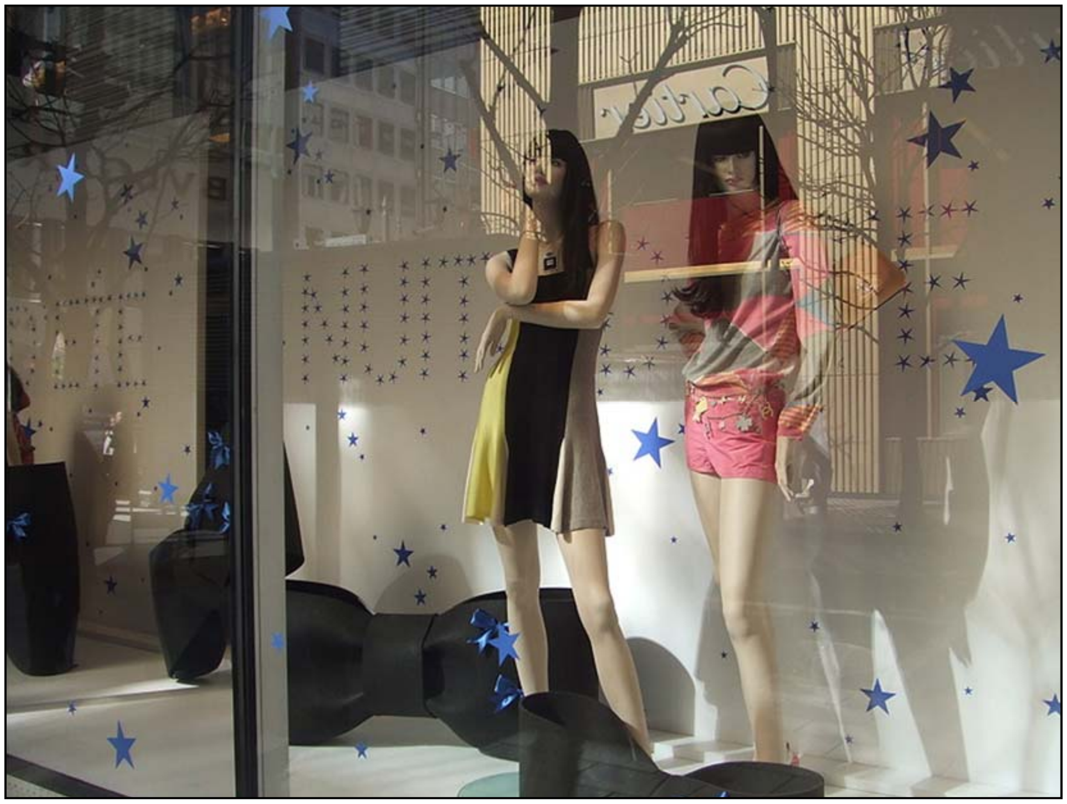
LDE Path

Let’s consider this scene to highlight why we claim global illumination is not solved. This path is called LDE path, where L is a light source, D is a diffuse reflection, and E is the viewer. If we see a diffuse object directly illuminated by a point light source, it is easy to construct a path of light.

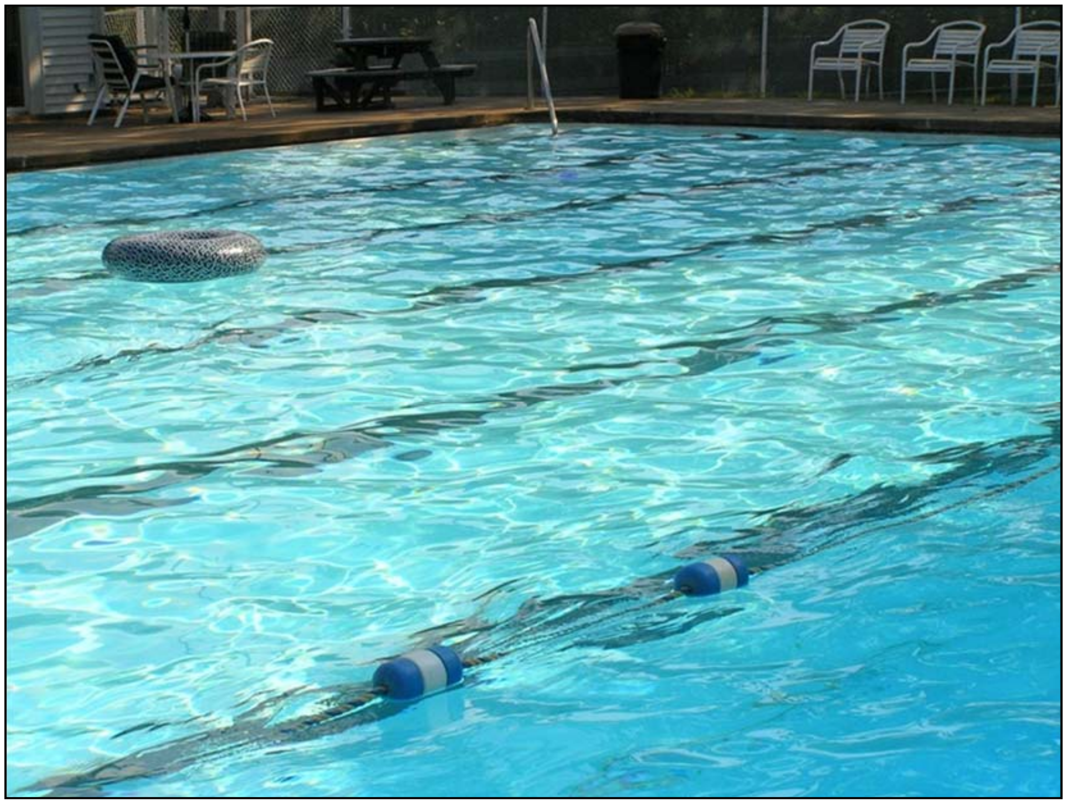


LSDSE Path

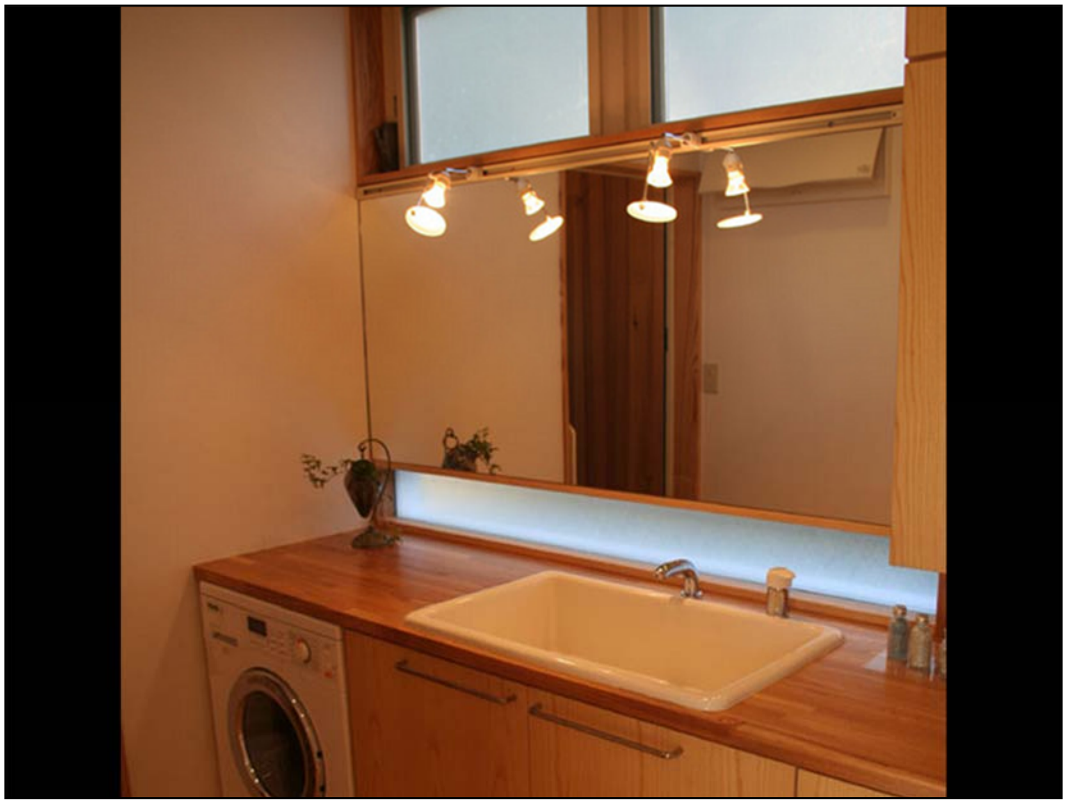
However, just by adding a refractive object on top of the diffuse object, it is no longer easy to construct a light path. The path which connects between the light source and the viewer refracts at the boundary of the refractive object, which is now called LSDSE path where S is a specular refraction. In order to construct a path, we need to find a point on the diffuse object that connects the viewer and the light source after refractions. If the light is a point light source, then it is in fact impossible to compute this path with any unbiased method.



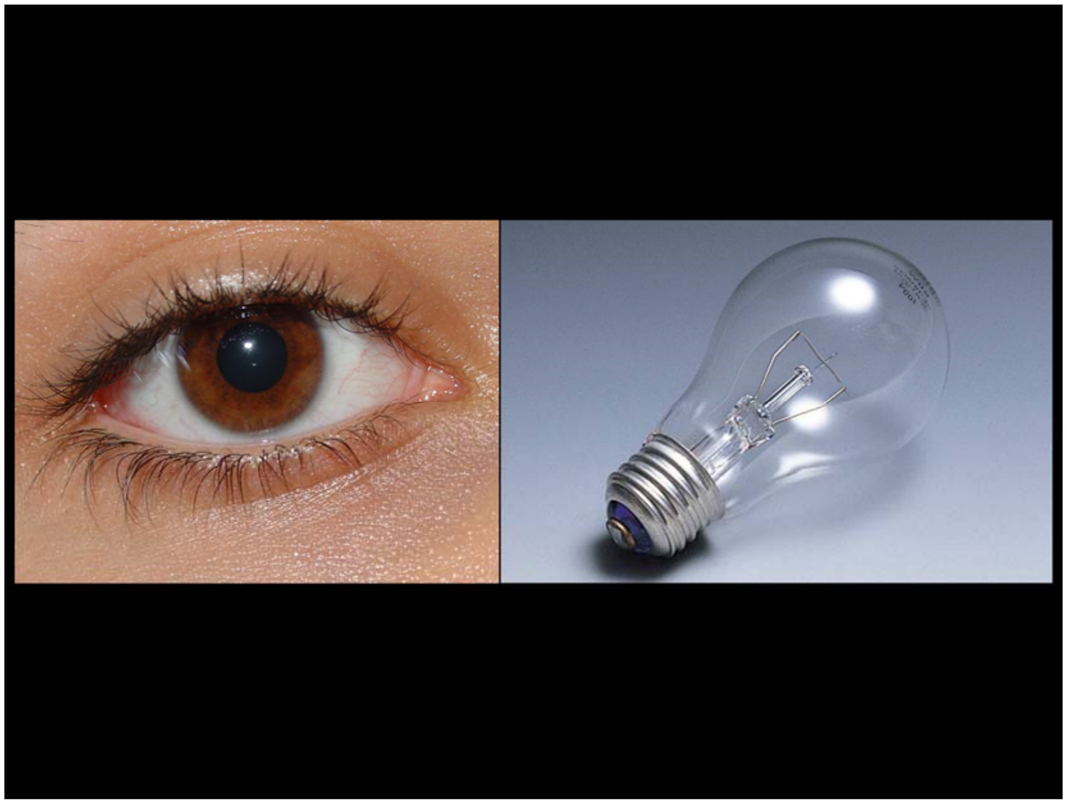
You might think it is a rare case, but we see this type of illumination very often in our daily life. For example, let’s take a look at this simple photograph of a store window. Since the window causes specular refractions and the sunlight is coming through the window, everything you see through the window contains an SDS path.



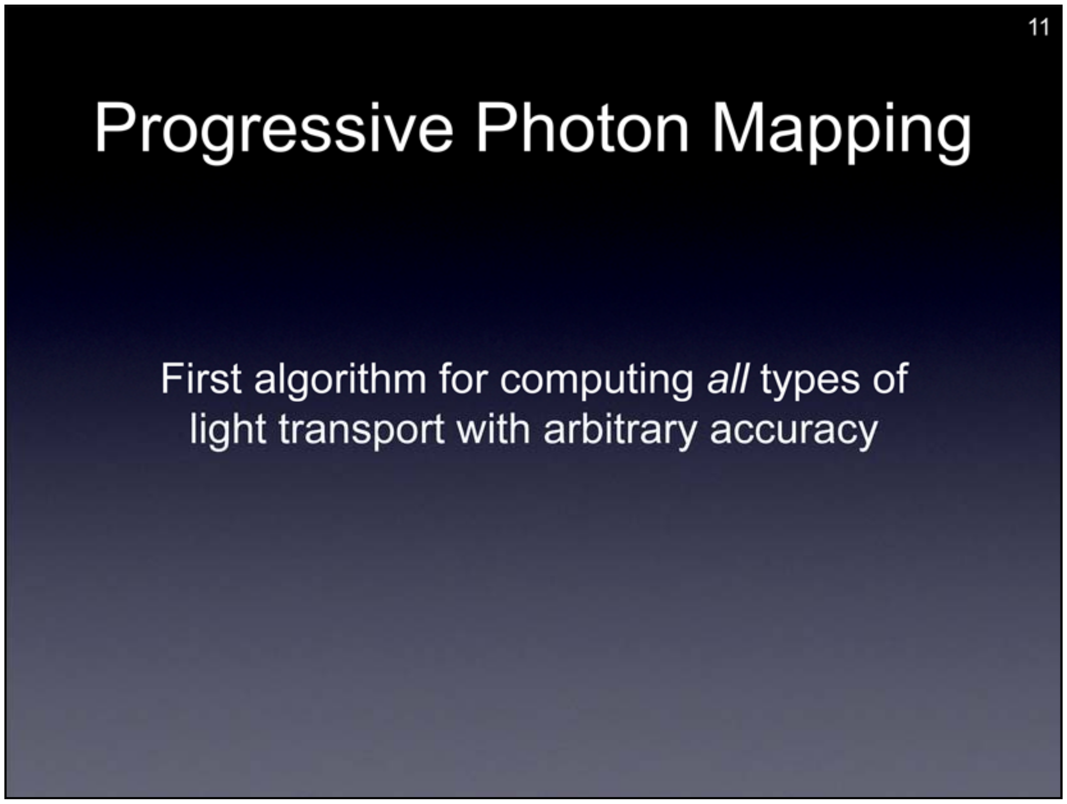
The bottom of a swimming pool is another example of SDS paths, where it is illuminated by light coming through the water surface, and we see that above the water surface.



This is a picture of an ordinary bathroom. Almost everything you see in the mirror is a SDS path. The reason is that a glass casing of light bulb causes specular refraction before illuminating any diffuse surface. Therefore, what we see in the mirror is dominated by SDS paths.



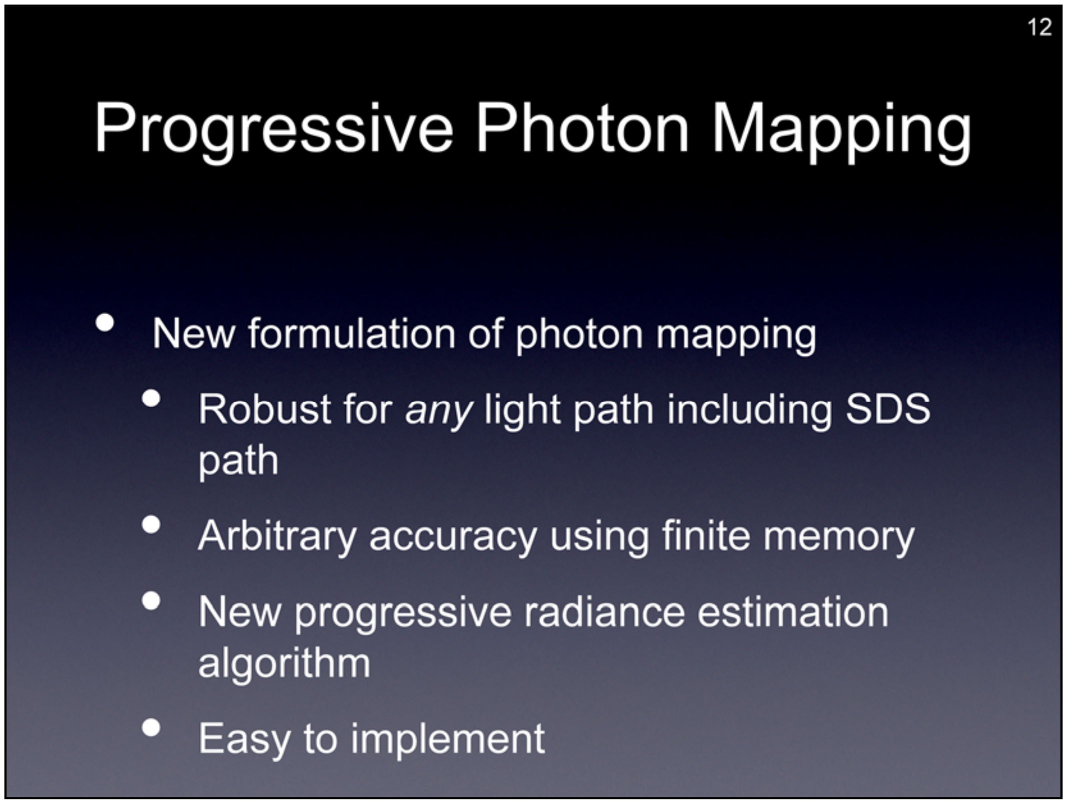
If you want to be extremely precise, you would need to consider that the lens of our eyes or camera and the glass casing around a light bulb, which ultimately create SDS paths everywhere. I hope you are convinced that it is important to handle SDS paths in global illumination.



Progressive Photon Mapping

First algorithm for computing *all* types of light transport with arbitrary accuracy

Our progressive photon mapping is the first method for computing all types of light transport, including SDS paths, with arbitrary accuracy.



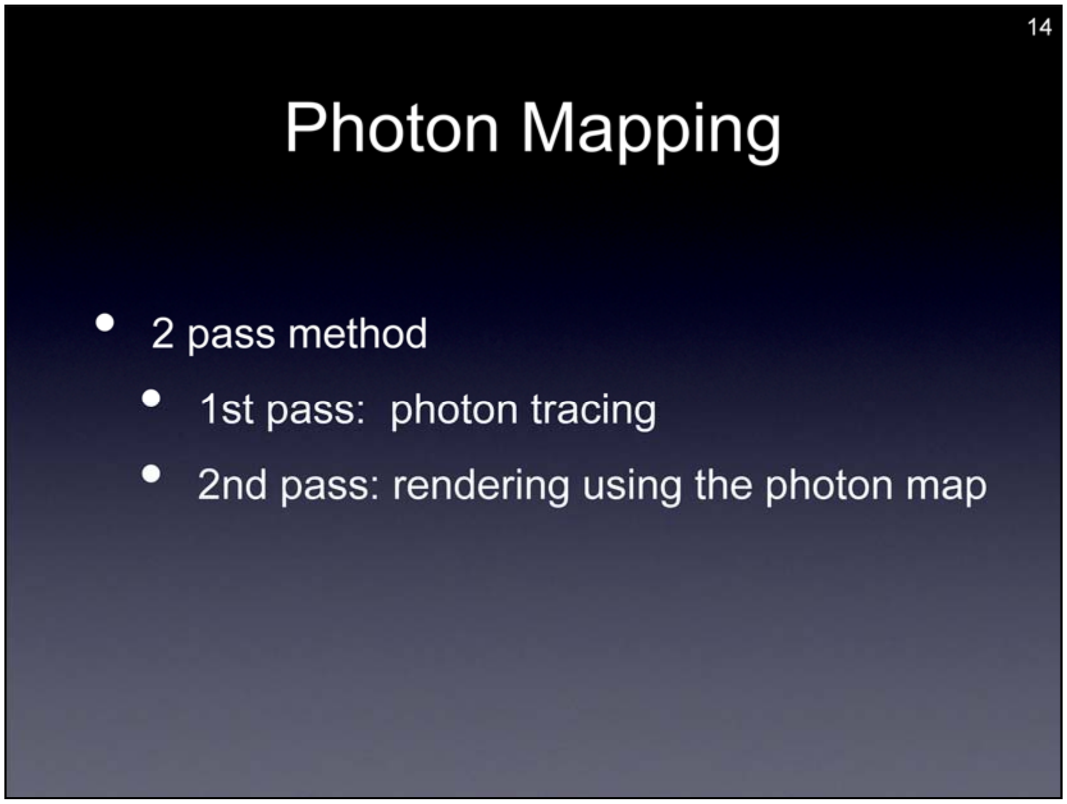
Progressive Photon Mapping

- New formulation of photon mapping
  - Robust for *any* light path including SDS path
  - Arbitrary accuracy using finite memory
  - New progressive radiance estimation algorithm
  - Easy to implement

To be more precise, our progressive photon mapping is a new formulation of photon mapping. Our method is robust for any light path including SDS path. We can compute images with arbitrary accuracy just by increasing the number of photons without storing all the photons. To do this, we introduce a new progressive radiance estimation algorithm, which is easy to implement.

Photon Mapping

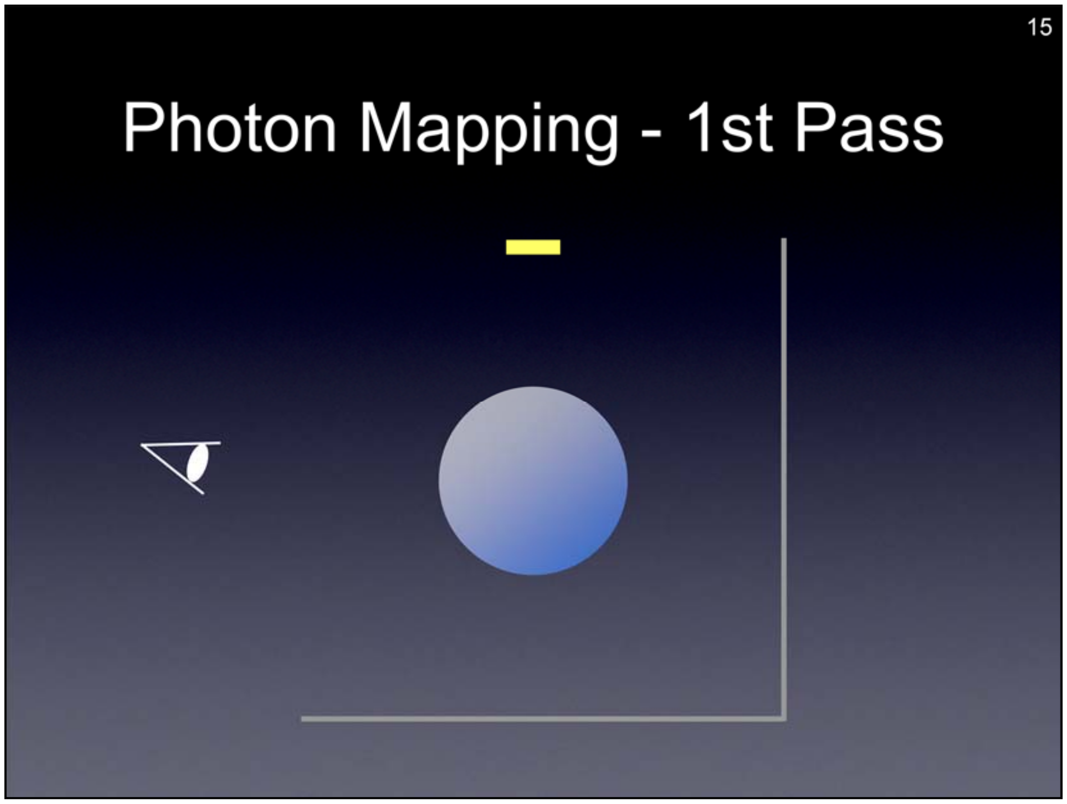
Since our method is based on photon mapping, let’s for a moment look at the standard photon mapping.



Photon Mapping

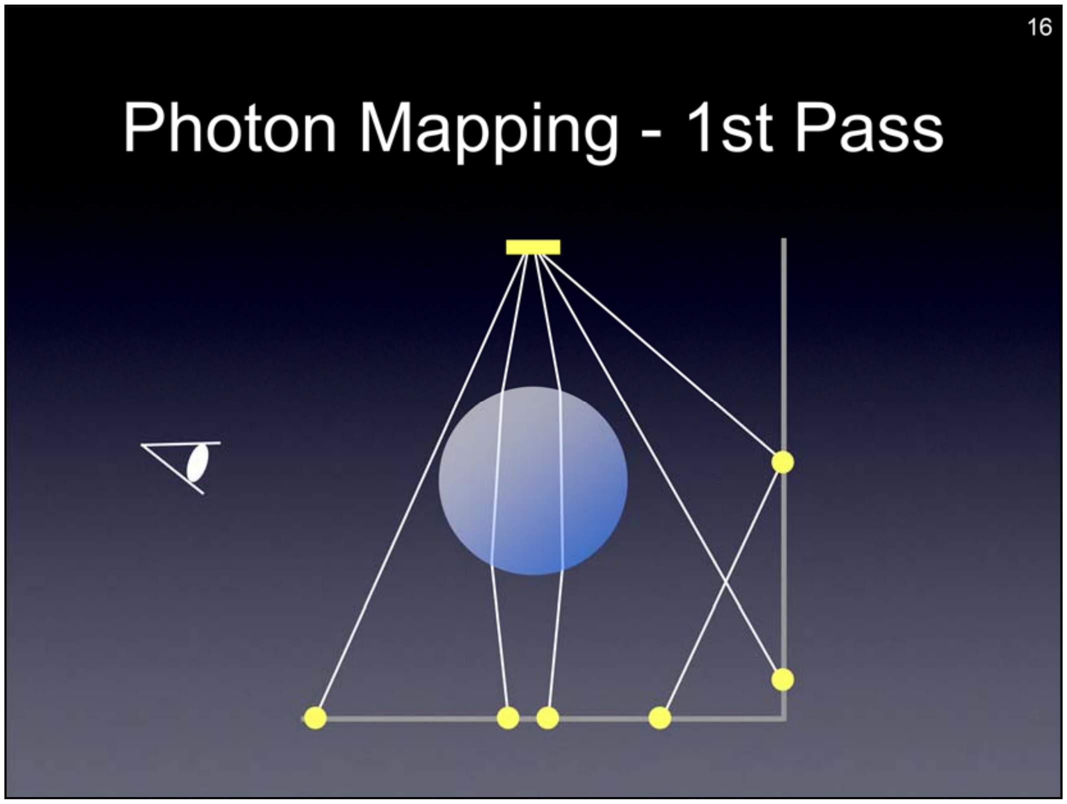
- 2 pass method
  - 1st pass: photon tracing
  - 2nd pass: rendering using the photon map

Photon mapping is a two pass method. In the first pass, photons are emitted from light sources and interactions of photons with surfaces are stored as a photon map. In the second pass, an image is rendered using the photon map from the first pass.



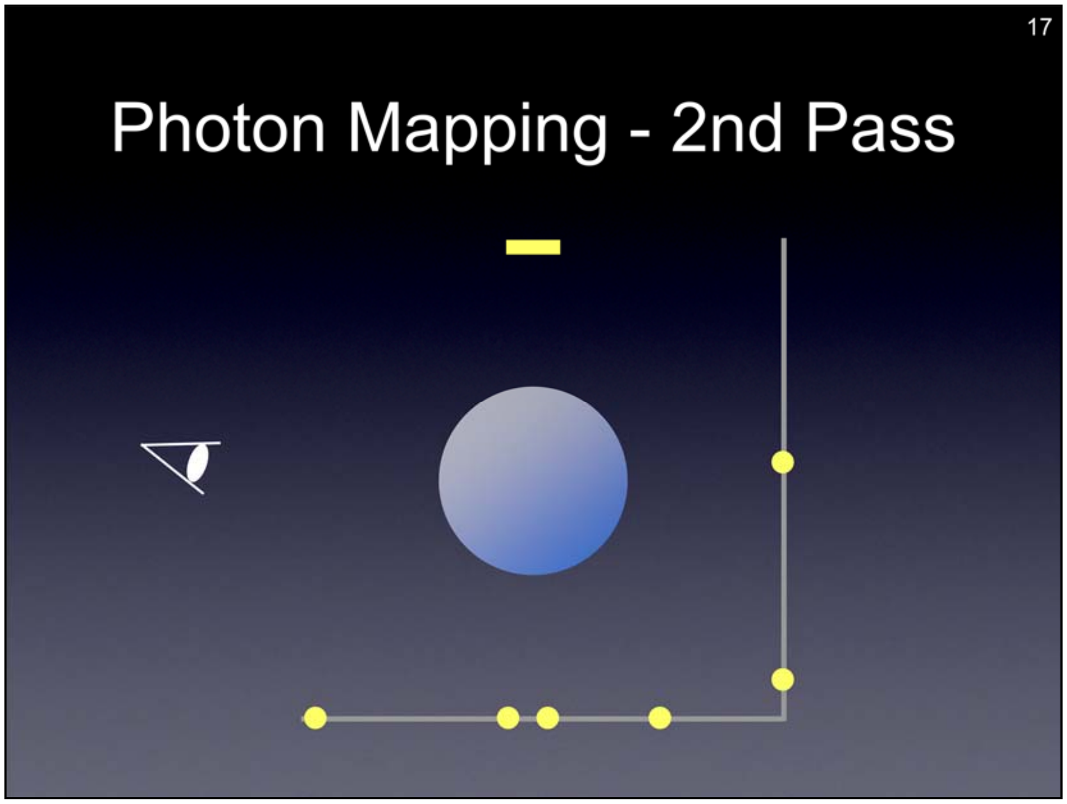
Photon Mapping - 1st Pass

Let’s look at this example scene, where there is a glass ball, diffuse walls, and the light source at the top.



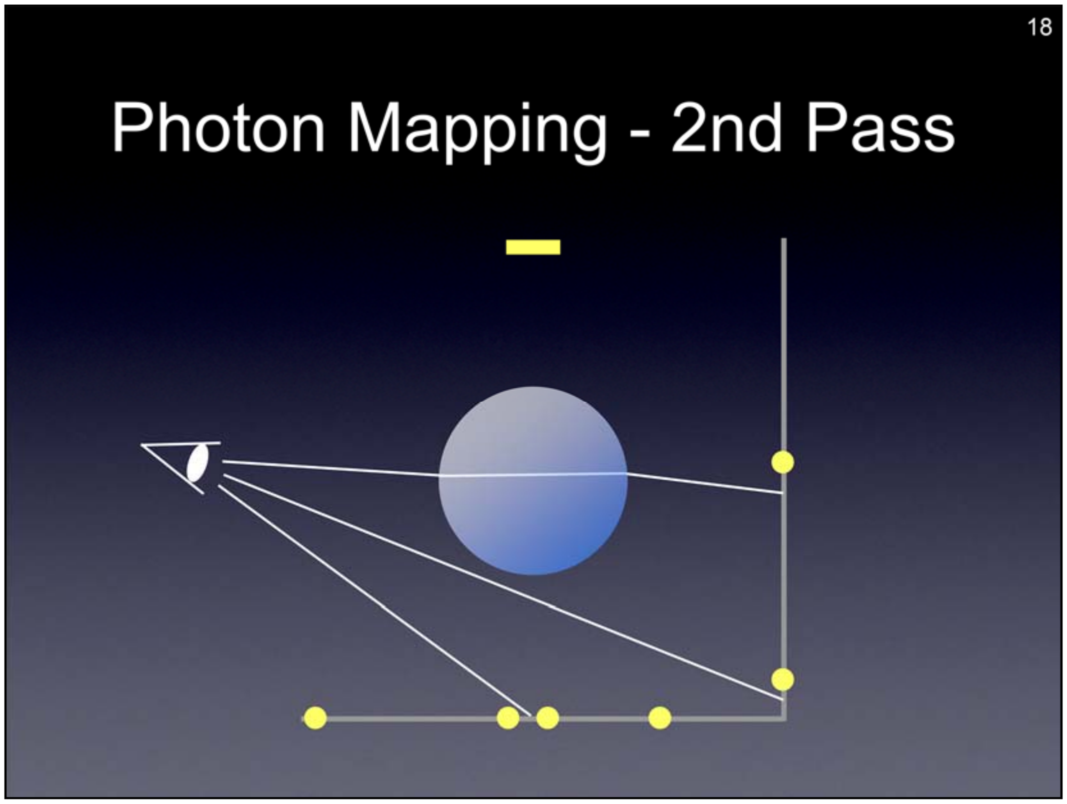
Photon Mapping - 1st Pass

In the first pass of photon mapping, we trace photons from the light source, and store the intersections with diffuse surfaces as a photon map.



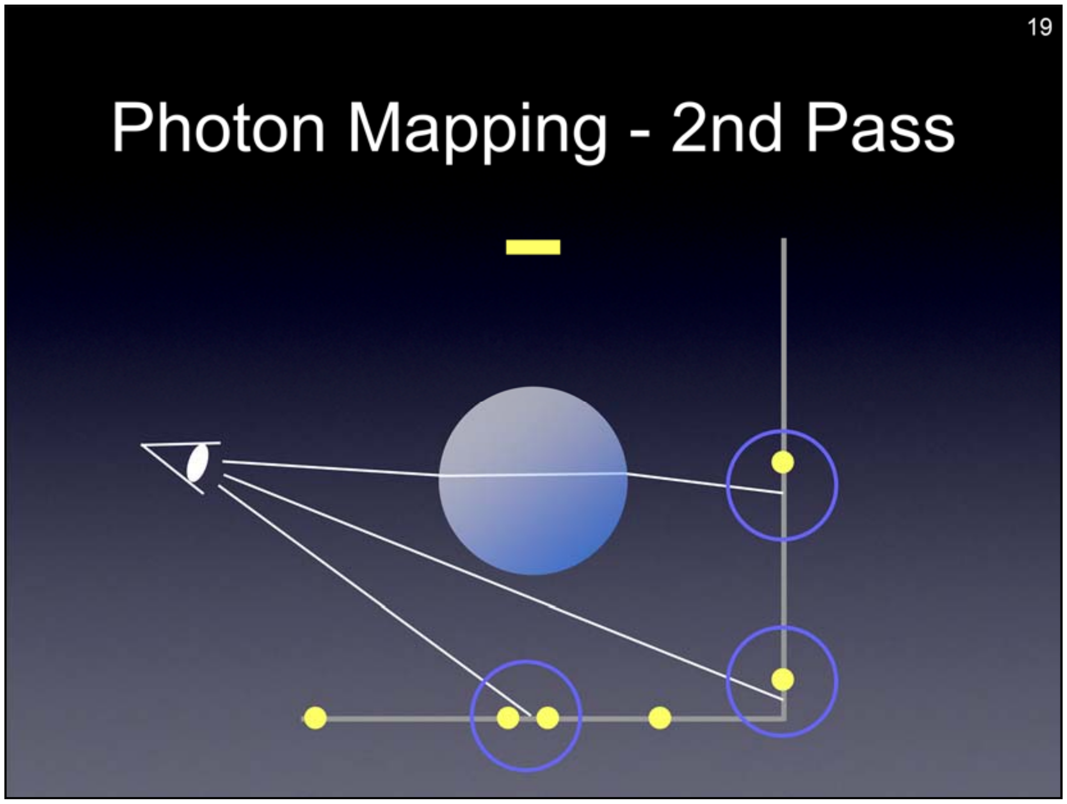
Photon Mapping - 2nd Pass

In the second pass, we trace rays from the eyes,

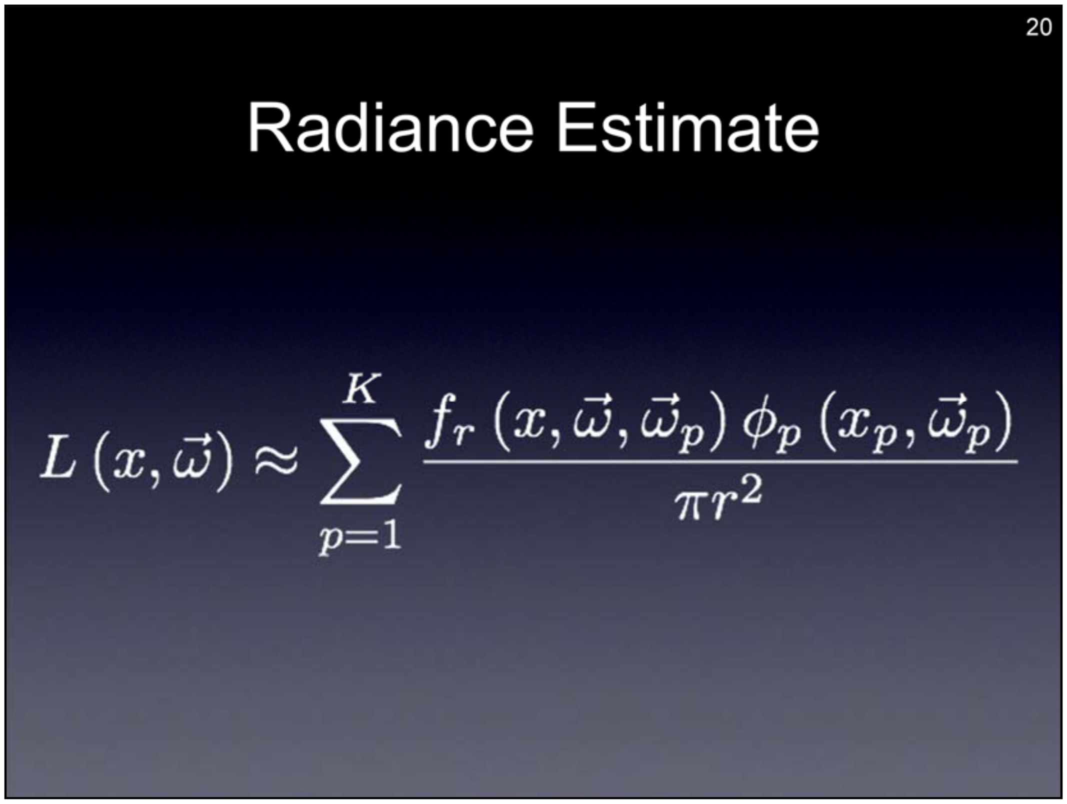


Photon Mapping - 2nd Pass

In the second pass, we trace rays from the eyes,



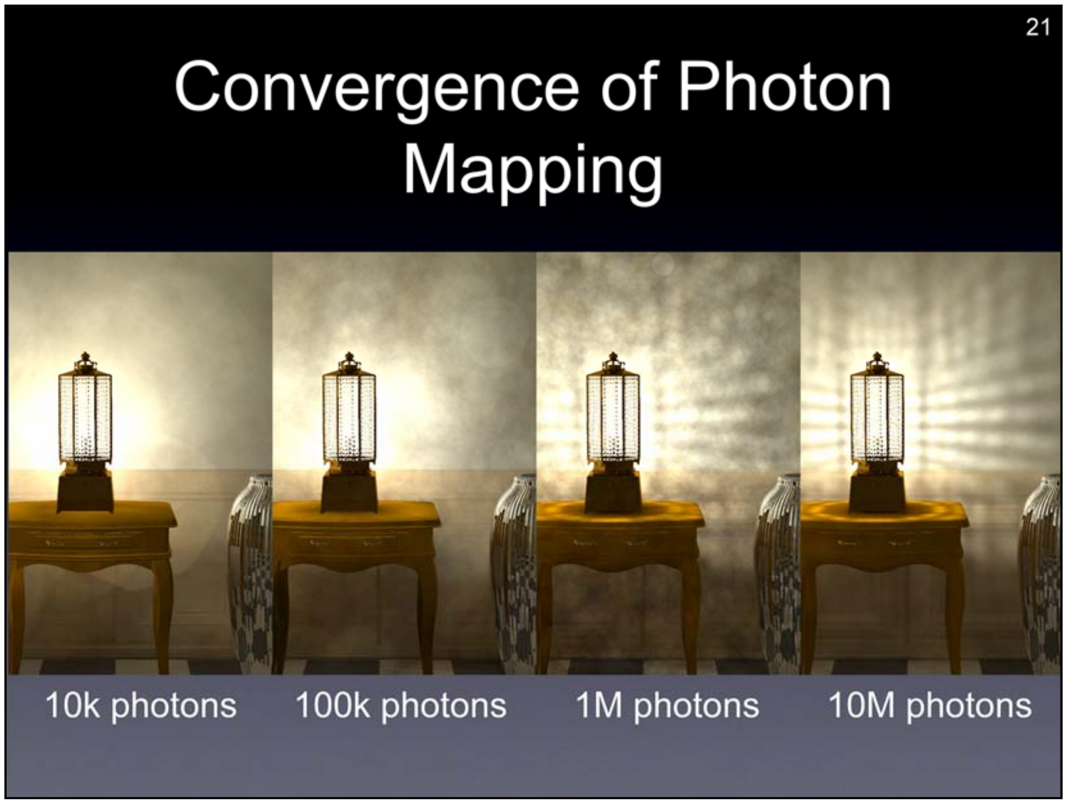
and estimate the resulting radiance by finding nearby photons around each intersection point of eye ray.



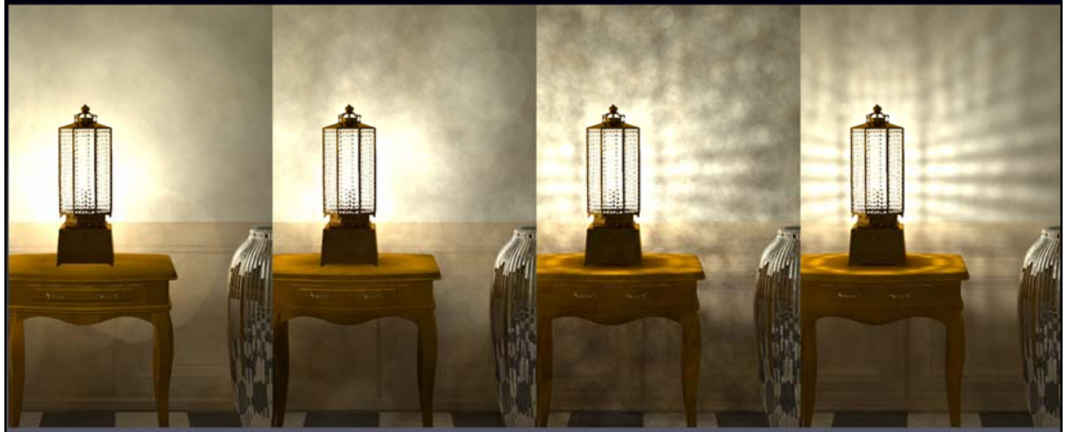
Radiance Estimate

$$L(x,\vec{\omega}) \approx \sum_{p=1}^{K} \frac{f_r(x,\vec{\omega},\vec{\omega}_p)\,\phi_p(x_p,\vec{\omega}_p)}{\pi r^2}$$

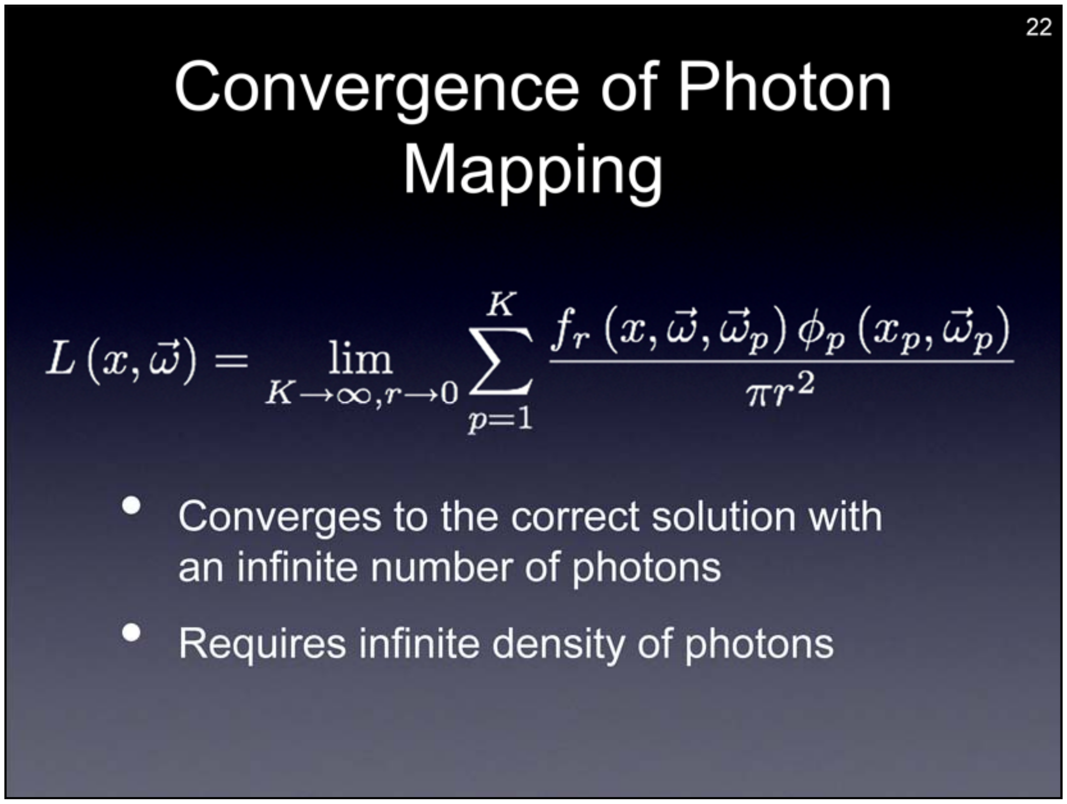
We use this equation to estimate radiance, where K is the number of nearby photons around x, f_r is a BRDF, phi_p is flux (or power) of each photon, and r is the search radius of all the nearby photons. Note that this equation is an approximation of the correct radiance using K photons.



Convergence of Photon Mapping



Although photon mapping is a biased method, it is a consistent method. It means is that the image rendered by photon mapping converges to the correct solution by increasing the number of photons.

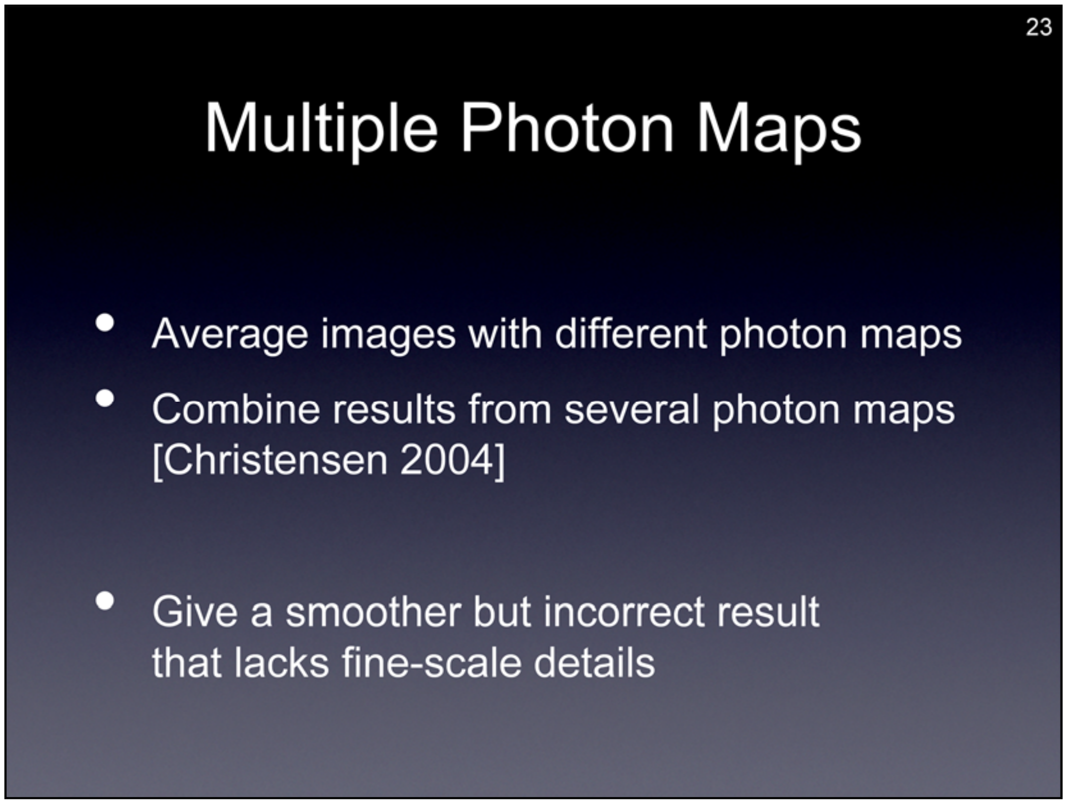


Convergence of Photon Mapping

$$L(x,\vec{\omega}) = \lim_{K\to\infty, r\to 0} \sum_{p=1}^{K} \frac{f_r(x,\vec{\omega},\vec{\omega}_p)\,\phi_p(x_p,\vec{\omega}_p)}{\pi r^2}$$

- Converges to the correct solution with an infinite number of photons
- Requires infinite density of photons

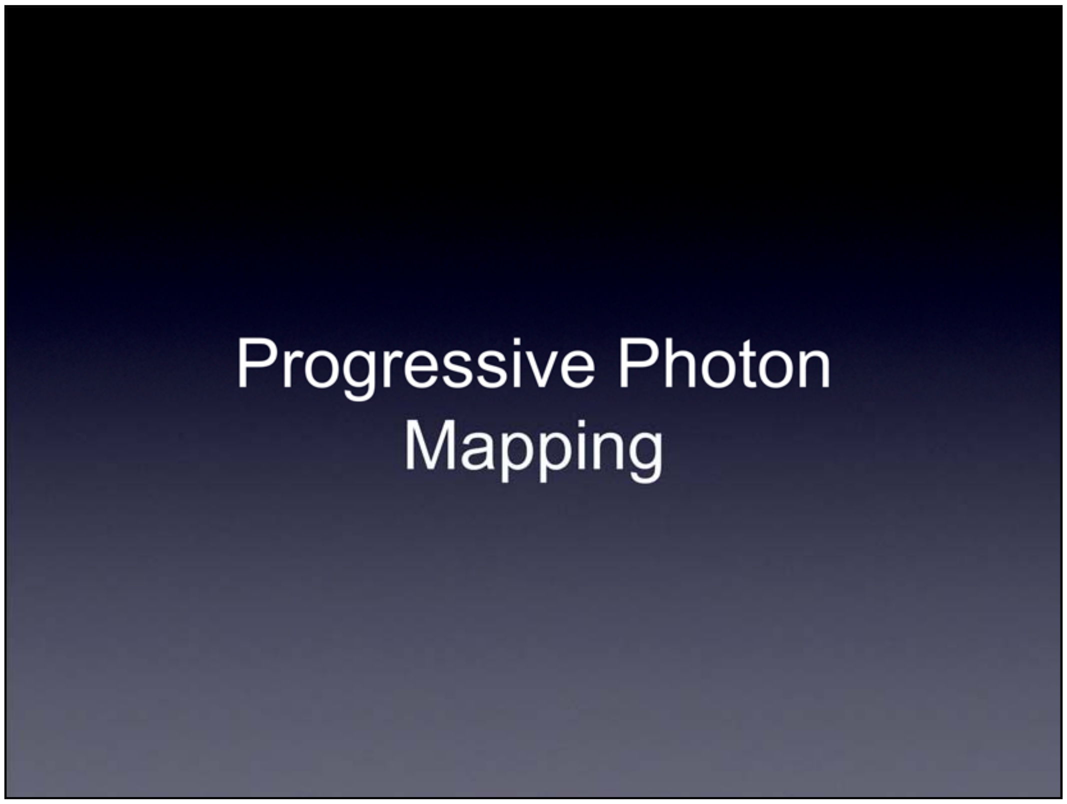
To be more precise, radiance computed from nearby photons converges to the correct solution of the rendering equation, if we use an infinite number of photons within an infinitely small search radius. Unfortunately this is not practical since it would require an infinite amount of memory.



Multiple Photon Maps

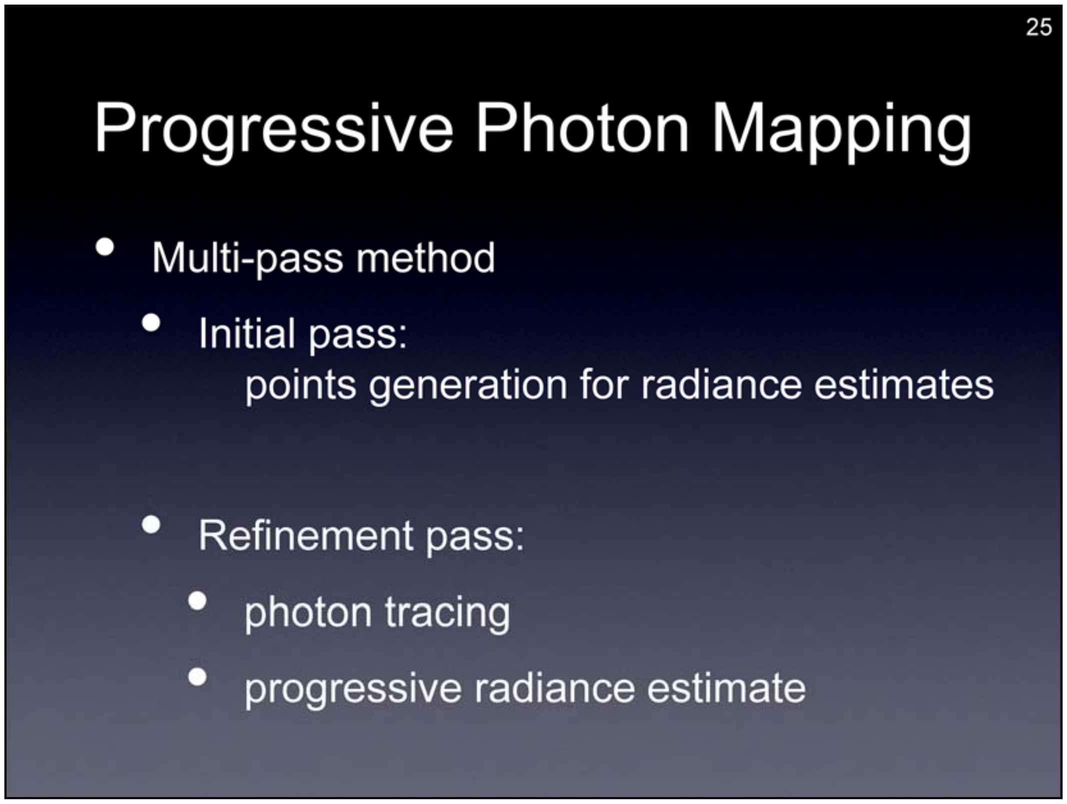
- Average images with different photon maps
- Combine results from several photon maps [Christensen 2004]
- Give a smoother but incorrect result that lacks fine-scale details

Instead of using single photon map with a large number of photons, one can think of combining results from several photon maps with a small number of photons, to increase the total number of photons. The simplest method would be to take the average of images rendered by different photon maps. Christensen presented a more sophisticated method to combine several photon maps. These methods give a smoother result, but details of lighting would be missing if they are not captured by individual photon map. In other words, the result does not converge to the correct solution even if an infinite number of photons is used.



Progressive Photon Mapping

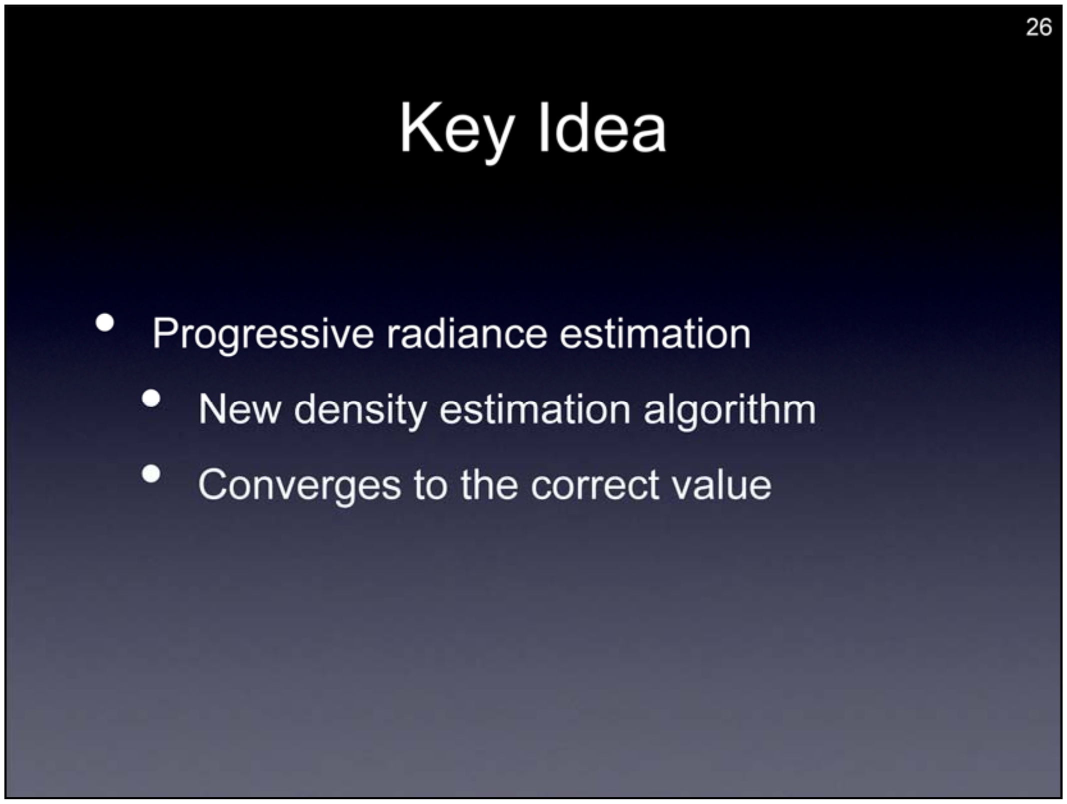
In contrast, progressive photon mapping converges to the correct solution, and I will now describe how this is achieved.



