

SIGGRAPH Asia 2013

State of the Art in Photon Density Estimation

Course Notes

Toshiya Hachisuka

Aarhus University

Wojciech Jarosz

Disney Research Zürich

Iliyan Georgiev

Saarland University

Anton S. Kaplanyan

Karlsruhe Institute of Technology

Derek Nowrouzezahrai

Université de Montréal

Abstract

Photon density estimation techniques are a popular choice for simulating light transport in scenes with complicated geometry and materials. This class of algorithms can be used to accurately simulate inter-reflections, caustics, color bleeding, scattering in participating media, and subsurface scattering. Since its introduction, photon density estimation has been significantly extended in computer graphics with the introduction of: specialized techniques that intelligently modify the positions or bandwidths to reduce visual error using a small number of photons, approaches which eliminate error completely in the limit, methods that use higher-order samples and queries to reduce error in participating media, and recent generalized formulations that bridge the gap between photon density estimation and other techniques.

This course provides the necessary insight to implement all these latest advances in photon density estimation. The course starts out with a short introduction to photon density estimation using classical photon mapping, but the remainder of the two-part course provides hands-on explanations of the latest developments in this area by the experts behind each technique. The course will give the audience concrete and practical understanding of the latest developments in photon density estimation techniques that have not been presented in prior similar SIGGRAPH/SIGGRAPH Asia courses.

Contents

Abstract	i
Table of Contents	ii
About the Organizers	iii
About the Lecturers	iv
Acknowledgements	iv
Course Syllabus	vi
Introduction	vii
I Course Materials	1
1 Regular Photon Density Estimation	2
1.1 Photon Mapping Basics	3
2 Progressive Photon Density Estimation	37
2.1 Progressive Photon Mapping Basics	38
2.2 Progressive Photon Mapping & Extensions	82
2.3 Probabilistic Formulation of Photon Density Estimation	141
2.4 Adaptive Progressive Photon Mapping	167
3 Participating Media	189
3.1 Participating Media Basics	190
3.2 From Photons to Beams	270
4 Beyond Photon Density Estimation	419
4.1 Photon Relaxation	420
4.2 Progressive Expectation–Maximization	441
4.3 Combination with MC integration and Path Space Regularization	474

About the Organizers

Toshiya Hachisuka <toshiya@cs.au.dk>

Aarhus University

<http://cs.au.dk/~toshiya/>

Toshiya Hachisuka is an Assistant Professor in the Department of Computer Science at Aarhus University. His main research interests are the development of general light transport simulation algorithms and the intersection of computational statistics and realistic image synthesis. He has published multiple work on those topics including a new formulation of photon density estimation and a multidimensional adaptive sampling framework for ray tracing. He received his Ph.D. in Computer Science from University of California, San Diego in 2011 and B.Eng. from the University of Tokyo in 2006.

Wojciech Jarosz <wjarosz@disneyresearch.com>

Disney Research Zürich

<http://zurich.disneyresearch.com/~wjarosz>

Wojciech Jarosz is a Research Scientist in charge of the rendering group at Disney Research Zürich, and an adjunct lecturer at ETH Zürich. Prior to joining Disney, Wojciech obtained his Ph.D. (2008) and M.S. (2005) in computer graphics from UC San Diego, and his B.S. (2003) in computer science from the University of Illinois, Urbana-Champaign.

Wojciech's main research interest is realistic image synthesis and his publications explore practical applications in a variety of areas in computer graphics including: participating media; complex illumination and materials; global illumination; Monte Carlo methods and efficient sampling; and high-dynamic range imaging. His work in these areas has been incorporated into production rendering systems and used in the making of feature films, including Disney's *Tangled* (2010).

About the Lecturers

Iliyan Georgiev <georgiev@cs.uni-saarland.de>

Saarland University

<http://www.ilayan.com/>

Iliyan is a graphics researcher at Saarland University, Germany, pursuing a Ph.D. degree. He received a B.Sc. degree in computer science from Sofia University, Bulgaria, and a M.Sc. degree in computer science from Saarland University. His primary research interests are ray tracing and Monte Carlo methods for physically-based global illumination rendering.

Anton S. Kaplanyan <anton.kaplanyan@kit.edu>

Karlsruhe Institute of Technology

<http://cg.ibds.kit.edu/kaplanyan/>

Anton S. Kaplanyan is a graphics researcher and a PhD candidate at Karlsruhe Institute of Technology (KIT), Germany. His primary research and recent publications are about advanced light transport methods for global illumination. Prior to joining academia Anton had been working at Crytek at various positions ranging from senior R&D graphics engineer to lead researcher, leading the development of novel real-time rendering approaches, including the famous Light Propagation Volumes technique. He received his M.Sc. in Applied Mathematics at National Research University of Electronic Technology, Moscow in 2007.

Derek Nowrouzezahrai <derek@iro.umontreal.ca>

Université de Montréal

<http://www.iro.umontreal.ca/~derek/>

Derek is an Assistant Professor at the University of Montreal and Co-director of the Computer Graphics Research Group (www.ligum.umontreal.ca). Prior to joining UdeM, Derek was a Post-Doctoral Researcher at Disney Research Zürich, as well as working at Microsoft Research, Microsoft, Amazon.com, Electronic Arts and Research in Motion. Derek's research deals with realistic appearance modeling, interactive rendering, non-photorealistic rendering, physically-based animation, fluid simulation and geometry processing. For more information on Derek and his work, please refer to his website.

Acknowledgements

We are very grateful to past contributors to this course who have graciously allowed us to include their course notes and slides (Henrik Wann Jensen, Matthias Zwicker, Wenzel Jakob, and Ben Spencer) as well as all co-authors of the papers discussed in the course.

Section	Time [min]	Description (Presenter)	Subtotal [min]
Regular Photon Density Estimation:		5 Introduction (Hachisuka or Jarosz) 15 Photon Mapping Basics (Hachisuka or Jarosz)	15
Progressive Photon Density Estimation:		15 Progressive Photon Mapping - Basics (Hachisuka) 20 Progressive Photon Mapping - Extensions (Hachisuka) 15 Probabilistic PPM (Hachisuka) 15 Adaptive PPM (Kaplanyan)	65
	15 *Break*		
Participating Media:		15 Participating Media Basics (Jarosz) 20 From Photons to Beams (Jarosz) 15 Photon Beams in Tangled (Nowrouzezarhai)	50
Beyond Density Estimation		20 Photon Relaxation + Extensions (Hachisuka or Jarosz) 15 Progressive EM (Jarosz) 30 Unified Path Space/Vertex Merging (Georgiev), and Regularization (Kaplanyan)	65
Total (out of 3.75 hours = 225 minutes)	220	5 Conclusions (Hachisuka or Jarosz)	

Figure 1: Session timing breakdown for the course.

Course Syllabus

Photon mapping techniques have progressed significantly in the last few years. The first part will include a brief introduction to light transport and the core ideas behind photon mapping. The second part will start with a brief introduction to recent advances in photon density estimation for surfaces, followed by details of its extensions. The third part will cover recent developments in volumetric photon mapping and its application. The last portion of the course will explain recent work that connect photon density estimation with other approaches such as density estimation techniques outside graphics, blue noise sampling, and Monte Carlo ray tracing. A detailed breakdown is provided in Figure 1.

Introduction

Many fields require visually accurate recreations of the real world. For example, computer-generated images used for product design need to be as visually close as possible to the real world correspondences. Architects need accurate ways to pre-visualize the appearance of buildings under different lighting configurations before construction. Movies, advertisements, and video games also need believable computer-generated images. Computer simulation of how light interacts with virtual shapes and materials, *light transport simulation*, is a natural approach to achieve this result.



Figure 2: Light transport simulation via photon density estimation in a complex architectural model. The sun is the only light source and most of the light is due to indirect bounces.

Among many different approaches of light transport simulation, one of the most popular is photon density estimation. The basic algorithm is composed of two passes. The first pass distributes packets

of light energy, called *photons*, using a light transport simulation starting from light sources, and the second pass computes illumination at locations within the scene by estimating the local photon density. This class of methods poses light transport as a density estimation problem. Figure 2 shows one example of light transport simulation using photon density estimation. Since its introduction to computer graphics, photon density estimation techniques have been used in many practical applications. At the same time, the original techniques have been significantly expanded, especially in the past several years.

Early Photon Density Estimation Methods

The earliest application of photon density estimation was Arvo’s backward ray tracing method [Arv86]¹. Arvo’s approach first distributes packets of light energy across the scene using photon tracing. It then computes the local density using the histogram method, by counting the number of photons in discrete bins on surfaces, to estimate the lighting intensity.

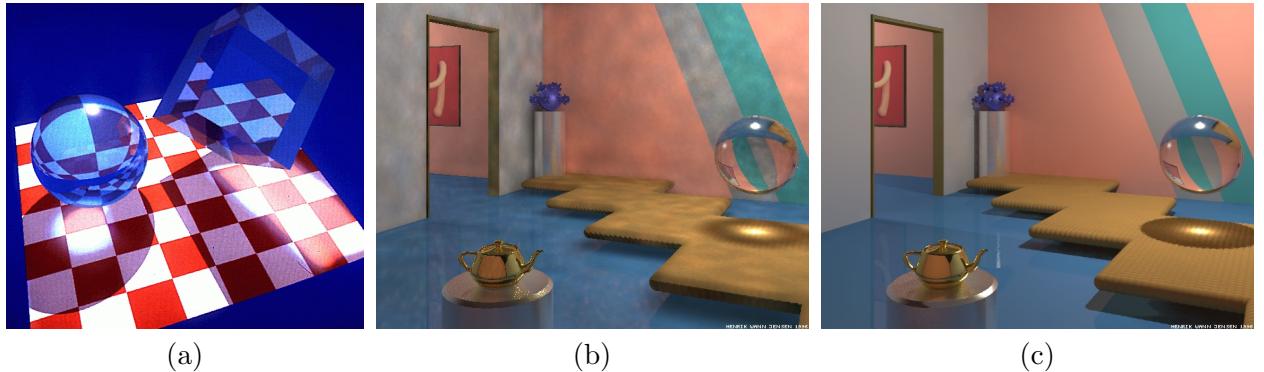


Figure 3: Rendered images with early photon density estimation techniques: (a) Caustics rendered by backward ray tracing [Arv86]. (b) Direct visualization of photon density estimation using photon mapping. (c) Rendered image using final gathering and the photon map in (b) [Jen96].

Since then, photon density estimation has attracted great interests from the research community due to its generality. For example, Shirley et al. [SWH⁺95] applied a photon density estimation method to compute a view-independent solution of light transport simulation on a triangle mesh. Perhaps one of the most successful photon density estimation algorithms is photon mapping [Jen96]. Photon mapping uses kNN density estimation and was the first to use photon density estimation for view-independent rendering. Photon mapping also popularized the concept of the two pass approach to photon density estimation. Examples from these early techniques are shown in Figure 3.

¹The word “backward” comes from the fact that traditional ray tracing [Whi80] traces rays from the eye, whereas backward ray tracing traces rays from light sources.

Improved Photon Density Estimation Methods

Initial applications of photon density estimation used standard techniques such as kNN kernel density estimation and the histogram method (a review of existing density estimation methods is available in the book by Hastie et al [HTF01]). More recent techniques have proposed novel density estimation methods customized for photon density estimation. Figure 4 shows some examples of such improvements. One such technique is a density estimator of segments of light transport paths, rather than the endpoints of segments (i.e., photons), which reduces bias in photon density estimation [LURM02].

Schjøth et al. applied the concept of ray differentials [Ige99] to lights transport paths for photons [SFES07]. The idea is to compute an approximated footprint of a photon by considering a photon path as a beam of light with finite thickness. The estimated footprints are used to determine appropriate bandwidths for photon density estimation. This concept was later extended to handle the temporal footprints of photons [SFES10].

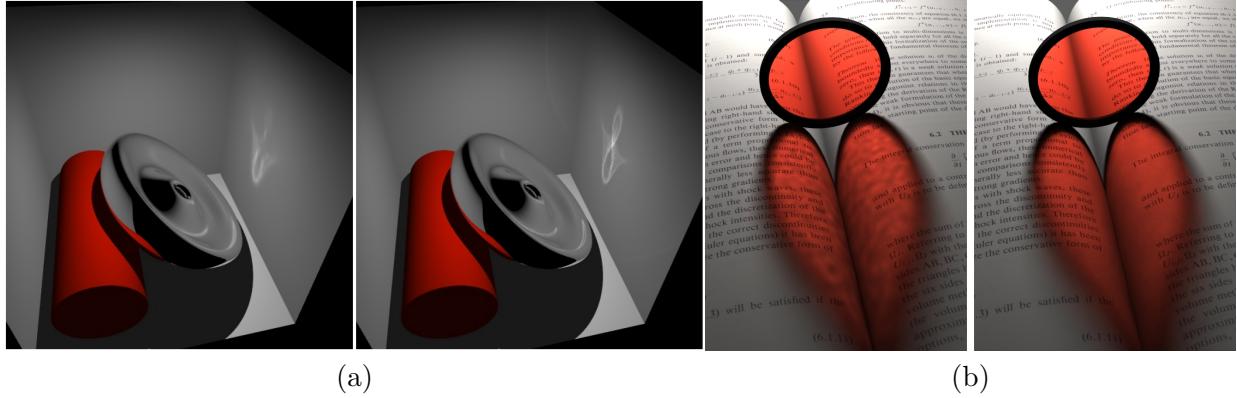


Figure 4: Results of improved photon density estimation methods: (a) Images rendered with (right) and without (left) photon differentials [SFES07]. (b) Images rendered with (right) and without (left) photon relaxation [SJ09].

For particular types of light transport paths such as caustics, noise in the photon distribution leads directly to artifacts in the rendered image. For example, even if photons are distributed according to a uniform distribution, with a finite number of photons, the randomness of photon locations makes the estimated density deviate from a constant intensity. Photon relaxation [SJ09] significantly reduces such artifacts by moving photons into a target distribution with lower discrepancy. The same authors [SJ13a] recently introduced a more stable approach with novel parameterization of photons.

Progressive Photon Density Estimation

One fundamental limitation in the standard photon density estimation algorithm is that we need to store all photons before we perform the density estimation pass. This limits the quality of the final result based on the amount of available storage. In contrast, the quality of light transport simulation based on Monte Carlo path integration is only limited by the computation time.

Hachisuka et al. [HOJ08] introduced a new framework of density estimation, called progressive density estimation, which completely removes this limitation. In progressive density estimation, the result is progressively refined as new photons samples are generated and the density estimate converges to the correct solution without storing photons. The quality of the progressive density estimate is only limited by the computation time, just like Monte Carlo path integration; however, unlike Monte Carlo path integration, progressive photon density estimation is robust to specular-diffuse-specular light transport (Figure 5).

The basic progressive density estimation was extended to incorporate distribution ray tracing effects such as depth of field and motion blur [HJ09]. Knaus and Zwicker later showed a different formulation of progressive density estimation [KZ11]. This new formulation is easier to implement than the original formulation, using conventional photon density estimation as a black box, and also supports distribution ray tracing effects. Figure 6 highlights some results from those papers. More recently, photon relaxation was incorporated into progressive photon density estimation [SJ13b]. The proposed technique also produces a view independent solution of progressive photon density estimation which allows us to change the viewpoint without computing illumination from scratch.

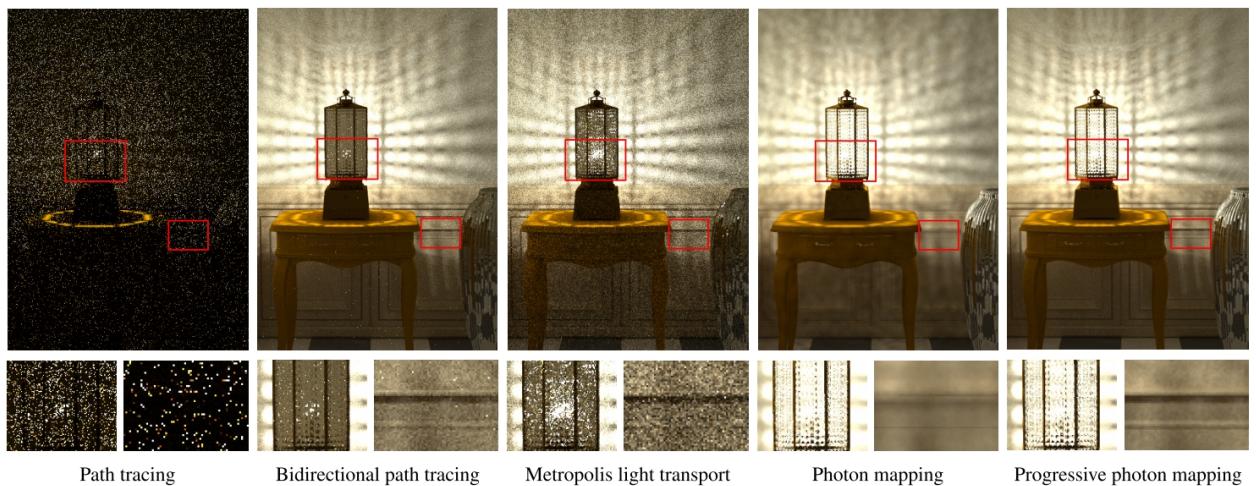


Figure 5: Light transport simulation of a glass lamp using different rendering methods [HOJ08].



Figure 6: Complex light transport simulation with distribution ray tracing effects: (a) Alarm clocks illuminated by a desk lamp with depth of field [HJ09]. (b) Rendering of diamonds with dispersions and depth of field [KZ11].

Rendering Participating Media using Photon Density Estimation

Photon density estimation is not just for solving light transport between surfaces. Volumetric photon mapping [JC98] is an extension of the original photon mapping method which can handle light transport within participating media such as smoke and fog. One recent improvement on this topic is the beam radiance estimate [JZJ08]. The idea is to replace the costly ray marching process in volumetric photon mapping by a single density estimation of photons along a line of sight (i.e., the “beam” of the eye path). Later, Jakob et al. [JRJ11] proposed an accelerated expectation–maximization technique to fit a compact gaussian mixture model to the underlying photon distribution. This results in a practical algorithm for photon density estimation in participating media which incorporates elements of photon relaxation and level-of-detail within a single framework. Figure 7 shows some results from those papers.

Jarosz et al. introduced a generalized theory of volumetric density estimation using points or beams [JNSJ11]. In particular, they introduced the concept of photon beams which replaces photons by segments of photon paths within participating media and significantly improves the quality of photon density estimation for participating media. Recently, the concept of photon beam was combined with progressive photon density estimation [JNT⁺11]. Like progressive photon mapping, the algorithm is robust to specular-diffuse-specular light transport and provably converges to the correct solution (Figure 8).

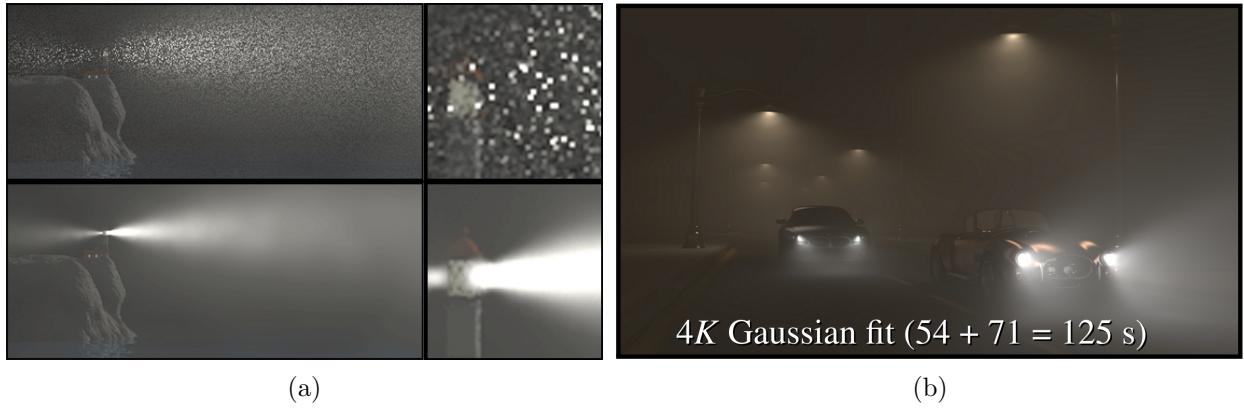


Figure 7: Rendered images of participating media using photon density estimation: (a) Comparison of original volumetric photon mapping (top) and the bream radiance estimate (bottom) [JZJ08]. (b) The result of fitting 4K Gaussians using progressive EM [JRJ11].

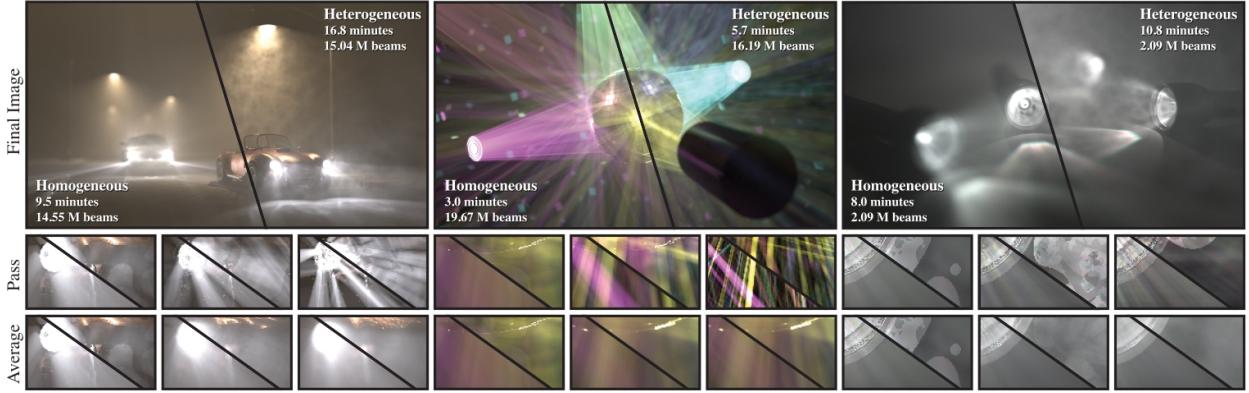


Figure 8: Rendered images using progressive photon beams for both homogeneous and heterogeneous media [JNT⁺11]. We generate a sequence of independent render passes (middle) where we progressively reduce the photon beam radii.

Rendering Systems using Photon Density Estimation

The combination of original photon mapping and final gathering has been widely used in many different commercial rendering systems. The basic approach is to use the result of photon mapping as a rough solution of global illumination, and refine this solution by another step that performs additional gathering of radiance by tracing one-bounce rays – called a final gathering step. This approach is used within Pixar’s Photorealistic RenderMan® software to incorporate global illumination in movie production [CB04]. Recent versions of the software include improvements in photon mapping tailored to more general global illumination and participating media rendering.

Recent developments in photon density estimation have also started making their ways into other rendering systems. LuxRender [Lux12], one of the open source rendering systems with a large

user base, is currently incorporating progressive photon mapping as a new feature. The photon beams method was also recently used for the production of the Disney’s animated film *Tangled* with extensions to allow artist-driven design of volumetric effects [NJS⁺11].

Connections between Photon Density Estimation and Other techniques

Recently, multiple researchers revealed connections between photon density estimation and other techniques. Gerorgiev et al. [GKDS12] and Hachisuka et al. [HPJ12] concurrently introduced unified frameworks that allows us to combine photon density estimation and Monte Carlo ray tracing. Traditionally, these two approaches have been considered completely different and users are forced to choose one approach over the other for rendering. These new frameworks clarify how these approaches can be unified by multiple importance sampling based on either contraction or expansion of the path space. Figure 9 shows one of the results.

Kaplanyan and Dachsbacher [KD13b] generalized formulation of a traditional photon density estimation technique as spatial blurring of the rendering equation. This generalization, *path space regularization*, introduces another possibility of using directional blurring of the rendering equation in combination of existing Monte Carlo ray tracing. The same authors also established a connection of progressive photon density estimation to recursive estimation and regression in statistics and derived the optimal estimation parameters for the asymptotic convergence [KD13a].



Figure 9: Comparisons with Monte Carlo ray tracing, photon density estimation, and the combination via the unified framework [HPJ12]. The combined approach results in the most accurate solution with the same computation time.

Bibliography

- [Arv86] James Arvo. Backward ray tracing. In *In ACM SIGGRAPH '86 Course Notes, Developments in Ray Tracing*, pages 259–263, 1986. viii
- [CB04] Per H. Christensen and Dana Batali. An irradiance atlas for global illumination in complex production scenes. In Alexander Keller and Henrik Wann Jensen, editors, *Proceedings of the 15th Eurographics Workshop on Rendering Techniques, Norkping, Sweden, June 21-23, 2004*, pages 133–142. Eurographics Association, 2004. xii
- [GKDS12] Iliyan Georgiev, Jaroslav Křivánek, Tomáš Davidovič, and Philipp Slusallek. Light transport simulation with vertex connection and merging. *ACM Trans. Graph.*, 31(6):192:1–192:10, November 2012. xiii
- [HJ09] Toshiya Hachisuka and Henrik Jensen. Stochastic progressive photon mapping. In *SIGGRAPH Asia '09: ACM SIGGRAPH Asia 2009 papers*, pages 1–8, New York, NY, USA, 2009. ACM. x, xi
- [HOJ08] Toshiya Hachisuka, Shinji Ogaki, and Henrik Jensen. Progressive photon mapping. *ACM Transactions on Graphics (SIGGRAPH Asia Proceedings)*, 27(5):Article 130, 2008. x
- [HPJ12] Toshiya Hachisuka, Jacopo Pantaleoni, and Henrik Wann Jensen. A path space extension for robust light transport simulation. *ACM Transactions on Graphics (TOG)*, 31(6):191, 2012. xiii
- [HTF01] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., New York, NY, USA, 2001. ix

- [Ige99] Homan Igehy. Tracing ray differentials. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '99, pages 179–186, New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co. ix
- [JC98] Henrik Jensen and Per Christensen. Efficient simulation of light transport in scenes with participating media using photon maps. In *Computer Graphics (Proceedings of SIGGRAPH 98)*, pages 311–320, New York, NY, USA, 1998. ACM Press. xi
- [Jen96] Henrik Jensen. Global illumination using photon maps. In Xavier Pueyo and Peter Schröder, editors, *Rendering Techniques '96*, Eurographics, pages 21–30. Springer-Verlag Wien New York, 1996. viii
- [JNSJ11] Wojciech Jarosz, Derek Nowrouzezahrai, Iman Sadeghi, and Henrik Wann Jensen. A comprehensive theory of volumetric radiance estimation using photon points and beams. *ACM Transactions on Graphics (Presented at ACM SIGGRAPH 2011)*, 30(1):5:1–5:19, January 2011. xi
- [JNT⁺11] Wojciech Jarosz, Derek Nowrouzezahrai, Robert Thomas, Peter-Pike Sloan, and Matthias Zwicker. Progressive photon beams. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH Asia 2011)*, 30(6), December 2011. xi, xii
- [JRJ11] Wenzel Jakob, Christian Regg, and Wojciech Jarosz. Progressive expectation–maximization for hierarchical volumetric photon mapping. *Computer Graphics Forum (Proceedings of EGSR 2011)*, 30(4), June 2011. xi, xii
- [JZJ08] Wojciech Jarosz, Matthias Zwicker, and Henrik Wann Jensen. The beam radiance estimate for volumetric photon mapping. *Computer Graphics Forum (Proceedings of Eurographics 2008)*, 27(2):557–566, April 2008. xi, xii
- [KD13a] Anton S. Kaplanyan and Carsten Dachsbacher. Adaptive progressive photon mapping. *ACM Transactions on Graphics*, 32(2):16:1–16:13, April 2013. xiii
- [KD13b] Anton S. Kaplanyan and Carsten Dachsbaucher. Path space regularization for holistic and robust light transport. *Computer Graphics Forum (Proc. of Eurographics 2013)*, 32(2), 2013. xiii
- [KZ11] Claude Knaus and Matthias Zwicker. Progressive photon mapping: A probabilistic approach. *ACM Trans. Graph.*, 30:25:1–25:13, May 2011. x, xi

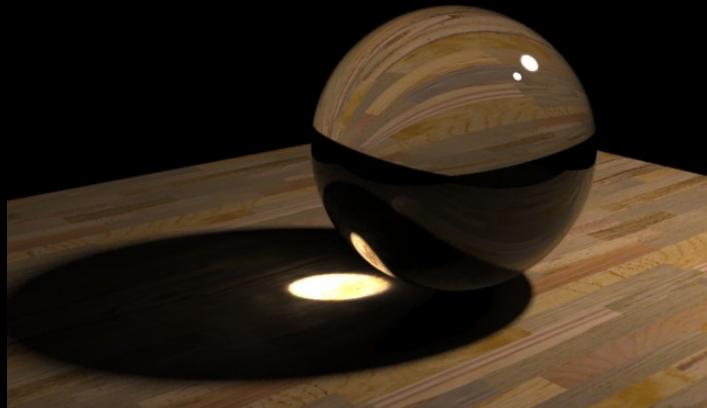
- [LURM02] M. Lastra, C. Urena, J. Revelles, and R. Montes. A particle-path based method for monte carlo density estimation. In *Rendering Techniques 2002 (Proceedings of the Thirteenth Eurographics Workshop on Rendering)*, June 2002. ix
- [Lux12] Luxrender. <http://www.luxrender.net>, http://www.luxrender.net/wiki/New_in_0-9, 2012. xii
- [NJS⁺11] Derek Nowrouzezahrai, Jared Johnson, Andrew Selle, Dylan Lacewell, Michael Kaschalk, and Wojciech Jarosz. A programmable system for artistic volumetric lighting. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2011)*, 30(4):29:1–29:8, August 2011. xiii
- [SFES07] Lars Schjøth, Jeppe Revall Frisvad, Kenny Erleben, and Jon Sporring. Photon differentials. In *Proceedings of the 5th international conference on Computer graphics and interactive techniques in Australia and Southeast Asia*, GRAPHITE ’07, pages 179–186, New York, NY, USA, 2007. ACM. ix
- [SFES10] Lars Schjøth, Jeppe Revall Frisvad, Kenny Erleben, and Jon Sporring. Temporal photon differentials. In *GRAPP’10*, pages 54–61, 2010. ix
- [SJ09] Ben Spencer and Mark W. Jones. Into the blue: Better caustics through photon relaxation. *Eurographics 2009, Computer Graphics Forum*, 28(2):319–328, March 2009. ix
- [SJ13a] Ben Spencer and Mark W Jones. Photon parameterisation for robust relaxation constraints. In *Computer Graphics Forum*, volume 32, pages 83–92. Wiley Online Library, 2013. ix
- [SJ13b] Ben Spencer and Mark W Jones. Progressive photon relaxation. *ACM Transactions on Graphics (TOG)*, 32(1):7, 2013. x
- [SWH⁺95] Peter Shirley, Bretton Wade, Philip Hubbard, David Zareski, Bruce Walter, and Donald Greenberg. Global illumination via density estimation. *Rendering Techniques 95 (Proceedings of Eurographics Workshop on Rendering 95)*, pages 219–230, 1995. viii
- [Whi80] Turner Whitted. An improved illumination model for shaded display. *Commun. ACM*, 23(6):343–349, 1980. viii

Course Materials

Regular Photon Density Estimation

Photon Mapping Basics

Photon Mapping



Henrik Wann Jensen

Computer Science and Engineering

University of California, San Diego

Motivation



Global Illumination

Motivation



Direct Illumination

Motivation

Before photon mapping

- Radiosity
 - ★ Mostly diffuse
 - ★ Mesh based lighting representation
- Monte Carlo path tracing
 - ★ Noisy
 - ★ Slow convergence

Motivation

$$L(x, \vec{\omega}) = L_e(x, \vec{\omega}) + \int_S f_r(x, \vec{\omega}, x' \rightarrow x) L(x' \rightarrow x) V(x, x') G(x, x') dA'$$

$$G(x, x') = \frac{(\vec{\omega}' \cdot \vec{n})(\vec{\omega}' \cdot \vec{n}')}{{|x' - x|}^2}$$

Motivation



Specular-Diffuse-Specular light transport is difficult

Basic Observation

- The lights are important
- The camera / eye is important

Photon Mapping

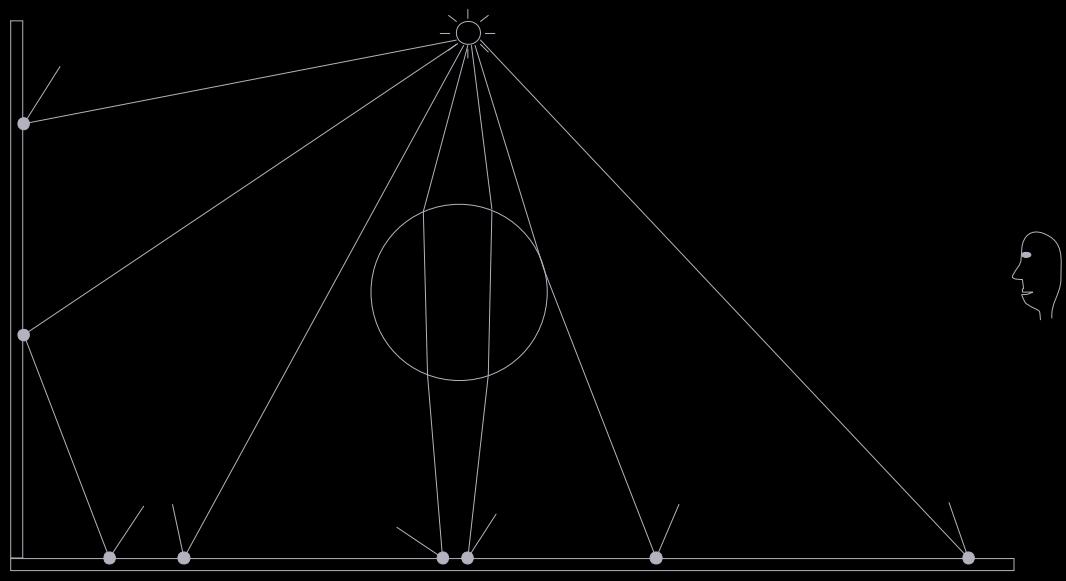
Photon Mapping

A two-pass method

Pass 1: Build the photon map (photon tracing)

Pass 2: Render the image using the photon map

Pass 1: Building the Photon Map



Photon Tracing

Photon Tracing

- Photon emission
- Photon scattering
- Photon storing

What is a photon?

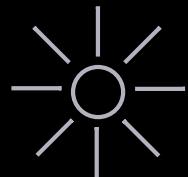
- Flux (power) - not radiance!
- Collection of physical photons
 - ★ A fraction of the light source power
 - ★ Several wavelengths combined into one entity

Photon emission

Given Φ Watt lightbulb.

Emit N photons.

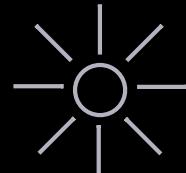
Each photon has the power $\frac{\Phi}{N}$ Watt.



- Photon power depends on the number of emitted photons.

Diffuse point light

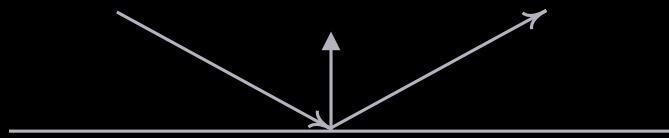
Generate random direction
Emit photon in that direction



```
// Find random direction
do {
    x = 2.0*random()-1.0;
    y = 2.0*random()-1.0;
    z = 2.0*random()-1.0;
} while ( (x*x + y*y + z*z) > 1.0 );
```

Photon scattering

Same as path tracing (with small exceptions).



$$\vec{d}_r = \vec{d}_i - 2\vec{n}(\vec{n} \cdot \vec{d}_i)$$

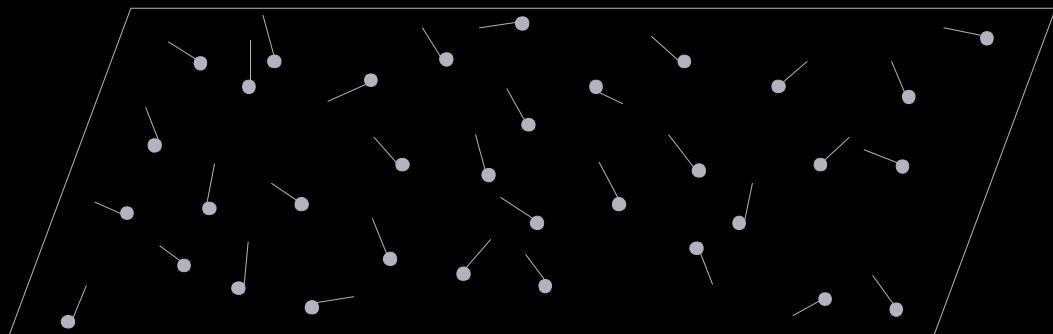
Photon scattering

Important optimizations:

- Use Russian roulette to avoid bias
- Create only one photon per scattering event

Photon storing

When a photon hits a non-specular surface information about it is stored in a global data structure called the photon map. The information includes the photon power, incoming direction and the hit location.

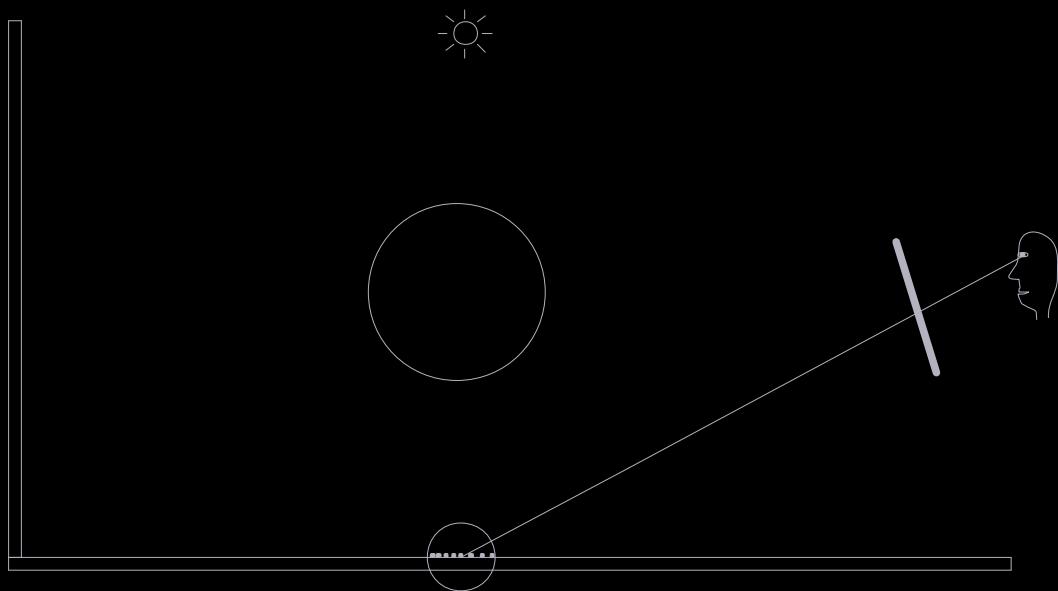


Photon map

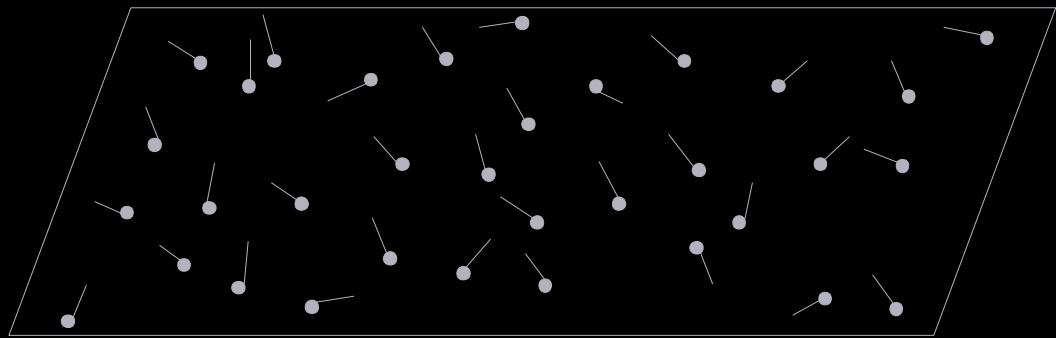
Pass 2: Rendering



Rendering with the photon map



Rendering with the photon map



We need radiance, but the photons carry flux

Radiance Estimate

$$L(x, \vec{\omega}) = \int_{\Omega} f_r(x, \vec{\omega}', \vec{\omega}) L'(x, \vec{\omega}') \cos \theta' d\omega$$

Radiance Estimate

$$\begin{aligned} L(x, \vec{\omega}) &= \int_{\Omega} f_r(x, \vec{\omega}', \vec{\omega}) L'(x, \vec{\omega}') \cos \theta' d\omega \\ &= \int_{\Omega} f_r(x, \vec{\omega}', \vec{\omega}) \frac{d\Phi^2(x, \vec{\omega}')}{d\omega \cos \theta' dA} \cos \theta' d\omega \end{aligned}$$

Radiance Estimate

$$\begin{aligned} L(x, \vec{\omega}) &= \int_{\Omega} f_r(x, \vec{\omega}', \vec{\omega}) L'(x, \vec{\omega}') \cos \theta' d\omega \\ &= \int_{\Omega} f_r(x, \vec{\omega}', \vec{\omega}) \frac{d\Phi^2(x, \vec{\omega}')}{d\omega \cos \theta' dA} \cos \theta' d\omega \\ &= \int_{\Omega} f_r(x, \vec{\omega}', \vec{\omega}) \frac{d\Phi^2(x, \vec{\omega}')}{dA} \end{aligned}$$

Radiance Estimate

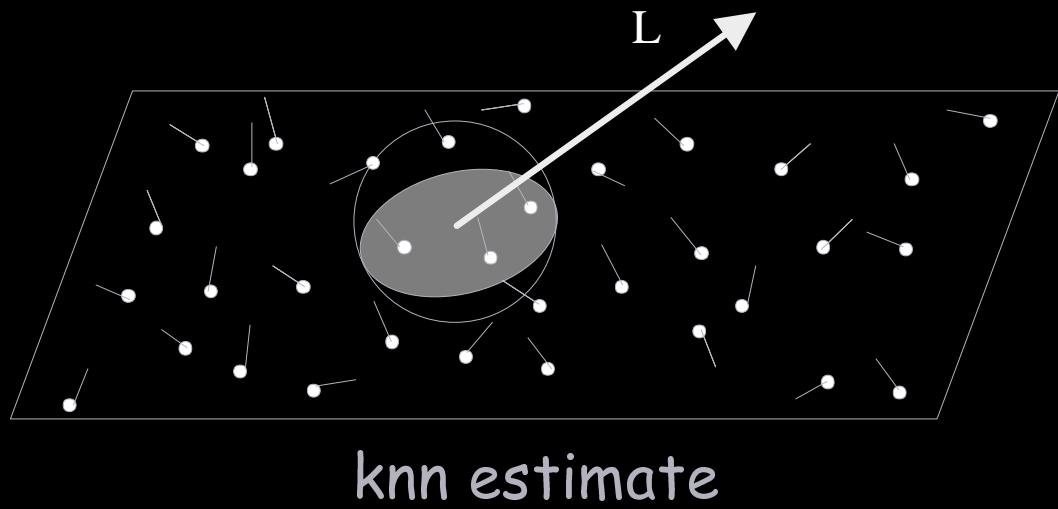
$$\begin{aligned} L(x, \vec{\omega}) &= \int_{\Omega} f_r(x, \vec{\omega}', \vec{\omega}) L'(x, \vec{\omega}') \cos \theta' d\omega \\ &= \int_{\Omega} f_r(x, \vec{\omega}', \vec{\omega}) \frac{d\Phi^2(x, \vec{\omega}')}{d\omega \cos \theta' dA} \cos \theta' d\omega \\ &= \int_{\Omega} f_r(x, \vec{\omega}', \vec{\omega}) \frac{d\Phi^2(x, \vec{\omega}')}{dA} \\ &\approx \sum_{p=1}^n f_r(x, \vec{\omega}'_p, \vec{\omega}) \frac{\Delta \Phi_p(x, \vec{\omega}'_p)}{\Delta A} \end{aligned}$$

Radiance Estimate

$$\begin{aligned} L(x, \vec{\omega}) &= \int_{\Omega} f_r(x, \vec{\omega}', \vec{\omega}) L'(x, \vec{\omega}') \cos \theta' d\omega \\ &= \int_{\Omega} f_r(x, \vec{\omega}', \vec{\omega}) \frac{d\Phi^2(x, \vec{\omega}')}{d\omega \cos \theta' dA} \cos \theta' d\omega \\ &= \int_{\Omega} f_r(x, \vec{\omega}', \vec{\omega}) \frac{d\Phi^2(x, \vec{\omega}')}{dA} \\ &\approx \sum_{p=1}^n f_r(x, \vec{\omega}'_p, \vec{\omega}) \frac{\Delta\Phi_p(x, \vec{\omega}'_p)}{\Delta A} \end{aligned}$$

Photon density estimate

Radiance Estimate



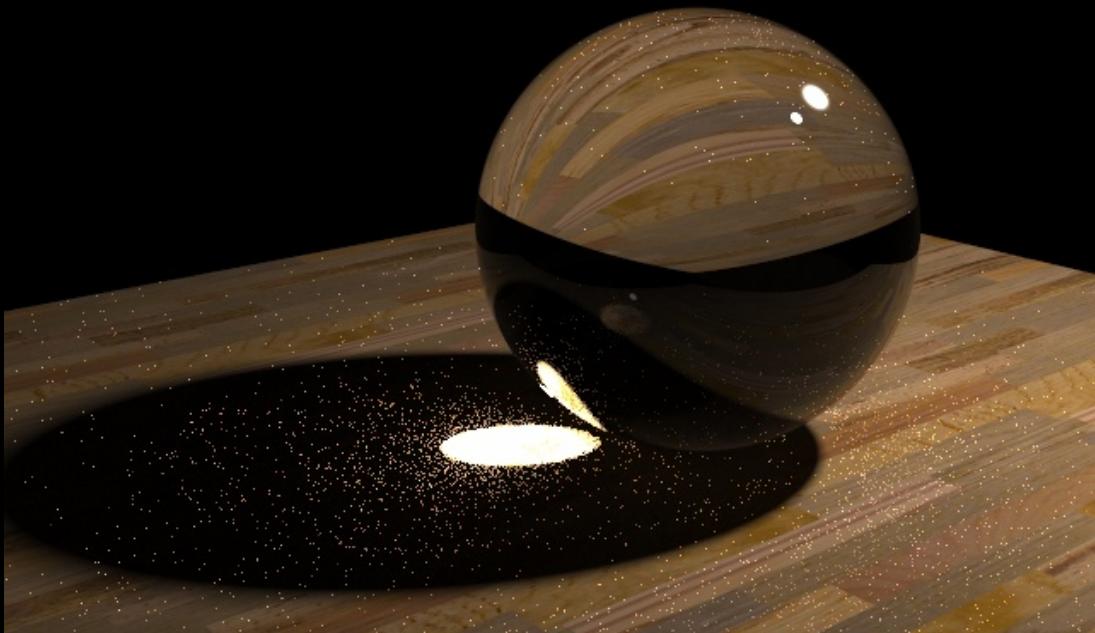
$$L(x, \vec{\omega}) \approx \sum_{p=1}^n f_r(x, \vec{\omega}'_p, \vec{\omega}) \frac{\Delta\Phi_p(x, \vec{\omega}'_p)}{\pi r^2}$$

Caustic from a Glass Sphere



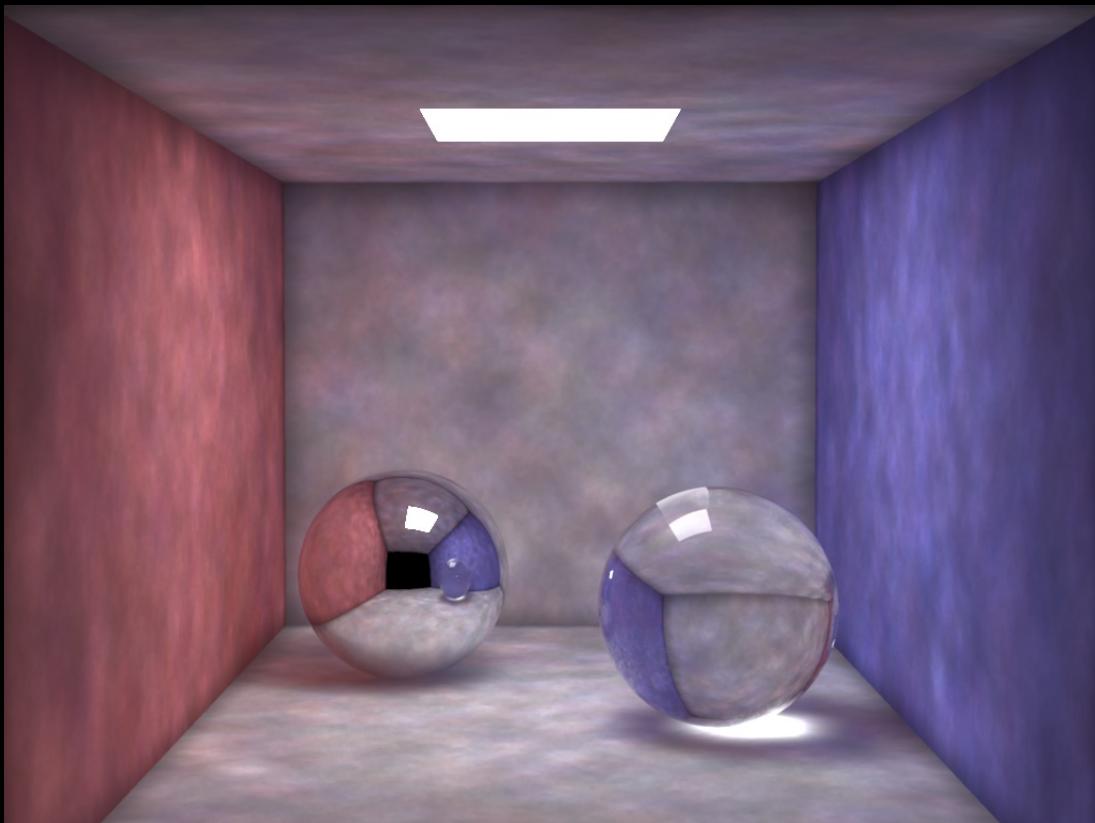
Photon Mapping: 10000 photons / 50 photons in radiance estimate