Progressive Photon Mapping

- Multi-pass method
  - Initial pass:
    points generation for radiance estimates
  - Refinement pass:
    - photon tracing
    - progressive radiance estimate

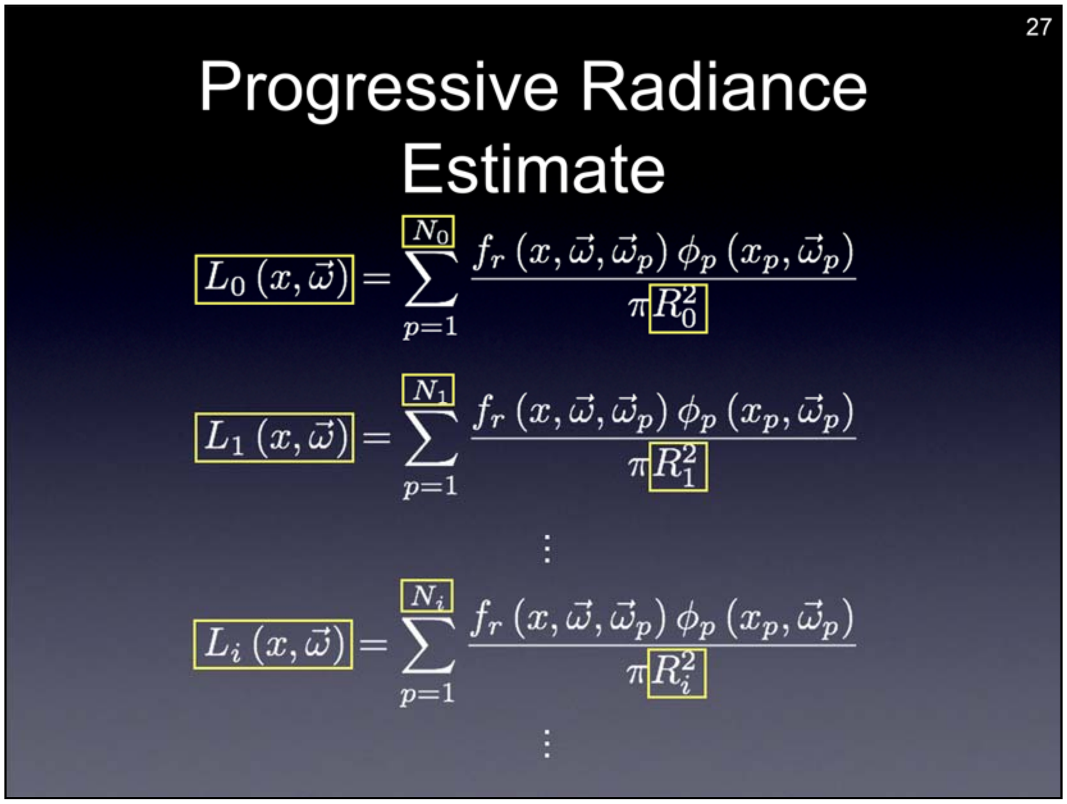
Progressive photon mapping is a multi-pass method. In the initial pass, we generate points where we want to estimate radiance, which is usually done by ray tracing. In the succeeding refinement passes, we trace photons exactly in the same way as the standard photon mapping. We then apply a new progressive radiance estimate to compute radiance at each point.



Key Idea

- Progressive radiance estimation
  - New density estimation algorithm
  - Converges to the correct value

The key idea of progressive photon mapping is in progressive radiance estimation. It is based on a new density estimation algorithm where the result converges to the correct value after an infinite number of refinement passes.



Progressive Radiance Estimate

$$\boxed{L_0(x,\vec{\omega})} = \sum_{p=1}^{\boxed{N_0}} \frac{f_r(x,\vec{\omega},\vec{\omega}_p)\,\phi_p(x_p,\vec{\omega}_p)}{\pi \boxed{R_0^2}}$$

$$\boxed{L_1(x,\vec{\omega})} = \sum_{p=1}^{\boxed{N_1}} \frac{f_r(x,\vec{\omega},\vec{\omega}_p)\,\phi_p(x_p,\vec{\omega}_p)}{\pi \boxed{R_1^2}}$$

$$\vdots$$

$$\boxed{L_i(x,\vec{\omega})} = \sum_{p=1}^{\boxed{N_i}} \frac{f_r(x,\vec{\omega},\vec{\omega}_p)\,\phi_p(x_p,\vec{\omega}_p)}{\pi \boxed{R_i^2}}$$

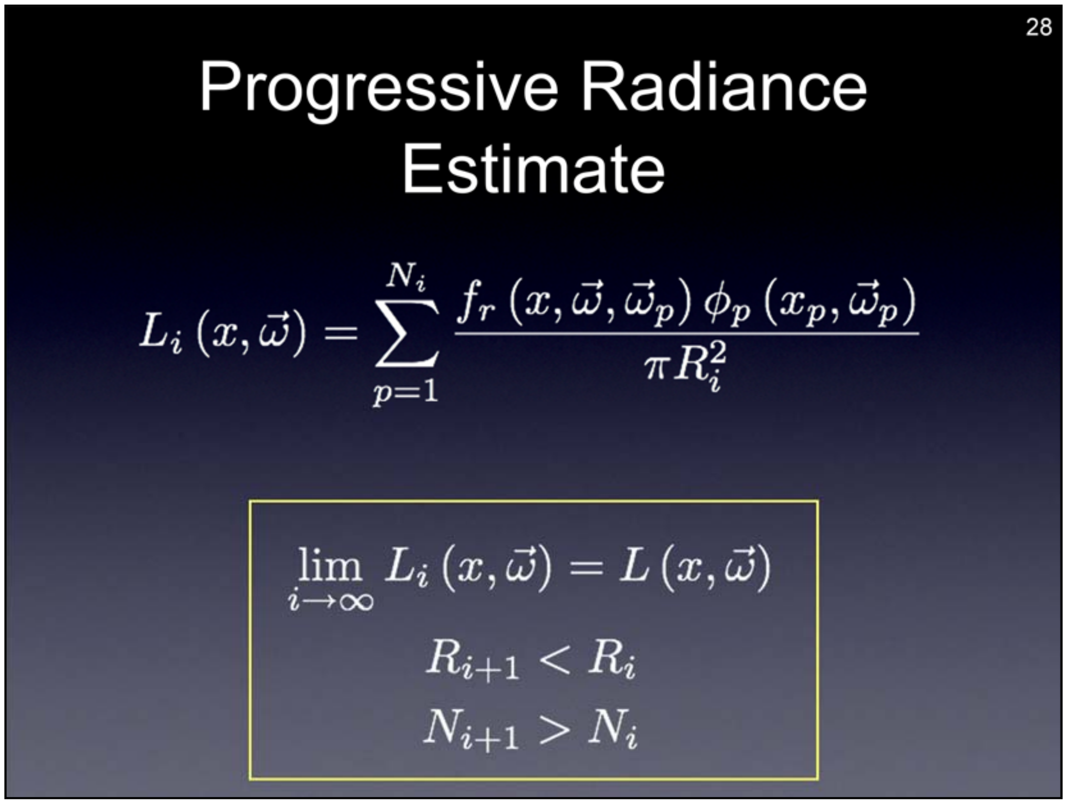
$$\vdots$$

In progressive radiance estimate, we estimate radiance at a specific point using an iterative approach.

In the first pass, we have N_0 photons within a disc of radius R_0, and we compute radiance using this equation.

In the second pass, we accumulate more photons and refine the estimated radiance.

We keep repeating this process to obtain more accurate radiance estimates.



Progressive Radiance Estimate

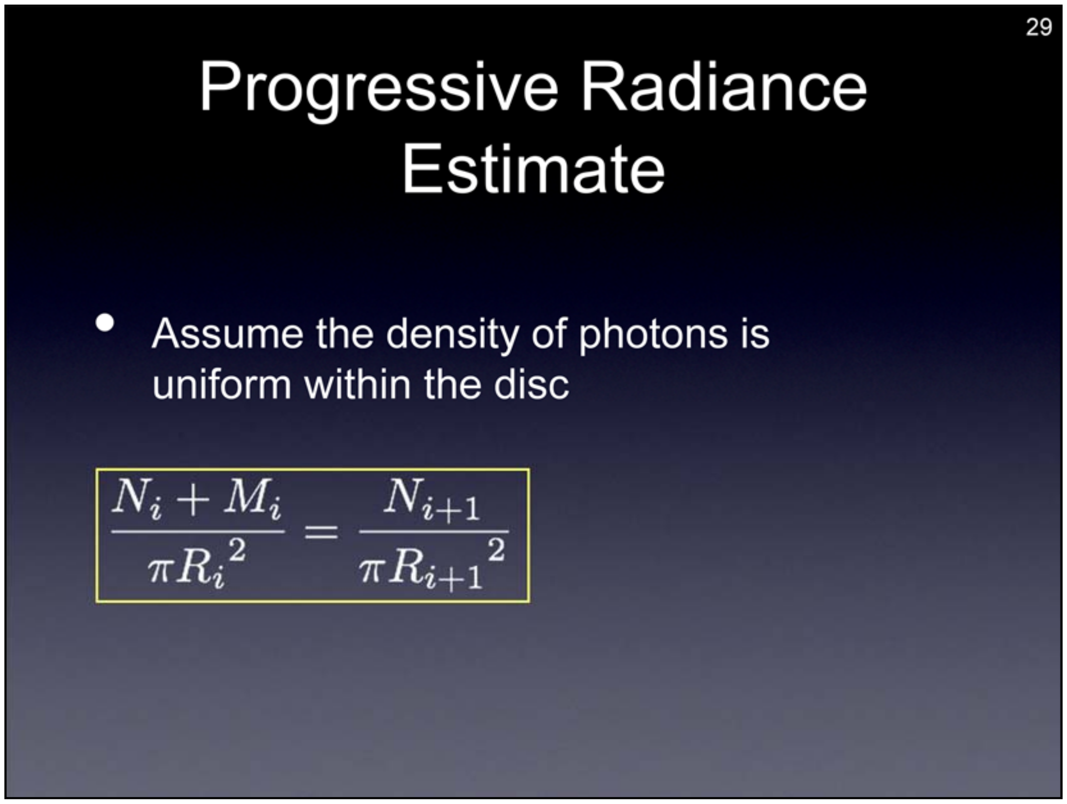
$$L_i(x,\vec{\omega}) = \sum_{p=1}^{N_i} \frac{f_r(x,\vec{\omega},\vec{\omega}_p)\,\phi_p(x_p,\vec{\omega}_p)}{\pi R_i^2}$$

$$\lim_{i\to\infty} L_i(x,\vec{\omega}) = L(x,\vec{\omega})$$

$$R_{i+1} < R_i$$

$$N_{i+1} > N_i$$

In order to achieve convergence to the correct value after an infinite number of refinement passes, we refine the estimate of radiance iteratively. After each iteration, the search radius should decrease and the number of nearby photons should increase to ensure convergence. I will now describe how this can be done.

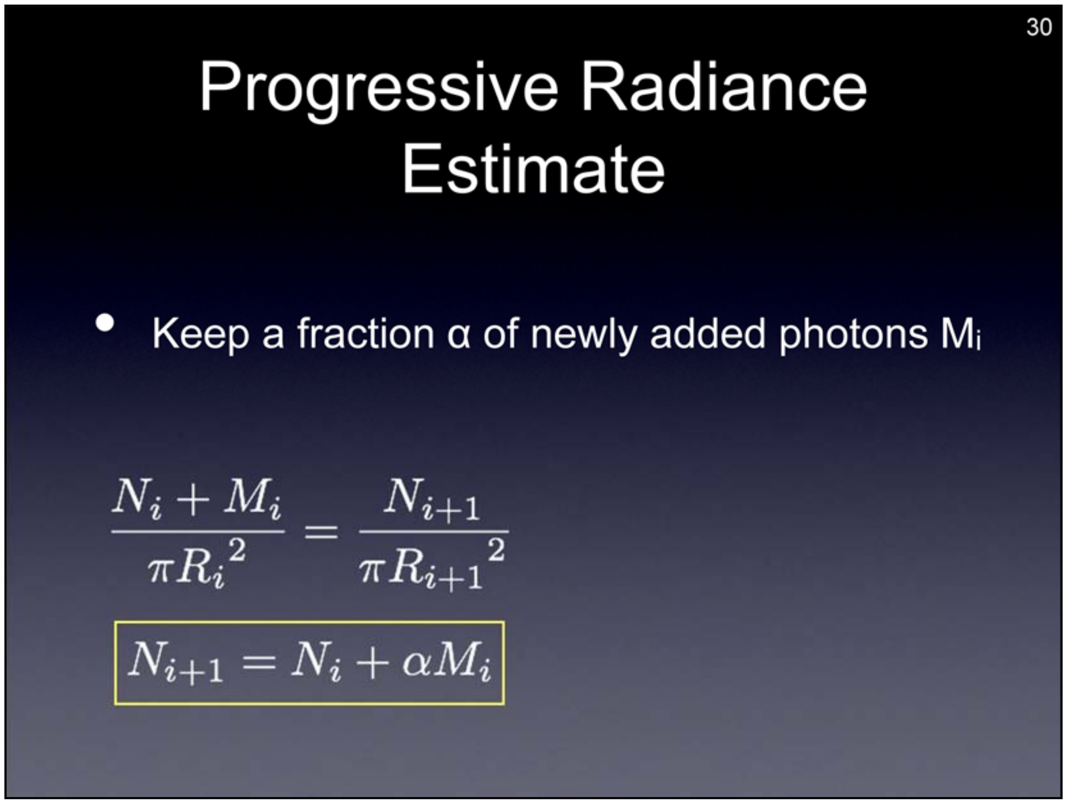


Progressive Radiance Estimate

- Assume the density of photons is uniform within the disc

$$\boxed{\frac{N_i + M_i}{\pi {R_i}^2} = \frac{N_{i+1}}{\pi {R_{i+1}}^2}}$$

Assume we have a point with N_i photons and we would like to add the contribution from M_i new nearby photons. Our goal is to obtain N_i+1 photons within a disc of radius R_i+1 under the conditions I showed before. If we assume that the density of photons within the disc is uniform, we can express the density before and after the iteration as shown in this slide.



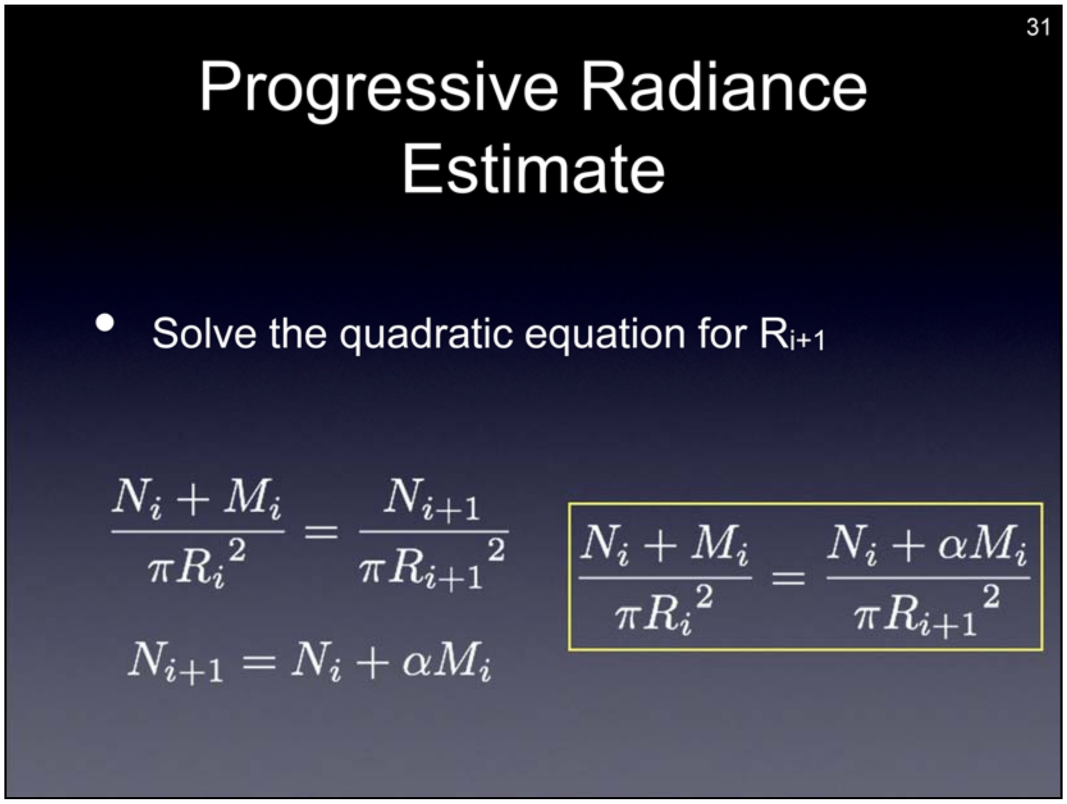
Progressive Radiance Estimate

- Keep a fraction α of newly added photons M_i

$$\frac{N_i + M_i}{\pi {R_i}^2} = \frac{N_{i+1}}{\pi {R_{i+1}}^2}$$

$$\boxed{N_{i+1} = N_i + \alpha M_i}$$

To ensure that the number of photons is increasing, we accumulate a fraction alpha of the new nearby photons.



Progressive Radiance Estimate

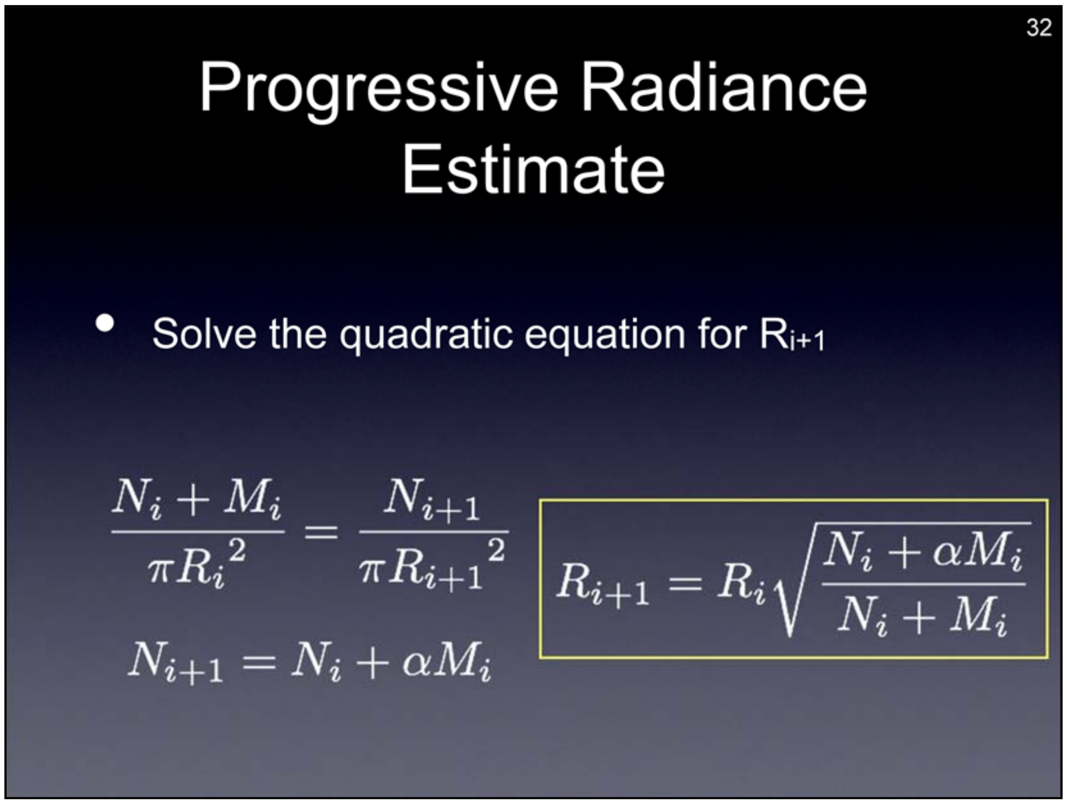
- Solve the quadratic equation for R_{i+1}

$$\frac{N_i + M_i}{\pi {R_i}^2} = \frac{N_{i+1}}{\pi {R_{i+1}}^2}$$

$$N_{i+1} = N_i + \alpha M_i$$

$$\boxed{\frac{N_i + M_i}{\pi {R_i}^2} = \frac{N_i + \alpha M_i}{\pi {R_{i+1}}^2}}$$

We then combine these two equations to obtain a quadratic equation of the new radius,



Progressive Radiance Estimate

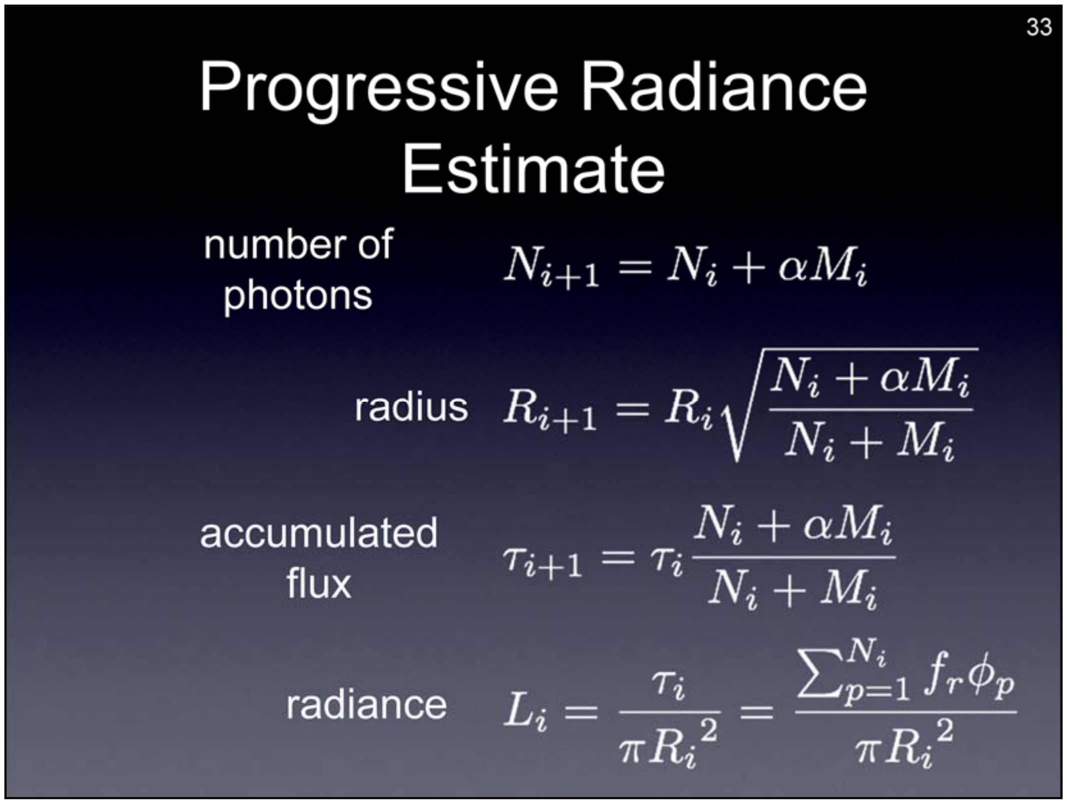
- Solve the quadratic equation for R_{i+1}

$$\frac{N_i + M_i}{\pi {R_i}^2} = \frac{N_{i+1}}{\pi {R_{i+1}}^2}$$

$$N_{i+1} = N_i + \alpha M_i$$

$$\boxed{R_{i+1} = R_i \sqrt{\frac{N_i + \alpha M_i}{N_i + M_i}}}$$

and solve this equation to get the new radius R_i+1. We use a similar approach to accumulate the flux associated with a point.



Progressive Radiance Estimate

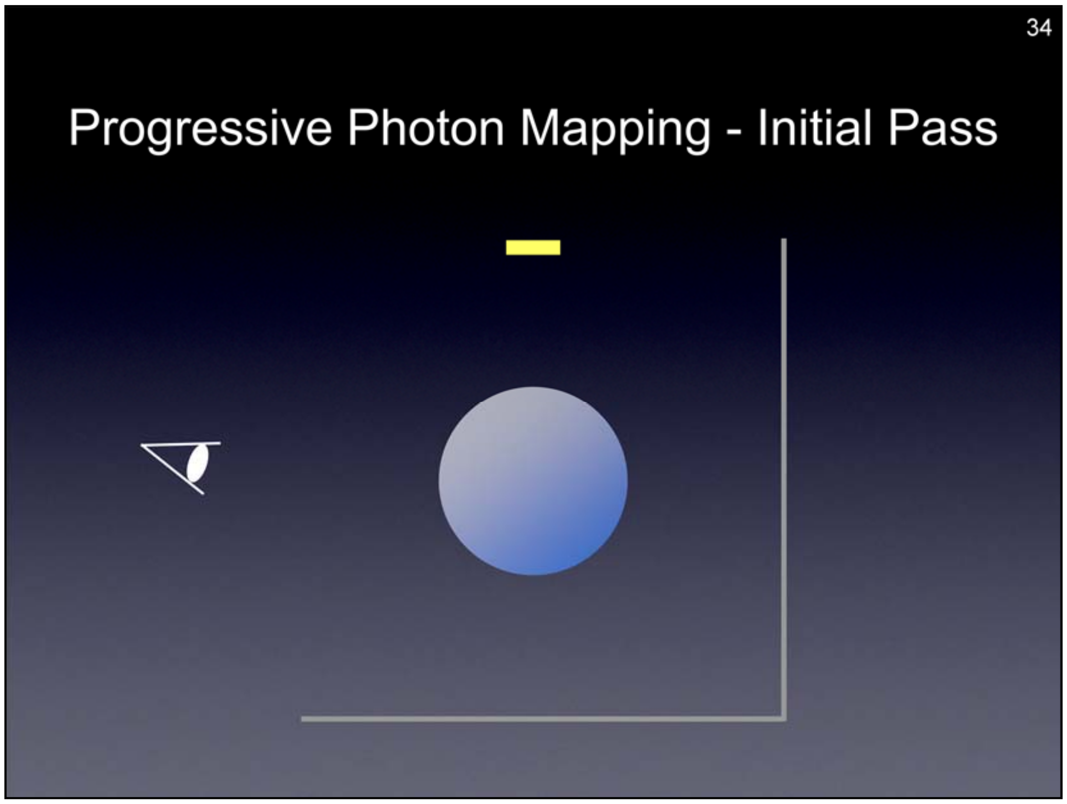
number of photons $N_{i+1} = N_i + \alpha M_i$

radius $R_{i+1} = R_i \sqrt{\dfrac{N_i + \alpha M_i}{N_i + M_i}}$

accumulated flux $\tau_{i+1} = \tau_i \dfrac{N_i + \alpha M_i}{N_i + M_i}$

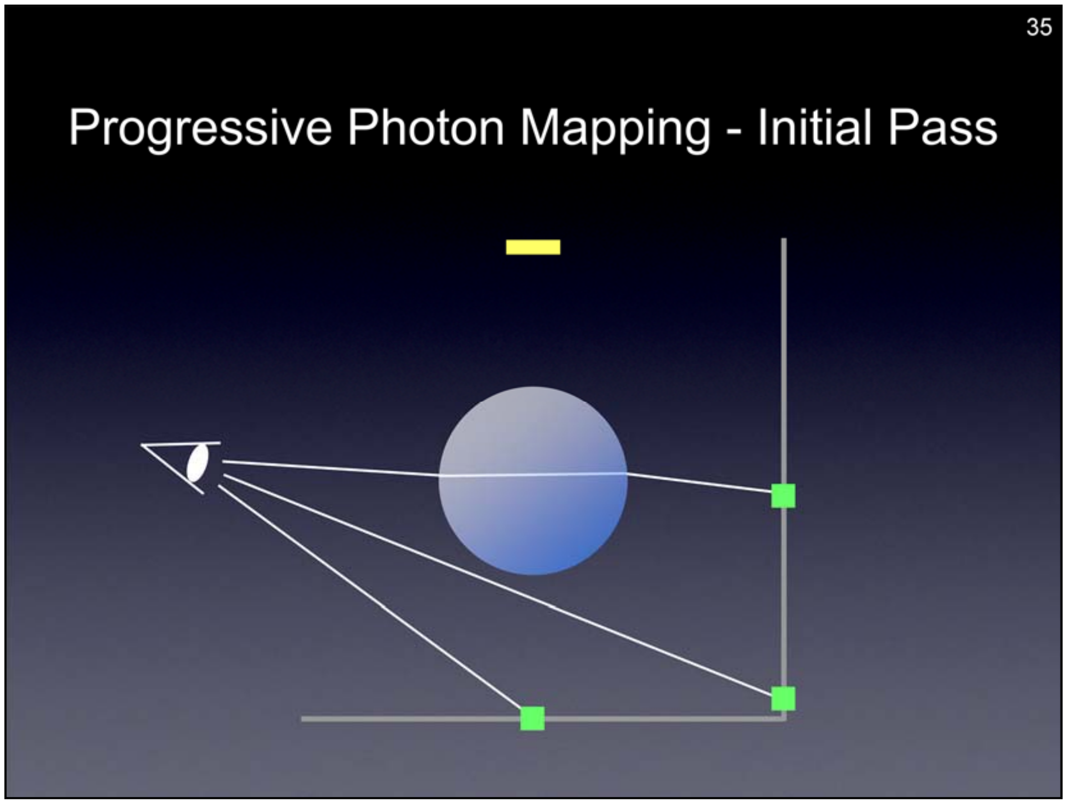
radiance $L_i = \dfrac{\tau_i}{\pi {R_i}^2} = \dfrac{\sum_{p=1}^{N_i} f_r \phi_p}{\pi {R_i}^2}$

In summary, we store the number of nearby photons, the search radius, the accumulated flux at each point, and simply update the values using these equations after each iteration.



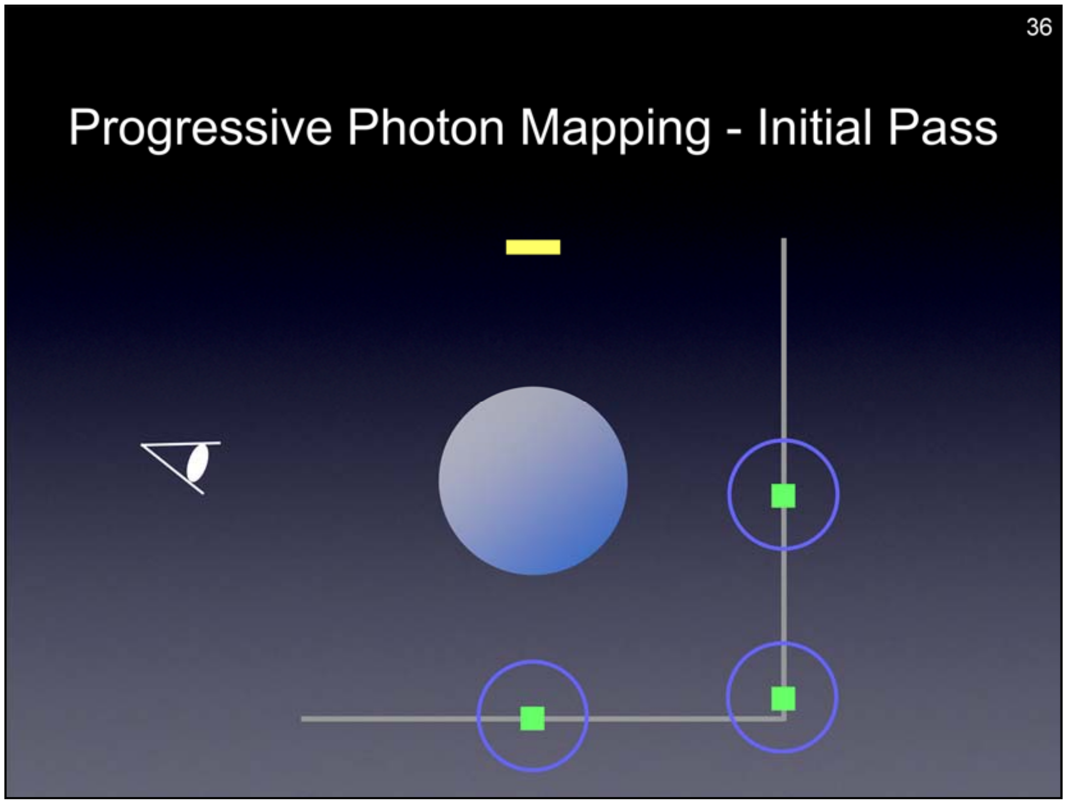
Progressive Photon Mapping - Initial Pass

To summarize the overall algorithm, let’s go back to the same example as we have seen in the standard photon mapping.



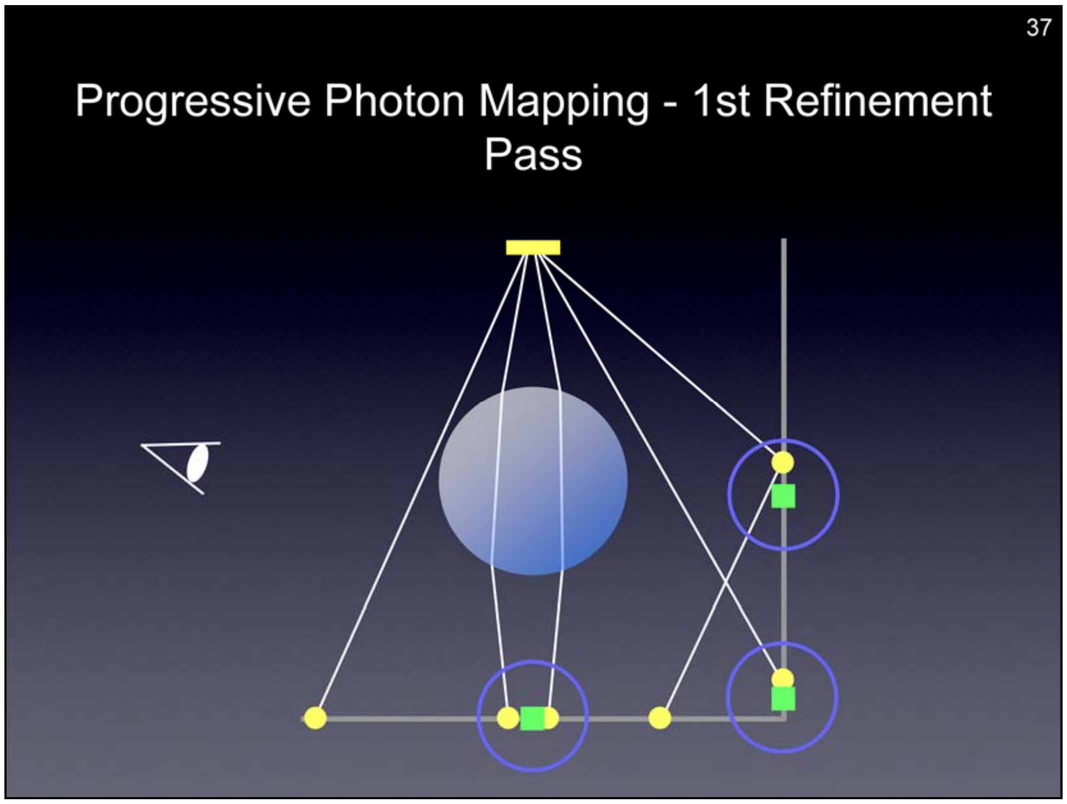
Progressive Photon Mapping - Initial Pass

In the initial pass of progressive photon mapping, we generate points where we want to estimate radiance by tracing rays from the viewer. This process is very similar to the second pass of the standard photon mapping except we now store information of each point.



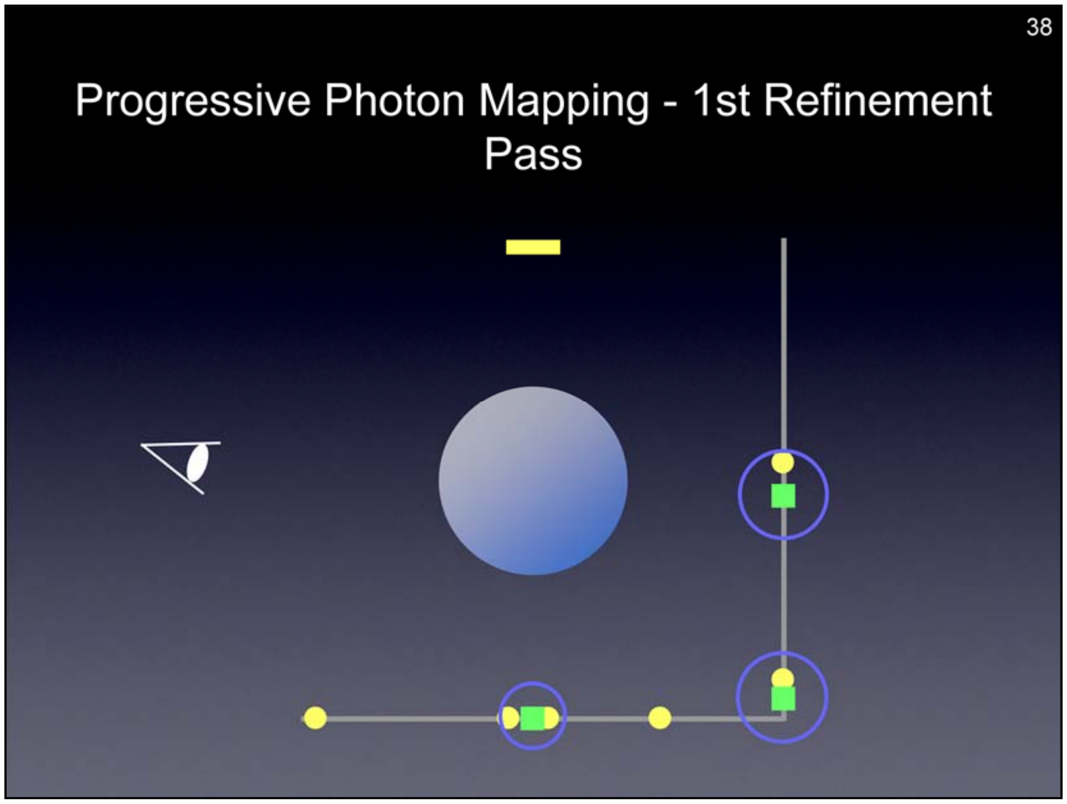
Progressive Photon Mapping - Initial Pass

After the first iteration, each point is assigned an initial search radius.



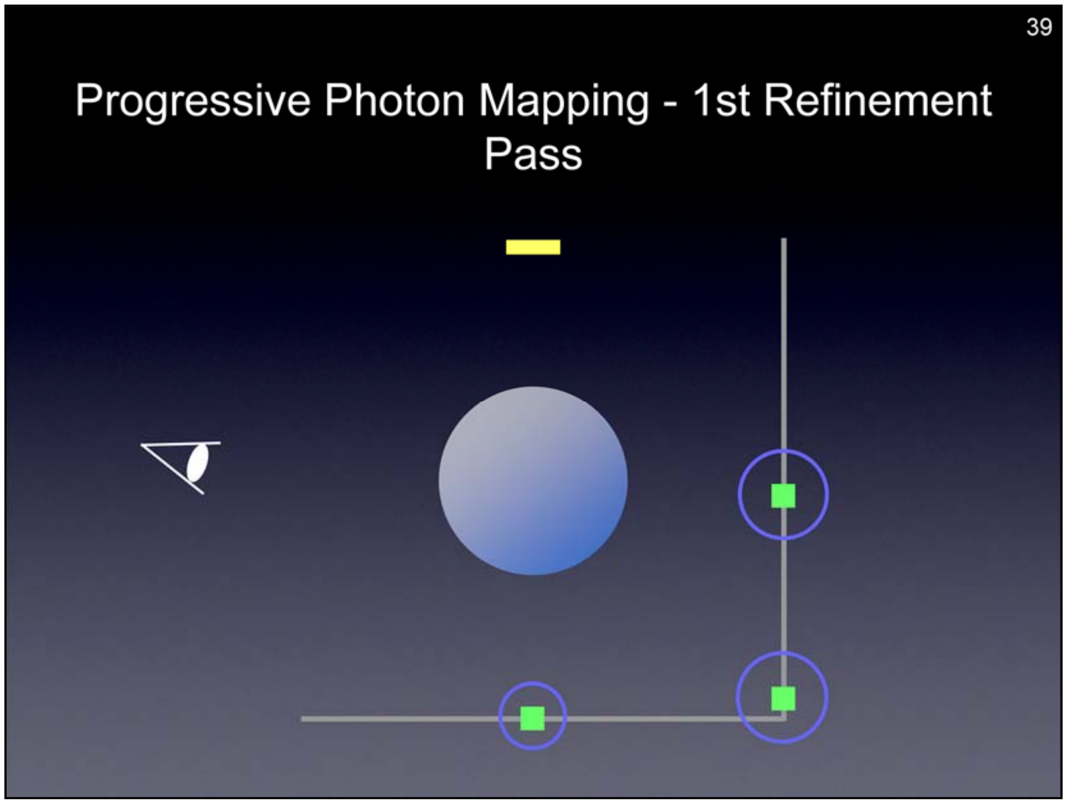
Progressive Photon Mapping - 1st Refinement Pass

In each refinement pass, we first trace photons in the same way as the standard photon mapping. We then find photons within the radius of each point.



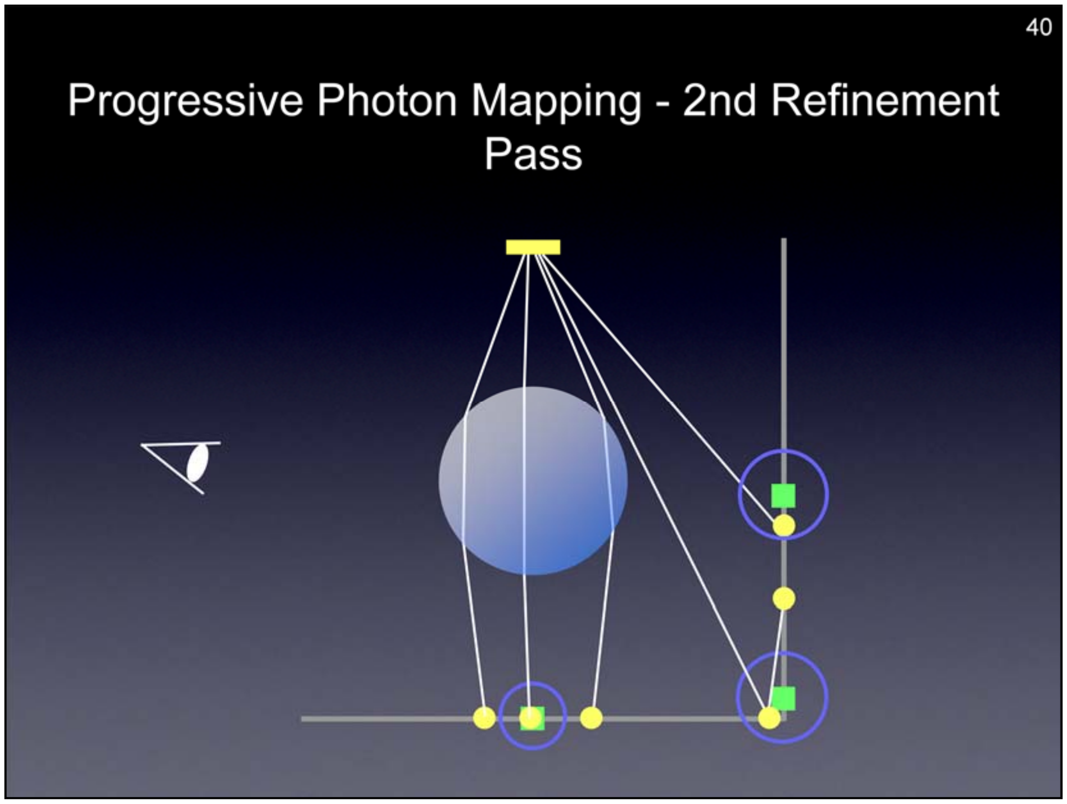
Progressive Photon Mapping - 1st Refinement Pass

Based on the nearby photons we update the statistics of each point. This includes reducing the radius as shown on the slide.



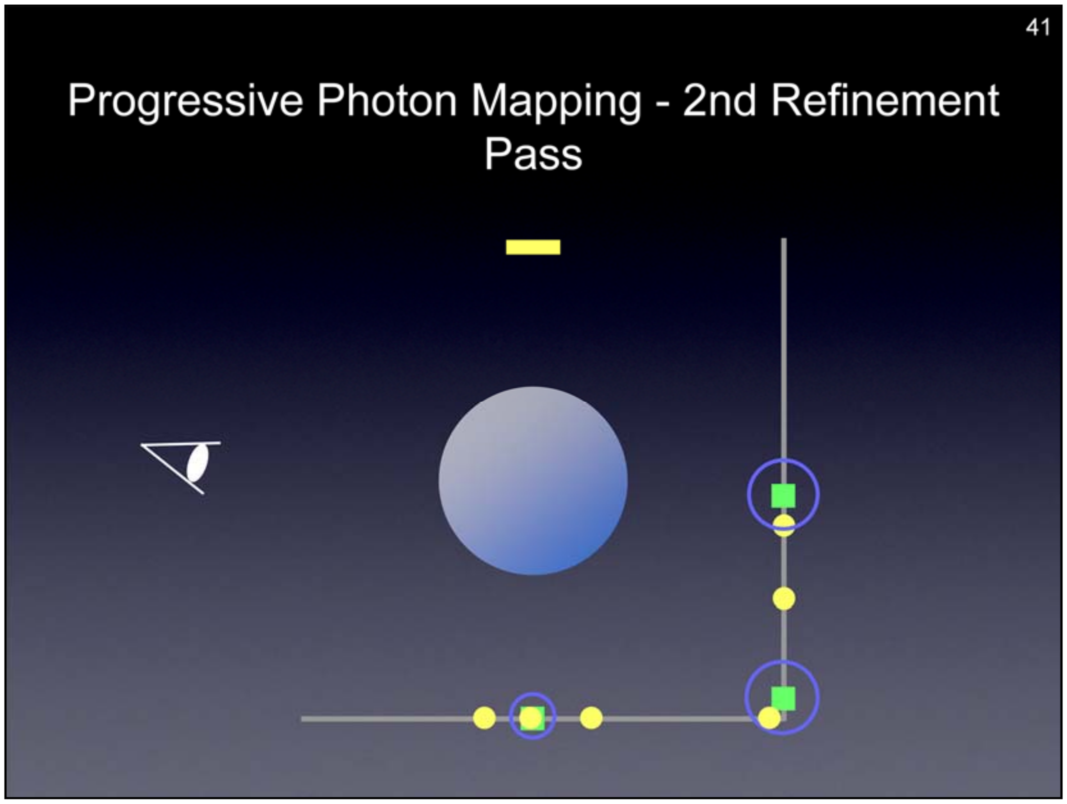
Progressive Photon Mapping - 1st Refinement Pass

We then discard all photons and prepare for the next iteration.



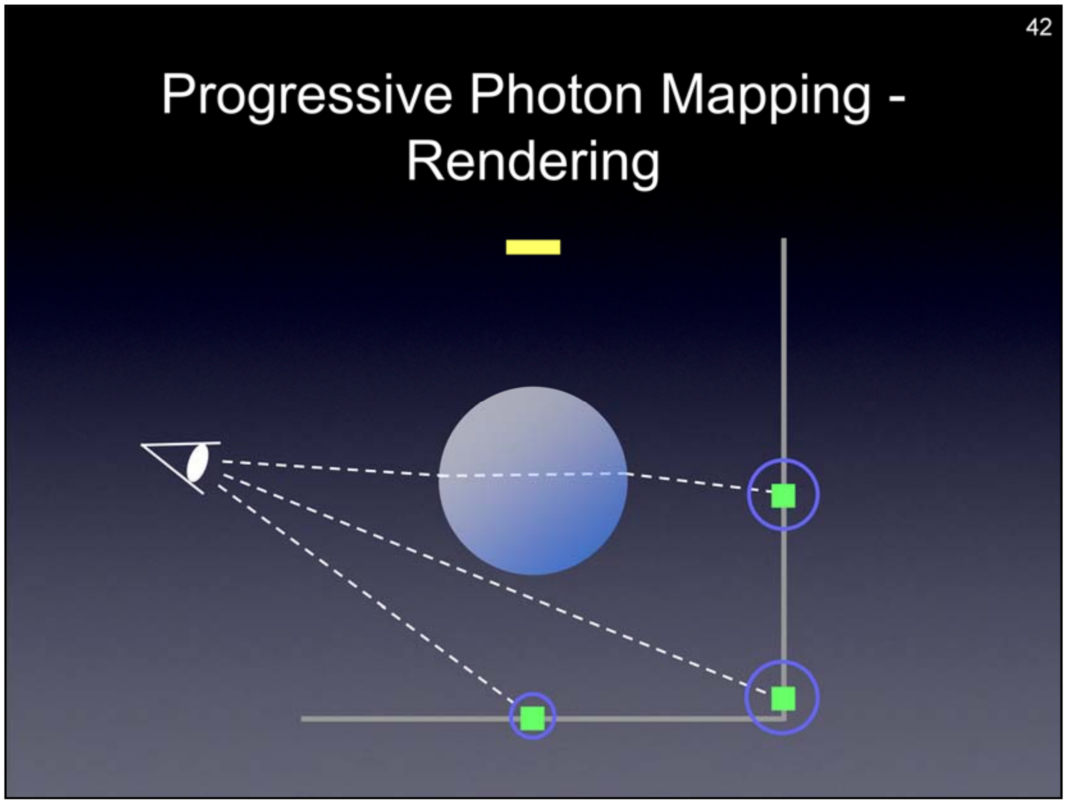
Progressive Photon Mapping - 2nd Refinement Pass

The succeeding refinement passes proceed exactly in the same way,



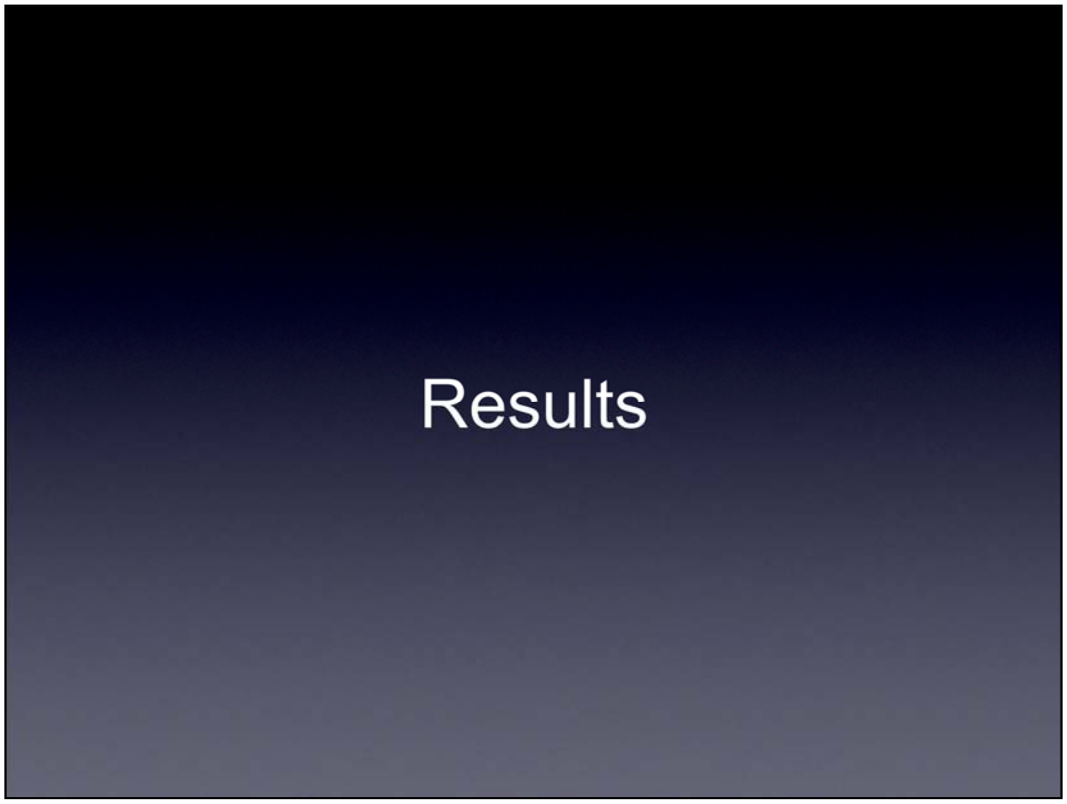
Progressive Photon Mapping - 2nd Refinement Pass

but we use updated radii and statistics.



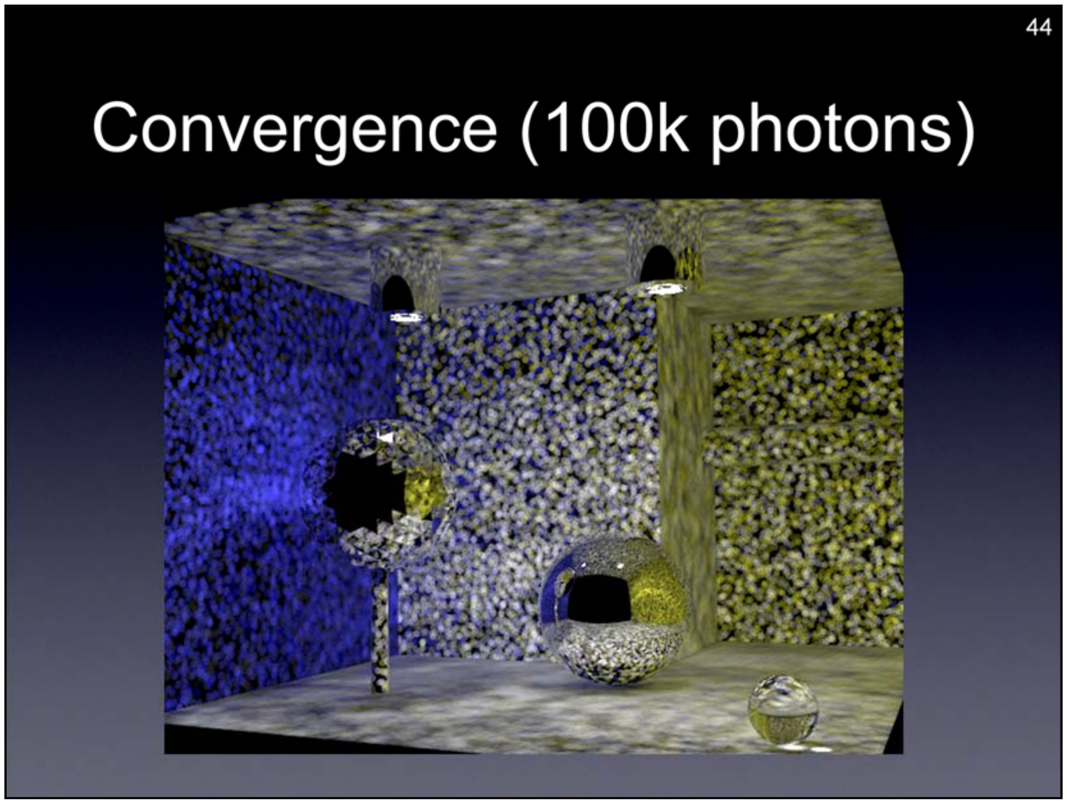
Progressive Photon Mapping - Rendering

Finally, we can render the image at any iteration by estimating the radiance at each point.

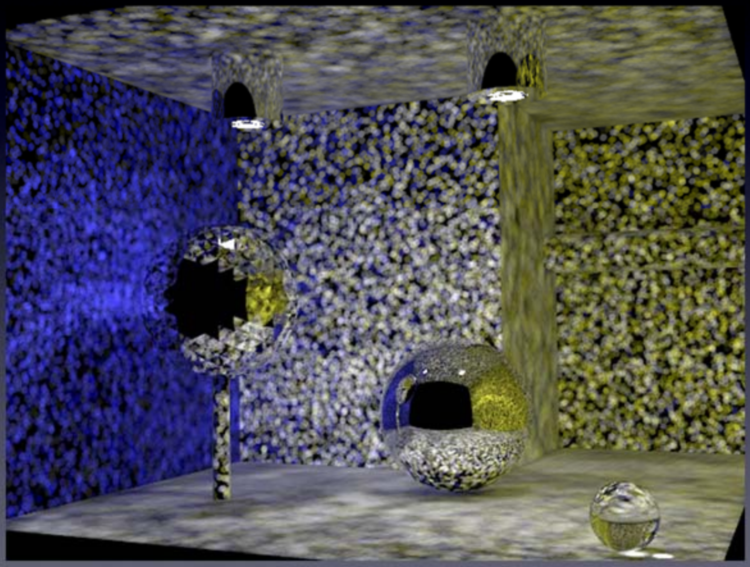


Results

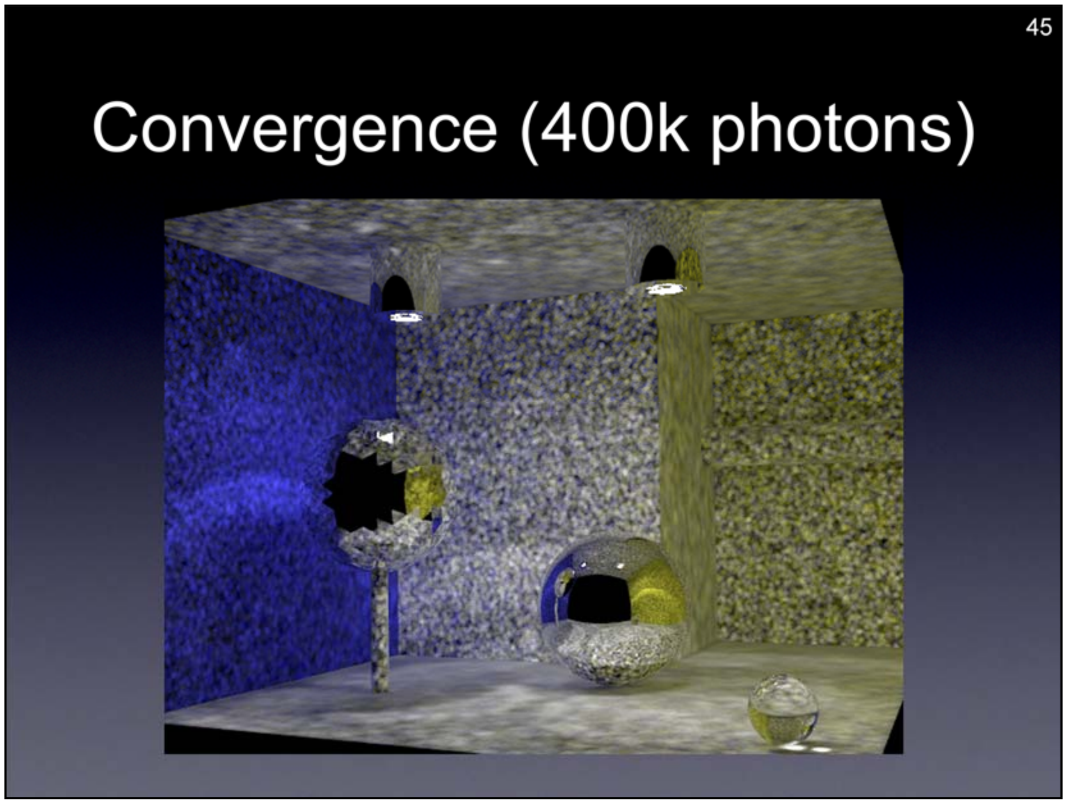
Now, I am going to talk about our results.



Convergence (100k photons)

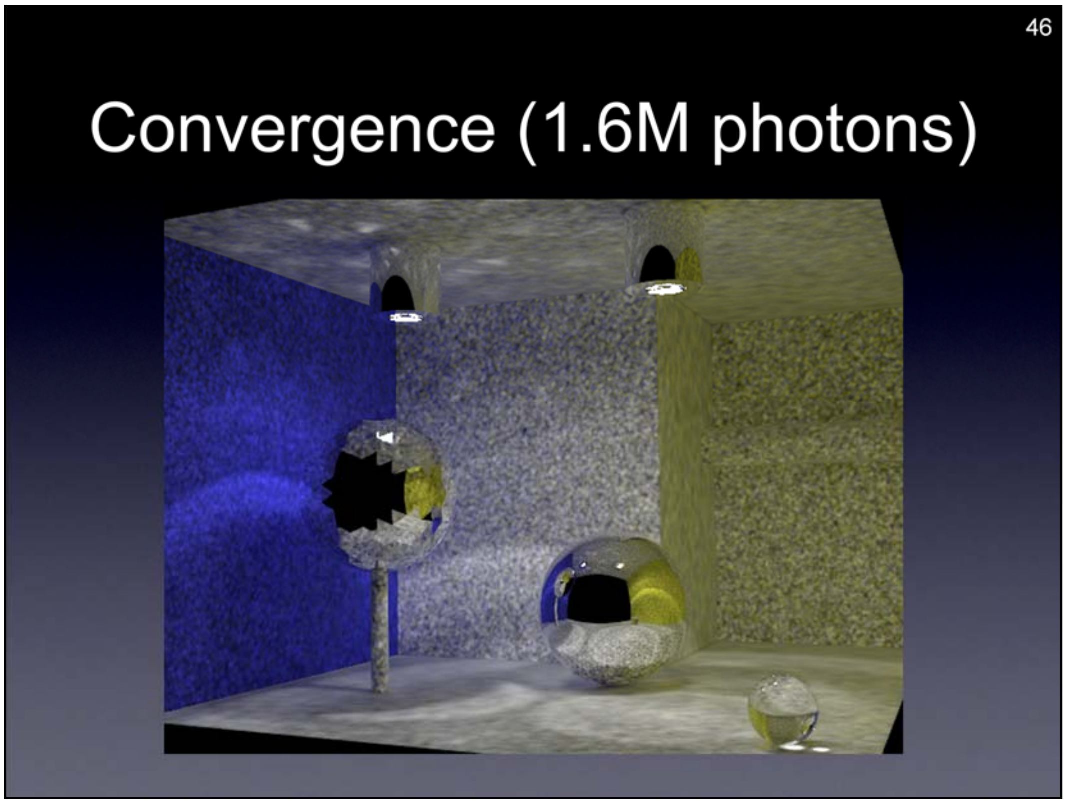


First I will show how images rendered by our method converges to the correct solution. This image is rendered using one hundred thousand photons.



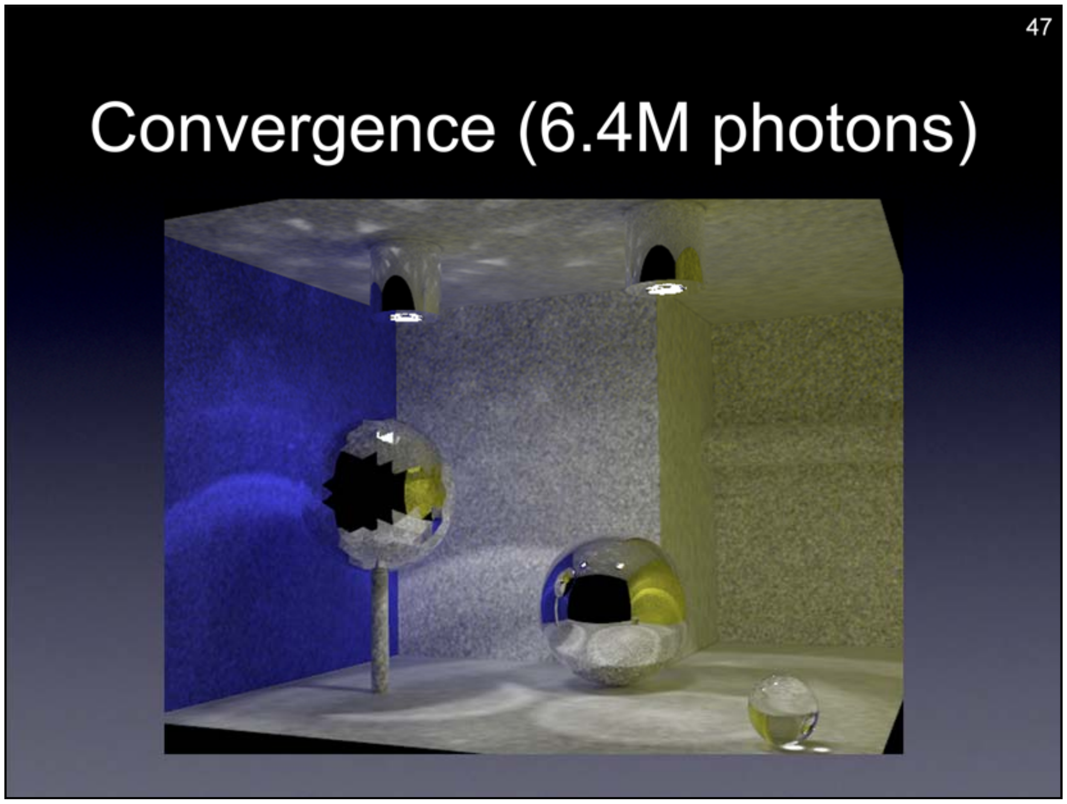
Convergence (400k photons)

as you increase the number of photons, thereby adding more refinement passes,



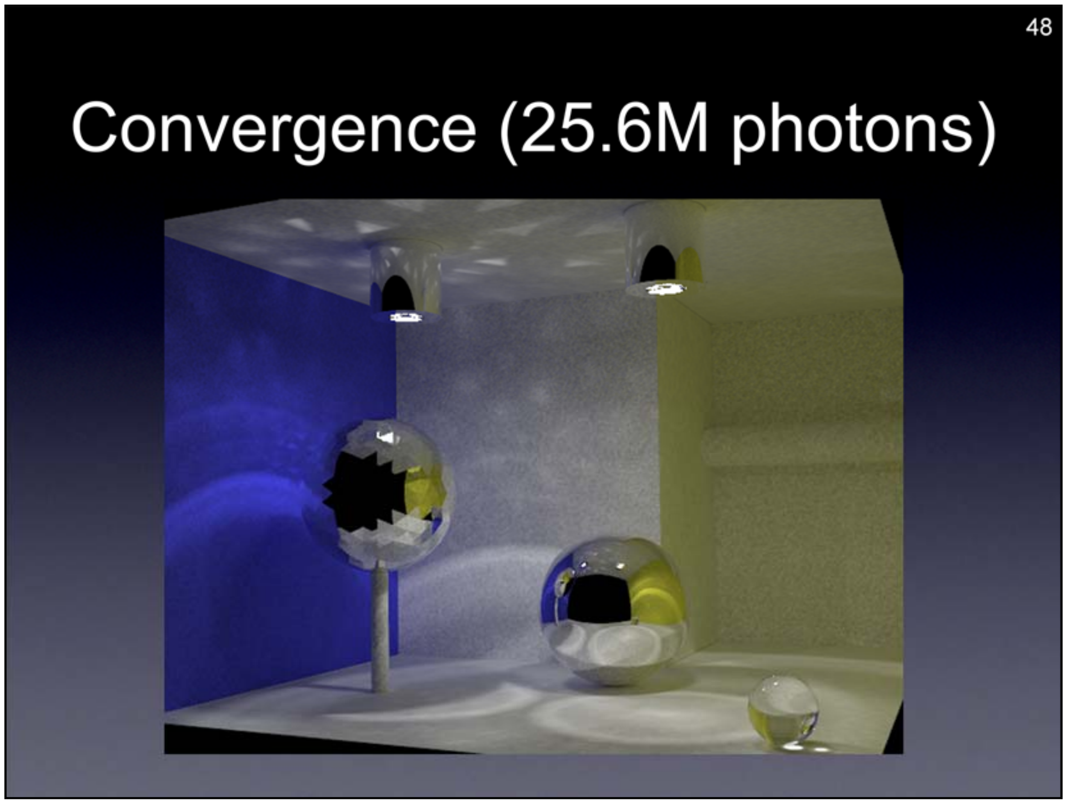
Convergence (1.6M photons)

we can obtain more details and smoother result. Note that even with a relatively low number of photons,



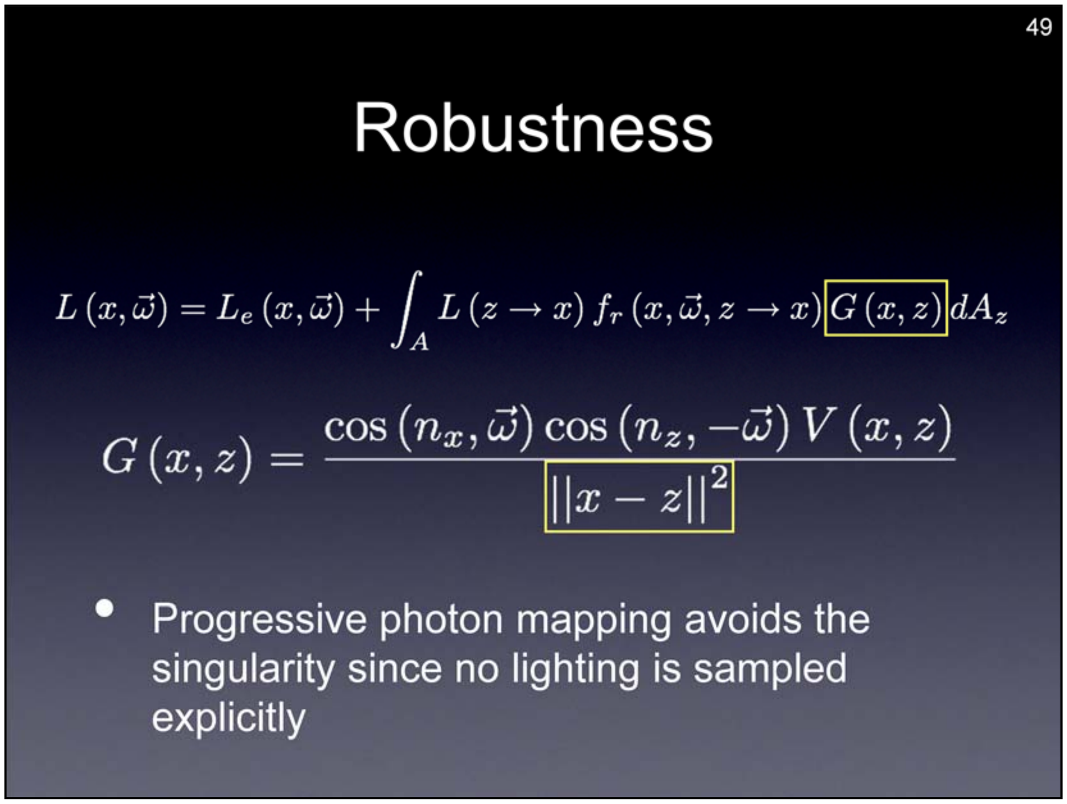
Convergence (6.4M photons)

the image already gives us an idea of the illumination in the scene.



Convergence (25.6M photons)

and finally this image is rendered using about twenty-five million photons in total. Note the absence of bright noisy pixels in the image sequence just shown.



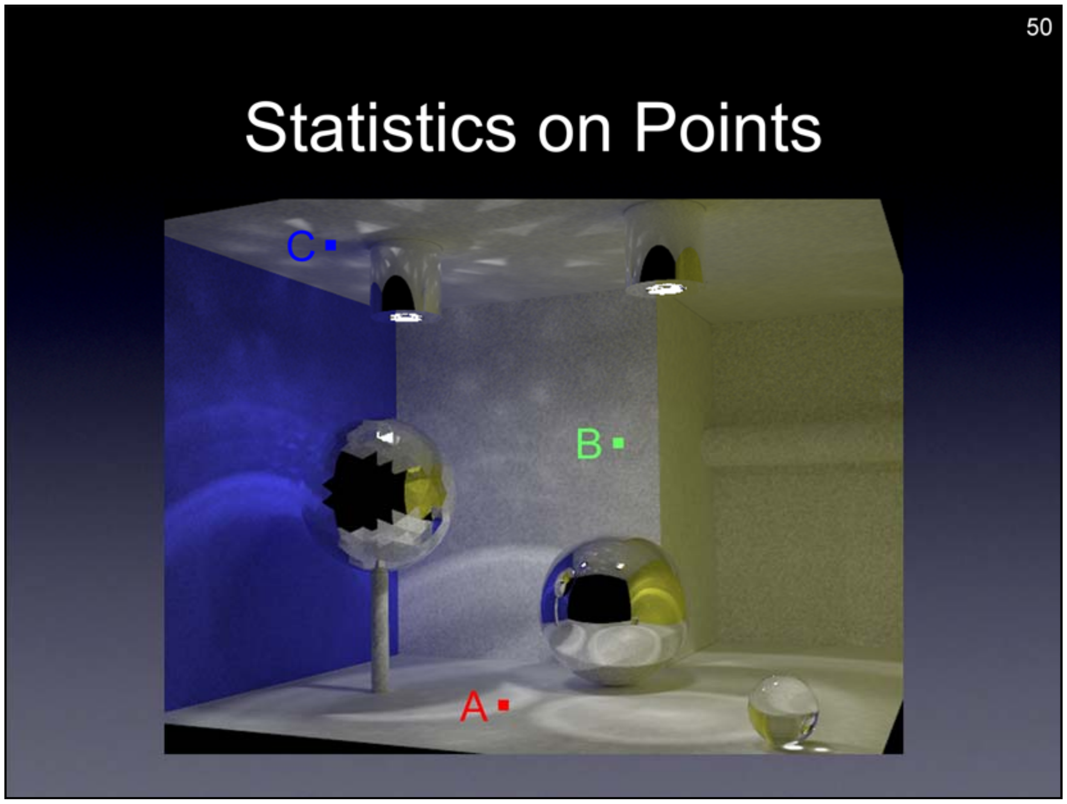
Robustness

$$L(x,\vec{\omega}) = L_e(x,\vec{\omega}) + \int_A L(z \to x)\, f_r(x,\vec{\omega}, z \to x)\, \boxed{G(x,z)}\, dA_z$$

$$G(x,z) = \frac{\cos(n_x,\vec{\omega})\cos(n_z,-\vec{\omega})\, V(x,z)}{\boxed{||x-z||^2}}$$

- Progressive photon mapping avoids the singularity since no lighting is sampled explicitly

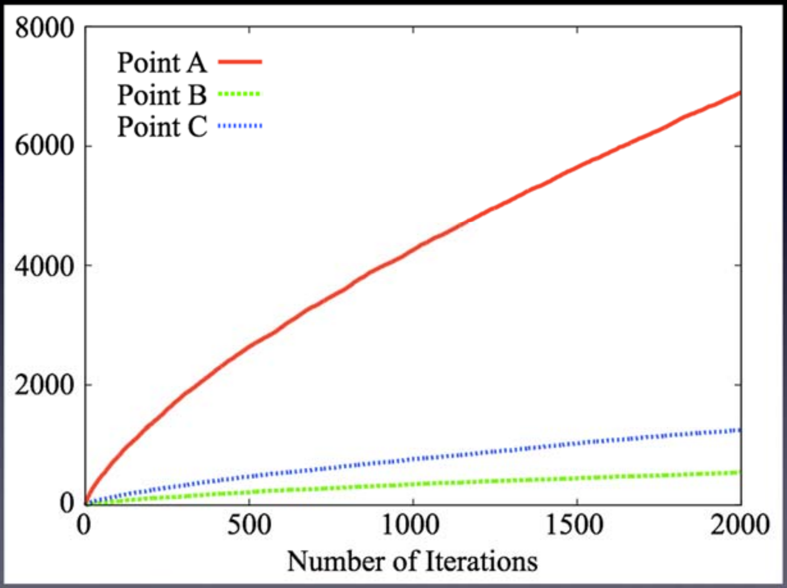
The reason for this is that progressive photon mapping avoids the singularity in the geometry term of the rendering equation by not sampling the light sources explicitly.



Statistics on Points

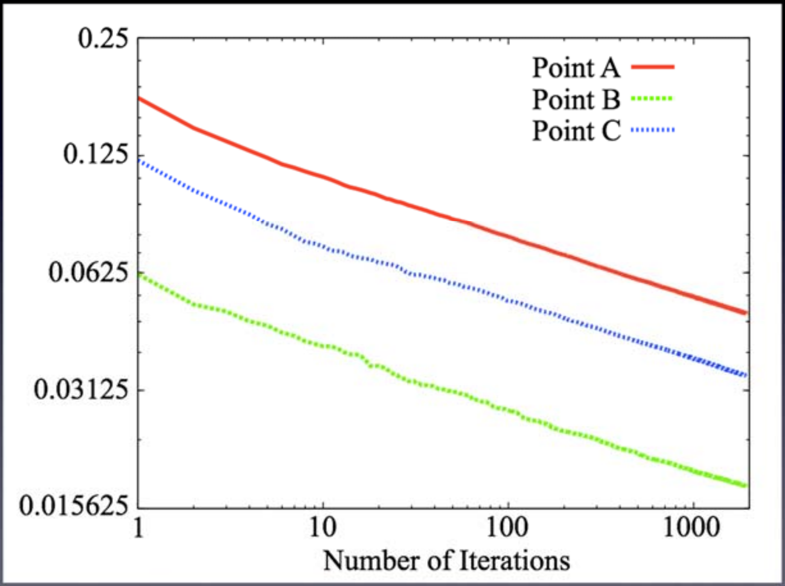
Let’s look at how the radius and the number photons change at different points on the scene. We plot graphs of the radius, the number of nearby photon, and the estimated radiance on the different point shown in here.

Number of Local Photons



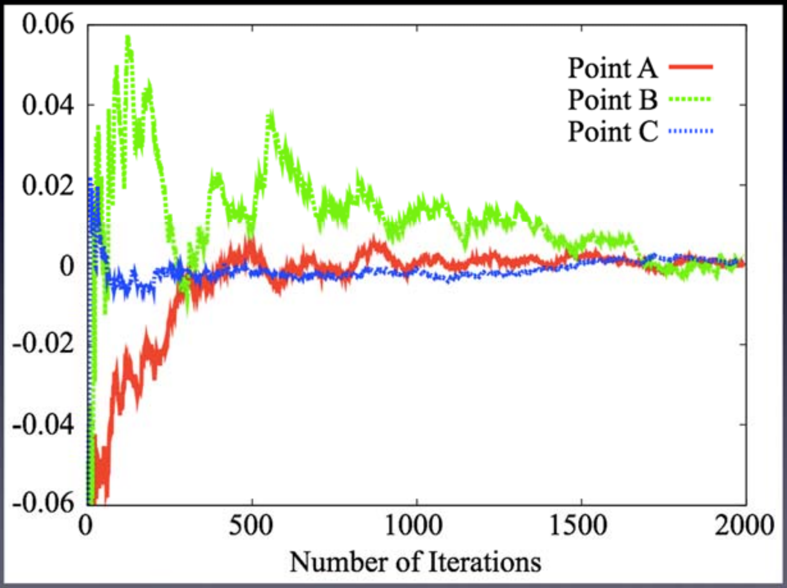
This graph shows the change of the number of photons within the radius of each point. As can be seen, the number of photons increases monotonically as we increase the number of iterations.

Radius

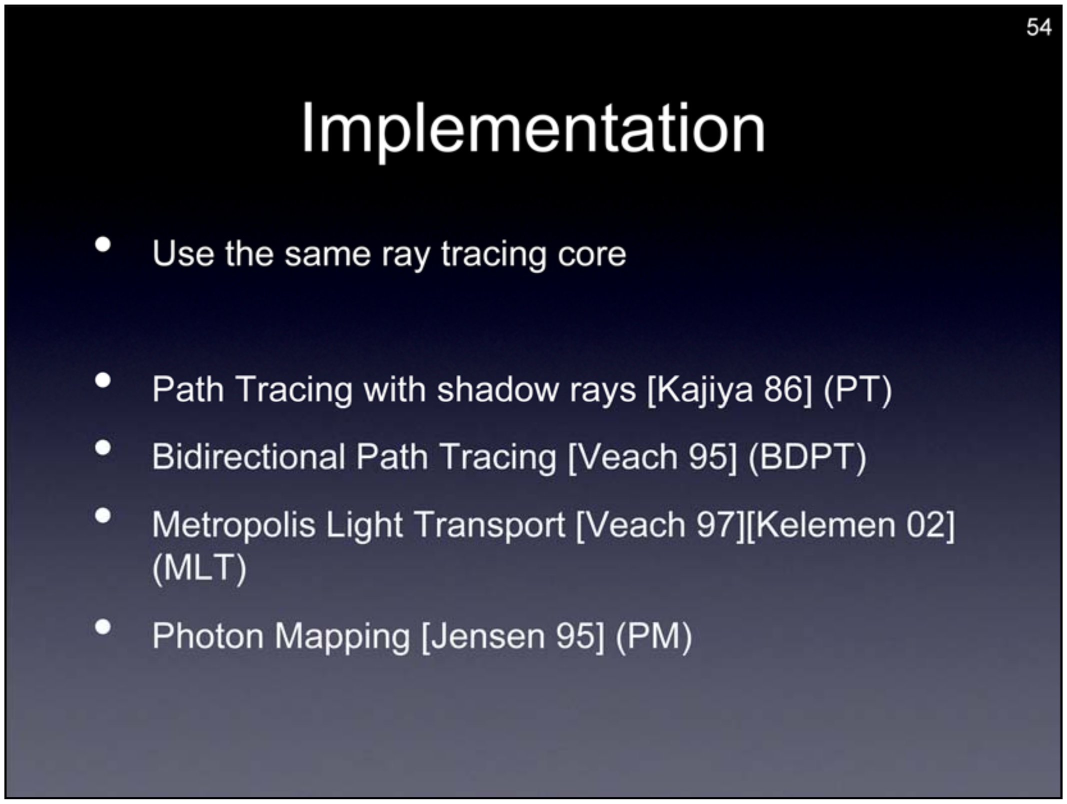


This graph shows the change of the radius. Note that the radius is monotonically decreasing as the number of iterations increase.

Error of Estimate



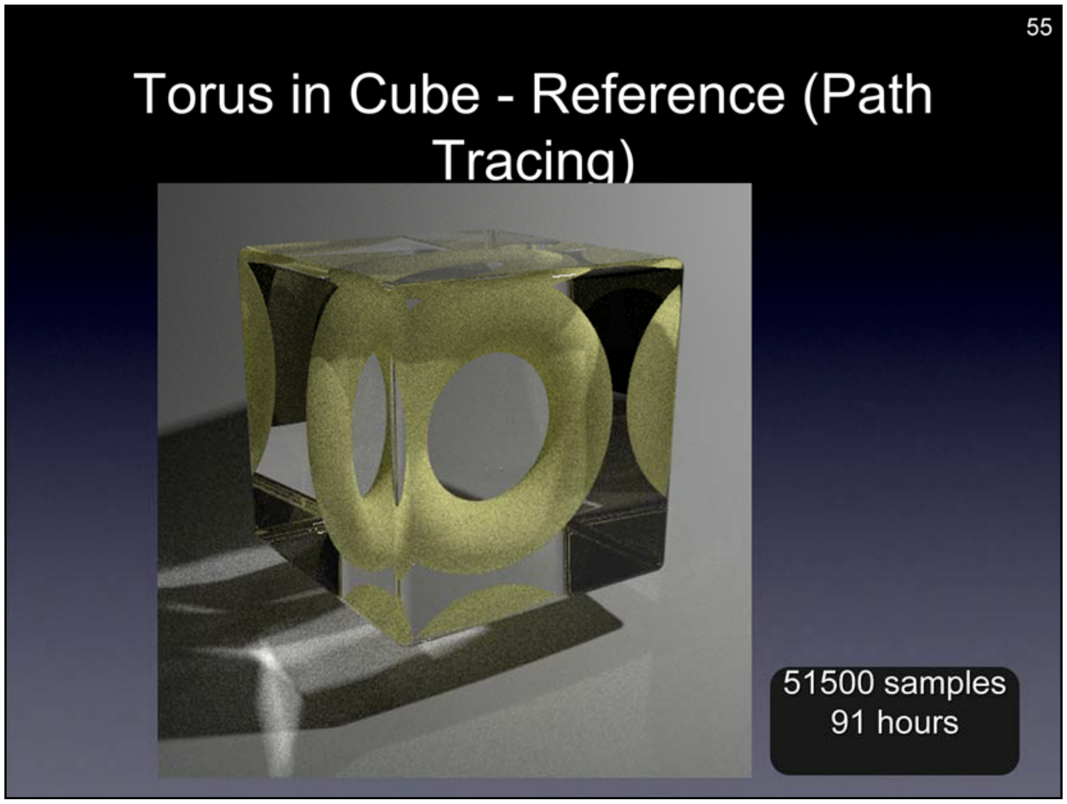
This graph shows the error of the radiance estimate. As can be seen, the radiance estimate converges to the correct solution.



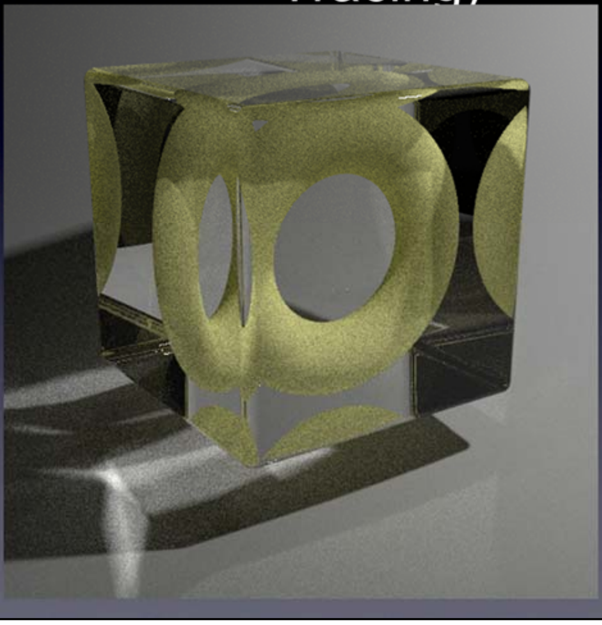
Implementation

- Use the same ray tracing core
- Path Tracing with shadow rays [Kajiya 86] (PT)
- Bidirectional Path Tracing [Veach 95] (BDPT)
- Metropolis Light Transport [Veach 97][Kelemen 02] (MLT)
- Photon Mapping [Jensen 95] (PM)

To compare our method with existing methods, we implemented these algorithms using the same ray tracing core.

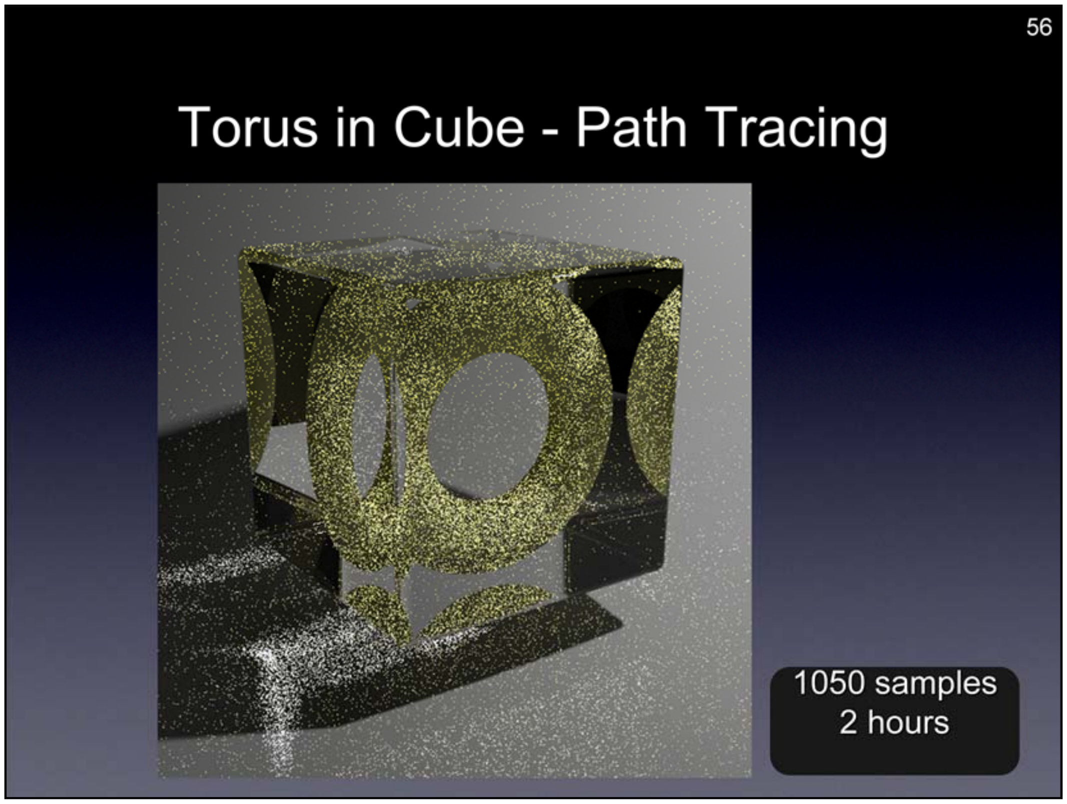


Torus in Cube - Reference (Path Tracing)

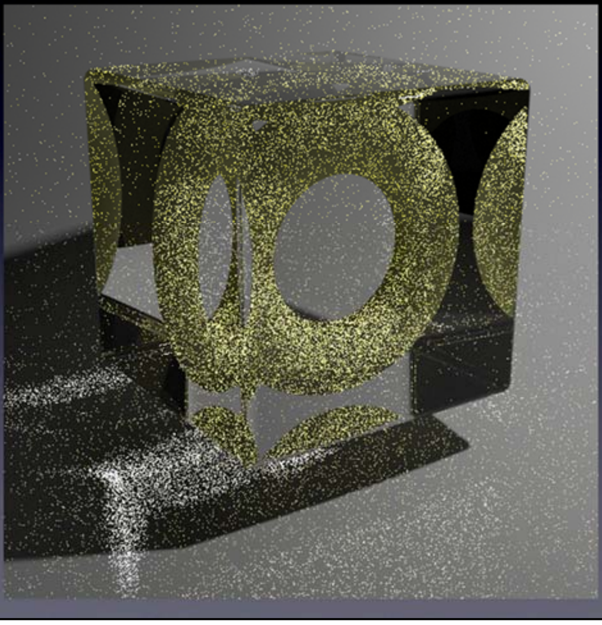


51500 samples
91 hours

First we rendered a torus embedded in a transparent cube illuminated by sunlight. Note that all illumination on the torus is caustic and we see the torus though specular transmission of the cube, which is a SDS path. This is the reference solution with 91 hours of rendering using path tracing.

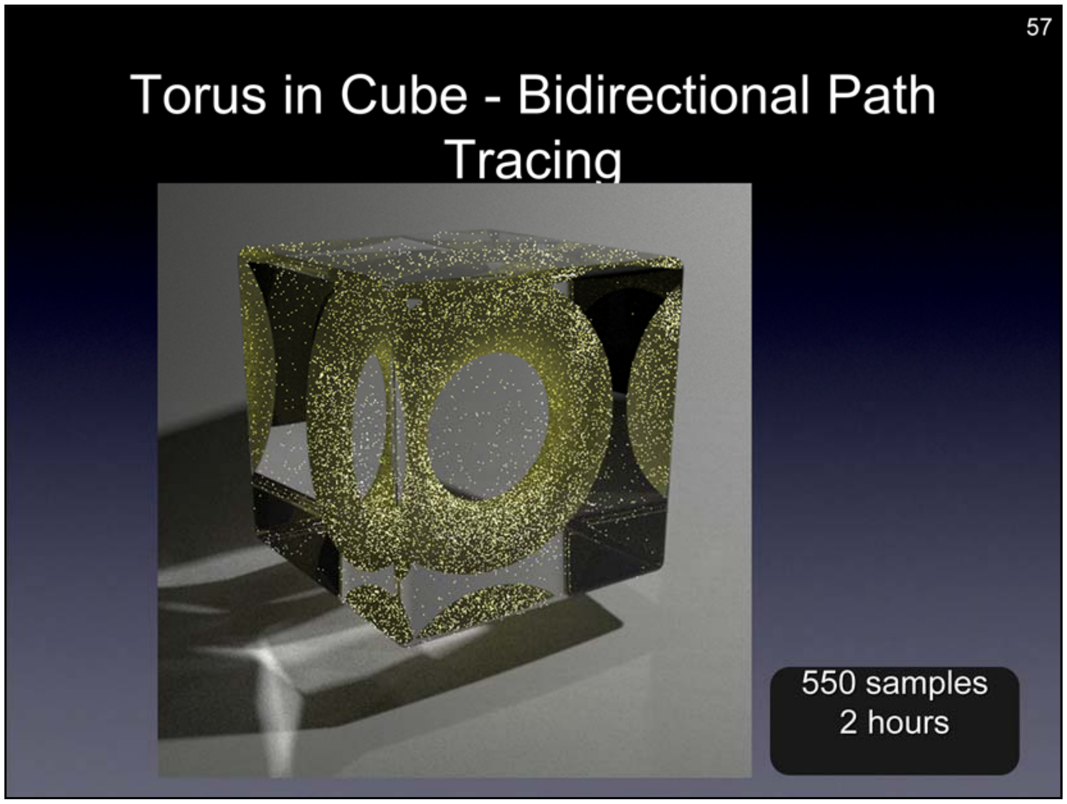


Torus in Cube - Path Tracing



1050 samples
2 hours

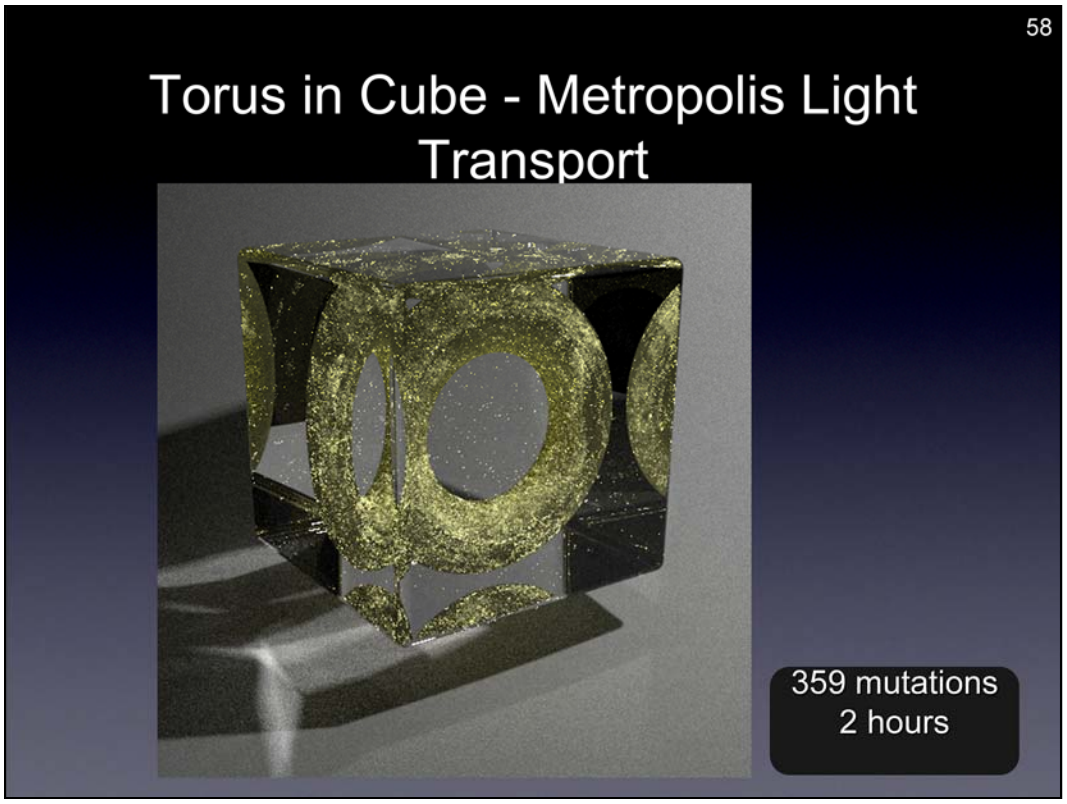
If we just use 2 hours with path tracing, the image looks very noisy.



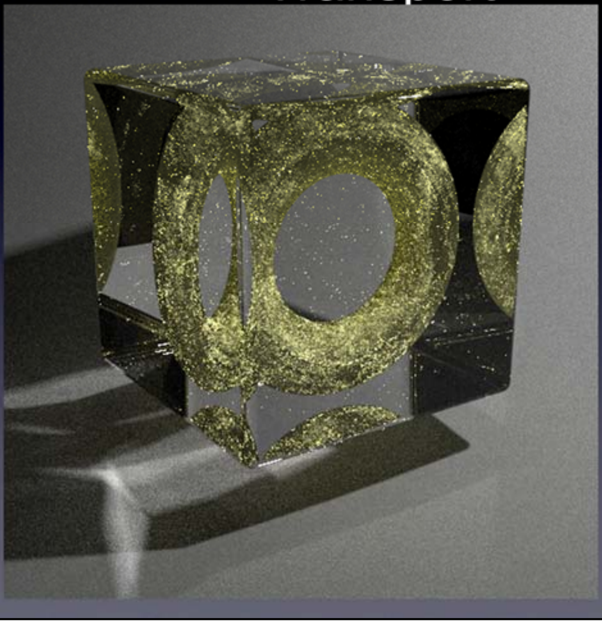
Torus in Cube - Bidirectional Path Tracing

550 samples
2 hours

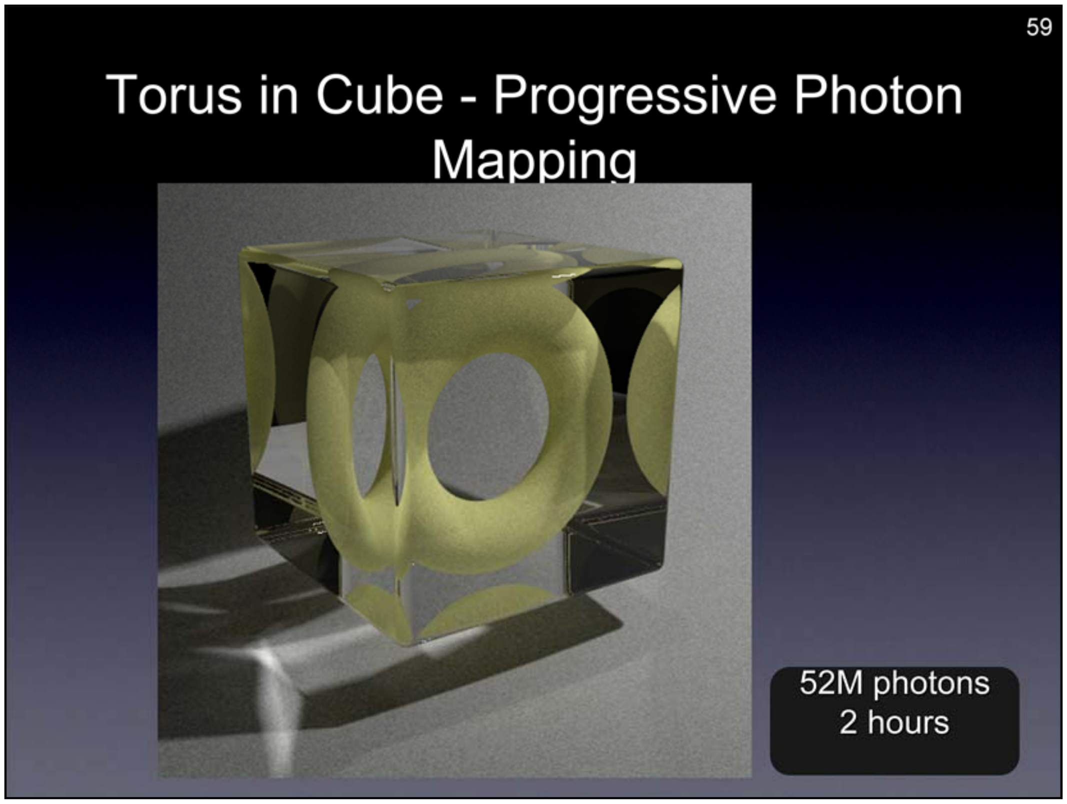
Bidirectional path tracing gives less noisy results especially for the caustics caused by the transparent cube, but the torus is still significantly noisy.



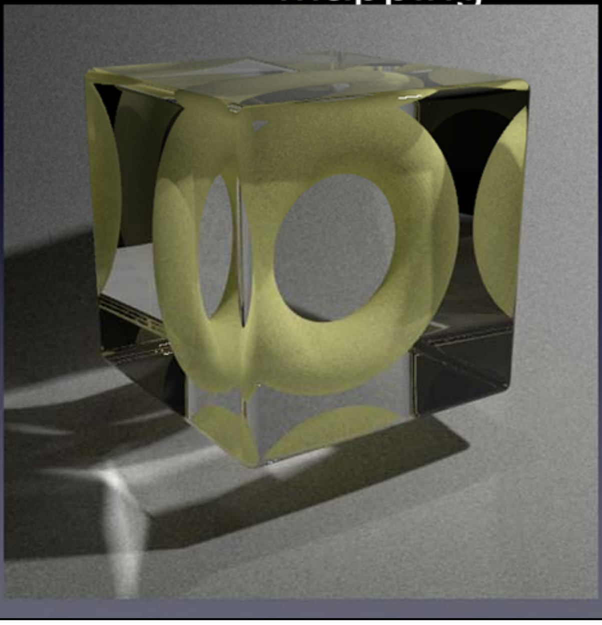
Torus in Cube - Metropolis Light Transport



Metropolis Light Transport does not really work well either in this scene, but it just gives us a different type of artifact.

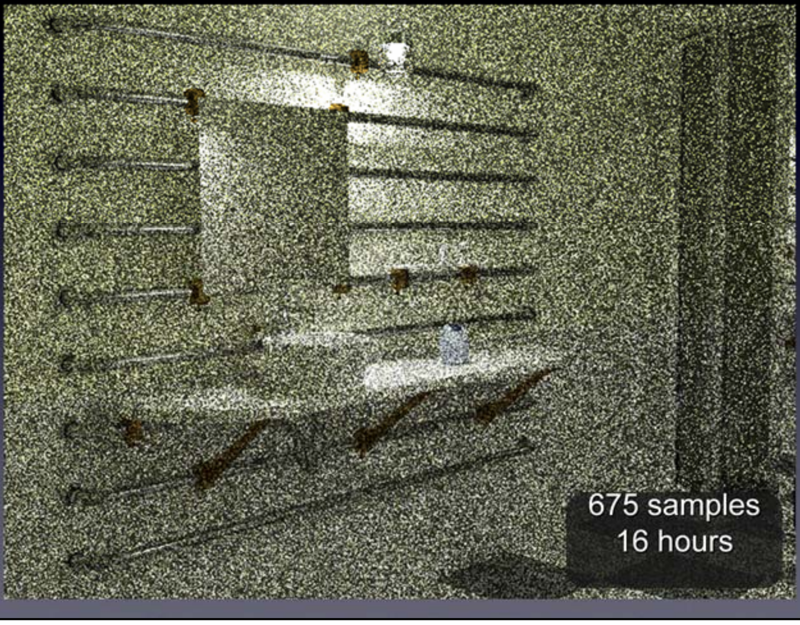


Torus in Cube - Progressive Photon Mapping

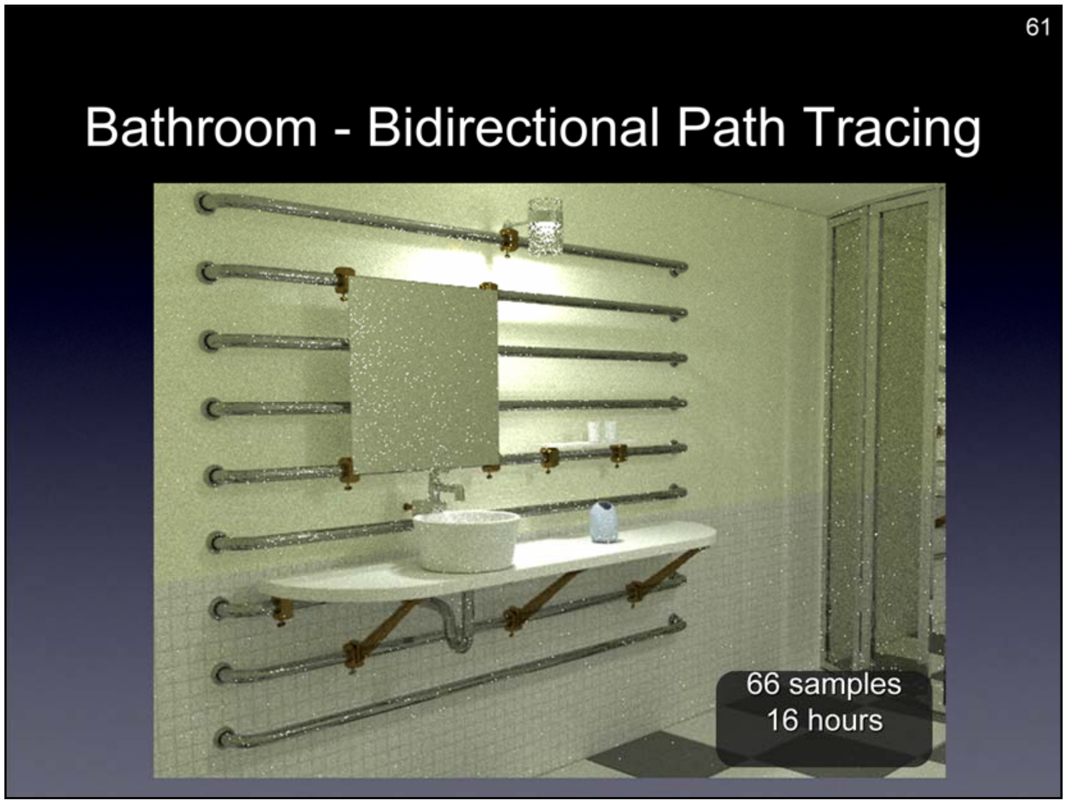


In the same rendering time, our method can handle illumination on the torus very robustly as well as caustics by the cube.

Bathroom - Path Tracing

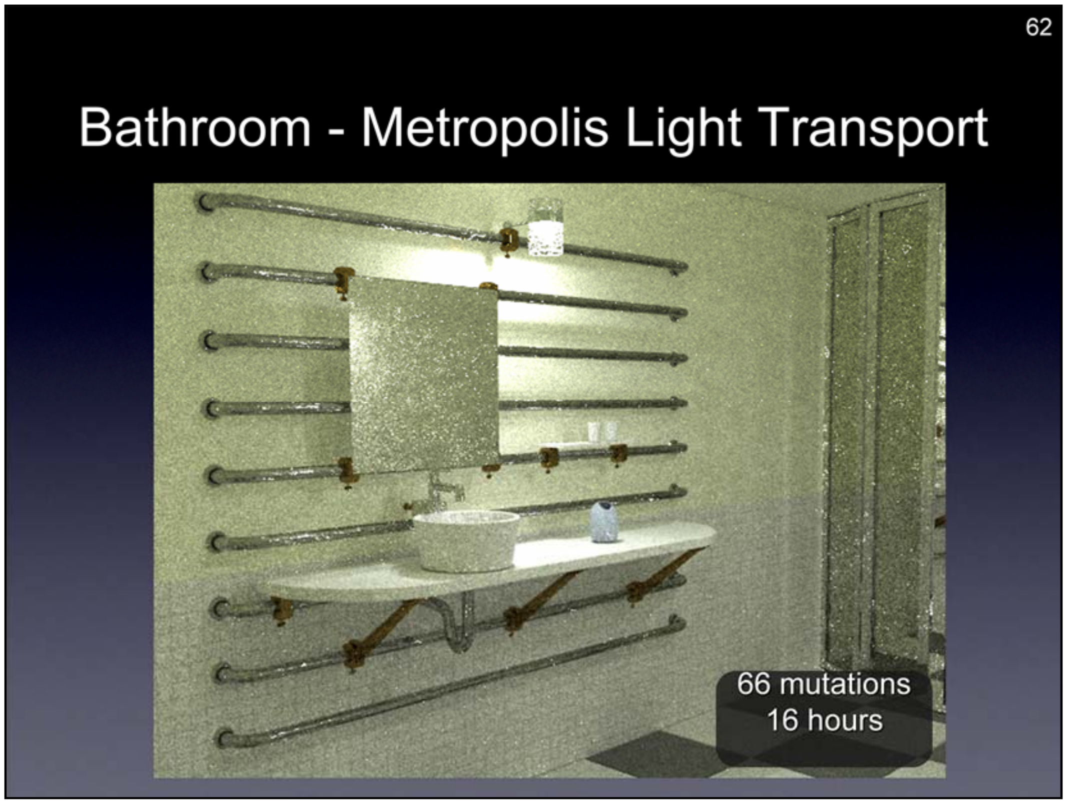


This bathroom scene shows an example of realistic lighting design. In this scene, there are two spherical light sources enclosed by glass casing, which is similar to a typical lighting fixture. Note that reflection on the mirror causes SDS paths. Path tracing results in noisy image because almost everything is illuminated by caustics in this scene.



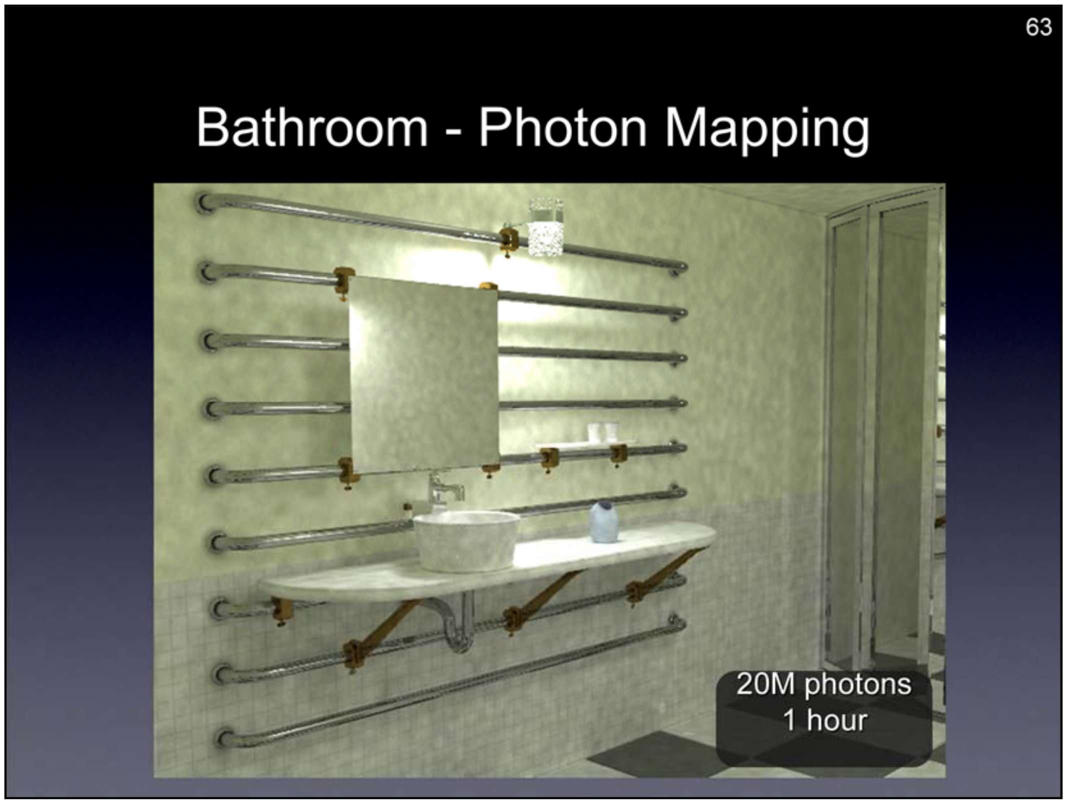
Bathroom - Bidirectional Path Tracing

Bidirectional path tracing gives you much better result in the same rendering time. However, note that the reflection of the light on the mirror is missing.

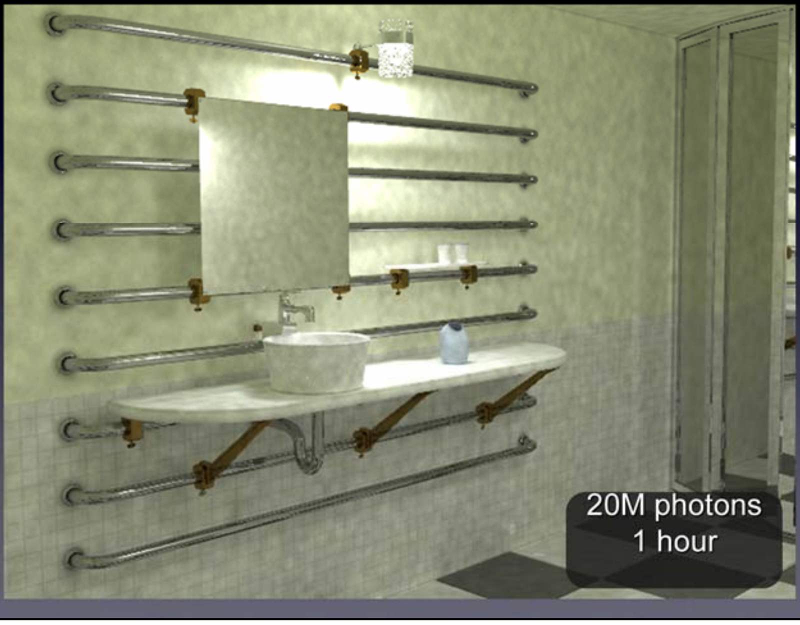


Bathroom - Metropolis Light Transport

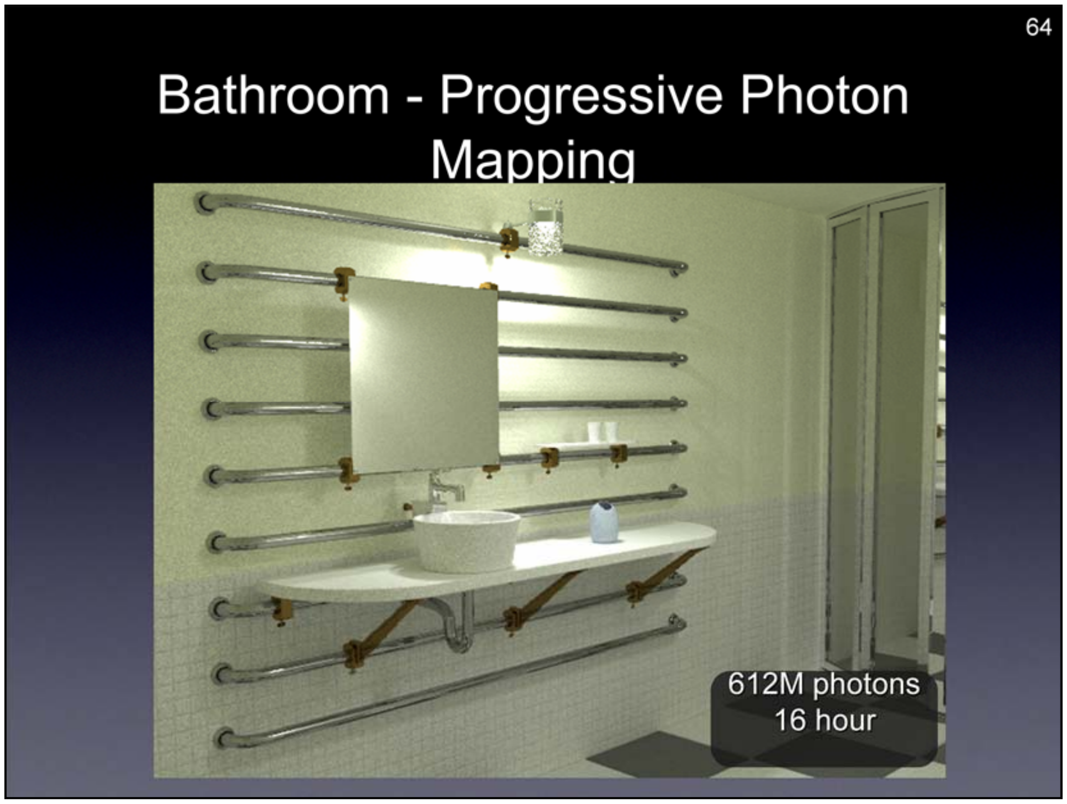
Metropolis Light Transport can capture some of reflections on the mirror, but the results looks still very noisy.



Bathroom - Photon Mapping



Here we used standard photon mapping. The number of photons is as large as we can use in 1GB of memory. Since the number of photons is limited by the amount of memory, the rendering time is actually faster. However the image looks blotchy because the number of photons is not enough to get rid of the noise. This means that the quality of the image is bounded by the available amount of memory.

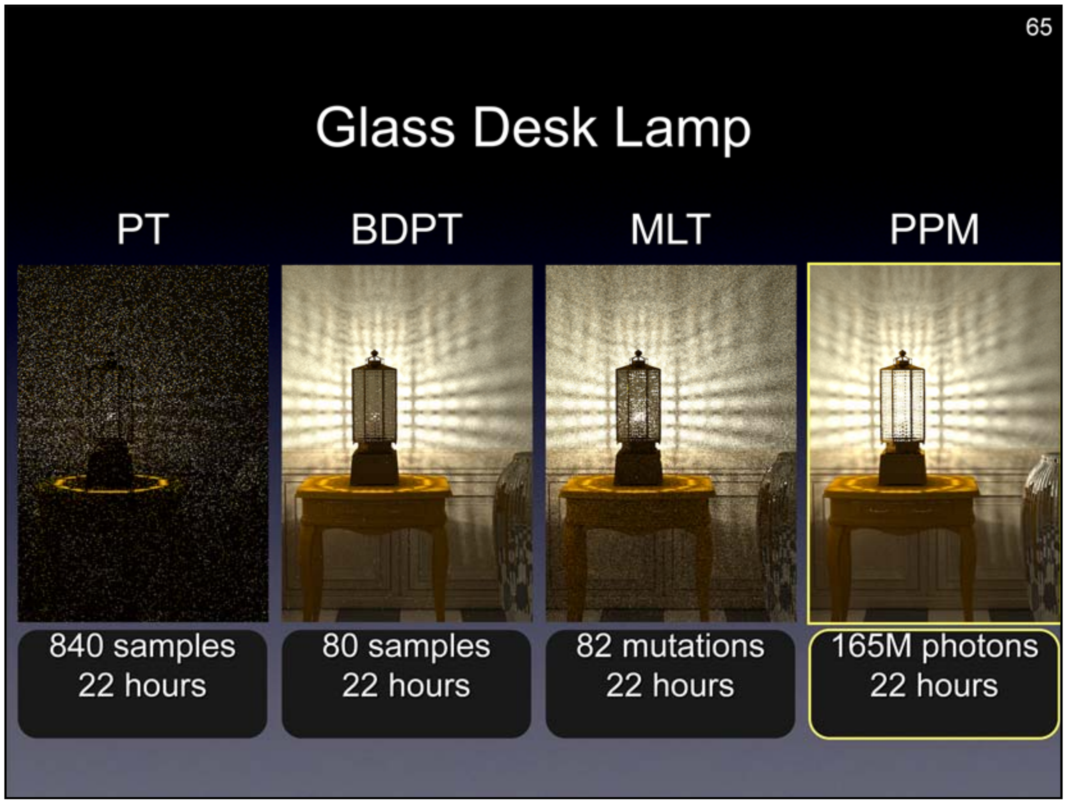


64

Bathroom - Progressive Photon Mapping

612M photons
16 hour

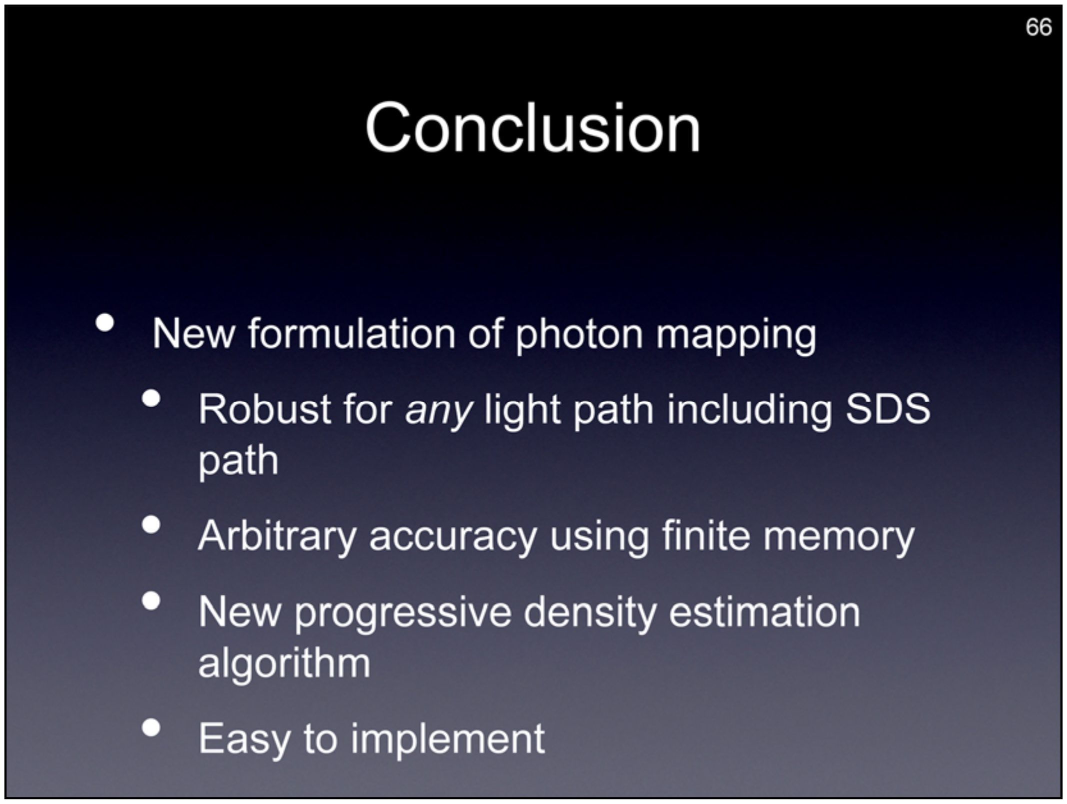
With Progressive Photon Mapping, we can use 612 million photons, which is equivalent to 30GB of photons without consuming that amount of memory. The image accurately captures the reflection in the mirror, as well as all other fine scale illumination details.