Caustic from a Glass Sphere



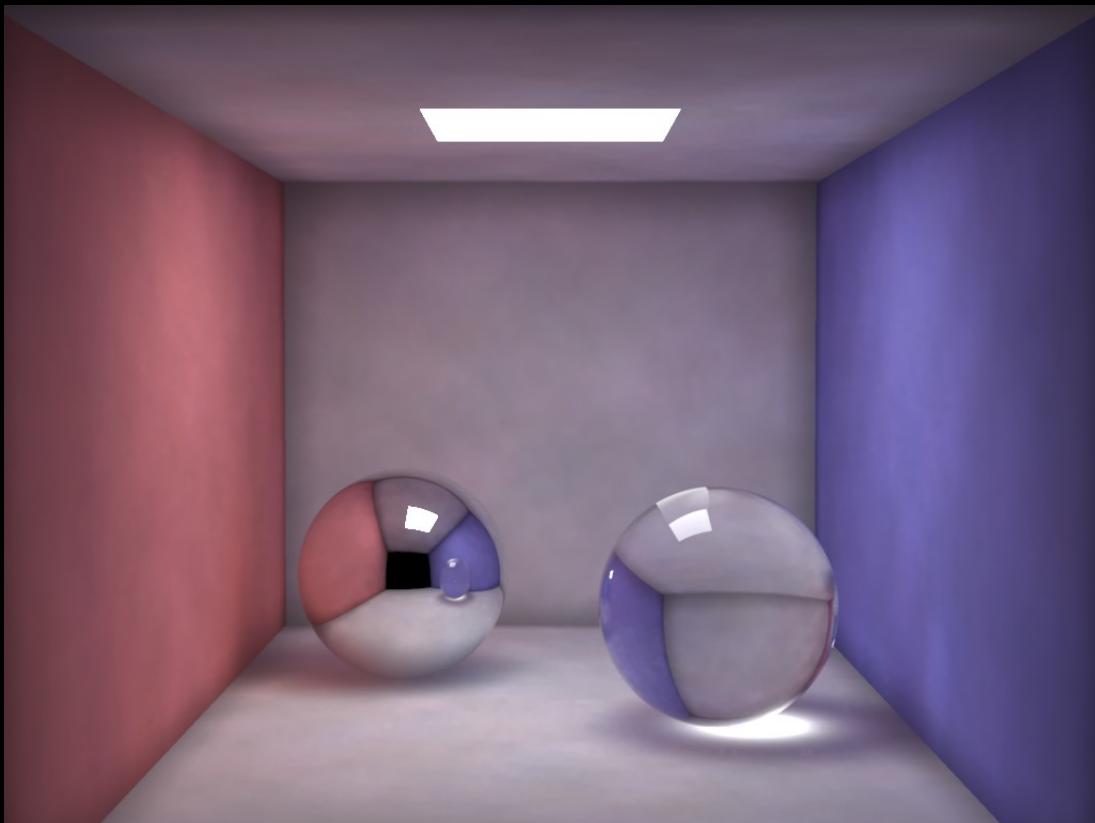
Path Tracing: 1000 paths/pixel

Global Illumination



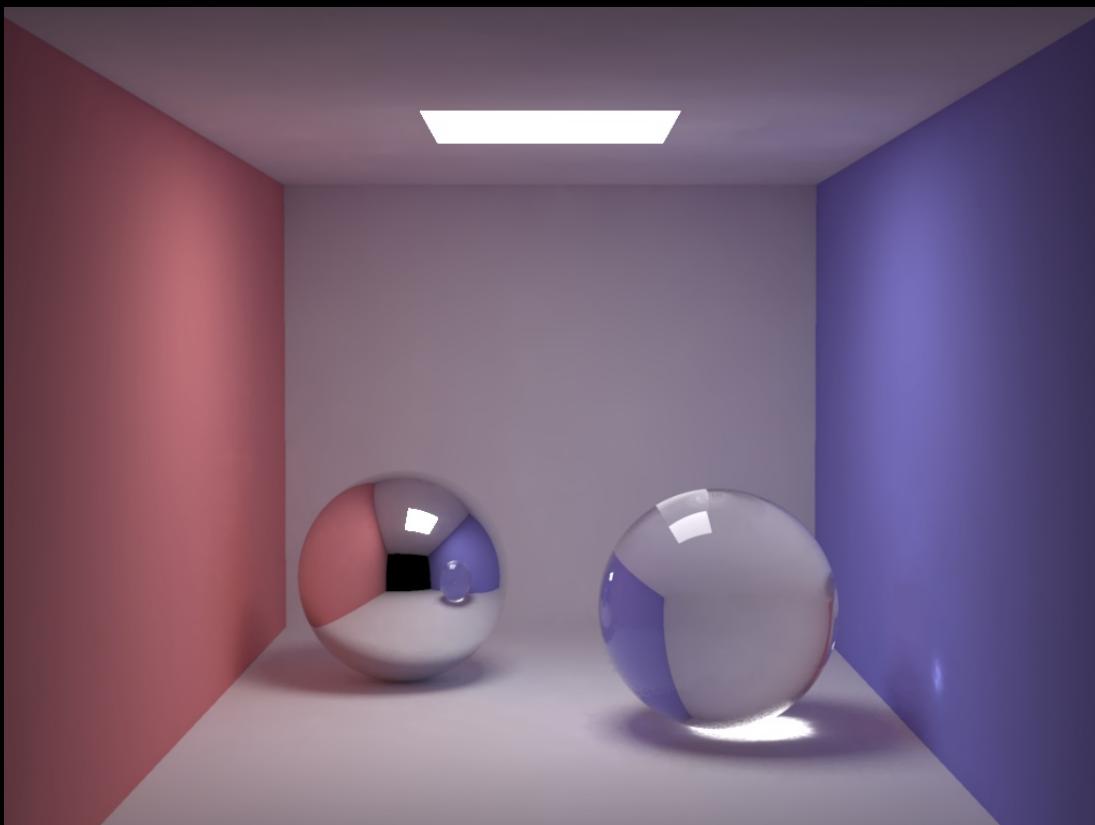
100000 photons / 50 photons in radiance estimate

Global Illumination



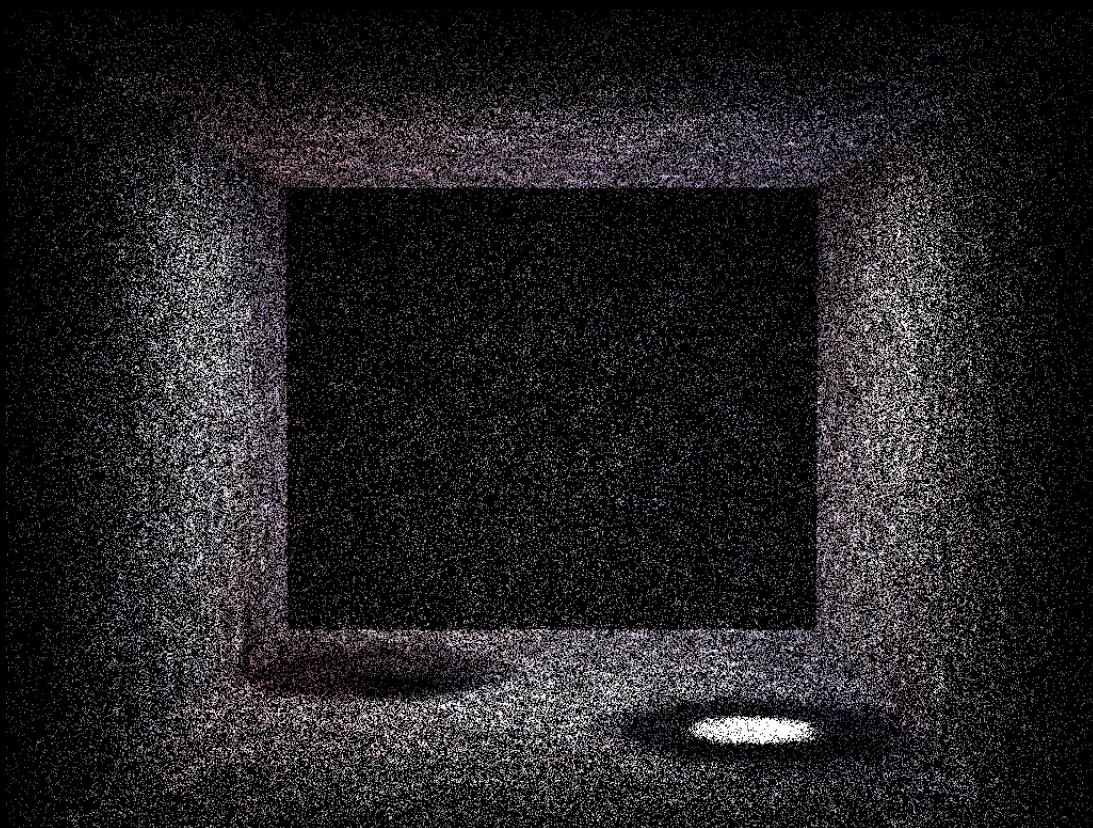
500000 photons / 500 photons in radiance estimate

Adding final gathering



200000 global photons, 50000 caustic photons

Global Photons



200000 global photons

Final observations

- Photon mapping is consistent
- Density estimate is the source of bias (blur)

Progressive Photon Density Estimation

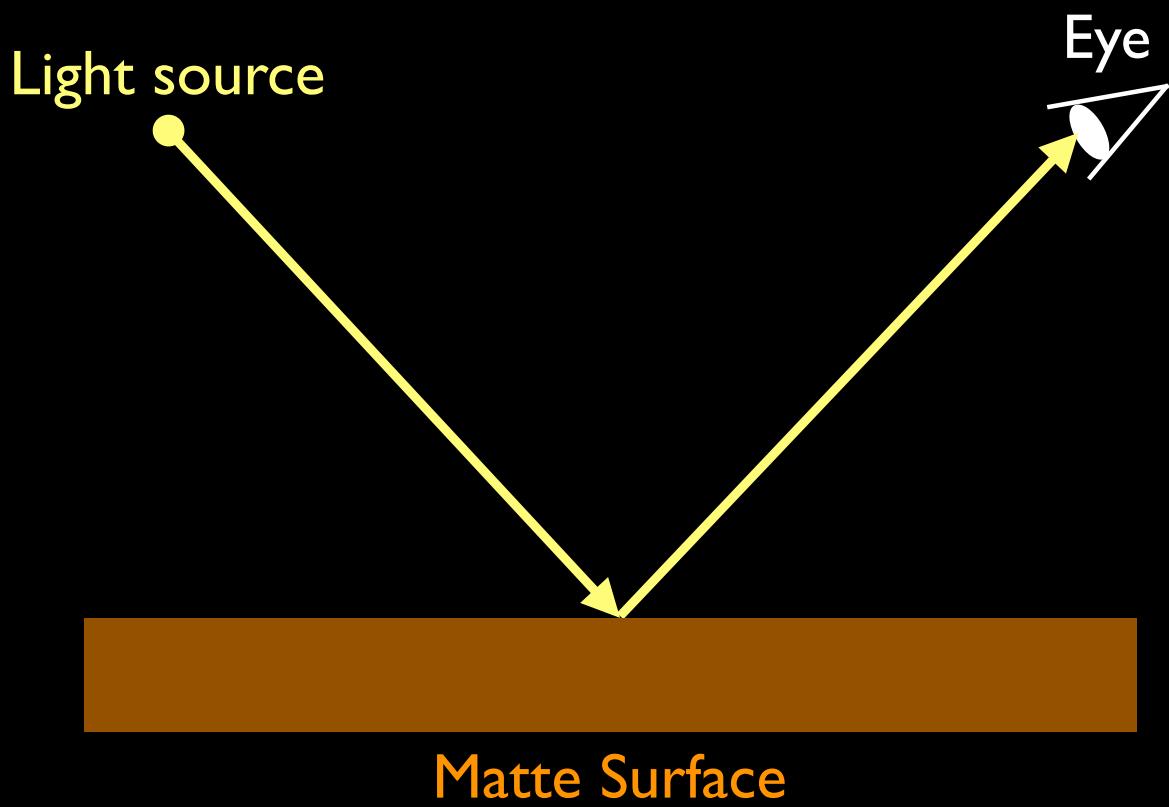
Progressive Photon Mapping Basics

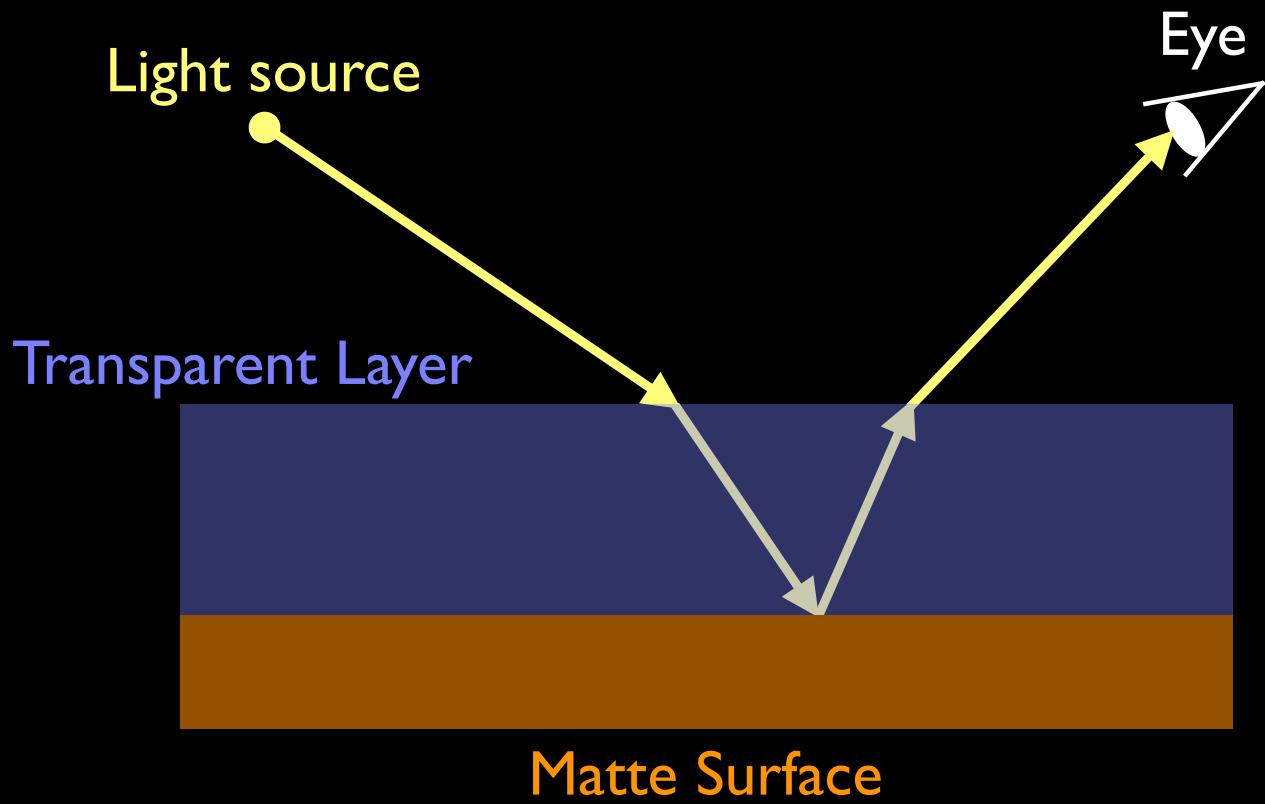


(c)Y. Kimura

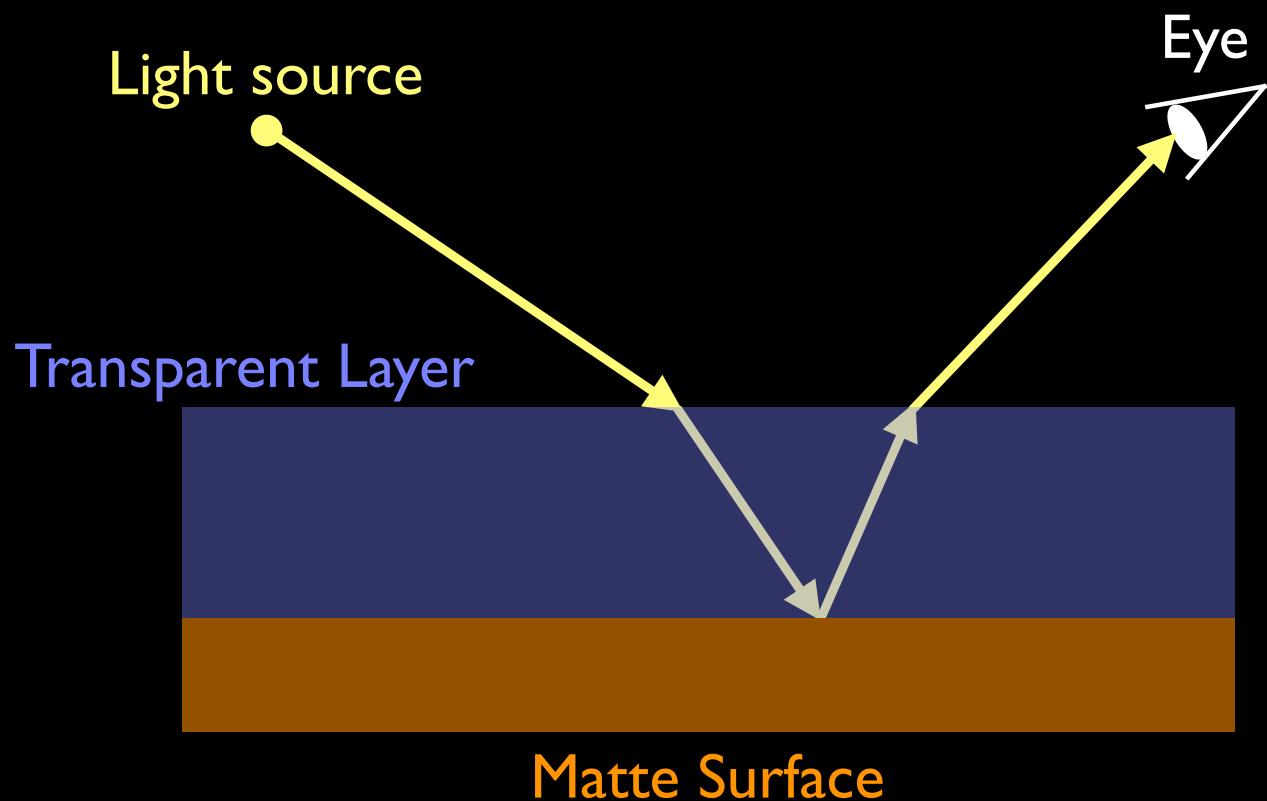
Global Illumination Algorithms

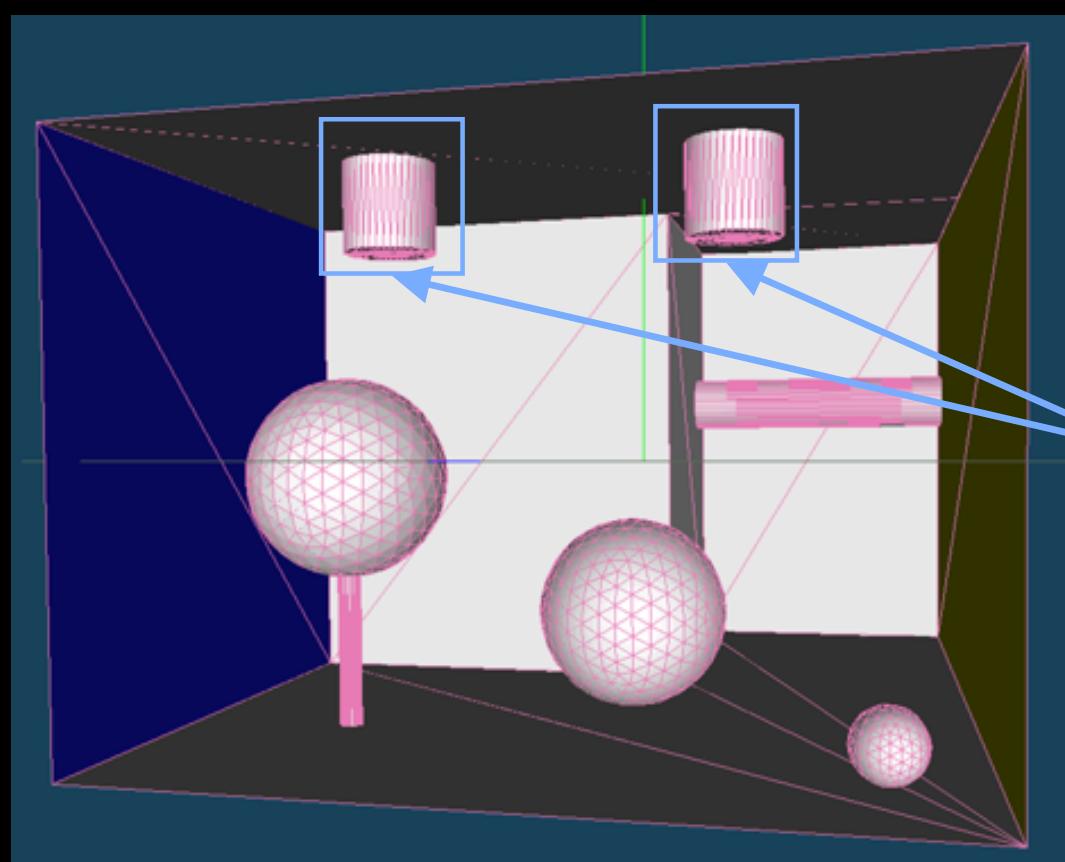
- Path Tracing [Kajiya 86]
- Light Tracing [Arvo 86][Dutr   93]
- Bidirectional Path Tracing [Lafortune 93][Veach 95]
- Photon Mapping [Jensen 95]
- Density Estimation [Shirley 95]
- Instant Radiosity [Keller 97]
- Metropolis Light Transport [Veach 97]
- Lightcuts [Walter 05]
- Energy Redistribution Path Tracing [Cline 05]
- ...



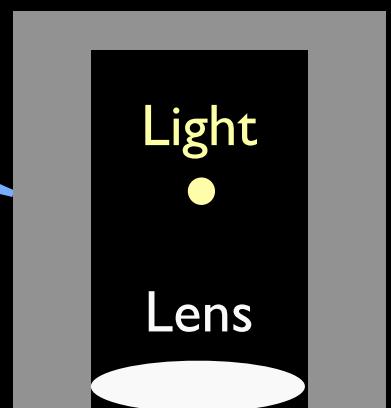


Specular-Diffuse-Specular Paths

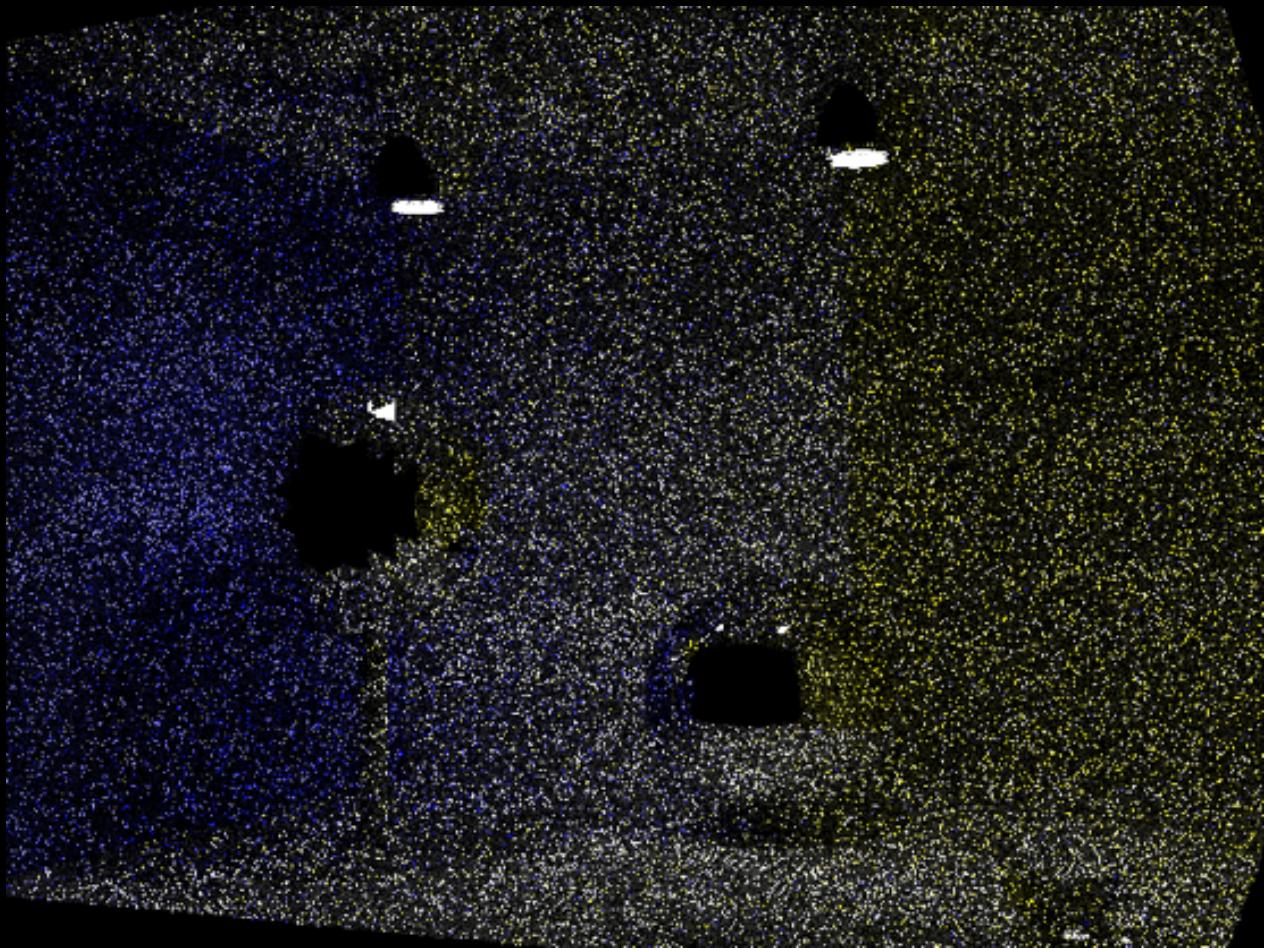




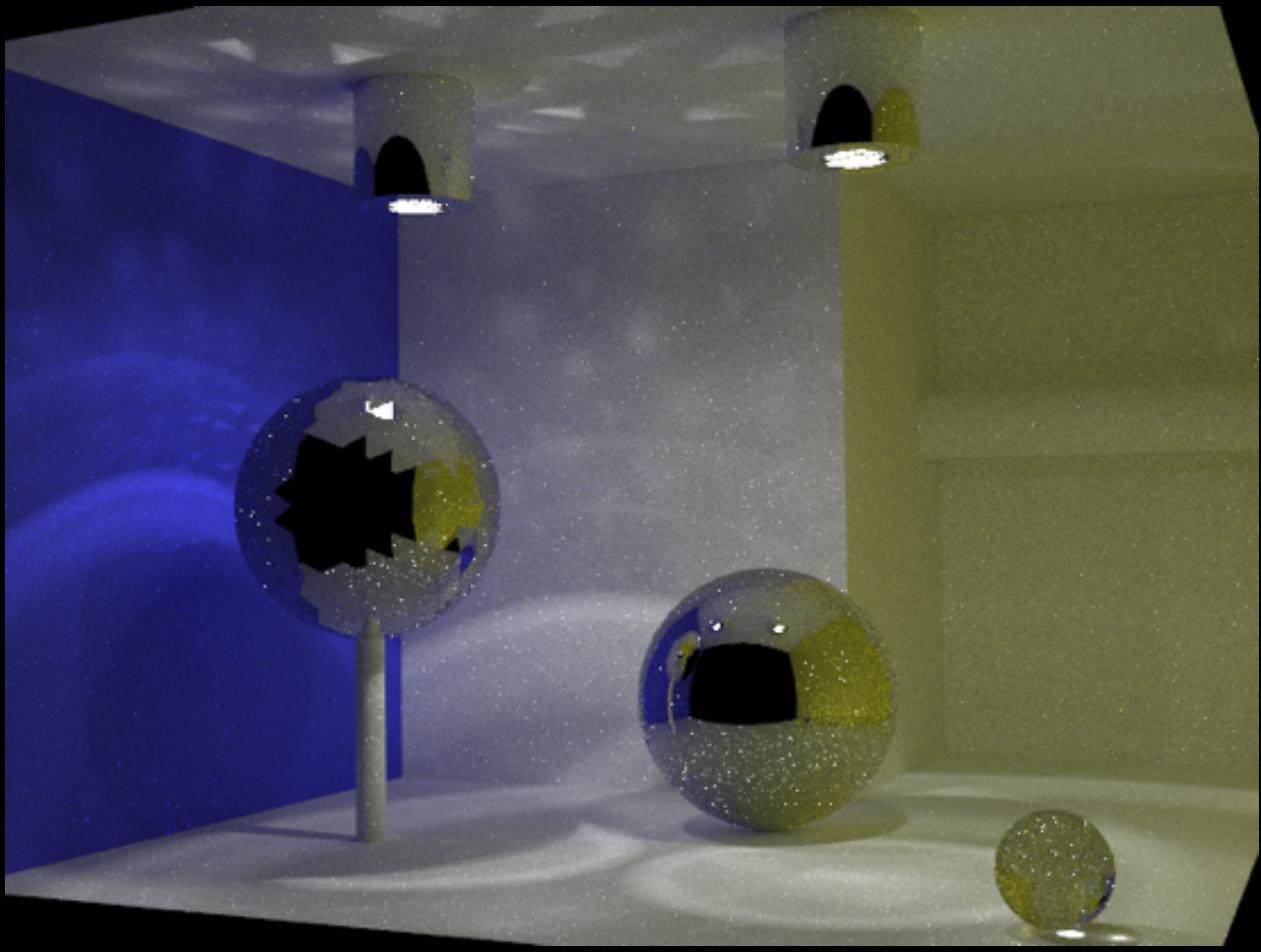
Metal tube



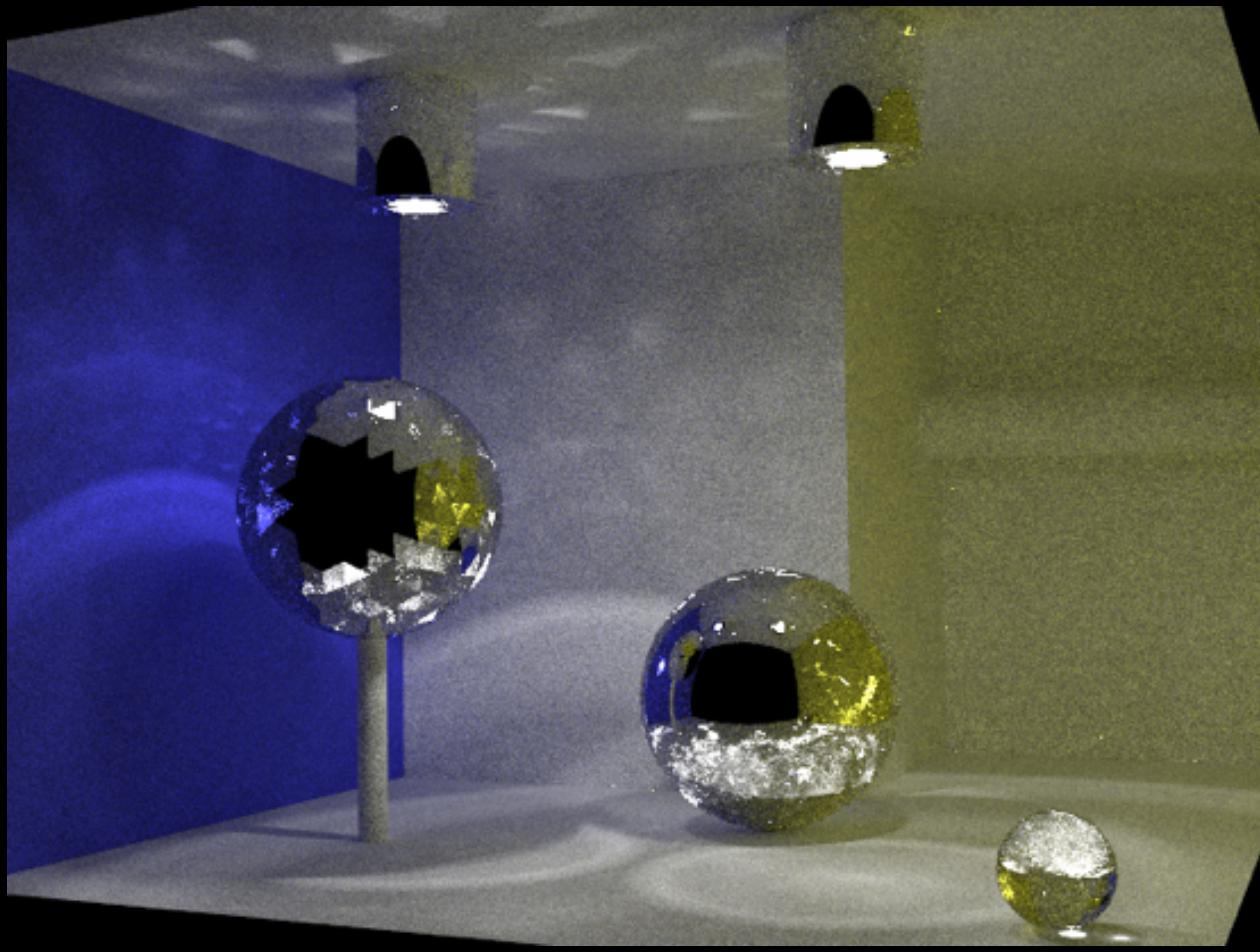
Path Tracing



Bidirectional Path Tracing

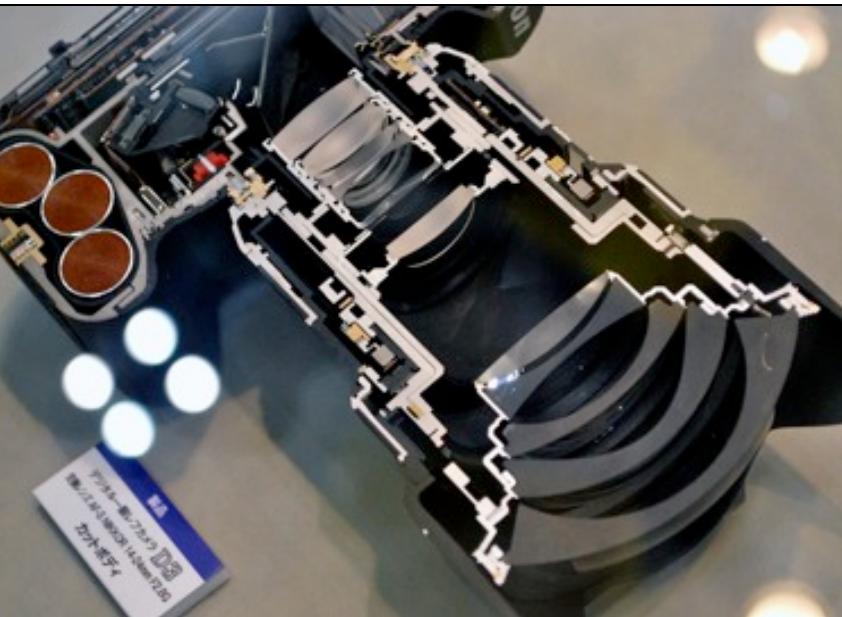


Metropolis Light Transport









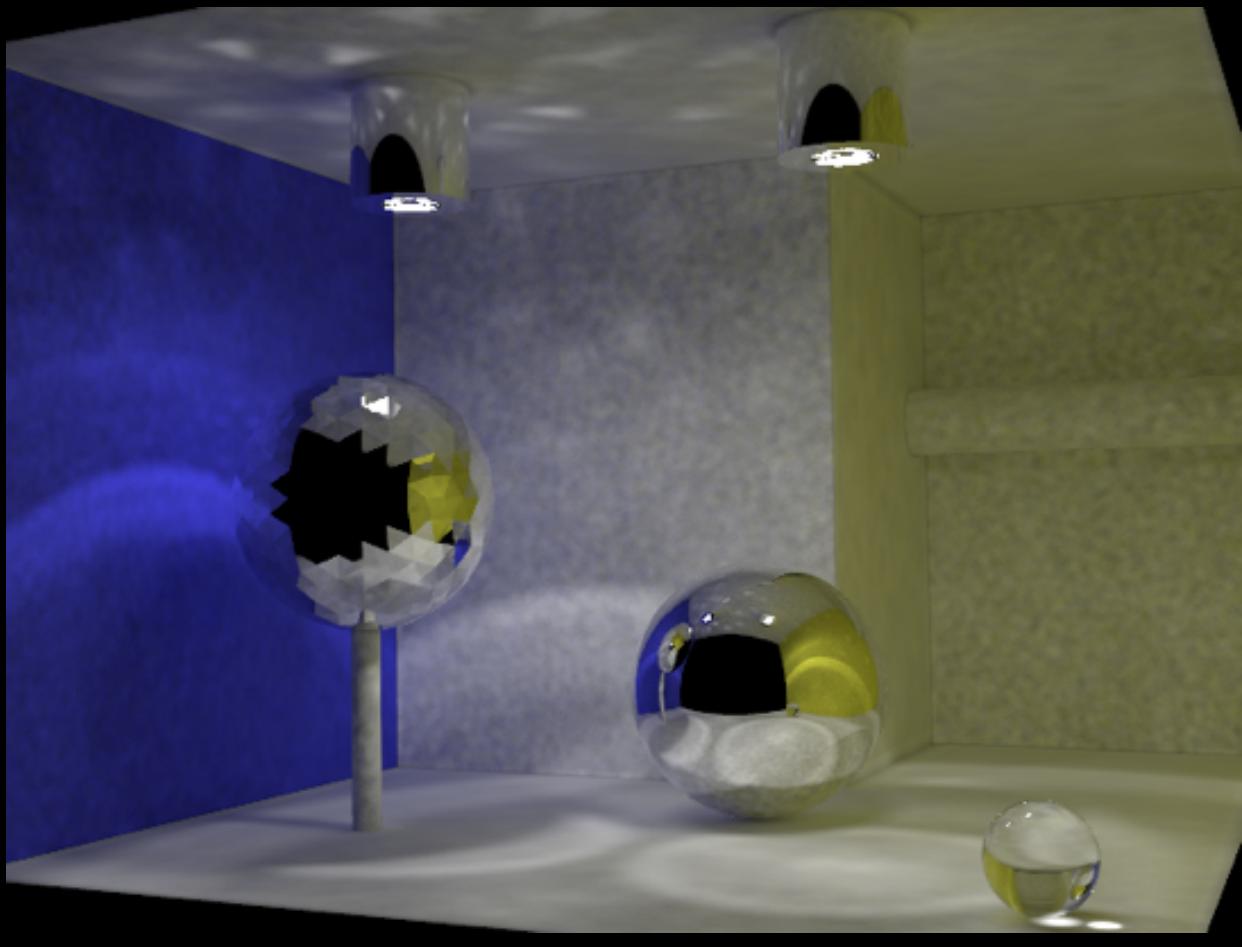
Specular-Diffuse-Specular Paths

- Existing methods are not robust for SDS paths
 - Path tracing
 - Bidirectional path tracing
 - Metropolis light transport
 - ...name your favorite

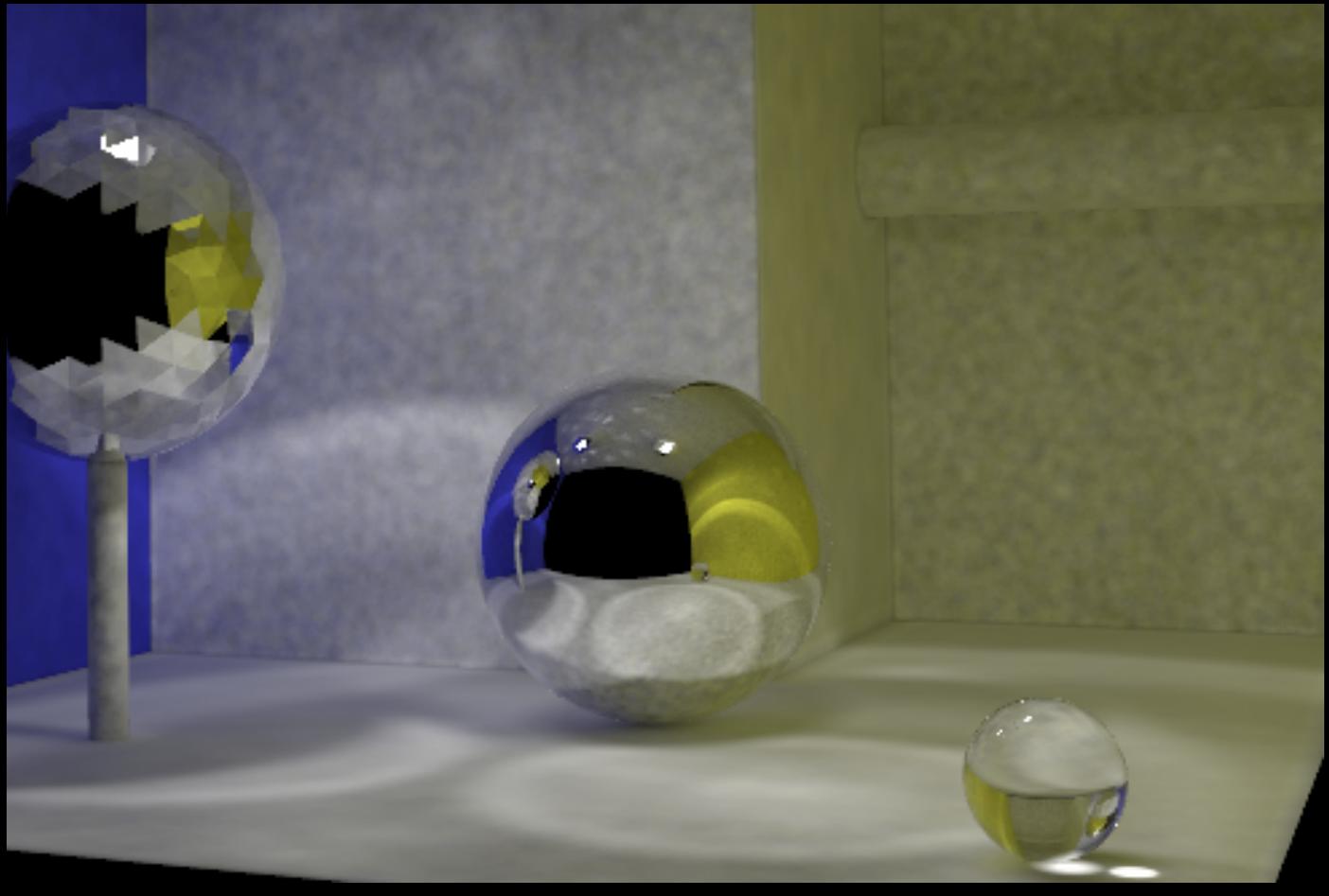
Specular-Diffuse-Specular Paths

- Existing methods are not robust for SDS paths
 - Path tracing
 - Bidirectional path tracing
 - Metropolis light transport
 - ...name your favorite
 - Photon mapping?