Glass Desk Lamp

PT	BDPT	MLT	PPM
840 samples 22 hours	80 samples 22 hours	82 mutations 22 hours	165M photons 22 hours

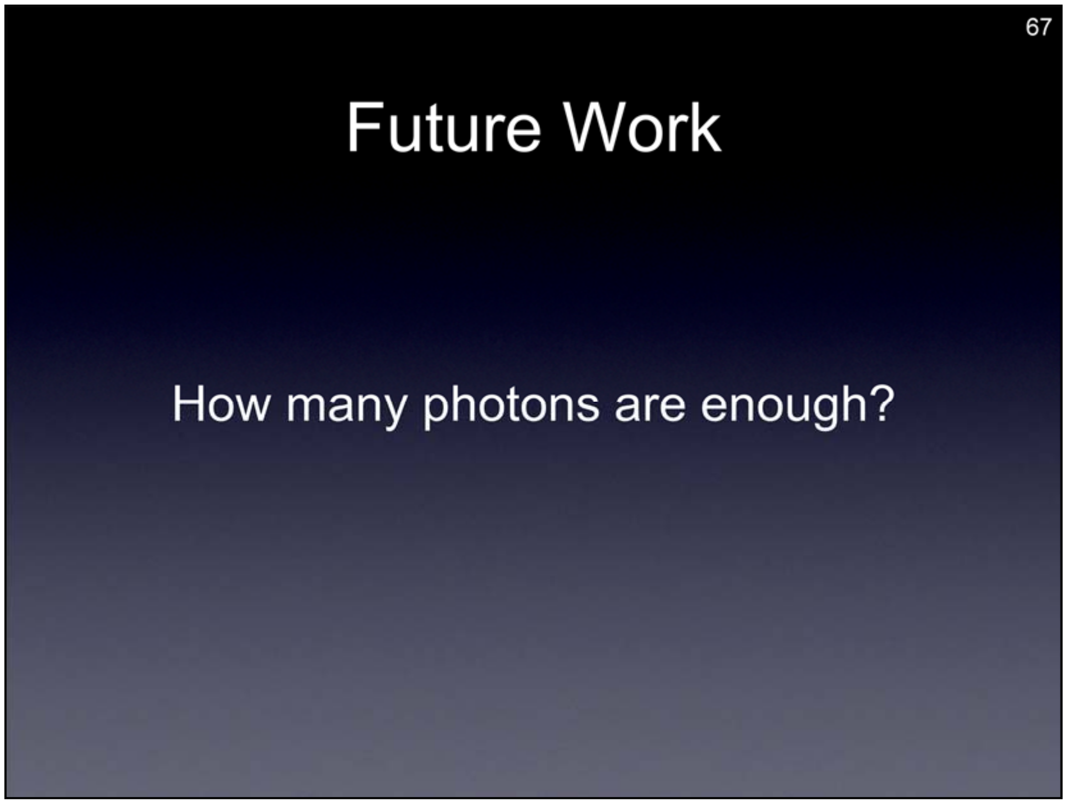
Finally, we rendered a glass desk lamp to show the robustness of our method. The results using existing methods are either too noisy, or they cannot handle the refraction through the lamp. Only progressive photon mapping is able to render this scene without noise.



Conclusion

- New formulation of photon mapping
  - Robust for *any* light path including SDS path
  - Arbitrary accuracy using finite memory
  - New progressive density estimation algorithm
  - Easy to implement

In conclusion, we have presented a new formulation of photon mapping, called progressive photon mapping. Our algorithm is robust and it can compute all types of light transport with arbitrary accuracy using a finite amount of memory. To achieve this, we have introduced a new progressive density estimation algorithm, which is easy to implement. We believe that our method has a lot of interesting future work,

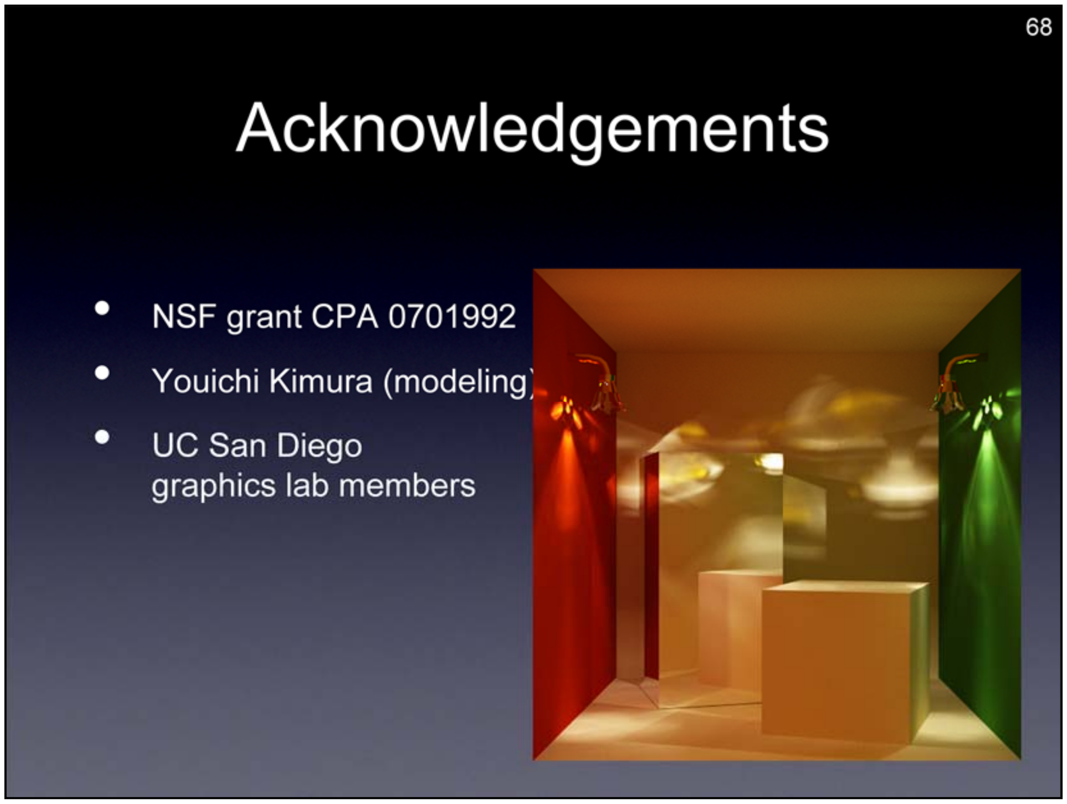


67

Future Work

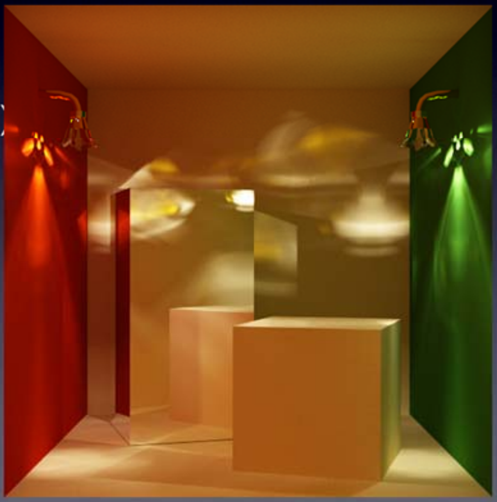
How many photons are enough?

but the most significant question we hope to answer is ‘how many photons are enough for a given error criterion?’
.

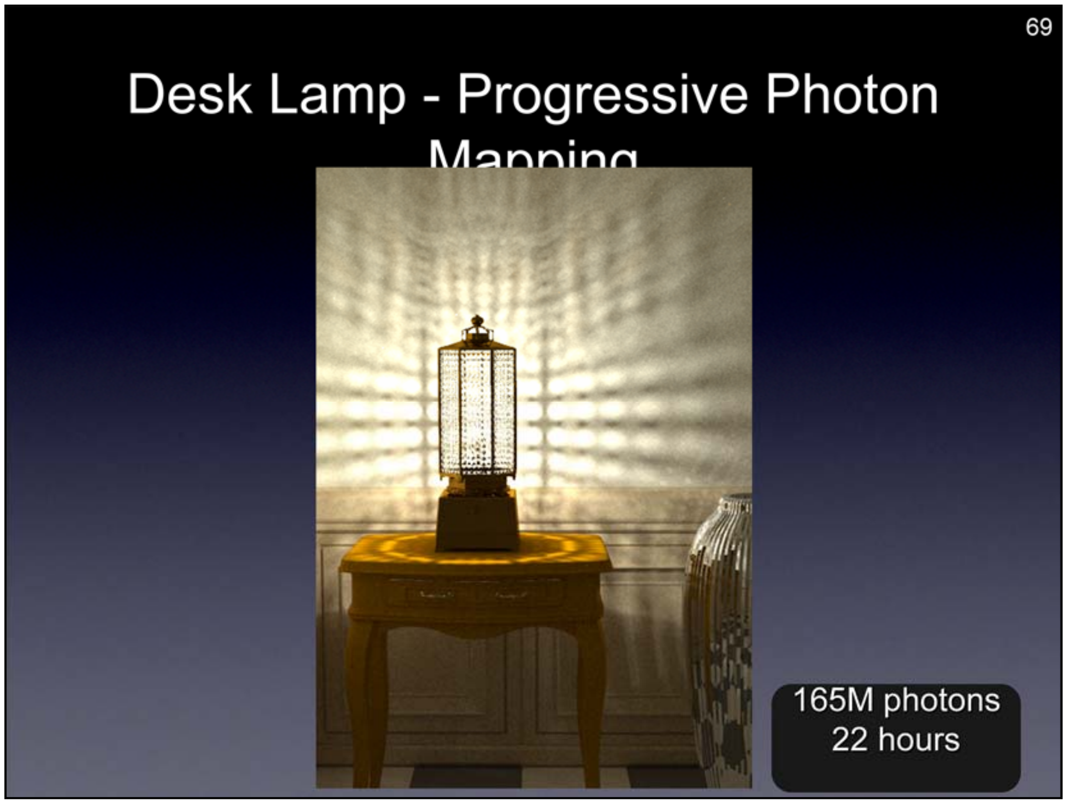


Acknowledgements

- NSF grant CPA 0701992
- Youichi Kimura (modeling)
- UC San Diego graphics lab members



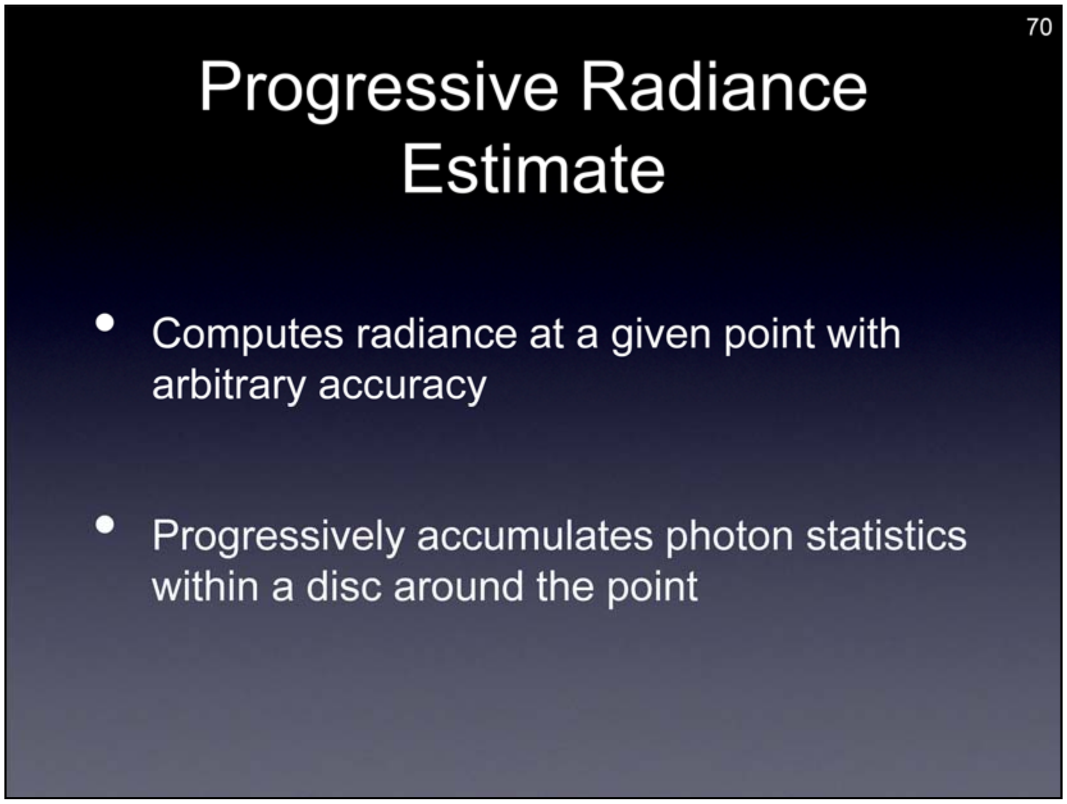
Here are acknowledgments, and thank you for your attention.



Desk Lamp - Progressive Photon Mapping

Our method is the only method that can robustly handle this difficult, yet simple illumination setting. We believe that our method is a robust alternative to render accurate images over existing unbiased Monte Carlo method.

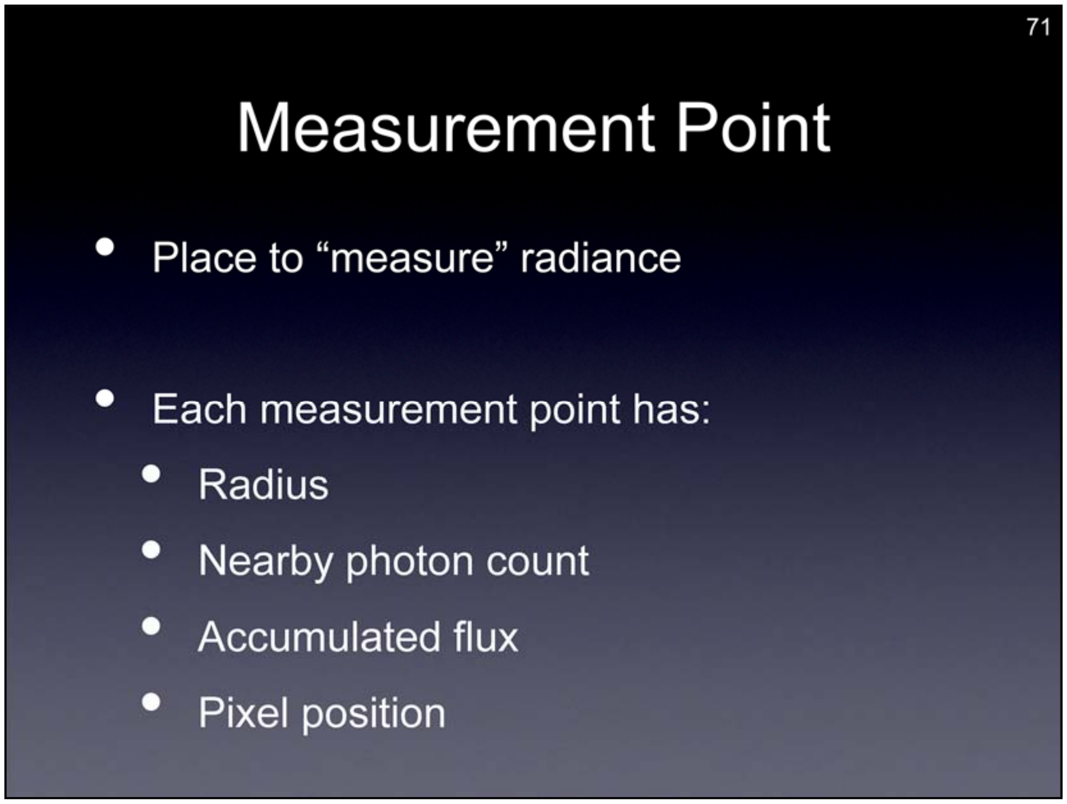
m



Progressive Radiance Estimate

- Computes radiance at a given point with arbitrary accuracy
- Progressively accumulates photon statistics within a disc around the point

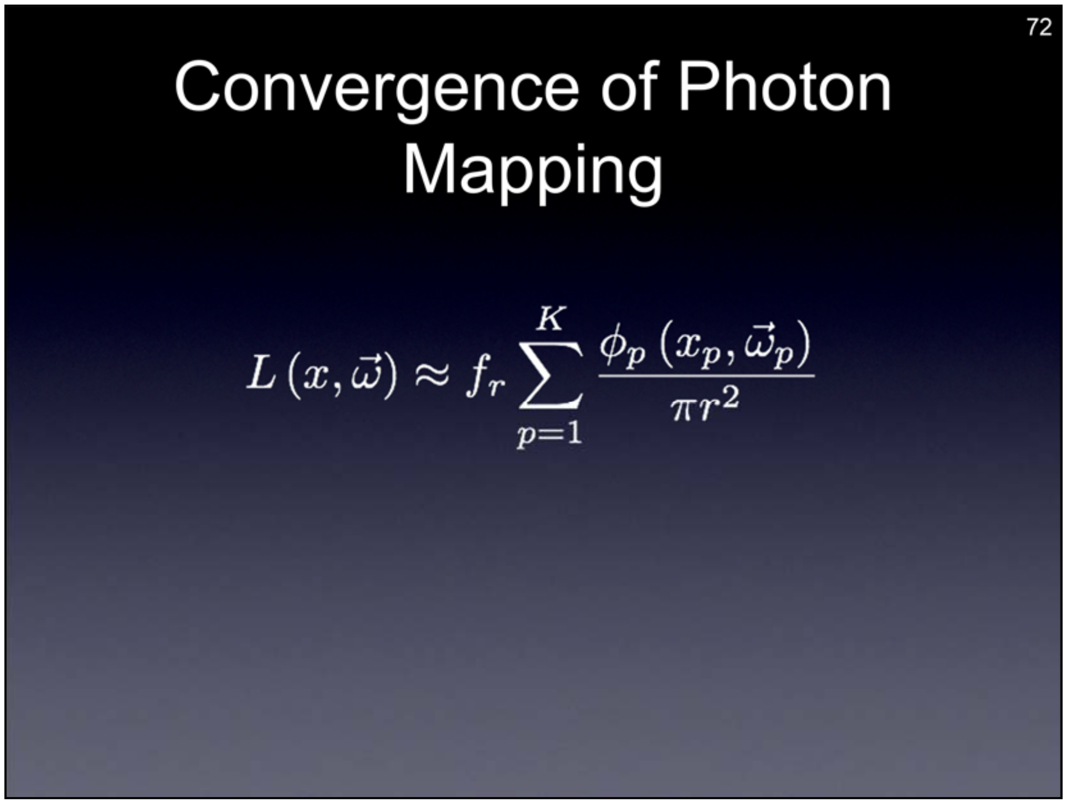
The goal of progressive photon mapping is to compute radiance at a given point with arbitrary accuracy. We achieve this by progressively accumulating photon statistics at



Measurement Point

- Place to “measure” radiance
- Each measurement point has:
    - Radius
    - Nearby photon count
    - Accumulated flux
    - Pixel position

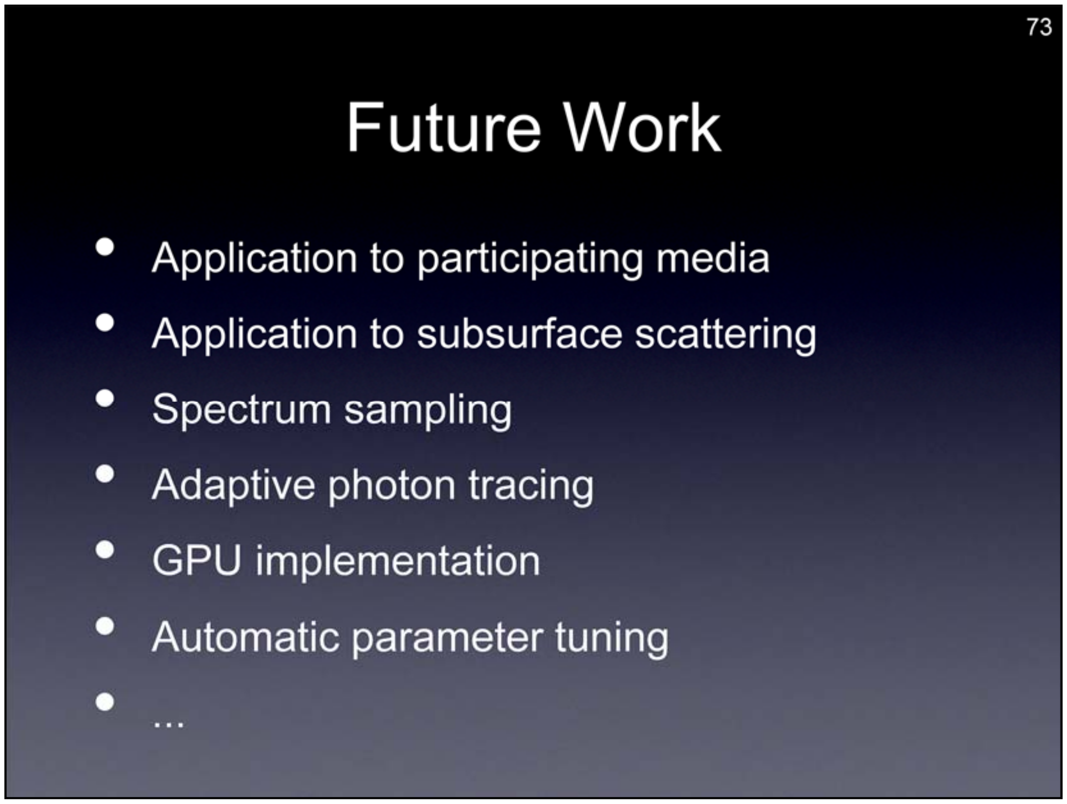
Measurement points generated in the first pass are places to measure radiance. Generating measurement points is typically done by ray tracing for rendering images. Each measurement point stores radius to search nearby photons, the number of nearby photons, accumulated flux of nearby photons and its pixel position.



Convergence of Photon Mapping

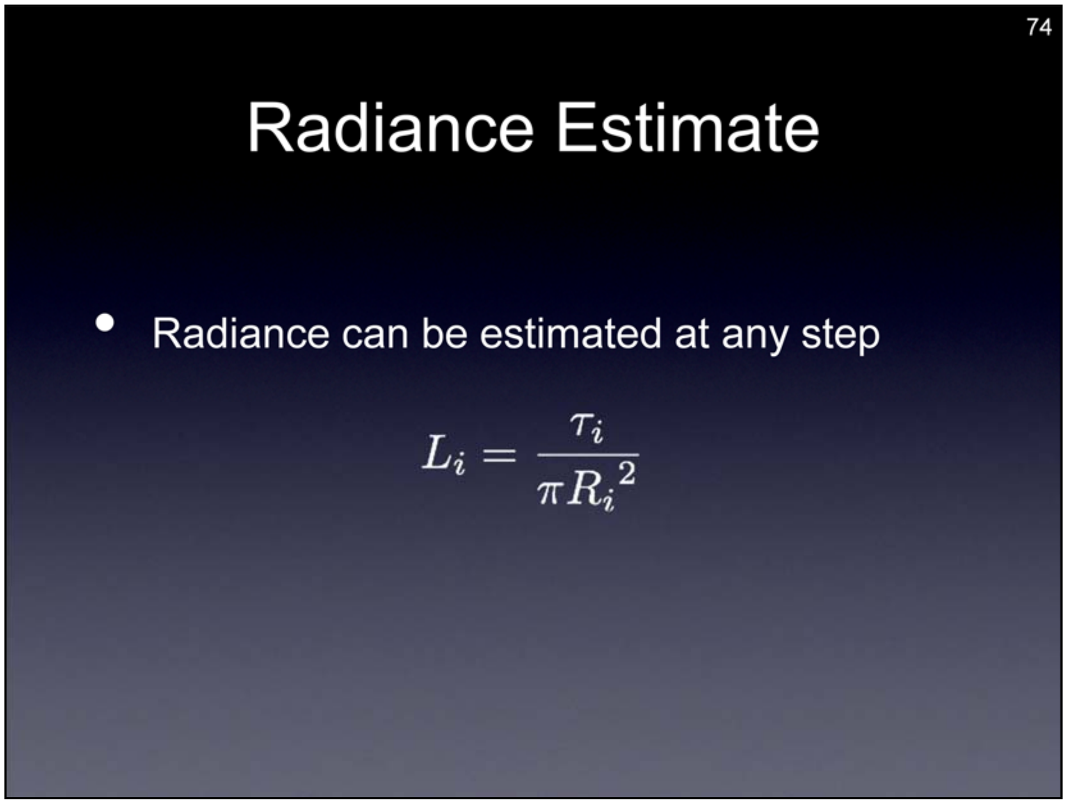
$$L(x,\vec{\omega}) \approx f_r \sum_{p=1}^{K} \frac{\phi_p(x_p,\vec{\omega}_p)}{\pi r^2}$$

To be more concrete, radiance computed from neighboring photons converges to the correct value based on the rendering equation if we use an infinite number of photons. Of course, directly utilizing this property is not practical because an infinite number of photons requires infinite amount of memory to store.



Future Work

- Application to participating media
- Application to subsurface scattering
- Spectrum sampling
- Adaptive photon tracing
- GPU implementation
- Automatic parameter tuning
- ...

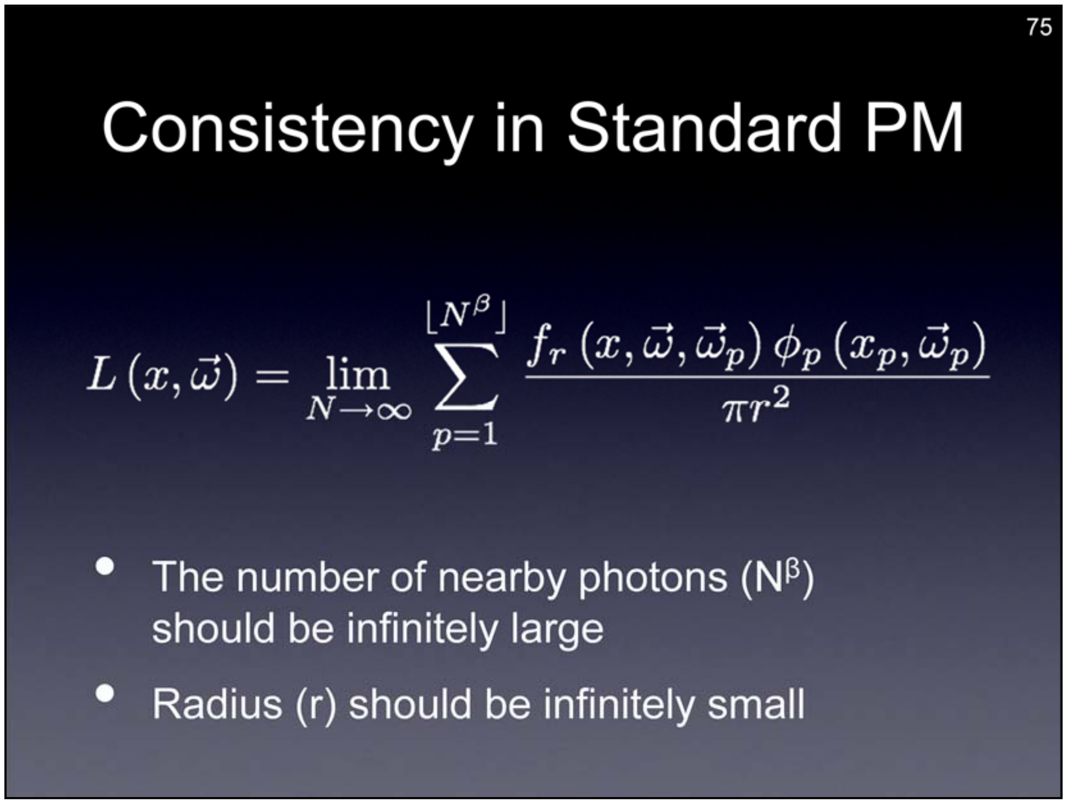


Radiance Estimate

- Radiance can be estimated at any step

$$L_i = \frac{\tau_i}{\pi {R_i}^2}$$

Since flux and radius is available in every refinement, we can estimate radiance to show intermediate images to user. The equation to compute is exactly the same as standard photon mapping, except it uses accumulated, but unnormalized photon flux, which needs to be divided by the number of emitted photons so far over all refinement passes.

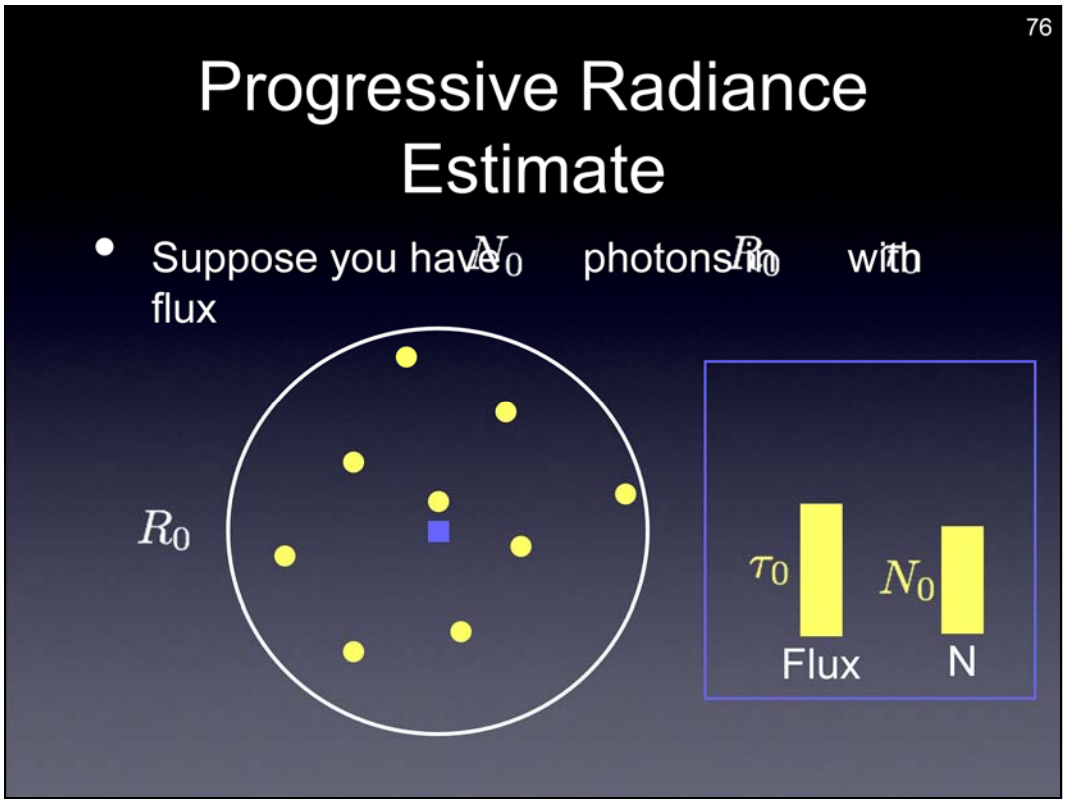


Consistency in Standard PM

$$L(x,\vec{\omega}) = \lim_{N\to\infty} \sum_{p=1}^{\lfloor N^{\beta} \rfloor} \frac{f_r(x,\vec{\omega},\vec{\omega}_p)\,\phi_p(x_p,\vec{\omega}_p)}{\pi r^2}$$

- The number of nearby photons (N^{β}) should be infinitely large
- Radius (r) should be infinitely small

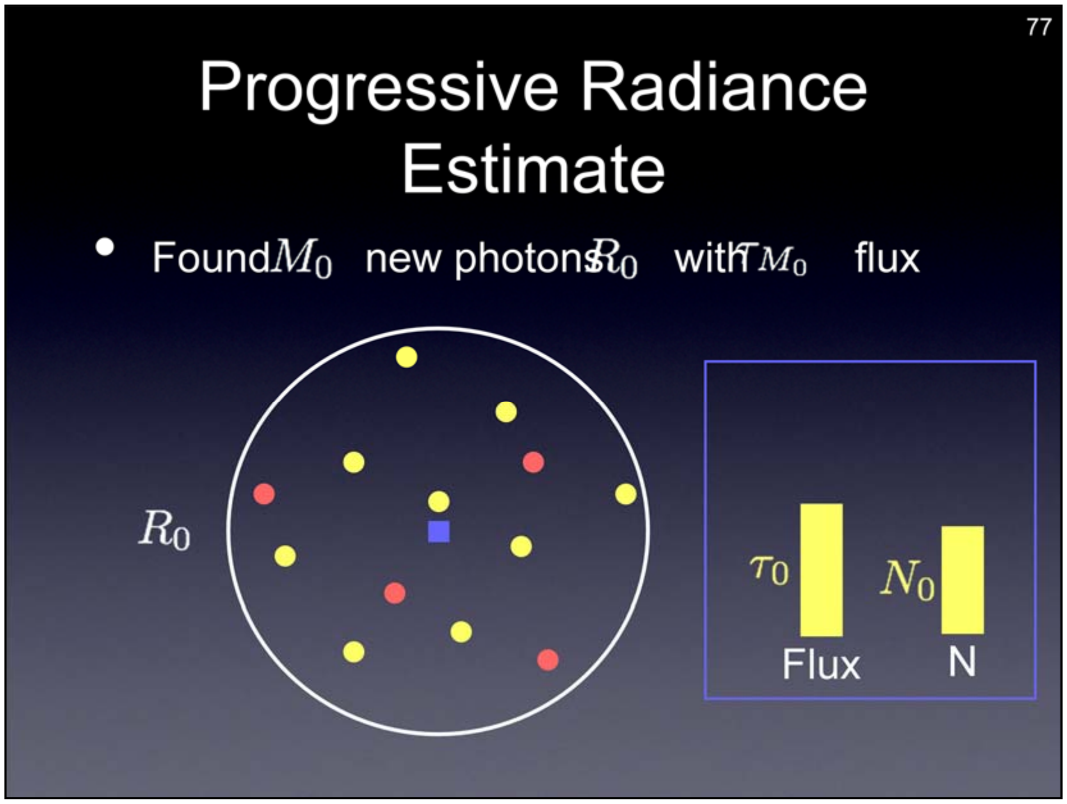
In order to describe how we update statistics on each measurement point, let me recap the consistency of the standard photon mapping. This equation basically says, photon mapping will give us correct radiance if the number of nearby photons is infinitely large and the radius that contains nearby photons is infinitely small.



Progressive Radiance Estimate

- Suppose you have N_0 photons in R_0 with τ_0 flux

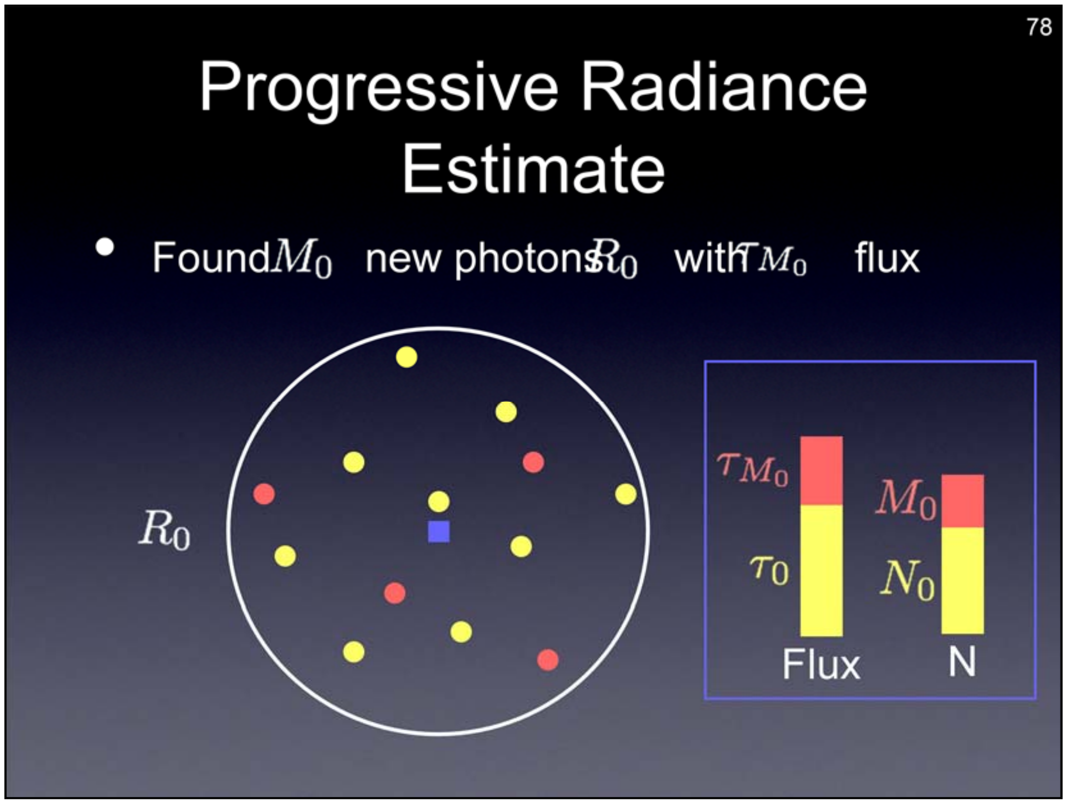
In order to understand how this can be achieved, let’s look a single measurement point in more details. Suppose that you already have N0 photons within the radius of R0 with tau0 as flux.



Progressive Radiance Estimate

- Found M_0 new photons R_0 with τ_{M_0} flux

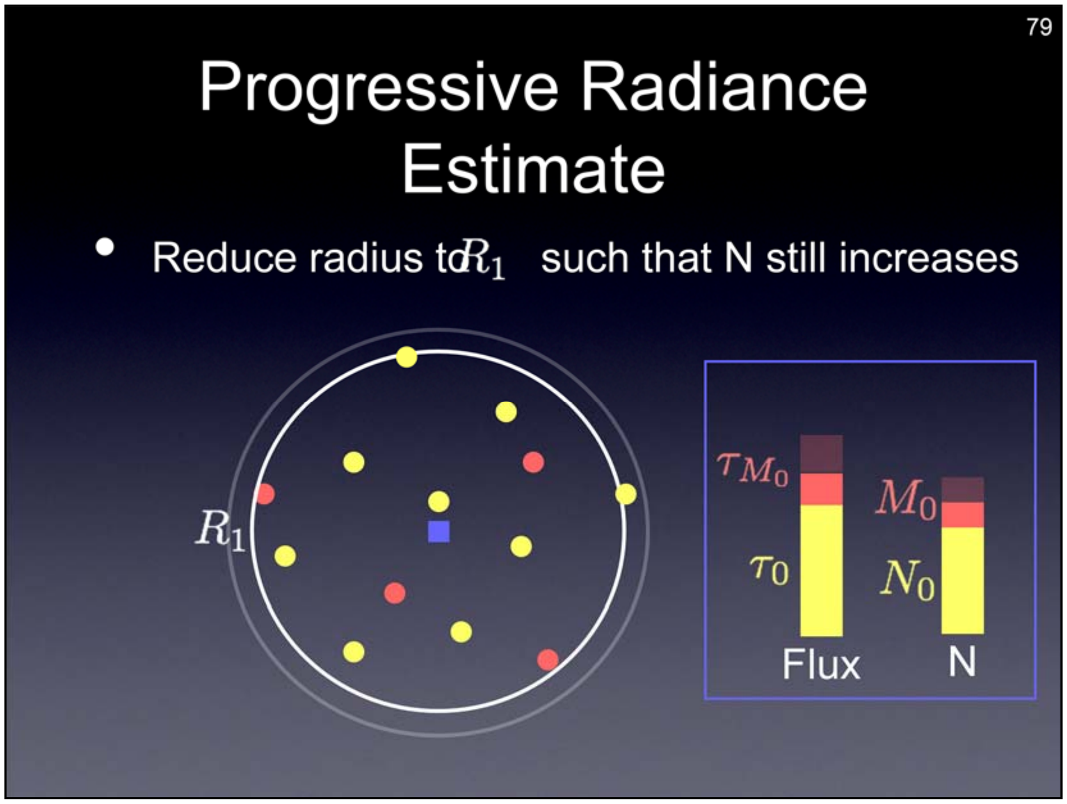
Now, say we find M0 new photons by shooting photons in this iteration.



Progressive Radiance Estimate

- Found M_0 new photons R_0 with τ_{M_0} flux

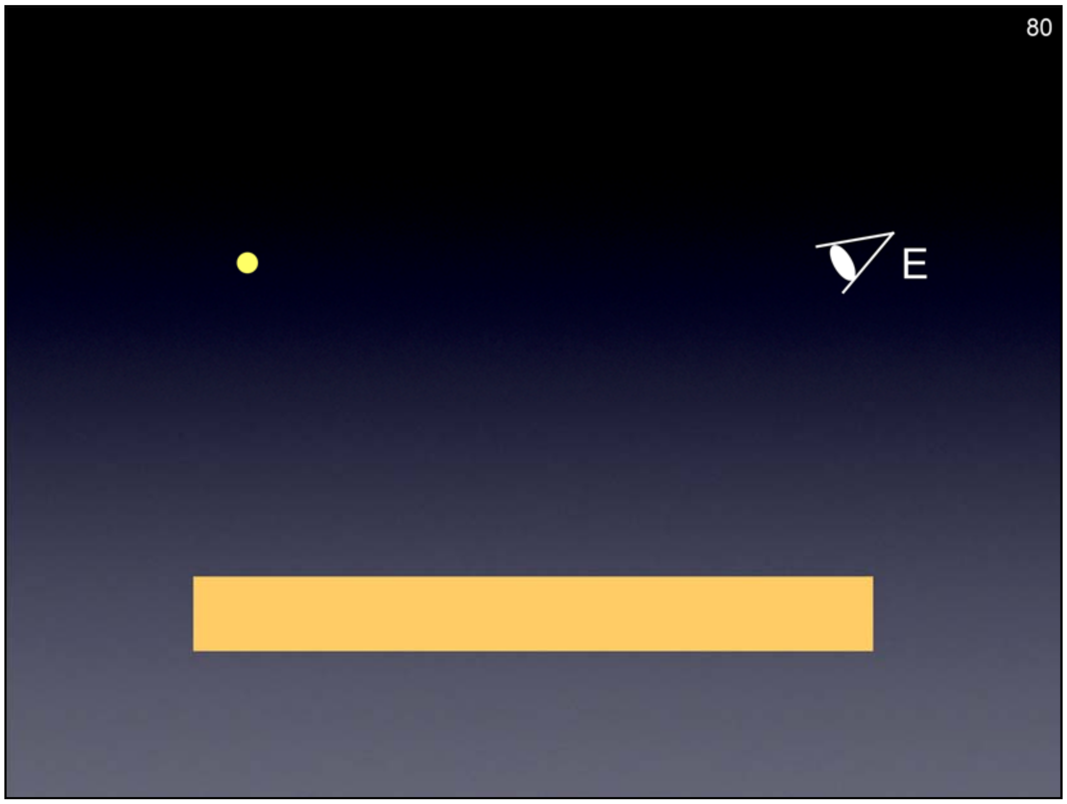
Since every photon is in the radius, we can simply accumulate flux and the number of local photons.



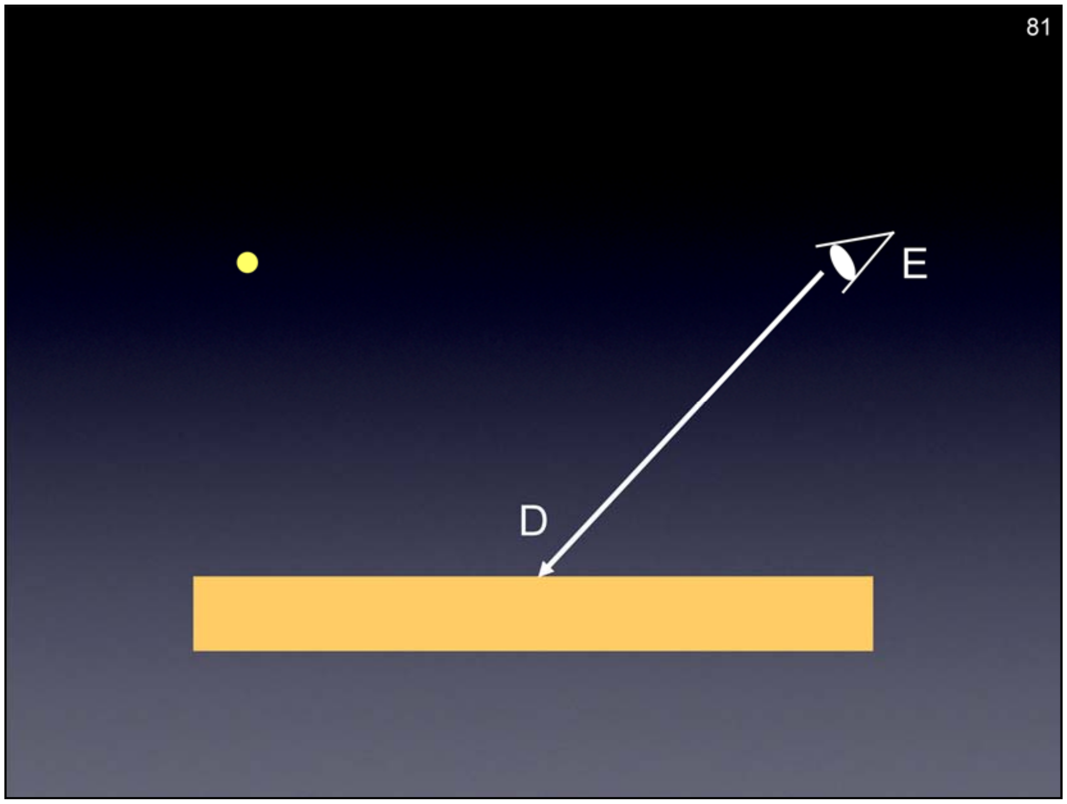
Progressive Radiance Estimate

- Reduce radius to R_1 such that N still increases

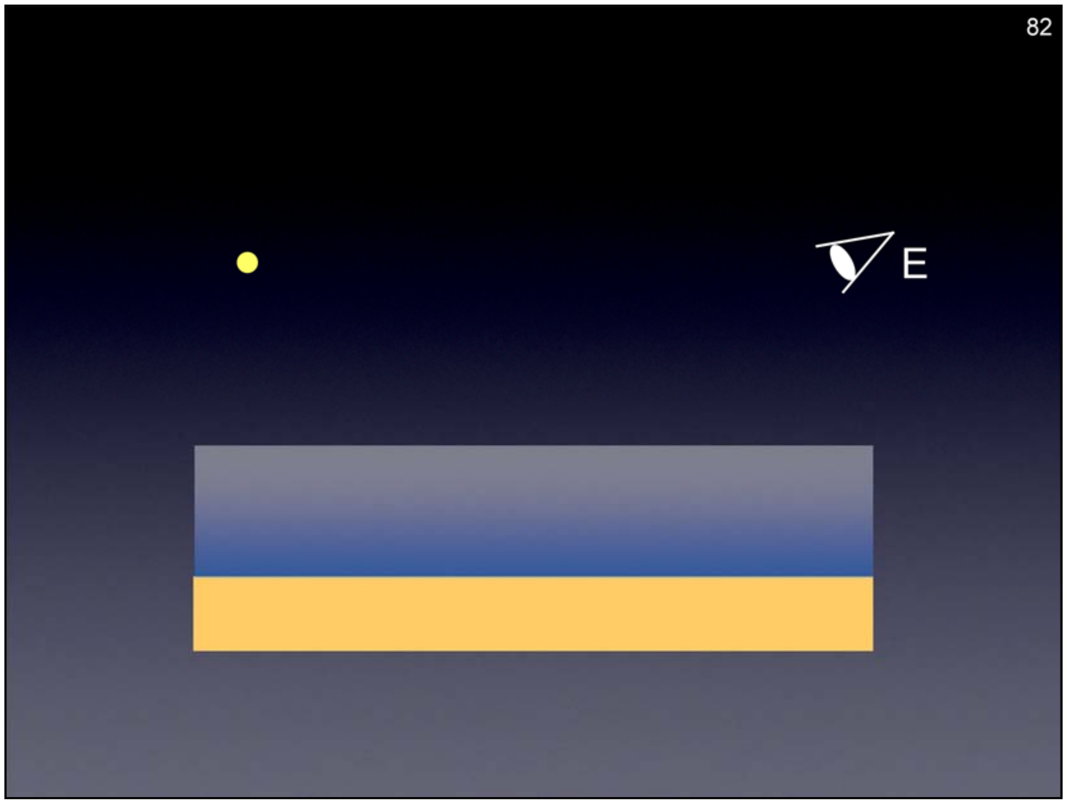
What we really want to do is to determine new radius R1, which is smaller than R0. R1 also needs to keep the number of photons N increasing as well as flux. In other words, we need to have non-zero gain in the number of photons and flux even after the reduction of radius.



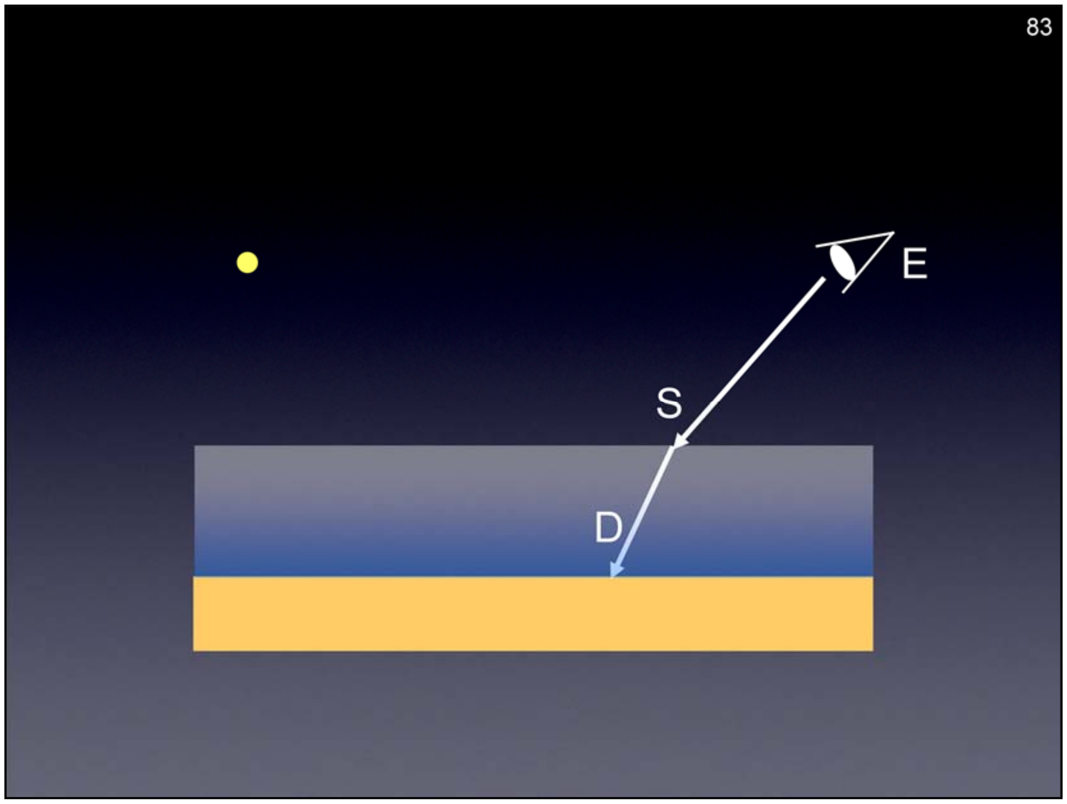
However, there is a type of path which is problematic. In order to see such example, let’s look at this simple scene, where a point light is illuminating a diffuse object. To compute the contribution from the light source to the eyes,



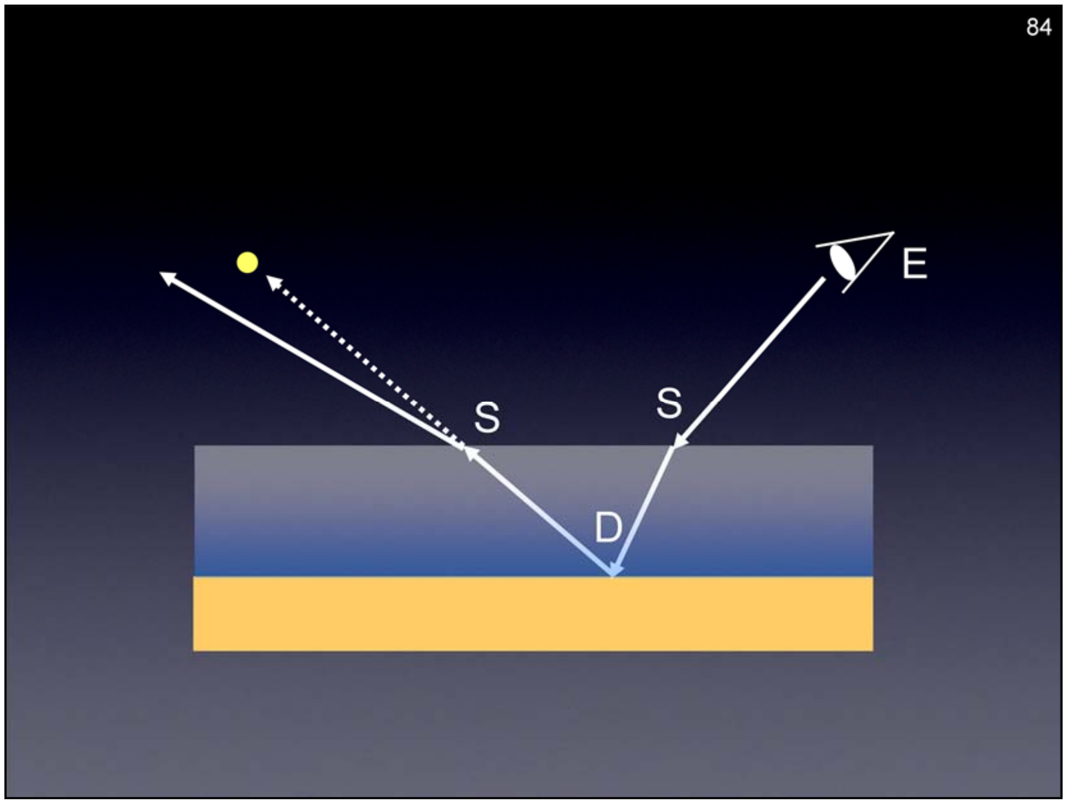
we shoot a ray from the eyes,



Let’s modify the scene slightly by adding a refractive object on top of the diffuse object. In order to compute the contribution from the light source to the eyes,

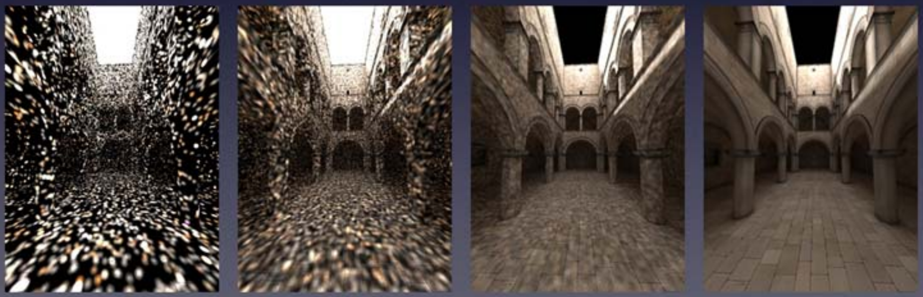


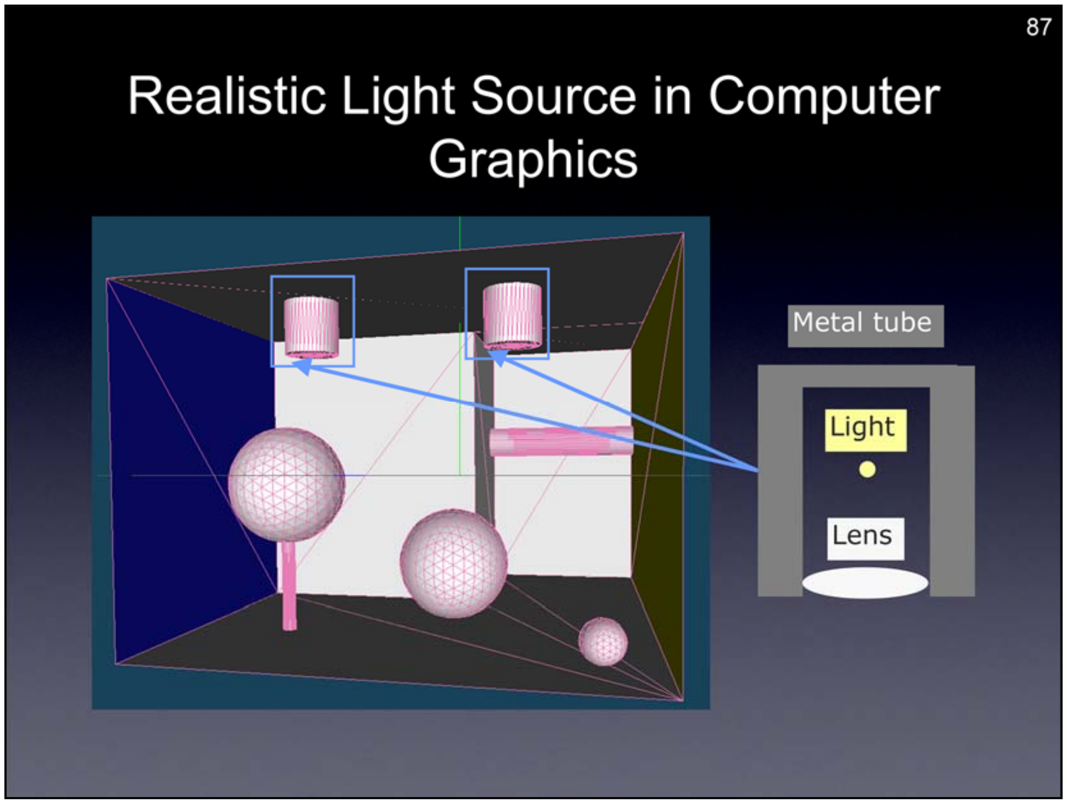
we first shoot a ray from the eyes as before. The only difference is that we need to take into account the specular refraction, which is denoted by S. So far, the computation is still as easy as before.



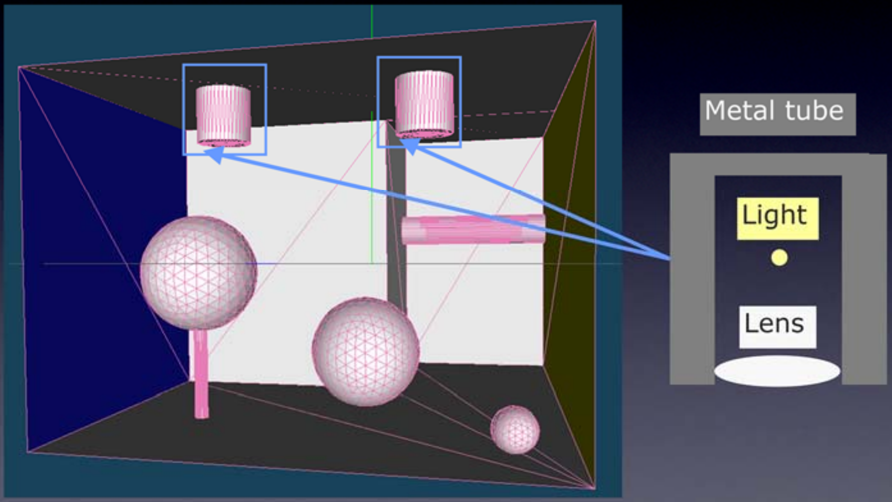
However, if we try to connect D with the light source by shooting a ray toward the light source, it could miss the light source because of the specular refraction.

Consistency of Photon Mapping

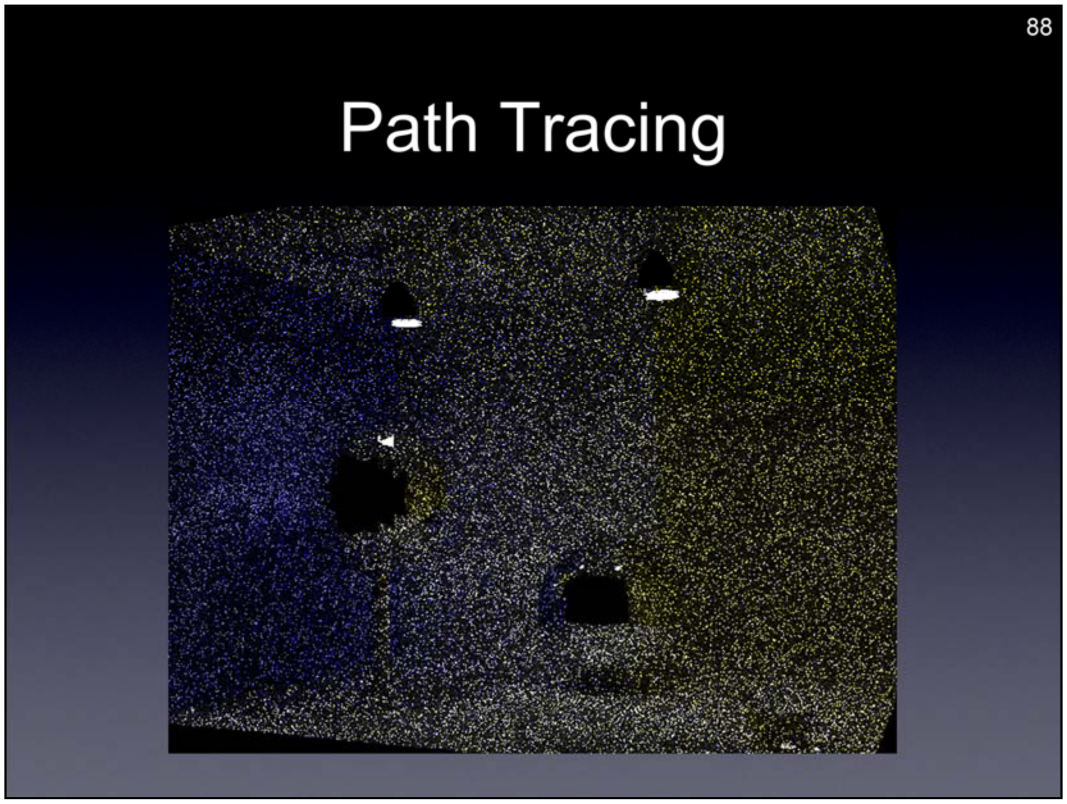




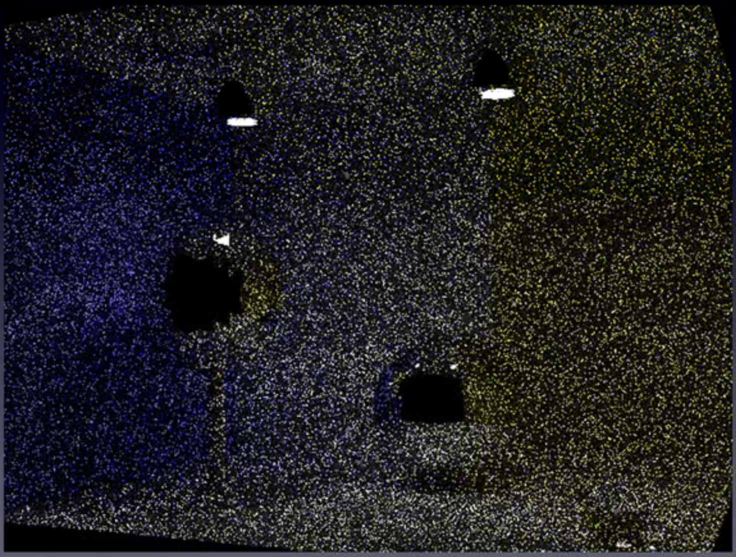
Realistic Light Source in Computer Graphics



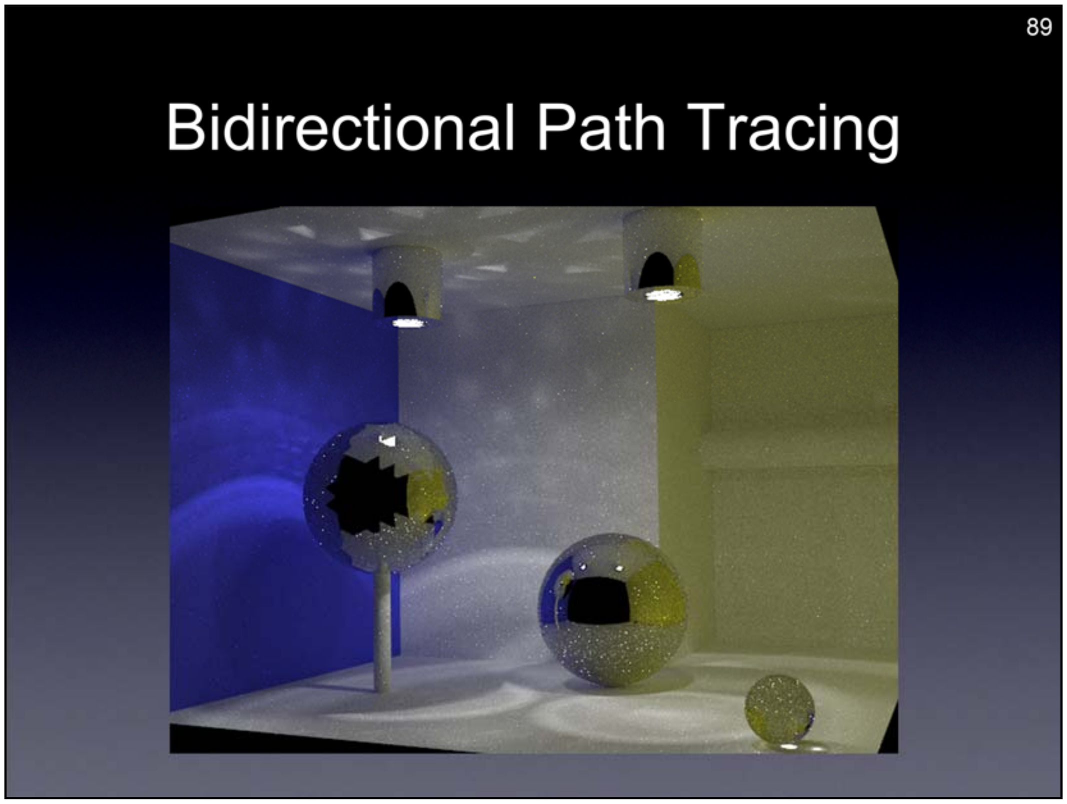
Let’s see if we can use this kind of realistic light sources in rendering. In this test scene, there is two small spherical light enclosed by a metal tube capped with lens which is similar to a real world light sourceI have shown before. Note that everything will be illuminated light coming through lens, which is caustic



Path Tracing

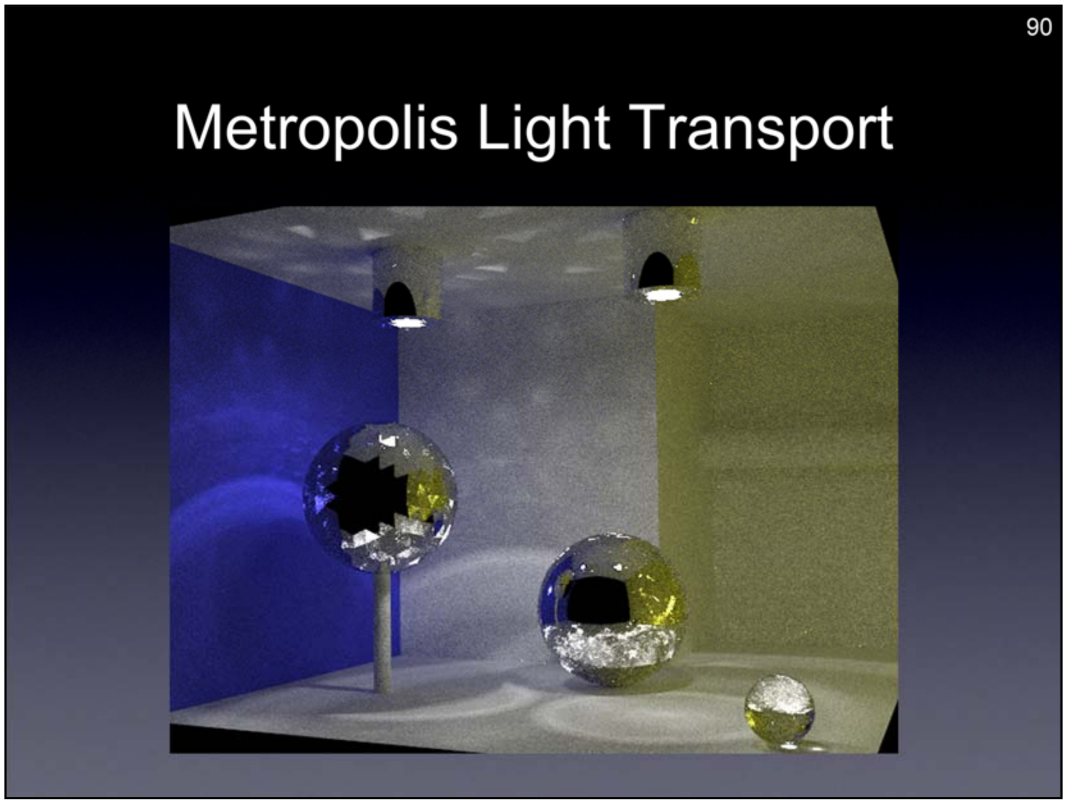


If you render this scene with path tracing, this is what you get. Image is very noisy because everything is illuminated by caustics and path tracing is not especially robust for rendering caustics.

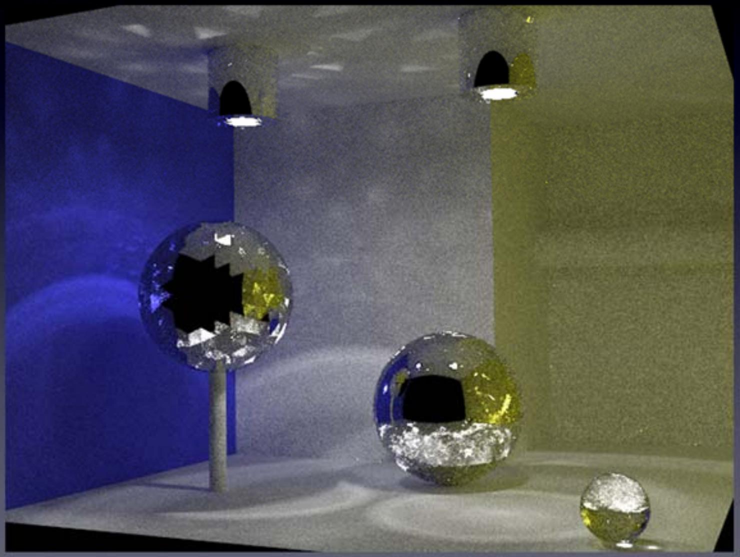


Bidirectional Path Tracing

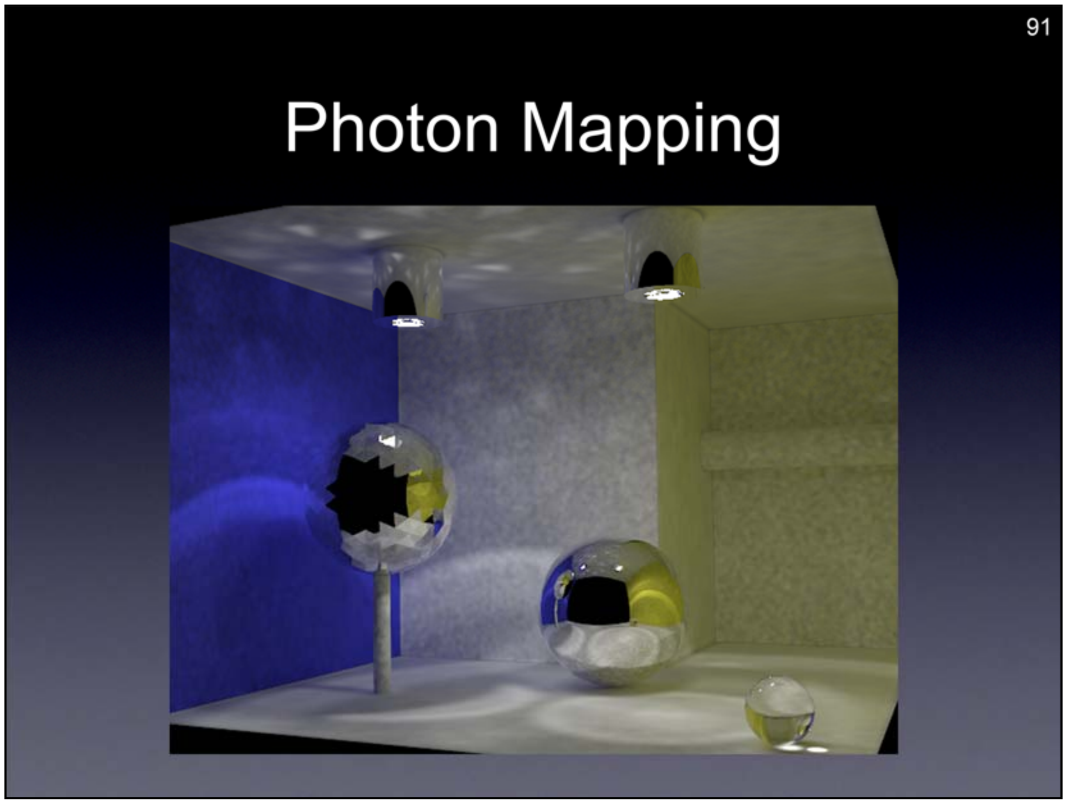
If you use bidirectional path tracing, it looks much better than path tracing. However, note that reflection on the mirror balls and refraction through the glass ball is very dark. This is because they are specular reflection or refraction of caustics, which is extremely difficult handle with any unbiased Monte Carlo ray tracing algorithm.



Metropolis Light Transport

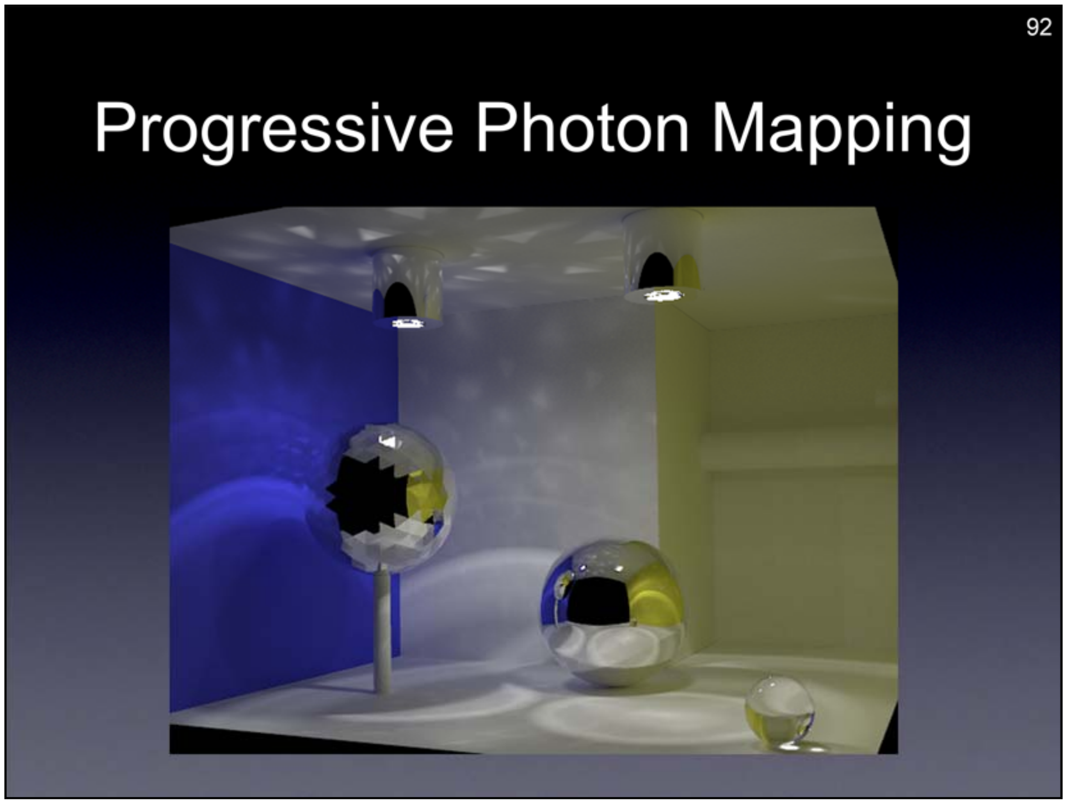


This is also the case even with Metropolis Light Transport which is considered to be the most robust algorithm to handle difficult lighting. Path in Metropolis Light Transport stacks reflection and refraction on the balls which caused bright spot noise.

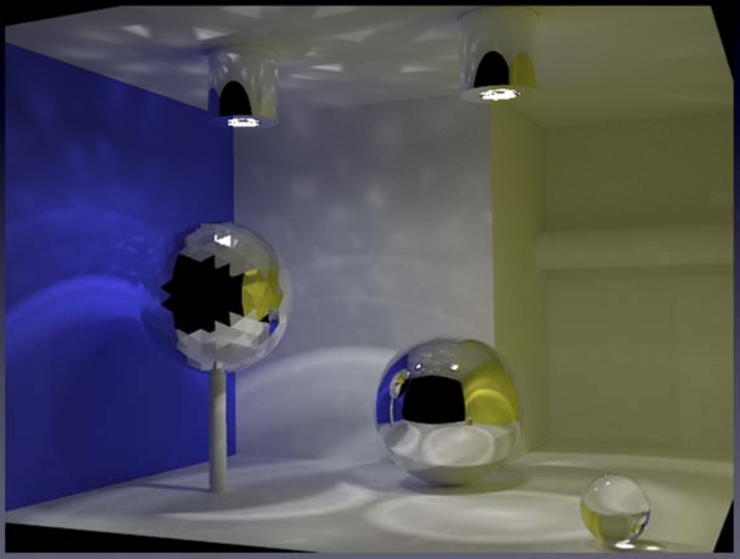


Photon Mapping

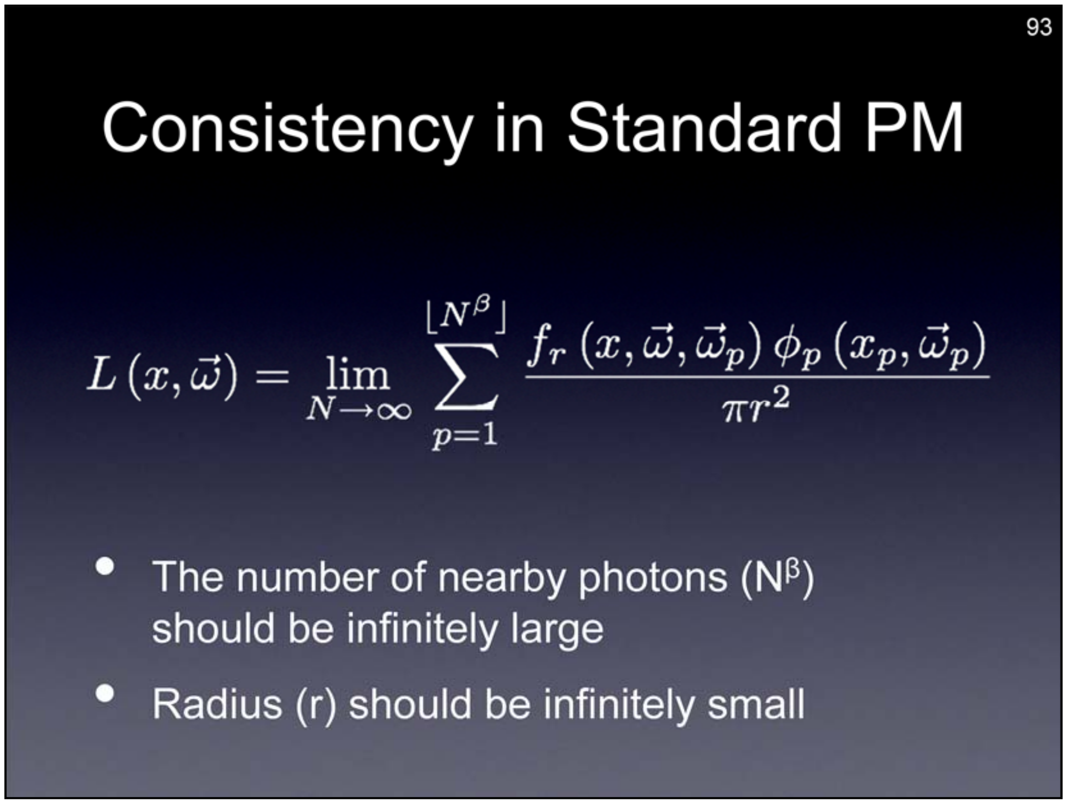
If we allow some bias in rendering, Photon Mapping can robustly render this type of illumination. However, Photon Mapping runs out all memory before obtaining noise-free results because we need to store all the photons. Note that we have to directly visualize photon map in this scene because everything is caustic.



Progressive Photon Mapping



If you use our method, you can finally render this simple, yet difficult scene accurately and robustly. I will describe how this can be done in this talk.

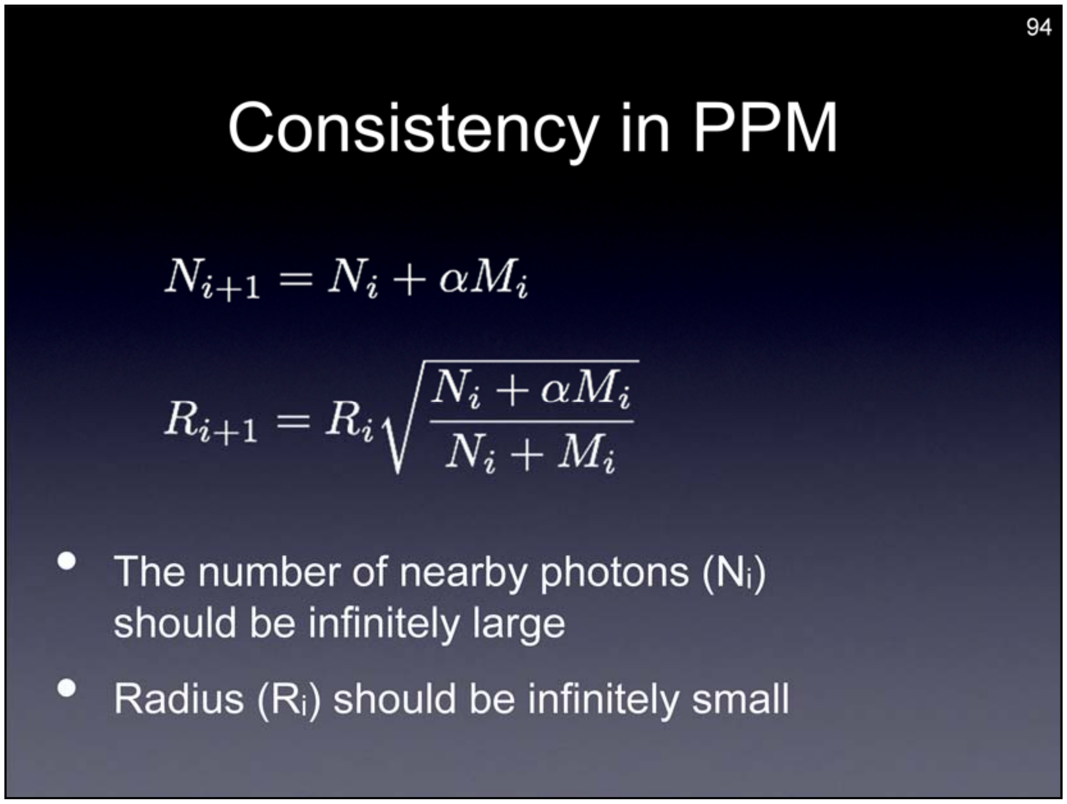


Consistency in Standard PM

$$L\left(x,\vec{\omega}\right) = \lim_{N\to\infty} \sum_{p=1}^{\lfloor N^{\beta} \rfloor} \frac{f_r\left(x,\vec{\omega},\vec{\omega}_p\right)\phi_p\left(x_p,\vec{\omega}_p\right)}{\pi r^2}$$

- The number of nearby photons (N^{β}) should be infinitely large
- Radius (r) should be infinitely small

Since flux and radius is available in every refinement, we can estimate radiance to show intermediate results to user. The equation to compute is exactly the same as standard photon mapping, except it uses accumulated, but unnormalized photon flux, which needs to be divided by the number of emitted photons so far over all refinement passes.



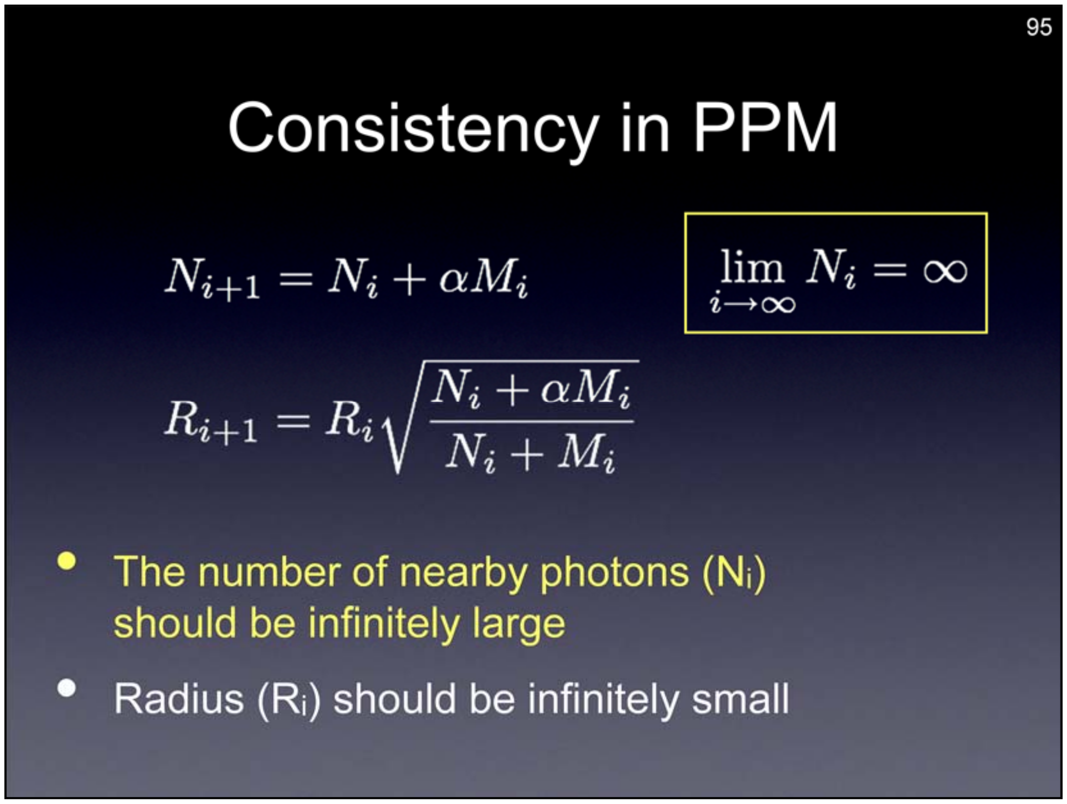
Consistency in PPM

$$N_{i+1} = N_i + \alpha M_i$$

$$R_{i+1} = R_i \sqrt{\frac{N_i + \alpha M_i}{N_i + M_i}}$$

- The number of nearby photons (N_i) should be infinitely large
- Radius (R_i) should be infinitely small

I will not describe the derivation in this talk, but it turns out that we just need to use these relatively simple equation to update statistics on each measurement point. Alpha here is the parameter chosen by user. All we need to do is to update the number of photons, radius and flux using these equations in each iteration.



Consistency in PPM

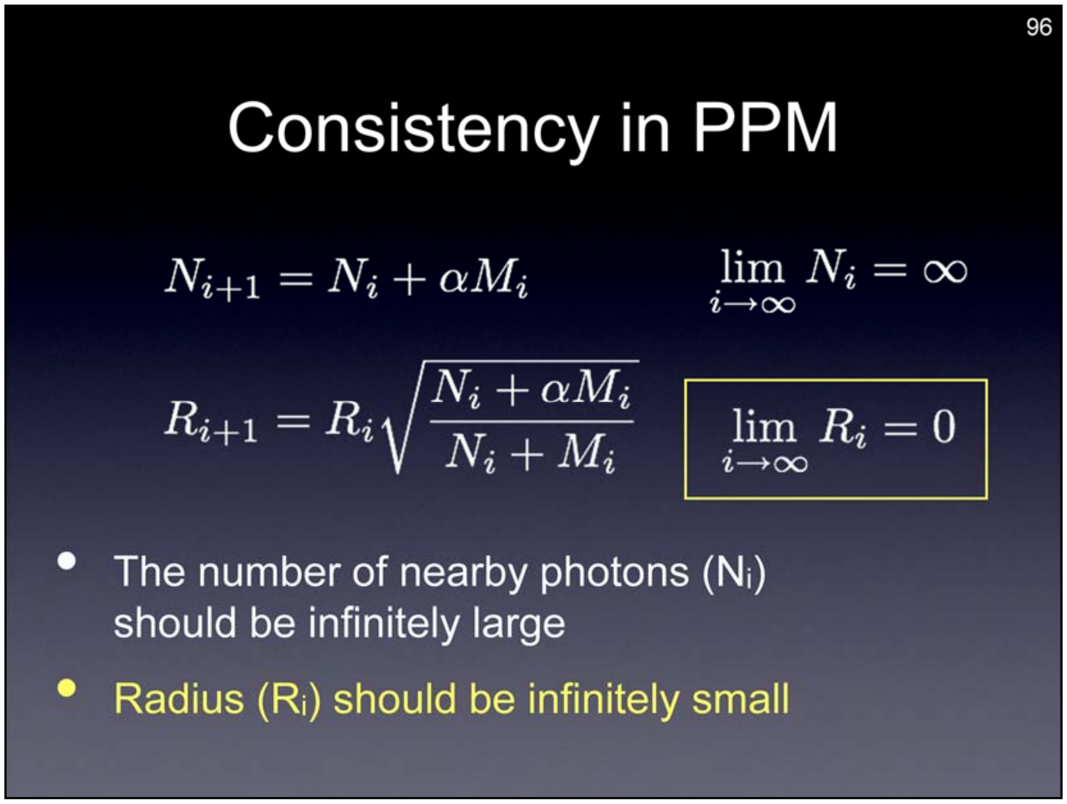
$$N_{i+1} = N_i + \alpha M_i$$

$$\boxed{\lim_{i \to \infty} N_i = \infty}$$

$$R_{i+1} = R_i \sqrt{\frac{N_i + \alpha M_i}{N_i + M_i}}$$

- The number of nearby photons (N_i) should be infinitely large
- Radius (R_i) should be infinitely small

I will not describe the derivation in this talk, but it turns out that we just need to use these relatively simple equation to update statistics on each measurement point. Alpha here is the parameter chosen by user. All we need to do is to update the number of photons, radius and flux using these equations in each iteration.



Consistency in PPM

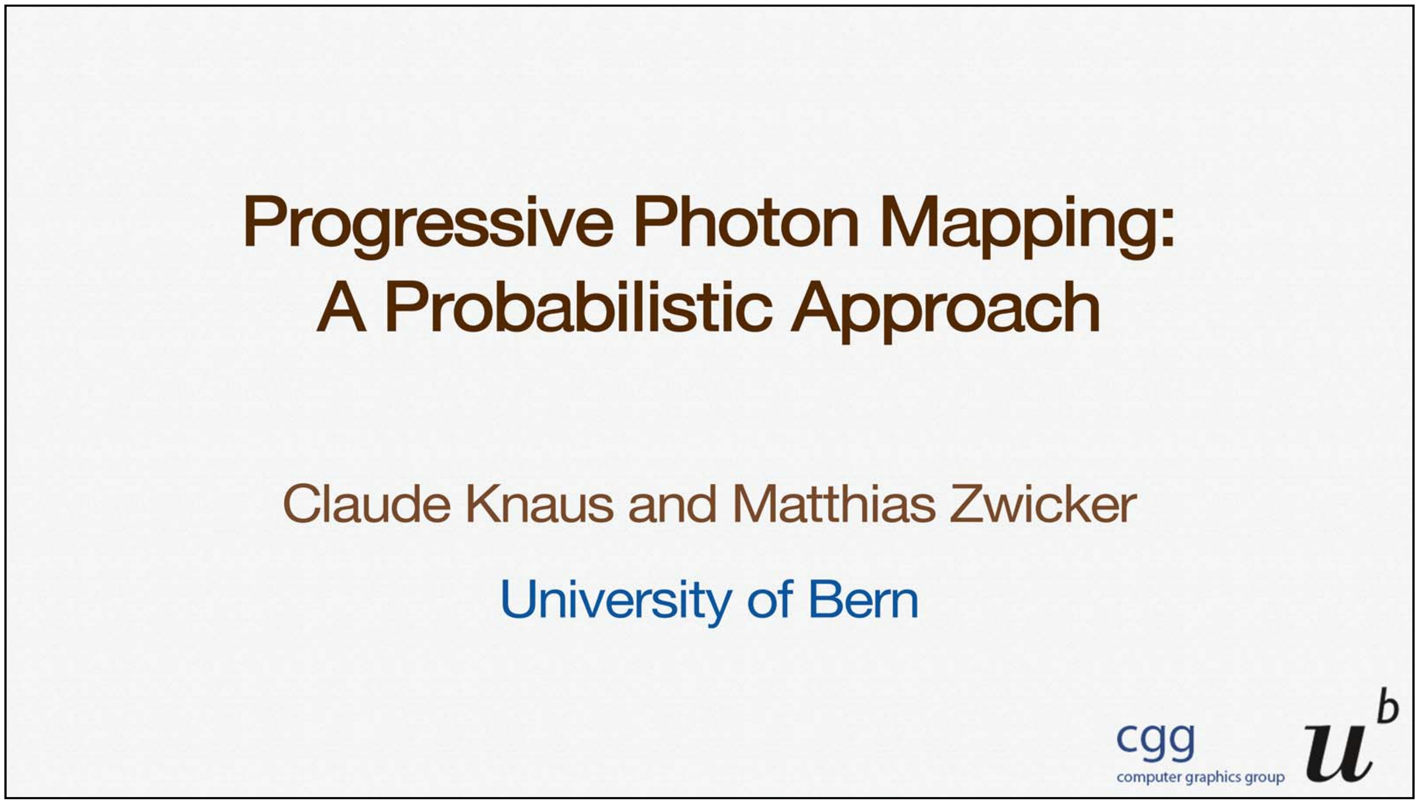
$$N_{i+1} = N_i + \alpha M_i \qquad \lim_{i \to \infty} N_i = \infty$$

$$R_{i+1} = R_i \sqrt{\frac{N_i + \alpha M_i}{N_i + M_i}} \qquad \boxed{\lim_{i \to \infty} R_i = 0}$$

- The number of nearby photons (N_i) should be infinitely large
- Radius (R_i) should be infinitely small

I will not describe the derivation in this talk, but it turns out that we just need to use these relatively simple equation to update statistics on each measurement point. Alpha here is the parameter chosen by user. All we need to do is to update the number of photons, radius and flux using these equations in each iteration.

SIGGRAPH2011



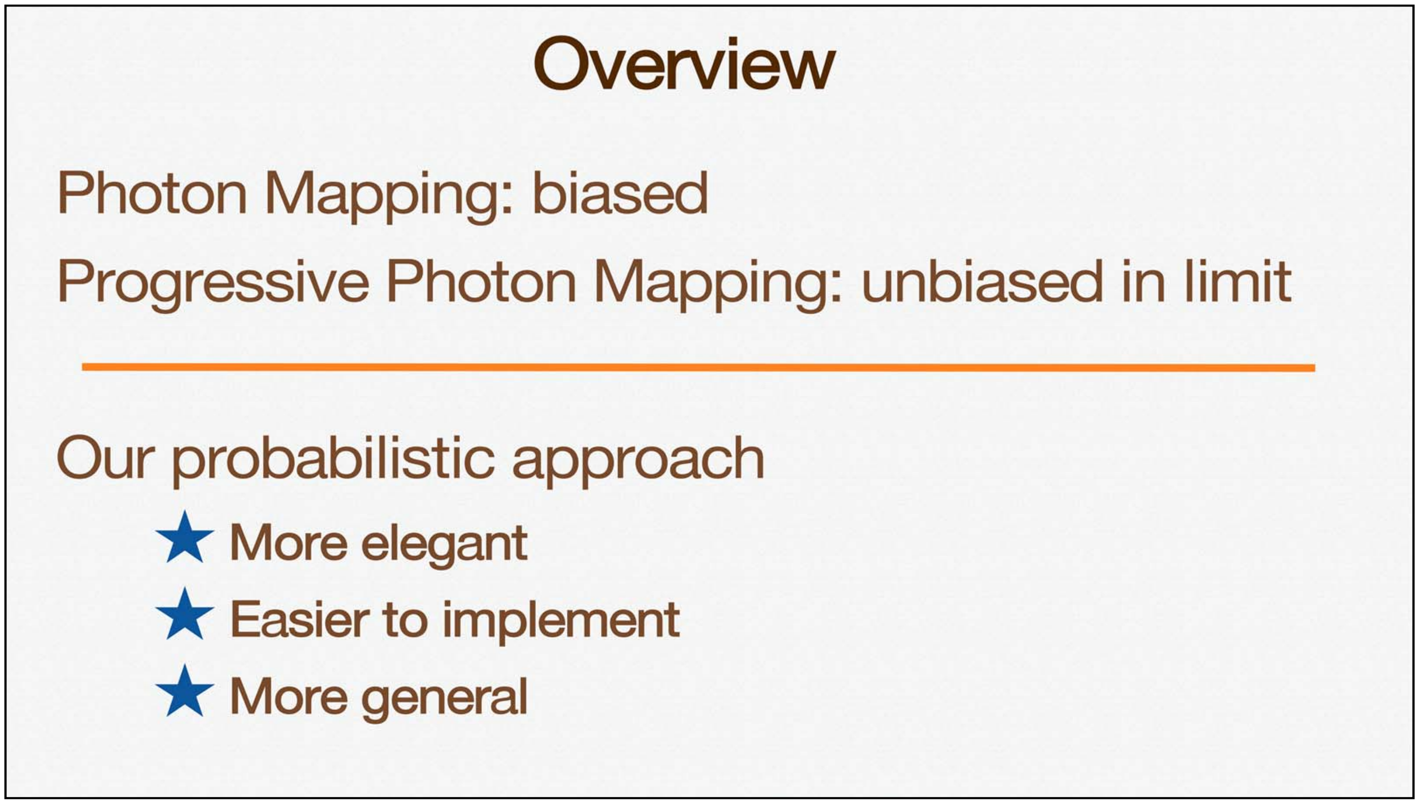
Progressive Photon Mapping: A Probabilistic Approach

Claude Knaus and Matthias Zwicker

University of Bern

cgg
computer graphics group

Thank you for the introduction.



Overview

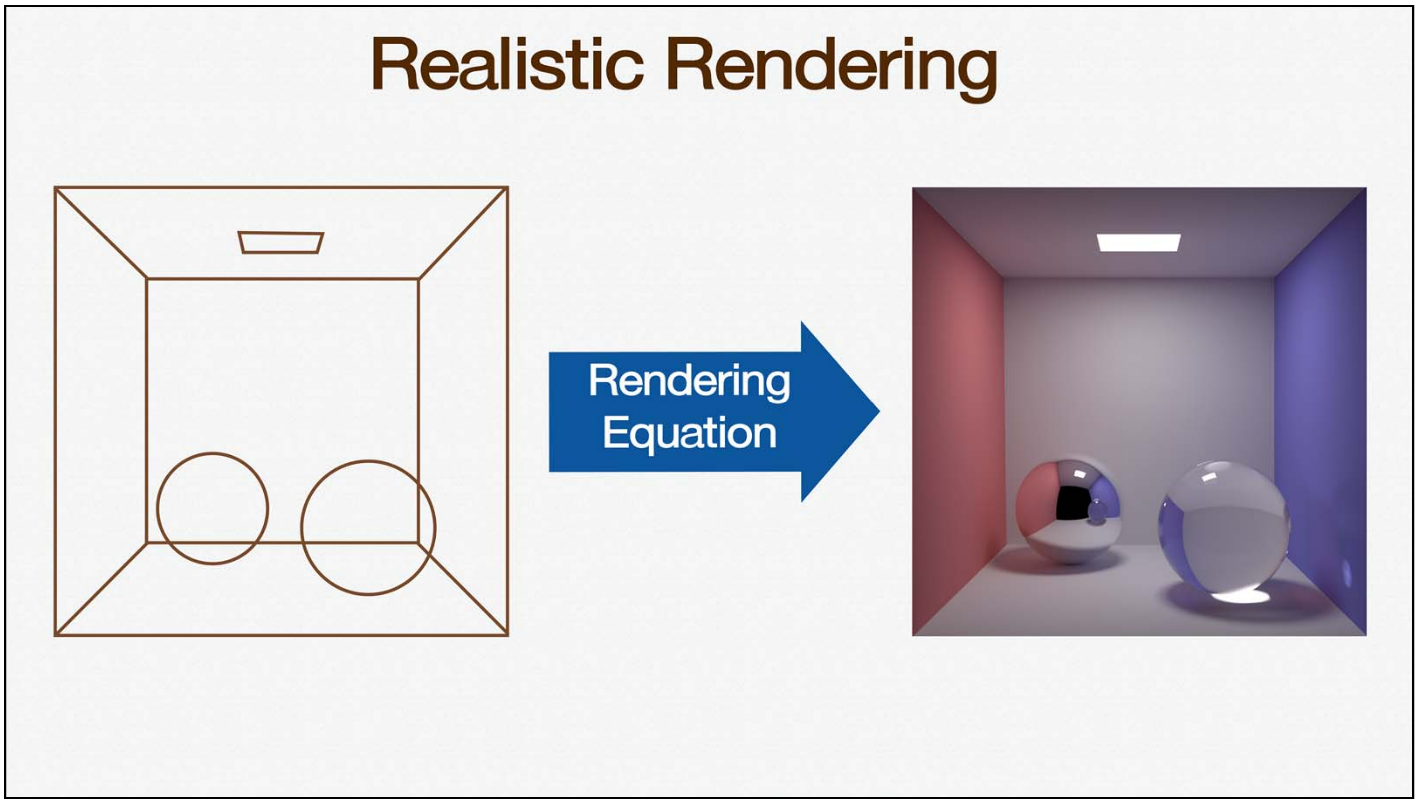
Photon Mapping: biased

Progressive Photon Mapping: unbiased in limit

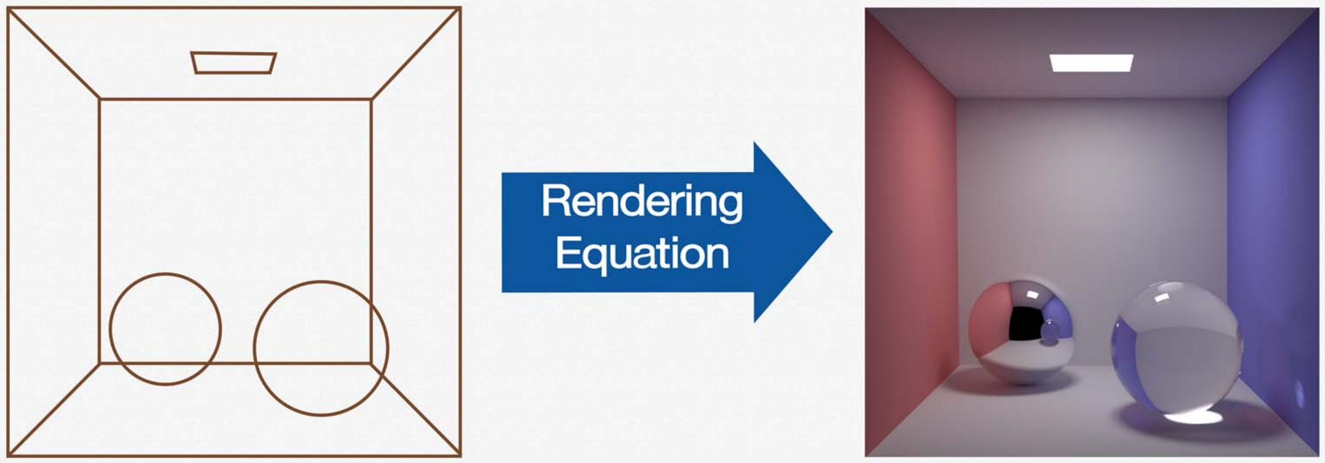
Our probabilistic approach

- ★ More elegant
- ★ Easier to implement
- ★ More general

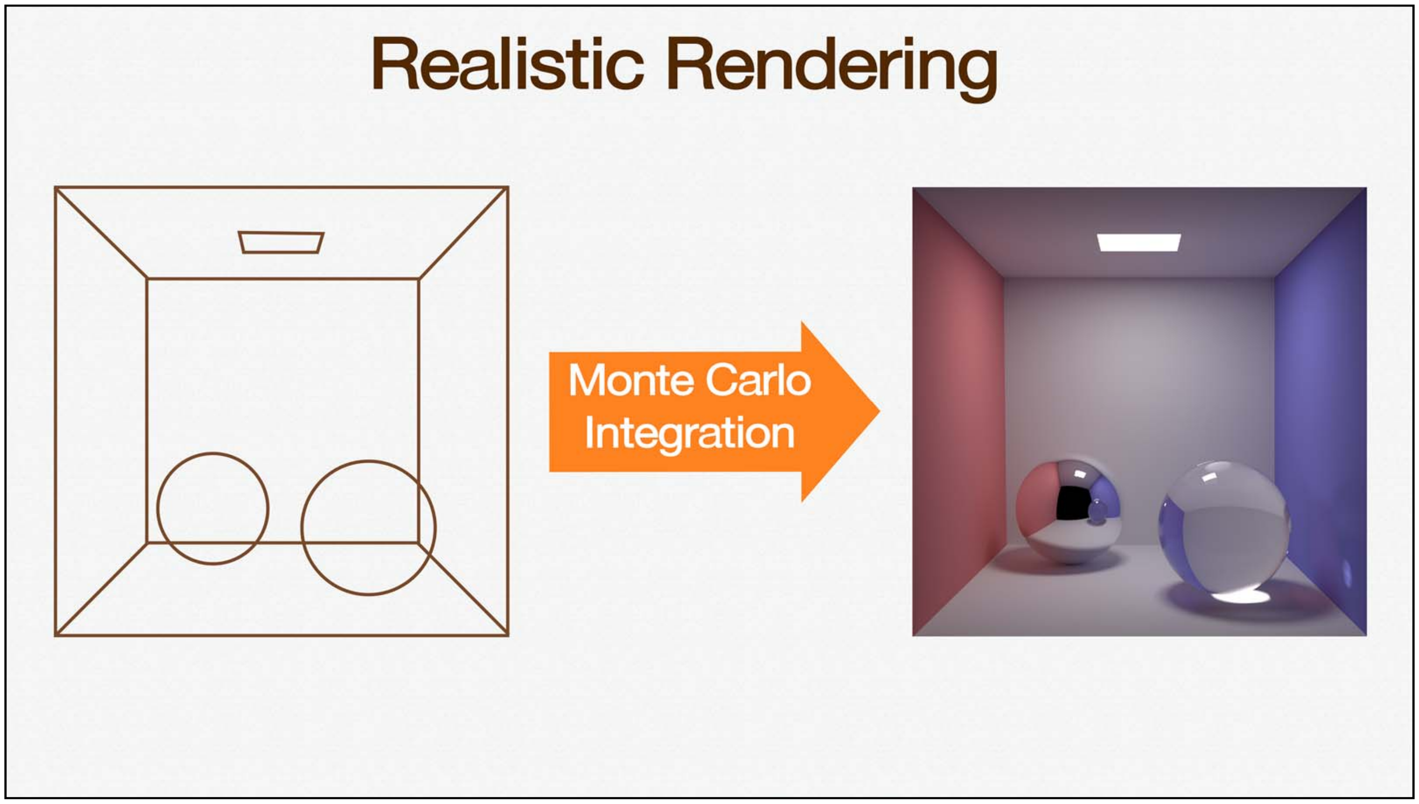
This talk consists of two parts. In the first part, we will review photon mapping and its problem of being biased, which means that there is a consistent error in the rendered image. This problem has been solved recently by progressive photon mapping, which is unbiased in the limit. In the second part, we propose an alternative, a probabilistic approach to progressive photon mapping, which is more elegant, easier to implement, and also more general.



Realistic Rendering

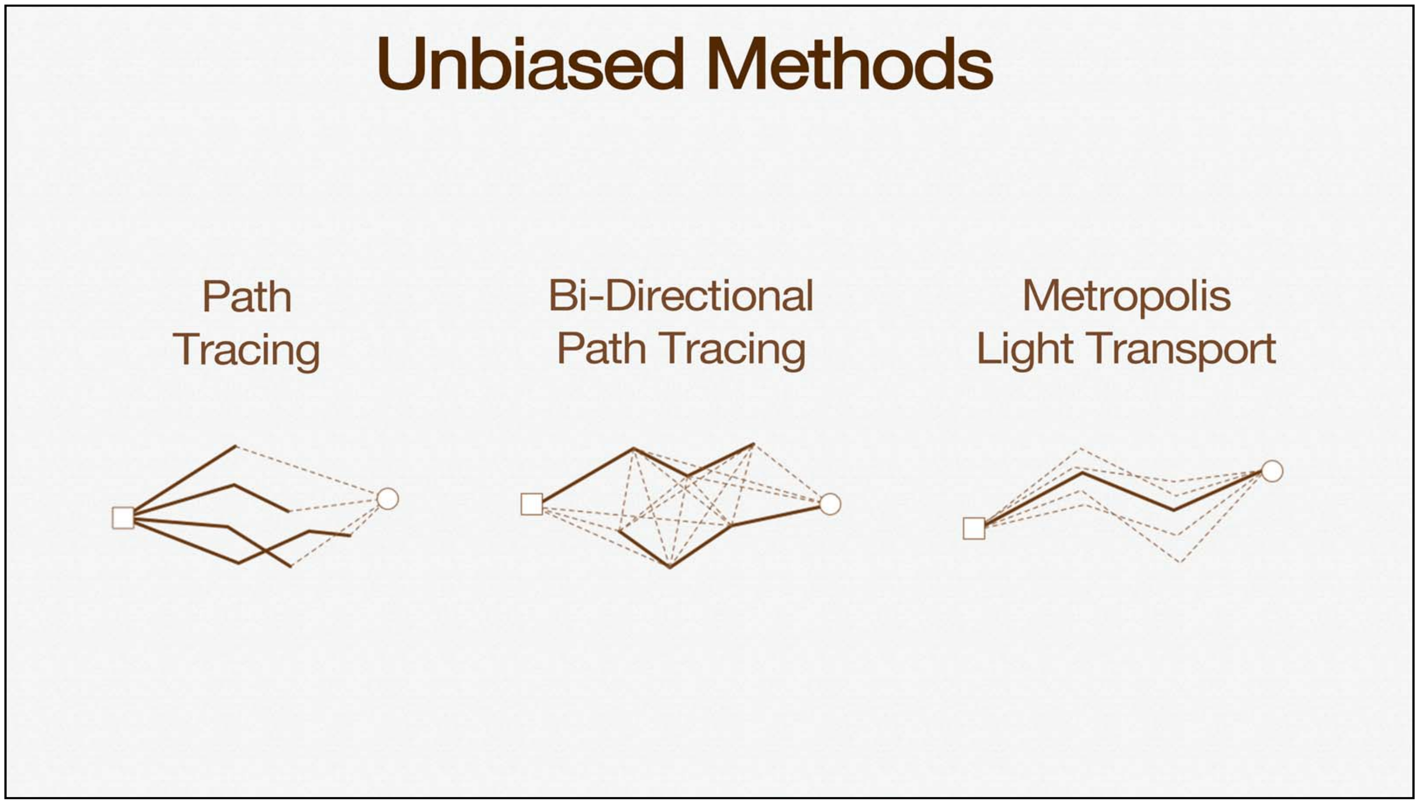


Realistic rendering is based on Kajiya’s rendering equation. The rendering equation involves an integral, which is usually numerically solved using Monte Carlo integration.



Realistic Rendering

Methods using Monte Carlo integration sum over randomly sampled light paths. The most commonly used methods exploiting Monte Carlo integration to solve the rendering equation are



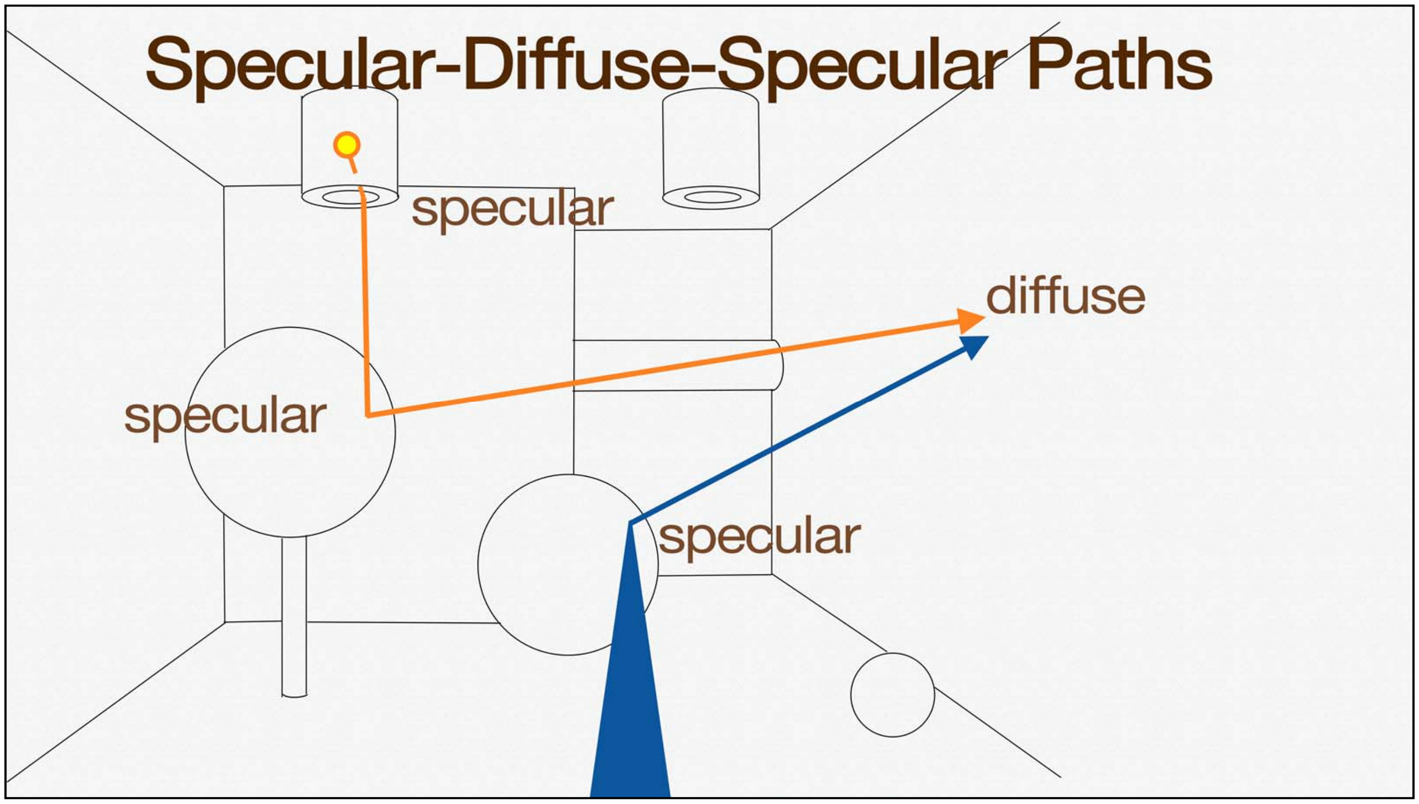
Unbiased Methods

Path Tracing

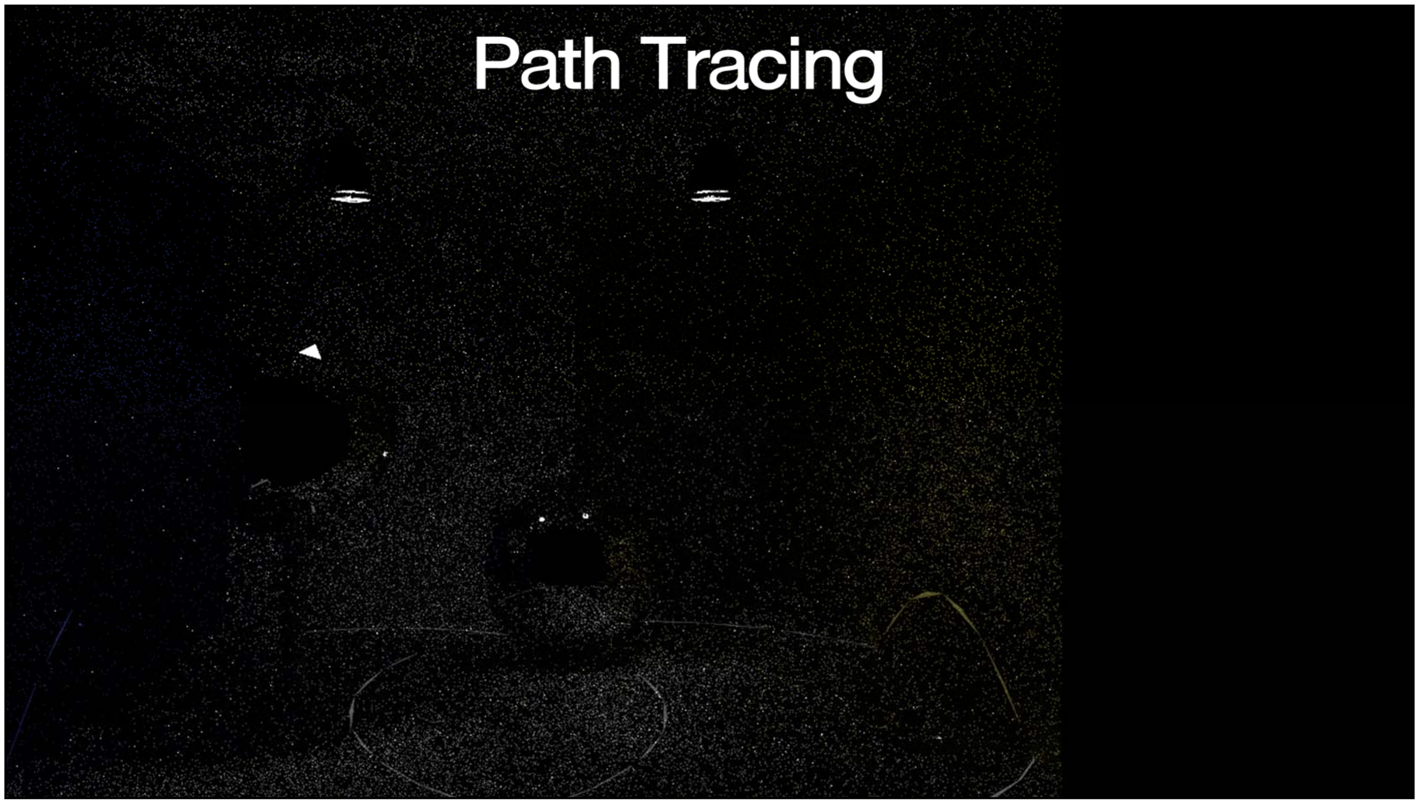
Bi-Directional Path Tracing

Metropolis Light Transport

path tracing, bi-directional path tracing, and metropolis light transport. These are unbiased methods.