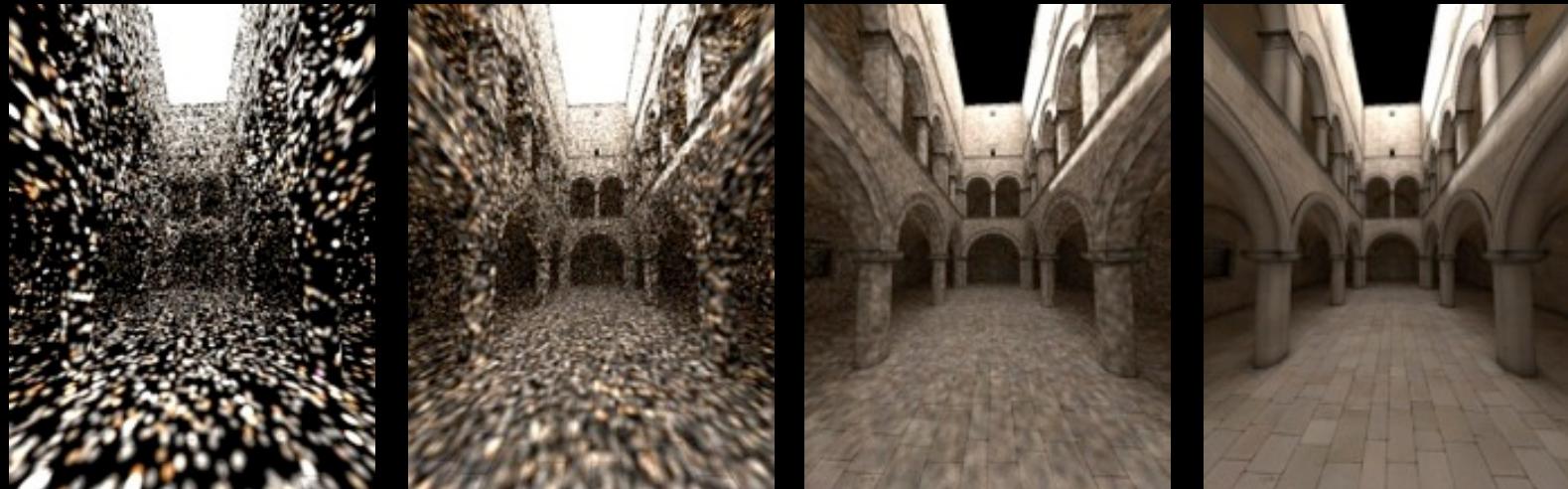
Photon Mapping



Photon Mapping



Convergence of Photon Mapping



More photons →

Convergence of Photon Mapping

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

- Infinite number of nearby photons ($N^\beta \rightarrow \infty$)
- Infinitely small radius ($r \rightarrow 0$)

Convergence of Photon Mapping

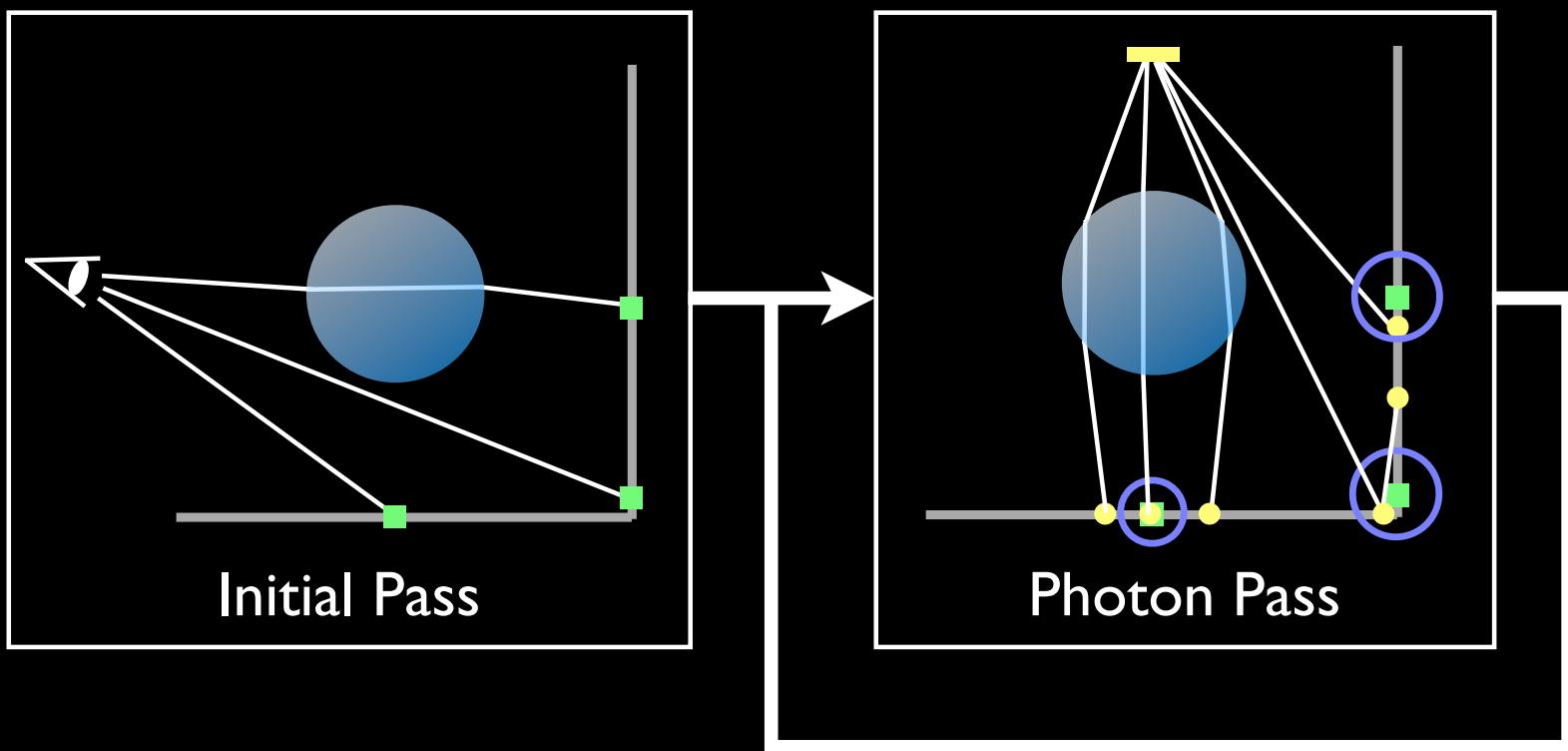
$$L(x, \vec{\omega}) = \lim_{N \rightarrow \infty} \sum_{p=1}^{N^\beta} \frac{f_r(x, \vec{\omega}, \vec{\omega}_p) \phi_p(x_p, \vec{\omega}_p)}{\pi r^2}$$

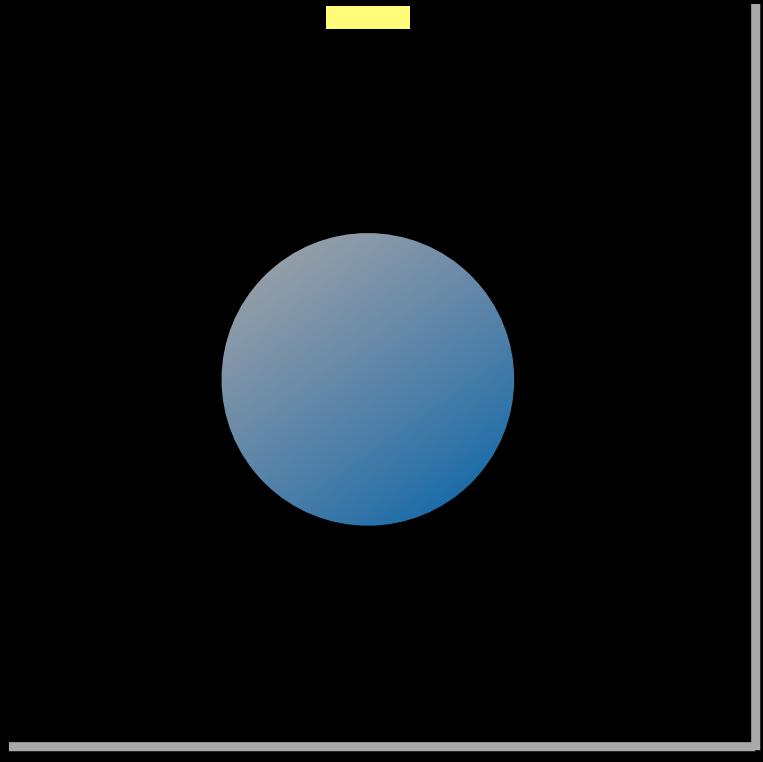
Infinite storage & photon tracing

- Infinite number of nearby photons ($N^\beta \rightarrow \infty$)
- Infinitely small radius ($r \rightarrow 0$)

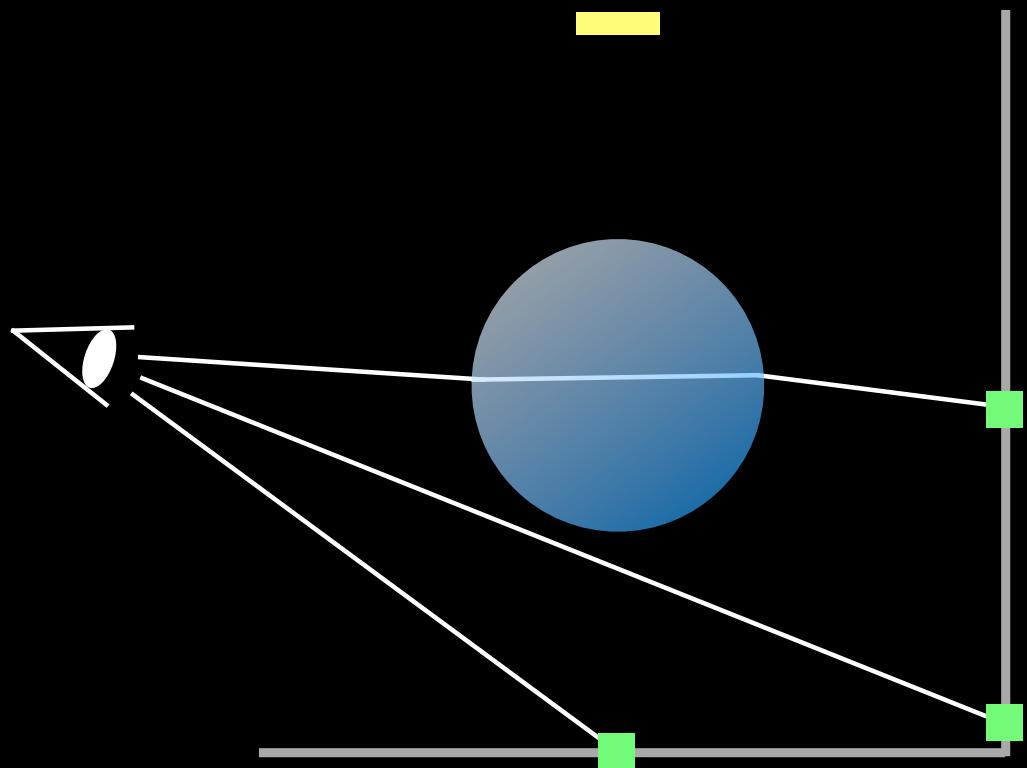
Solution: Progressive Photon Mapping

Overview

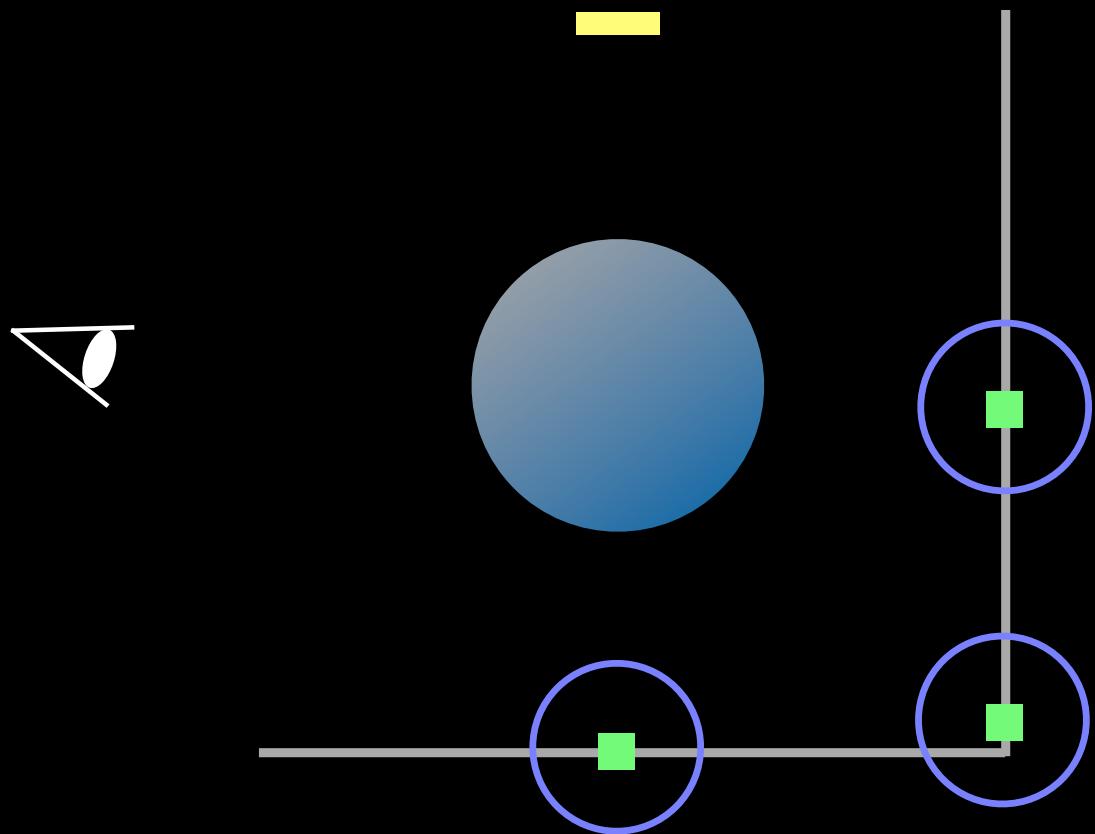




Initial Pass



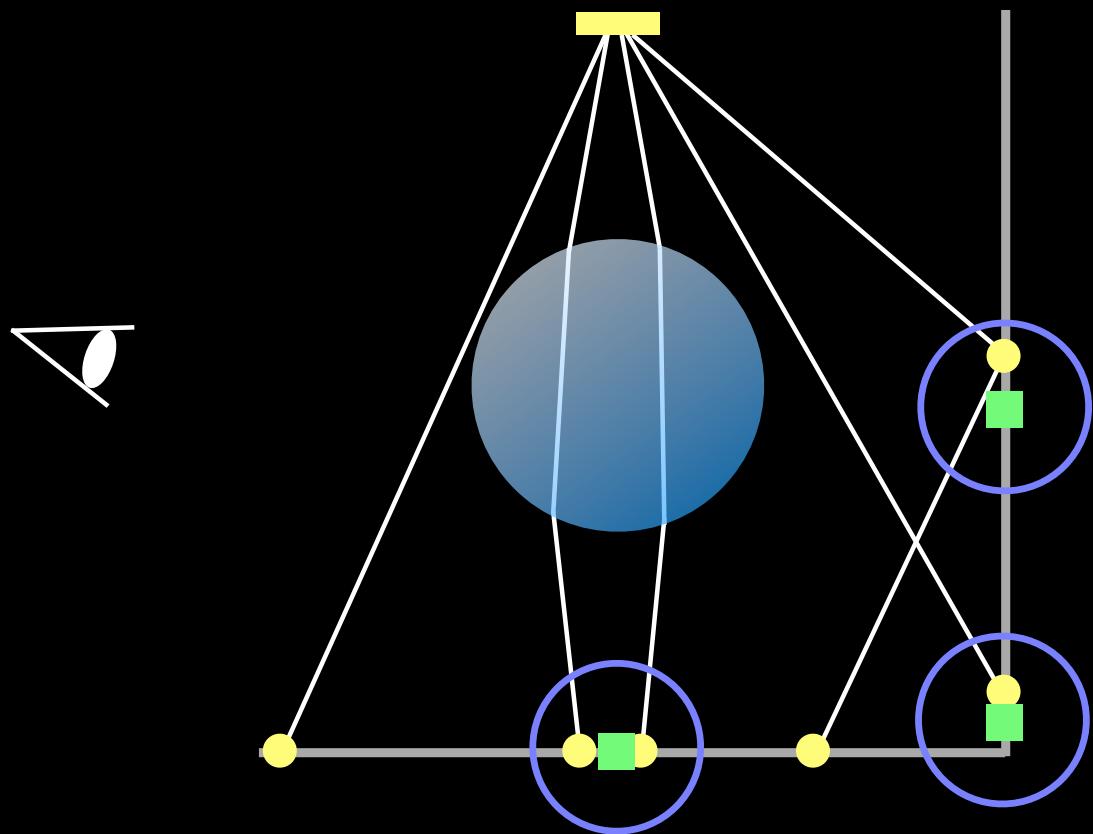
Initial Pass



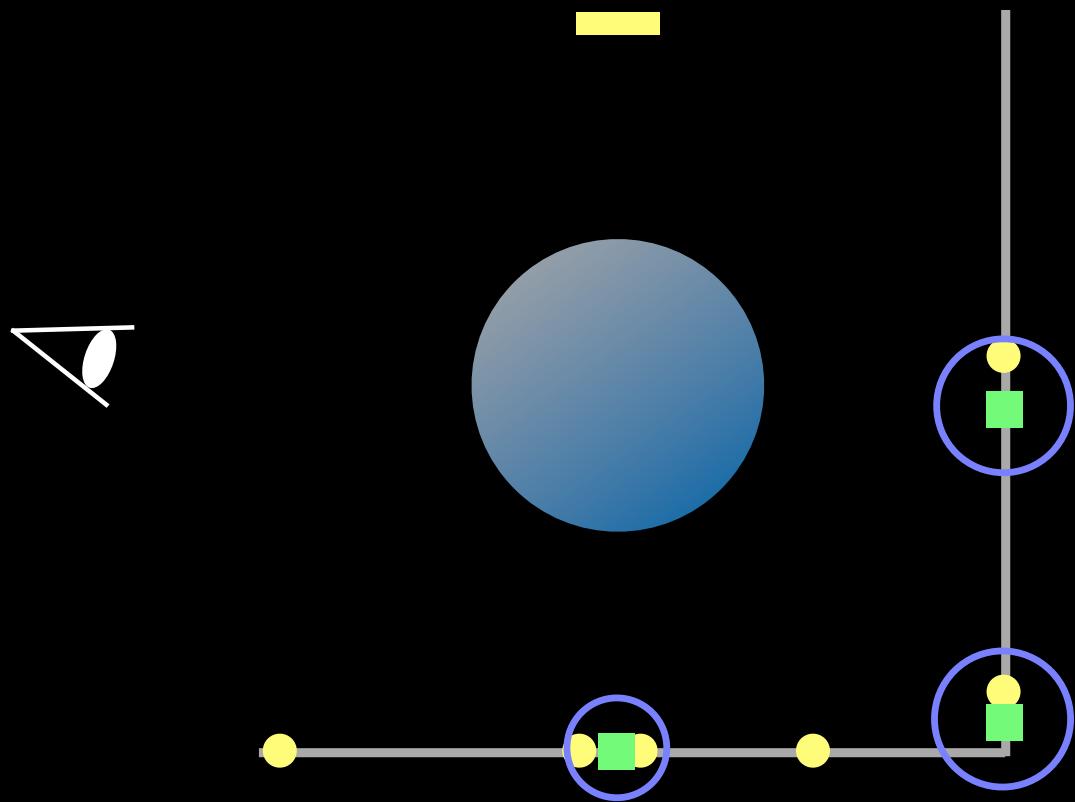
Photon Statistics

- Each measurement point:
 - Accumulated flux times BRDF $\tau_i(x, \vec{\omega})$
 - Search radius $R_i(x)$
 - Local photon count $N_i(x)$
- Global:
 - Emitted photon count $N_e(i)$

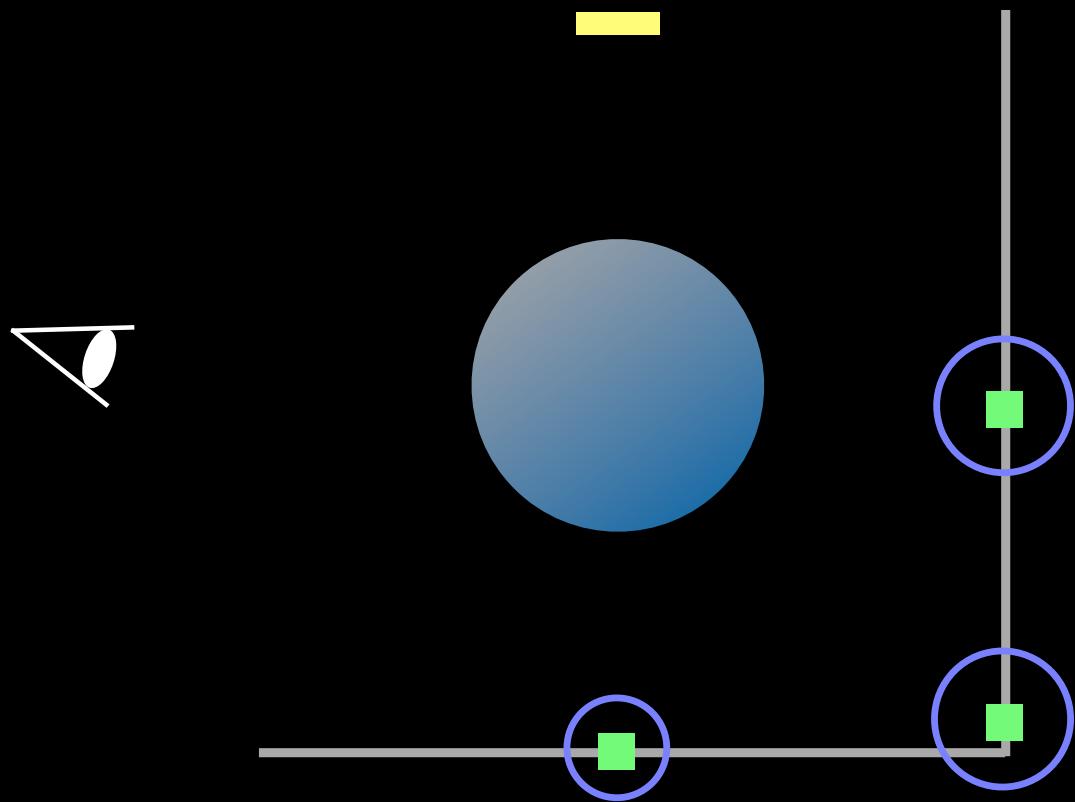
Photon Pass



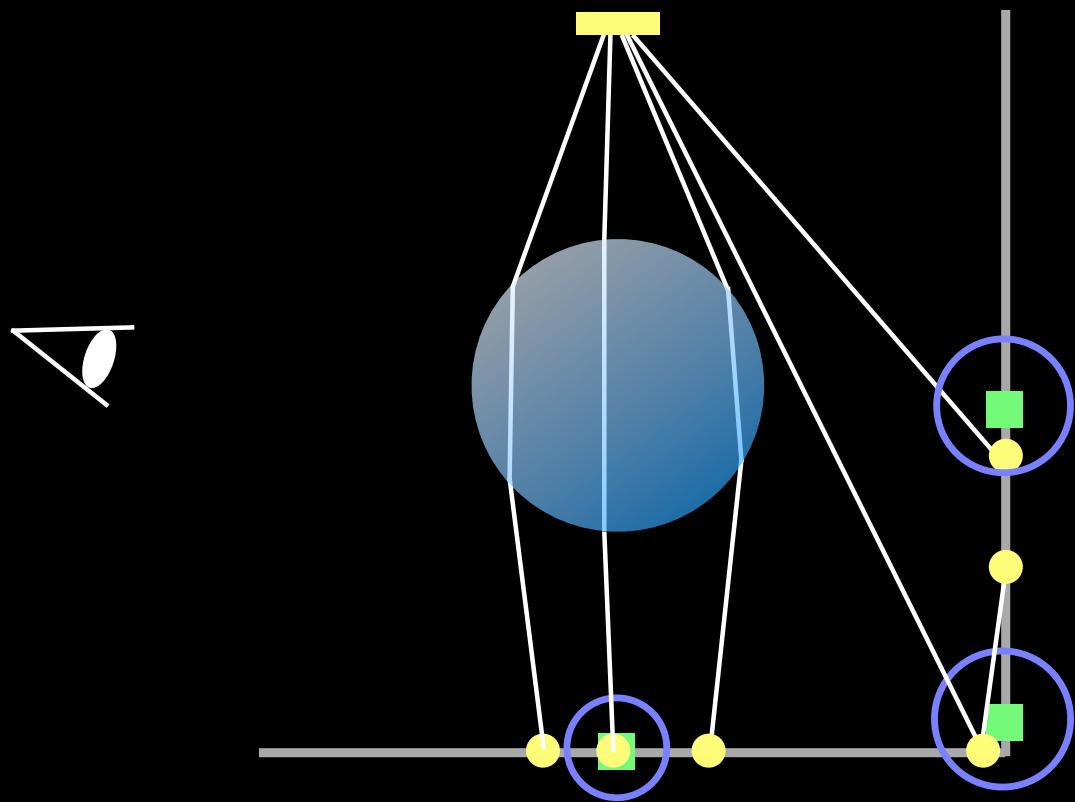
Photon Pass



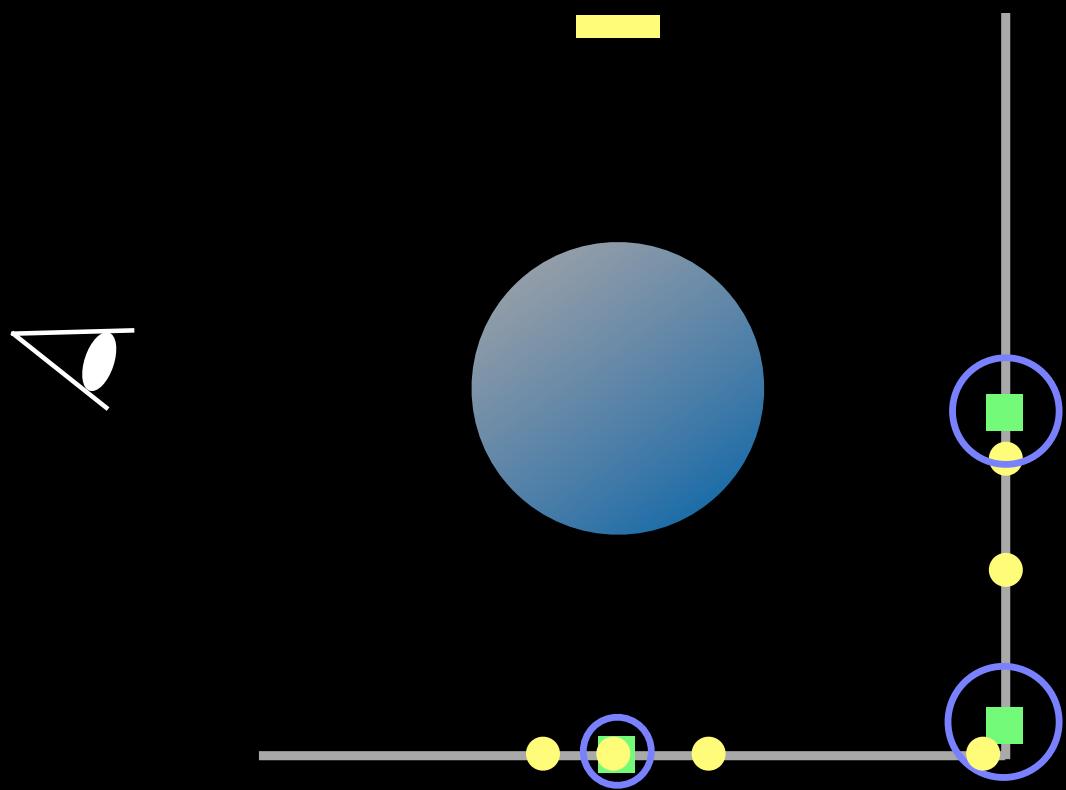
Photon Pass



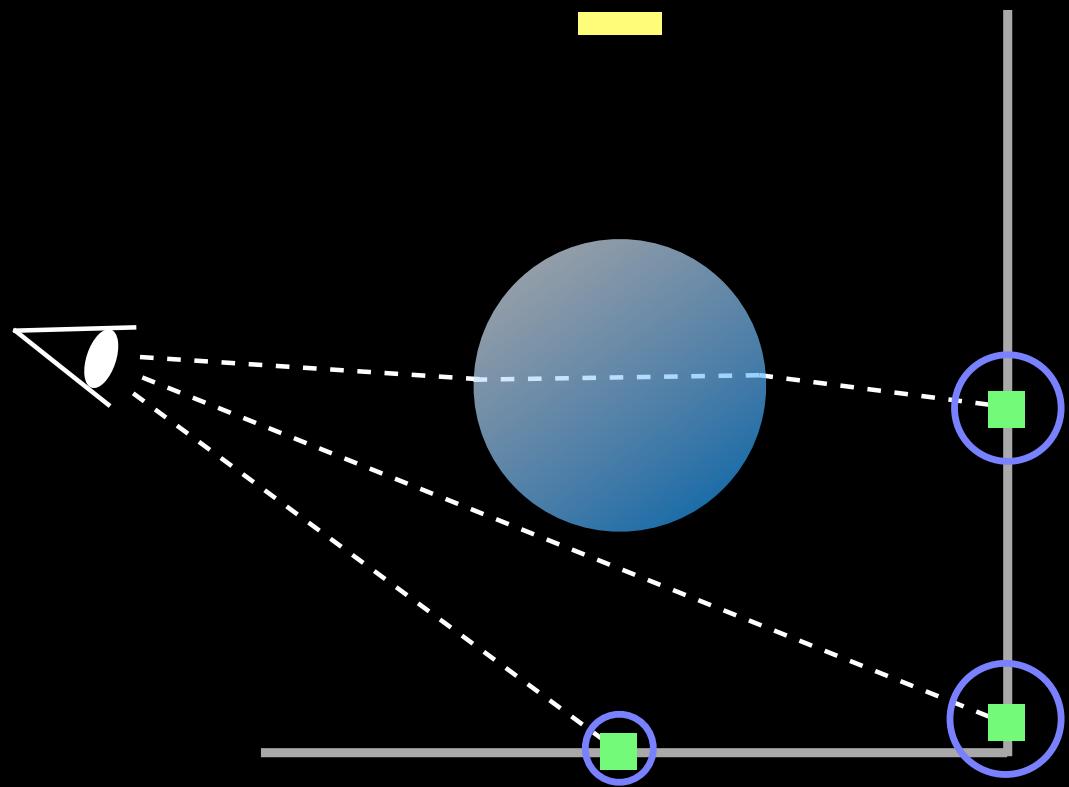
Next Photon Pass



Next Photon Pass



Rendering

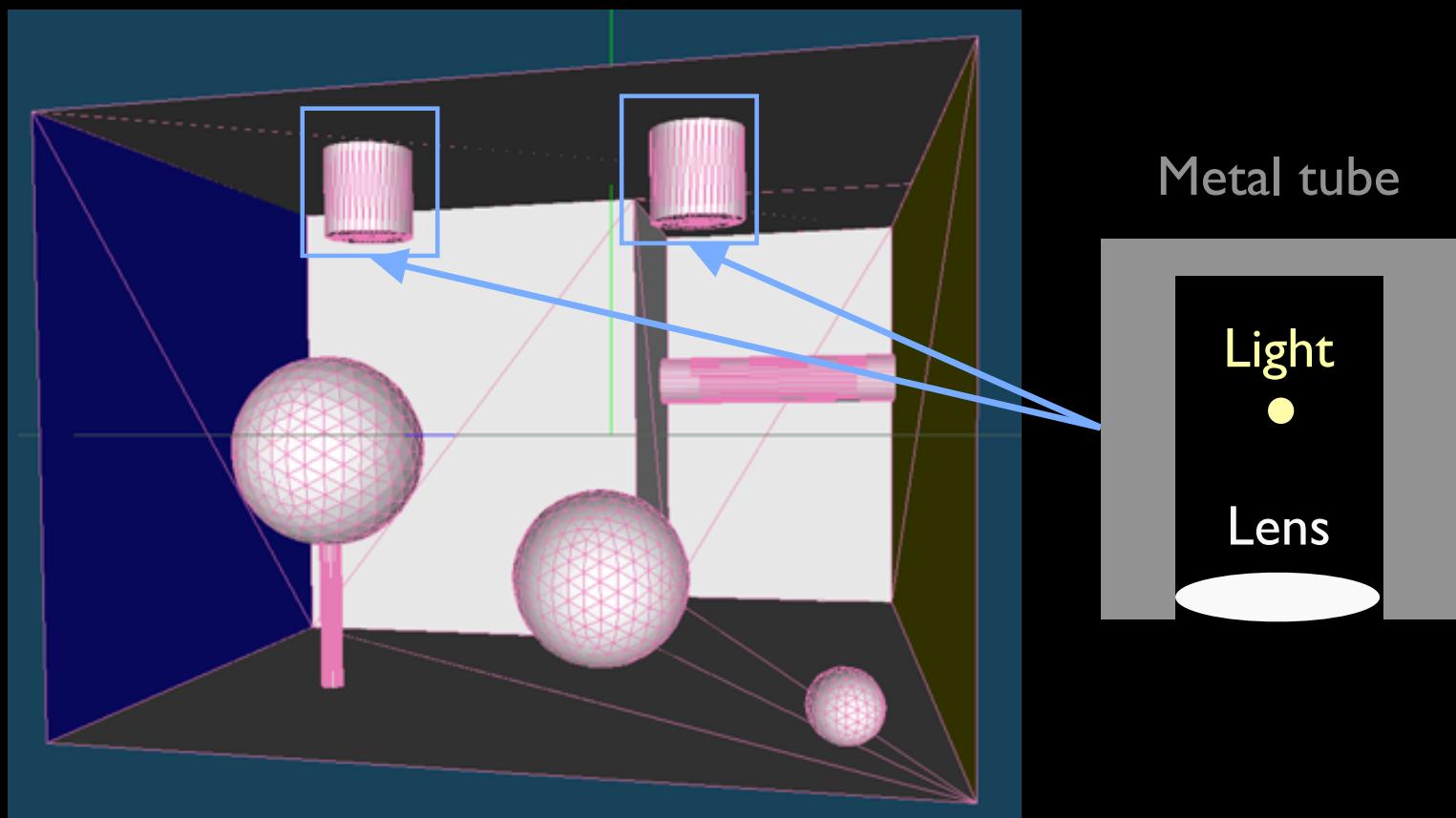


Progressive Density Estimation

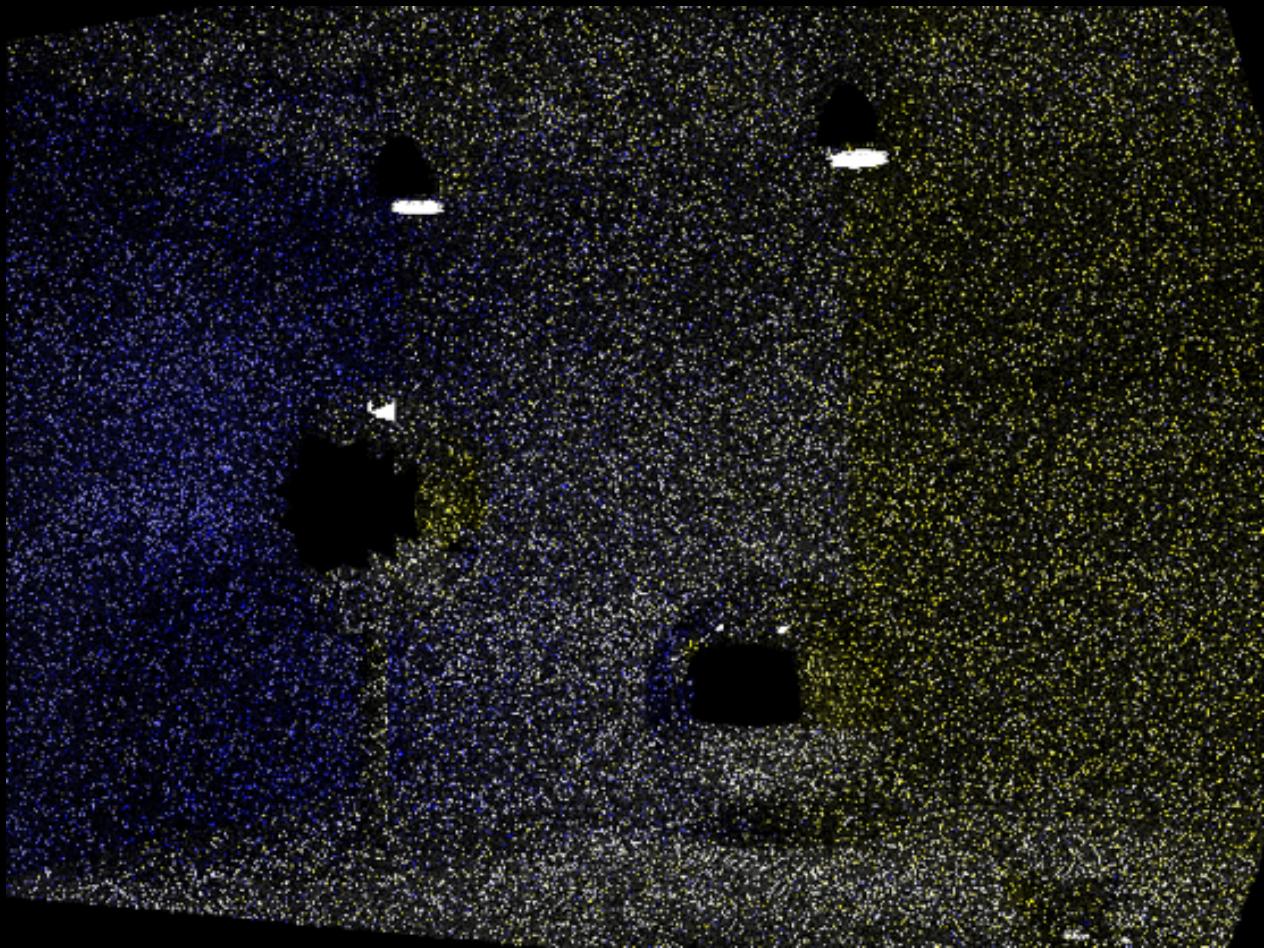
$$L_i(\vec{\omega}) = \sum_{p=1}^{N_i} \frac{f_r(\vec{\omega}, \vec{\omega}_p) \phi_p(\vec{\omega}_p)}{\pi R_i^2}$$

- Converges to the correct solution
 - Infinite number of photons
 - Infinitely small radius

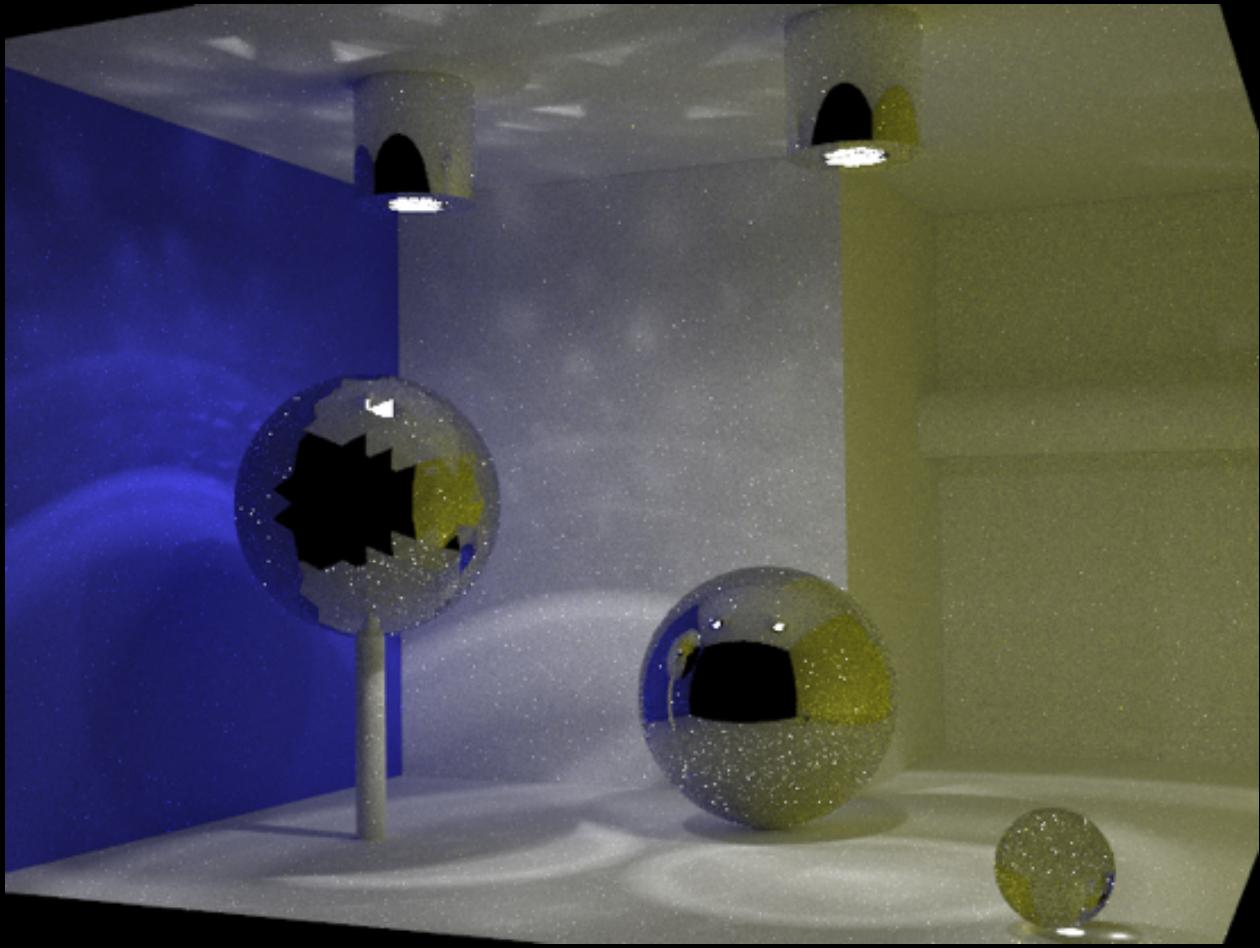
Equal-time Comparisons



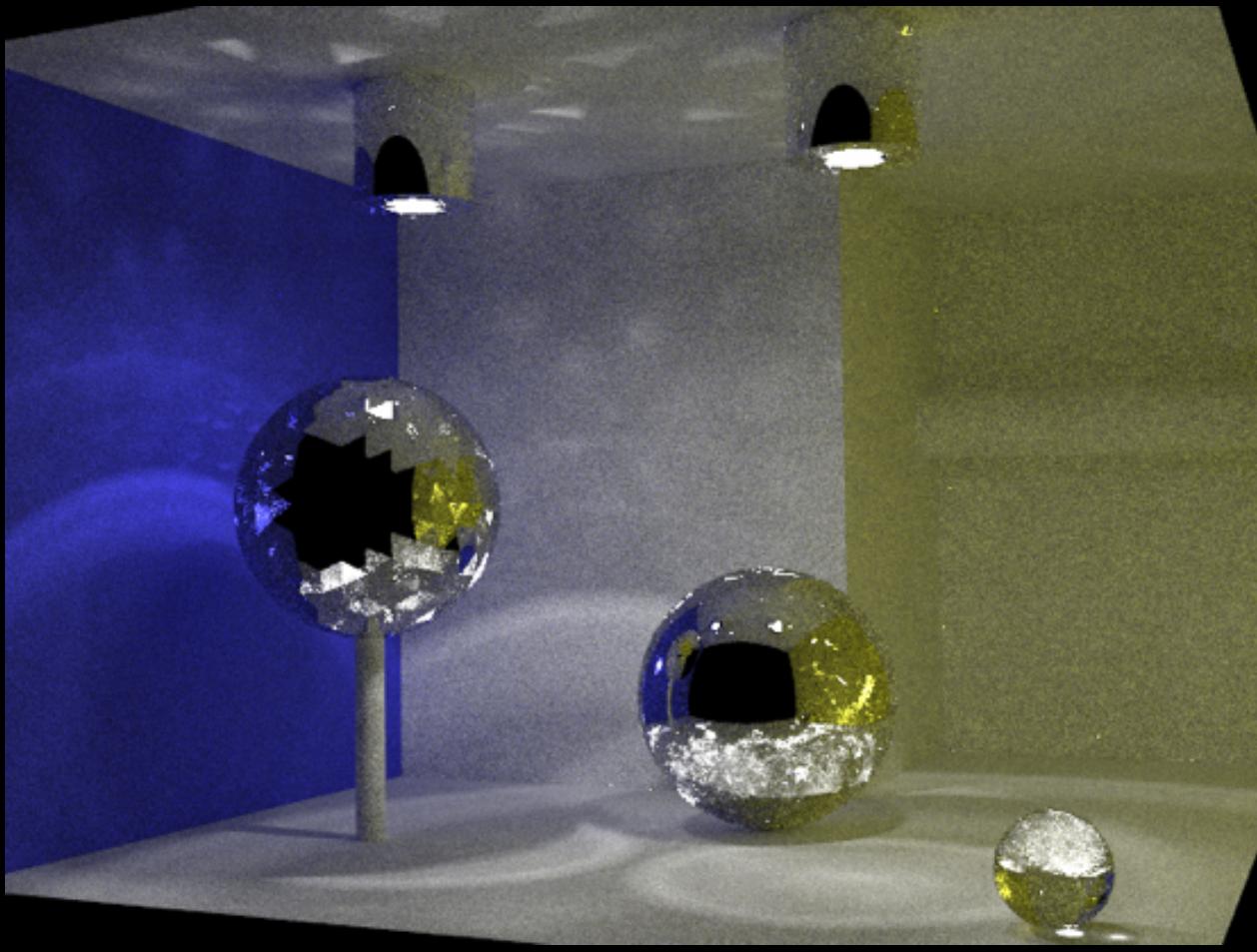
Path Tracing



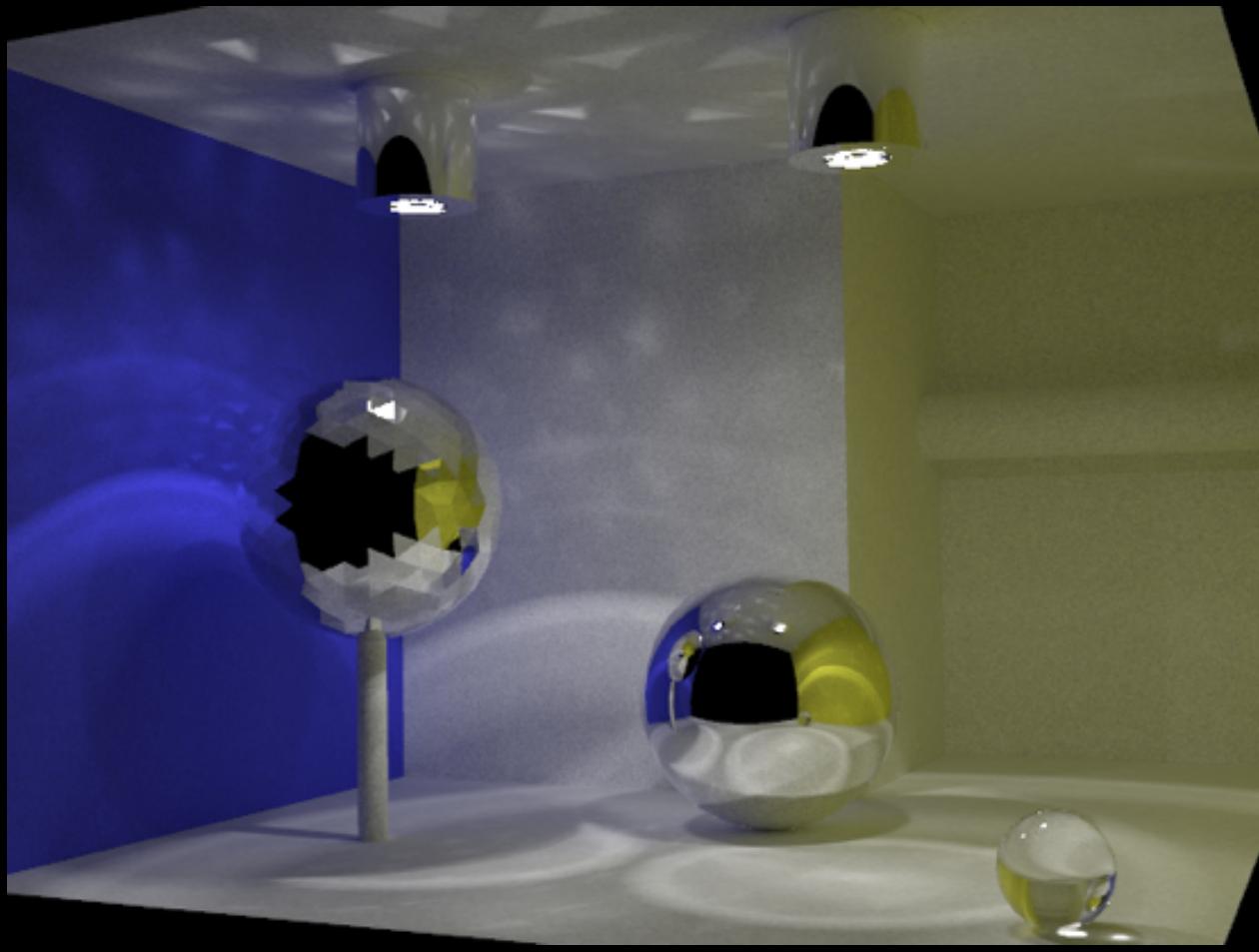
Bidirectional Path Tracing



Metropolis Light Transport



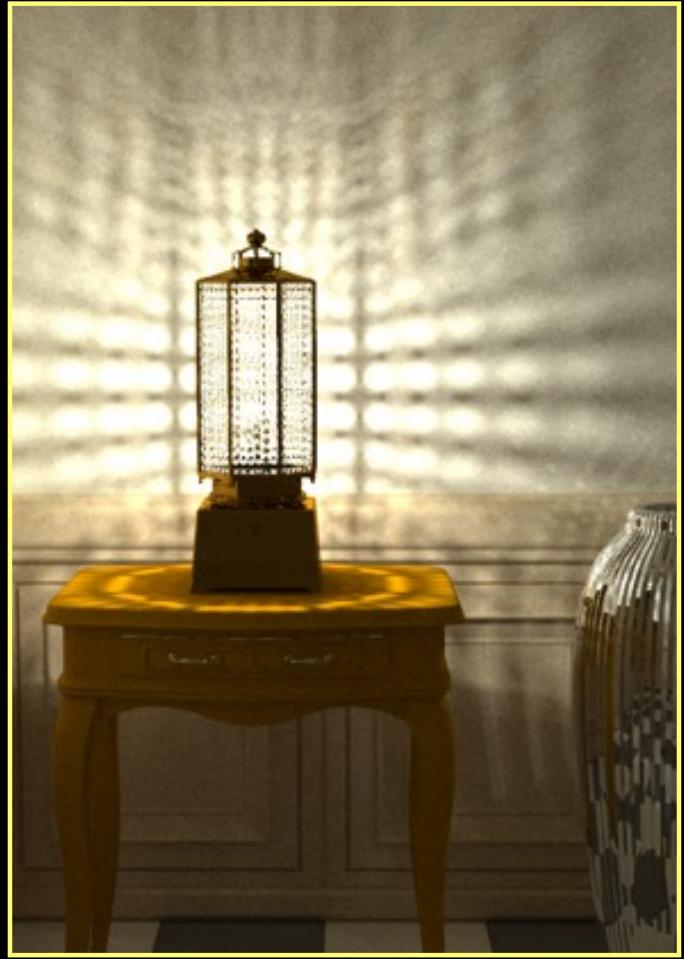
Progressive Photon Mapping



Bidirectional Path Tracing



Progressive Photon Mapping



Metropolis Light Transport

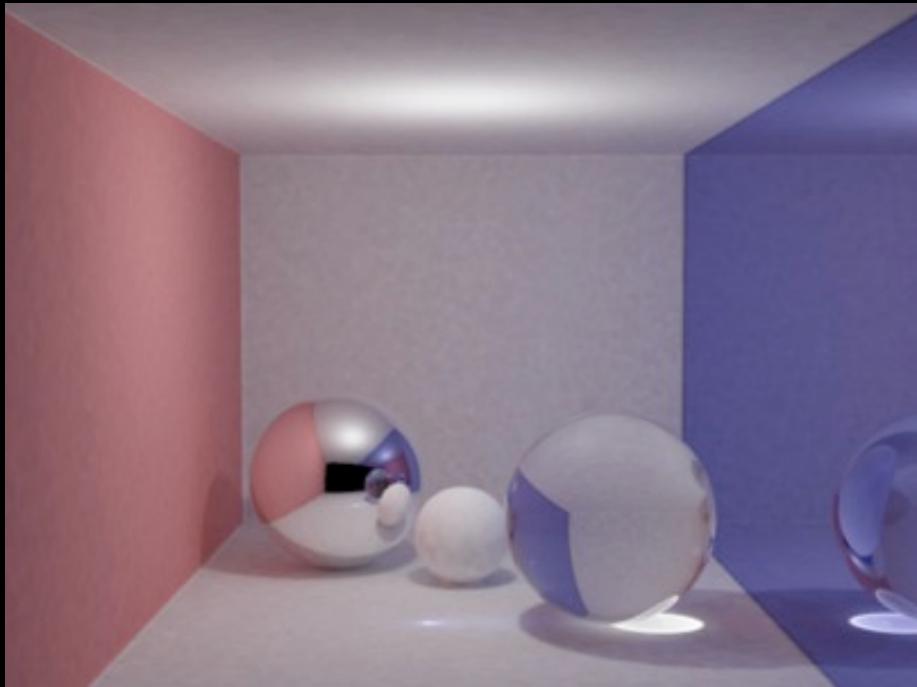


Progressive Photon Mapping



Sample Code

- smallppm - 128 lines of working PPM code



cs.au.dk/~toshiya/smallppm.cpp

Summary

- Infinite number of photons without storing them
 - “Path tracing”nization of photon mapping
- Robust to specular-diffuse-specular paths
- Converges to the correct solution
- Easy to implement

PPM in the Wild



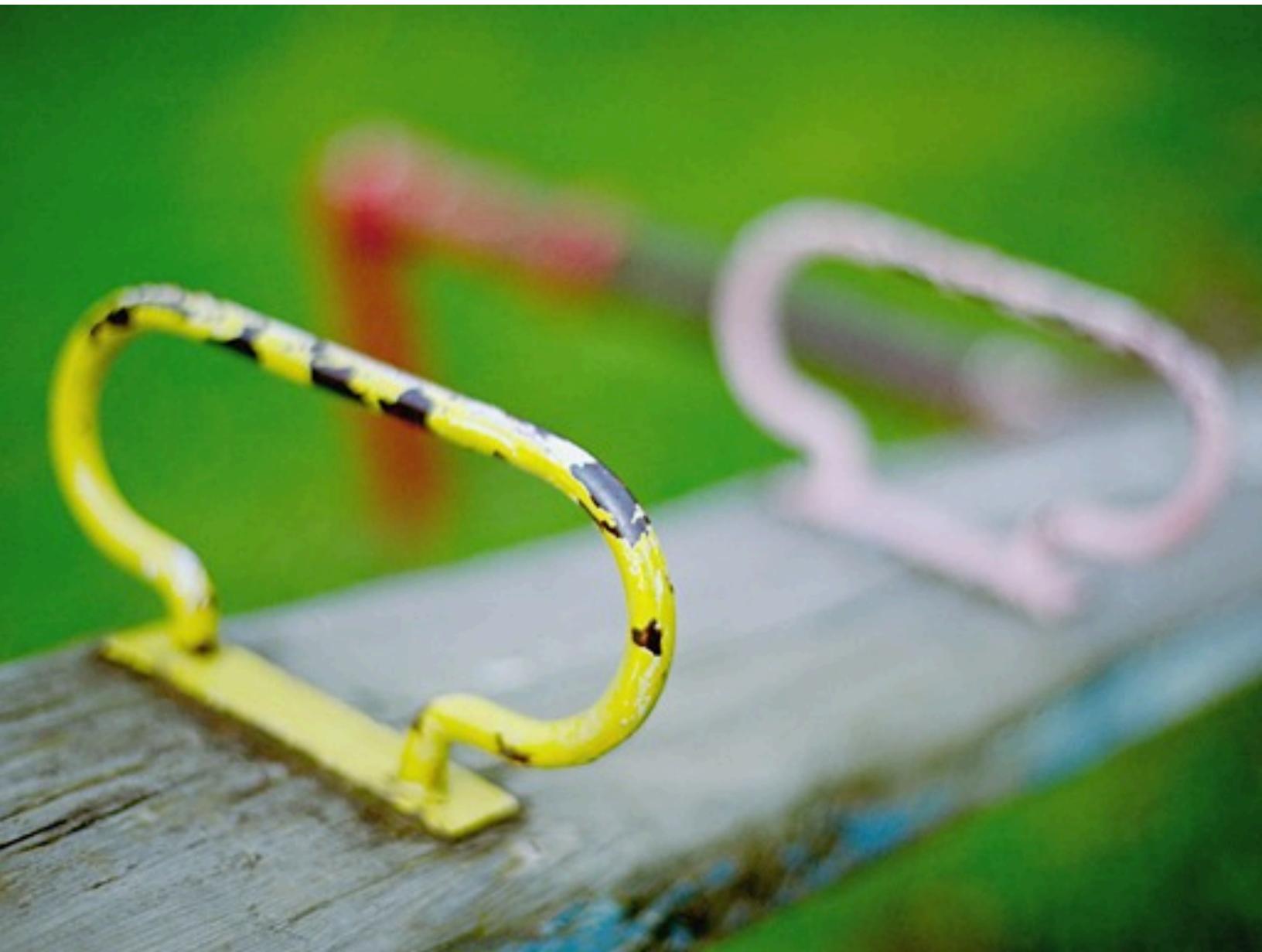
NVIDIA.