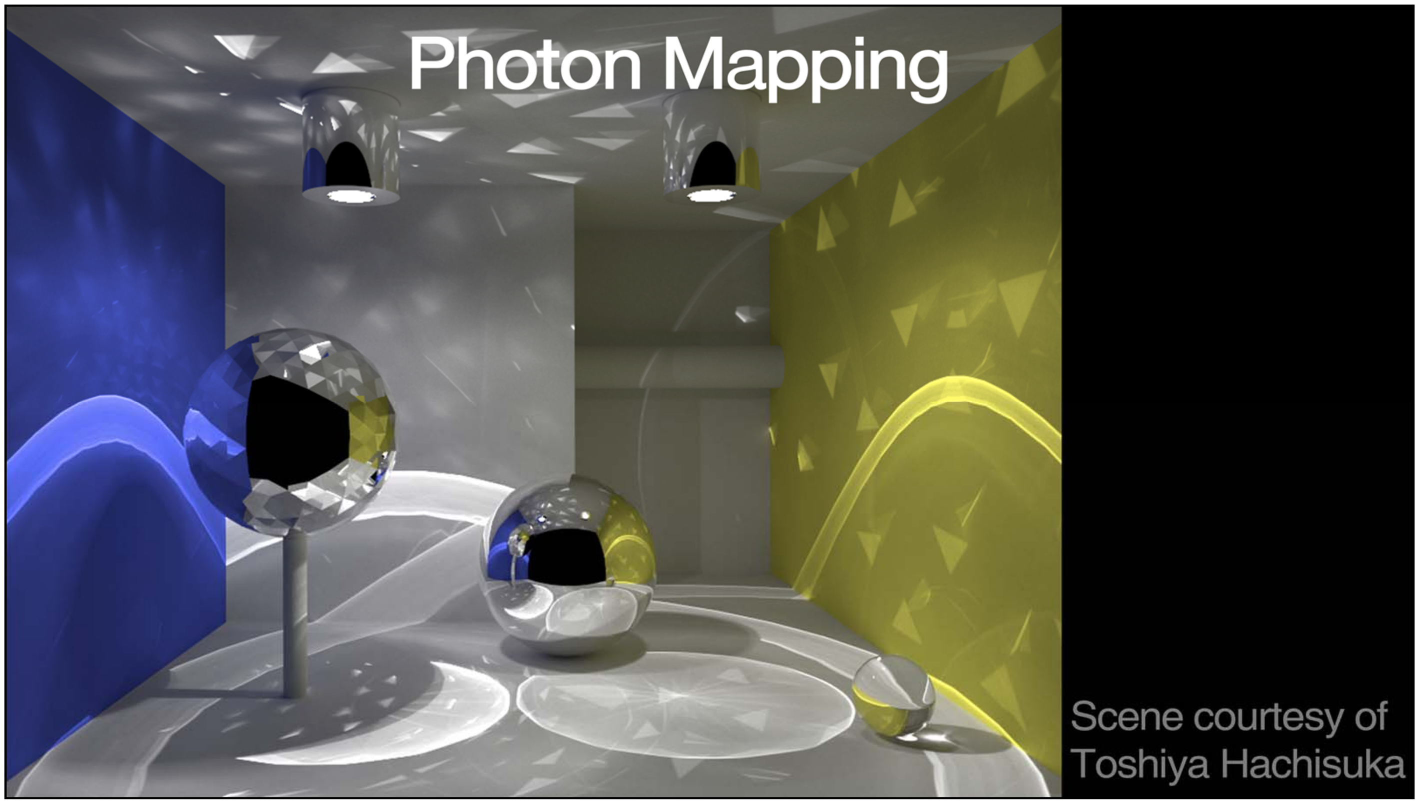


Unbiased methods have difficulties with rendering scenes which include so-called specular-diffuse-specular light paths. Such scenes are typical for realistic light settings, where the light source is not directly visible, but covered behind a refracting surface like a lens or reflected by a mirror.

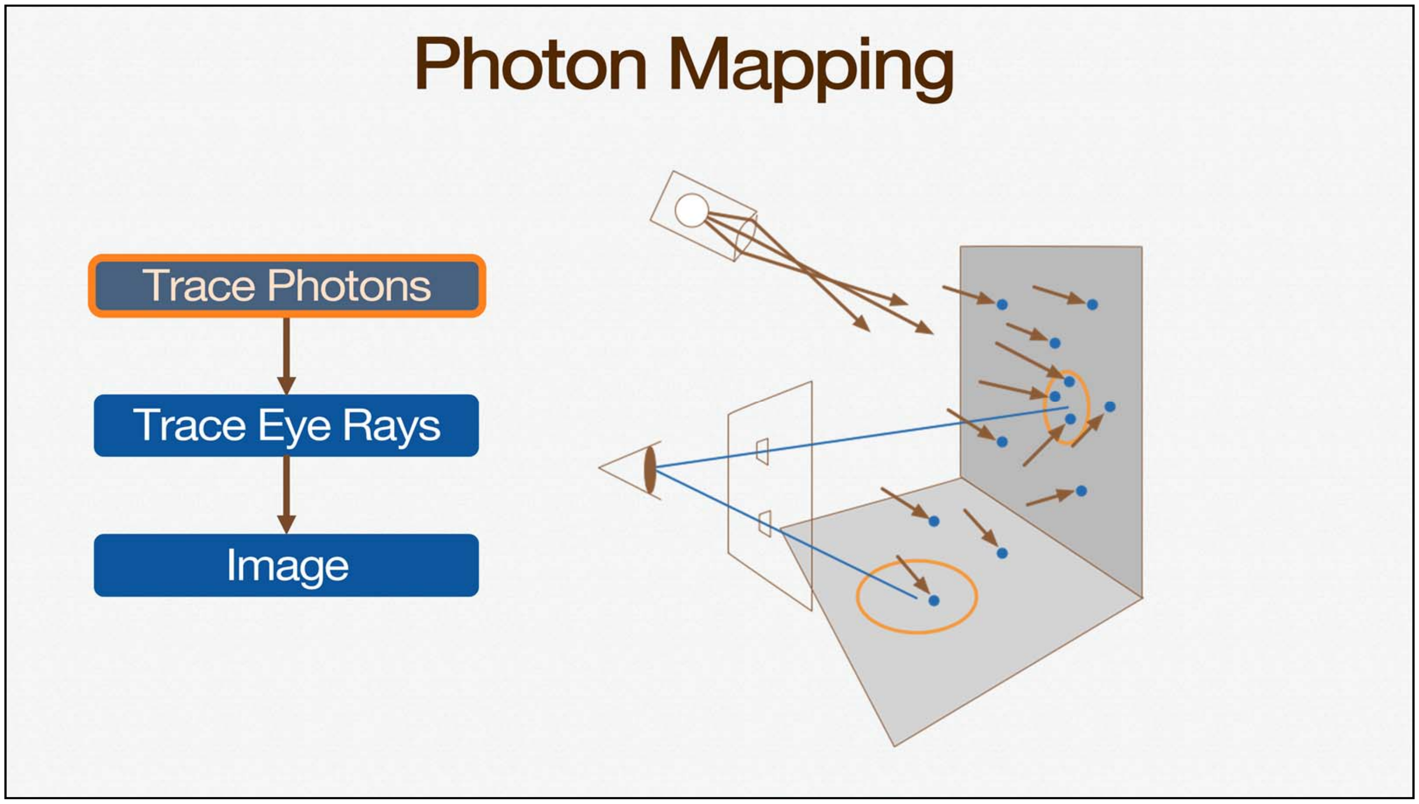


Path Tracing

If we were to render such a scene using path tracing, we would see almost nothing. This is because the specular-diffuse-specular paths are sampled with probability close to 0.



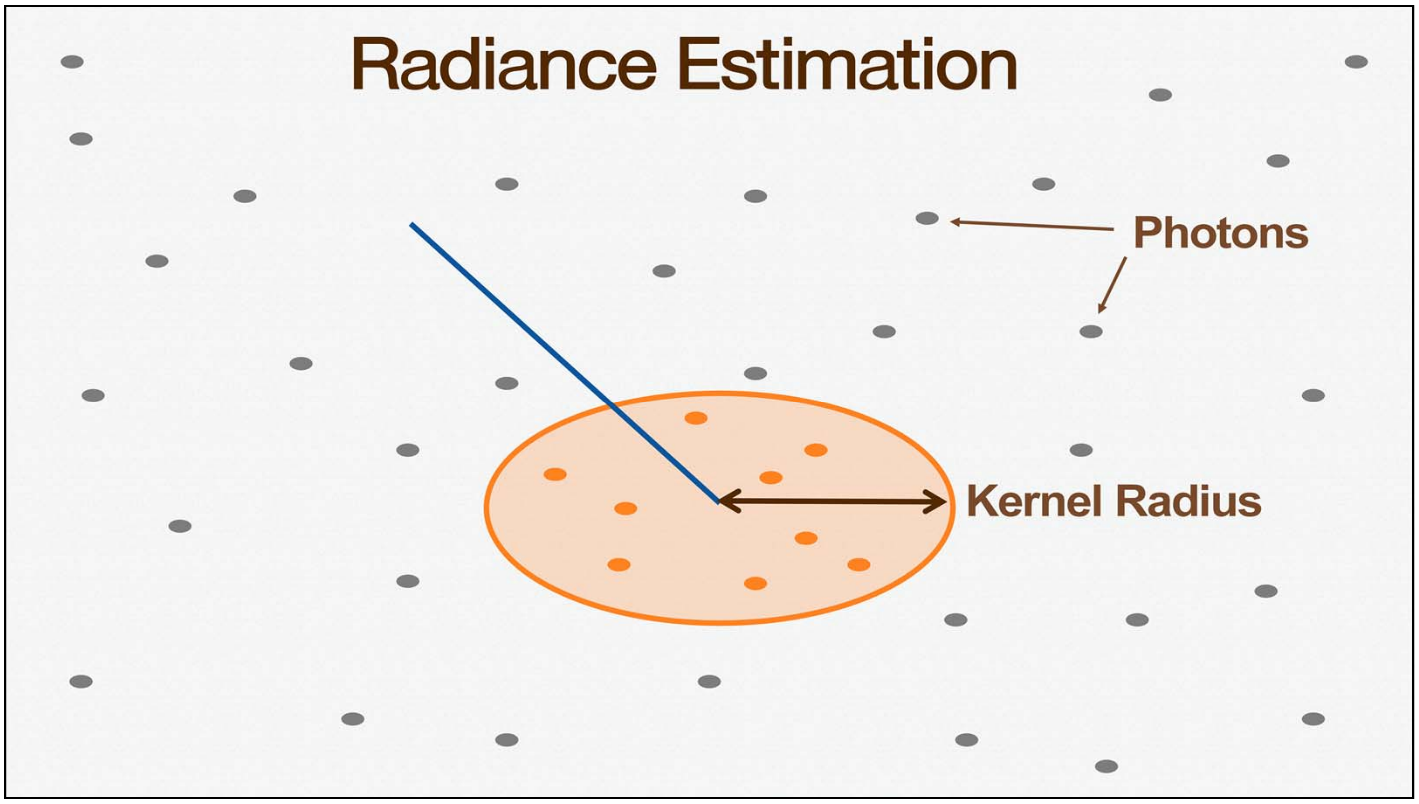
This is the same scene rendered using photon mapping. Arguably, there is much less noise.



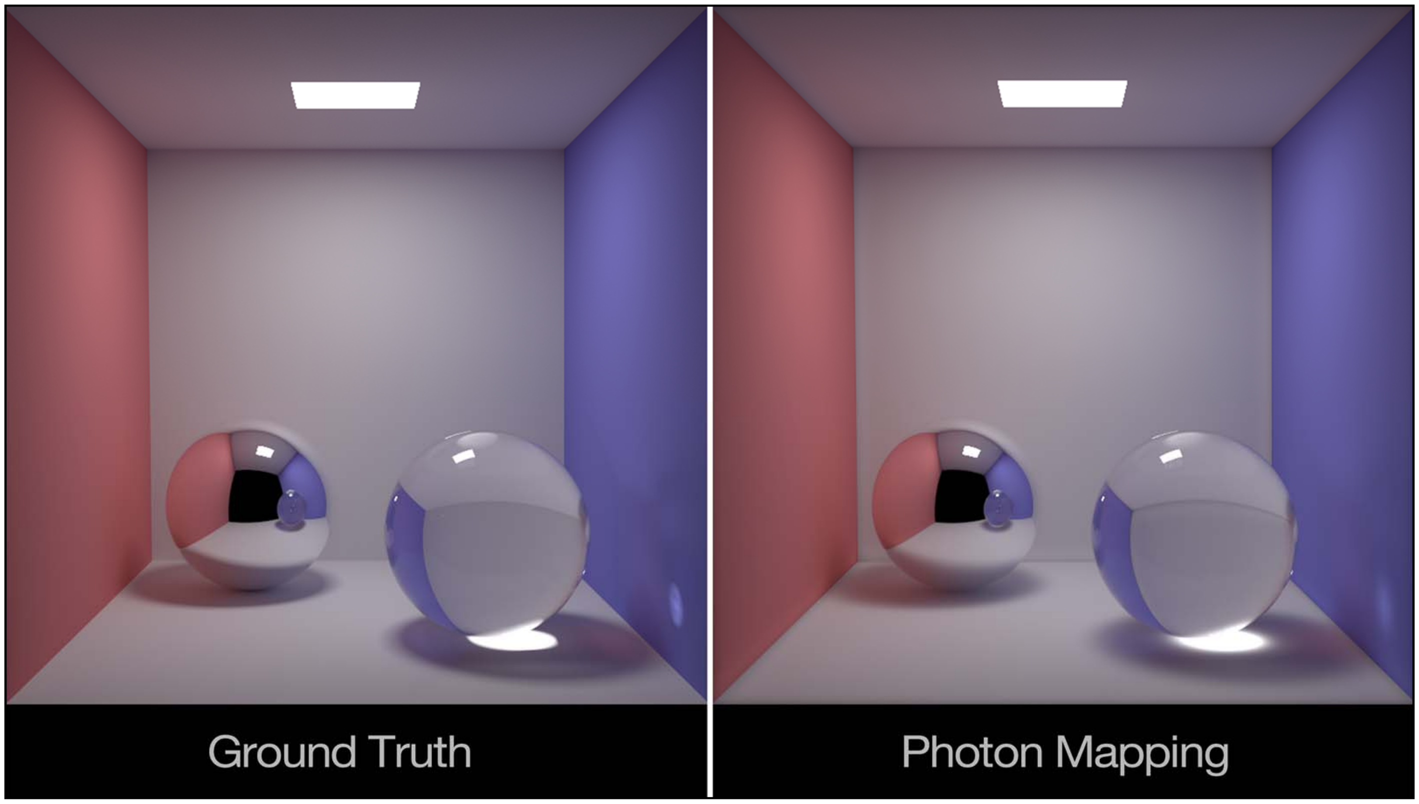
Photon Mapping

- Trace Photons
- Trace Eye Rays
- Image

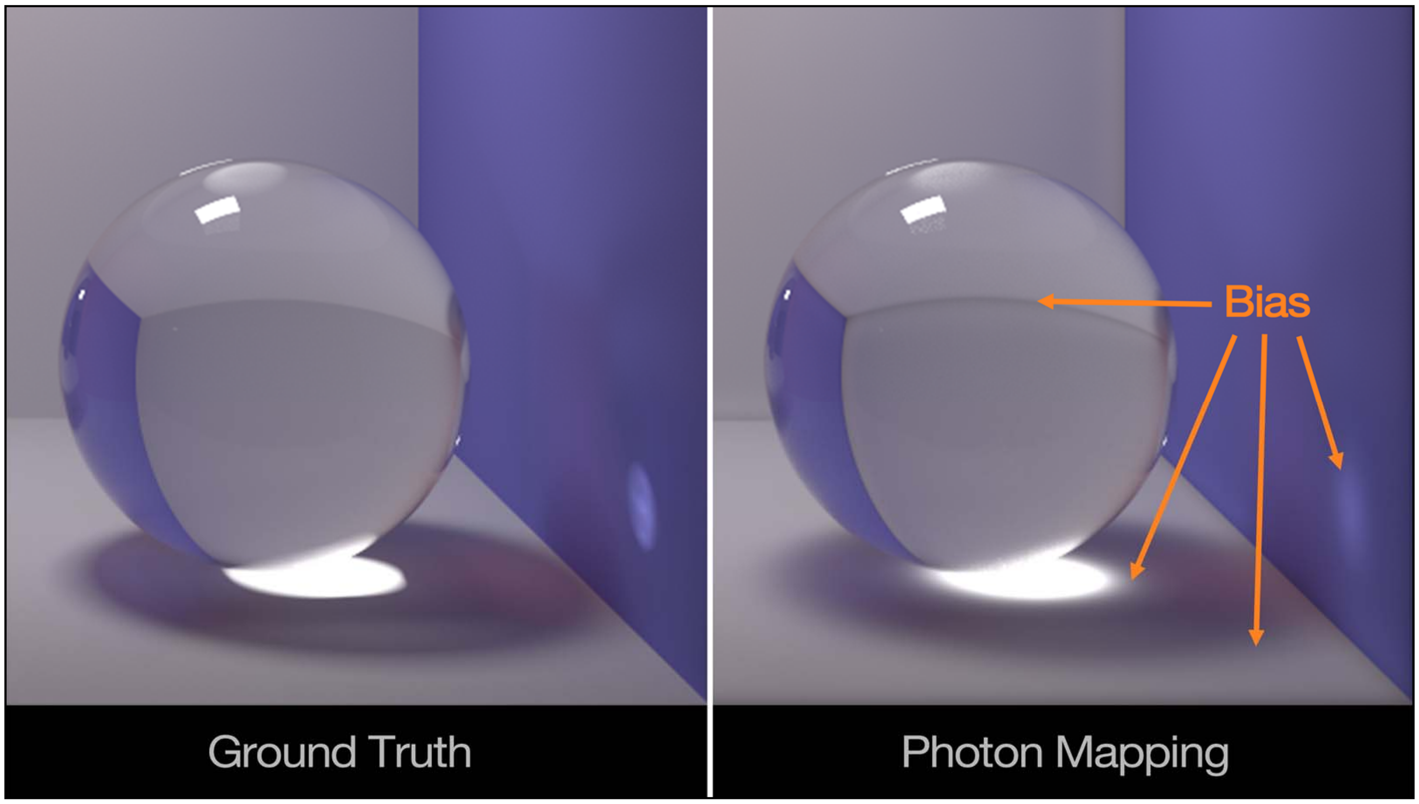
What makes photon mapping so good in rendering such scenes? Photon Mapping is a biased method. While unbiased methods sample entire paths from sensor to light, photon mapping samples partial paths. In a first step, it samples paths emitted from the light source and caches them as photons in a spatial data structure, the photon map. In the second step it samples paths from the eye and connects them to the previously cached light paths. It is this caching and reusing of light paths which makes photon mapping efficient.



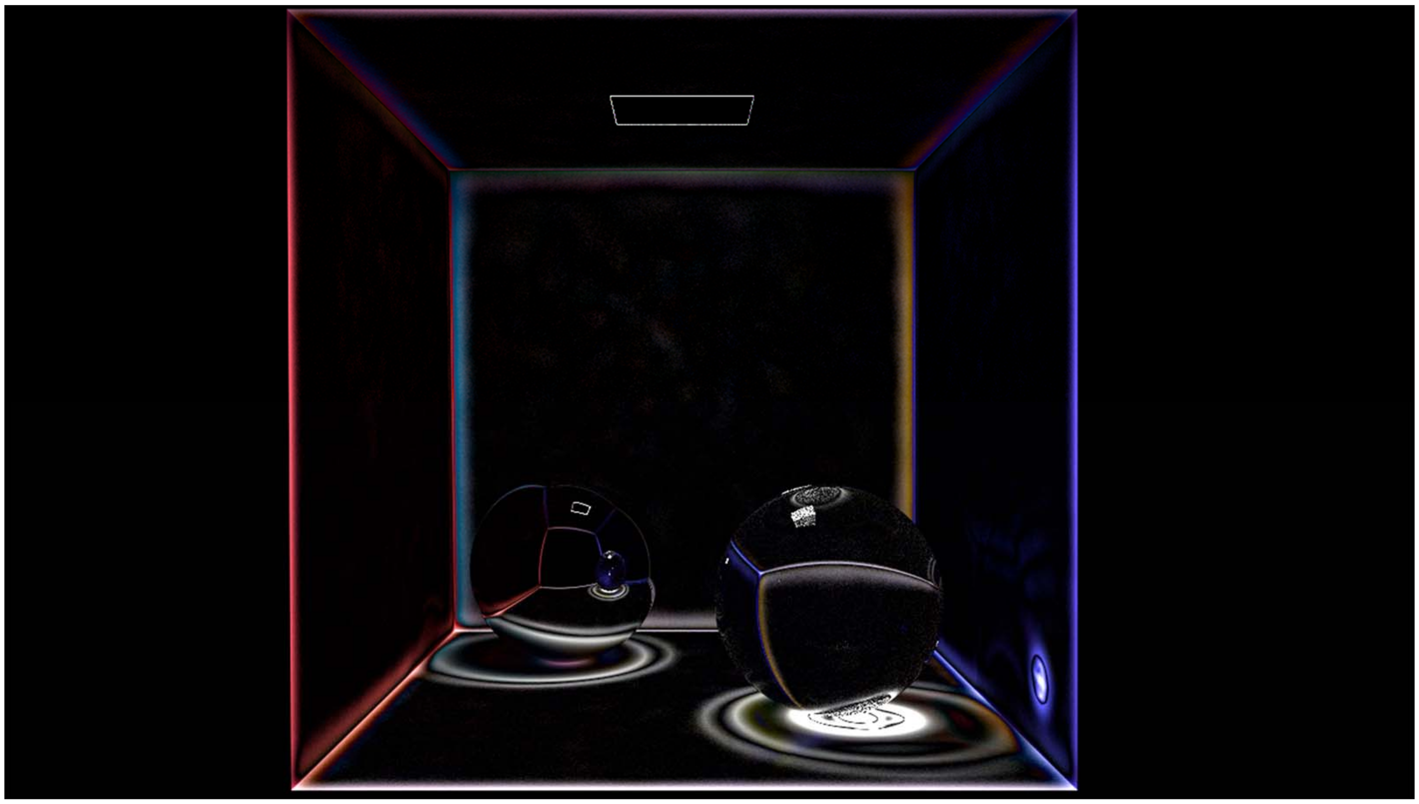
The partial paths are connected by so called radiance estimation. It involves the use of a kernel, which has a certain radius.



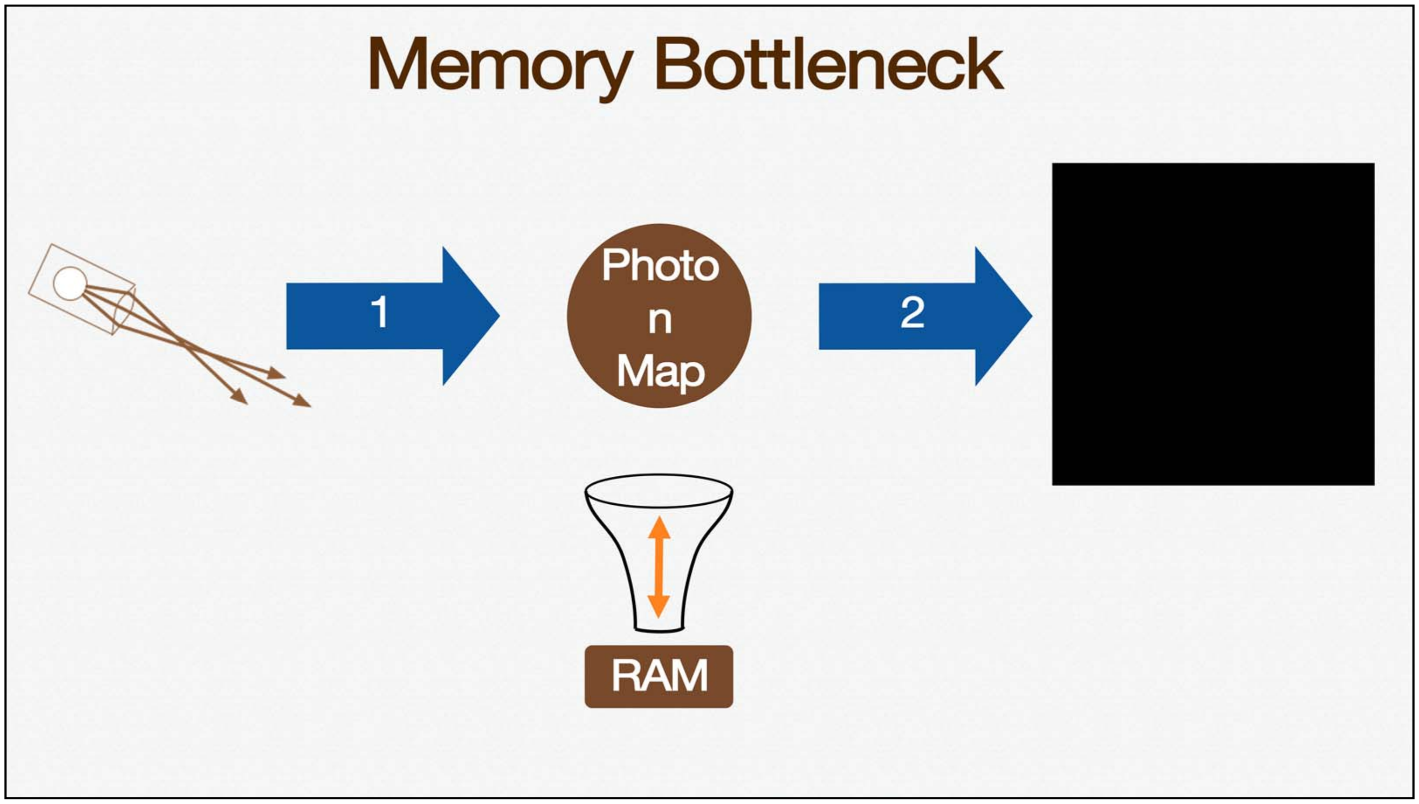
It is this radius which is the source of to the biggest critique of photon mapping, that it is biased.
Radiance estimation involves a convolution or interpolation, which often introduces blurriness, and therefore there is a consistent error, which is called bias.



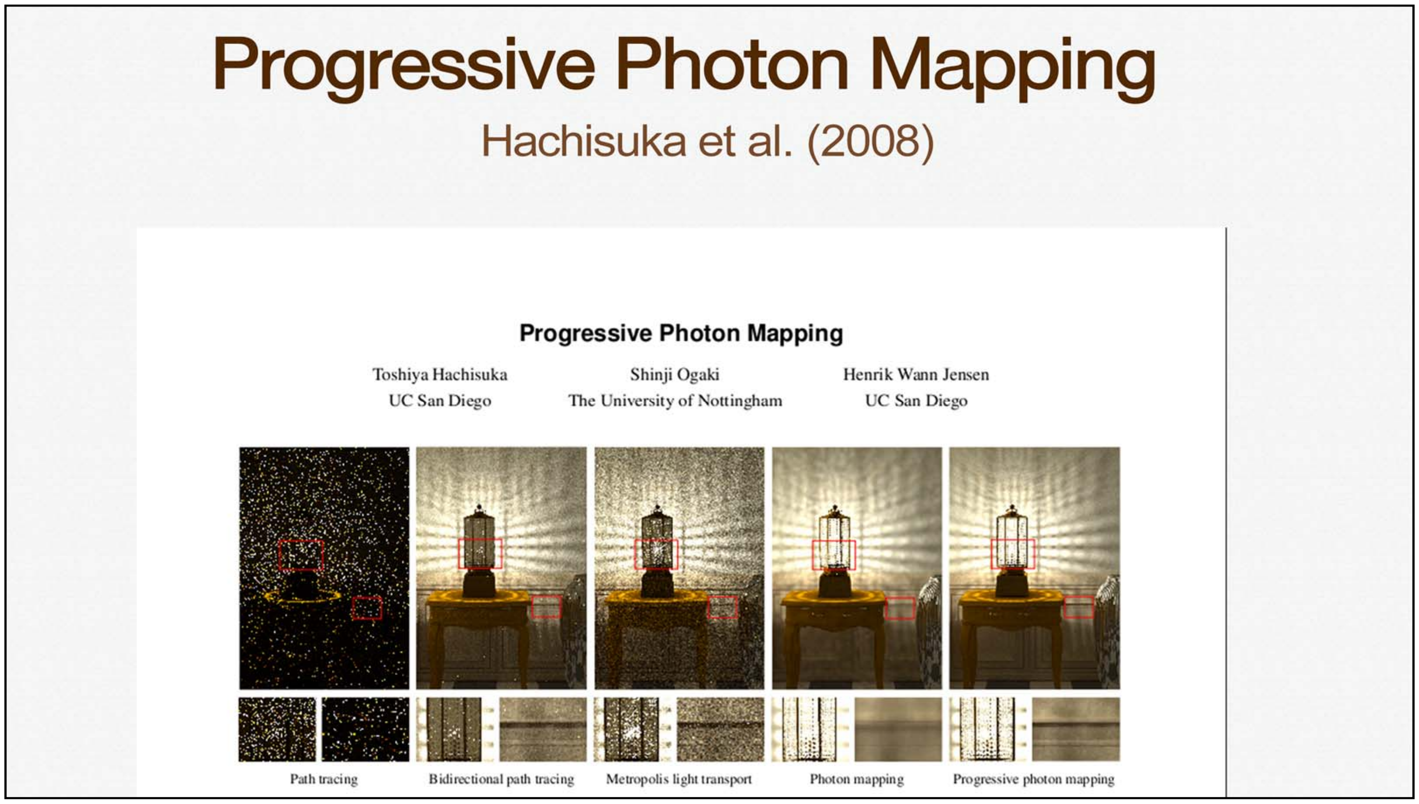
It is this radius which is the source of to the biggest critique of photon mapping, that it is biased.
Radiance estimation involves a convolution or interpolation, which introduces blurriness, and therefore there is a consistent error, which is called bias.



Here is another image of the bias, shown as the difference between the two previous images.


Memory Bottleneck

Although photon mapping is biased, it is still a consistent method, which means that with an infinite number of samples, the bias vanishes and the resulting image is the correct. But here lies the practical problem of photon mapping: since the photons representing the light paths must be cached between the two stages, the quality of the image is limited by the available memory.



Progressive Photon Mapping

Hachisuka et al. (2008)

Progressive Photon Mapping

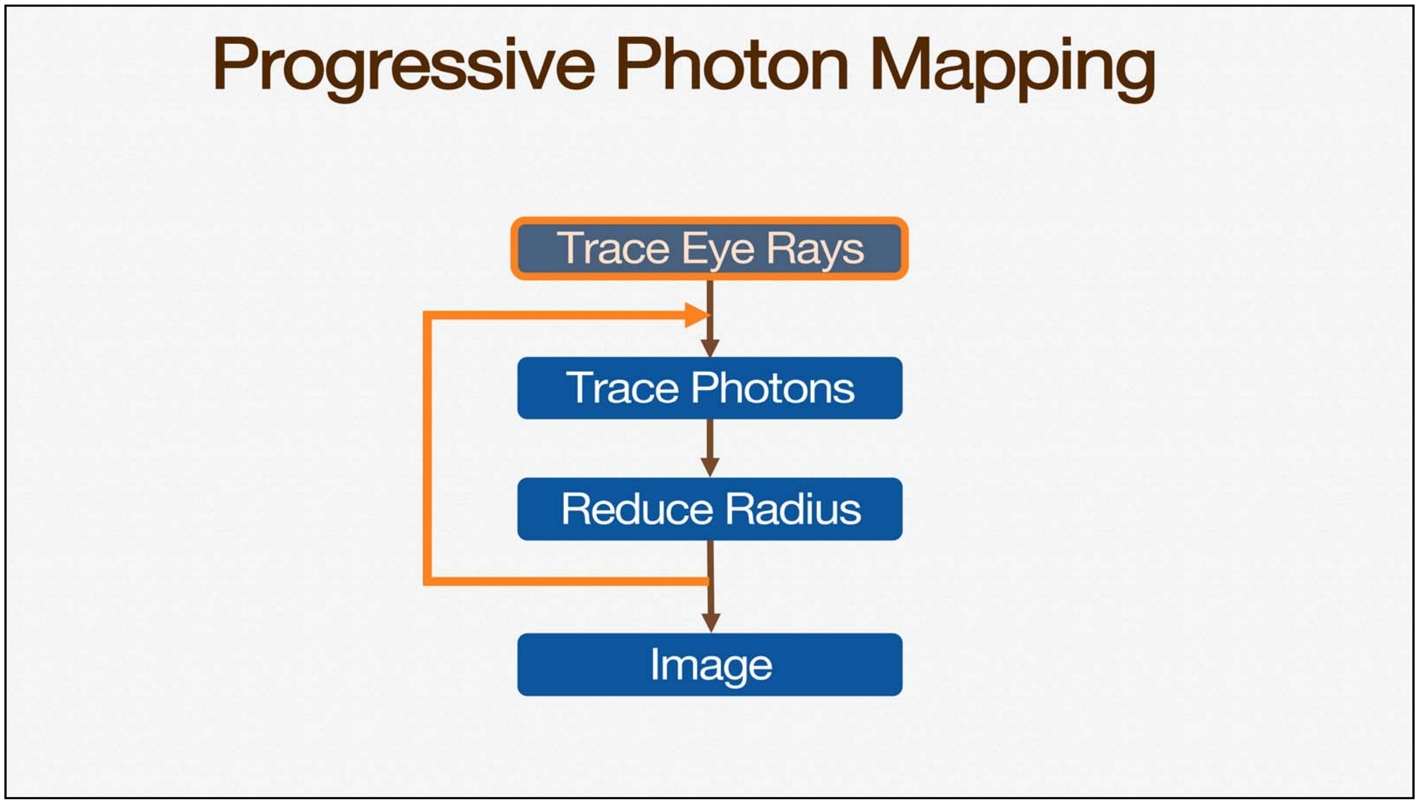
Toshiya Hachisuka
UC San Diego

Shinji Ogaki
The University of Nottingham

Henrik Wann Jensen
UC San Diego

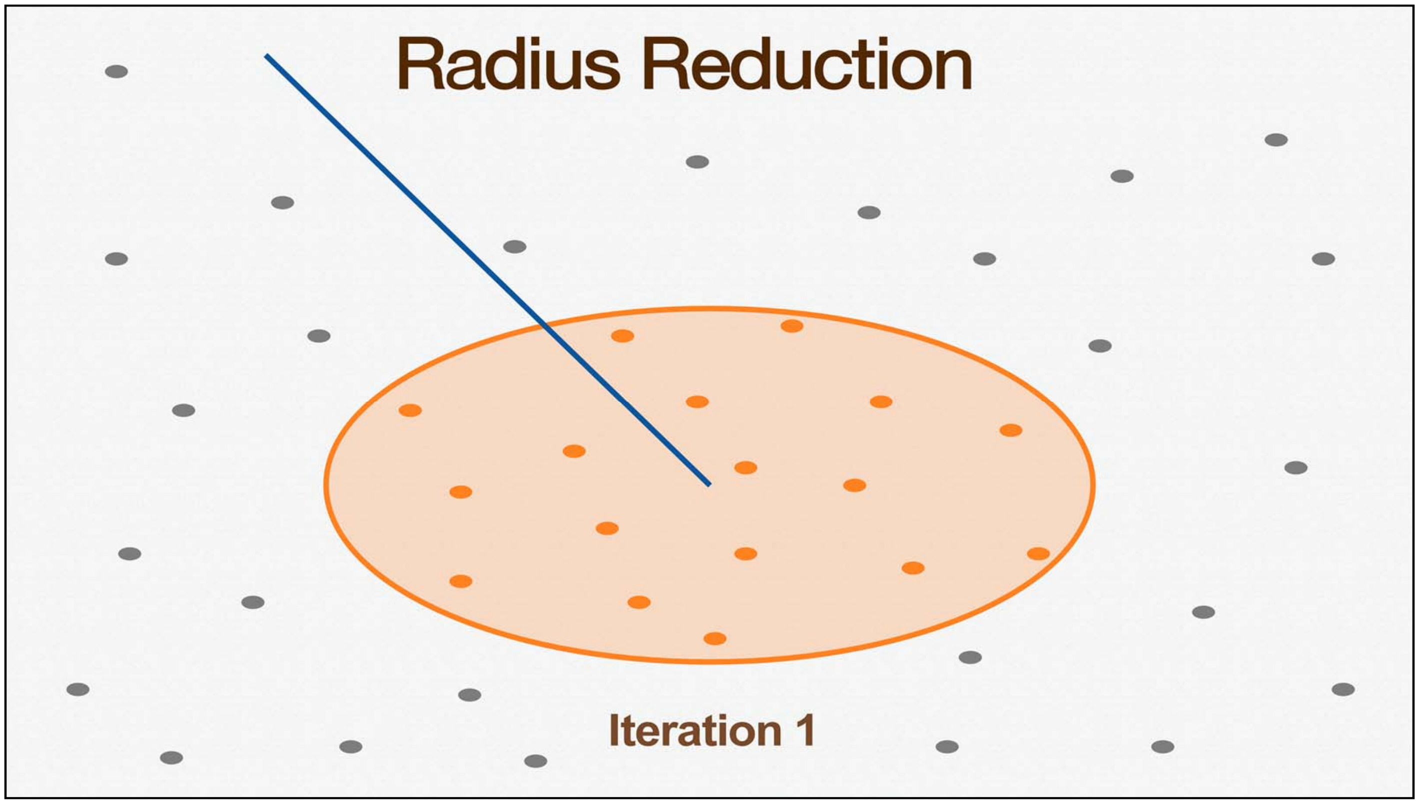
Path tracing
Bidirectional path tracing
Metropolis light transport
Photon mapping
Progressive photon mapping

In 2008, Hachisuka et al. have solved this memory bottleneck with progressive photon mapping. The idea of progressive photon mapping is to update incrementally a sequence of photon mapping results using a limited number of photons at a time.



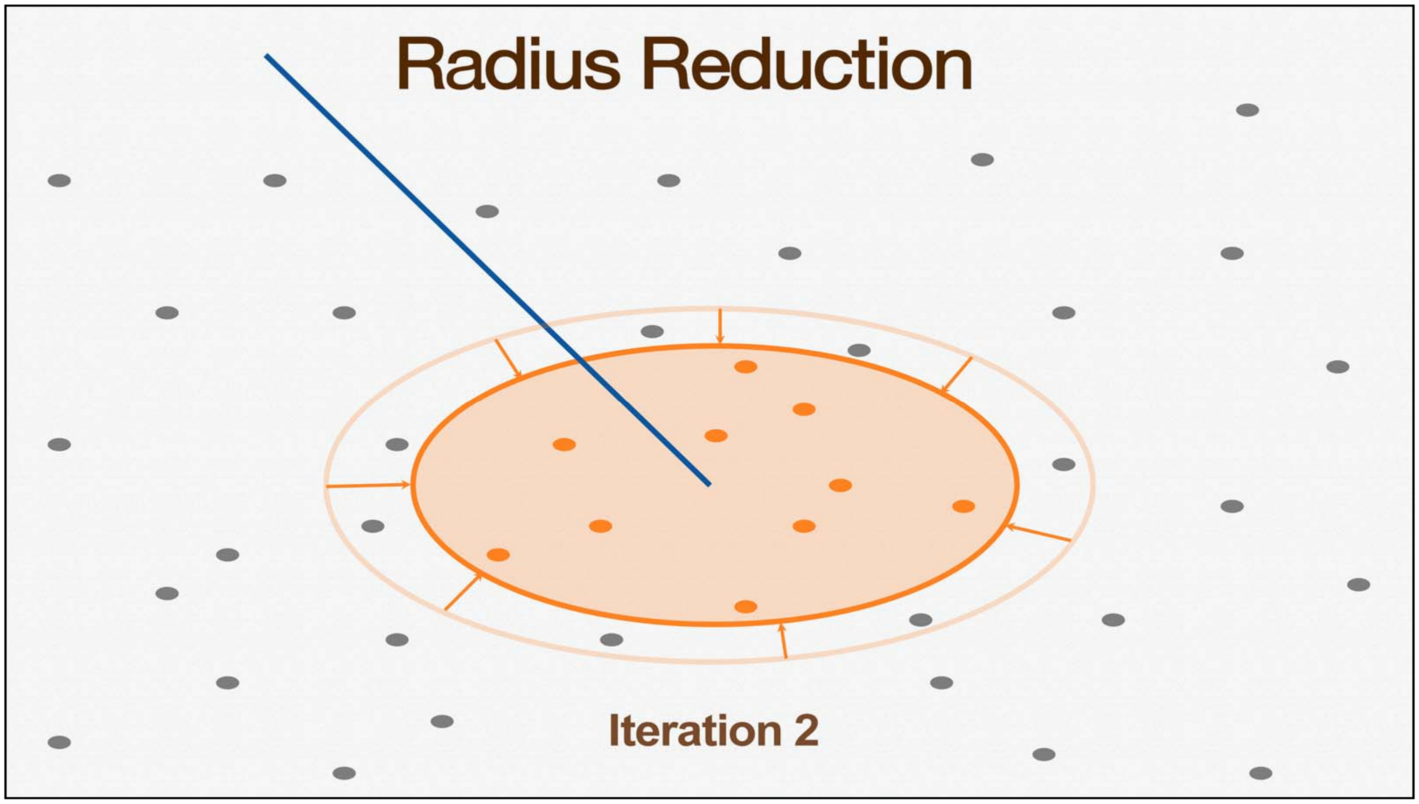
Progressive Photon Mapping

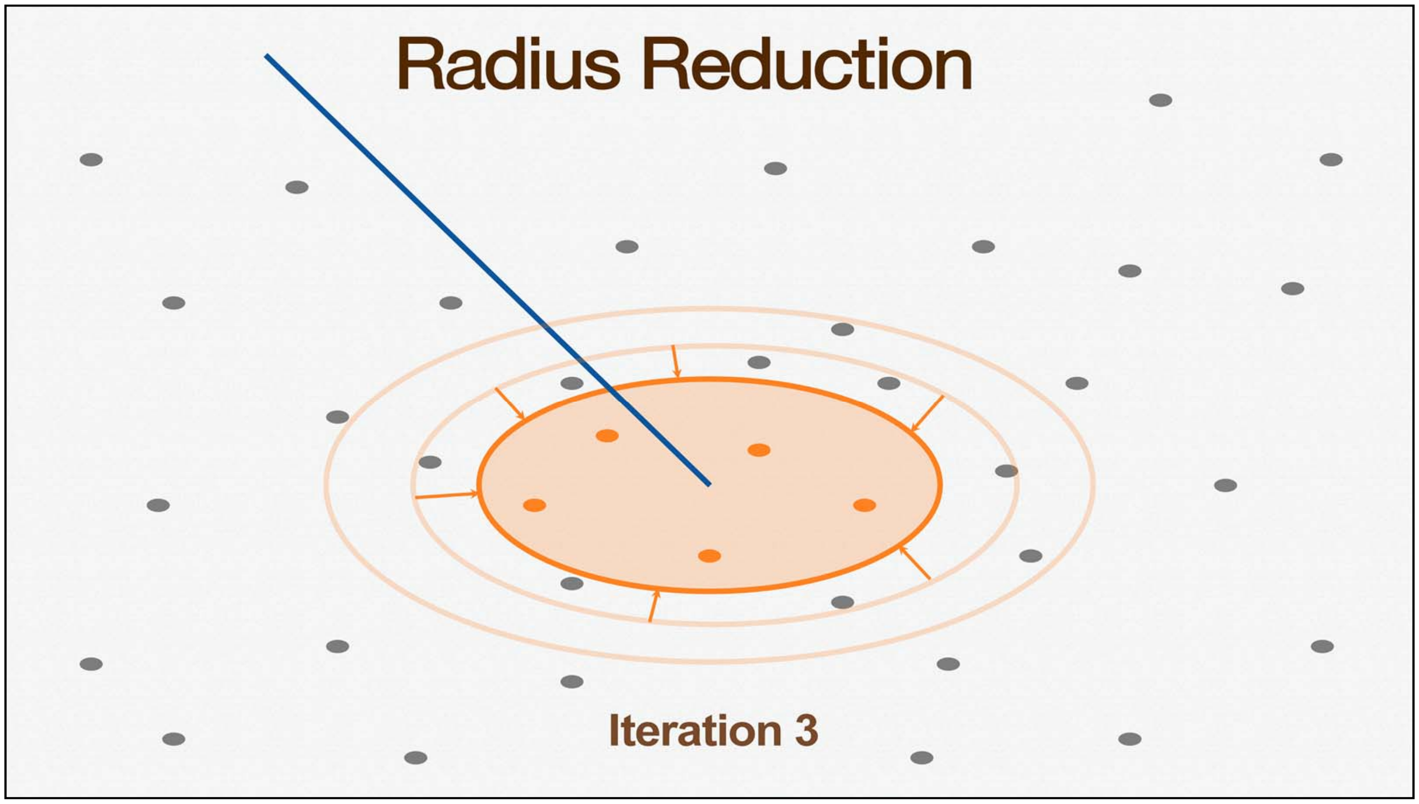
Progressive photon mapping is an iterative method. Before the iteration, eye rays are traced and stored in locations. Then, for every iteration, new (independent) photons are traced. Finally, the most important step, is when the kernel radius is reduced; and then a new iteration begins.



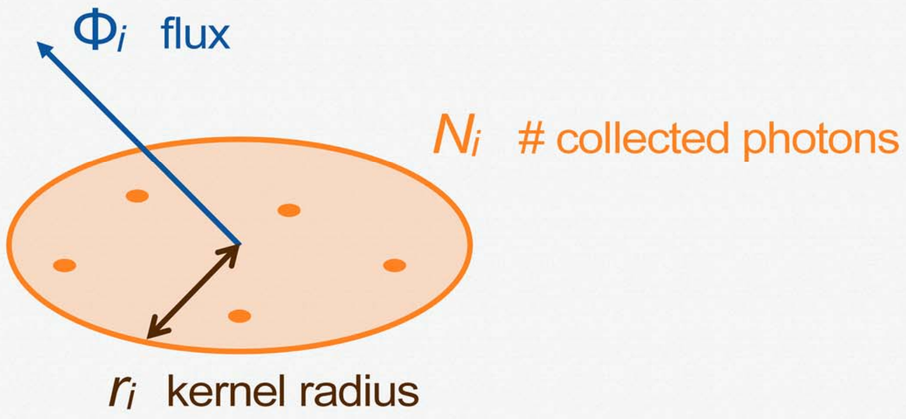
The key of progressive photon mapping is to reduce the kernel radius in every iteration, such that the bias vanishes.

Radius Reduction

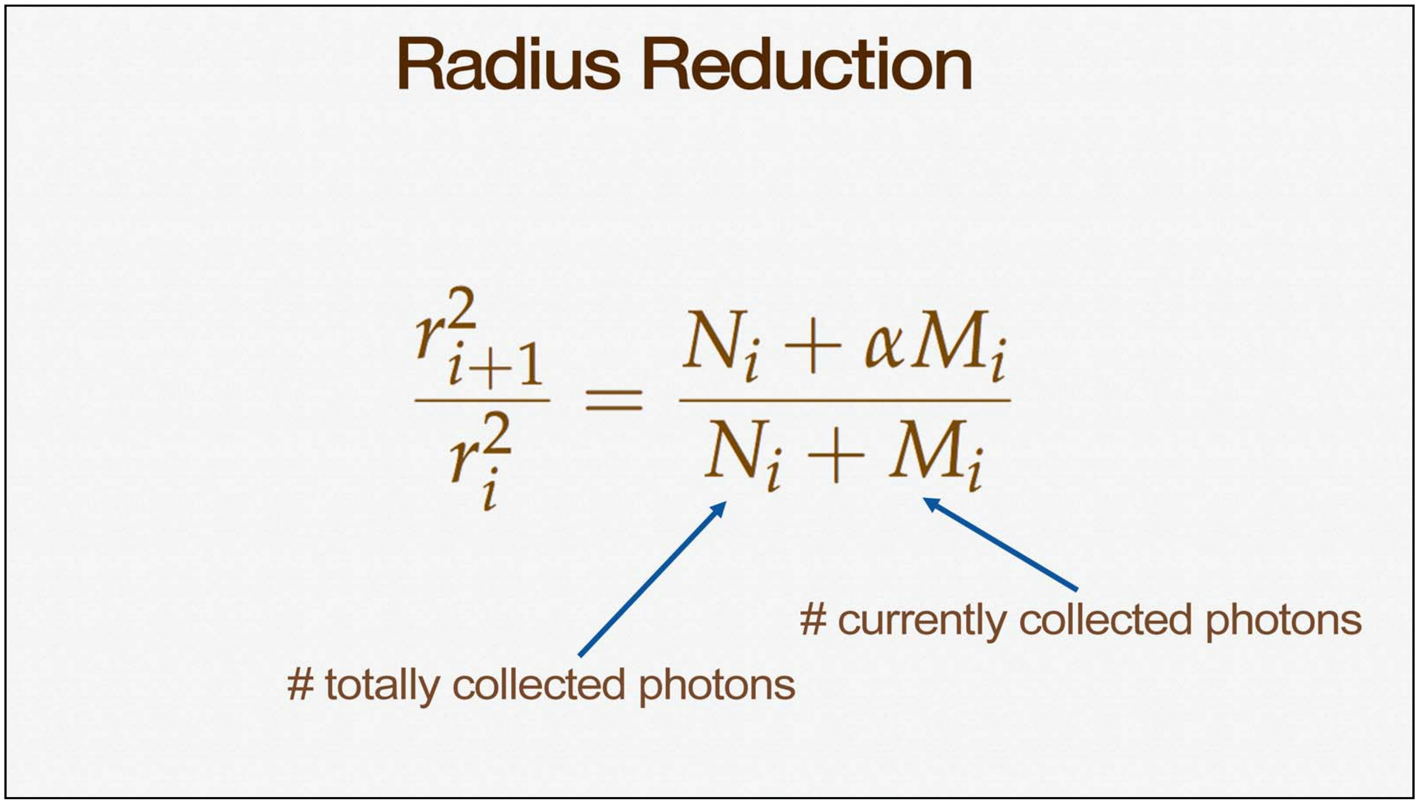



Radius Reduction
Iteration 3

Locations with Statistics



It does this by using statistics which are stored in every location of radiance estimation, namely the number of collected photons, the flux, and the kernel radius.



Radius Reduction

$$\frac{r_{i+1}^2}{r_i^2} = \frac{N_i + \alpha M_i}{N_i + M_i}$$

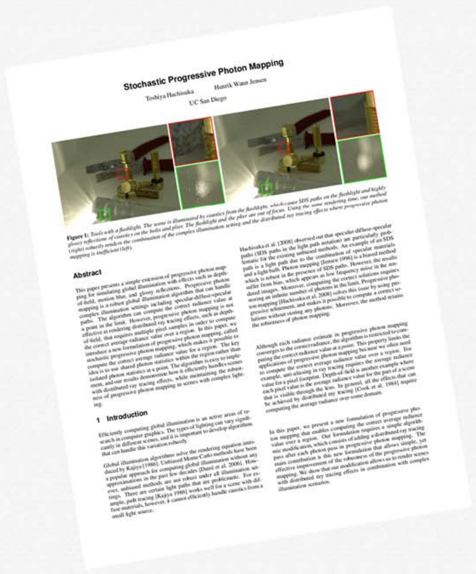
currently collected photons

totally collected photons

The exact radius reduction from iteration i to i + 1 is calculated using this update rule, which involves the use of these local statistics, such as the number of collected photons.

Stochastic PPM

Hachisuka & Jensen (2009)



Stochastic Progressive Photon Mapping

Toshiya Hachisuka Henrik Wann Jensen

UC San Diego

Figure 1: *Tools with a flashlight. The scene is illuminated by caustics from the flashlight, which cause SDS paths on the flashlight and highly glossy reflections of caustics on the bolts and plier. The flashlight and the plier are out of focus. Using the same rendering time, our method (right) robustly renders the combination of the complex illumination setting and the distributed ray tracing effects where progressive photon mapping is inefficient (left).*

Abstract

This paper presents a simple extension of progressive photon mapping for simulating global illumination with effects such as depth-of-field, motion blur, and glossy reflections. Progressive photon mapping is a robust global illumination algorithm that can handle complex illumination settings including specular-diffuse-specular paths. The algorithm can compute the correct radiance value at a point in the limit. However, progressive photon mapping is not effective at rendering distributed ray tracing effects, such as depth-of-field, that requires multiple pixel samples in order to compute the correct average radiance value over a region. In this paper, we introduce a new formulation of progressive photon mapping, called stochastic progressive photon mapping, which makes it possible to compute the correct average radiance value for a region. The key idea is to use shared photon statistics within the region rather than isolated photon statistics at a point. The algorithm is easy to implement, and our results demonstrate how it efficiently handles scenes with distributed ray tracing effects, while maintaining the robustness of progressive photon mapping in scenes with complex lighting.

1 Introduction

Efficiently computing global illumination is an active area of research in computer graphics. The types of lighting can vary significantly in different scenes, and it is important to develop algorithms that can handle this variation robustly.

Global illumination algorithms solve the rendering equation introduced by Kajiya [1986]. Unbiased Monte Carlo methods have been a popular approach for computing global illumination without any approximations in the past few decades [Dutré et al. 2006]. However, unbiased methods are not robust under all illumination settings. There are certain light paths that are problematic. For example, path tracing [Kajiya 1986] works well for a scene with diffuse materials, however, it cannot efficiently handle caustics from a small light source.

Hachisuka et al. [2008] observed out that specular-diffuse-specular paths (SDS paths in the light path notation) are particularly problematic for the existing unbiased methods. An example of an SDS path is a light path due to the combination of specular materials and a light bulb. Photon mapping [Jensen 1996] is a biased method which is robust in the presence of SDS paths. However, the results suffer from bias, which appears as low frequency noise in the rendered images. Moreover, computing the correct solutions requires storing an infinite number of photons in the limit. Progressive photon mapping [Hachisuka et al. 2008] solves this issue by using progressive refinement, and makes it possible to compute a correct solutions without storing any photons. Moreover, the method retains the robustness of photon mapping.

Although each radiance estimate in progressive photon mapping converges to the correct radiance, the algorithm is restricted to computing the correct radiance value at a point. This property limits the applications of progressive photon mapping because we often need to compute the correct average radiance value over a region. For example, anti-aliasing in ray tracing requires the average radiance value for a pixel footprint. Depth-of-field is another example where each pixel value is the average radiance value for the part of a scene that is visible through the lens. In general, all the effects that can be achieved by distributed ray tracing [Cook et al. 1984] require computing the average radiance over some domain.

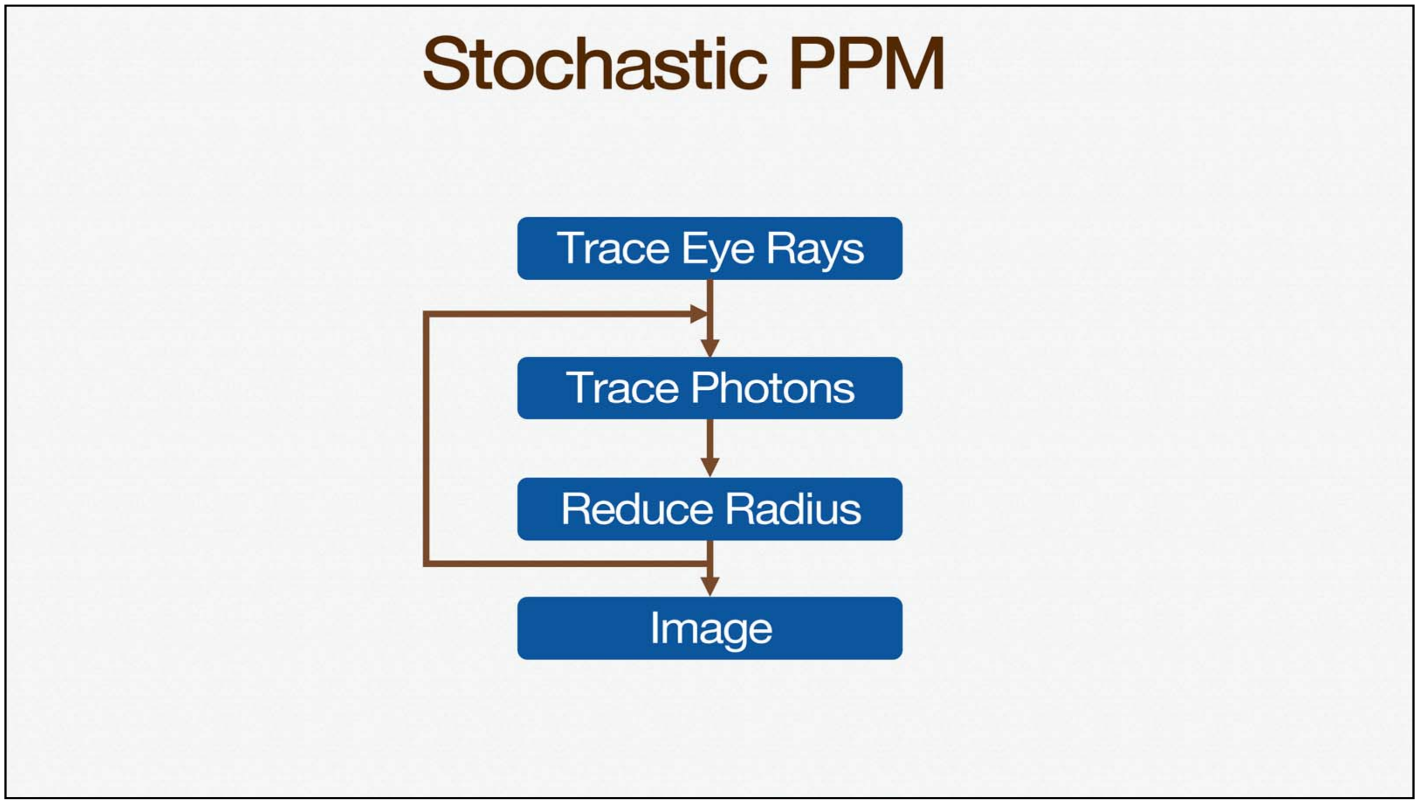
In this paper, we present a new formulation of progressive photon mapping that enables computing the correct average radiance value over a region. Our formulation requires a simple algorithmic modification, which consists of adding a distributed ray tracing pass after each photon pass in progressive photon mapping. The main contribution is this new formulation that allows simple, yet effective improvement of the robustness of the progressive photon mapping. We show that our modification allows us to render scenes with distributed ray tracing effects in combination with complex illumination scenarios.

★ Glossy reflections

★ Depth of field

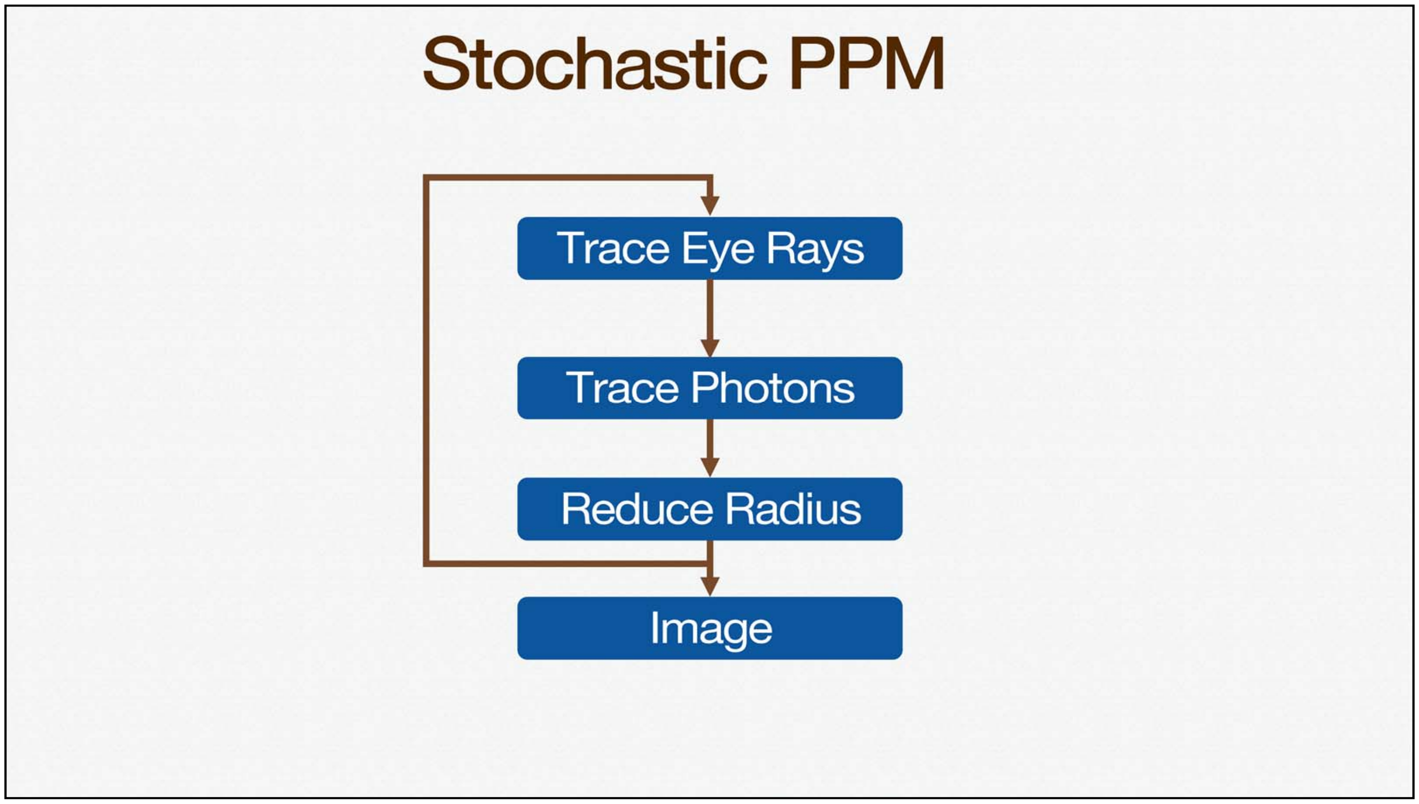
★ Motion blur

In 2009, in a follow up work, Hachisuka and Jensen have generalized the method to stochastic progressive photon mapping. It includes additional effects like glossy reflections, depth of field, and motion blur.



Stochastic PPM

The main change of stochastic progressive photon mapping is that instead of tracing eye rays once, the rays are traced in every iteration.

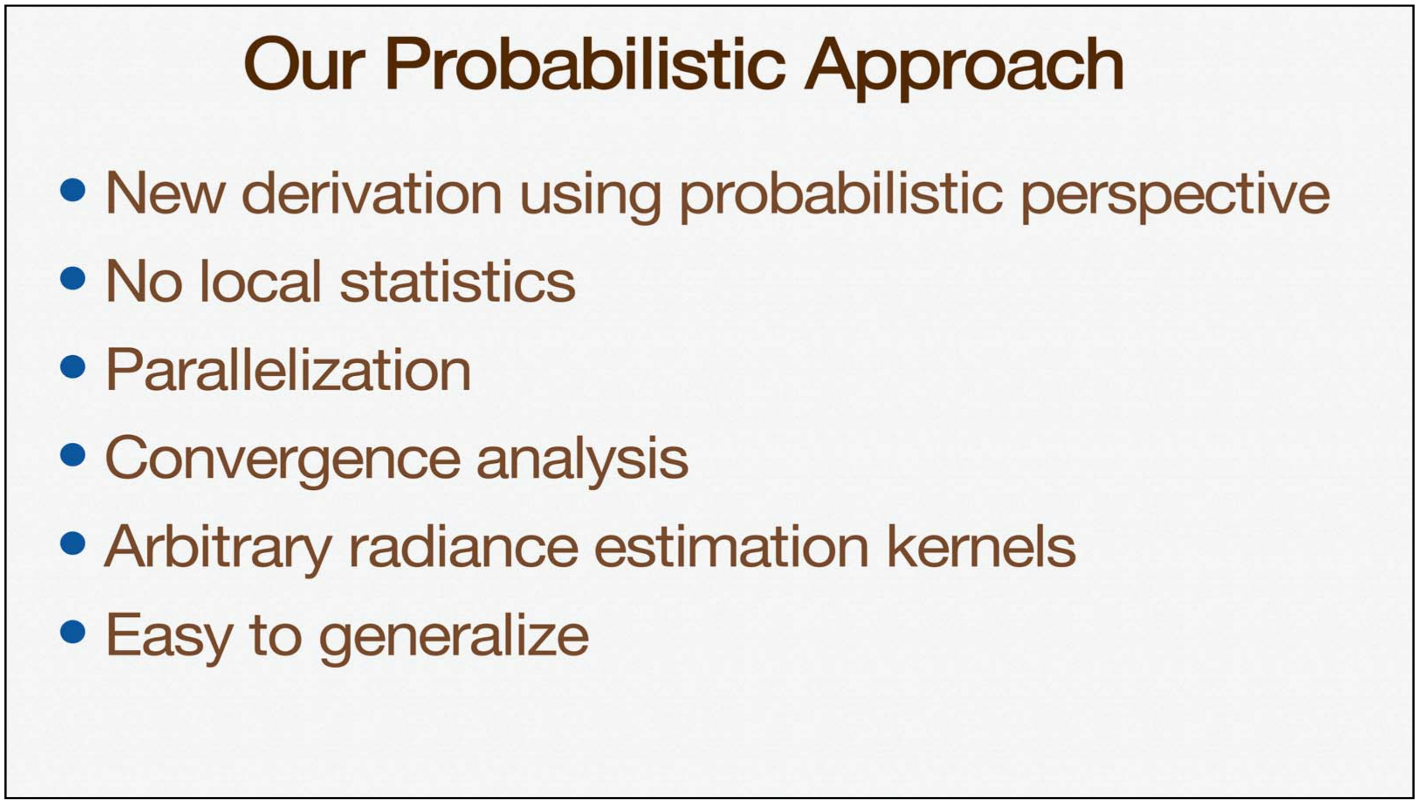


Stochastic PPM

The main change to progressive photon mapping is that instead of tracing eye rays once, they are traced in every iteration.

Great, but ...

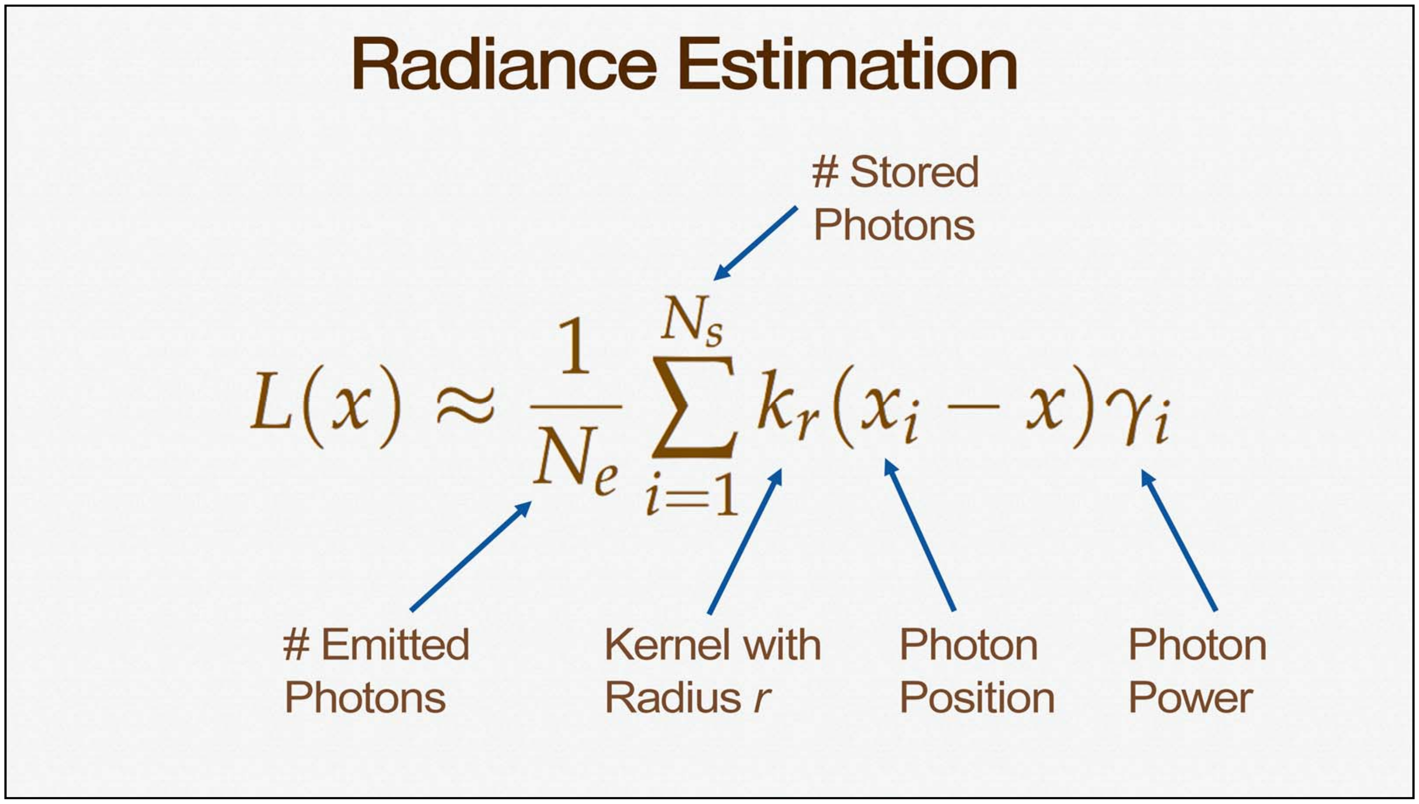
While progressive photon mapping is great, we wanted to make a step back and look at it from a different perspective.



Our Probabilistic Approach

- New derivation using probabilistic perspective
- No local statistics
- Parallelization
- Convergence analysis
- Arbitrary radiance estimation kernels
- Easy to generalize

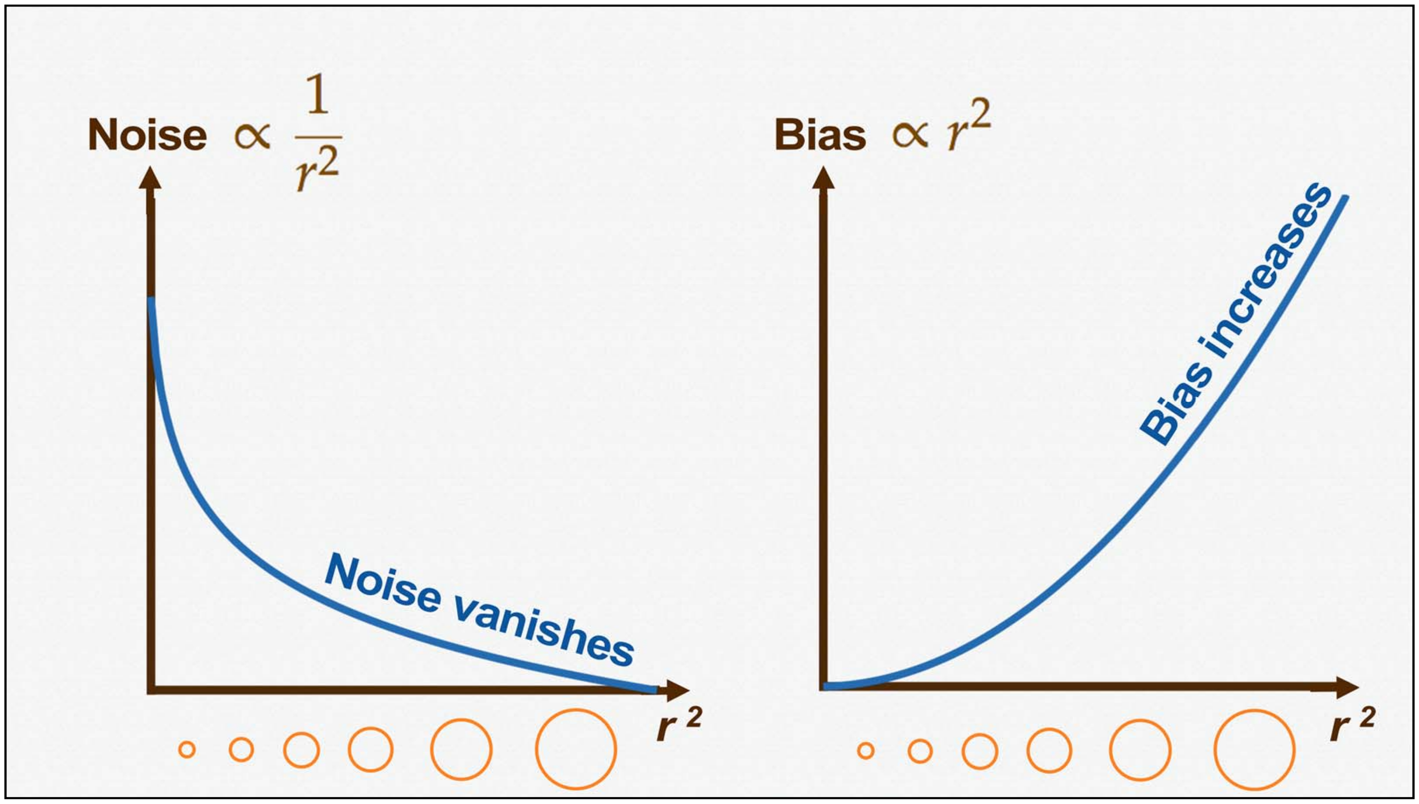
We found a new derivation of progressive photon mapping which does not require local statistics and is trivial to parallelize. We also provide convergence analysis. Furthermore, our reformulation of progressive photon mapping generalizes to arbitrary radiance estimation kernels. And finally, it is easy to generalize to other radiance estimates like volumetric photon mapping and the recent work of beam radiance estimates.



Radiance Estimation

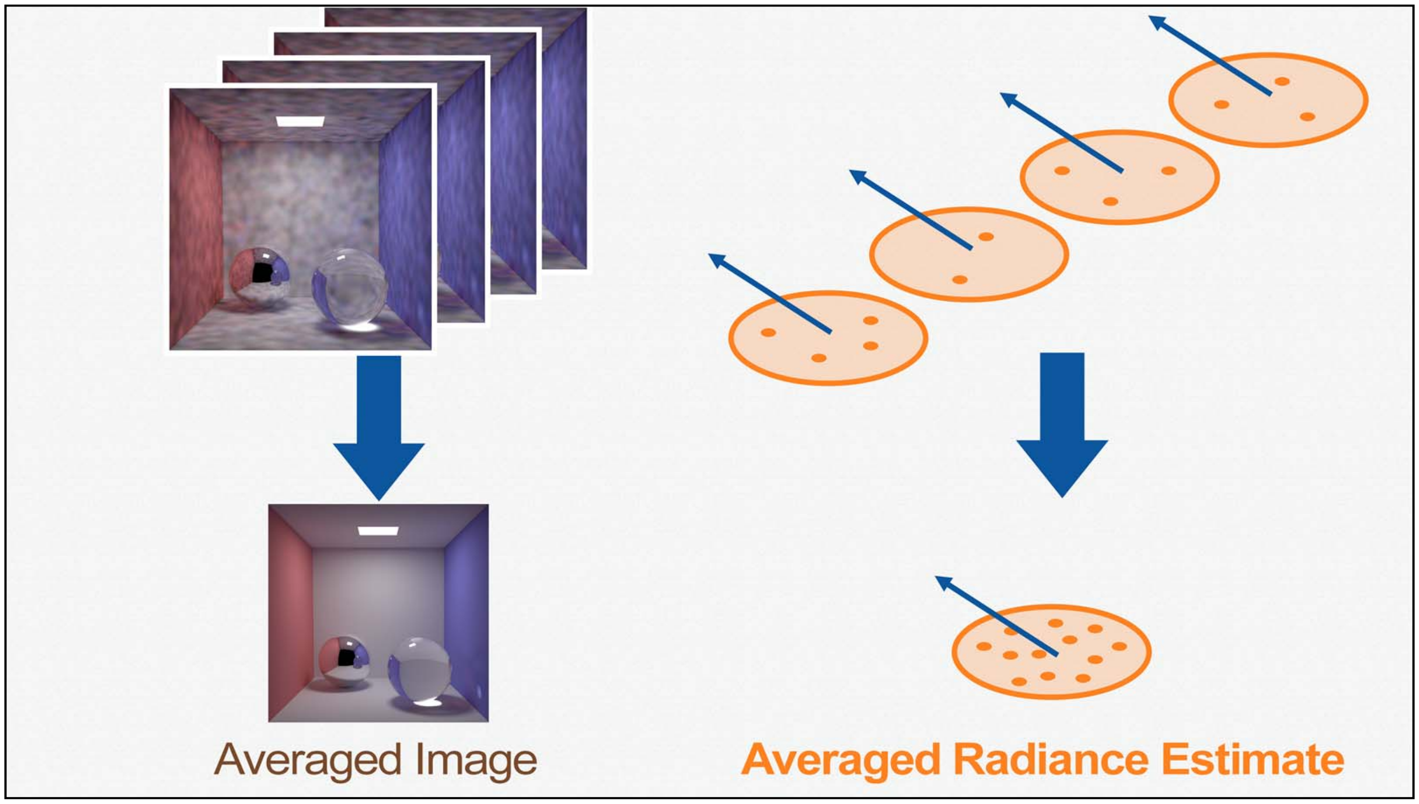
$$L(x) \approx \frac{1}{N_e} \sum_{i=1}^{N_s} k_r(x_i - x)\gamma_i$$

Let us start with a probabilistic analysis of the radiance estimation. A radiance estimate is a Monte Carlo integral. N_e is the number of emitted photons. N_s is the number of stored photons. k_r is a kernel with radius r. x_i is the position of the photon and gamma_i is the power of the photon, already pre-multiplied with the BRDF, which amounts to the reflected radiance. Since a Monte Carlo integral is a stochastic method, we can look at the statistics of radiance estimation.

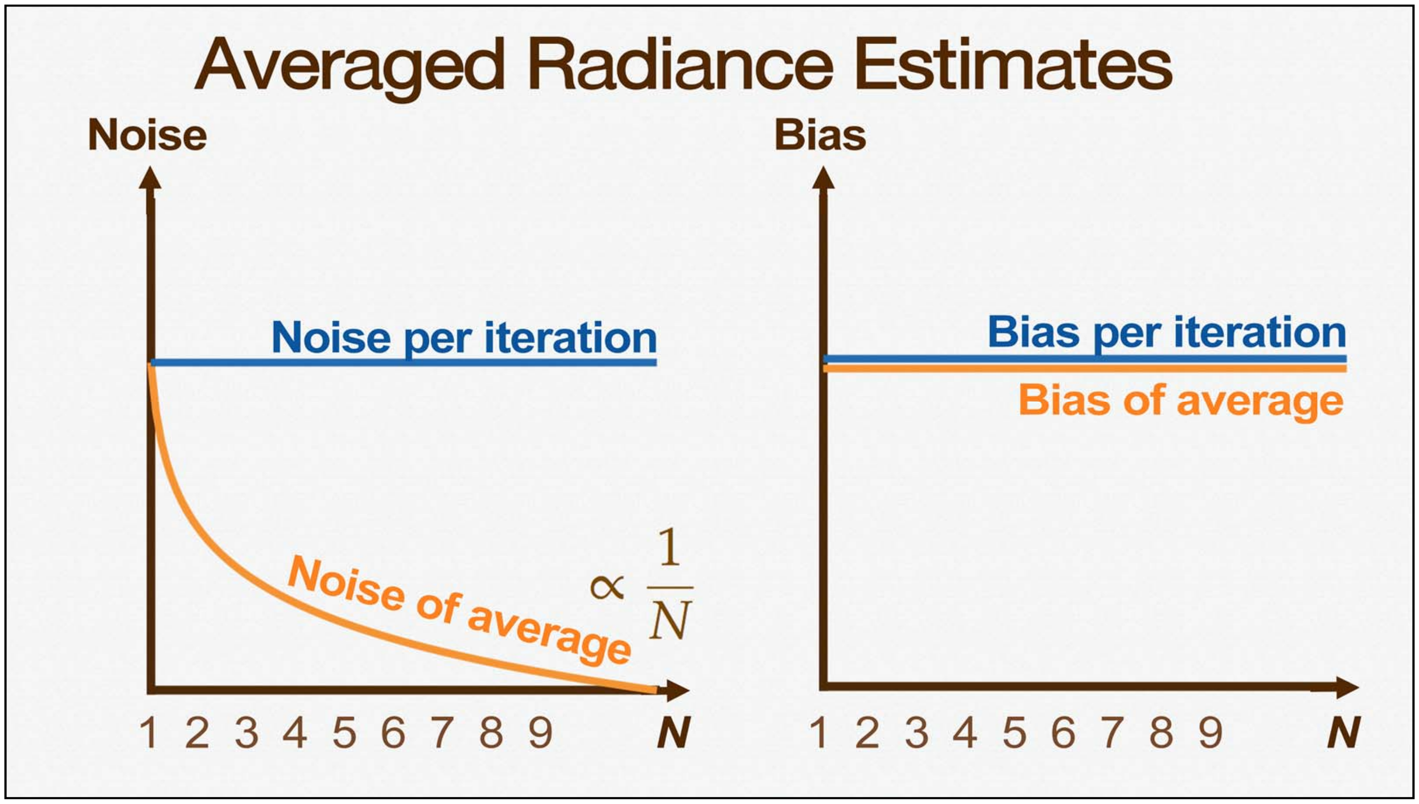


Namely, the noise and the bias. It turns out that the noise of the radiance estimate is inverse proportional to the squared radius. Intuitively, it is clear that increasing the radius has the effect of averaging over more photons, and will therefore reduce the noise.

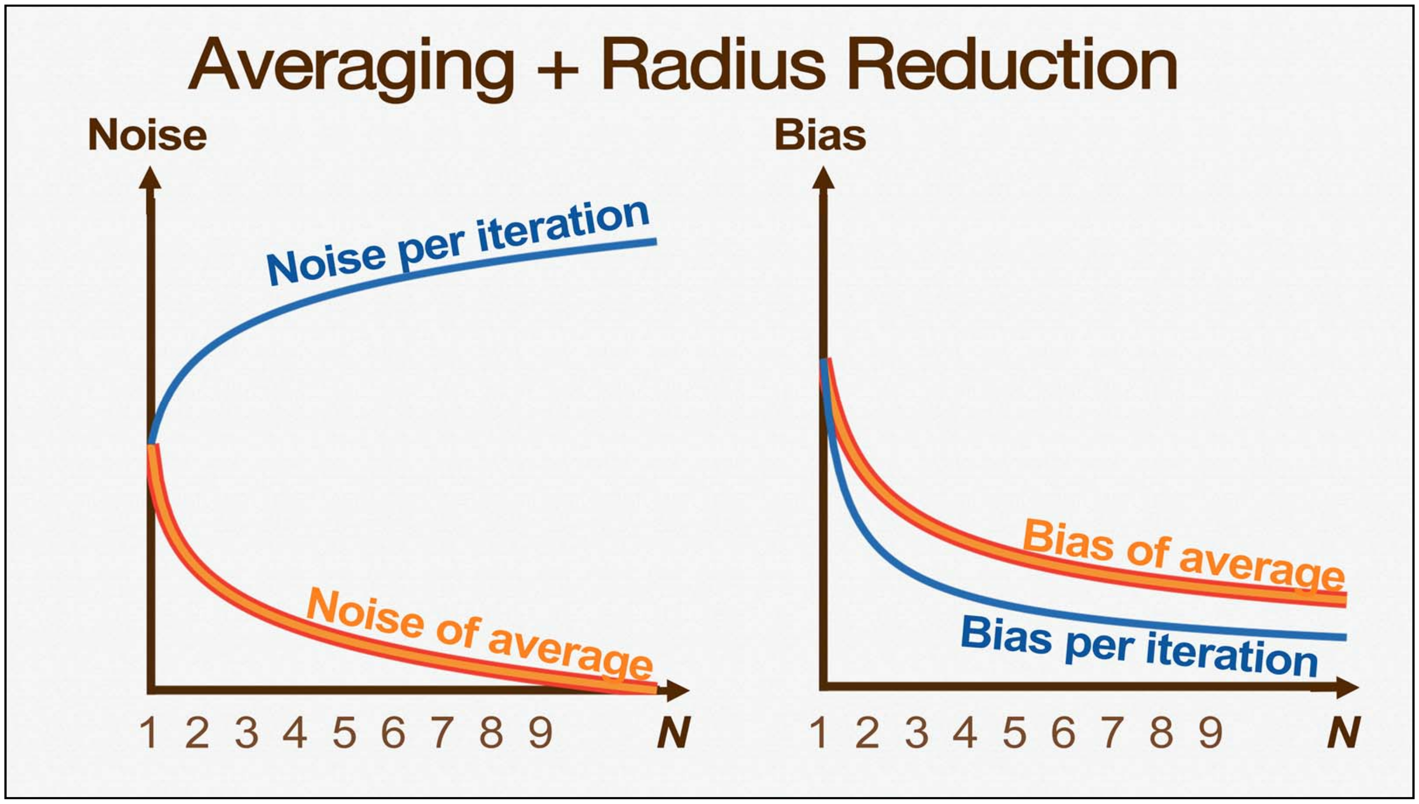
The bias, on the other hand, is proportional to the squared radius. In this case, increasing the radius will include more features like caustics, and therefore will increase the bias. Note that the bias only vanishes if the radius is 0. For any fixed radius, there is this classic trade-off where we can only achieve smoothness or unbiasedness but not both at the same time.



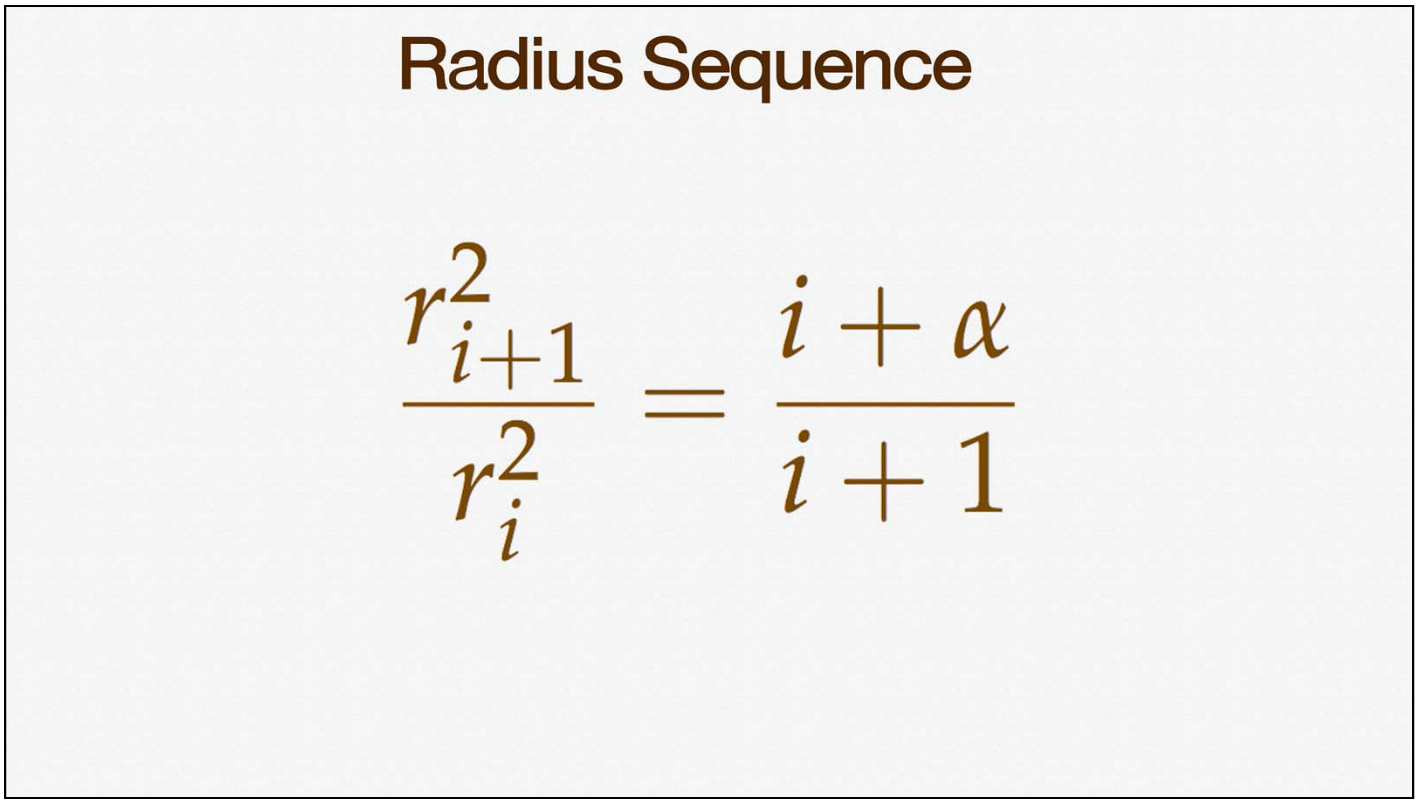
Following the idea of the original progressive photon mapping, we would like to split up the rendering into multiple iterations in order to remove the memory bottleneck. The easiest way to combine the iterations is to average the images. We show in the paper that the averaged image converges if the averaged radiance estimates converge. Let's see what happens if we average the radiance estimate over many iterations.



While the noise per iteration remains constant, the noise of the average will vanish with 1/N and the averaged image becomes smoother. This is great -- but how about the bias? The bias per iteration remains constant, and unfortunately, the bias of the averaged radiance estimate remains constant as well. This is not surprising, because we are simply averaging images, and this will reduce the noise, but not the bias.



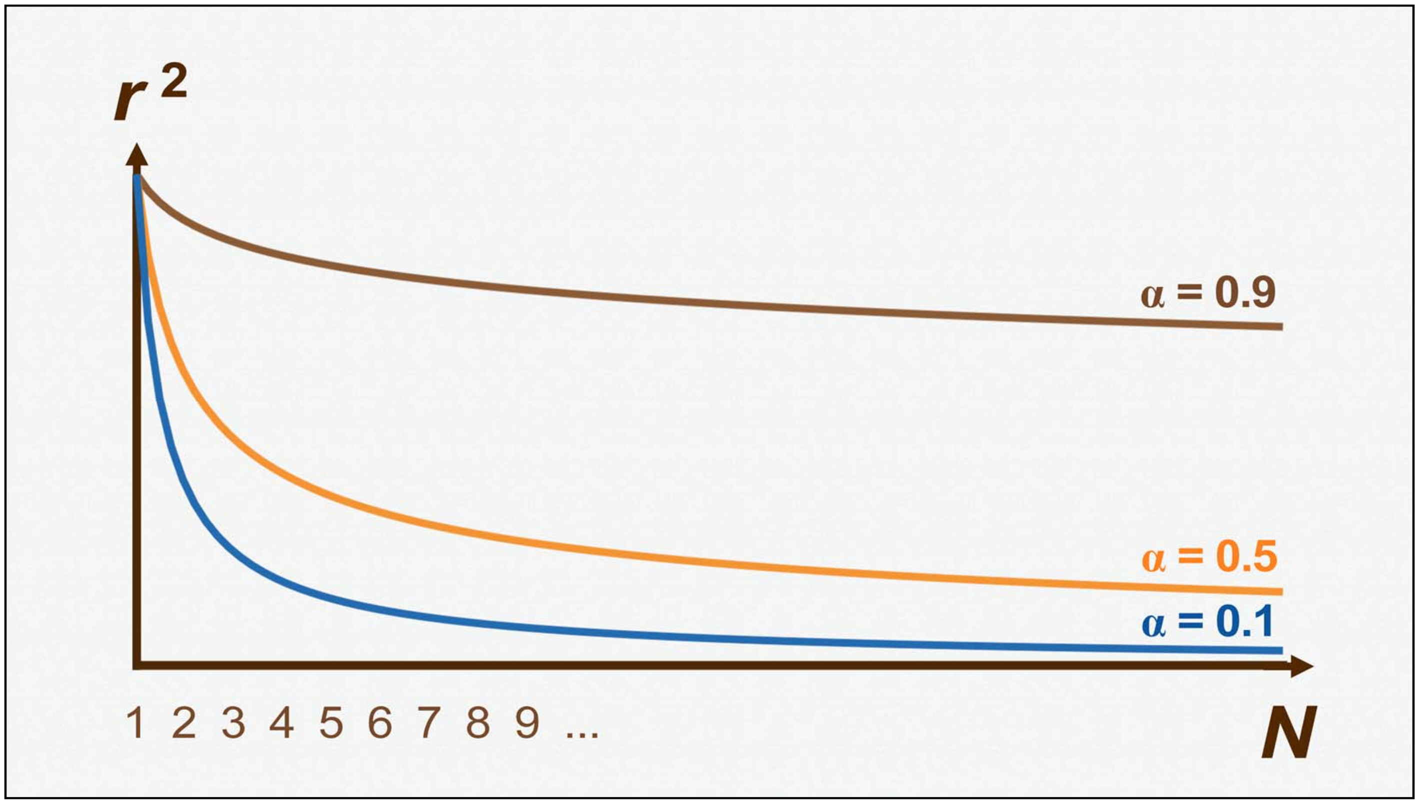
The only way to reduce the bias of the average is by reducing the kernel radius. If the radius is continuously reduced, not only the bias per iteration, but also the the bias of the average vanishes. But, if we reduce the radius, as we have seen before, the noise per iteration will increase. The trick of progressive photon mapping is to let the radius decrease slow enough such that the noise of the *averaged* radiance estimate still vanishes in the limit. To summarize: we look for a radius sequence which reduces slow enough for both the noise and bias of the average to vanish in the limit.



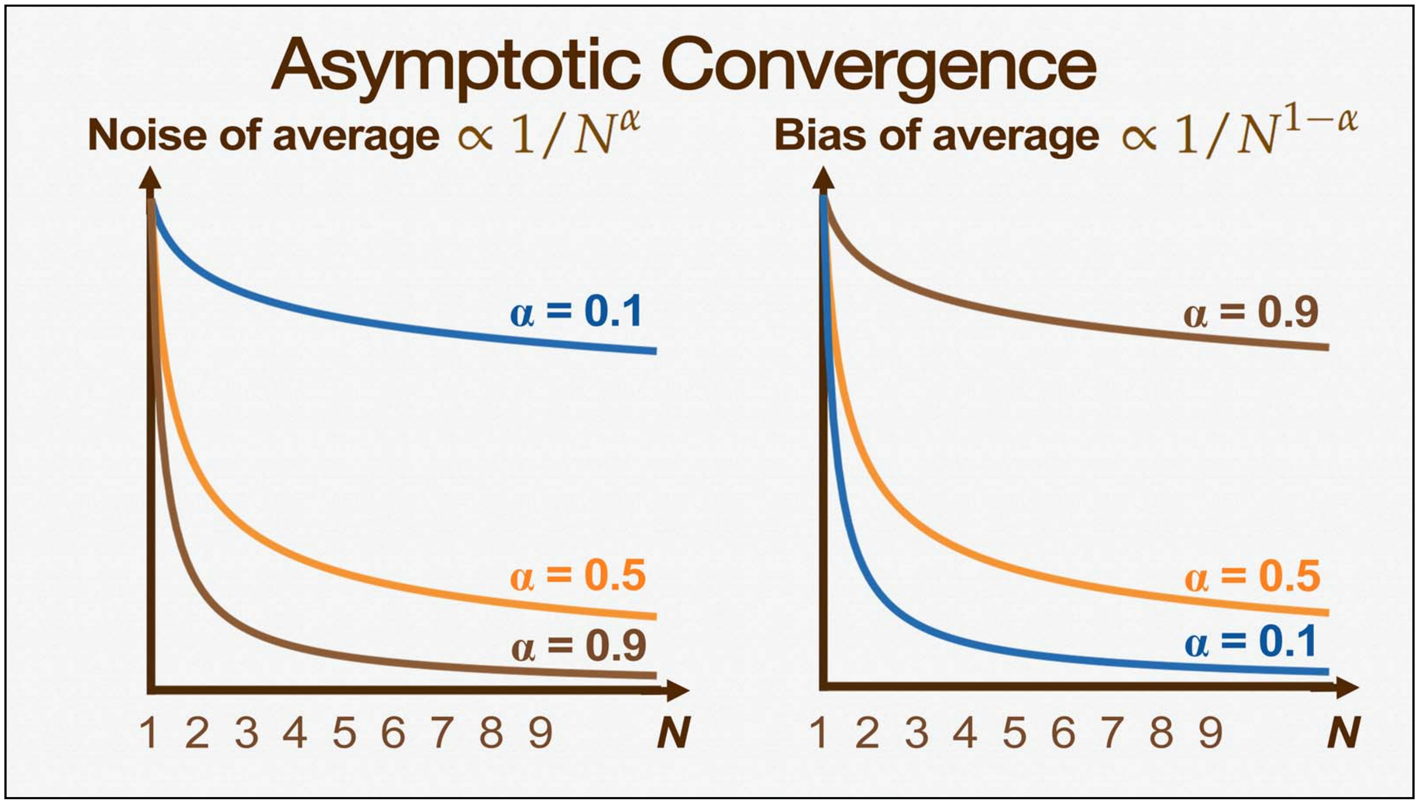
Radius Sequence

$$\frac{r_{i+1}^2}{r_i^2} = \frac{i+\alpha}{i+1}$$

The radius sequence which allows us to walk this fine line where both noise and bias of averaged radiance estimates vanish in the limit is given by this formula. A detailed proof can be found in our paper. Our proposed radius sequence is defined recursively. i is the iteration number, and alpha is a parameter which must be between 0 and 1.

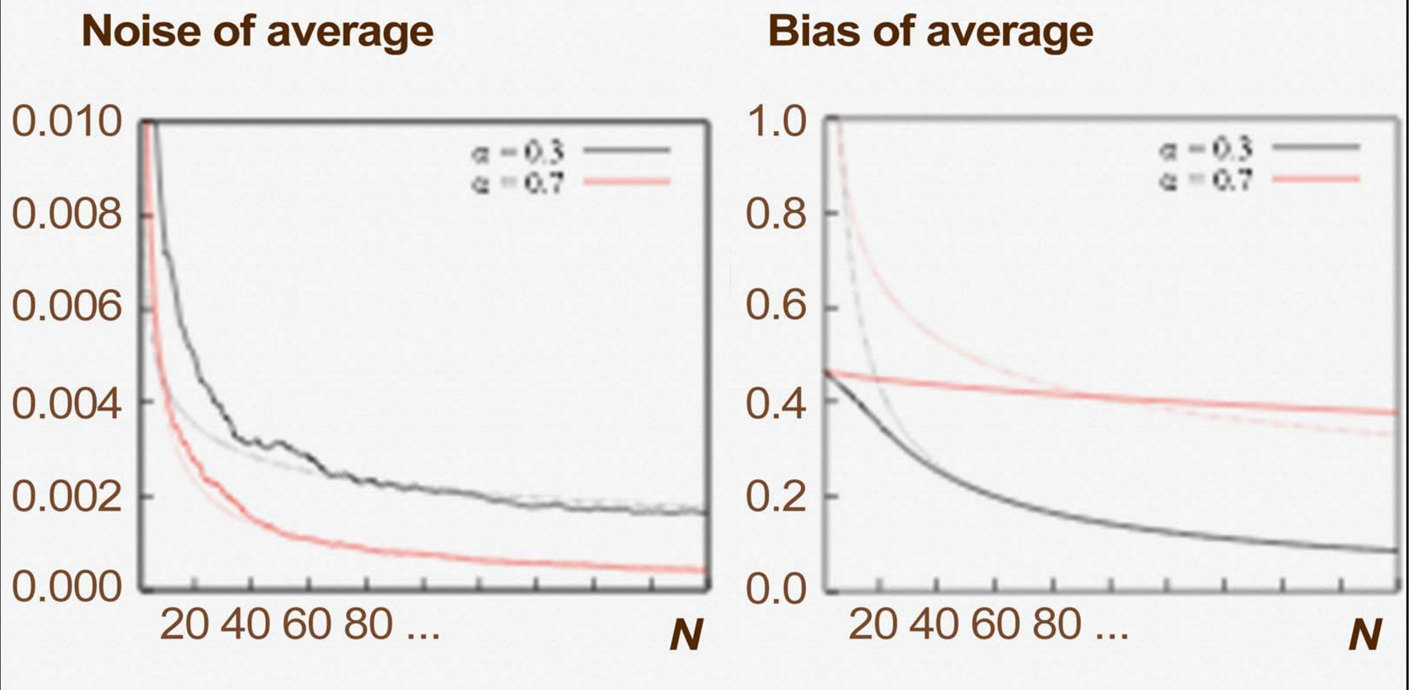


The parameter alpha controls how fast the radius is reduced with the number of iterations.

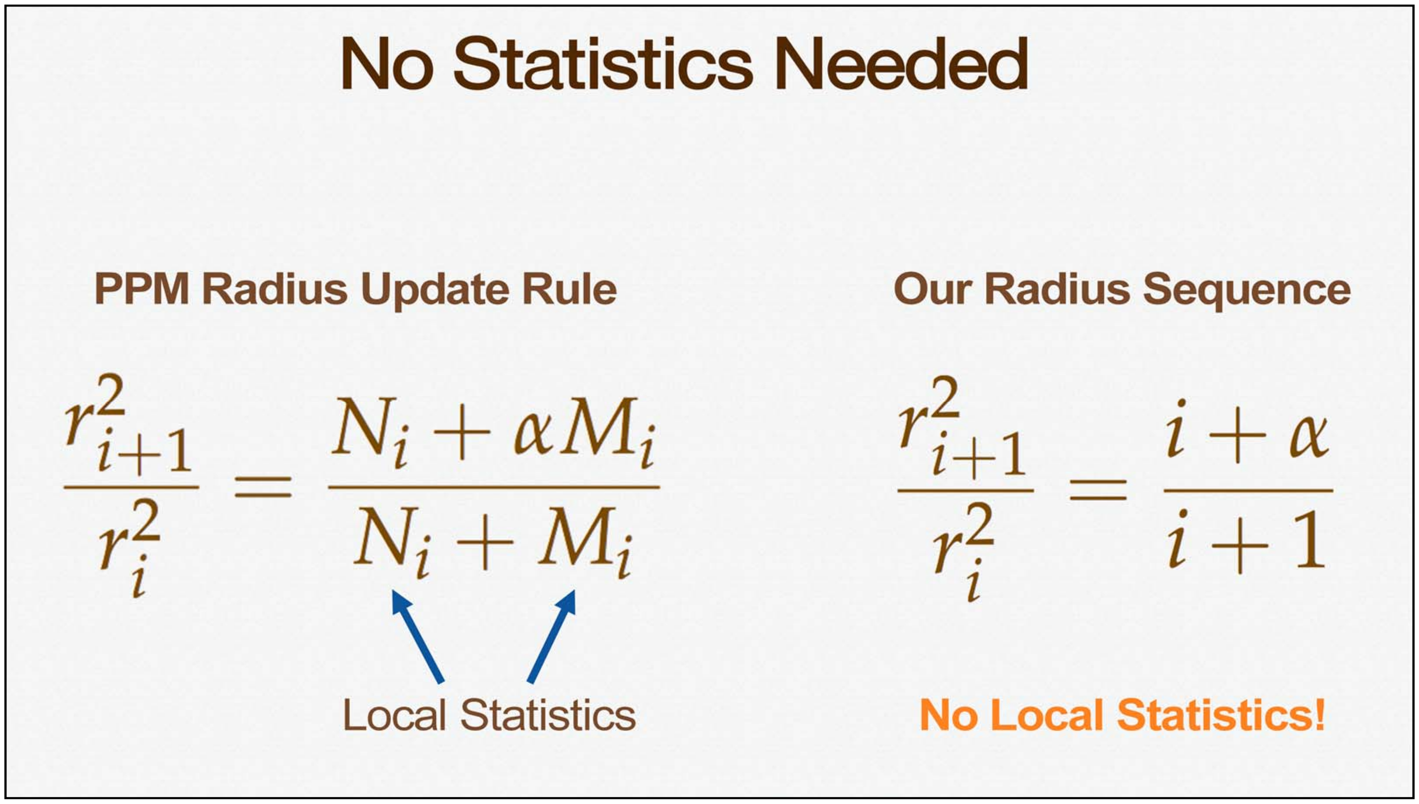


Using our proposed radius sequence, we found the following asymptotic convergence. The noise of the averaged radiance estimate vanishes proportionally to 1/N^alpha, where N is the number of iterations. The bias of the averaged radiance estimate vanishes proportionally to 1/N^(1-alpha). We can see here how alpha controls the convergence speed of noise and bias. A small alpha value like the blue curve reduces the noise slowly, but the bias will vanish quickly. A large alpha on the other hand, the brown curve, reduces the noise quickly, but the bias will go down slowly.

Empirical Validation



We have also empirically verified this asymptotic convergence. We see the convergence for a sequence of radiance estimates taken in the previous scene. Here the solid lines are the measured noise and bias for two different alpha values. The dotted lines are the asymptotic curves.



No Statistics Needed

PPM Radius Update Rule

$$\frac{r_{i+1}^2}{r_i^2} = \frac{N_i + \alpha M_i}{N_i + M_i}$$

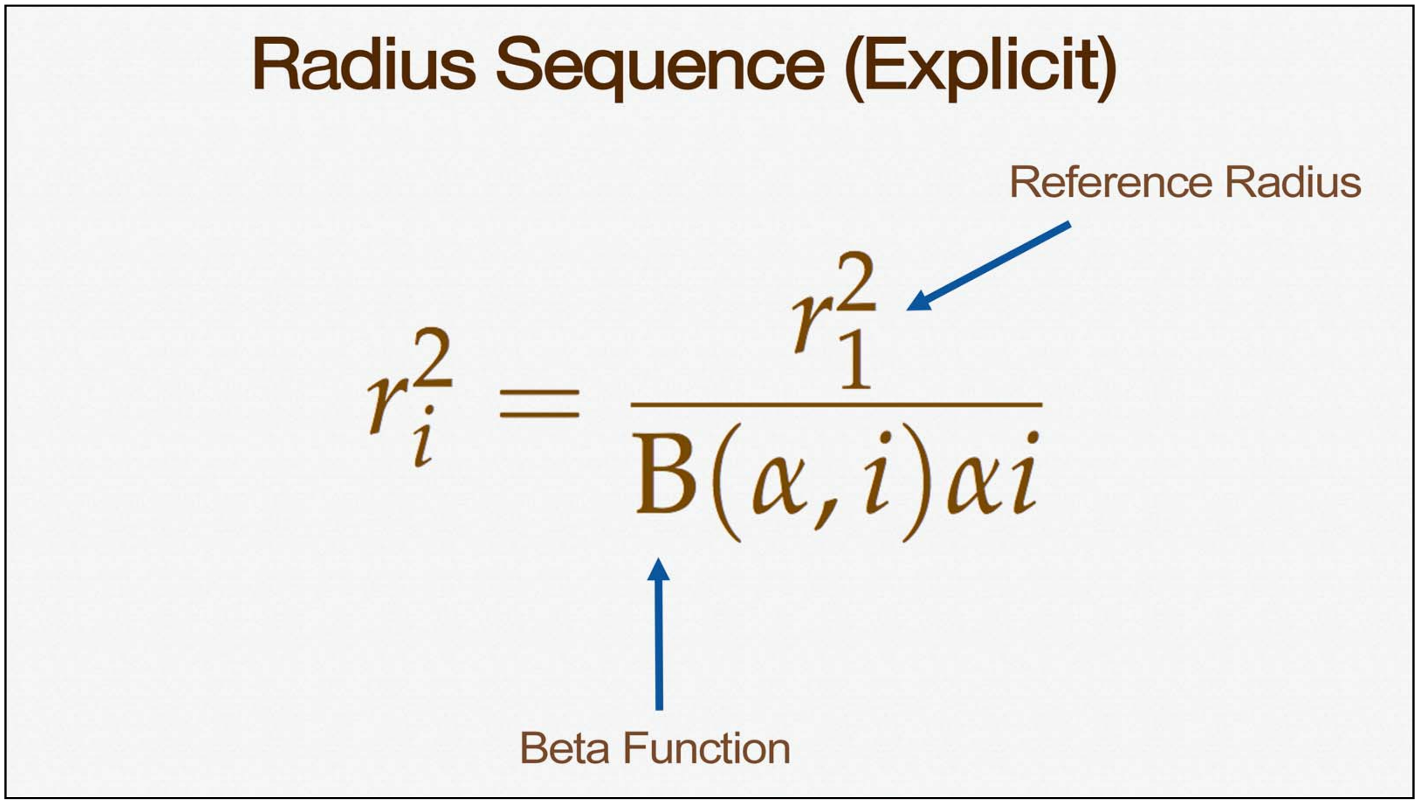
Local Statistics

Our Radius Sequence

$$\frac{r_{i+1}^2}{r_i^2} = \frac{i + \alpha}{i + 1}$$

No Local Statistics!

Our radius sequence is similar to the radius update rule proposed by Hachisuka et al., but with the important difference that it is entirely independent of local statistics such as collected number of photons, or local photon density. In fact, we show in the paper that, for locally constant photon density, the radius sequence is the same as the one from Hachisuka et al.



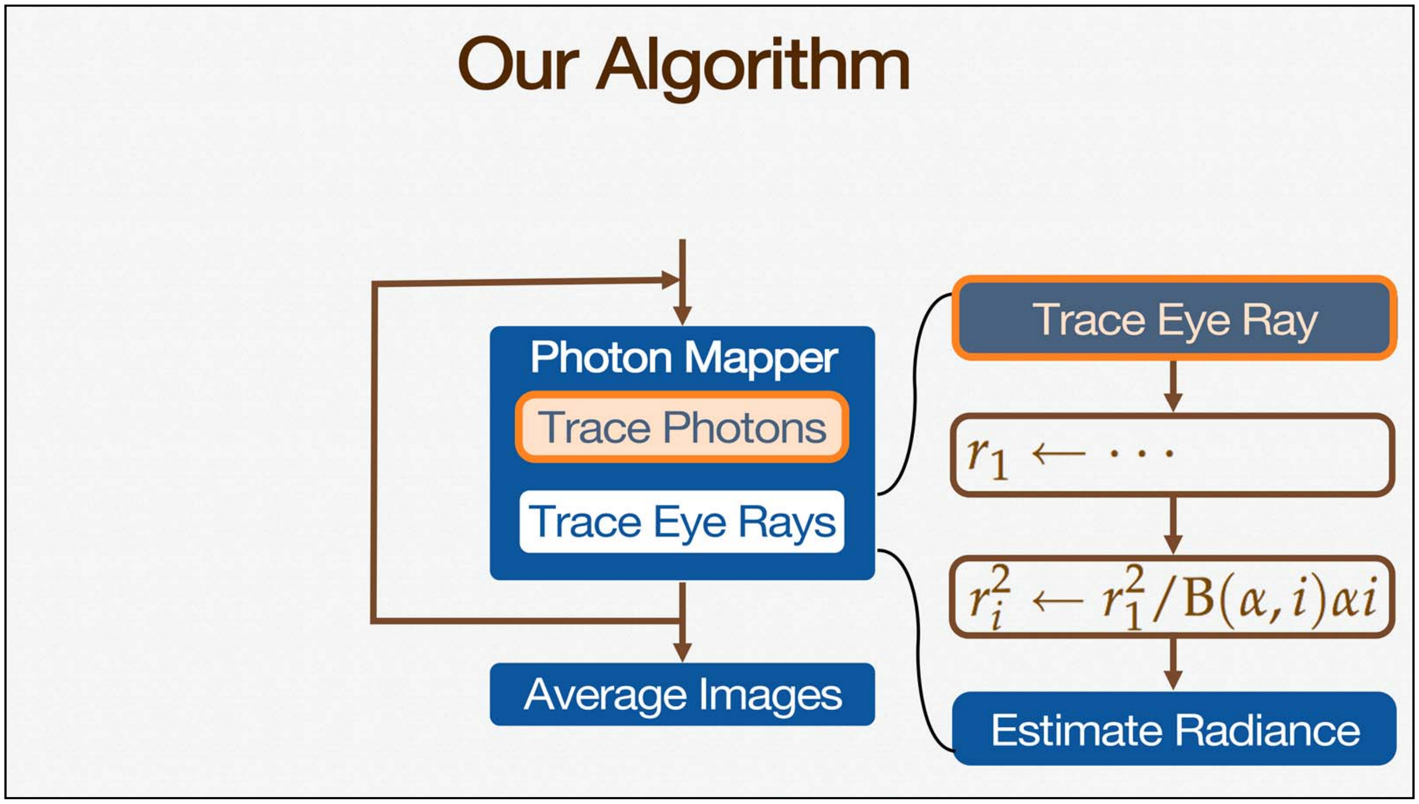
Radius Sequence (Explicit)

Reference Radius

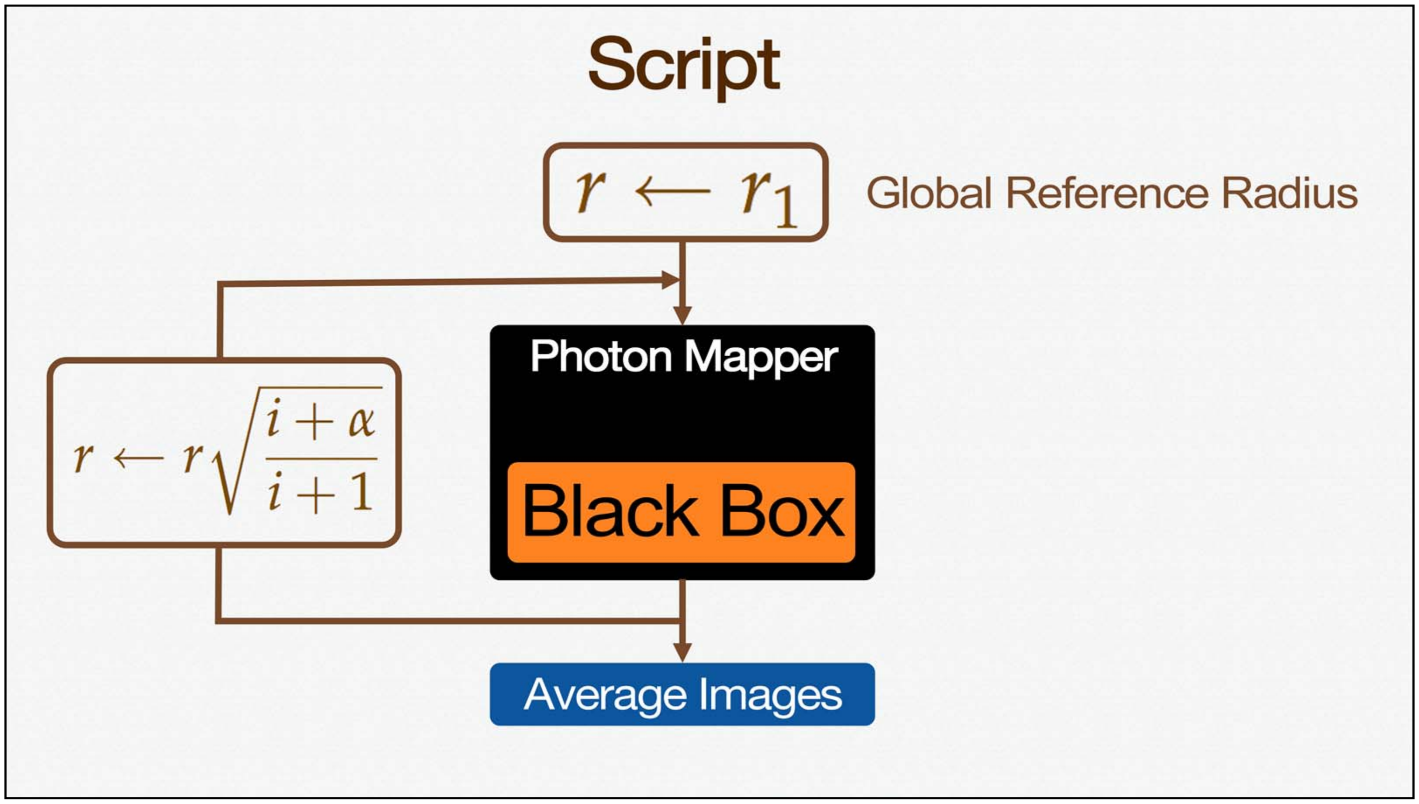
$$r_i^2 = \frac{r_1^2}{\mathrm{B}(\alpha, i)\alpha i}$$

Beta Function

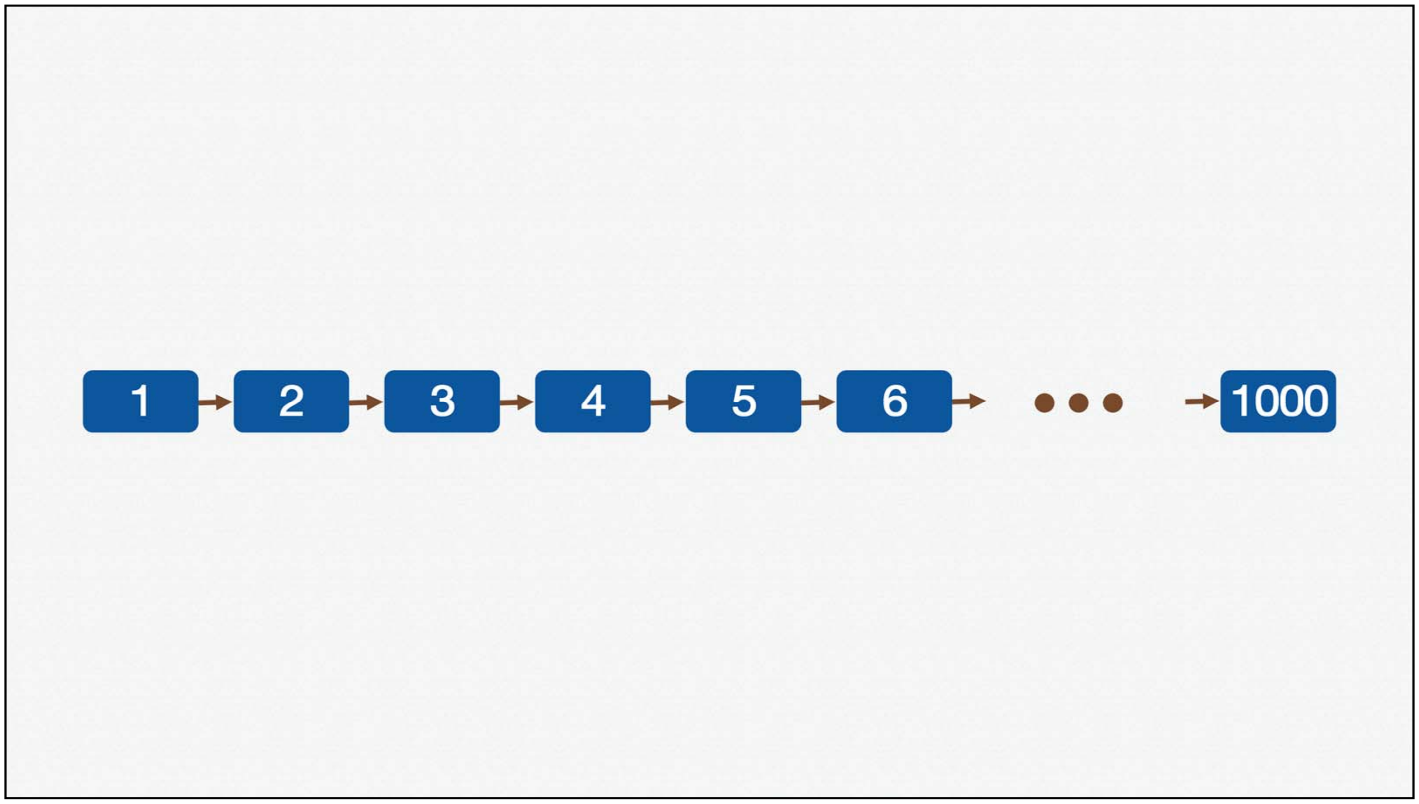
Instead of using a recursive formula, our radius sequence can also be written explicitly. r_1 is the initial reference radius which anchors the sequence. And B stands for the the Euler Beta function which is related the binomial coefficients.