VRED

PROFESSIONAL



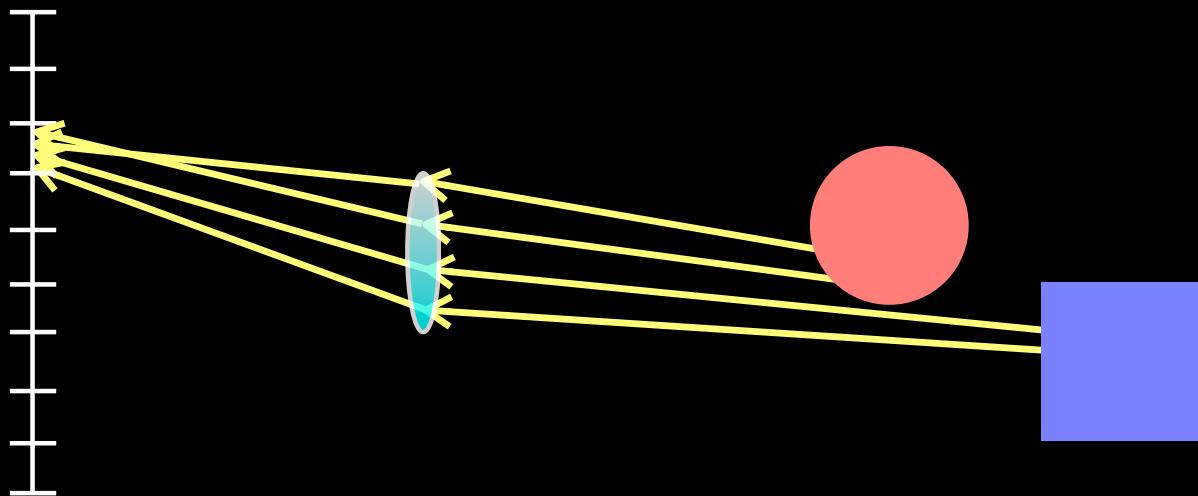
Progressive Photon Mapping & Extensions



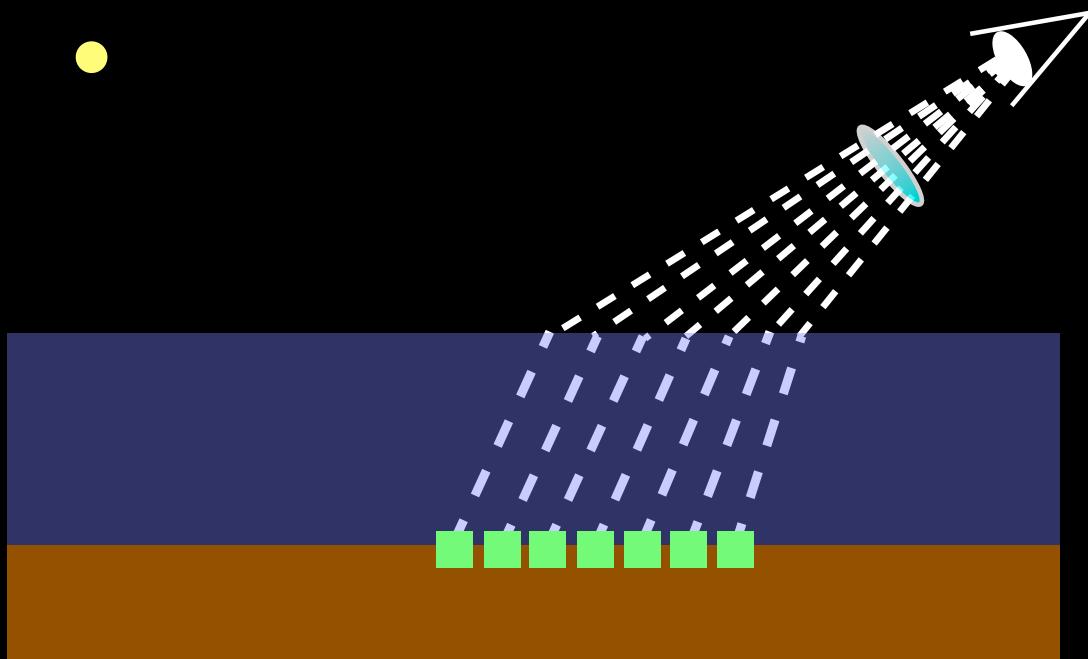


Distributed Ray Tracing

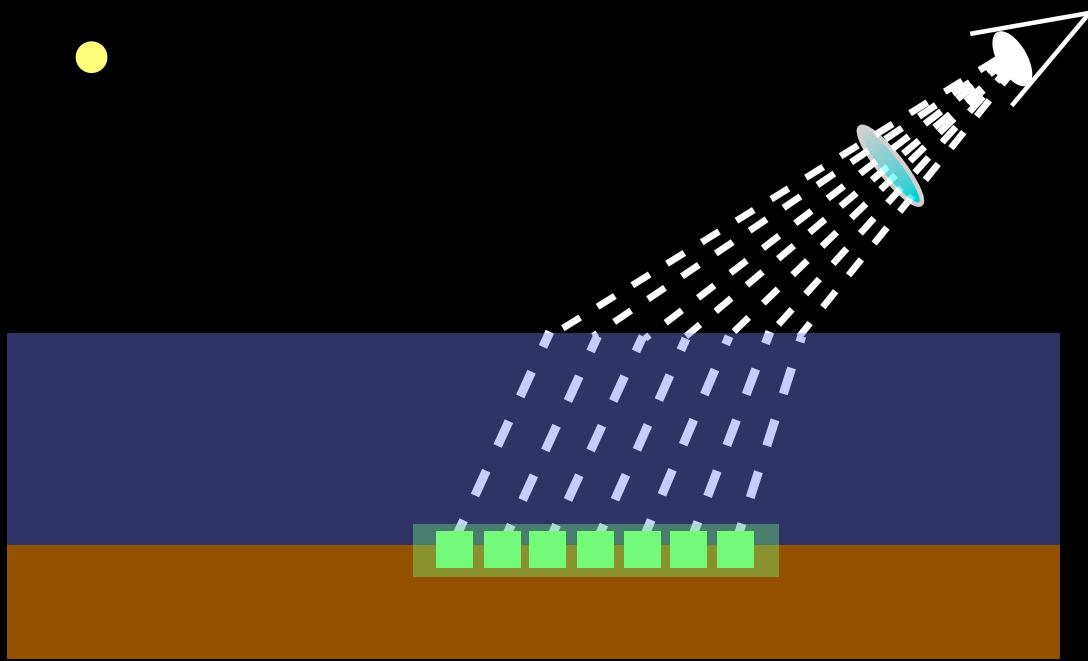
- Computes average illumination [Cook et al. 84]



Lens Simulation with PPM



Lens Simulation with PPM



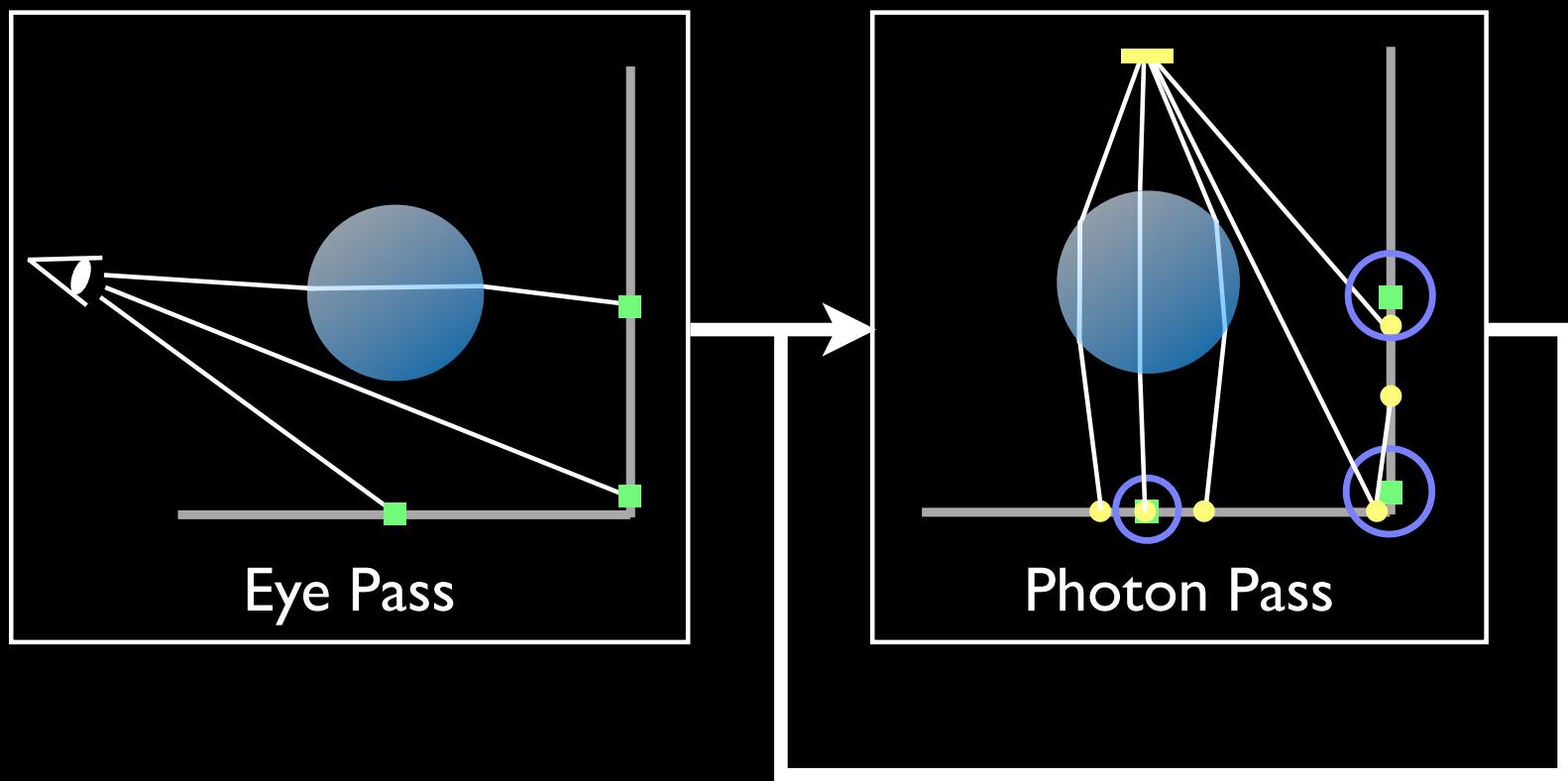
Infinite number of measurement points

Stochastic Progressive Photon Mapping

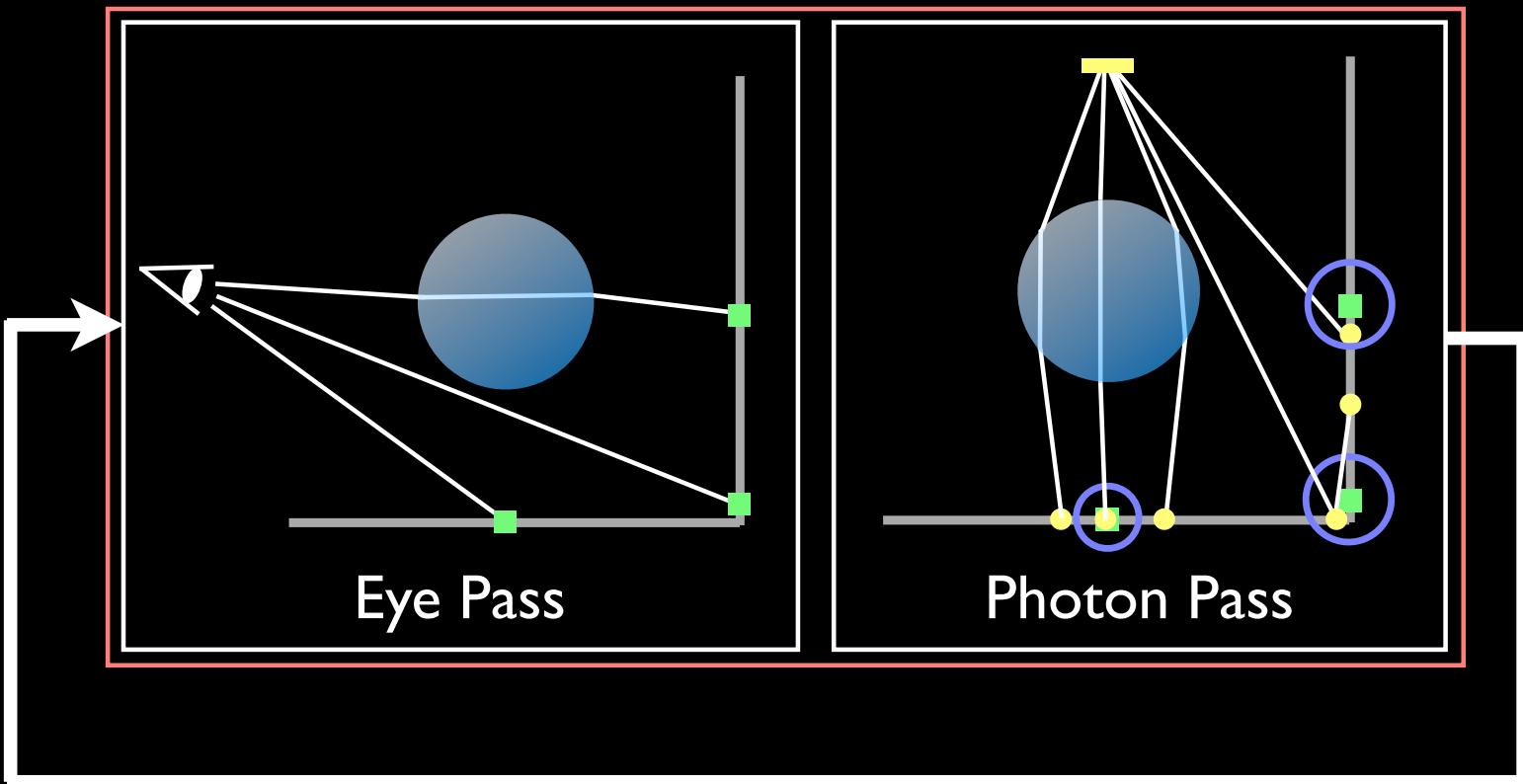
Toshiya Hachisuka Henrik Wann Jensen

Published at SIGGRAPH Asia 2009

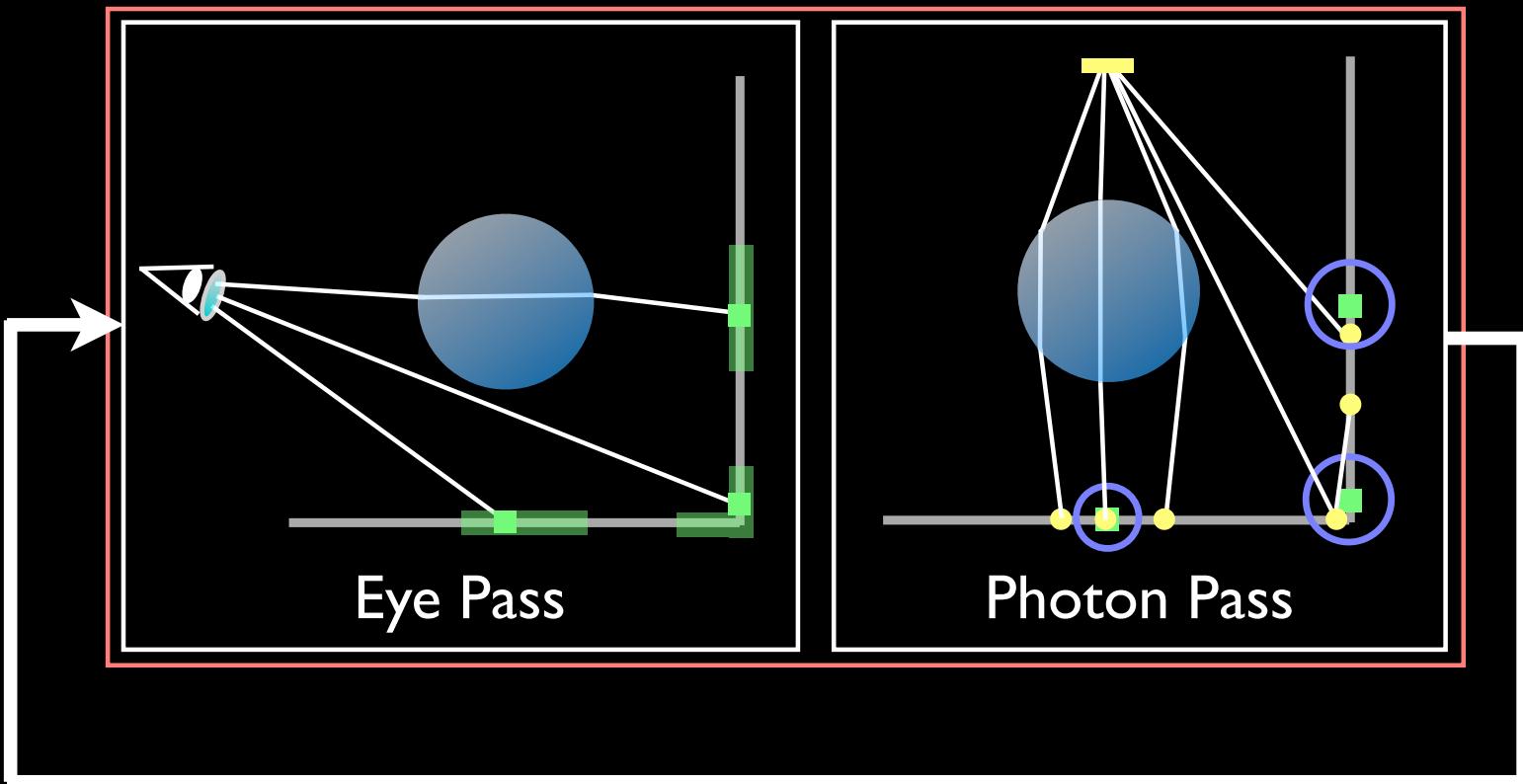
PPM



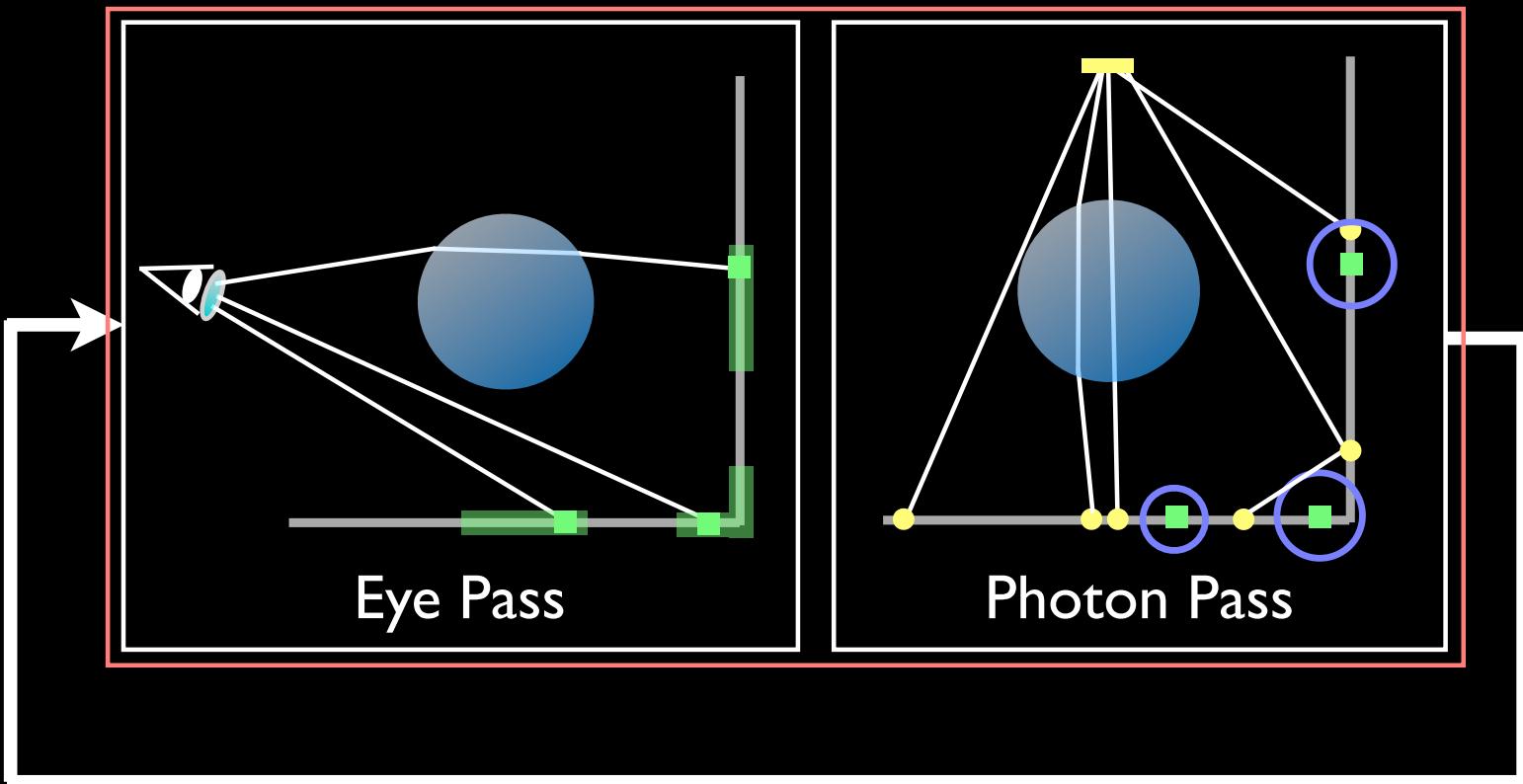
Stochastic PPM



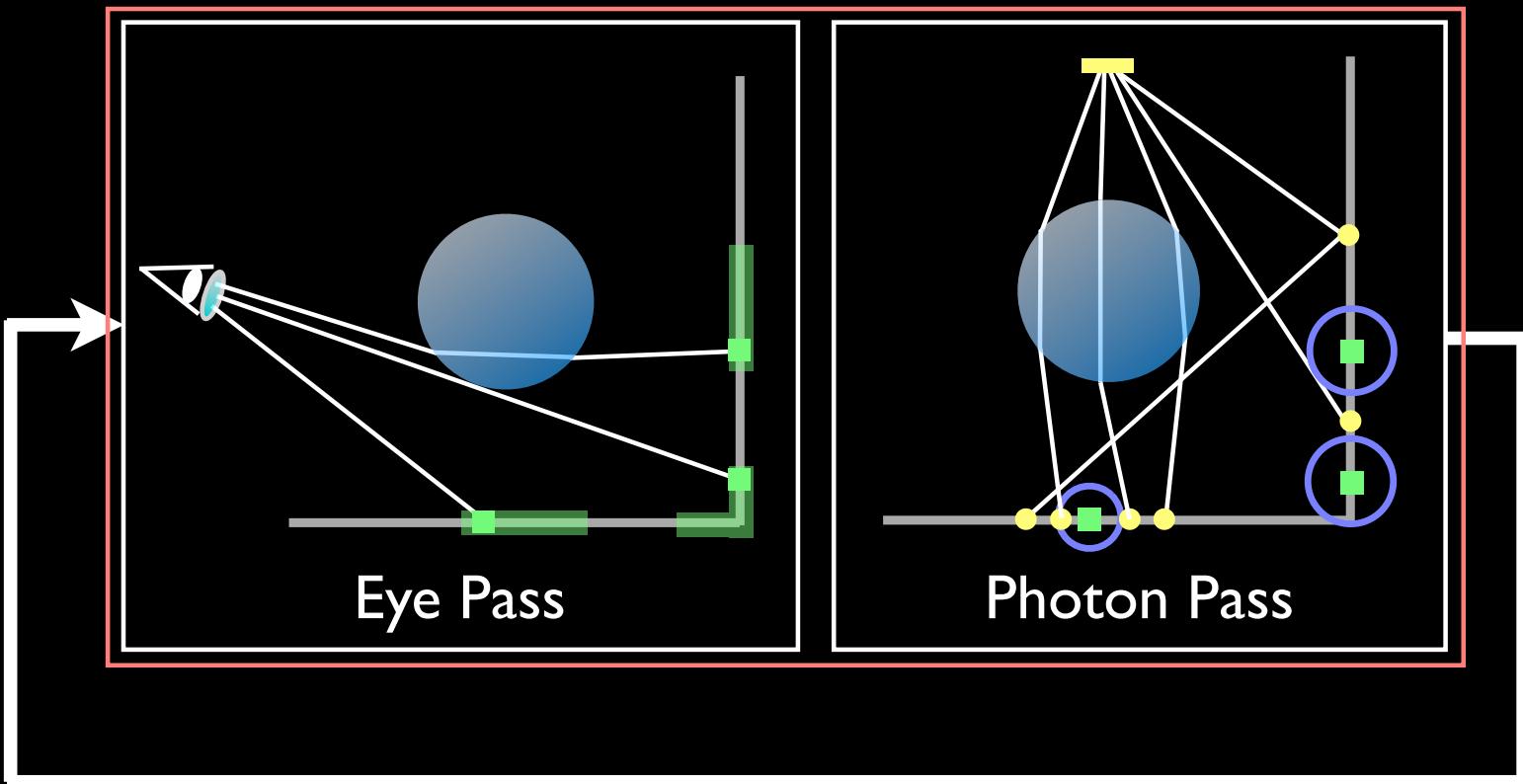
Stochastic PPM



Stochastic PPM



Stochastic PPM



Stochastic Progressive Density Estimation

$$L_i(S, \vec{\omega}) = \frac{\tau_i(S, \vec{\omega})}{\pi R_i(S)^2 N_e(i)}$$

$$\boxed{\lim_{i \rightarrow \infty} L_i(S, \vec{\omega}) = L(S, \vec{\omega})}$$

Provable convergence to
average photon density over a region S

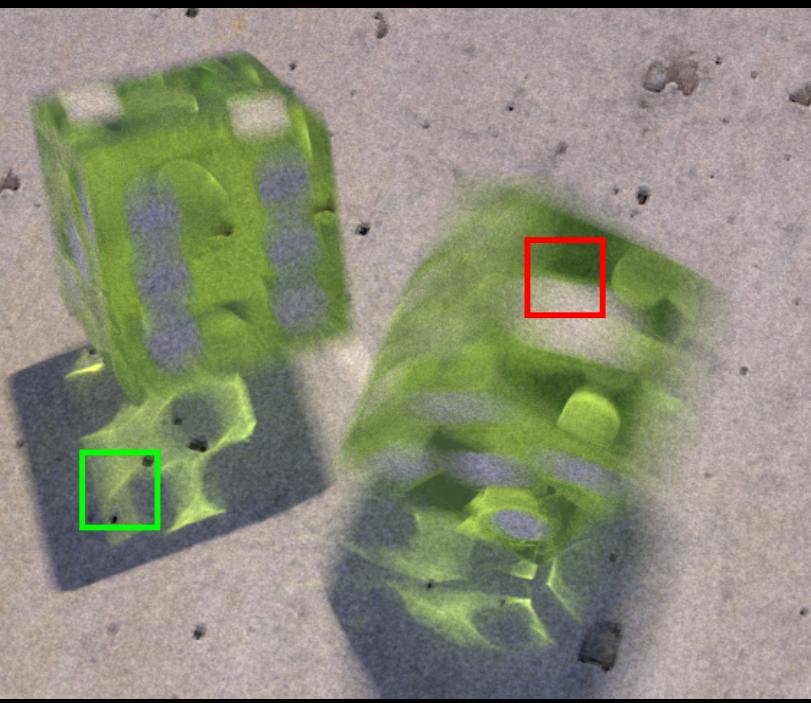
Bidirectional Path Tracing



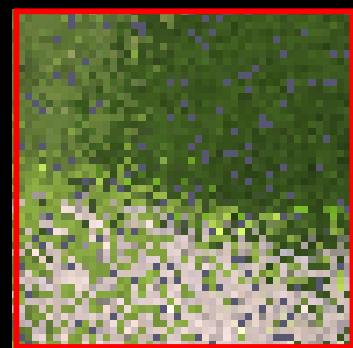
Stochastic PPM



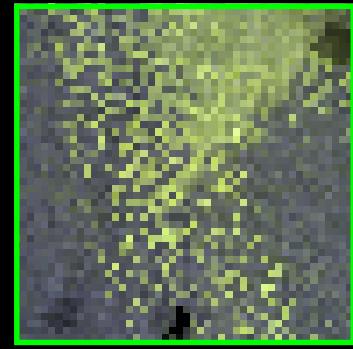
Motion Blur



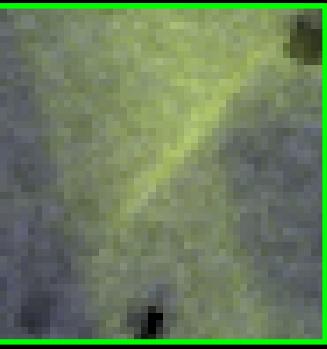
Equal time, Equal memory



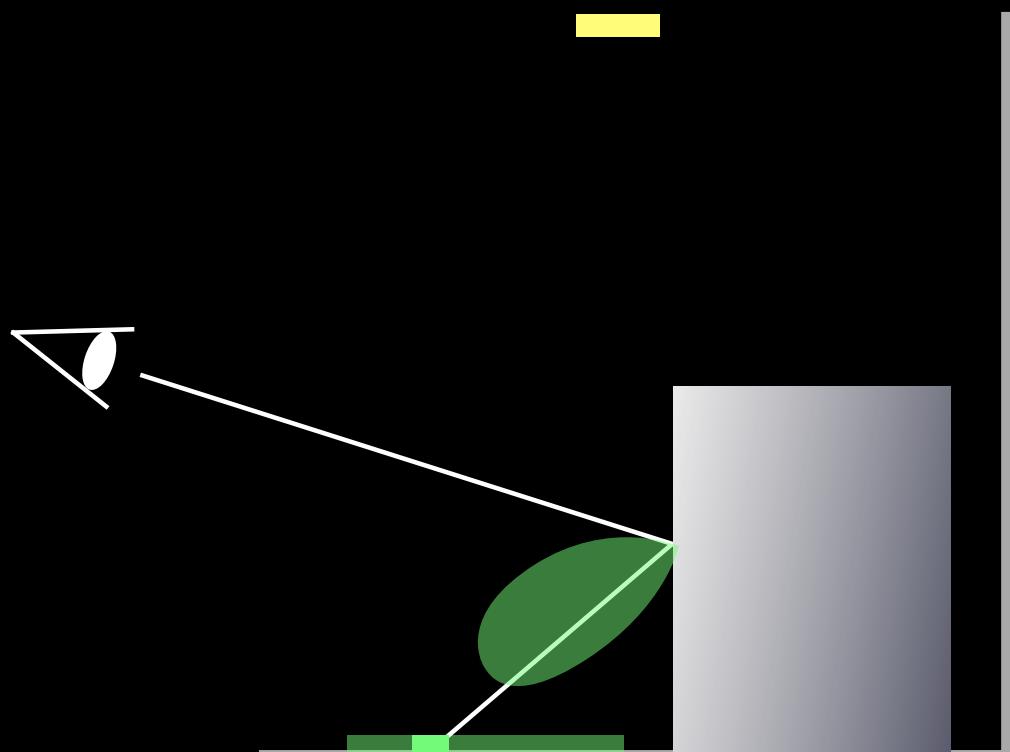
PPM



SPPM

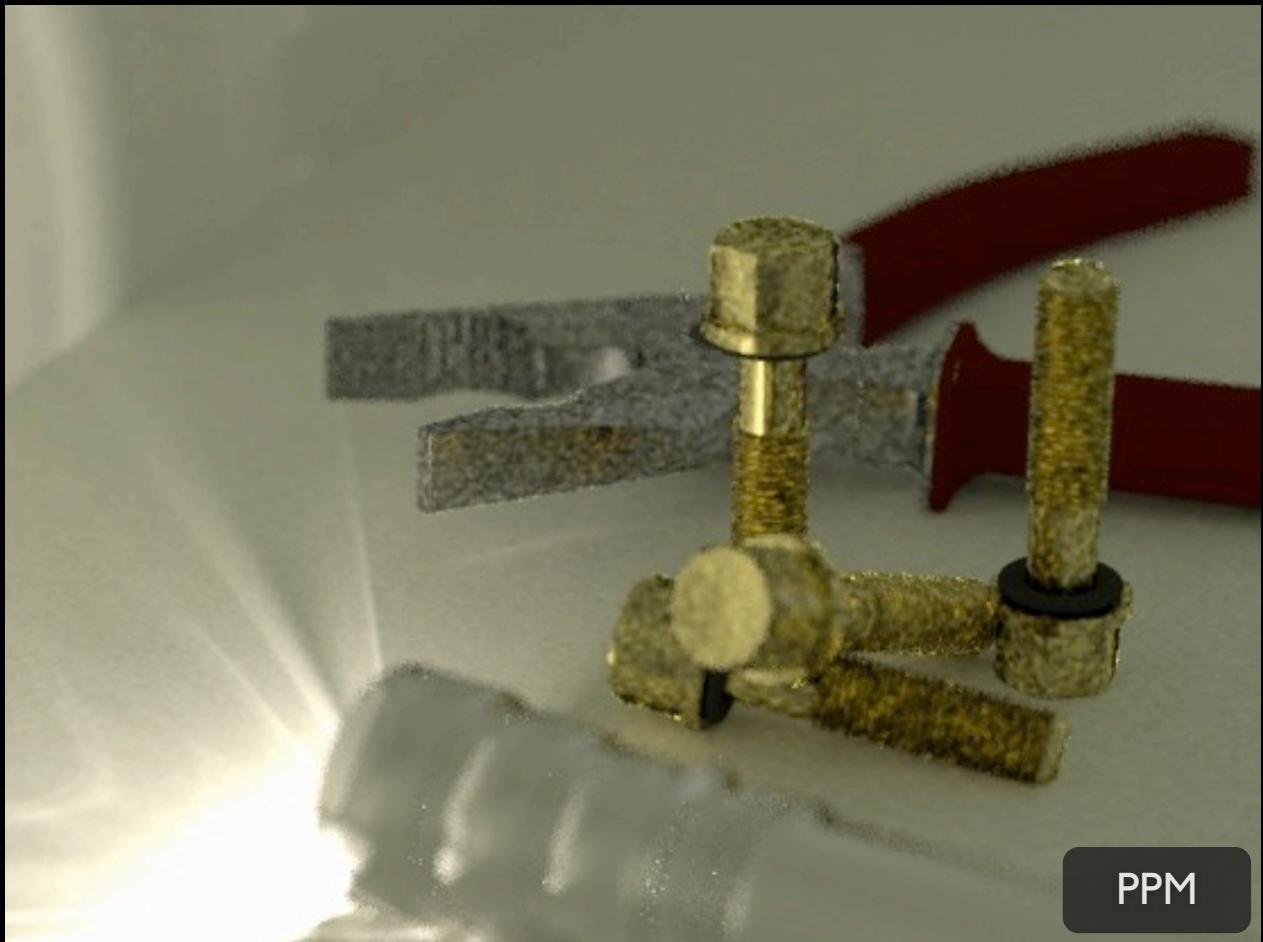


Glossy Materials with SPPM

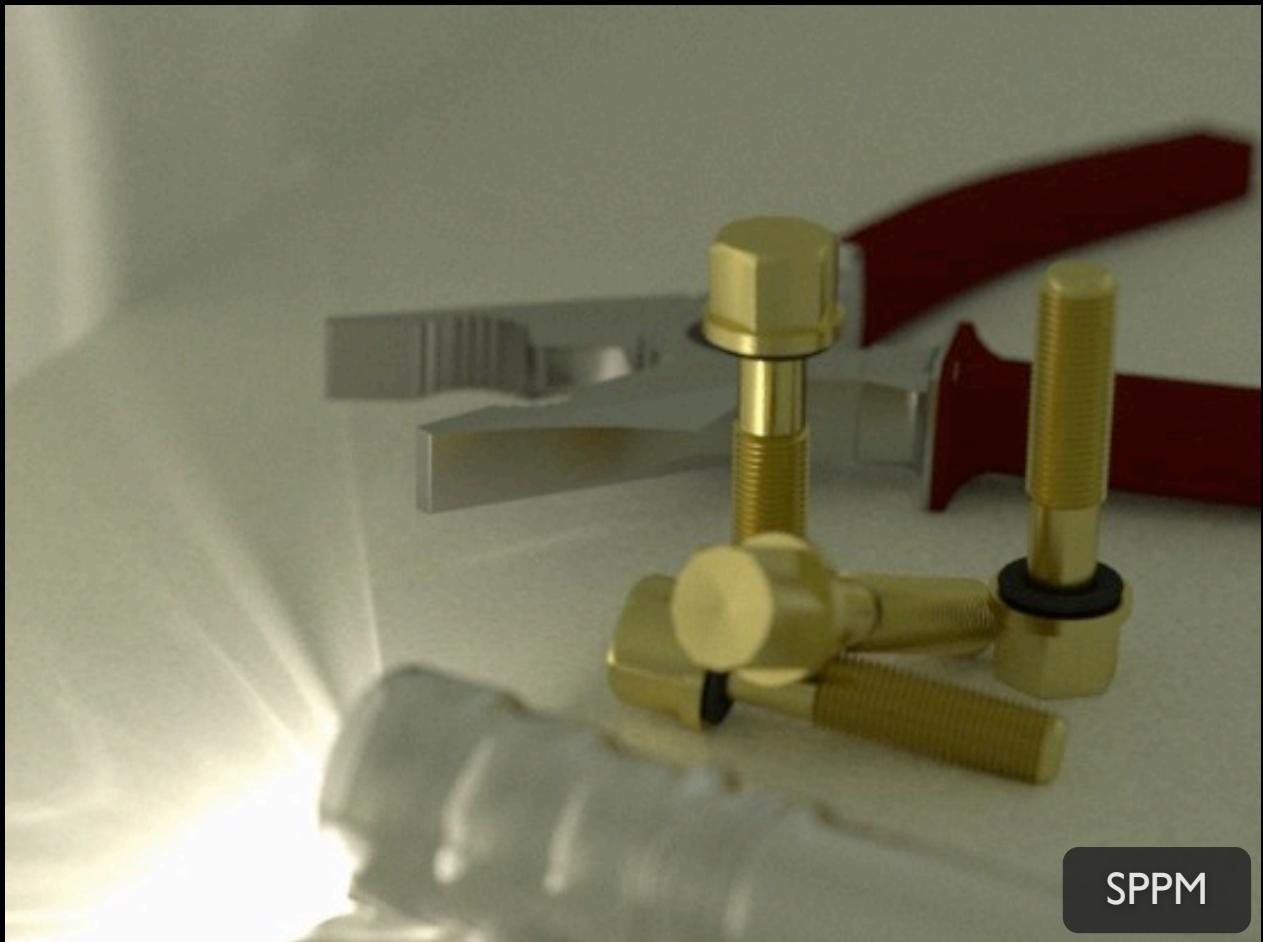


Trace one bounce rays

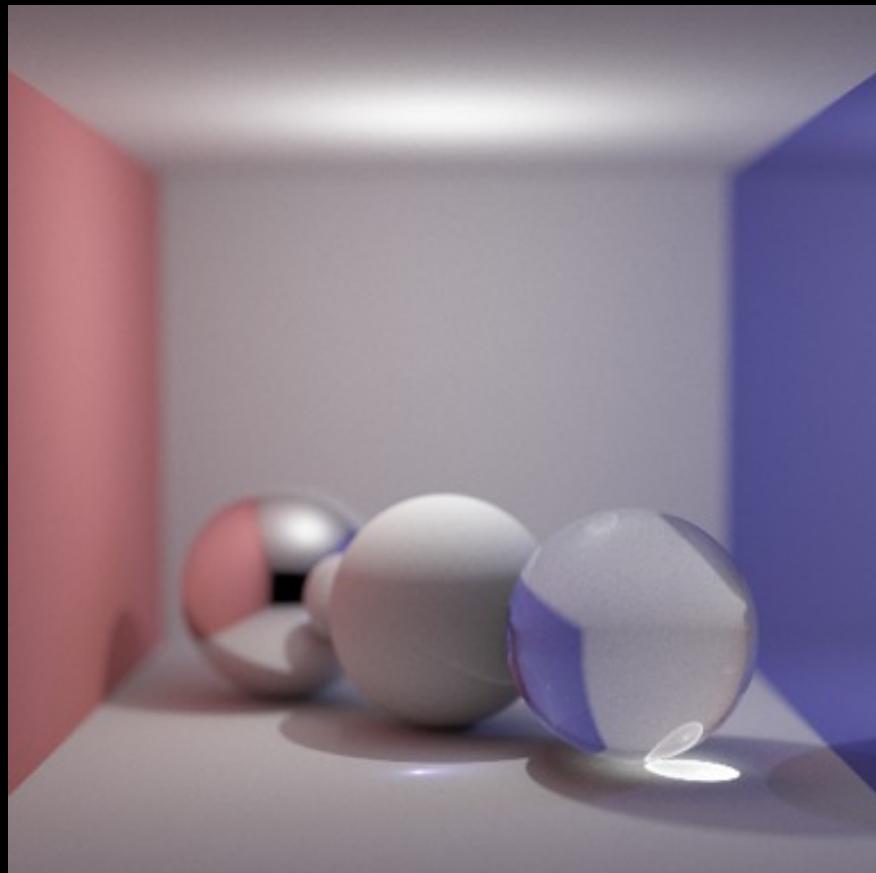
DOF + Glossy Reflection + Caustics



DOF + Glossy Reflection + Caustics



GPUSPPM



cs.au.dk/~toshiya/gpusppm.zip

How much computation is enough?

A Progressive Error Estimation Framework for Photon Density Estimation

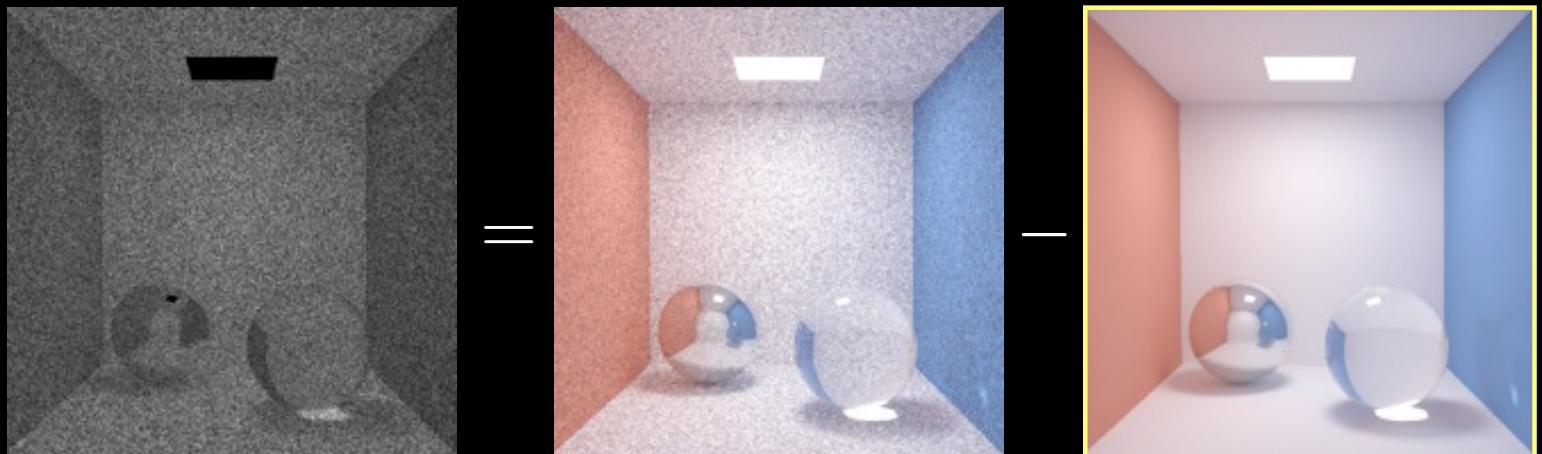
Toshiya Hachisuka Wojciech Jarosz Henrik Wann Jensen

Presented & Published at SIGGRAPH/SIGGRAPH Asia 2010

Definition of Error

- Difference between computed and exact

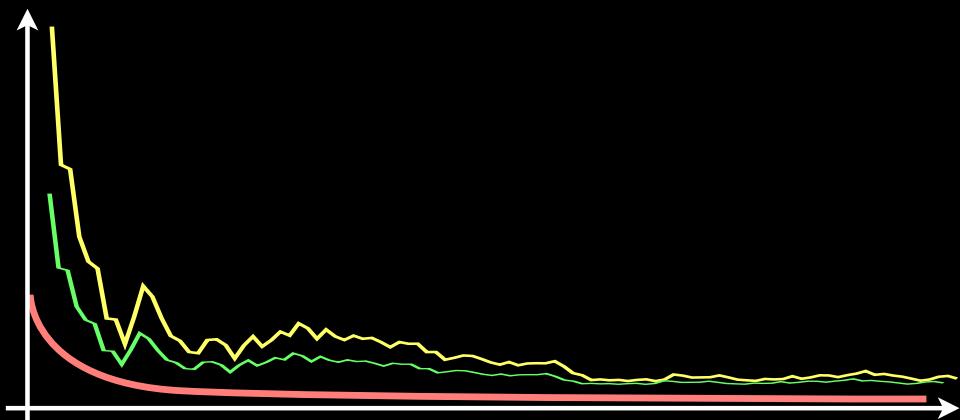
$$E_i = L_i - \boxed{L} \text{ Unknown}$$



Decomposition of Error

- Bias-Noise decomposition

$$E_i = L_i - L = B_i + N_i$$



Stochastic Error Bound Derivation

$$E_i = L_i - L = B_i + N_i$$

Stochastic error bound User-defined Probability

$$\boxed{P(|E_i| \leq E_{b,i})} \leq \boxed{1 - \beta}$$

$$E_{b,i} = C_{i,1-\frac{\beta}{2}} \sqrt{\frac{\text{Variance}}{i}} + |B_i|$$

Stochastic Error Bound Derivation

$$E_i = L_i - L = B_i + N_i$$

$$P(|E_i| \leq E_{b,i}) \leq 1 - \beta$$

$$E_{b,i} = \boxed{C_{i,1-\frac{\beta}{2}} \sqrt{\frac{\text{Variance}}{i}} + |B_i|}$$

Error due to Noise

Stochastic Error Bound Derivation

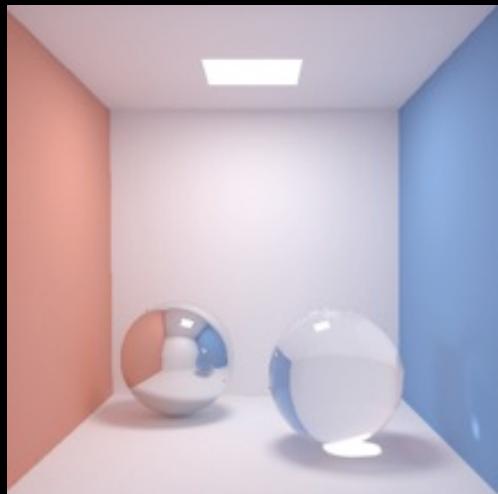
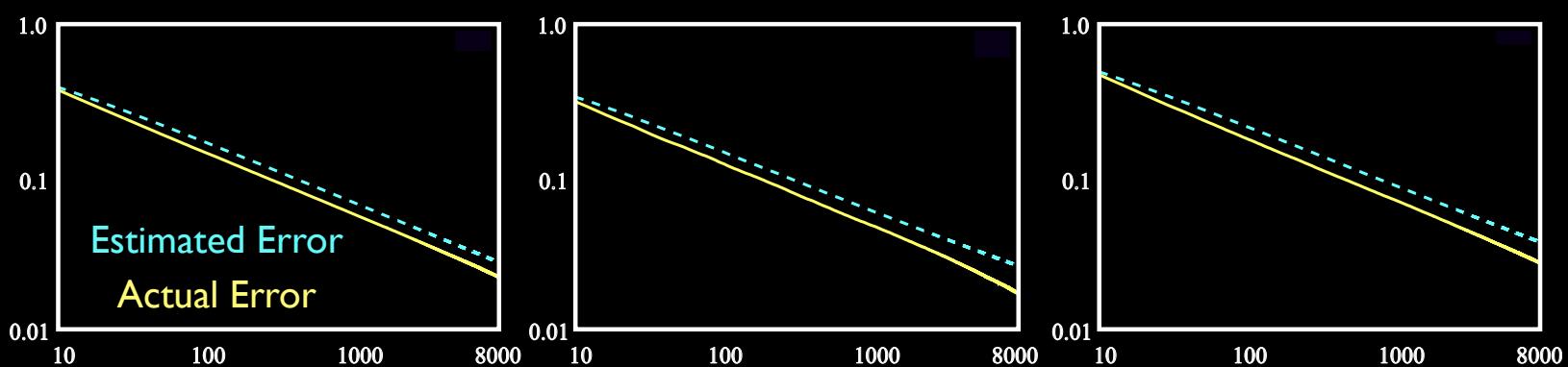
$$E_i = L_i - L = B_i + N_i$$

$$P(|E_i| \leq E_{b,i}) \leq 1 - \beta$$

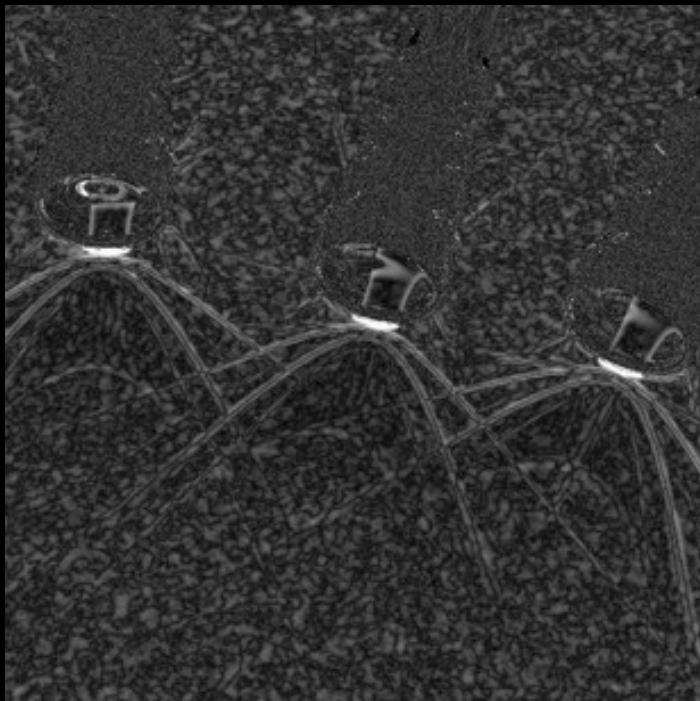
$$E_{b,i} = C_{i,1-\frac{\beta}{2}} \sqrt{\frac{\text{Variance}}{i}} + |B_i|$$

Error due to Bias

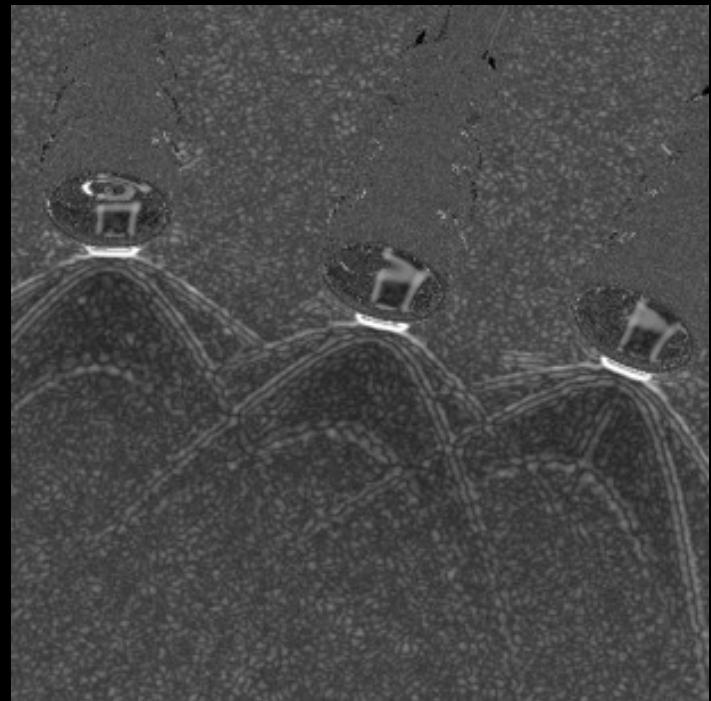
Error Estimation



Error Estimation

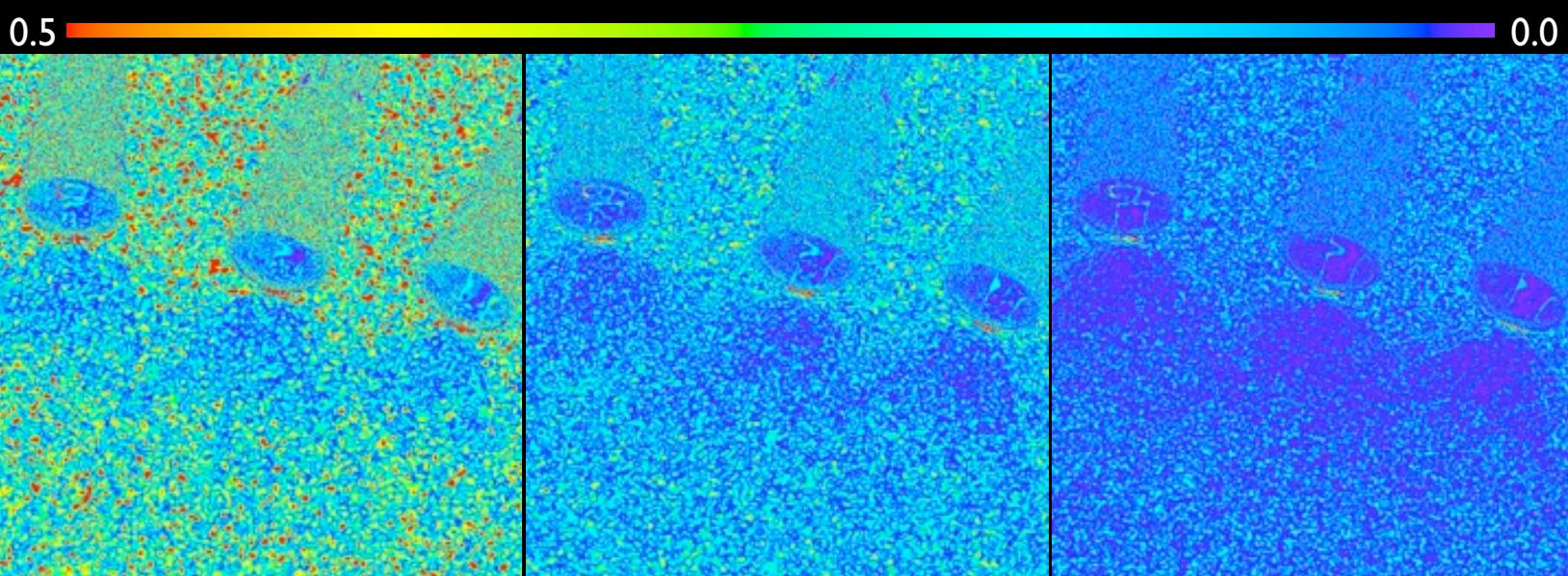


Actual Error



Estimated Error Bound

Automatic Rendering Termination



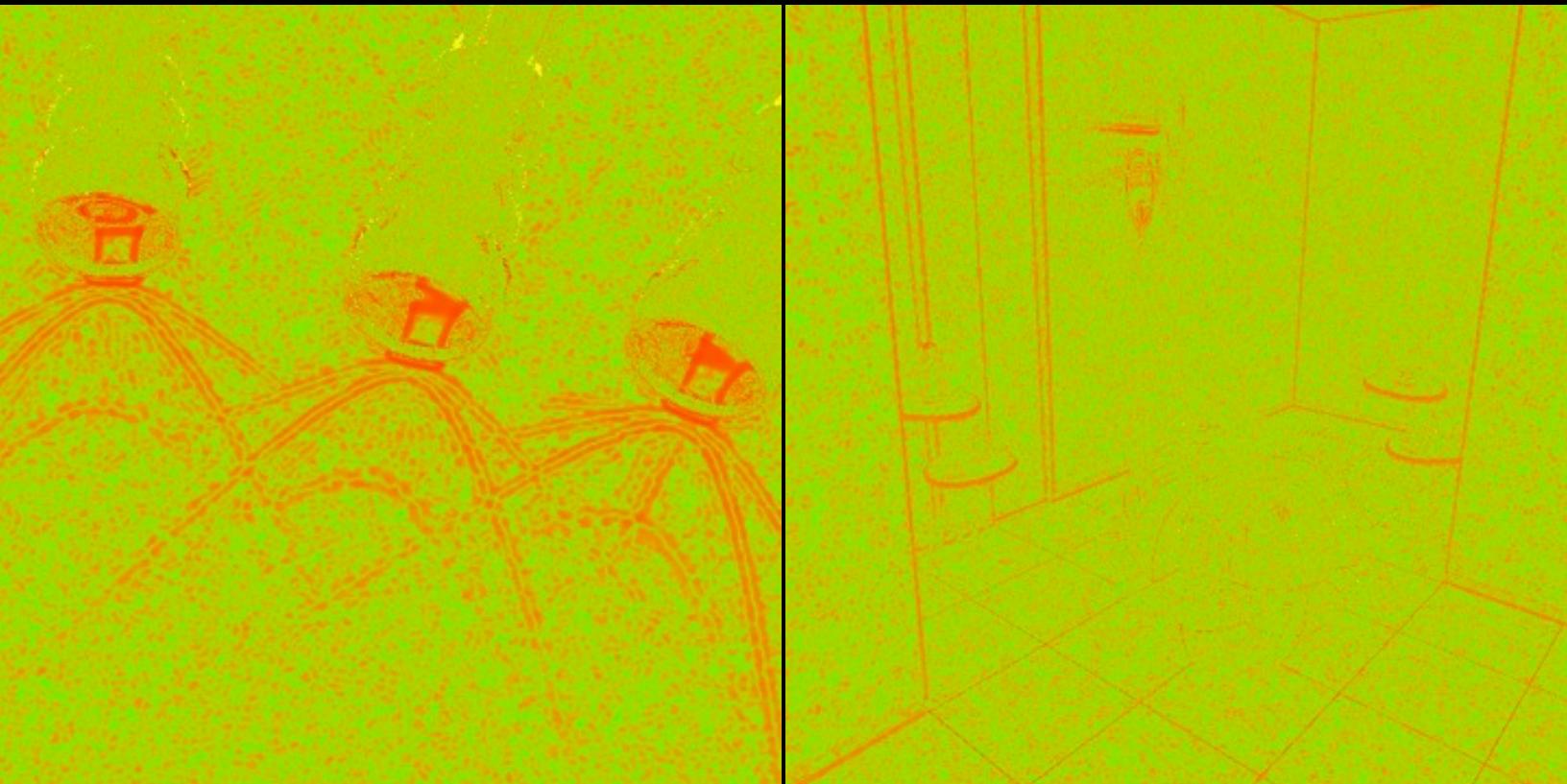
specified: 0.25
actual: 0.1916

specified: 0.125
actual: 0.09294

specified: 0.0625
actual: 0.04482

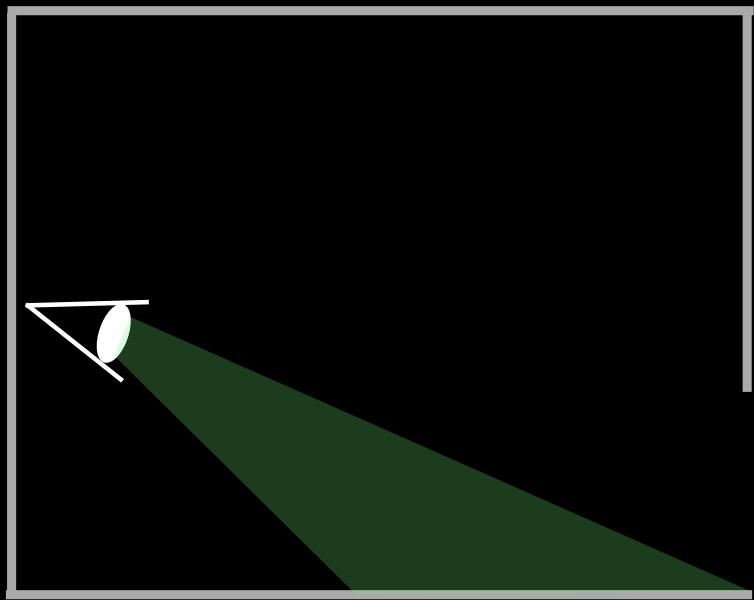
1.3 times overestimation on average

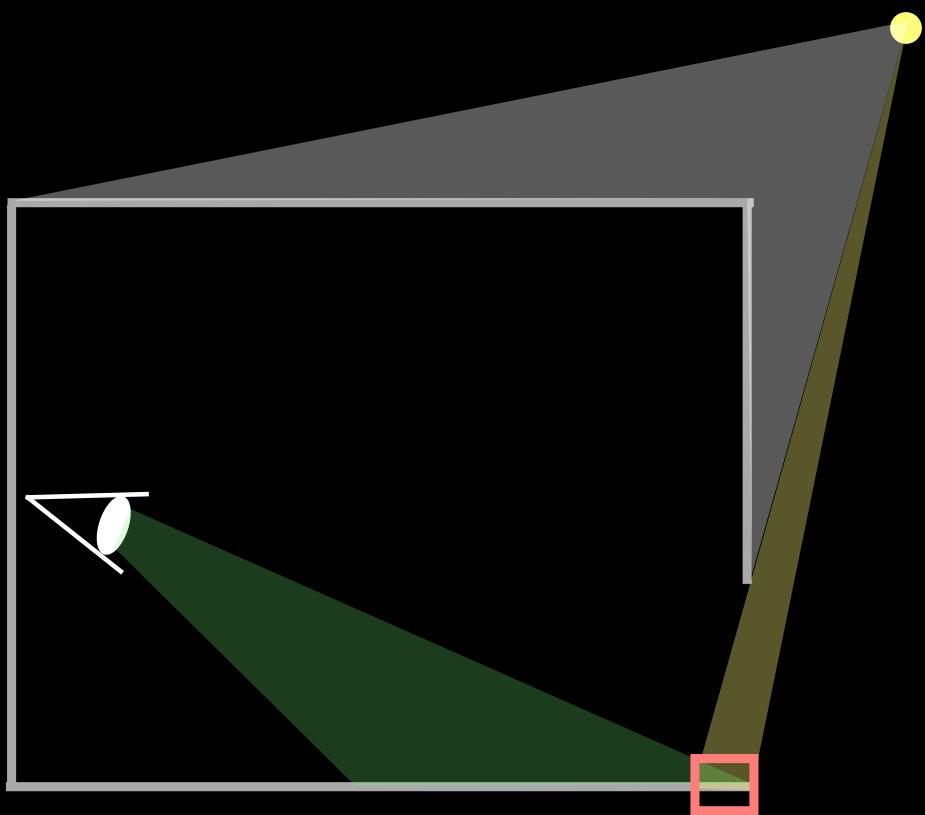
Noise-Bias Ratio



Inefficient Case





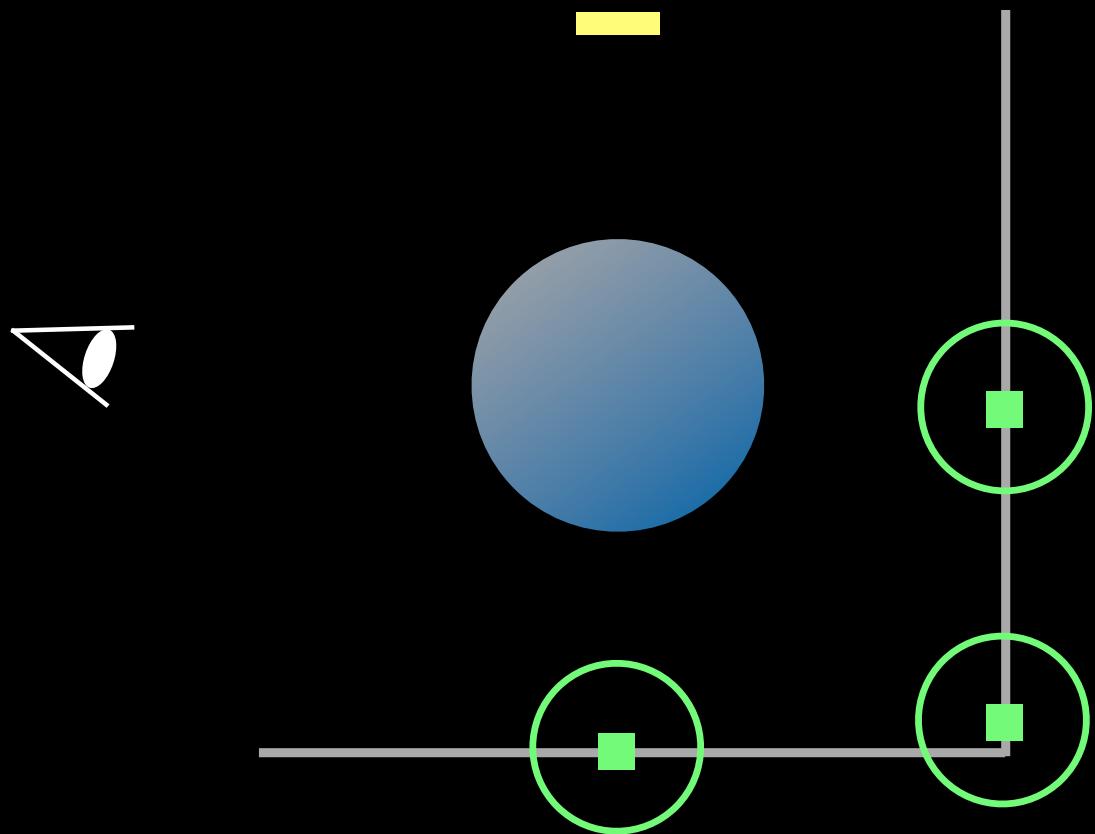


Robust Adaptive Photon Tracing using Photon Path Visibility

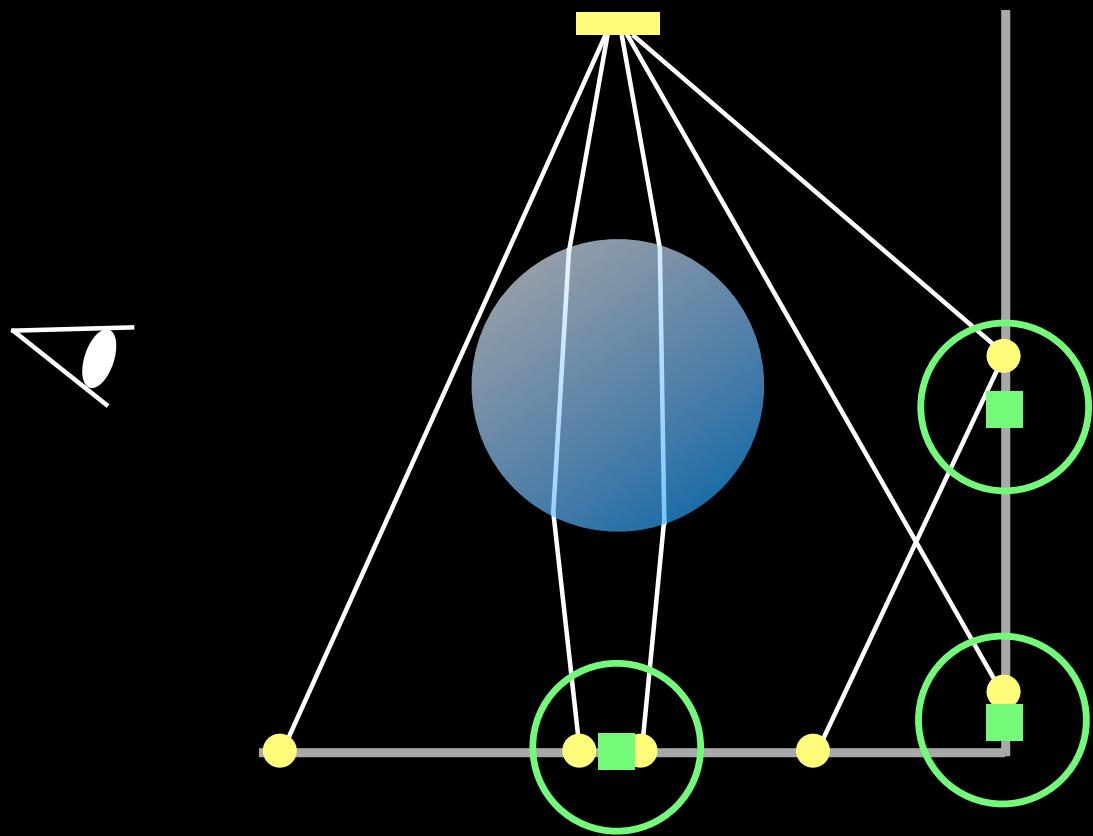
Toshiya Hachisuka Henrik Wann Jensen

Published in ACM Transaction of Graphics
(to be presented at SIGGRAPH 2013)

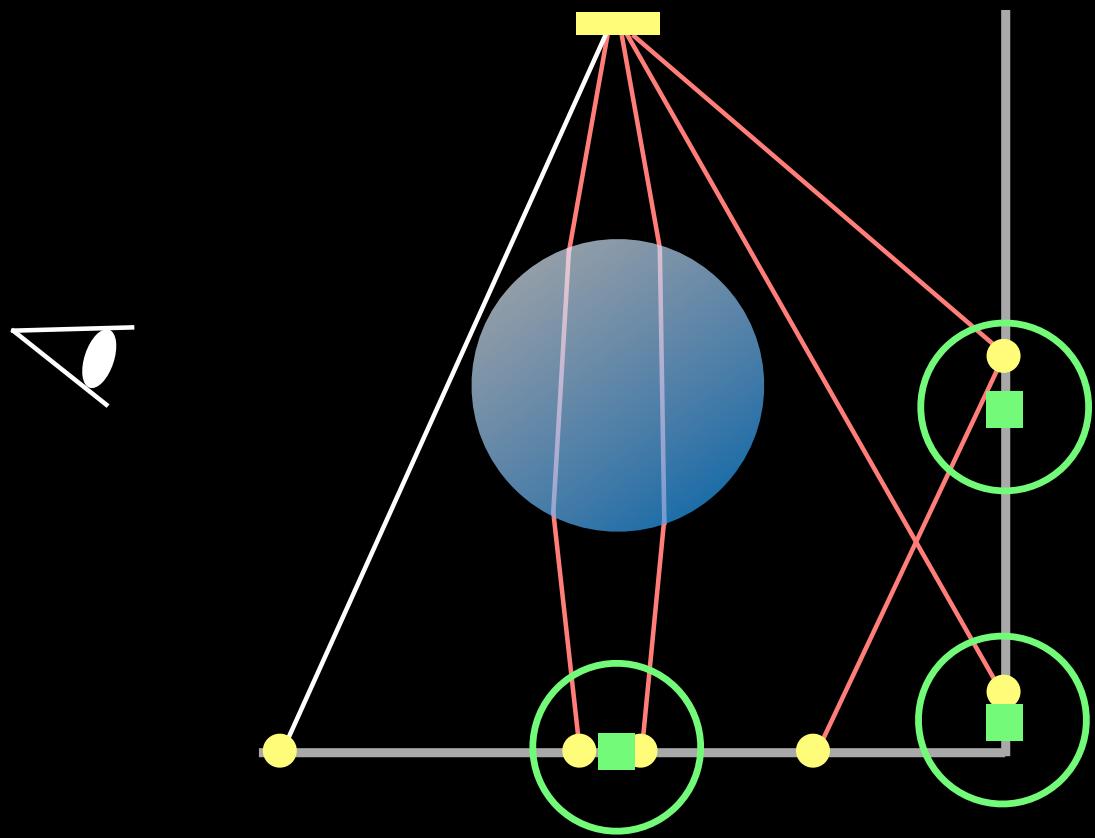
Photon Tracing using Visibility



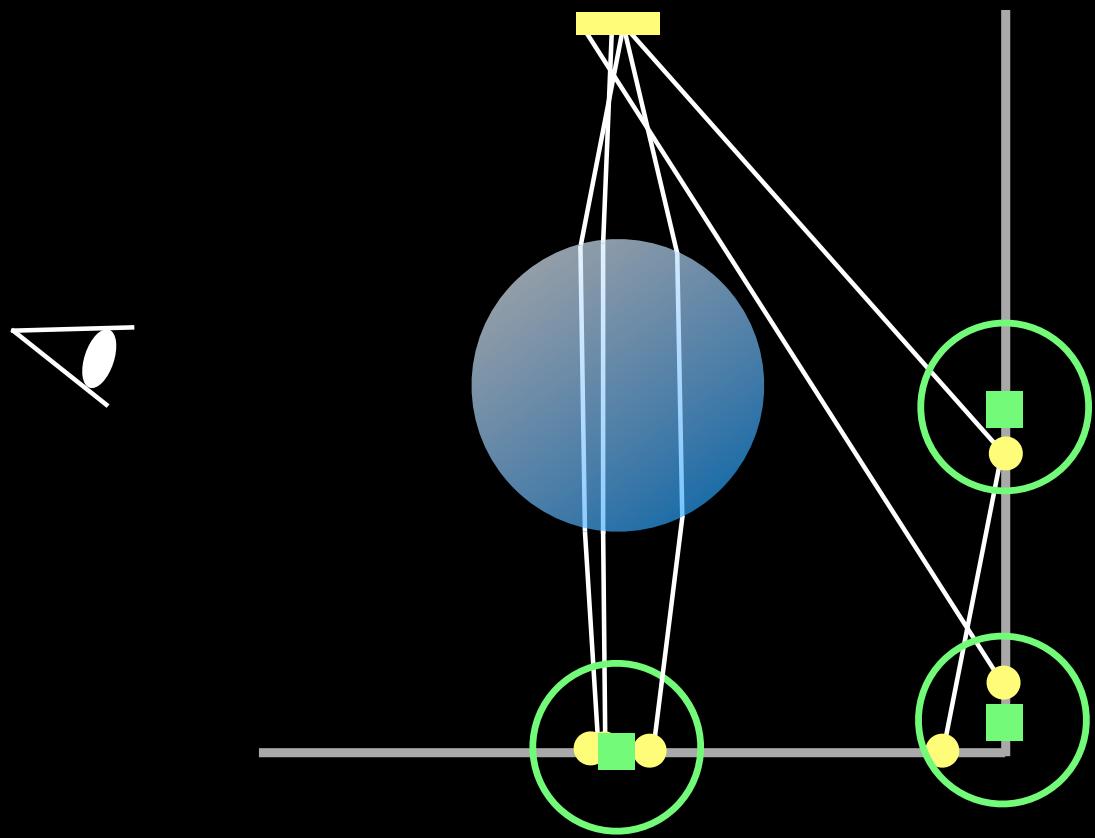
Photon Tracing using Visibility



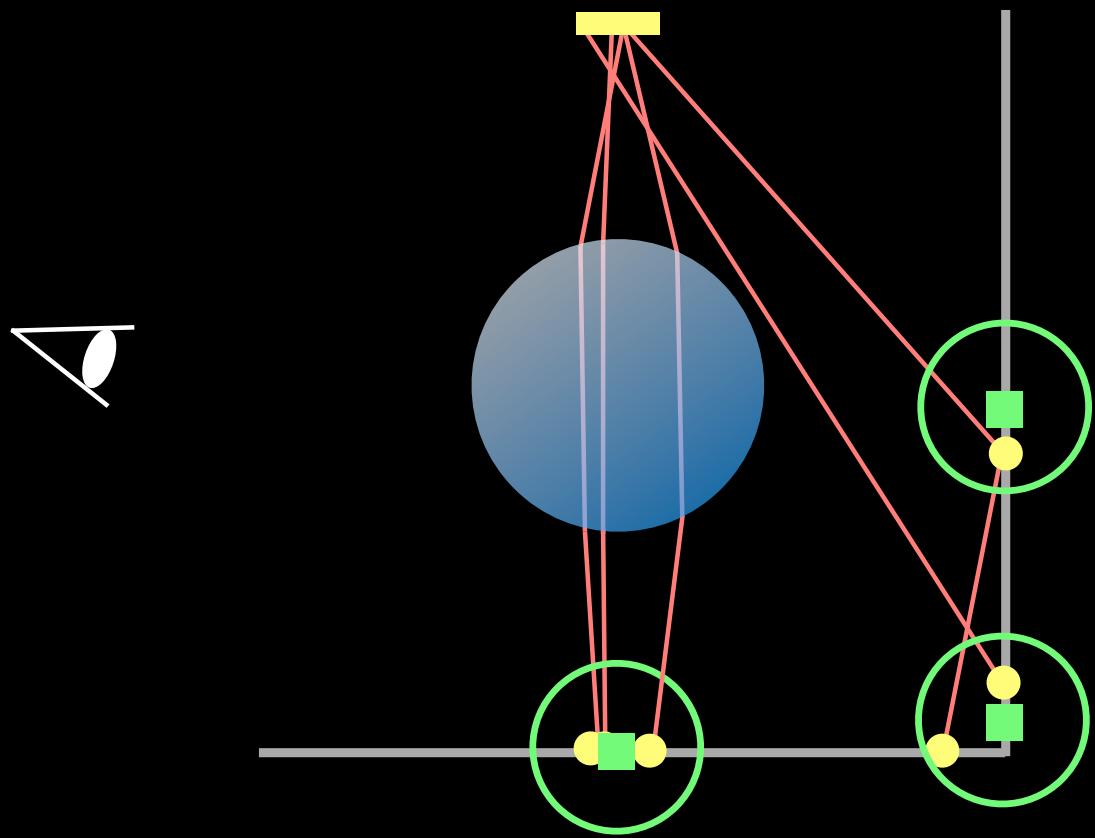
Photon Tracing using Visibility



Photon Tracing using Visibility



Photon Tracing using Visibility



Approach

- Consider space of random numbers $\vec{u} \in (0, 1)^N$
- Photon path visibility function

If the photon is not visible:

$$V(\vec{u}) = 0$$

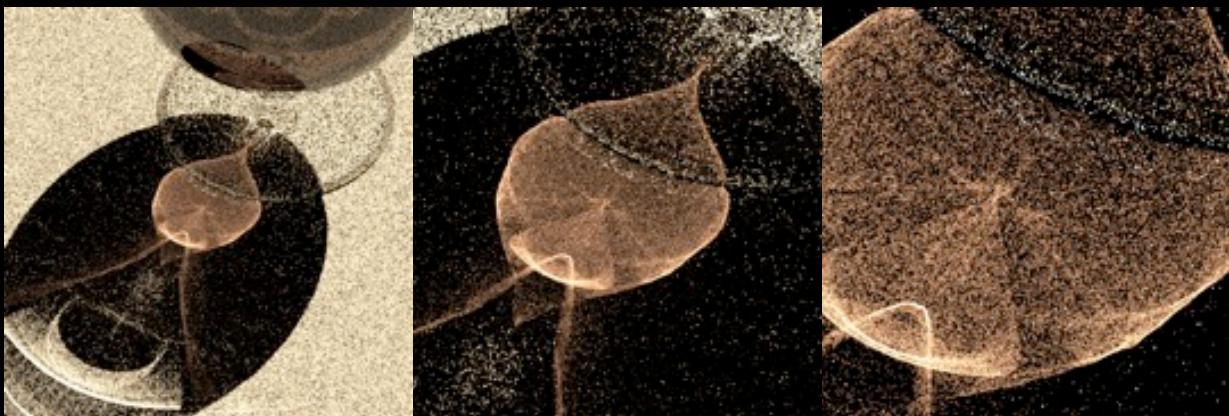
If the photon is visible:

$$V(\vec{u}) = 1$$

Sample $V(\vec{u})$ using Markov chain Monte Carlo Methods

Small-scale Lighting Details

Uniform



Adaptive



Automatic Parameter Tuning



Value is too small

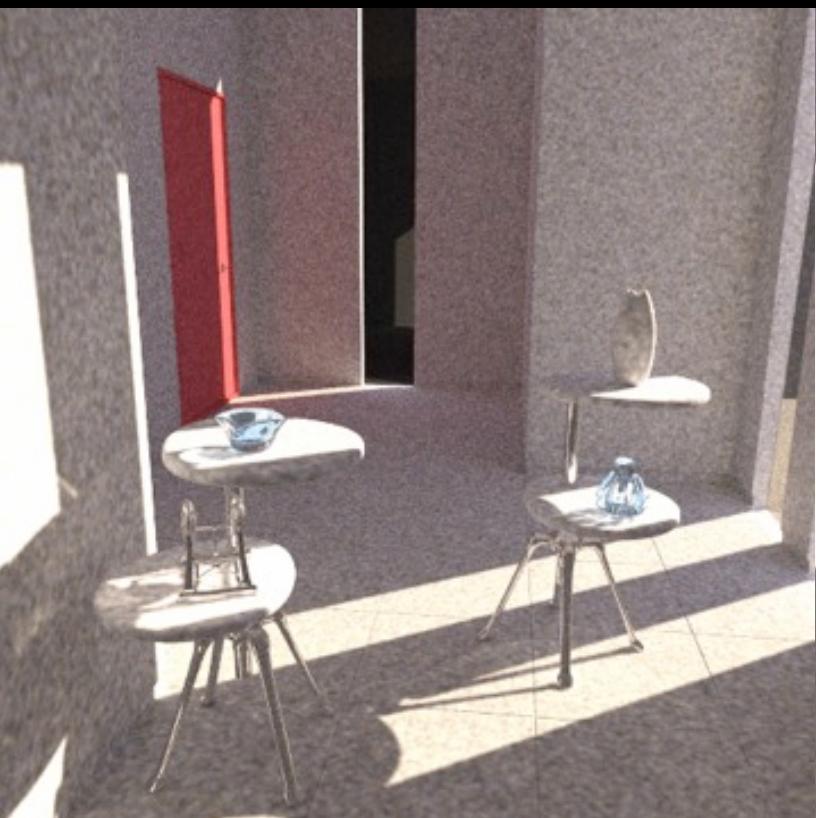


Automatic

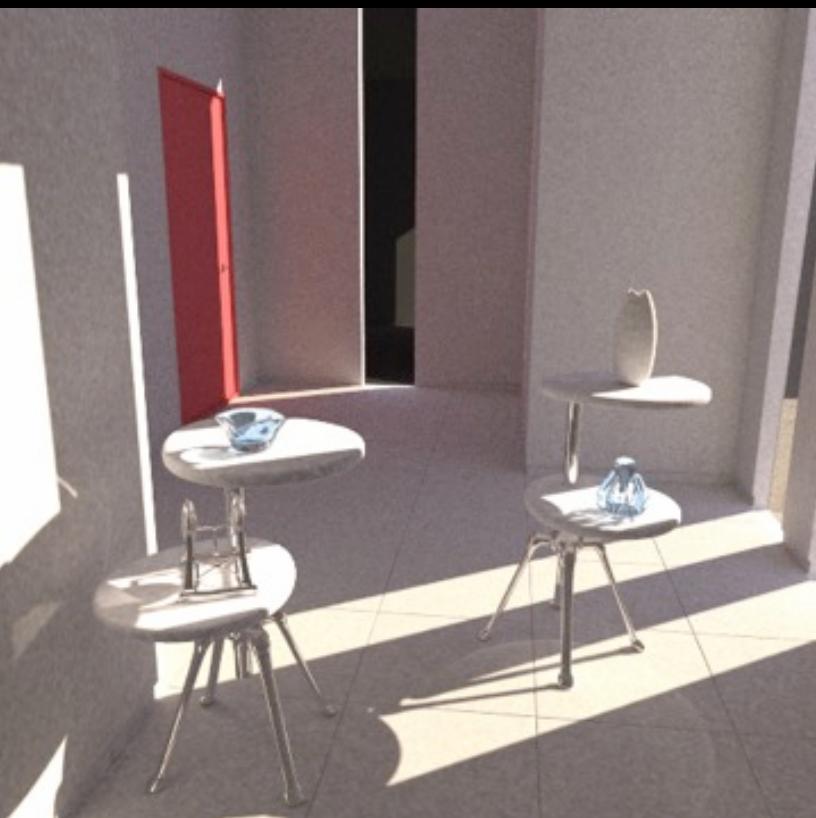


Value is too large

Sunlit Room



Uniform

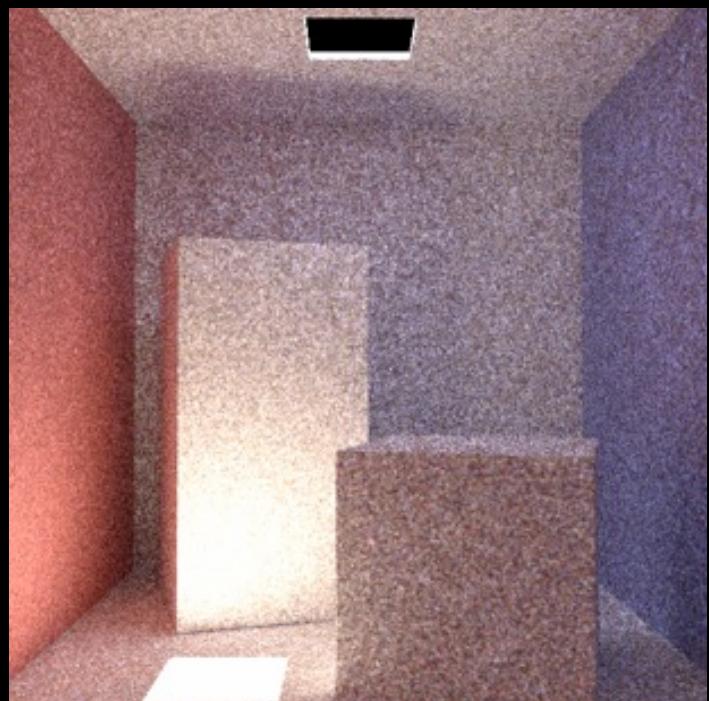


Adaptive

Another Example



Uniform



Adaptive

Another Example



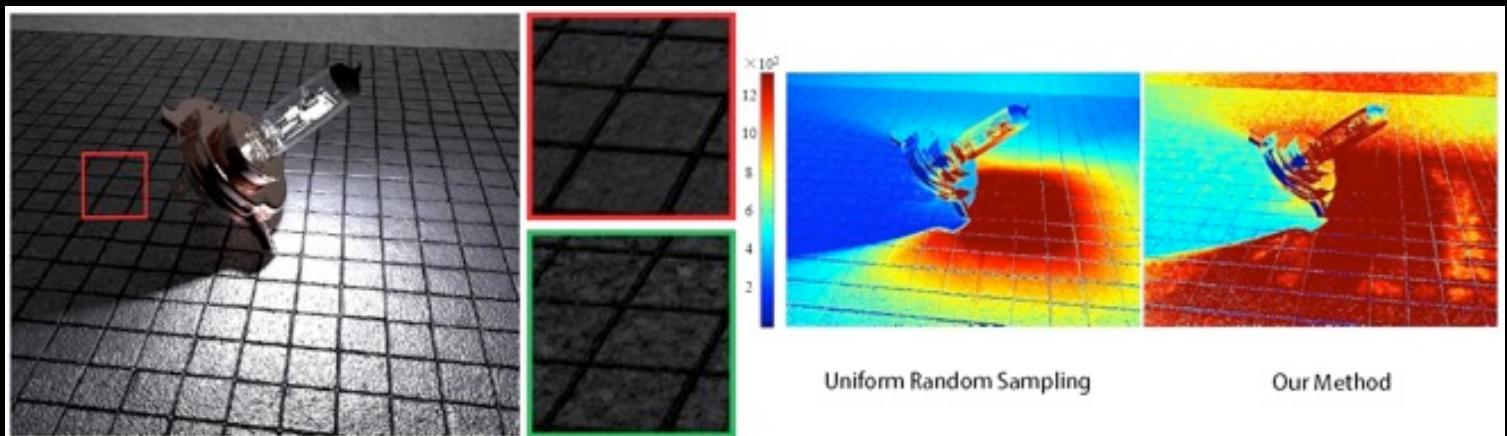
Uniform



Adaptive

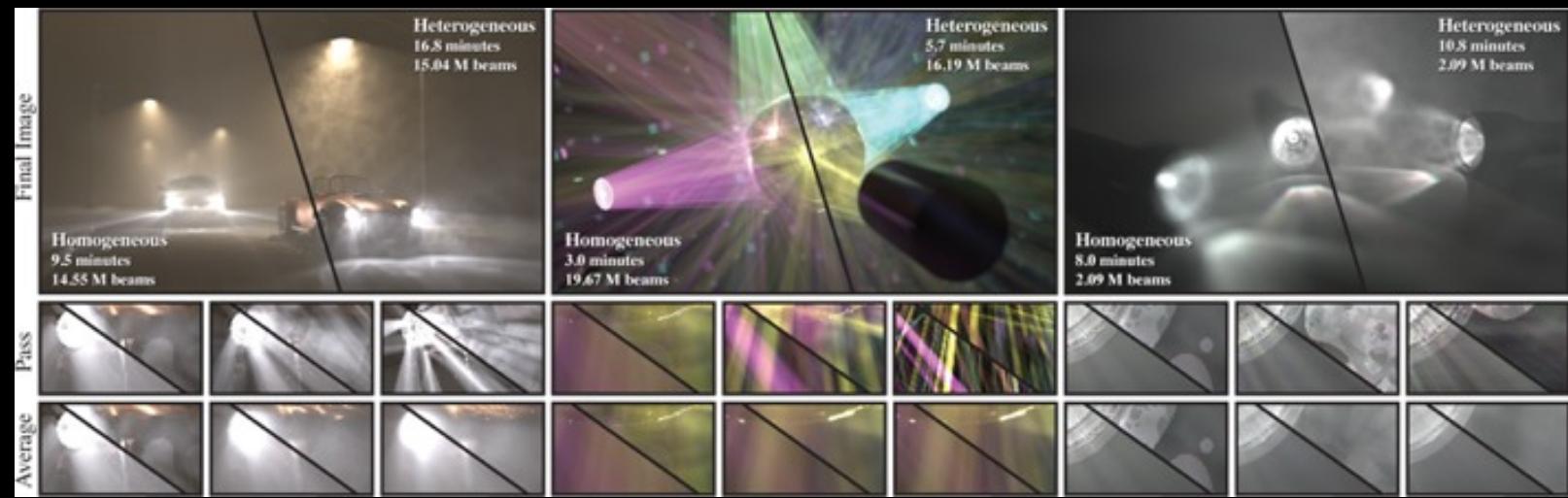
Some Related Work

- Adaptive photon tracing based on photon density on the image [Chen et al. 2011]



Some Related Work

- Progressive photon beams [Jarosz et al. 2011]



Some Related Work

- Efficient rendering of dynamic scenes
[Weiss and Grosch 2012]



Some Related Work

- Combine density estimation and MC integration
[Hachisuka et al. 2012]



FAQs

- Q: Is PPM unbiased?

FAQs

- Q: Is PPM unbiased?
- A: It is biased and consistent, but does not matter in practice.

$$E[X] = \lim_{N \rightarrow \infty} \sum_{i=1}^N x_i$$

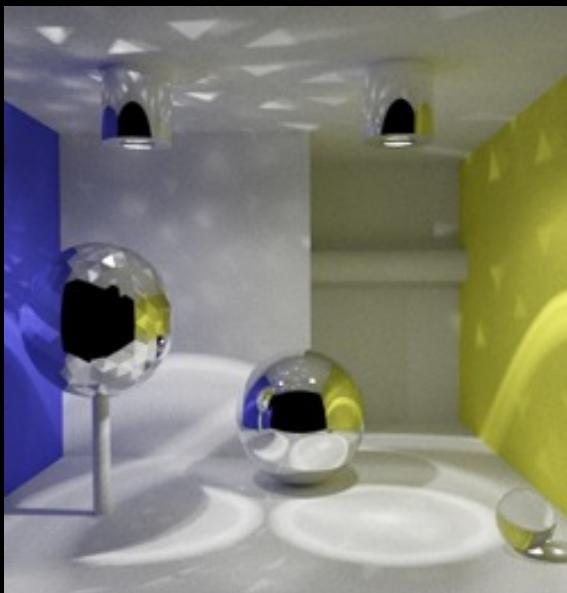
BOTH unbiased and consistent methods need inf. samples!

FAQs

- Q: Do we still use global + caustics separation?

FAQs

- Q: Do we still use global + caustics separation?
- A: Just render everything as one.

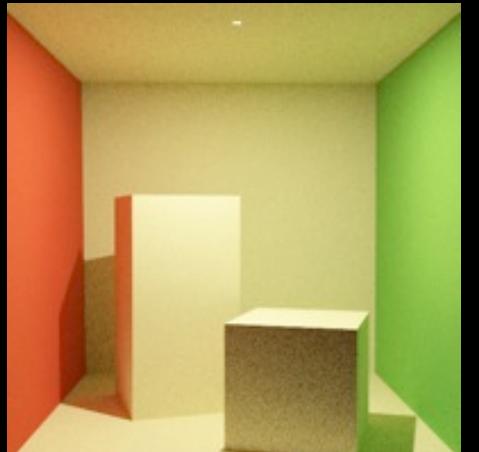


FAQs

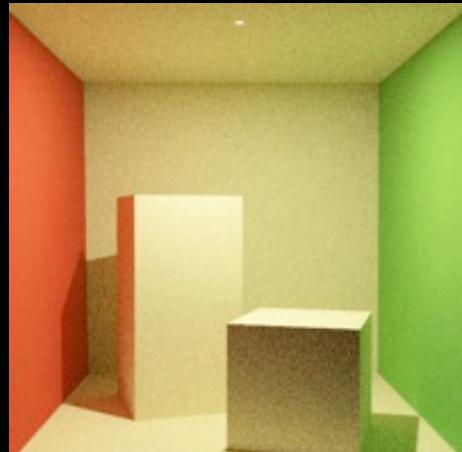
- **Q:** Is PPM slower for diffuse scenes than other methods?

FAQs

- **Q:** Is PPM slower for diffuse scenes than other methods?
- **A:** True, but not much, and you can do more.



PT



PPM

Summary

- SPPM = PPM + Distributed Ray Tracing
 - Error estimation is available
 - Adaptive photon tracing based on visibility
 - Lots of interesting extensions
-
- My opinion: PPM + Extensions is the hardest rendering algorithm to “break” at the moment

cs.au.dk/~toshiya



Next Talk

- Probabilistic formulation of PPM
- “How to turn your PM into PPM in a minute!”

Probabilistic Formulation of Photon Density Estimation

Probabilistic PPM

Matthias Zwicker
University of Bern, Switzerland



Probabilistic PPM

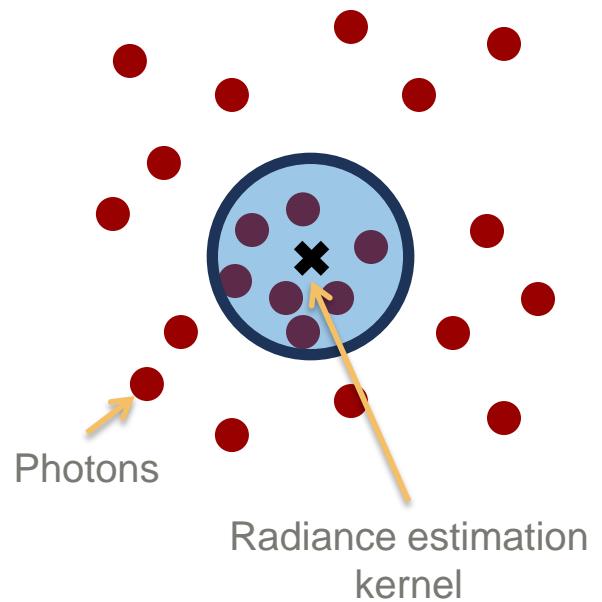
- Alternative derivation of PPM
- Fixed radius reduction, no need for statistics
- Asymptotic convergence analysis
- Trivial to implement
- General radiance estimates (volumes, beams,...)
- General kernels



Bias-variance trade-off

- Larger kernels
 - Lower variance
 - Higher bias
- Smaller kernels
 - Higher variance
 - Lower bias
- Vanishing variance and bias
 - Infinitely many photons
 - Infinitely small kernels

Radiance estimation





Progressive photon mapping

- Breaking the variance-bias trade off

Basic algorithm

- Iterate over photon mapping steps
- Reduce kernel size in each step
- Accumulate results

Advantages

- Bias and variance vanish over iterations
- Avoid memory bottleneck



Progressive photon mapping

What's the right strategy to reduce kernel sizes?