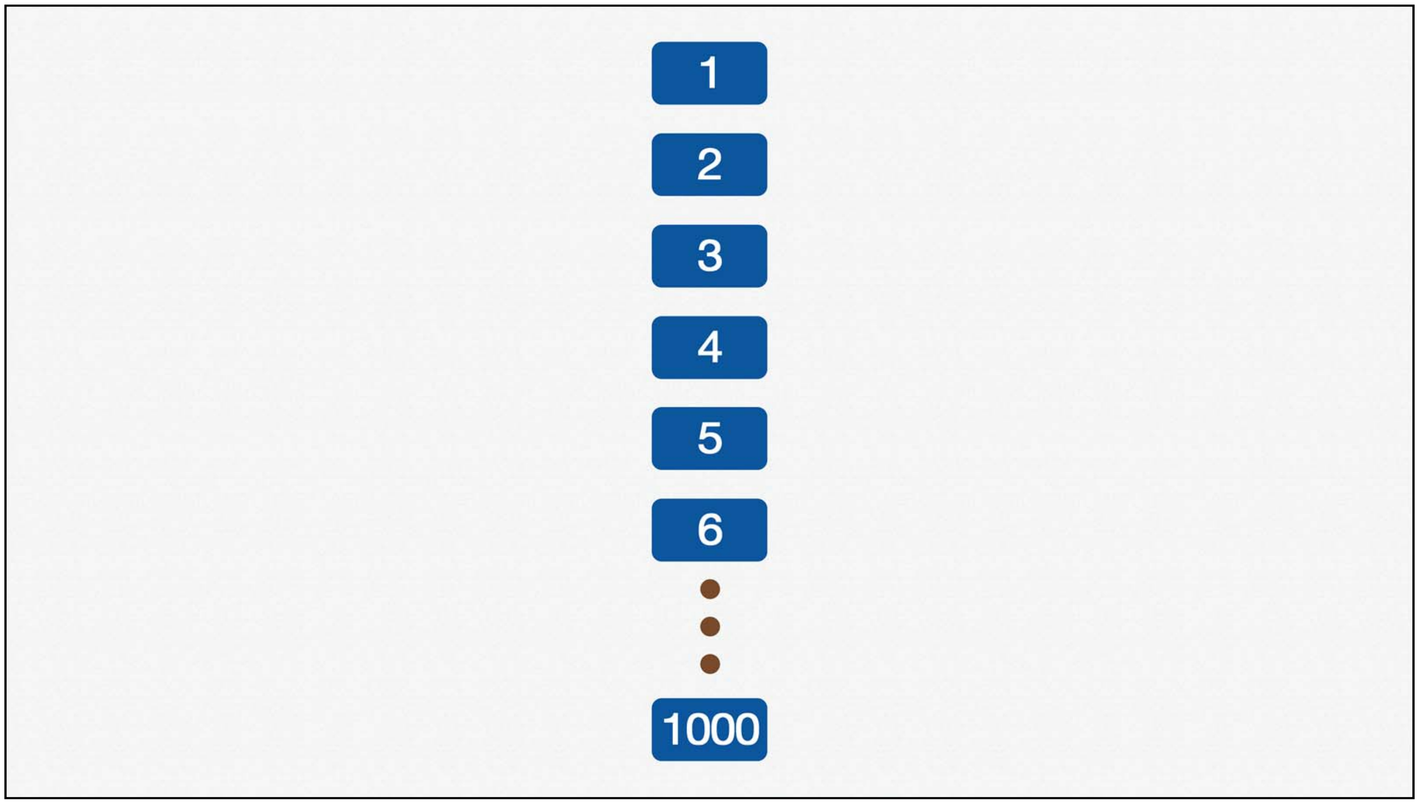
Using this formula, we propose the following algorithm. Our algorithm is a simple loop over a number of photon mapping iterations. The only specialty here is how we determine the kernel radius. In every iteration, just as in standard photon mapping, we perform the first pass by tracing photons and storing them in a photon map. Then in the second pass, for every traced eye ray, we determine a reference radius, and compute the radius of the current iteration using the explicit formula just shown before. This radius is then used to estimate the radiance with a range query.

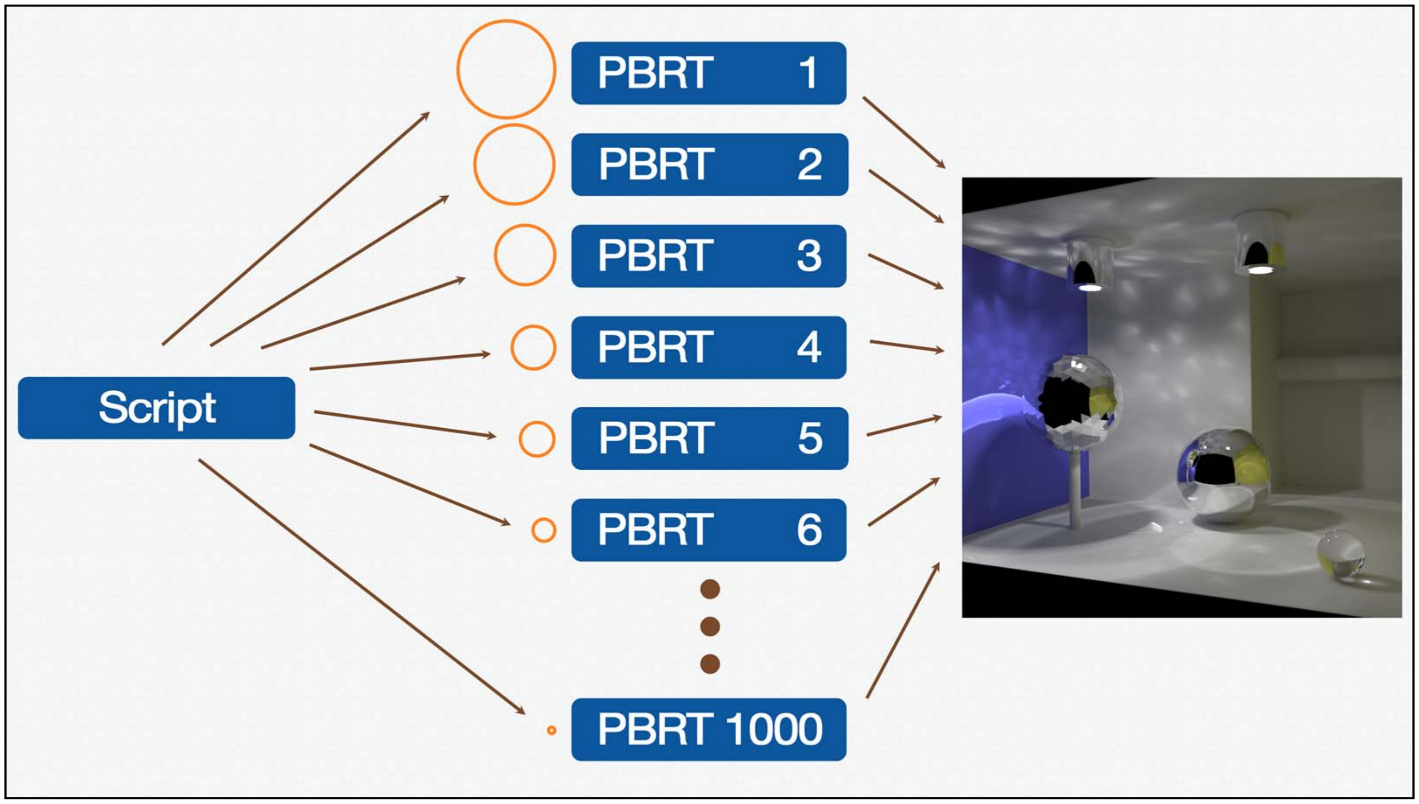


There are different strategies to define the reference radius. The simplest solution is to define this reference radius globally. In this case, the implementation becomes much simpler; we can factor out the entire radius sequence and pass it as a parameter to the photon mapper, effectively treating it as a black box. Simultaneously, our progressive photon mapping algorithm collapses to a script.

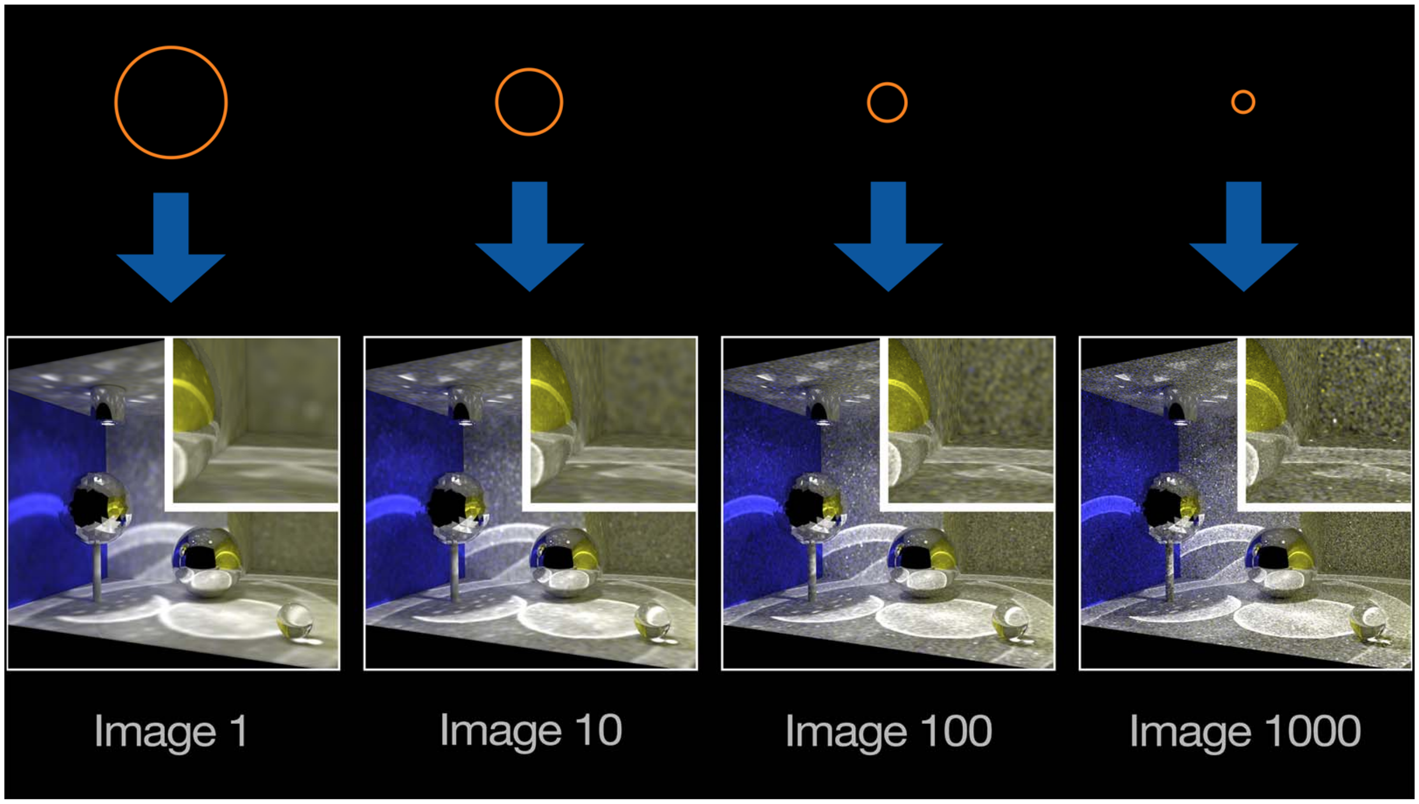


In the original progressive photon mapping, the iterations were dependent on each other, because local statistics had to be carried over from one iteration to the next. Our iterations, on the other hand, are independent of any statistics and can therefore be executed in parallel.

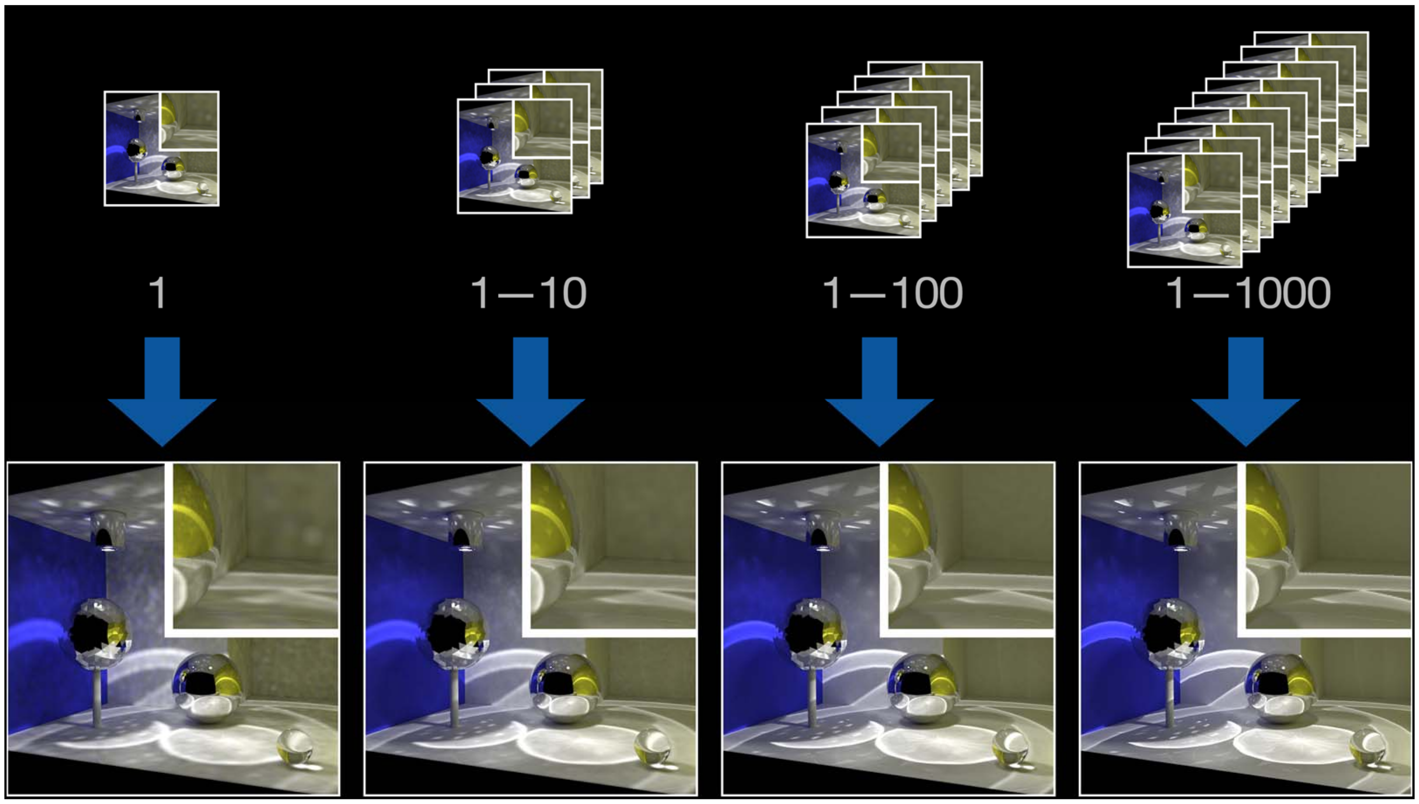

1
2
3
4
5
6
⋮
1000



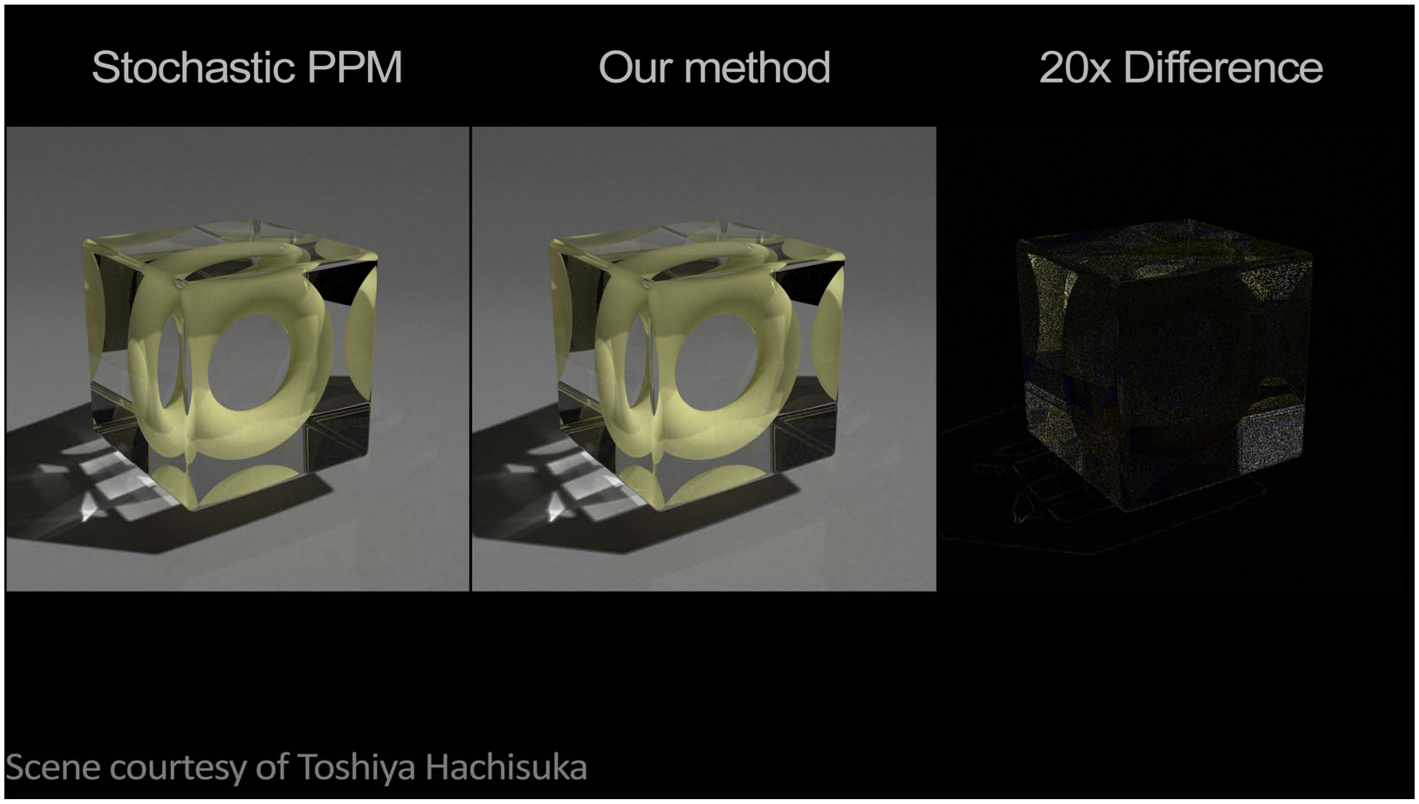
As a proof of concept, we have written a script to drive PBRT as a black box and to execute on a cluster. Every PBRT instance was fed with a radius from our radius sequence. The resulting images were then averaged on a single machine.



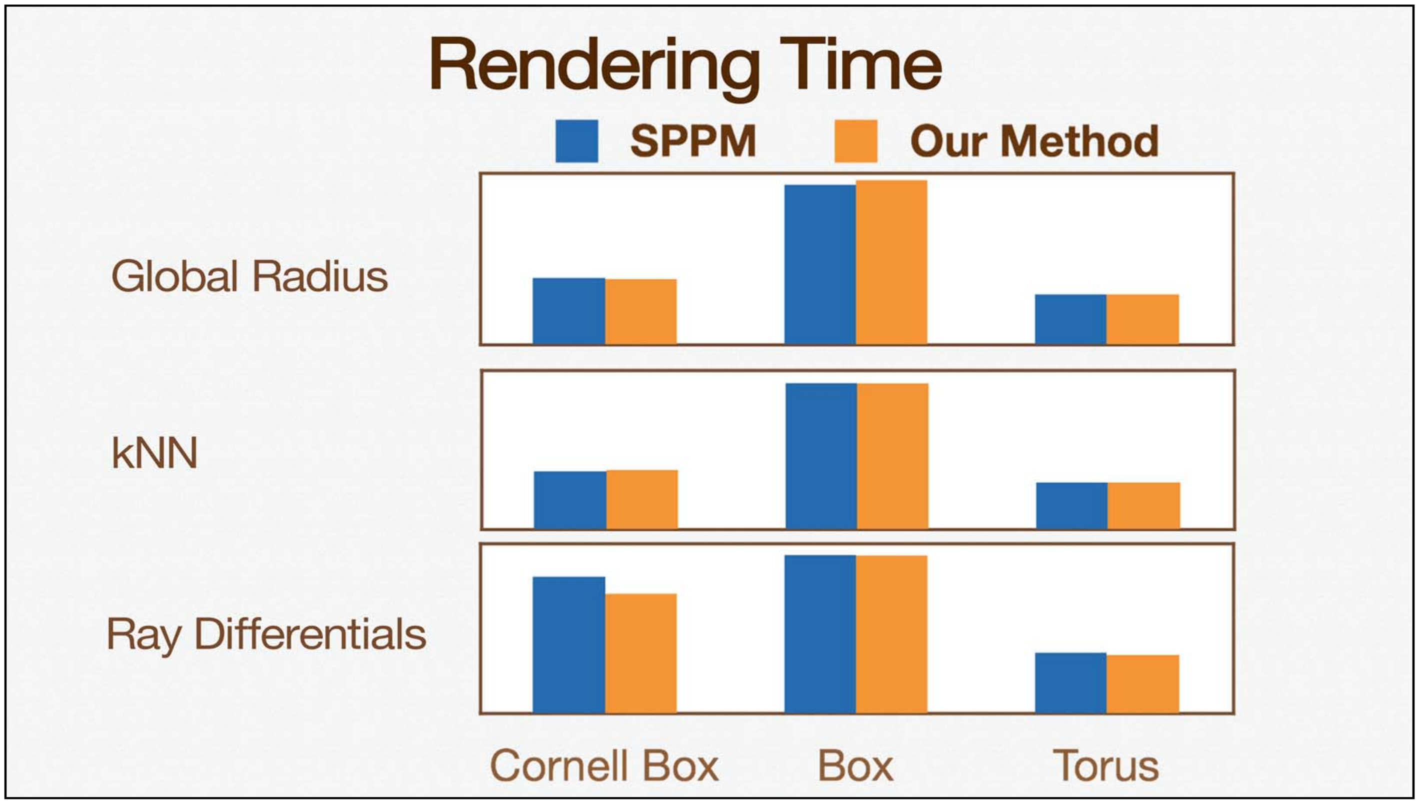
Here we see the results. As expected, the rendered images are noisier when the kernel radius is smaller.



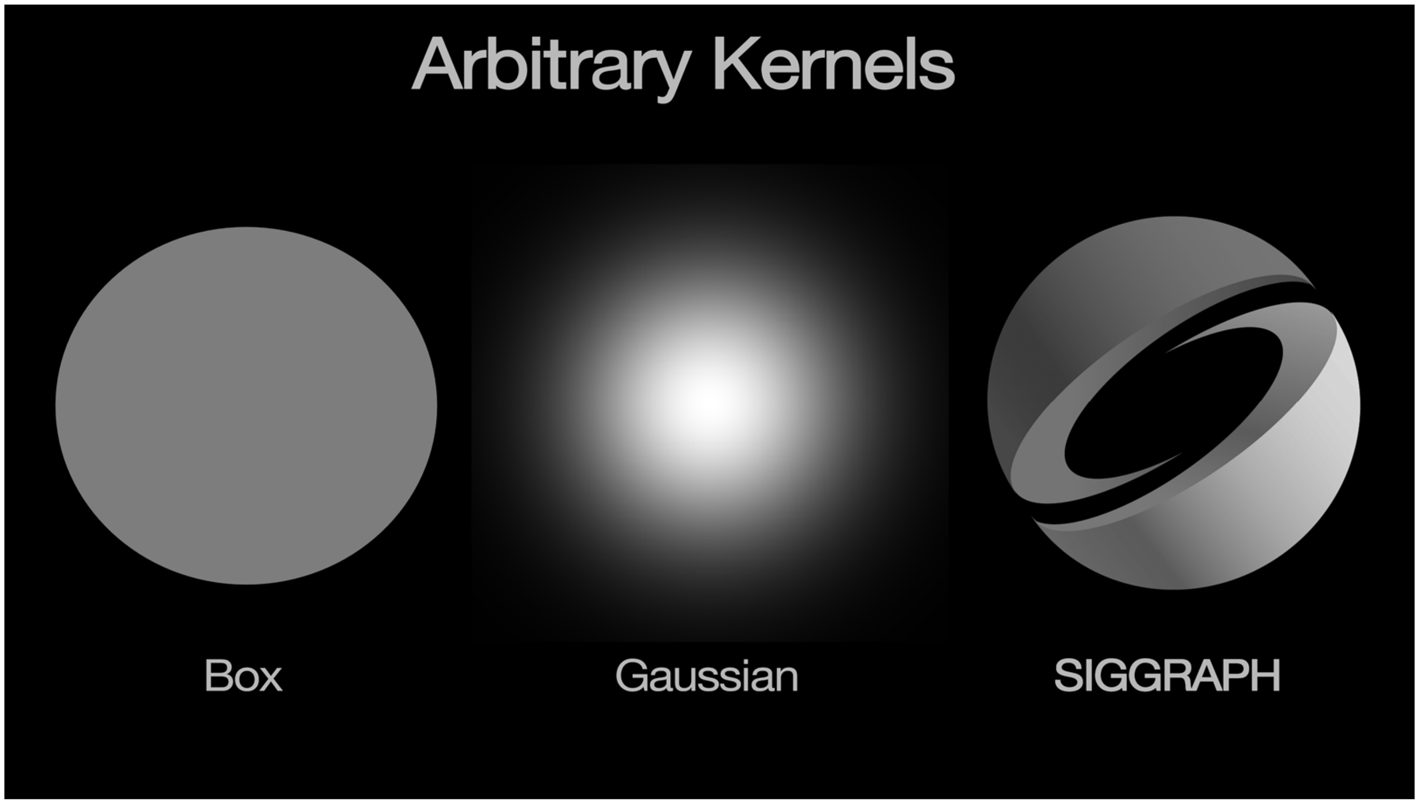
But let's see what happens when we average over the images. Here, we see just the first image. Now, the first 10 images averaged; already much less noise. Now, averaged over first 100 images. And finally, over 1000 images without visible noise or bias.



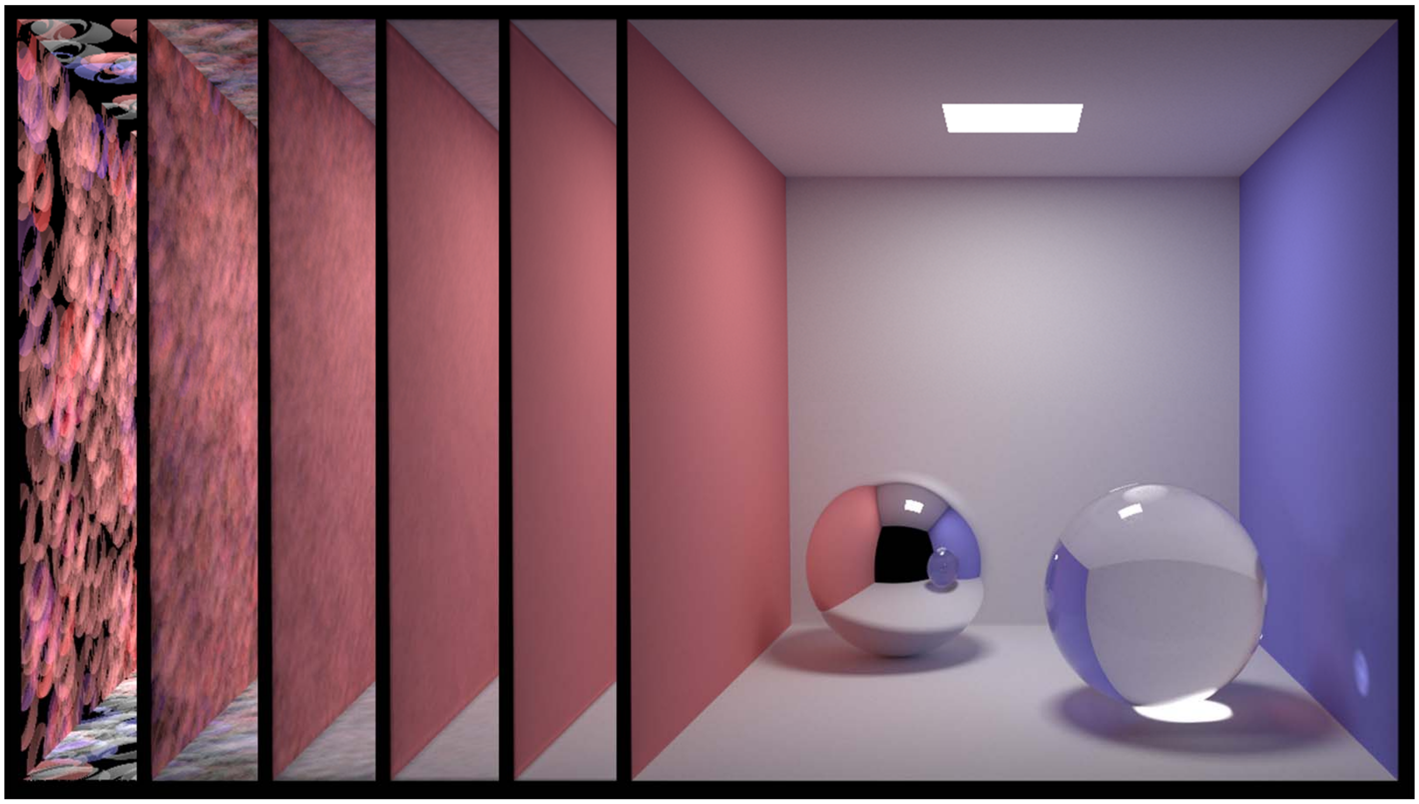
Quality-wise, our method stands on equal foot with traditional progressive photon mapping. Here we used a global reference radius for both methods. The difference between using traditional progressive photon mapping and our method is only in the noise.



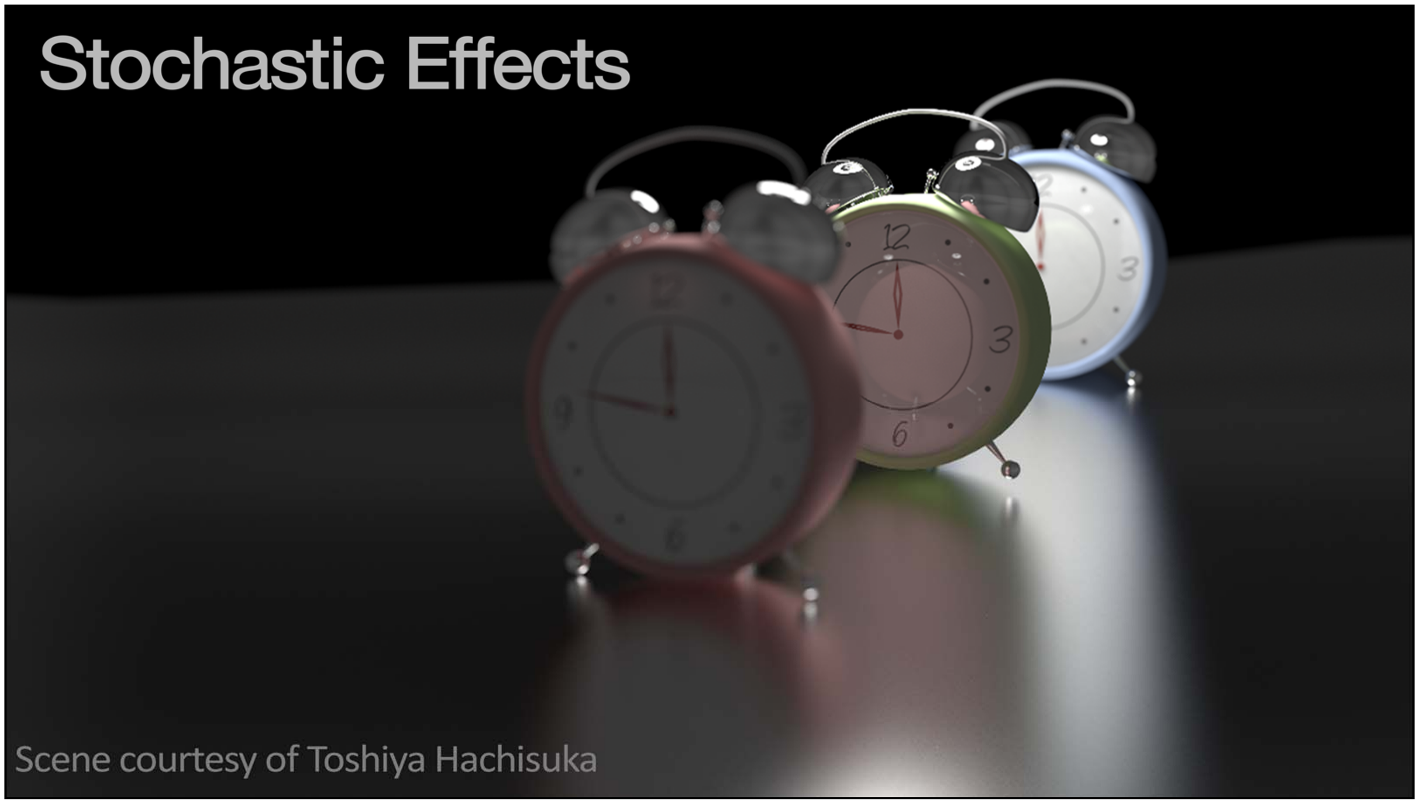
Performance is also the same. We compare three reference radius strategies, global radius, k-nearest-neighbors, and ray differentials. In all cases, the difference in rendering time is negligible. This is not so surprising since the overhead for both methods is minimal.



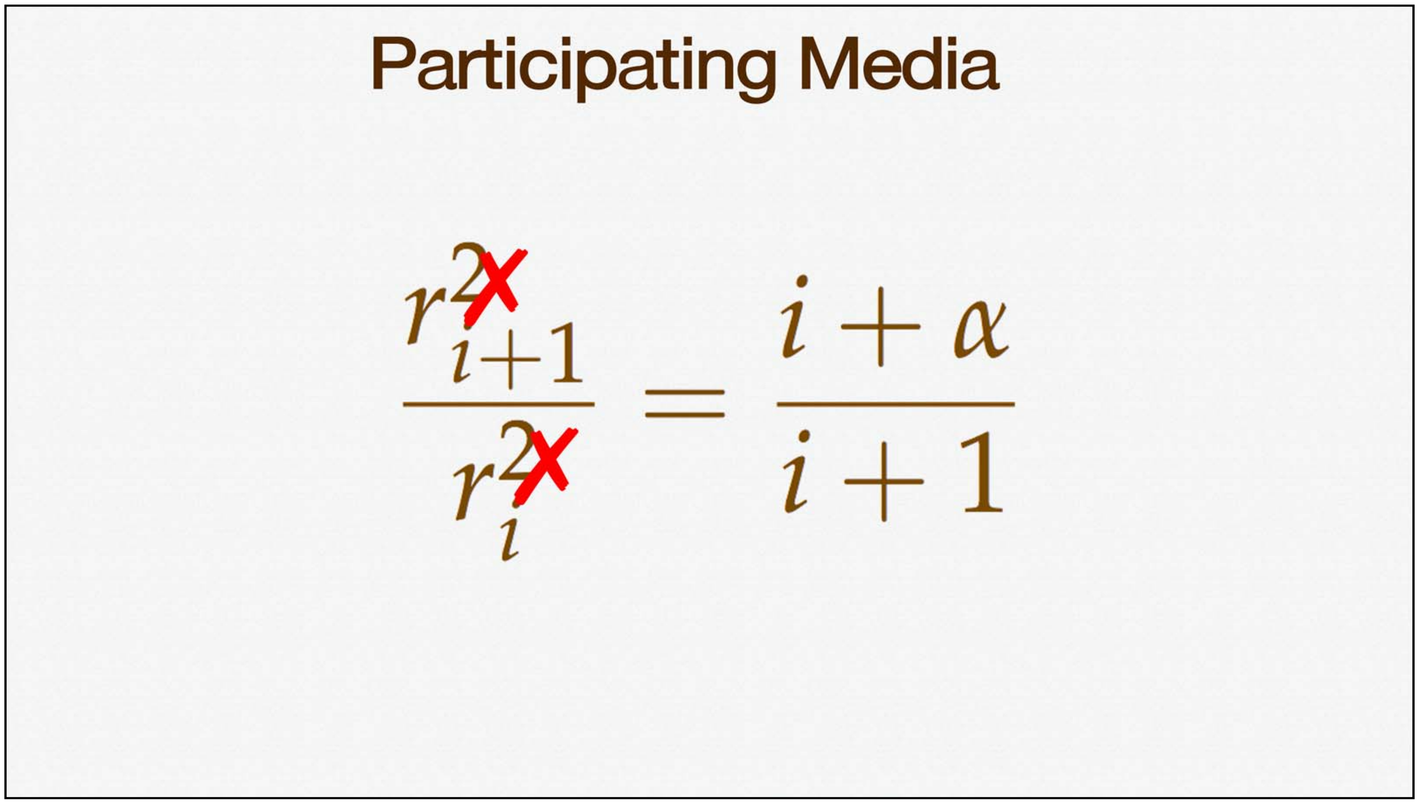
Our reformulation of progressive photon mapping is more general. We can use arbitrary kernels for radiance estimation. Let’s see what happens if we use the right most kernel.



Here, we use only one thousand photons per iteration to show the kernel. Well, even in this case, after many iterations, we observe that the image sequence converges.



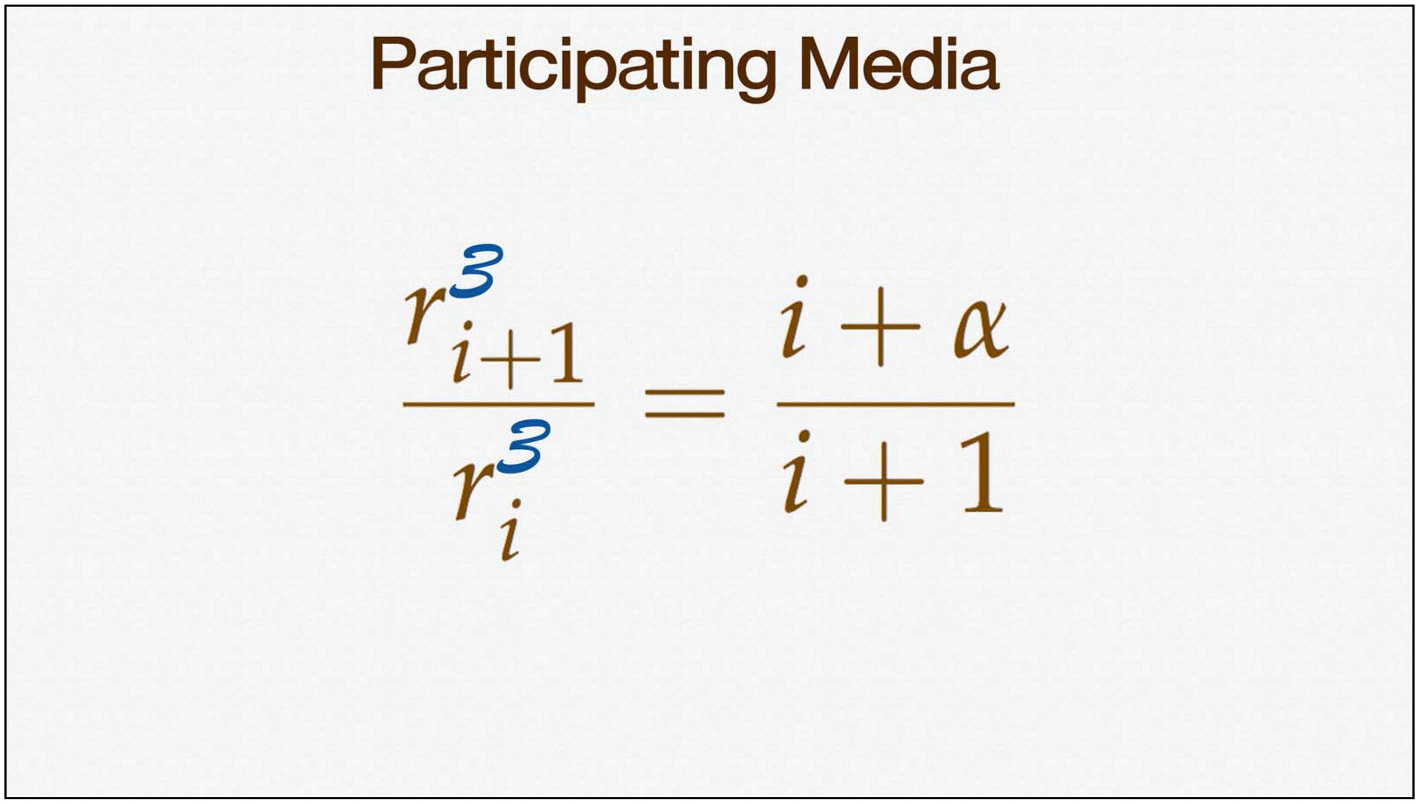
Just like stochastic PPM, our method includes stochastic effects as well. We demonstrate here depth of field and glossy surfaces.



Participating Media

$$\frac{r_{i+1}^{\cancel{2}}}{r_i^{\cancel{2}}} = \frac{i+\alpha}{i+1}$$

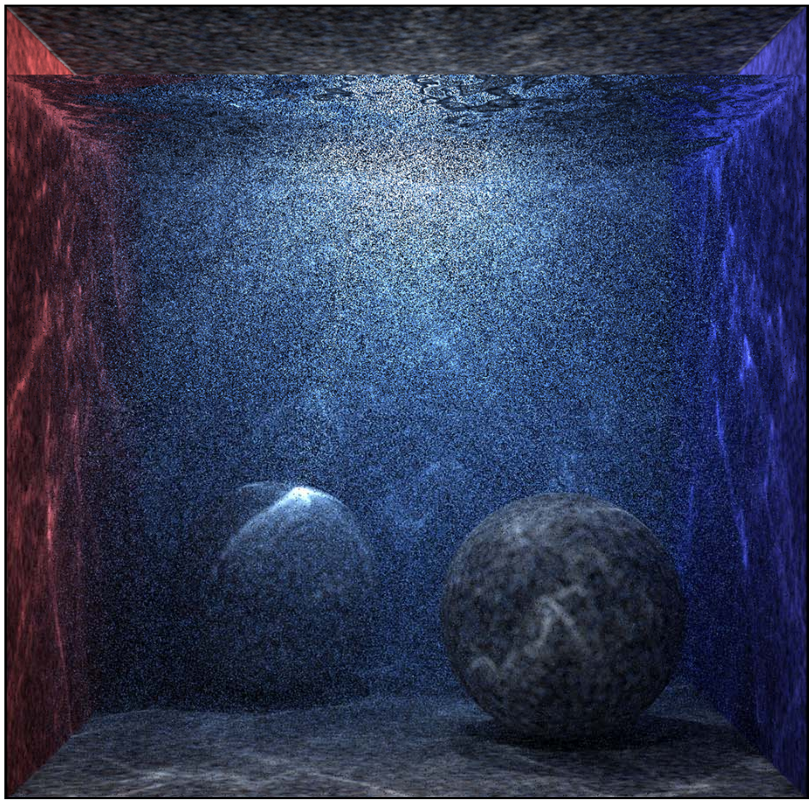
Last not least, our derivation of radius sequence and analysis extends to participating media as well. In volumetric photon mapping, the radiance is not defined on surfaces, but in volume. The kernel used for radiance estimate becomes therefore three dimensional.



Participating Media

$$\frac{r_{i+1}^{3}}{r_{i}^{3}} = \frac{i+\alpha}{i+1}$$

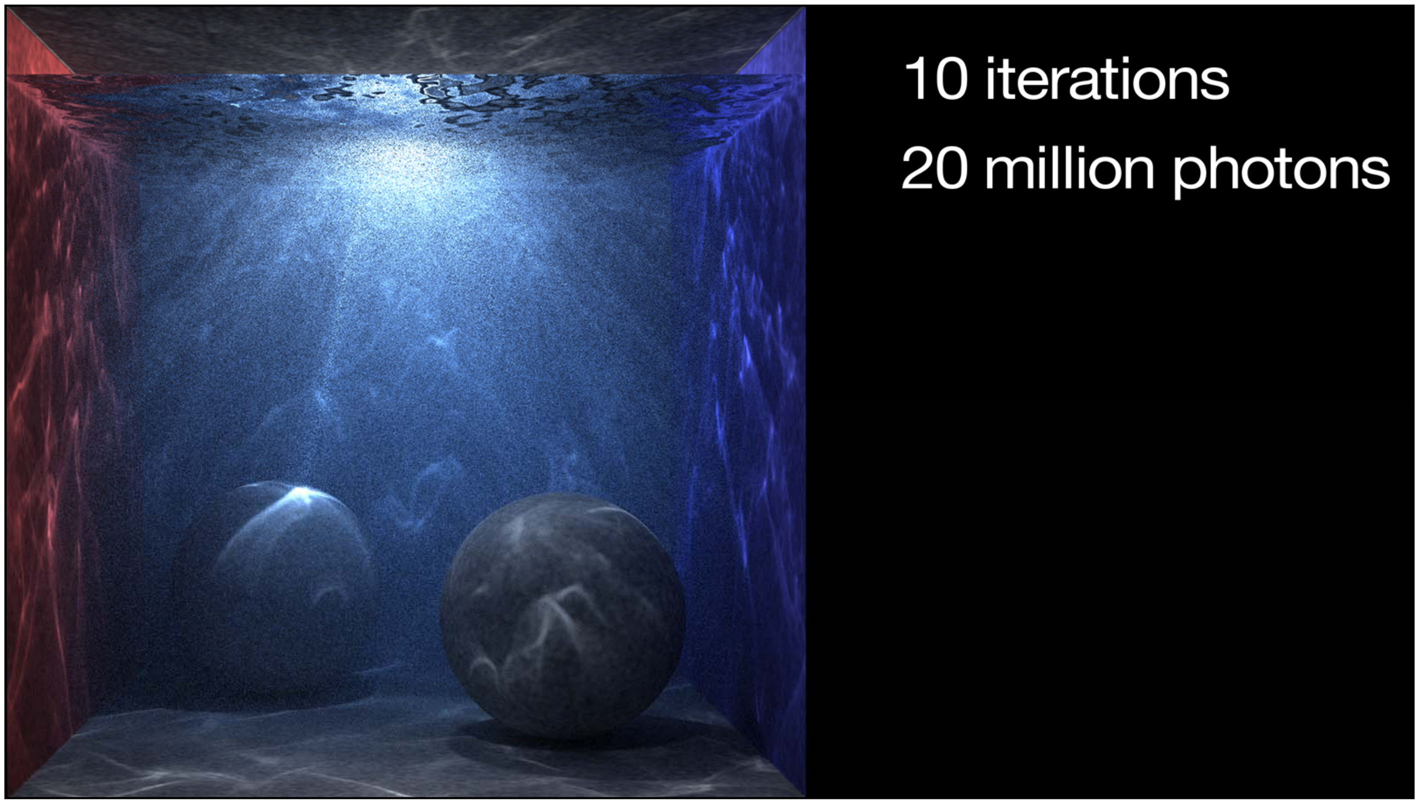
Following our analysis, we only have to replace in the radius sequence the exponents of the radii with 3, allowing us to integrate over a three dimensional domain.



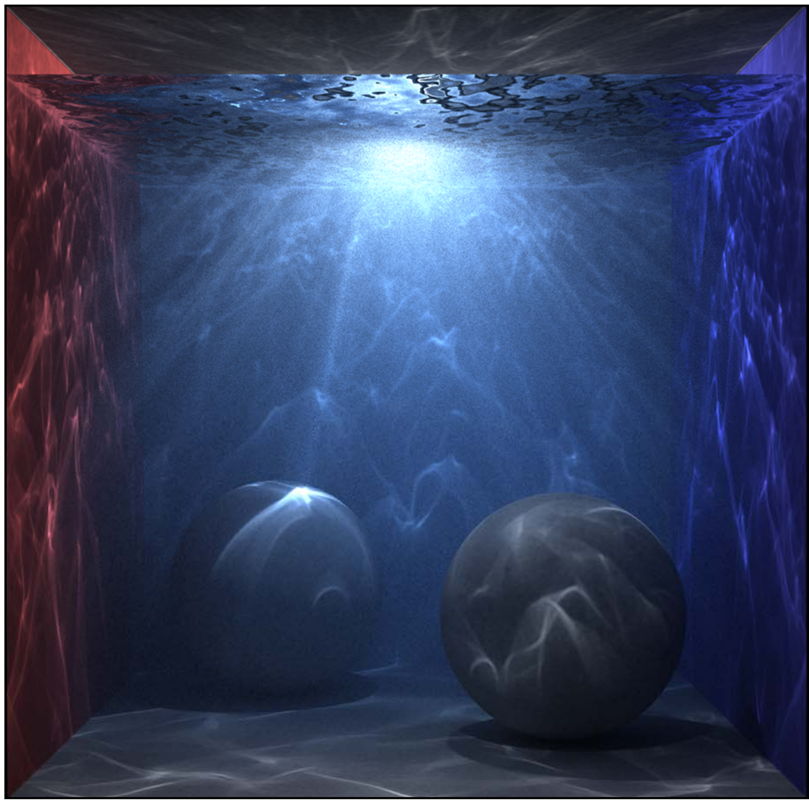
1 iteration

2 million photons

Here is a an example. This is a cornell box filled with water. At the beginning, with only 2 million photons, we only see noise.



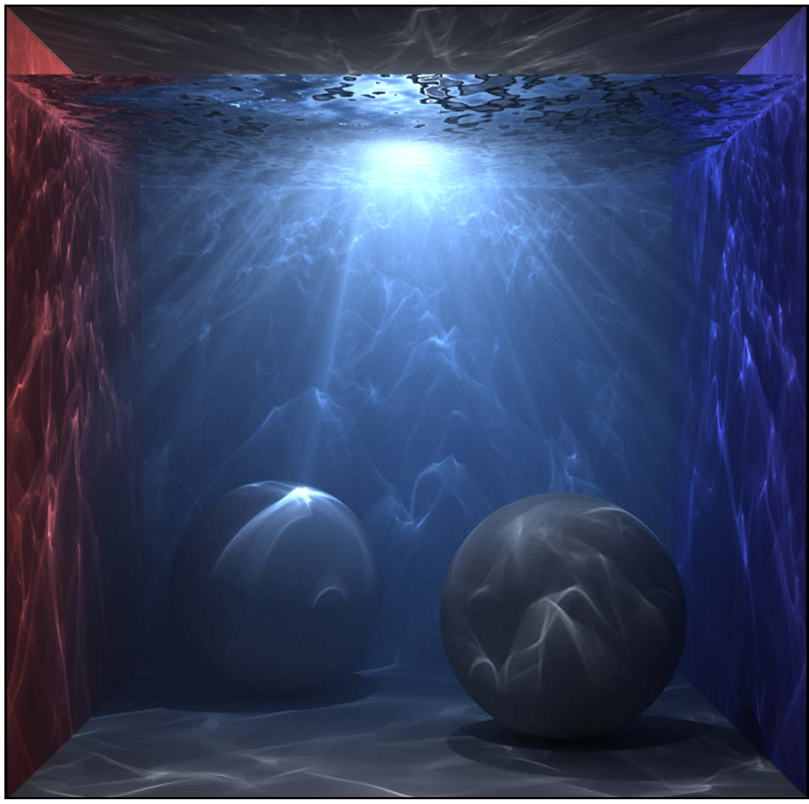
But if we shoot enough photons



100 iterations

200 million photons

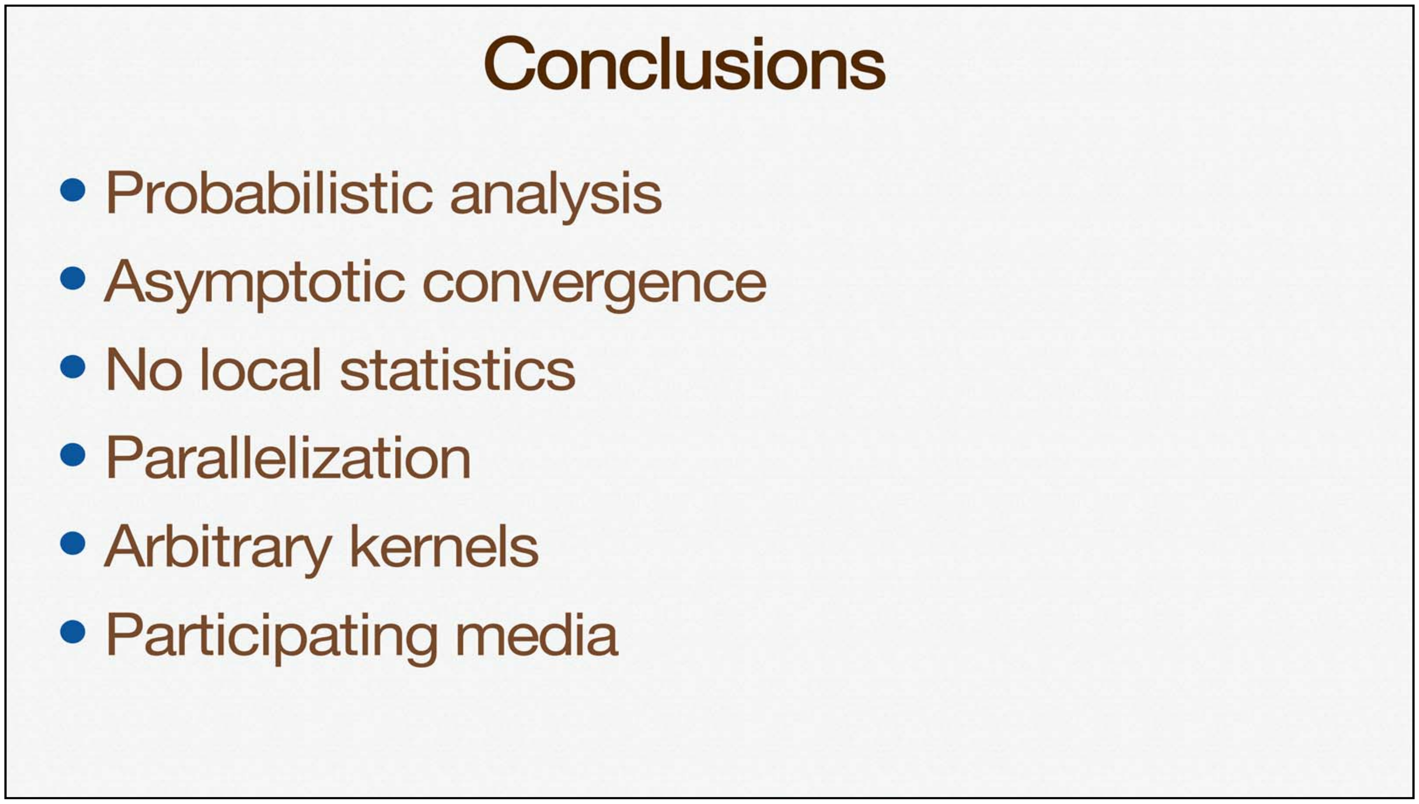
we start seeing god rays.



1000 iterations

2 billion photons

And eventually, after 1000 iterations and 2 billion photons, sharp caustics appear and the noise is gone.



Conclusions

- Probabilistic analysis
- Asymptotic convergence
- No local statistics
- Parallelization
- Arbitrary kernels
- Participating media

Let me review our results. We provided a probabilistic analysis and asymptotic convergence of progressive photon mapping. Most importantly, it leads to simpler implementations, which in the simplest case amounts to writing a script and using a standard photon mapper as a black box. Our method has the advantage of being parallelizable allowing the use of clusters. Furthermore, our reformulation

generalizes progressive photon
mapping to arbitrary kernels.
And finally, our method trivially extends
to other radiance estimates like
volumetric radiance estimates and beam
radiance estimates.

Acknowledgments

- ★ Reviewers
- ★ Hachisuka et al.
- ★ Swiss National Science Foundation

Lastly, I would like to thank the reviewers for providing valuable feedback. I would also like to mention that all this work would not have been possible without the preceding work by Hachisuka et al. who developed the ingenious progressive photon mapping. Thank you for your attention!