Hachisuka et al. [SIGGRAPH Asia 2008]

- Reduce kernel based on empirical statistics

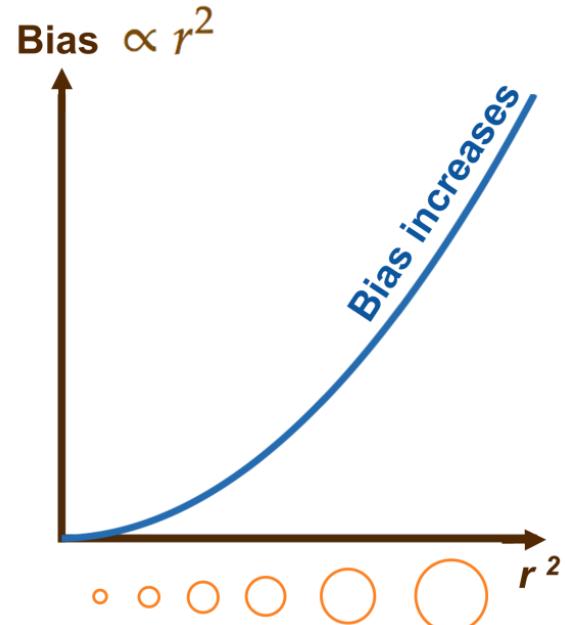
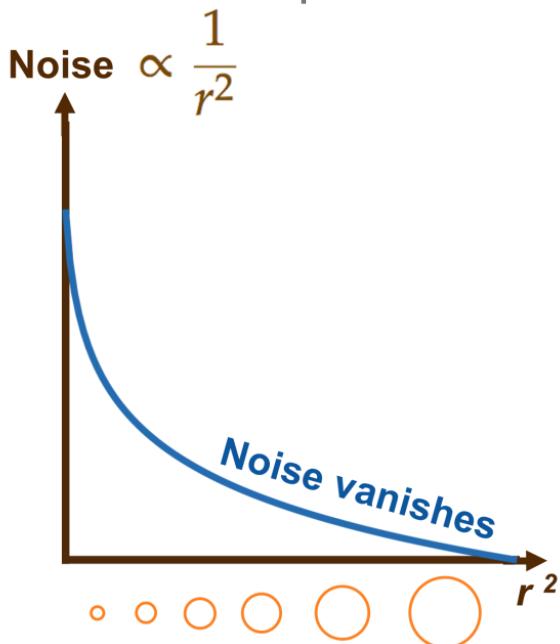
Our approach [ACM TOG 2011]

- “Blindly” reduce kernel by fixed factor



Probabilistic analysis

- Consider output of radiance estimation as random variable





Radius reduction

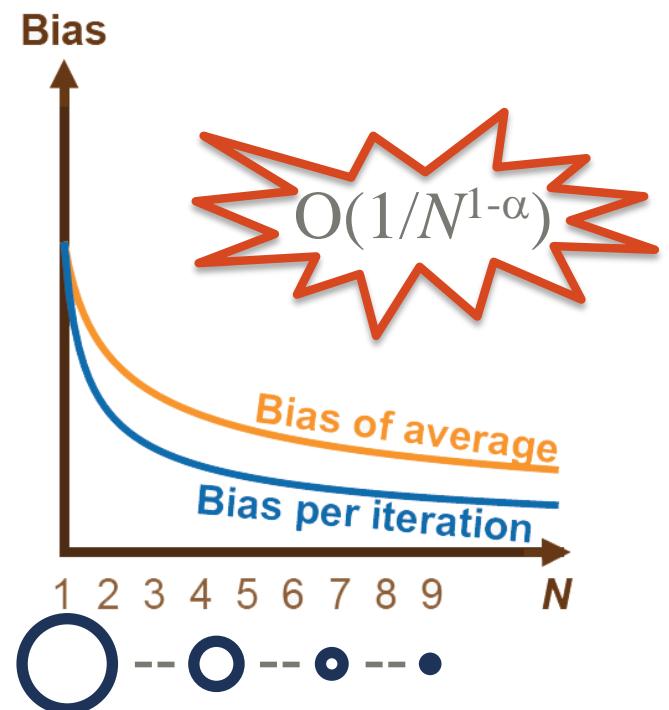
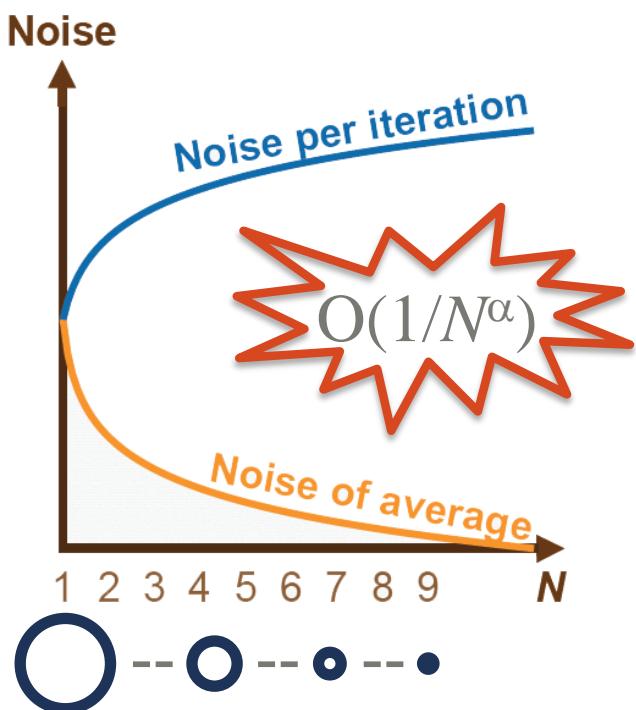
- Iteration step i
- Radiance estimation radius r_i
- Parameter $0 < \alpha < 1$

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

Theory and derivation [Knaus and Zwicker 2011]



Convergence analysis





Radius reduction

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!



Mean squared error (MSE)

- $\text{MSE} = \text{Variance} + \text{Bias}^2$
- Asymptotically, $\text{MSE} = O(1/N^\alpha + 1/N^{2-2\alpha})$
- Minimize wrt. α : $\text{MSE} = O(1/N^{2/3})$ for $\alpha = 2/3$



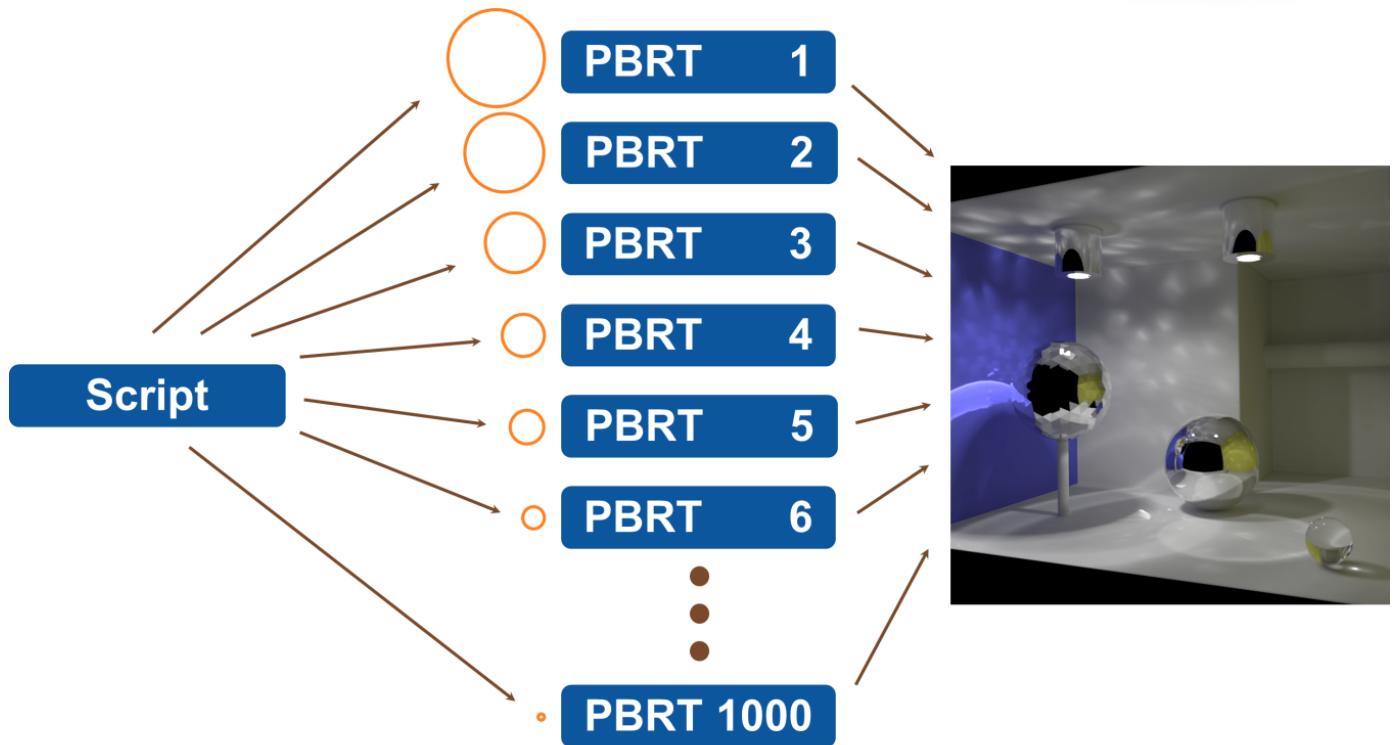
Implementation

Photon Mapper

Black Box



Implementation



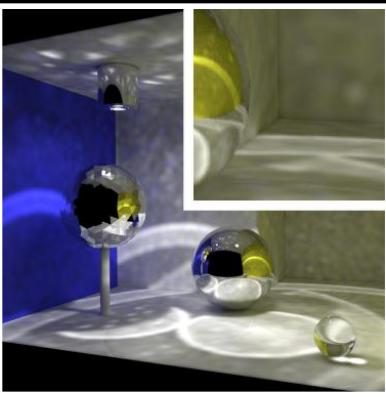


Image 1

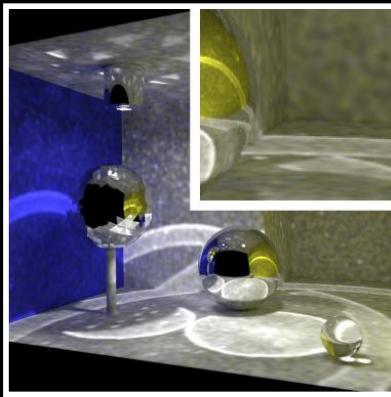
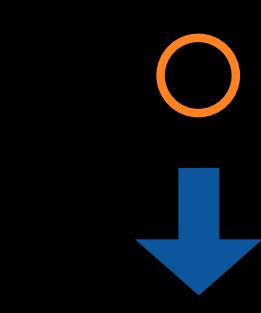


Image 10

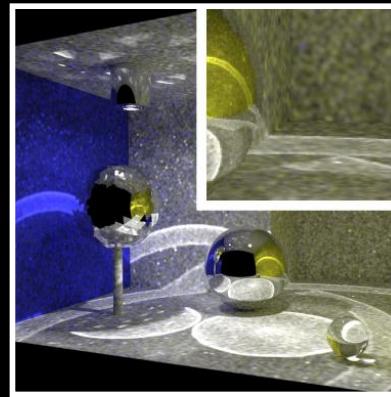
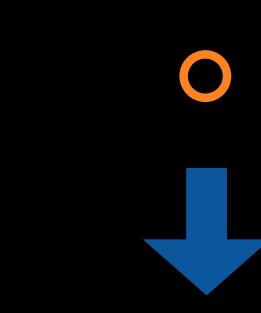


Image 100

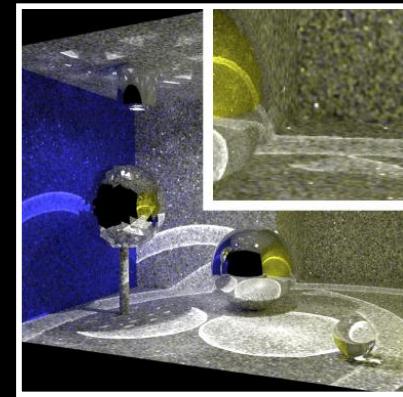
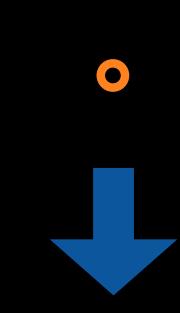
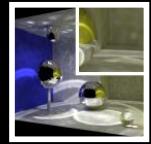


Image 1000



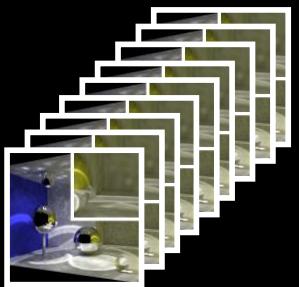
1



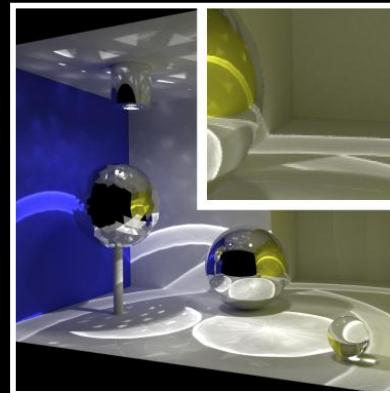
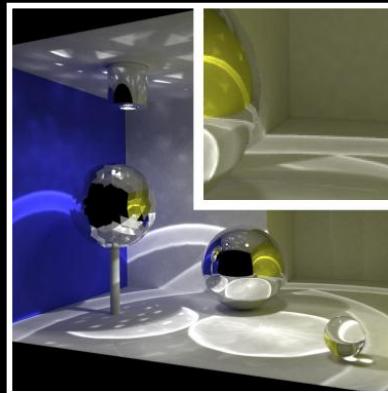
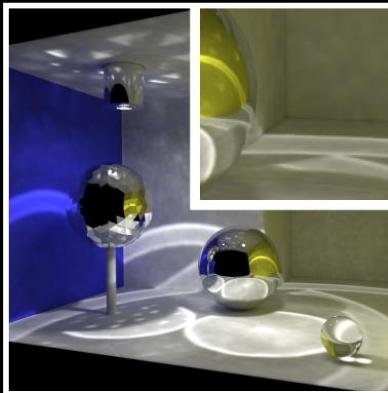
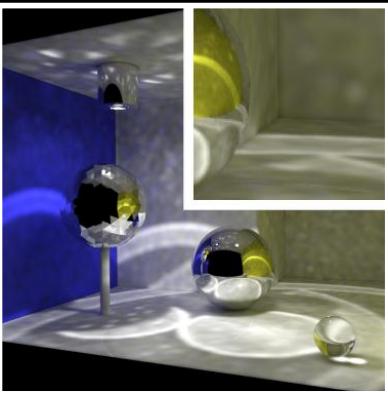
1—10



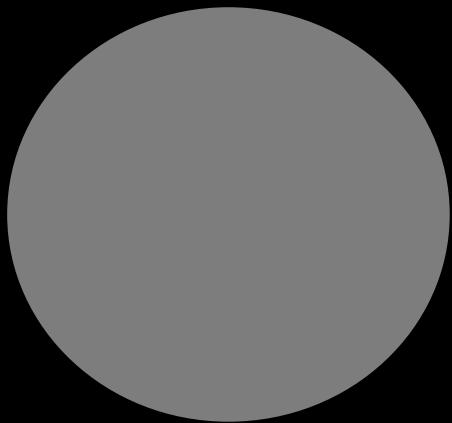
1—100



1—1000



Arbitrary Kernels



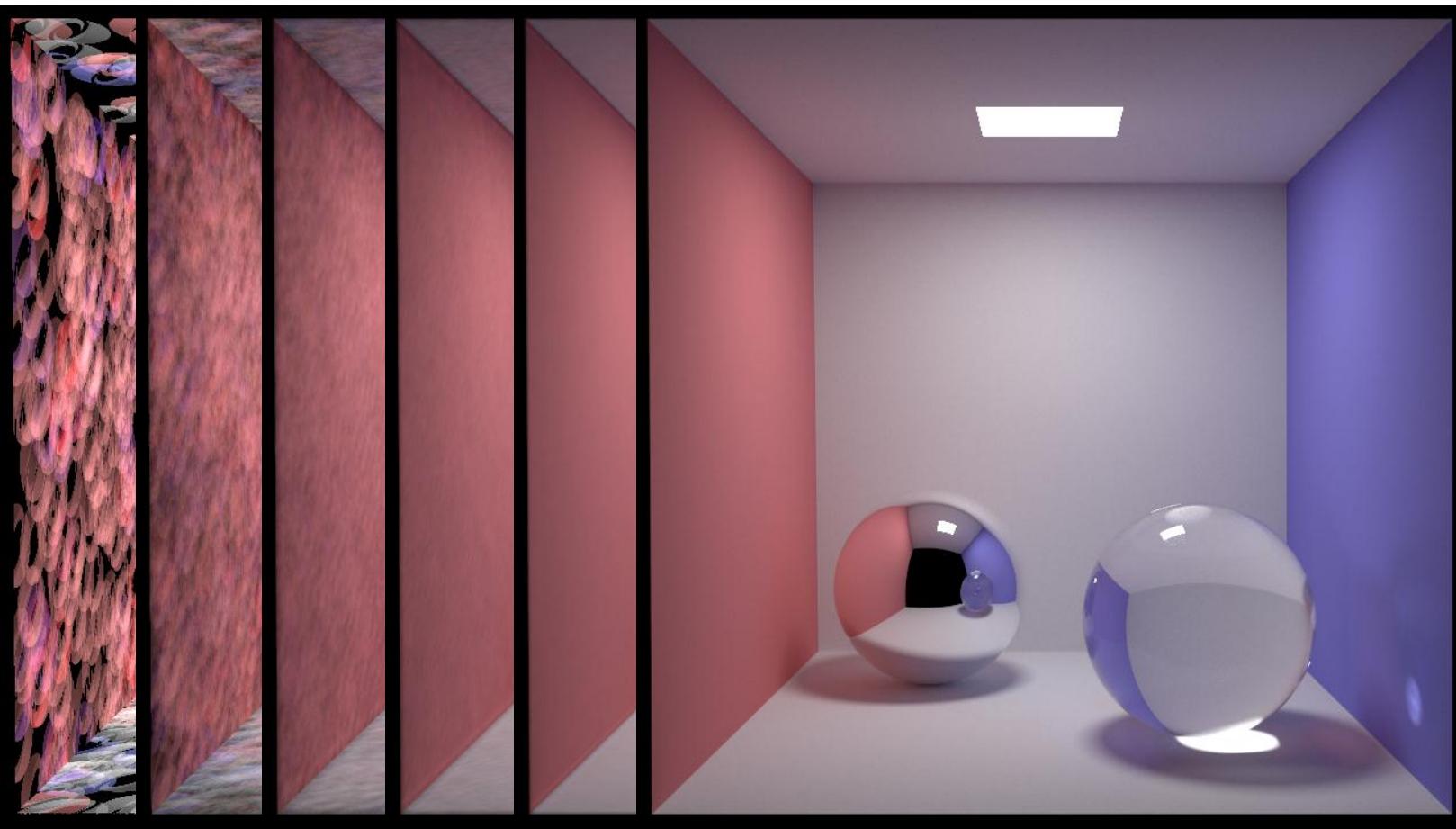
Box



Gaussian



SIGGRAPH

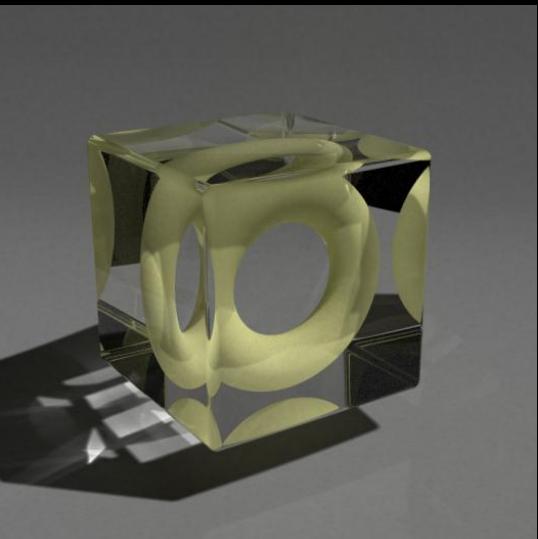


Stochastic Effects

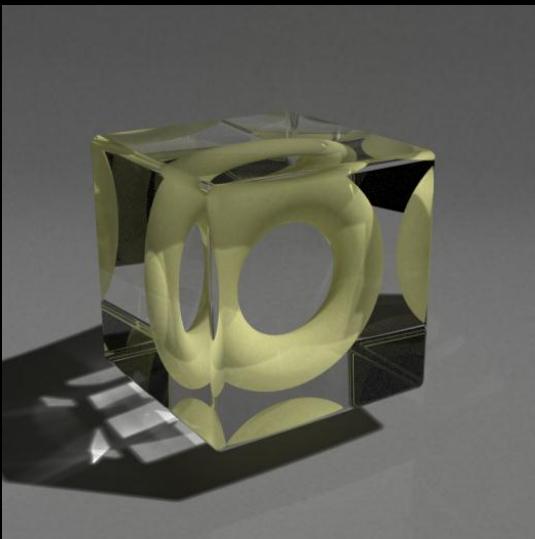


Scene courtesy of Toshiya Hachisuka

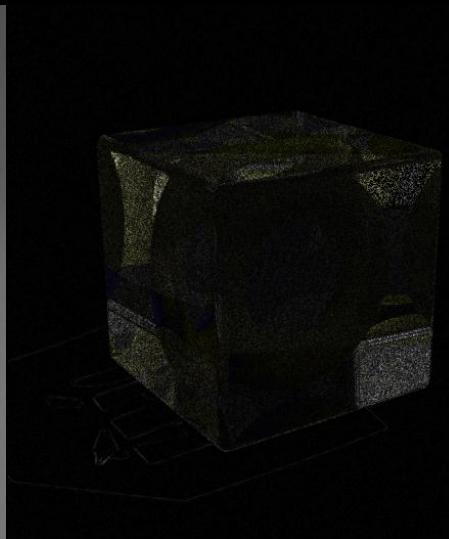
Stochastic PPM



Our method



20x Difference



Scene courtesy of Toshiya Hachisuka



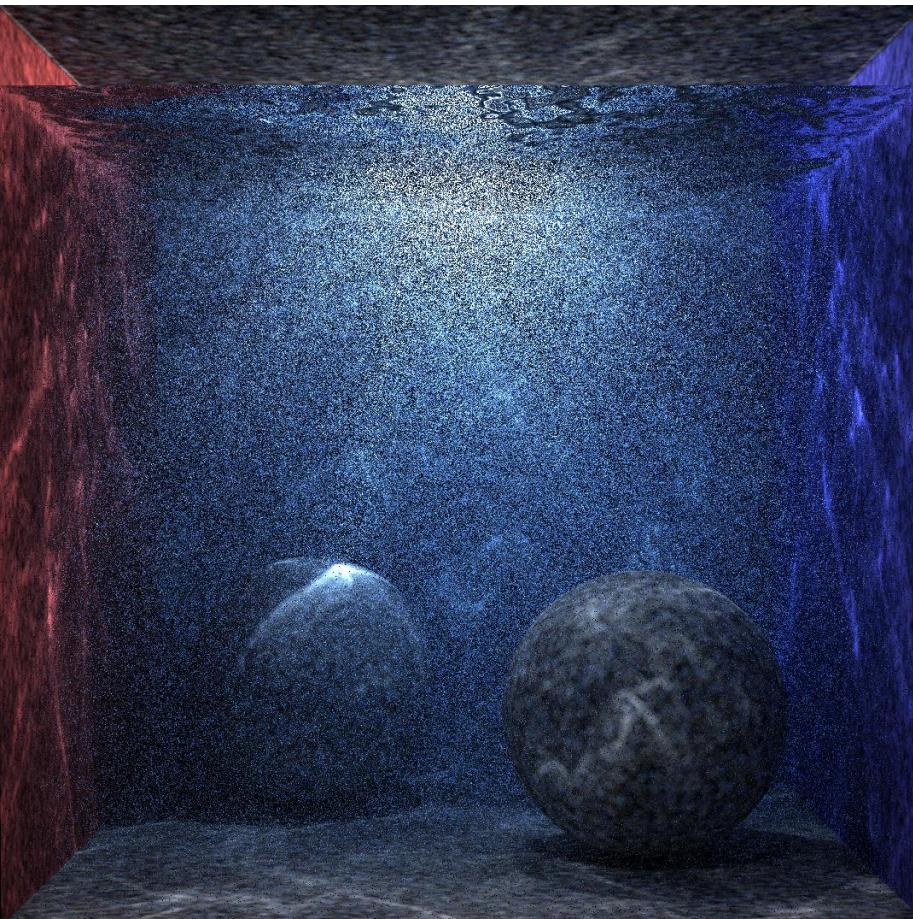
Participating media

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

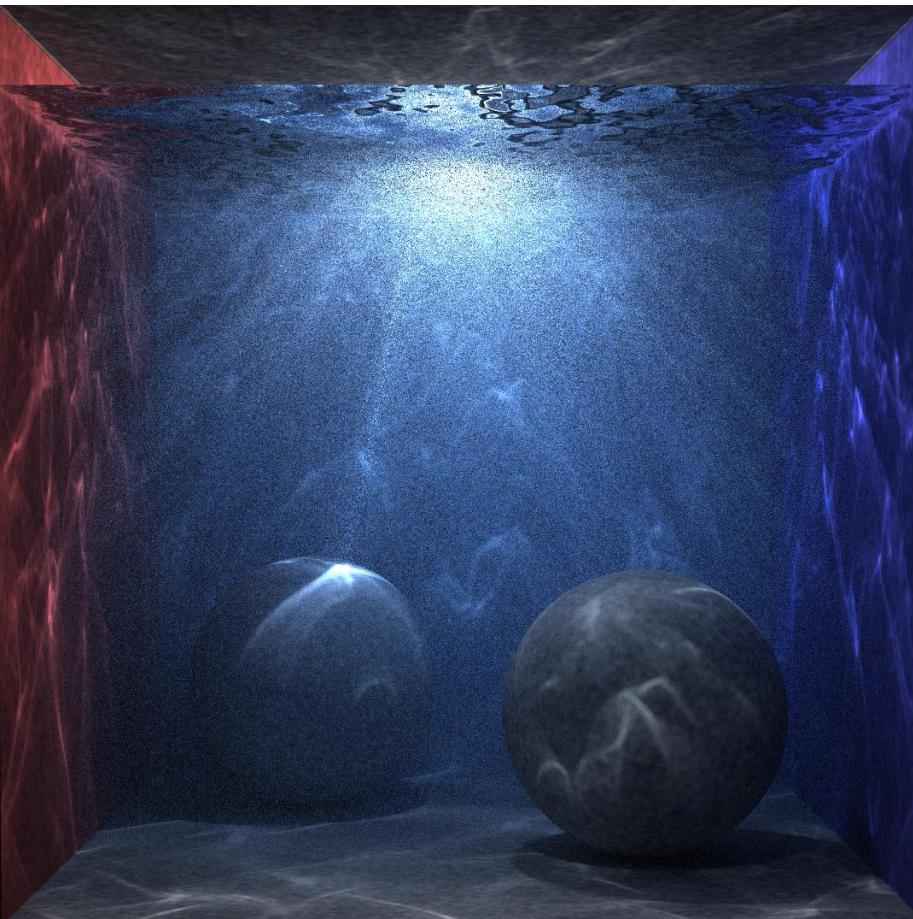


Participating media

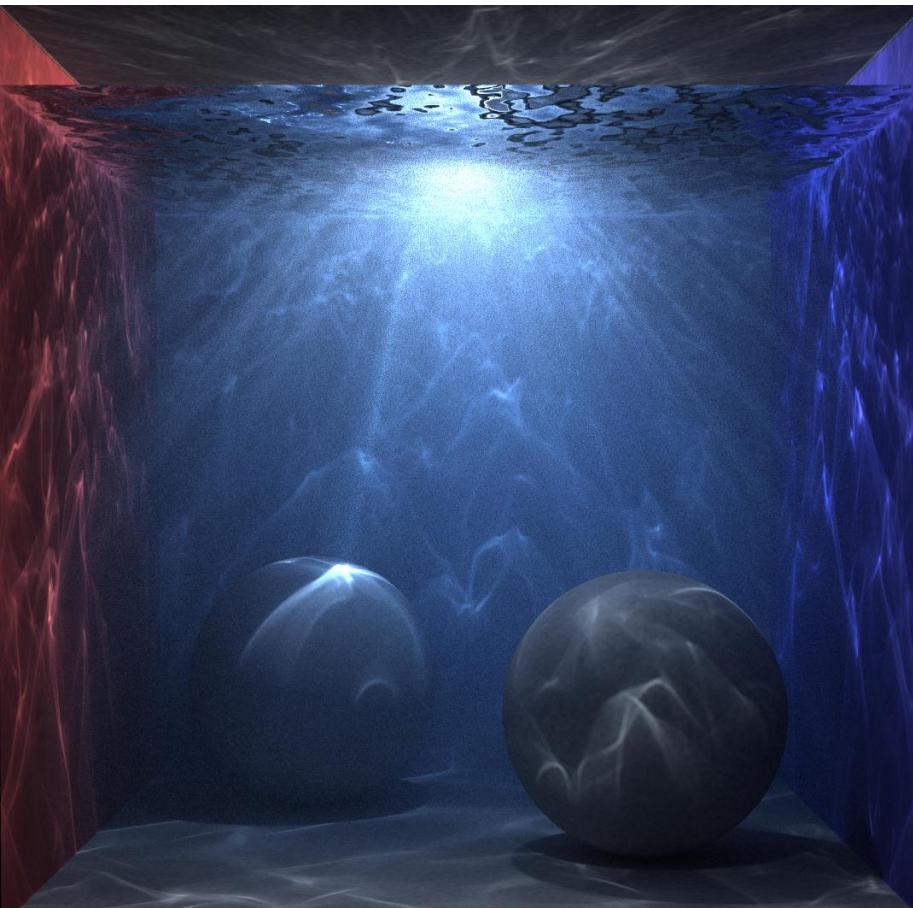
$$\frac{r_{i+1}^3}{r_i^3} = \frac{i + \alpha}{i + 1}$$



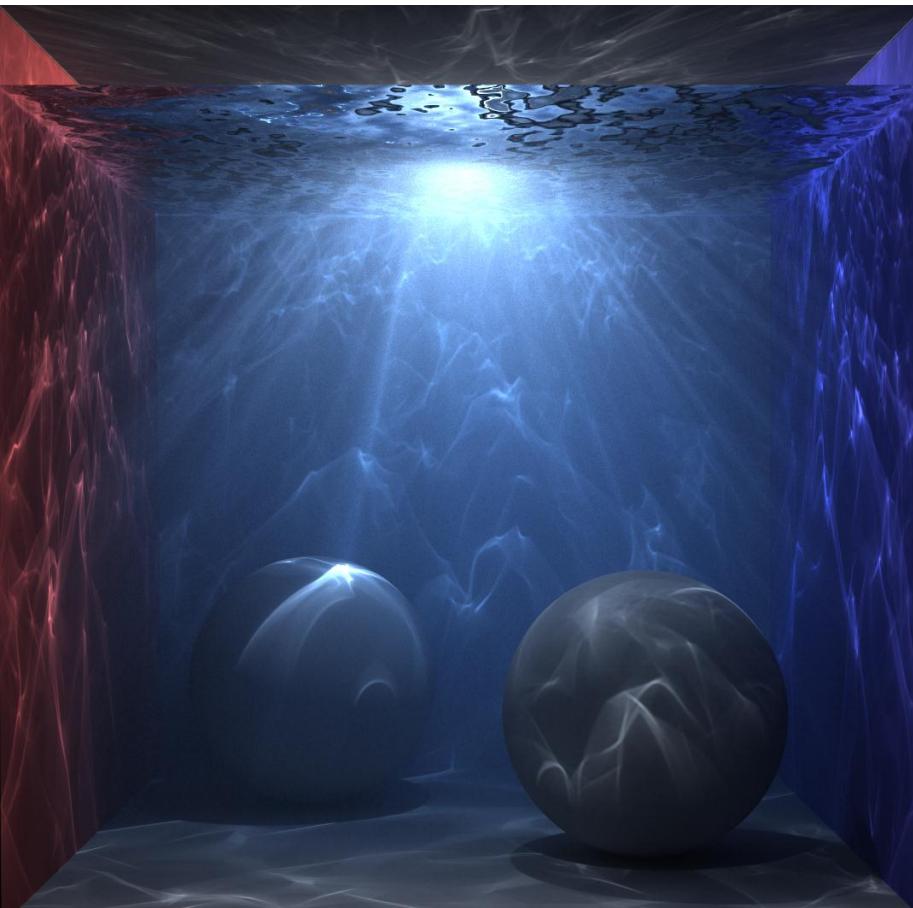
1 iteration
2 million photons



10 iterations
20 million photons



100 iterations
200 million photons

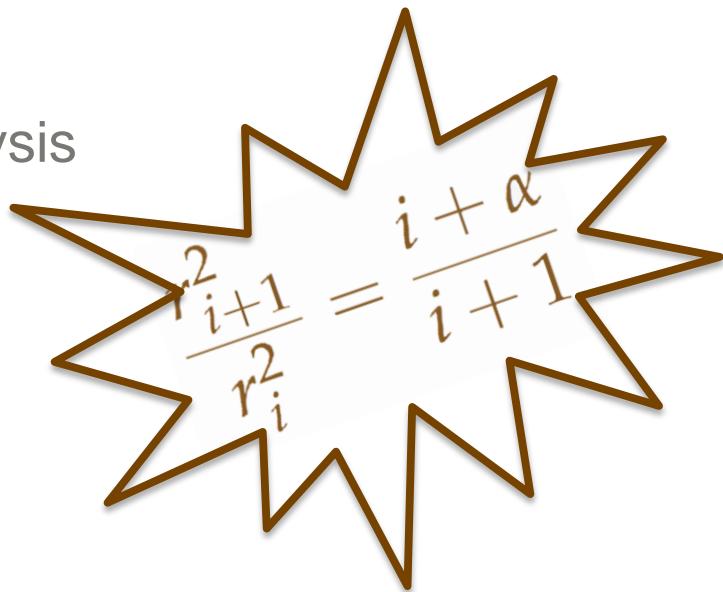


1000 iterations
2 billion photons



Conclusions

- Progressive photon mapping without statistics
- Simple radius update
- Asymptotic convergence analysis
- Trivial implementation
- Applicable to arbitrary kernels
- Generalizable to volumes, beams [Jarosz et al. 2011]


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

Adaptive Progressive Photon Mapping



Adaptive Progressive Photon Mapping

Anton S. Kaplanyan
Karlsruhe Institute of Technology, Germany

- Shrink radius (bandwidth) for N^{th} photon map

$$r_N^2 = r_0^2 N^{\alpha-1}, \quad \alpha \in (0; 1)$$

- User-defined parameters r_0 and shrinkage rate α

Problem:

- Optimal value r_0 of and α are unknown
- Usually globally constant / k-NN defined

2

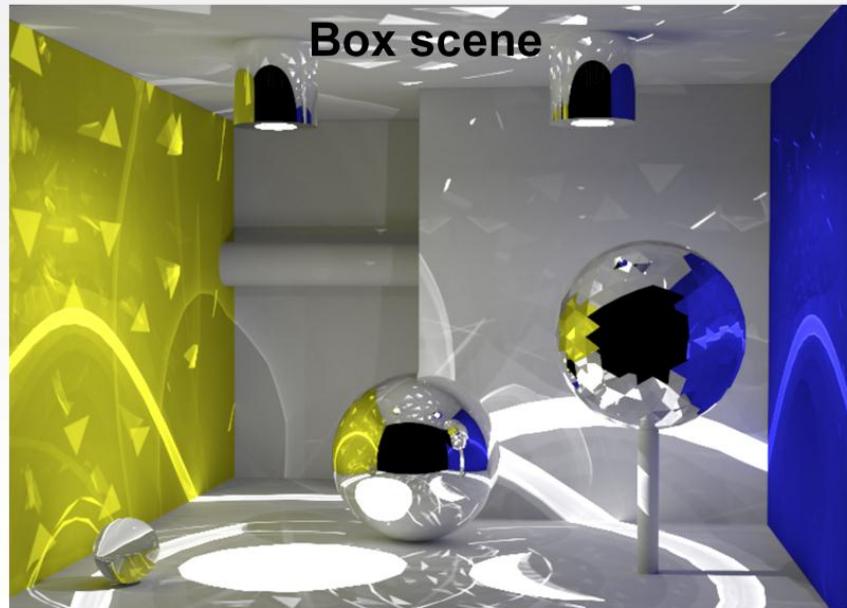
Radius, or kernel bandwidth, is the central parameter for efficiency of the kernel estimation in general.

Knaus and Zwicker shown that the radius must be decreased exponentially.

This equation has two extremely important parameters.

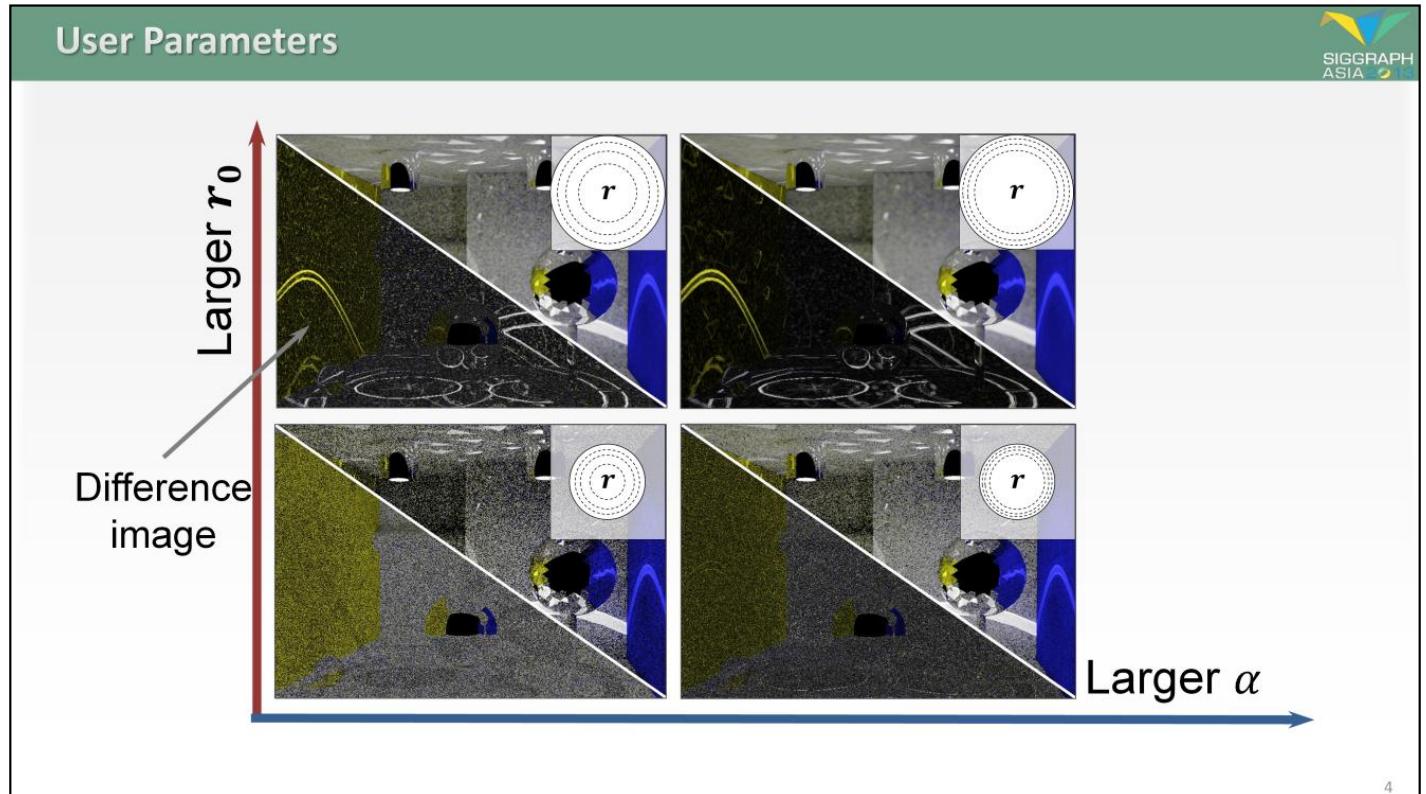
It is an initial radius; and a shrinkage rate alpha.

The alpha parameter is usually defined by user, while the initial radius can be selected using k-NN.



3

This is a Box scene with complicated illumination.
We show how these two parameters affect the rendering.



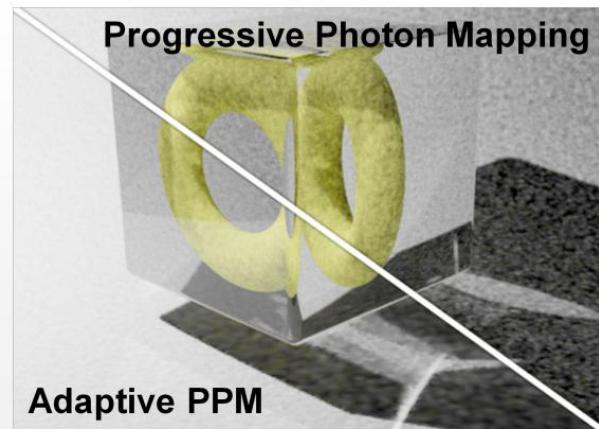
First, we initialize the algorithm with a large initial k NN-selected radius and a small alpha of 0.5.

This leads to a lot of bias, like smoothed caustics due to the large initial radius, yet the noise quickly becomes apparent due to a fast radius shrinkage.

The second case suffers from high image noise that appears quickly due to a small and quickly shrinking initial radius.

The third case demonstrates high bias that appears due to initially large and slowly shrinking radius.

And in the fourth, a too-small initial radius can lead to high noise, while bias becomes apparent due to slow shrinkage rate.



ADAPTIVE PROGRESSIVE PHOTON MAPPING

As we have seen, these parameters are extremely important for rendering.
We will show how we can choose them wisely.

OPTIMAL CONVERGENCE OF PROGRESSIVE PHOTON MAPPING

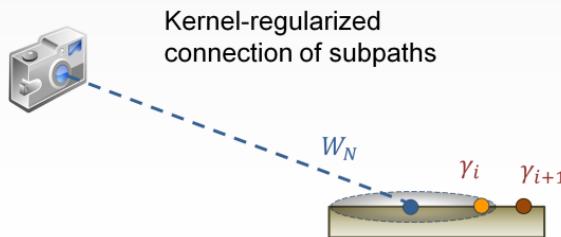
First, we will show how to choose alpha parameter for asymptotically optimal convergence.

- Pixel estimate with multiple eye and light subpaths

$$I \approx \sum_{N,i} W_N k(x_N - y_i) \gamma_i$$

Eye subpath importance Photon radiance

- Generate full path by joining subpaths



7

Progressive photon mapping can be interpreted as an estimator that constructs the full paths from camera to light by constantly connecting two fresh subpaths.

The estimation is a sum, where W_N is the importance of the eye subpath and γ_i is the radiance of the light subpath (photon).

So, the actual construction is a regularized (imprecise) junction of two end vertices.

- PPM = recursive (online) estimation [Yamato71]

$$I_N = \frac{N-1}{N} I_{N-1} + W_N \sum_i k(x - x_i) \gamma_i$$

- Rearrange the sum

$$I_N = \frac{N-1}{N} I_{N-1} + \sum_i k(x - x_i) [W_N \gamma_i]$$

Kernel estimation
Path contribution

8

First off, we write this estimation for every photon map in a recursive manner. This is the first step that allows us to apply the knowledge about recursive (aka online) estimation accumulated in the statistical research.

Next, we push the W_n term into the sum to reduce it to the canonical form of kernel estimation for the new function, which now represents the complete path throughput.

Using this form, we can apply many statistical results from recursive estimation field.

$$\text{MSE} = \text{Var}_{\text{Est}} + \text{Bias}_{\text{Kernel}}^2$$

As one of these statistical results, we know that the mean squared error of this recursive kernel estimator consists of variance and squared bias.

$$\text{MSE} = \text{Var}_{\text{Meas}} + \text{Var}_{\text{Kernel}} + \text{Bias}_{\text{Kernel}}^2$$

- ▶ Variance is two-fold: measurement and kernel

10

We expand the variance and split it into two subterms: the variance directly related to the kernel estimation (i.e. the noisy distribution of photons) and the variance of the measurement equation.

$$\text{MSE} \approx \text{Var}_{\text{Meas}} + \text{Bias}_{\text{Kernel}}^2$$

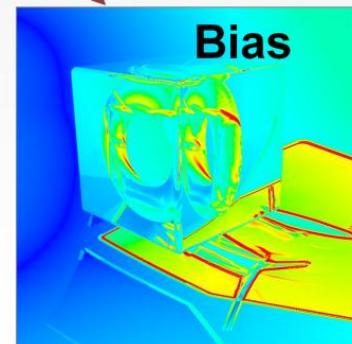
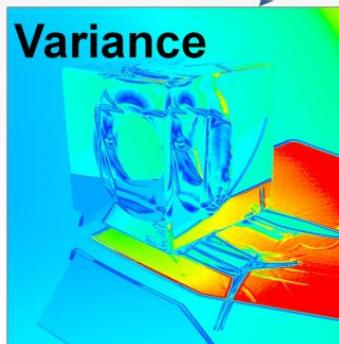
- ▶ Measurement variance is higher $\text{Var}_{\text{Meas}} \gg \text{Var}_{\text{Kernel}}$

11

The measurement variance is significantly higher (see the paper for details).

► So, MSE has **noise** (path variance) and **bias**

$$\text{MSE} \approx \text{Var}_{\text{Meas}} + \text{Bias}_{\text{Kernel}}^2$$

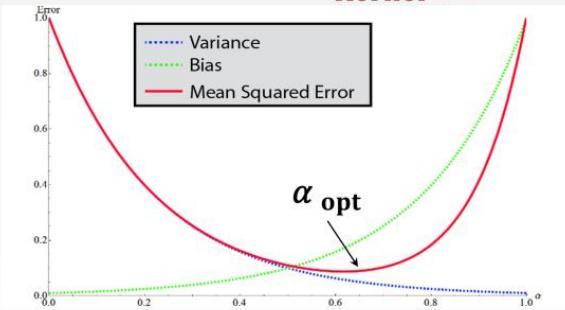


12

Thus the major contributors to the PM error are the variance stemming from the path sampling (image noise) and the bias introduced by kernel estimation (blurred illumination). This error directly depends on the two parameters we described.

- Both variance and bias depend on α [KnausZwicker11]

$$\text{Var}_{\text{Meas}}(\alpha) \sim N^{-\alpha} \quad \text{Bias}_{\text{Kernel}}^2(\alpha) \sim N^{\alpha-1}$$



► Optimal convergence is $\text{MSE} \propto N^{-2/3}$ with $\alpha_{\text{opt}} = 2/3$

► Asymptotic convergence

► Unbiased Monte Carlo methods are faster: $\text{MSE} \propto N^{-1}$

13

First, we consider the dependence on the shrinkage rate parameter alpha.

As pointed out by Knaus and Zwicker, both bias and variance depend on this parameter.

If we put them together into MSE (with some additional analysis described in the paper), we can deduce the asymptotic behavior of MSE with respect to alpha.

As we can clearly see from the MSE plot, it has a minimum.

The optimal asymptotic convergence rate is thus achieved with alpha = 2/3.

Interestingly, photon mapping methods are asymptotically slower than unbiased methods.

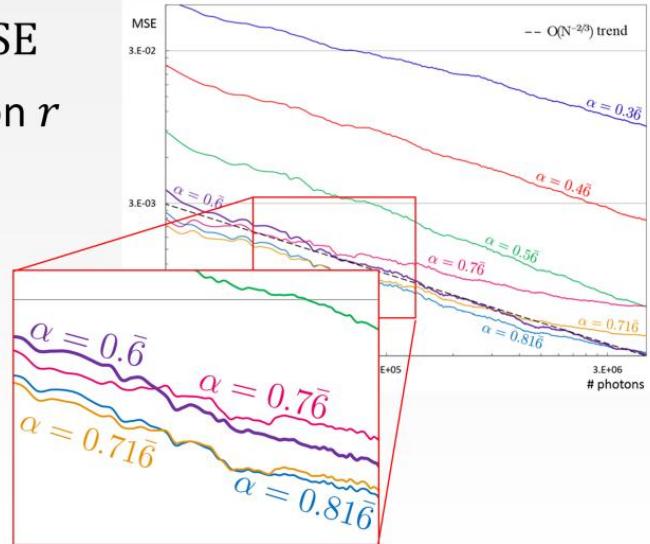
This was the asymptotic analysis. In practice, we have to worry about behavior of MSE on a finite number of photons.

ADAPTIVE BANDWIDTH SELECTION

We will show how to select the bandwidth (radius) parameter r to achieve faster convergence in finite time.

Adaptive Bandwidth Selection

- ▶ α_{opt} might not yield minimal MSE
- ▶ Variance and bias also depend on r
 - ▶ $\text{Var}_{\text{Meas}}(r) \sim r^{-2}$
 - ▶ $\text{Bias}_{\text{Kernel}}(r) \sim \Delta I r^2$
 - ▶ $\Delta I = \Delta(W_N \gamma_i)$ is a pixel Laplacian
- ▶ Minimize MSE with respect to r
 - ▶ Achieve variance \leftrightarrow bias tradeoff
 - ▶ Select optimal r using past samples
- ▶ Pixel Laplacian ΔI is unknown



15

Minimizing MSE w.r.t alpha is not enough for practical rendering.

As we can see from this plot (vertically – MSE, horizontally – number of photons), even the optimal alpha 2/3 (magenta) might lead to high MSE at early stages of rendering, even though providing the steepest slope.

To minimize the MSE on a finite set of samples, we optimize it w.r.t. the second free parameter, kernel radius (bandwidth) r .

In recursive estimation, this is also a well-known approach called bandwidth selection.

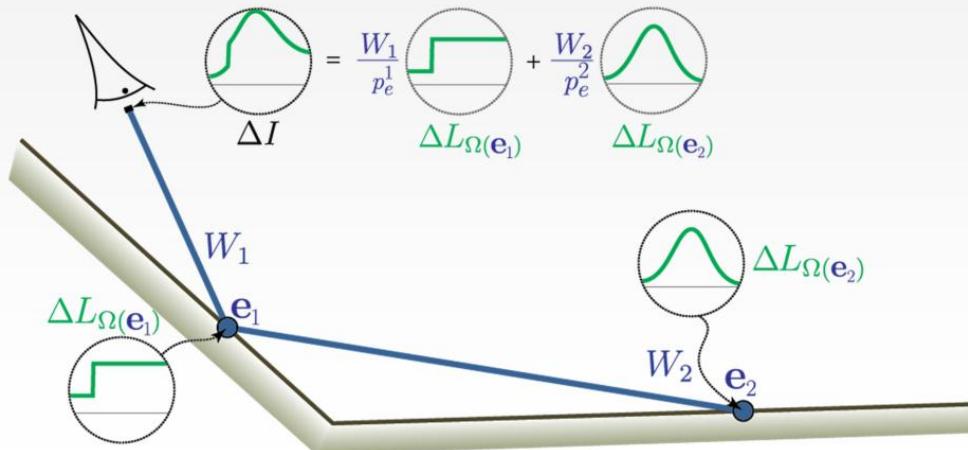
We showed that both variance and bias depend on kernel bandwidth.

It is possible to select the optimal bandwidth in a way to achieve the trade-off between variance and bias, using the past photon maps.

Moreover, bias also depends on the unknown pixel Laplacian ΔI , which is intuitively the curvature of the estimated radiance around the every estimation point.

► $\Delta I = \Delta(W_N \gamma_i)$ consists of Laplacians at all shading points

► Weighted per-vertex Laplacians $\Delta \gamma_i = \Delta L$ with importance W_N



16

So, how can we estimate the last unknown value: pixel Laplacian?

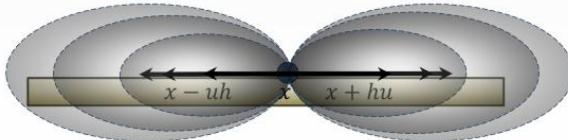
Laplacian can be seen as a curvature measure of a function.

Pixel Laplacian ΔI intuitively defines a radiance curvature around pixel support.

We can estimate it by summing up the per-vertex Laplacians ΔL at each shading point, weighted by the importance of this shading point.

- ▶ Estimate per-vertex Laplacian at each shading point
- ▶ Recursive finite differences in tangent space [Ngen11]
 - ▶ Yet another recursive estimator
 - ▶ Another shrinking bandwidth h
 - ▶ Robust estimation on discontinuities

$$\Delta L_u = \frac{L_{x+uh} + L_{x-uh} - 2L_x}{h^2}$$



17

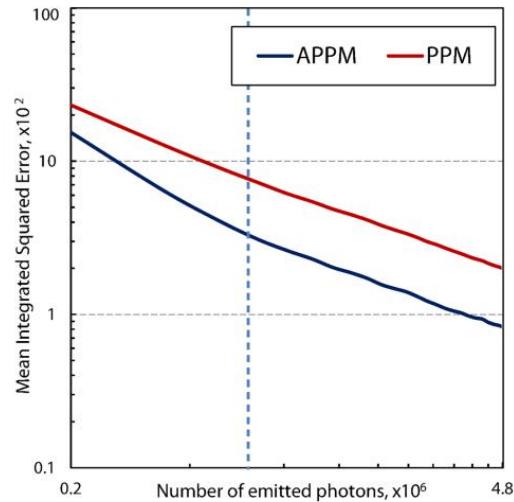
In order to estimate each per-vertex Laplacian, we use the recent recursive finite differencing along tangent axes of the surface.

This is a standalone recursive estimator with independent shrinking bandwidth (please see the paper).

This recursive finite differencing is also robust to discontinuities, which is important in context of light transport.

► Fast convergence on finite samples

- ▶ Data-driven bandwidth selector
- ▶ Lower starting error
- ▶ Keeps noise-bias balance



18

Now we have all the unknowns to select the optimal bandwidth on the fly.

This gives us faster convergence in practical scenarios of finite time.

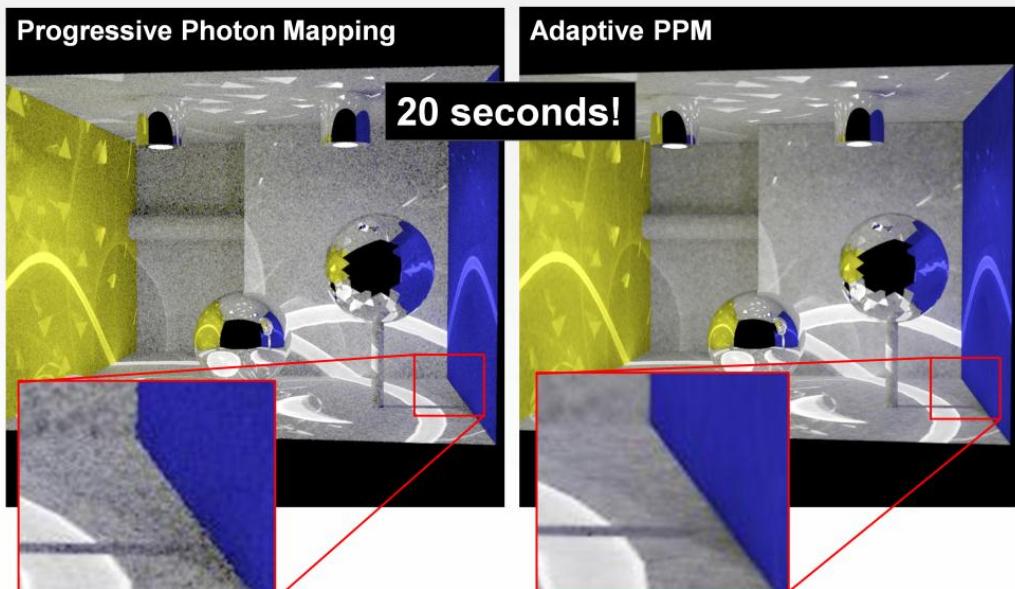
The selector uses past samples to estimate several auxiliary values, so it learns from the data.

(Image) After a few photon maps, MSE becomes already smaller than with the best manually-chosen parameters.

However, at some point, the lines become parallel, which means that the asymptotic convergence rate stays the same.

The proposed selector always tries to keep the balance between image noise and bias.

Results

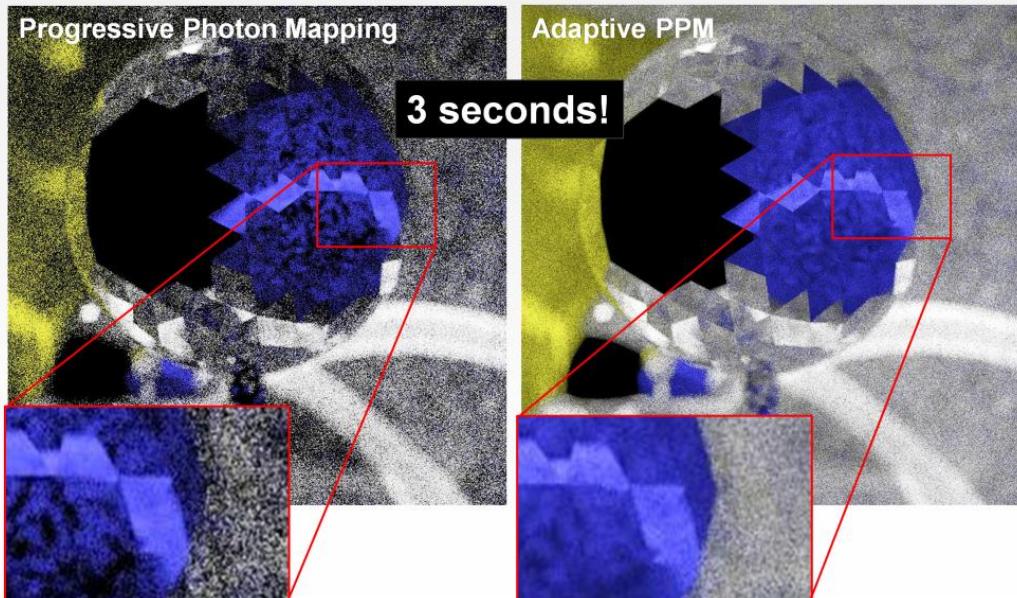


19

Here are some results.

Equal time comparison. Less variance, yet edges are sharp.

Results



20

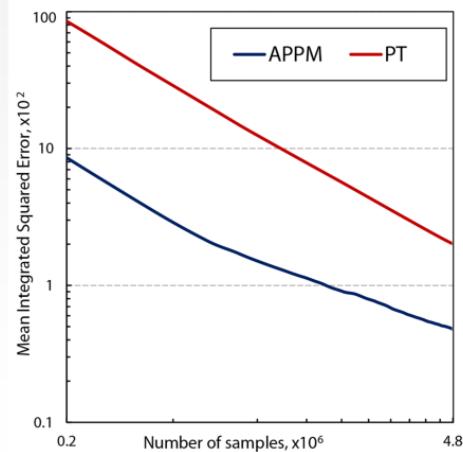
Equal time comparison. Heavy out of focus region. Less variance, caustics are sharp.

► Optimal asymptotic convergence rate

- ▶ Take-home message: Slower than unbiased methods

► Adaptive bandwidth selection

- ▶ Based on previous samples
- ▶ Balances estimation error
- ▶ Speeds up convergence
- ▶ Attractive for interactive previews



21

The benefits of the presented results are two-fold:

Firstly, we have deduced the **asymptotic** convergence rate for photon mapping class methods.

The take-home message here is that the convergence rate is slower comparing to unbiased methods whenever one applies a regularization of some kind.

The second contribution is the adaptive bandwidth (radius) selection based on a high-dimensional estimation of radiance curvature along the path.

The method uses the same photon map to estimate this curvature, thus an additional sampling is not needed.

This selection technique allows to significantly cut the amount of bias and variance in the image starting from the first photon map.

This property makes it attractive for fast interactive preview renderers, but also reduces the bias in final images.

Participating Media

Participating Media Basics



Fog



2

Thursday, August 23, 12

- In this portion of the class we are interested in understanding how to render scattering media.
- so far we have assumed that photons can travel unobstructed between objects.
- In reality, the “solid objects” in our world are embedded within a medium.
- In fog for instance, the interaction of light with the tiny water particles in the air can create stunning effects such as the volumetric shadow beams emanating from the trees in this image.



Clouds & Crepuscular rays



3

Thursday, August 23, 12

- clouds are another example of this, and since we see them much further away, we can very easily see their heterogeneous nature



Surface or Volume?



4

Thursday, August 23, 12

- In fact, the concept of “solid objects” is really a simplification, and the boundary between media and surfaces is actually not very easy to define.
- for instance, this iceberg may appear as a solid object, but really the light penetrates through it and you could also think of it as a medium



Surface or Volume?



5

Thursday, August 23, 12

- In reality it's more accurate to think of solid objects as just the boundary between different media, in this case air and jade.
- And by generalizing photon mapping to account for this, we can get effects such as subsurface scattering of light past the boundary



Antelope Canyon, Az.



6

Thursday, August 23, 12

- It's also important to point out that we cannot consider surface illumination and media illumination in isolation.
- In this photograph almost all illumination on the walls is indirect light that has either bounced off of the small illuminated patches on the ground, or off of tiny dust particles in the air.
- There is a coupling between these two, which is quite easy to account for with photon mapping.

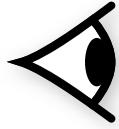


Outline

- Theoretical background
- Extending photon mapping to media



Participating Media



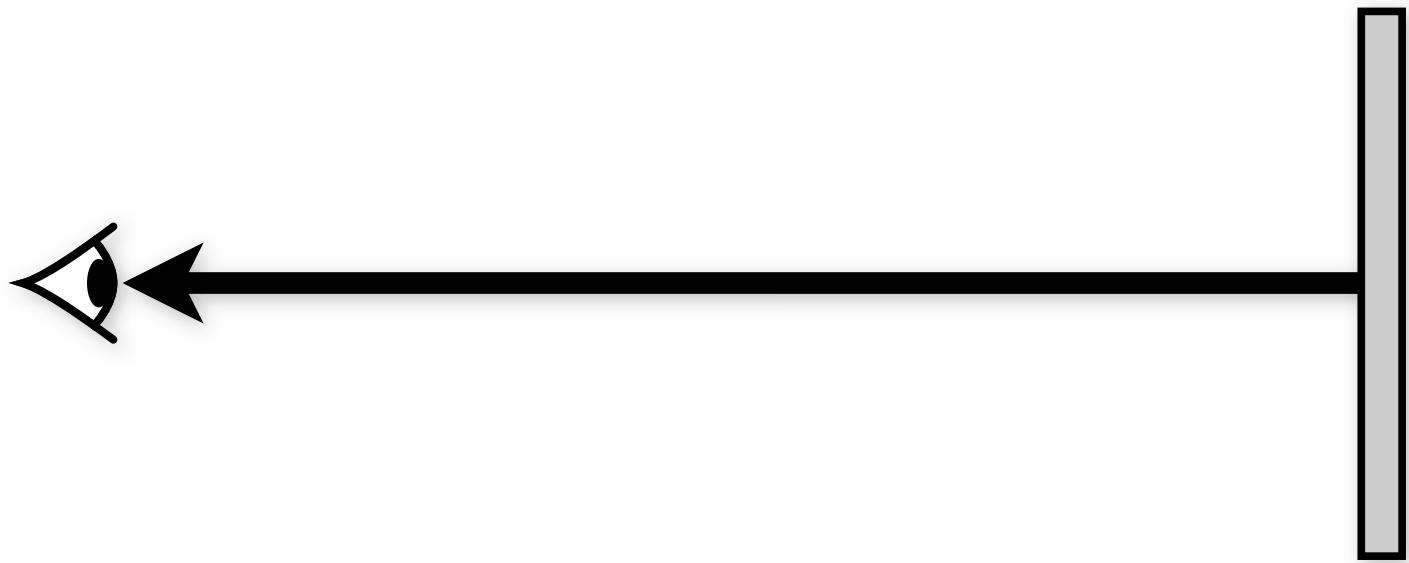
8

Thursday, August 23, 12

- Part. media can be thought of as a collection of particles suspended in a vacuum.
- As a photon travel through the scene, it may interact with the medium by hitting one of these particles.
- We will not model each of these particles individually, since this would be impractical, but instead assume the particles are small and statistically model their aggregate properties.
- If we consider an infinitesimal segment of media, when a photon travels through this segment, a number of possible interactions might occur.



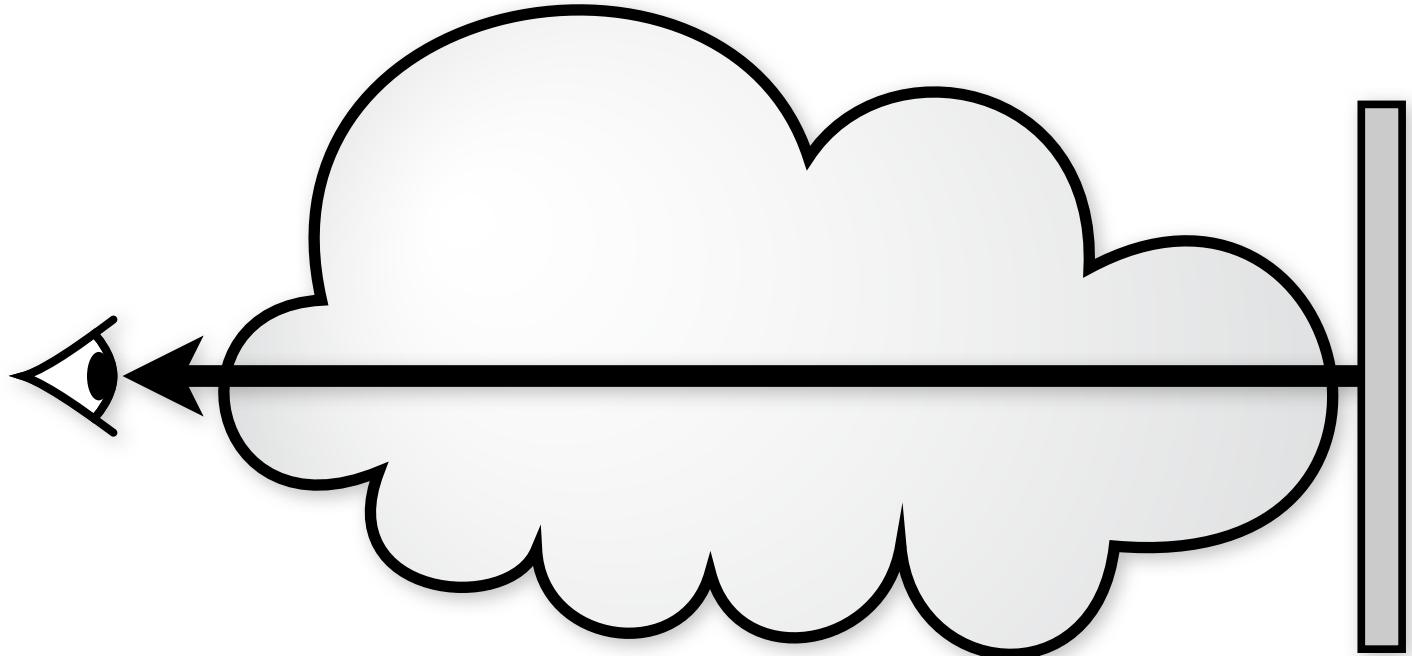
Participating Media



- Part. media can be thought of as a collection of particles suspended in a vacuum.
- As a photon travel through the scene, it may interact with the medium by hitting one of these particles.
- We will not model each of these particles individually, since this would be impractical, but instead assume the particles are small and statistically model their aggregate properties.
- If we consider an infinitesimal segment of media, when a photon travels through this segment, a number of possible interactions might occur.



Participating Media

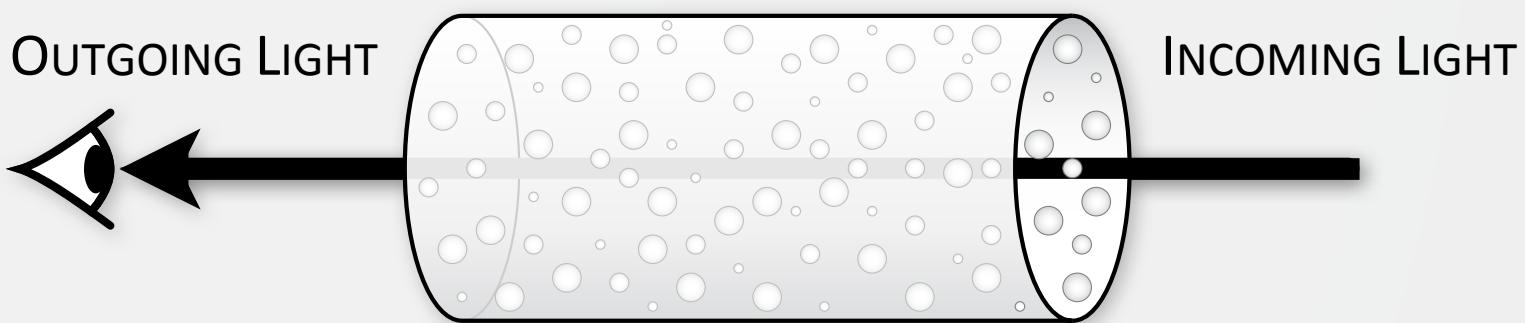


- Part. media can be thought of as a collection of particles suspended in a vacuum.
- As a photon travel through the scene, it may interact with the medium by hitting one of these particles.
- We will not model each of these particles individually, since this would be impractical, but instead assume the particles are small and statistically model their aggregate properties.
- If we consider an infinitesimal segment of media, when a photon travels through this segment, a number of possible interactions might occur.



Participating Media

MEDIUM INTERACTION



9

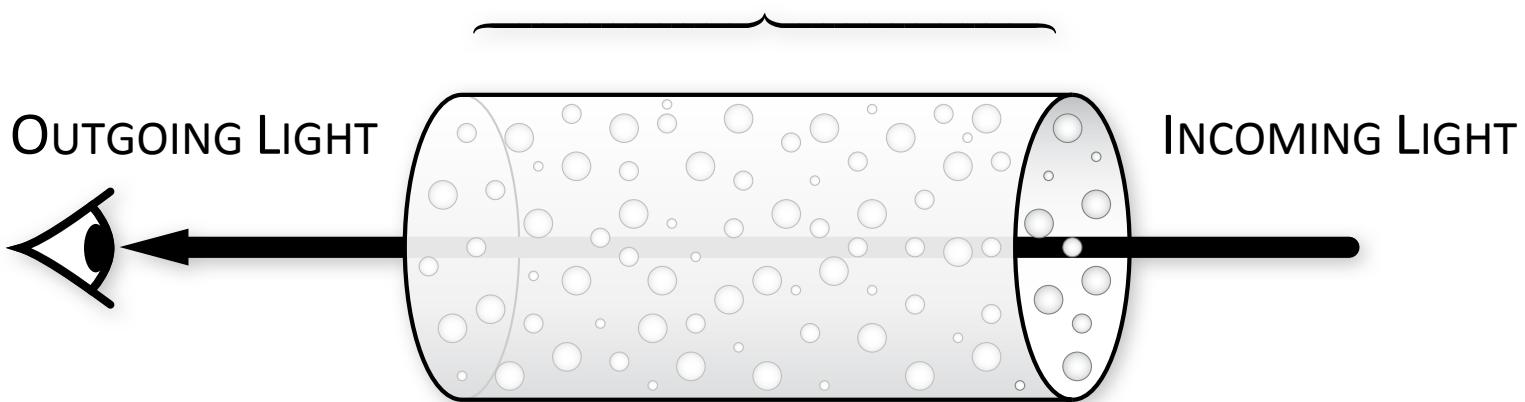
Thursday, August 23, 12

- Part. media can be thought of as a collection of particles suspended in a vacuum.
- As a photon travel through the scene, it may interact with the medium by hitting one of these particles.
- We will not model each of these particles individually, since this would be impractical, but instead assume the particles are small and statistically model their aggregate properties.
- If we consider an infinitesimal segment of media, when a photon travels through this segment, a number of possible interactions might occur.



Participating Media

MEDIUM INTERACTION



9

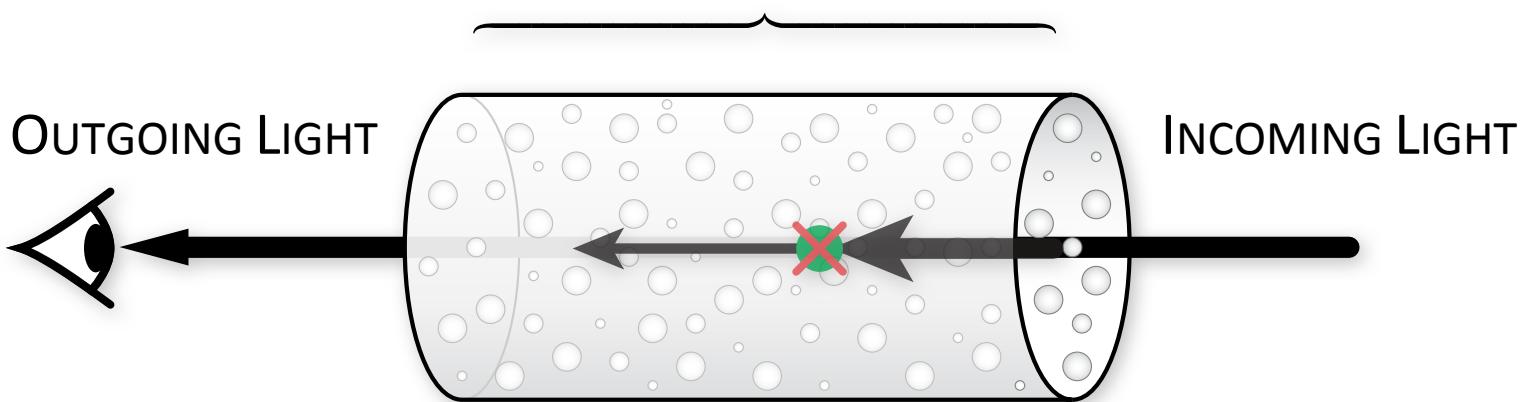
Thursday, August 23, 12

- Part. media can be thought of as a collection of particles suspended in a vacuum.
- As a photon travel through the scene, it may interact with the medium by hitting one of these particles.
- We will not model each of these particles individually, since this would be impractical, but instead assume the particles are small and statistically model their aggregate properties.
- If we consider an infinitesimal segment of media, when a photon travels through this segment, a number of possible interactions might occur.



Participating Media

ABSORPTION



$\sigma_a(x)$: absorption coefficient [1/m]

10

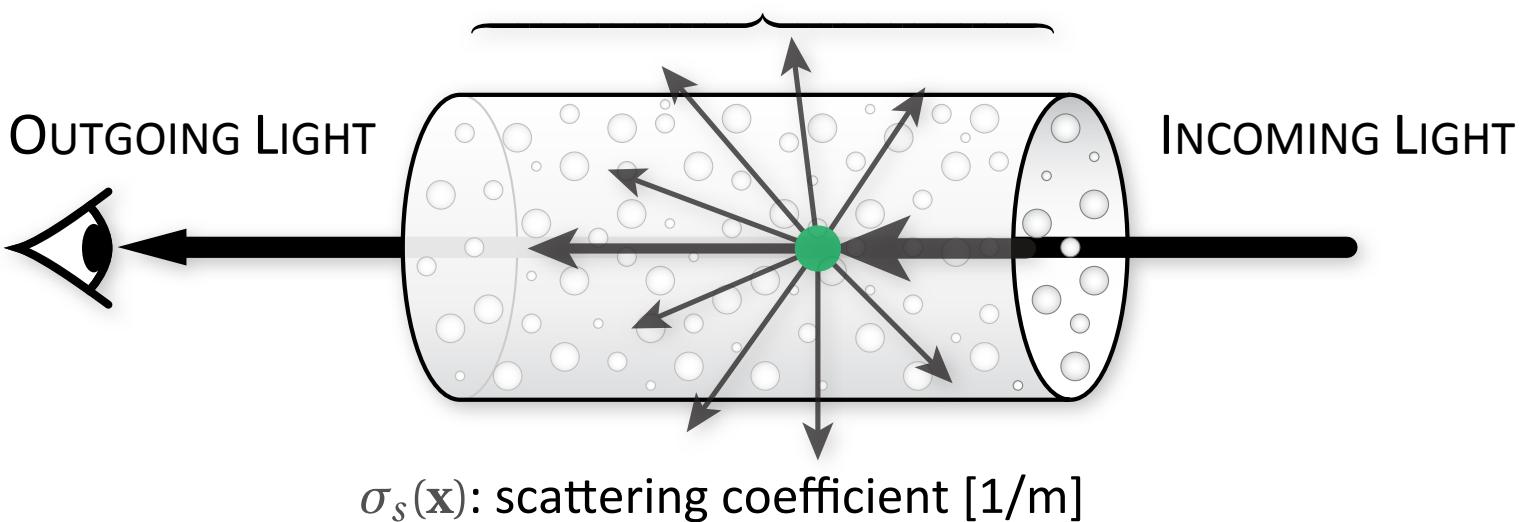
Thursday, August 23, 12

- One possibility is an absorption event.
- This means that some portion of the photons that enter the infinitesimal segment become absorbed by the particles, which means that less photons leave the segment.
- The amount of radiance loss depends on the absorption coefficient of the medium. This can be any non-negative value. If this coefficient is 0, then no absorption happens, and the radiance remains unchanged. The larger the value, the denser and darker the medium becomes.
- Media such as black smoke can be well approximated using only absorption events.



Participating Media

OUT-SCATTERING

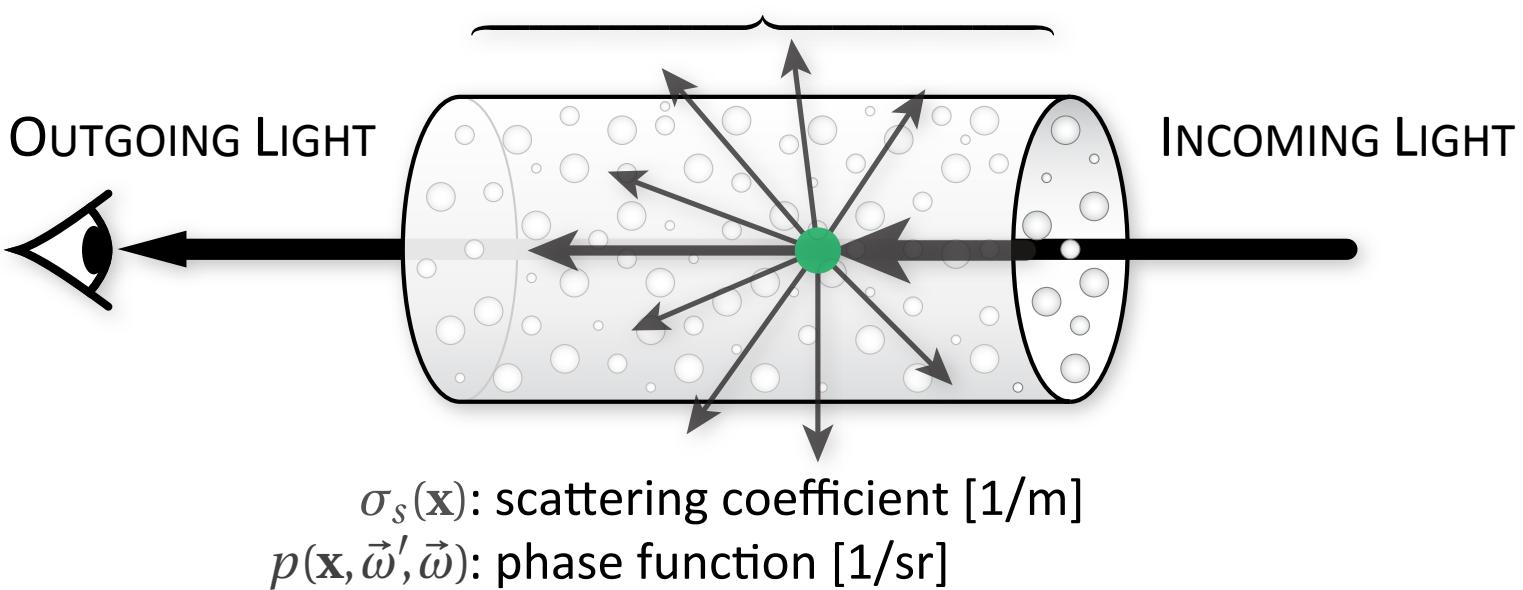


- Another thing that can happen is that the photons entering the medium segment are scattered into some other directions after hitting one of the particles.
- Out-scattering produces a net loss in radiance along the original direction.
- The amount of loss is determined by the scattering coefficient of the medium. As with the absorption coefficient it can take on any non-negative value, and if it is 0, no scattering happens.
- The directional distribution of scattering is described by the phase function, which is basically the analog of the BRDF, but for media



Participating Media

OUT-SCATTERING



11

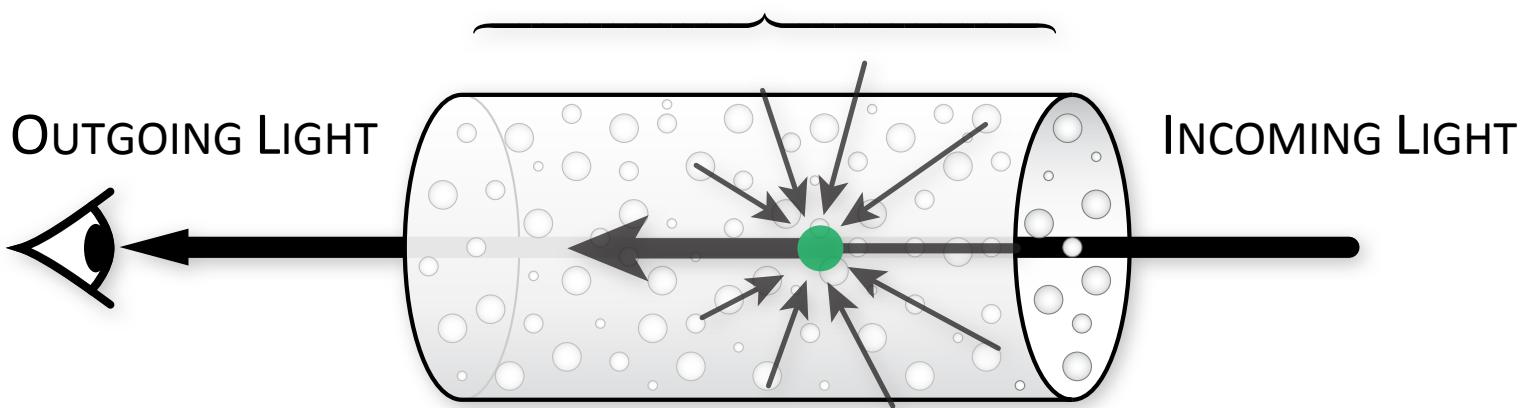
Thursday, August 23, 12

- Another thing that can happen is that the photons entering the medium segment are scattered into some other directions after hitting one of the particles.
- Out-scattering produces a net loss in radiance along the original direction.
- The amount of loss is determined by the scattering coefficient of the medium. As with the absorption coefficient it can take on any non-negative value, and if it is 0, no scattering happens.
- The directional distribution of scattering is described by the phase function, which is basically the analog of the BRDF, but for media



Participating Media

IN-SCATTERING



$\sigma_s(x)$: scattering coefficient [1/m]

$p(x, \vec{\omega}', \vec{\omega})$: phase function [1/sr]

12

Thursday, August 23, 12

- It is also possible for a photon originally traveling in a different direction to get scattered into the direction we are considering.
- This is called in-scattering and produces a net increase in radiance along the ray.
- Out-scattering and in-scattering are really just two sides of the same coin. A photon that out-scatters, reduces radiance in the original direction, but becomes the in-scattered photon in the new direction.
- We will see that this in-scattering is really the most complex and interesting lighting interaction to account for, and photon mapping will make this efficient



Photon Mapping

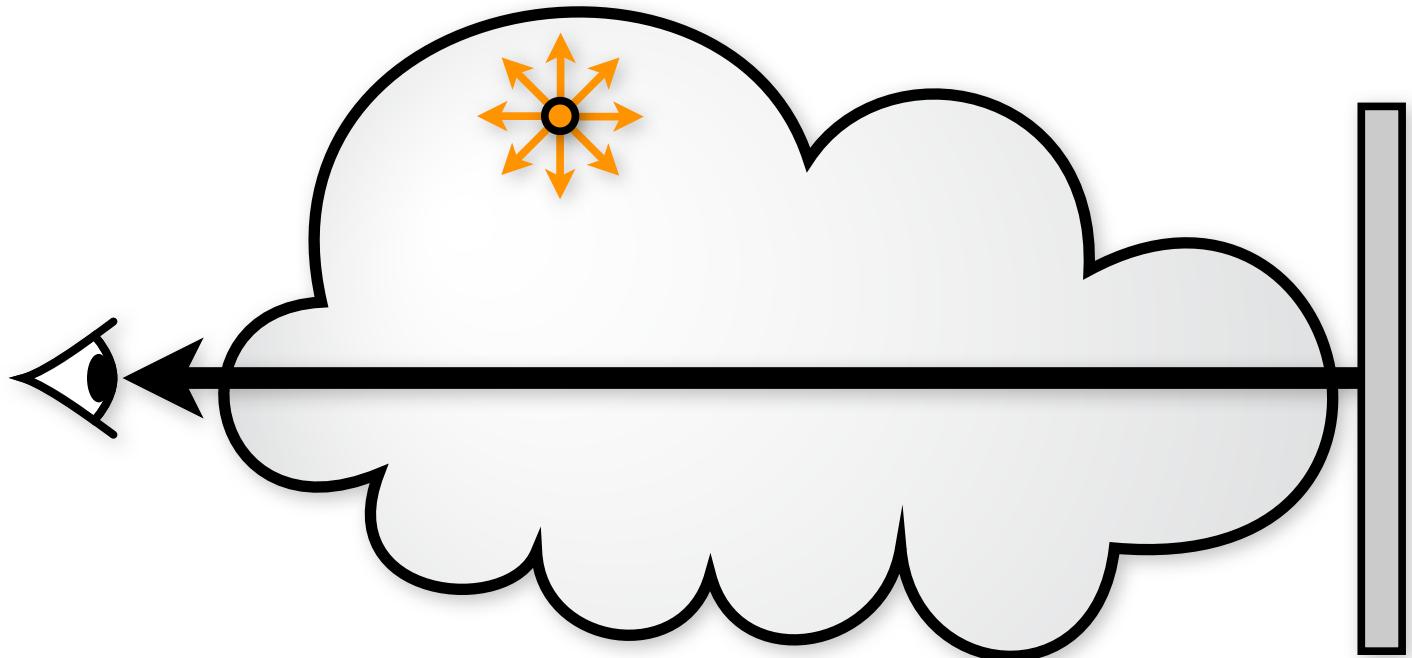
1) Photon tracing

2) Rendering / Radiance Estimation

- To account for media in the photon mapping algorithm, we will need to modify both the photon tracing stage, and also the rendering or radiance estimation stage



Photon Tracing in Participating Media



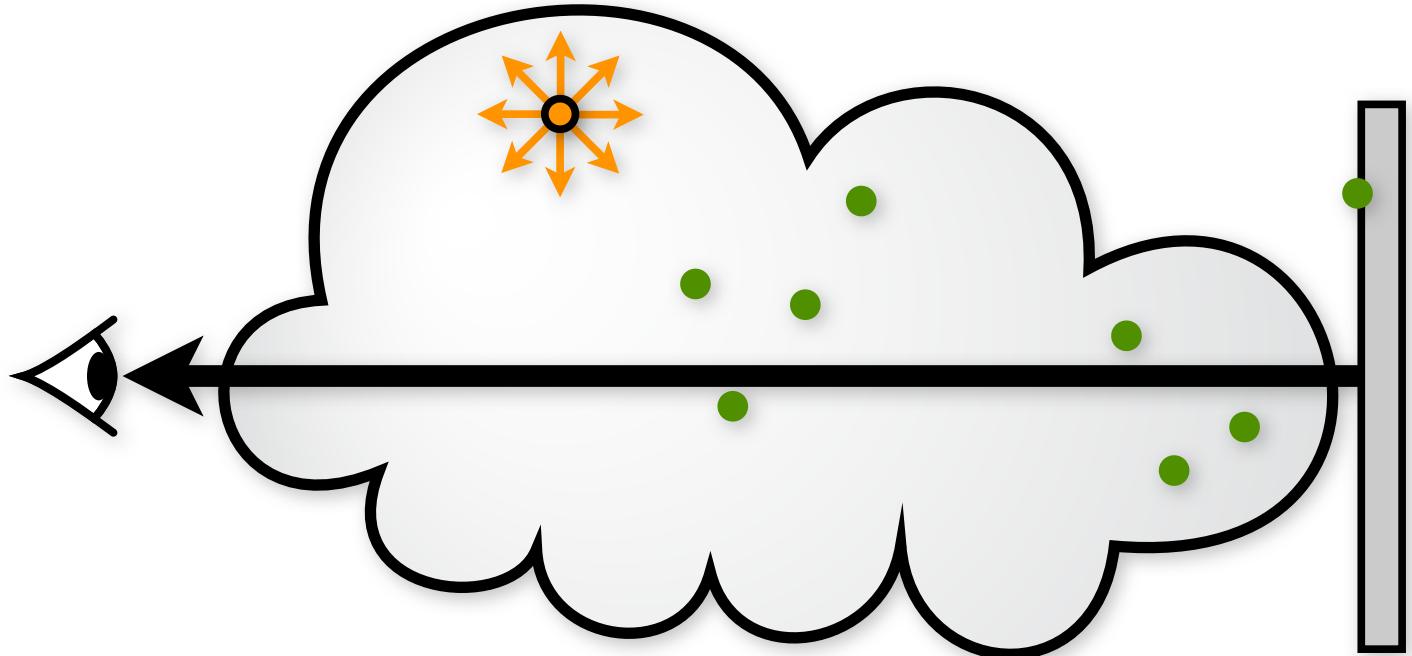
14

Thursday, August 23, 12

- For photon tracing, instead of emitted photons traveling directly to the surfaces, these photons now have some probability of scattering or being absorbed by the medium along their path.
- So we now trace paths that may have vertices within the medium, and at some point these paths will be probabilistically terminated when absorption happens.



Photon Tracing in Participating Media



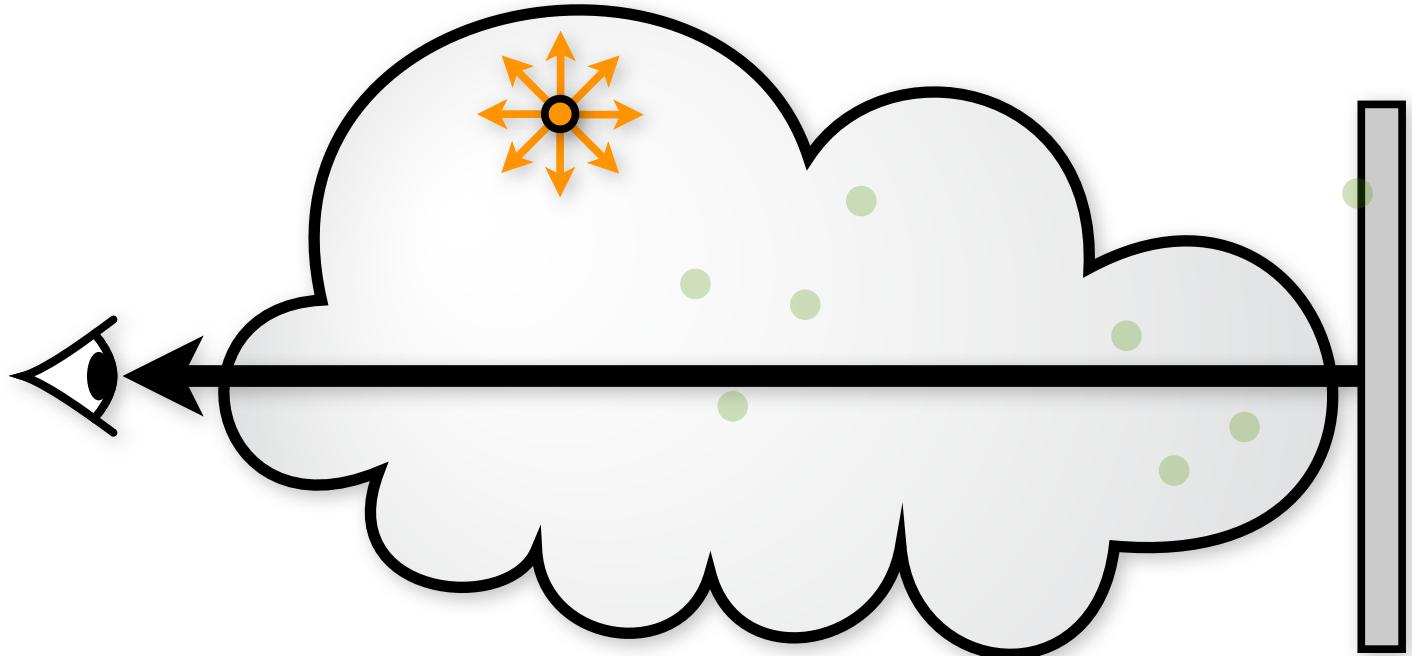
14

Thursday, August 23, 12

- For photon tracing, instead of emitted photons traveling directly to the surfaces, these photons now have some probability of scattering or being absorbed by the medium along their path.
- So we now trace paths that may have vertices within the medium, and at some point these paths will be probabilistically terminated when absorption happens.



Photon Tracing in Participating Media



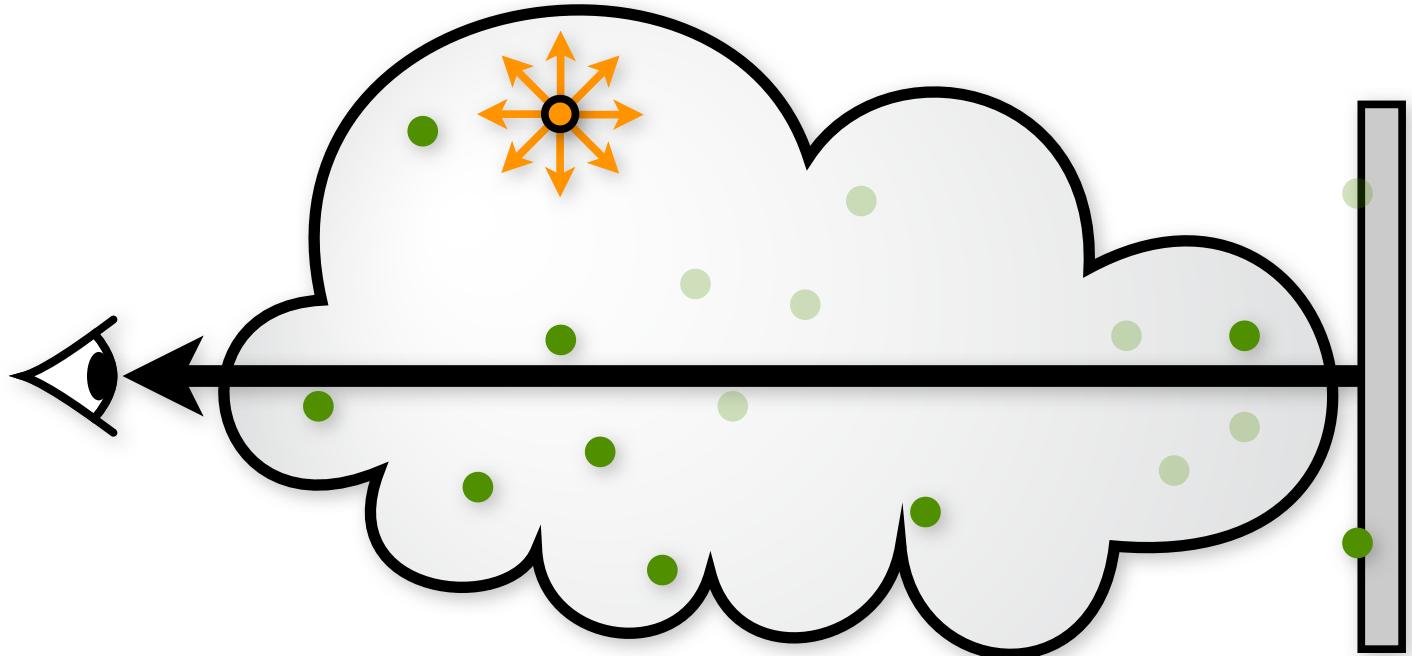
15

Thursday, August 23, 12

- For photon tracing, instead of emitted photons traveling directly to the surfaces, these photons now have some probability of scattering or being absorbed by the medium along their path.
- So we now trace paths that may have vertices within the medium, and at some point these paths will be probabilistically terminated when absorption happens.



Photon Tracing in Participating Media



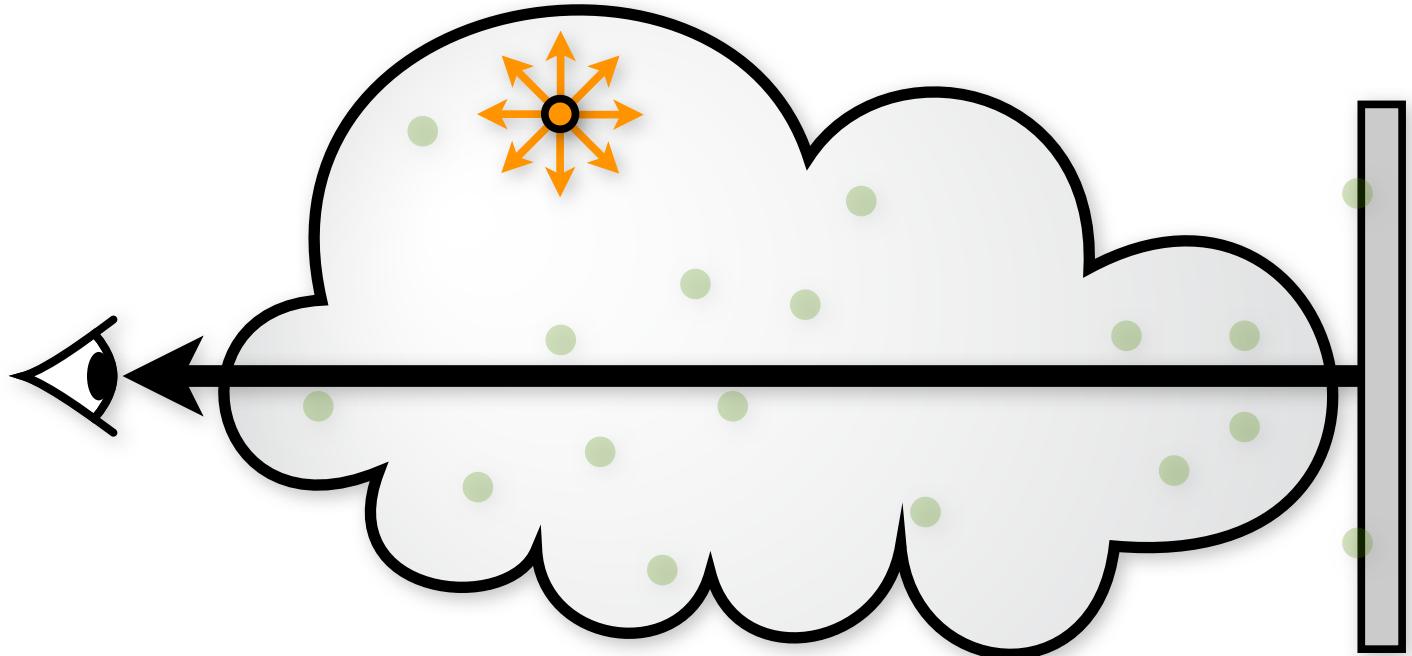
15

Thursday, August 23, 12

- For photon tracing, instead of emitted photons traveling directly to the surfaces, these photons now have some probability of scattering or being absorbed by the medium along their path.
- So we now trace paths that may have vertices within the medium, and at some point these paths will be probabilistically terminated when absorption happens.



Photon Tracing in Participating Media



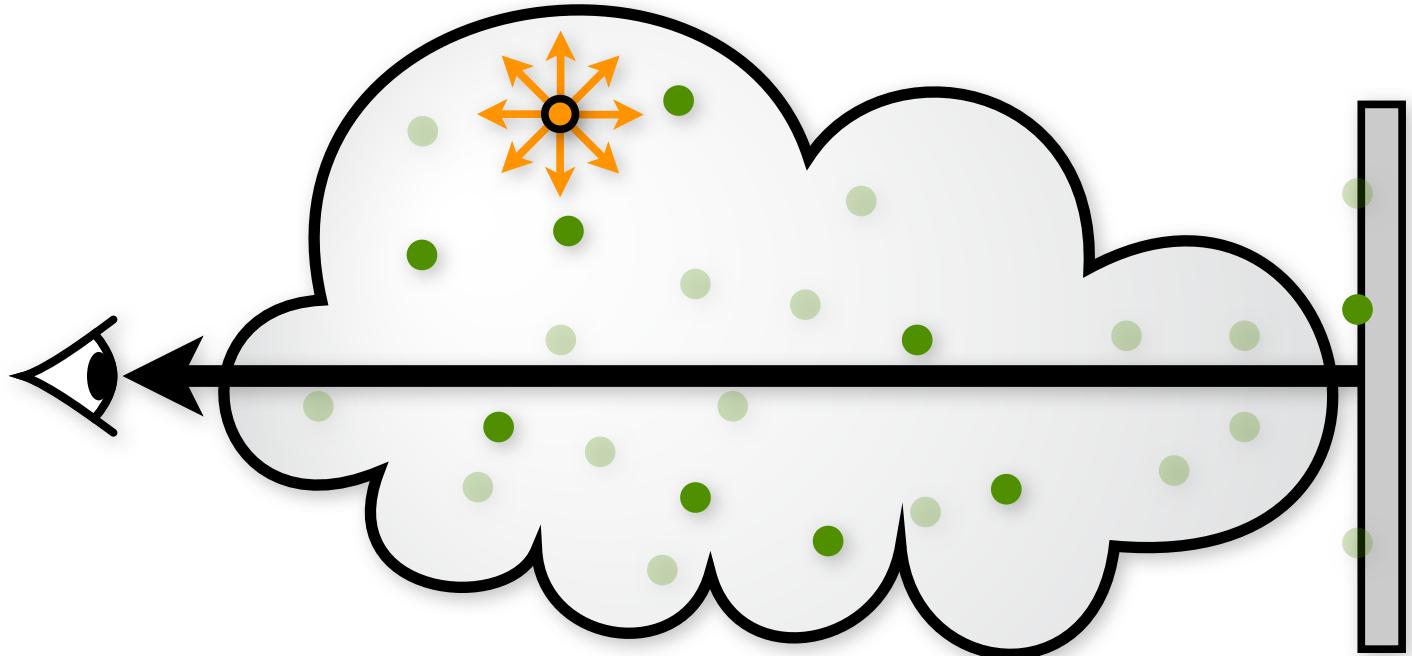
16

Thursday, August 23, 12

- For photon tracing, instead of emitted photons traveling directly to the surfaces, these photons now have some probability of scattering or being absorbed by the medium along their path.
- So we now trace paths that may have vertices within the medium, and at some point these paths will be probabilistically terminated when absorption happens.



Photon Tracing in Participating Media



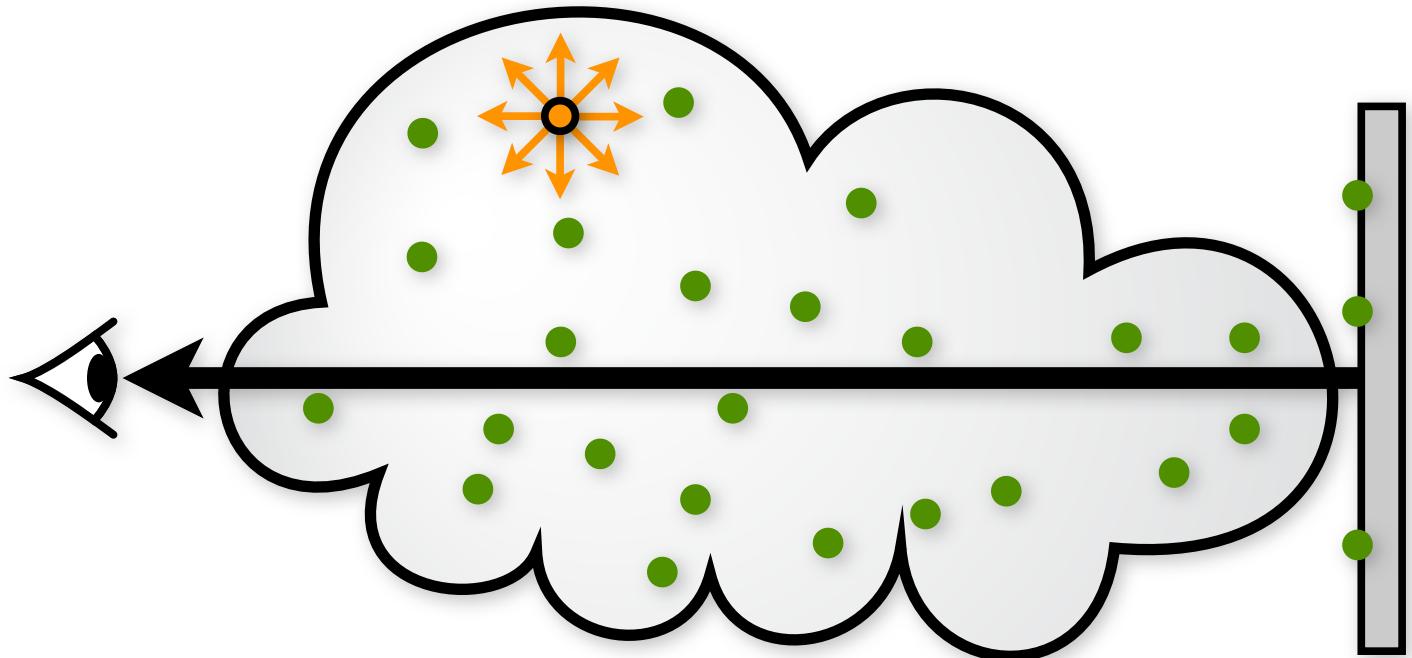
16

Thursday, August 23, 12

- For photon tracing, instead of emitted photons traveling directly to the surfaces, these photons now have some probability of scattering or being absorbed by the medium along their path.
- So we now trace paths that may have vertices within the medium, and at some point these paths will be probabilistically terminated when absorption happens.



Photon Tracing in Participating Media



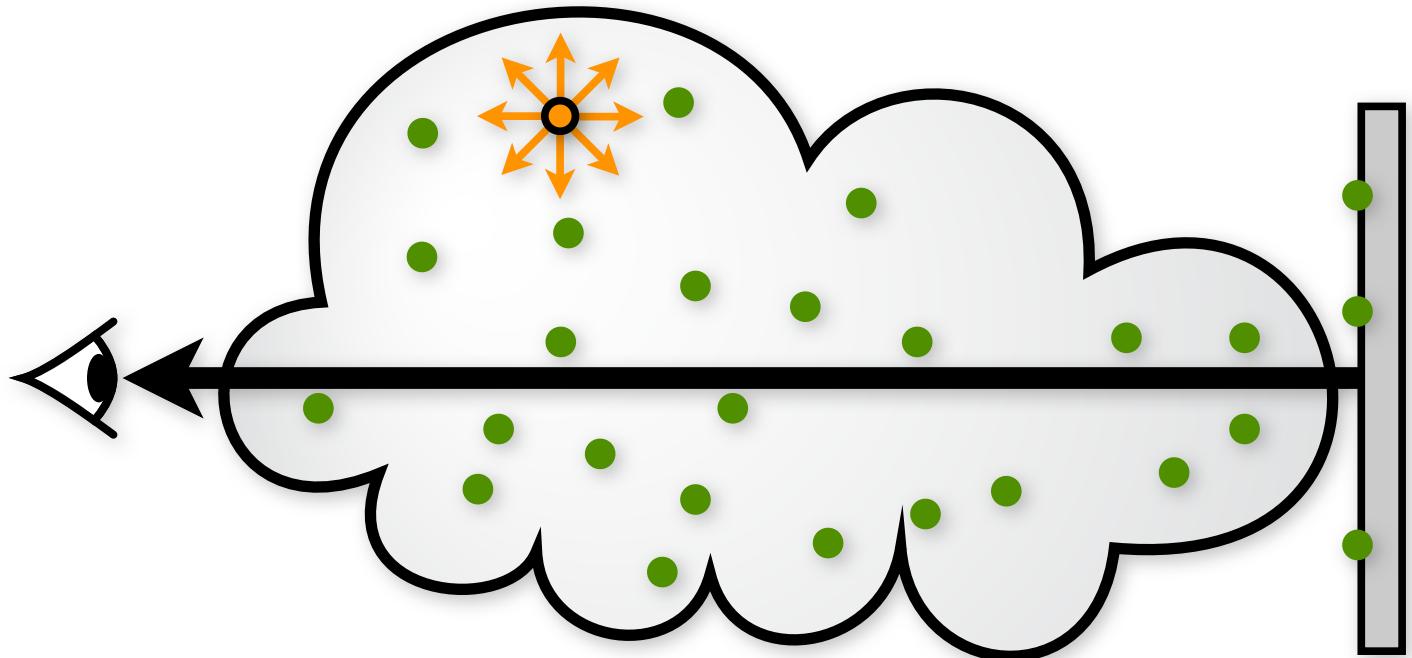
17

Thursday, August 23, 12

- At the end of this process we will have a collection of photons both on surfaces, and also some suspended in the medium



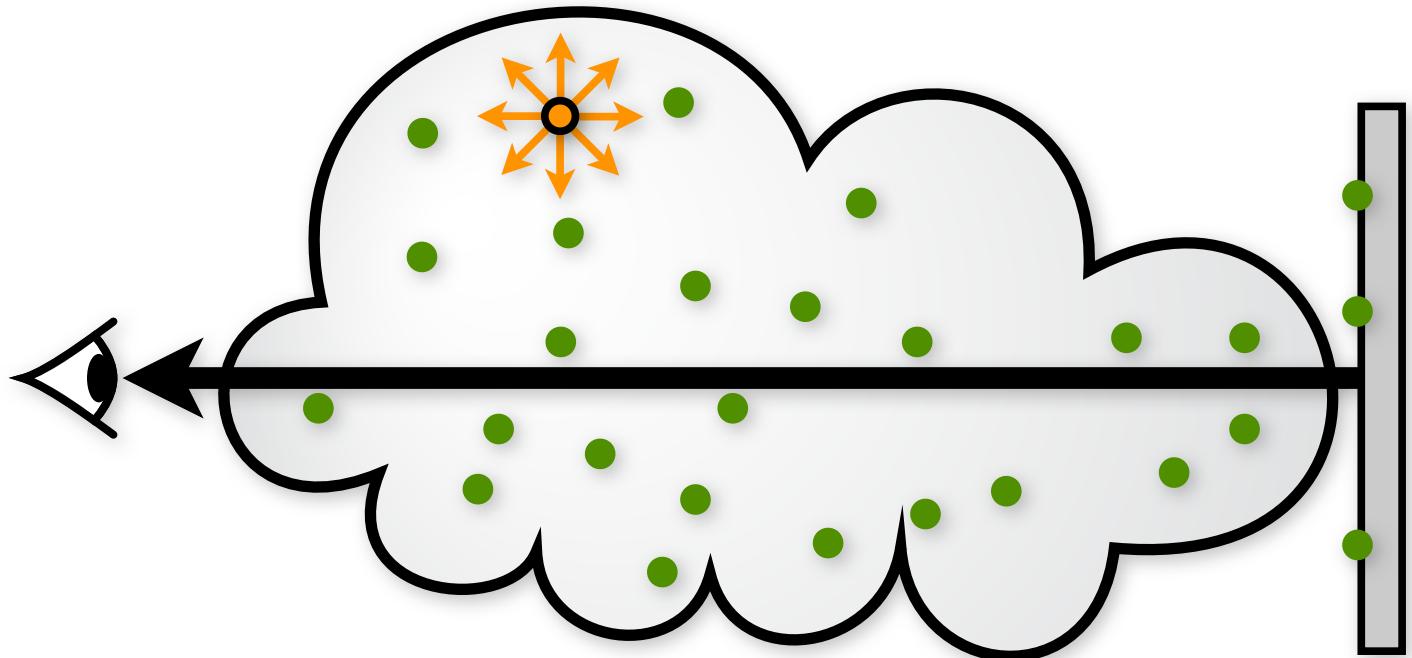
Photon Tracing in Participating Media



- At the end of this process we will have a collection of photons both on surfaces, and also some suspended in the medium



Photon Tracing in Participating Media



- At the end of this process we will have a collection of photons both on surfaces, and also some suspended in the medium



Basic Surface Photon Tracer

```
void vPT(o, ω, Φ)  
    s = nearestSurfaceHit(o, ω)
```

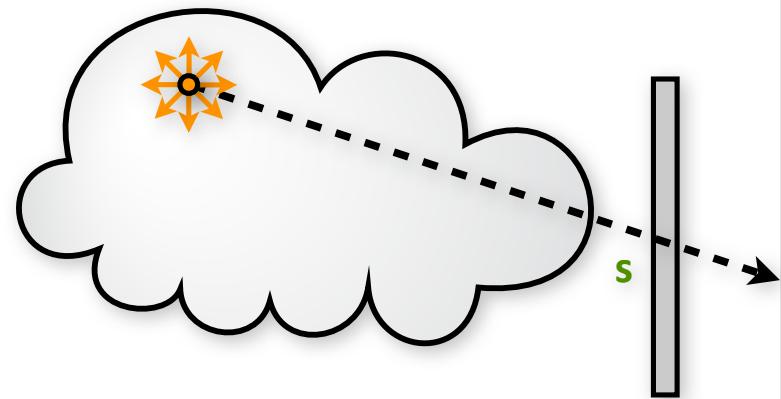


- In the most basic form, a surface photon tracing algorithm might look something like this.
- We trace a ray to figure out the nearest surface hit in that direction, we then propagate the photon to the surface, store it, and recursively scattering based on the BRDF



Basic Surface Photon Tracer

```
void vPT(o, ω, Φ)  
    s = nearestSurfaceHit(o, ω)
```

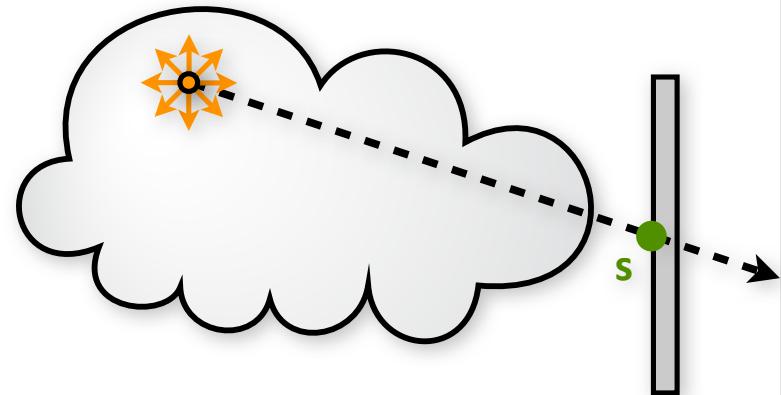


- In the most basic form, a surface photon tracing algorithm might look something like this.
- We trace a ray to figure out the nearest surface hit in that direction, we then propagate the photon to the surface, store it, and recursively scattering based on the BRDF



Basic Surface Photon Tracer

```
void vPT(o, ω, Φ)
    s = nearestSurfaceHit(o, ω)
    o += s*ω      // propagate photon
    storeSurfacePhoton(o, ω, Φ)
```

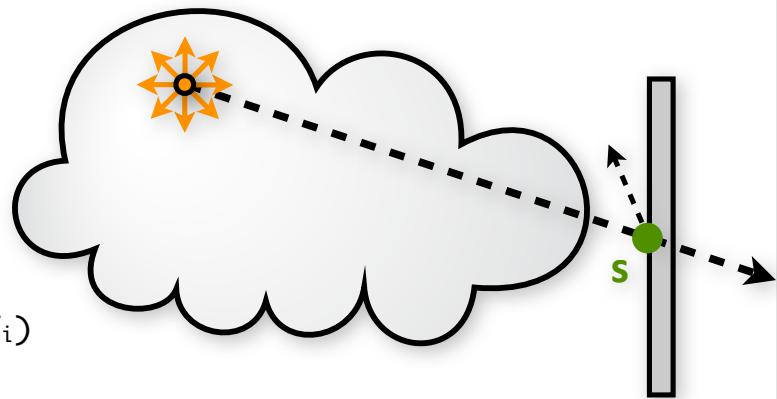


- In the most basic form, a surface photon tracing algorithm might look something like this.
- We trace a ray to figure out the nearest surface hit in that direction, we then propagate the photon to the surface, store it, and recursively scattering based on the BRDF



Basic Surface Photon Tracer

```
void vPT(o, ω, Φ)
    s = nearestSurfaceHit(o, ω)
    o += s*ω      // propagate photon
    storeSurfacePhoton(o, ω, Φ)
    (ωi, pdfi) = sampleBRDF(o, ω)
    return vPT(o, ωi, Φ * BRDF(o, ω, ωi) / pdfi)
```

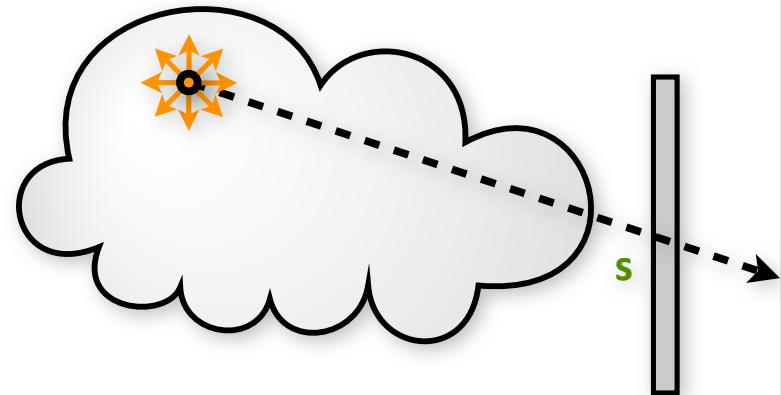


- In the most basic form, a surface photon tracing algorithm might look something like this.
- We trace a ray to figure out the nearest surface hit in that direction, we then propagate the photon to the surface, store it, and recursively scattering based on the BRDF



Basic Volumetric Photon Tracer

```
void vPT(o, ω, Φ)  
    s = nearestSurfaceHit(o, ω)
```



```
o += s*ω          // propagate photon  
storeSurfacePhoton(o, ω, Φ)  
(ωi, pdfi) = sampleBRDF(o, ω)  
return vPT(o, ωi, Φ * BRDF(o, ω, ωi) / pdfi)
```

19

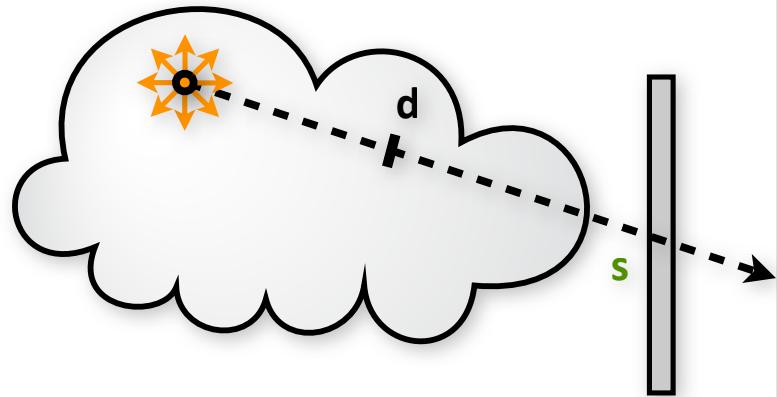
Thursday, August 23, 12

- We can pretty easily generalize this to a volumetric photon tracer
- We will still need to perform the surface computation, but first, we will compute what's called the free flight distance.
- This is the random distance that a photon will travel before probabilistically interacting with the medium
- Now, if this distance d is less than the distance s to the nearest surface, then we have a medium event, and we propagate the photon, and then simulate a recursive medium scattering event
- Only if the distance d is greater than s will the photon actually reach the surface, which is where we use our surface photon scattering code



Basic Volumetric Photon Tracer

```
void vPT(o, ω, Φ)
    s = nearestSurfaceHit(o, ω)
    d = freeFlightDistance(o, ω)
```



```
o += s*ω      // propagate photon
storeSurfacePhoton(o, ω, Φ)
(ωi, pdfi) = sampleBRDF(o, ω)
return vPT(o, ωi, Φ * BRDF(o, ω, ωi) / pdfi)
```

19

Thursday, August 23, 12

- We can pretty easily generalize this to a volumetric photon tracer
- We will still need to perform the surface computation, but first, we will compute what's called the free flight distance.
- This is the random distance that a photon will travel before probabilistically interacting with the medium
- Now, if this distance d is less than the distance s to the nearest surface, then we have a medium event, and we propagate the photon, and then simulate a recursive medium scattering event
- Only if the distance d is greater than s will the photon actually reach the surface, which is where we use our surface photon scattering code

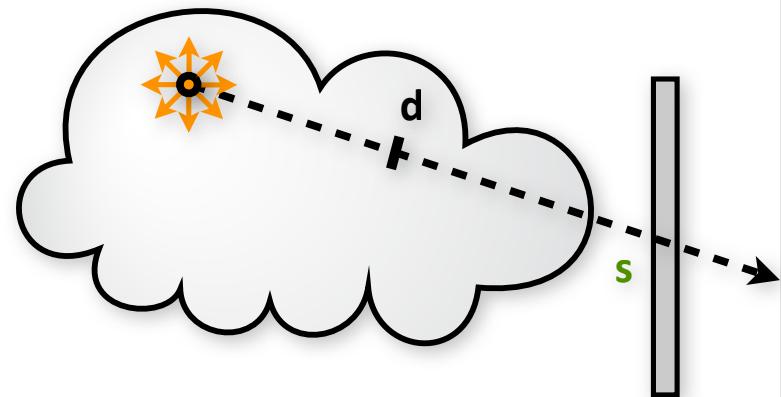


Basic Volumetric Photon Tracer

```

void vPT(o, ω, Φ)
    s = nearestSurfaceHit(o, ω)
    d = freeFlightDistance(o, ω)
    if (d < s)          // media scattering
        ...
    else                  // surface scattering
        o += s*ω         // propagate photon
        storeSurfacePhoton(o, ω, Φ)
        (ωi, pdfi) = sampleBRDF(o, ω)
        return vPT(o, ωi, Φ * BRDF(o, ω, ωi) / pdfi)

```



- We can pretty easily generalize this to a volumetric photon tracer
- We will still need to perform the surface computation, but first, we will compute what's called the free flight distance.
- This is the random distance that a photon will travel before probabilistically interacting with the medium
- Now, if this distance d is less than the distance s to the nearest surface, then we have a medium event, and we propagate the photon, and then simulate a recursive medium scattering event
- Only if the distance d is greater than s will the photon actually reach the surface, which is where we use our surface photon scattering code

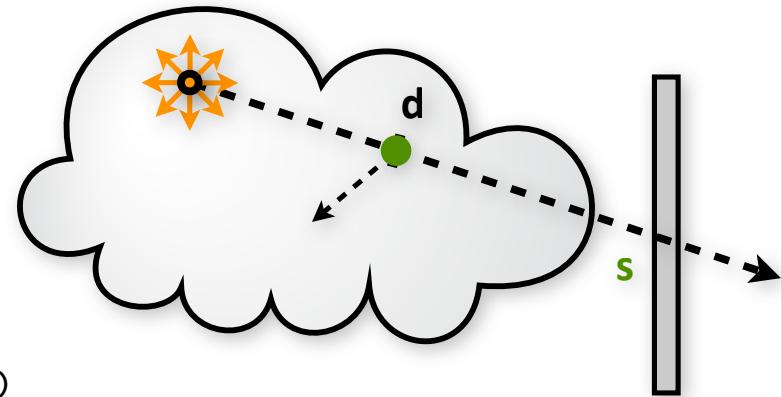


Basic Volumetric Photon Tracer

```

void vPT(o, ω, Φ)
    s = nearestSurfaceHit(o, ω)
    d = freeFlightDistance(o, ω)
    if (d < s)          // media scattering
        o += d*ω         // propagate photon
        storeVolumePhoton(o, ω, Φ)
    return vPT(o, samplePFC(), Φ * σs / σt)
    else                // surface scattering
        o += s*ω         // propagate photon
        storeSurfacePhoton(o, ω, Φ)
        (ωi, pdfi) = sampleBRDF(o, ω)
    return vPT(o, ωi, Φ * BRDF(o, ω, ωi) / pdfi)

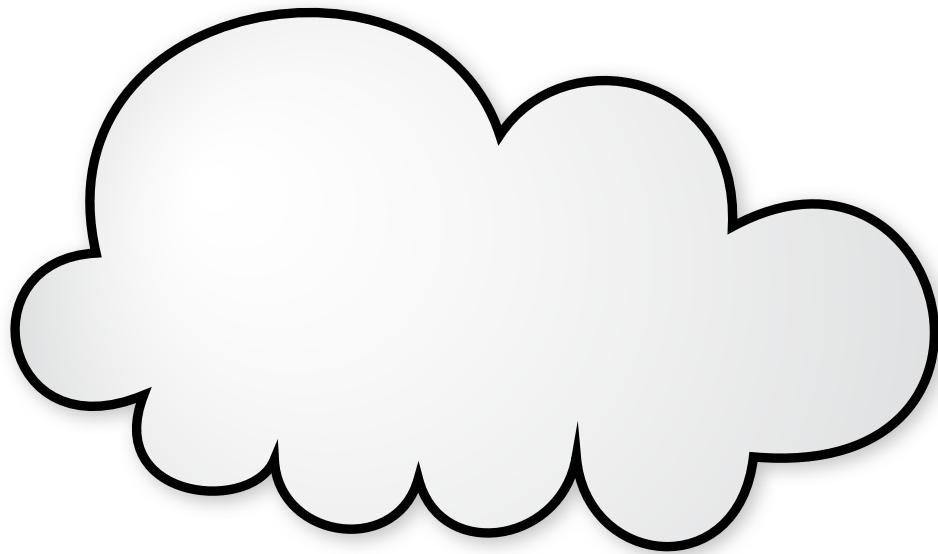
```



- We can pretty easily generalize this to a volumetric photon tracer
- We will still need to perform the surface computation, but first, we will compute what's called the free flight distance.
- This is the random distance that a photon will travel before probabilistically interacting with the medium
- Now, if this distance d is less than the distance s to the nearest surface, then we have a medium event, and we propagate the photon, and then simulate a recursive medium scattering event
- Only if the distance d is greater than s will the photon actually reach the surface, which is where we use our surface photon scattering code



Volume Rendering Equation



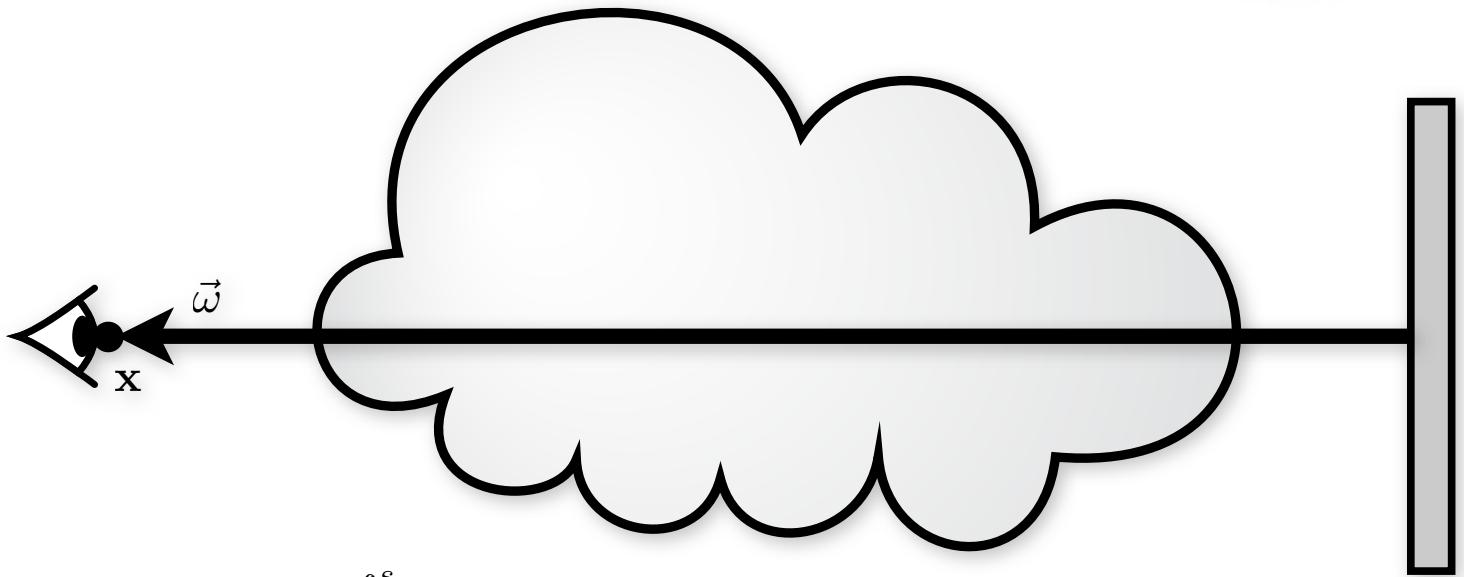
20

Thursday, August 23, 12

- For rendering the final image, we also need to consider not just the rendering equation, but the volume rendering equation
- Which is shown below
- And this is really just the sum to two main terms



Volume Rendering Equation



$$L(\mathbf{x}, \vec{\omega}) = \int_0^s T_r(\mathbf{x} \leftrightarrow \mathbf{x}_t) \sigma_s(\mathbf{x}_t) L_i(\mathbf{x}_t, \vec{\omega}) dt + T_r(\mathbf{x} \leftrightarrow \mathbf{x}_s) L(\mathbf{x}_s, \vec{\omega})$$

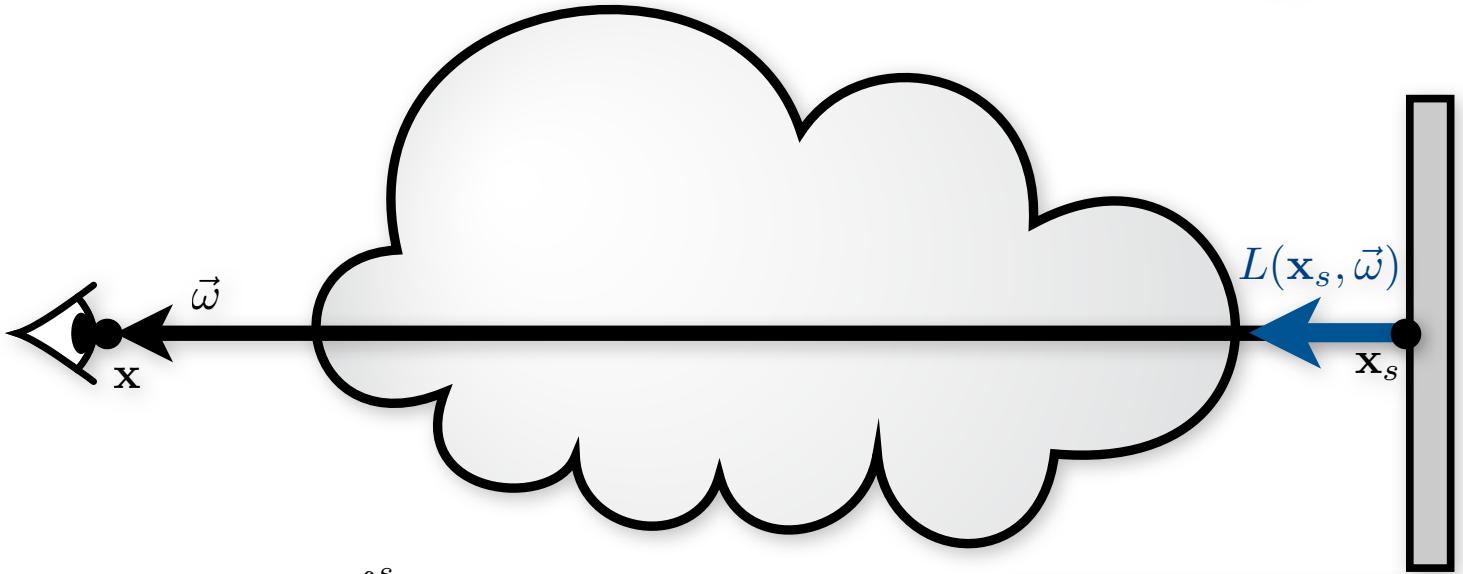
20

Thursday, August 23, 12

- For rendering the final image, we also need to consider not just the rendering equation, but the volume rendering equation
- Which is shown below
- And this is really just the sum to two main terms



Volume Rendering Equation



$$L(\mathbf{x}, \vec{\omega}) = \int_0^s T_r(\mathbf{x} \leftrightarrow \mathbf{x}_t) \sigma_s(\mathbf{x}_t) L_i(\mathbf{x}_t, \vec{\omega}) dt + T_r(\mathbf{x} \leftrightarrow \mathbf{x}_s) L(\mathbf{x}_s, \vec{\omega})$$

Reduced surface radiance

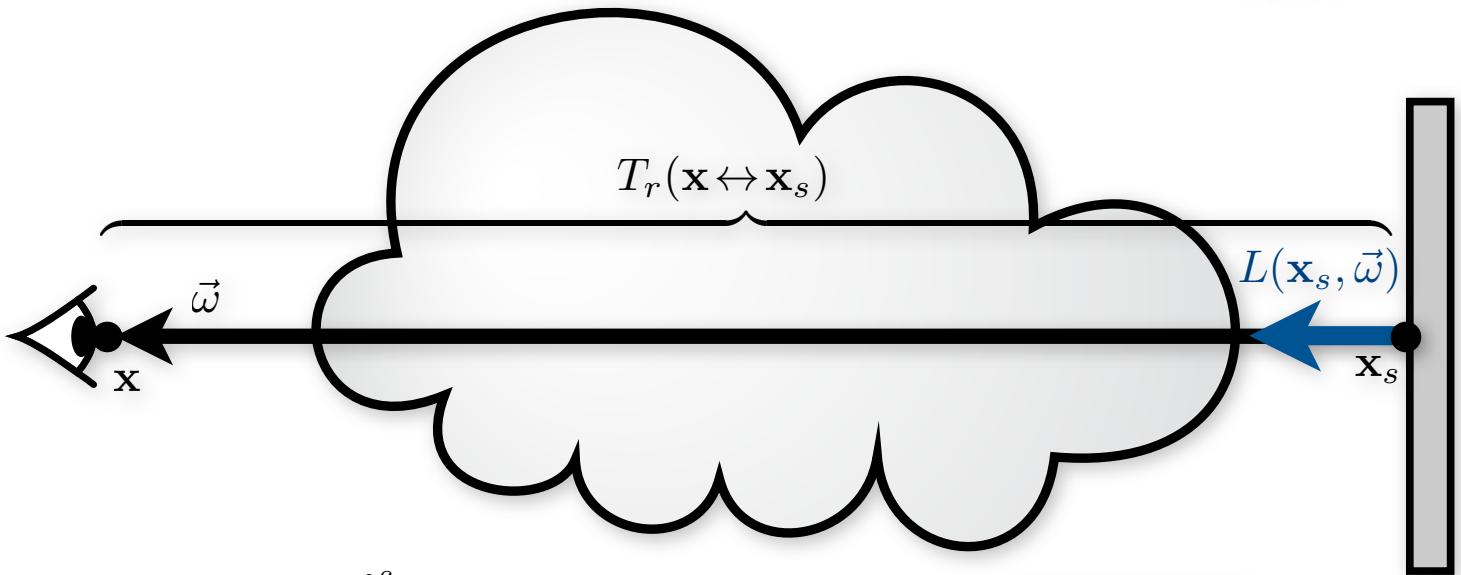
21

Thursday, August 23, 12

- The right-hand term incorporates lighting arriving from a surface
- However, before reaching the eye, this radiance must travel through the medium and so is attenuated by a transmission term



Volume Rendering Equation



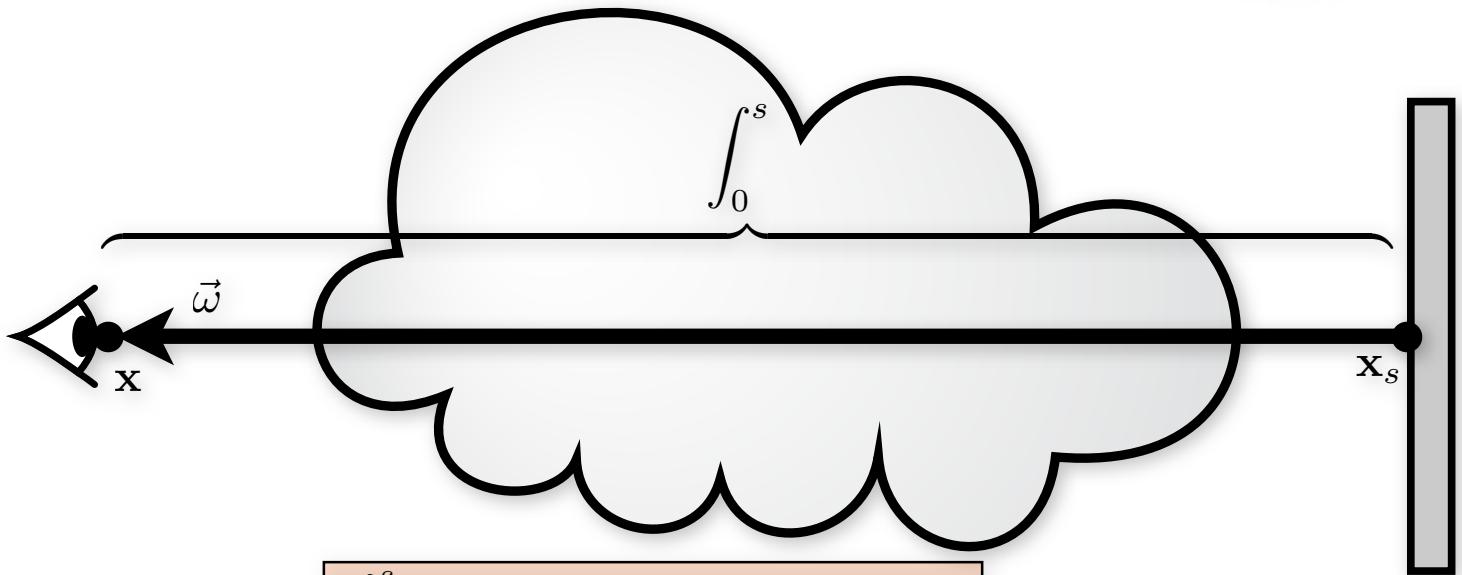
$$L(\mathbf{x}, \vec{\omega}) = \int_0^s T_r(\mathbf{x} \leftrightarrow \mathbf{x}_t) \sigma_s(\mathbf{x}_t) L_i(\mathbf{x}_t, \vec{\omega}) dt + T_r(\mathbf{x} \leftrightarrow \mathbf{x}_s) L(\mathbf{x}_s, \vec{\omega})$$

Transmittance: fractional visibility between two points

- The right-hand term incorporates lighting arriving from a surface
- However, before reaching the eye, this radiance must travel through the medium and so is attenuated by a transmission term, which is really just a fractional visibility between the two points



Volume Rendering Equation



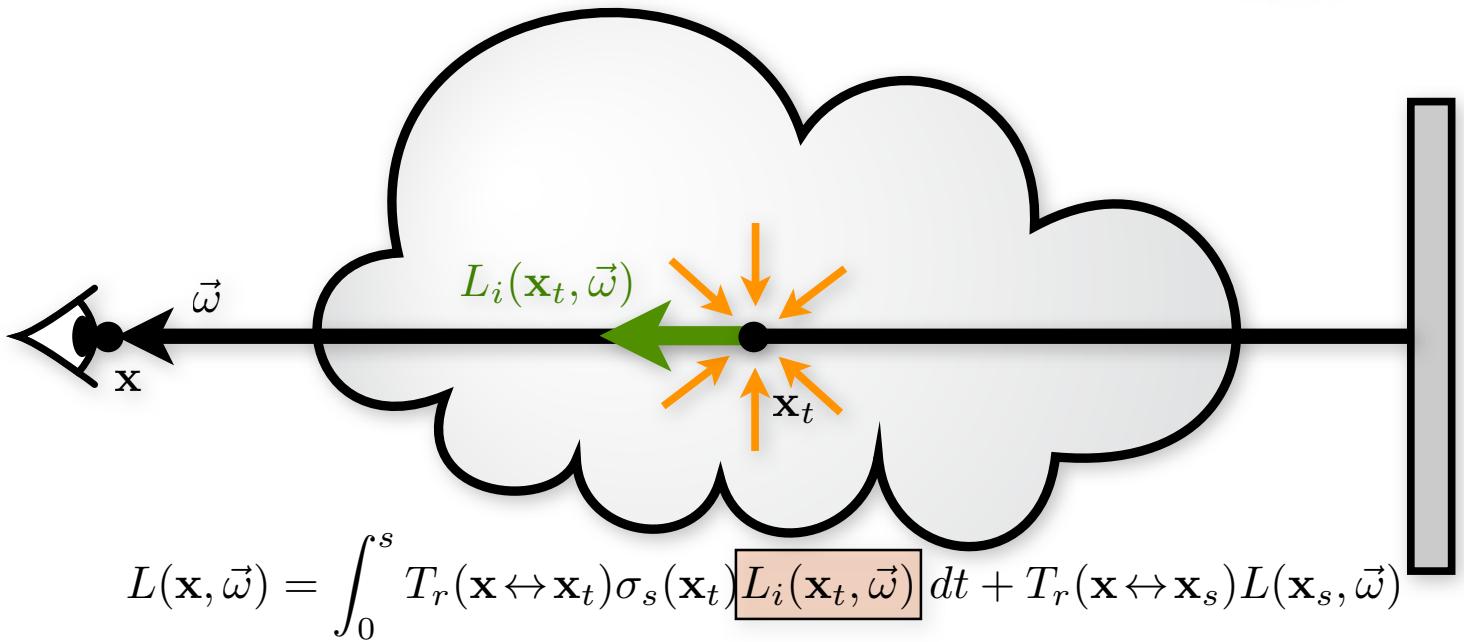
$$L(\mathbf{x}, \vec{\omega}) = \int_0^s T_r(\mathbf{x} \leftrightarrow \mathbf{x}_t) \sigma_s(\mathbf{x}_t) L_i(\mathbf{x}_t, \vec{\omega}) dt + T_r(\mathbf{x} \leftrightarrow \mathbf{x}_s) L(\mathbf{x}_s, \vec{\omega})$$

Accumulated in-scattered radiance

- However, light may also actually come from the medium
- The left-hand term integrates the scattering of light from the medium along the whole length of the ray



Volume Rendering Equation



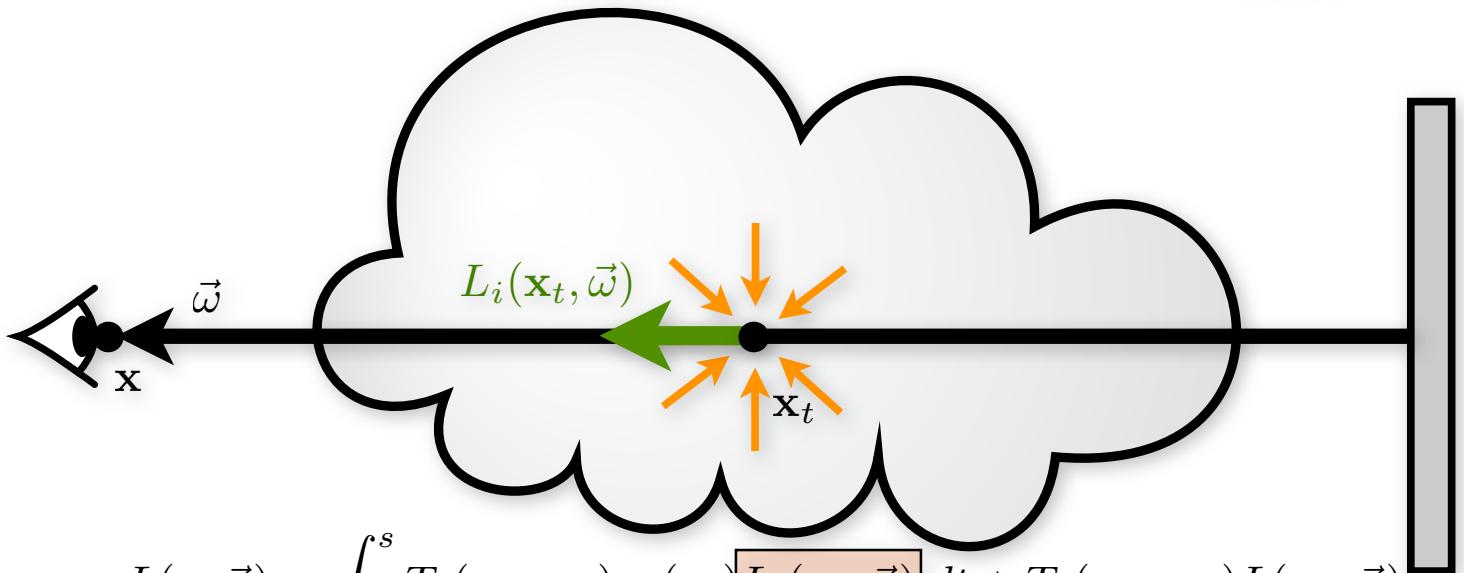
24

Thursday, August 23, 12

- And the main quantity that is integrated, L_i , is in-scattered radiance
- L_i itself is an integral. it represents the amount of light that reaches some point in the volume (from any other location in the scene), and then subsequently scatters towards the eye.
- This is similar to the reflection equation on surfaces, but using the phase function
- Computing this in-scattered radiance is really what makes volume rendering expensive



Volume Rendering Equation



$$L(\mathbf{x}, \vec{\omega}) = \int_0^s T_r(\mathbf{x} \leftrightarrow \mathbf{x}_t) \sigma_s(\mathbf{x}_t) [L_i(\mathbf{x}_t, \vec{\omega})] dt + T_r(\mathbf{x} \leftrightarrow \mathbf{x}_s) L(\mathbf{x}_s, \vec{\omega})$$
$$[L_i(\mathbf{x}_t, \vec{\omega})] = \int_{\Omega_{4\pi}} p(\mathbf{x}_t, \vec{\omega}_t, \vec{\omega}) L(\mathbf{x}_t, \vec{\omega}_t) d\omega_t$$

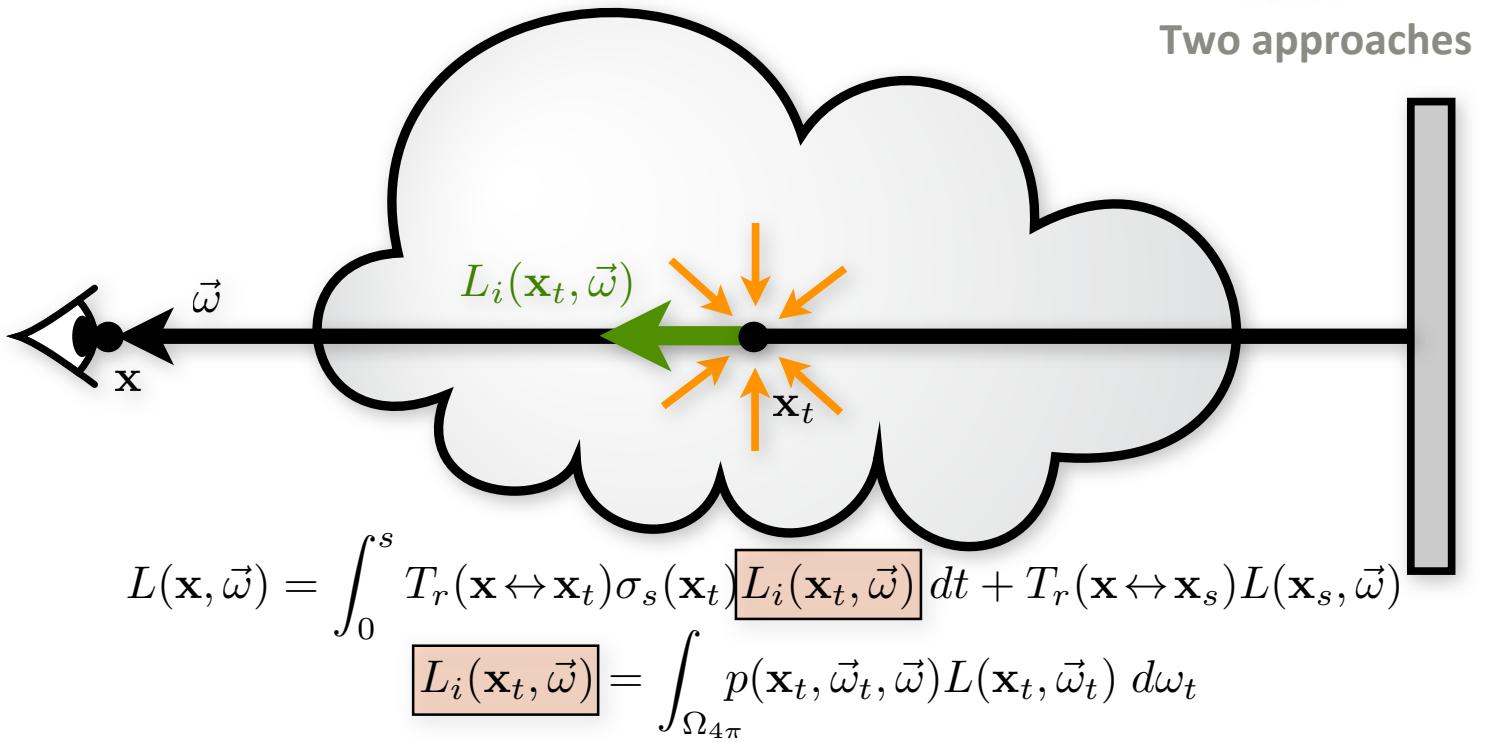
24

Thursday, August 23, 12

- And the main quantity that is integrated, L_i , is inscattered radiance
- L_i itself is an integral. it represents the amount of light that reaches some point in the volume (from any other location in the scene), and then subsequently scatters towards the eye.
- This is similar to the reflection equation on surfaces, but using the phase function
- Computing this inscattered radiance is really what makes volume rendering expensive



Volume Rendering Equation



24

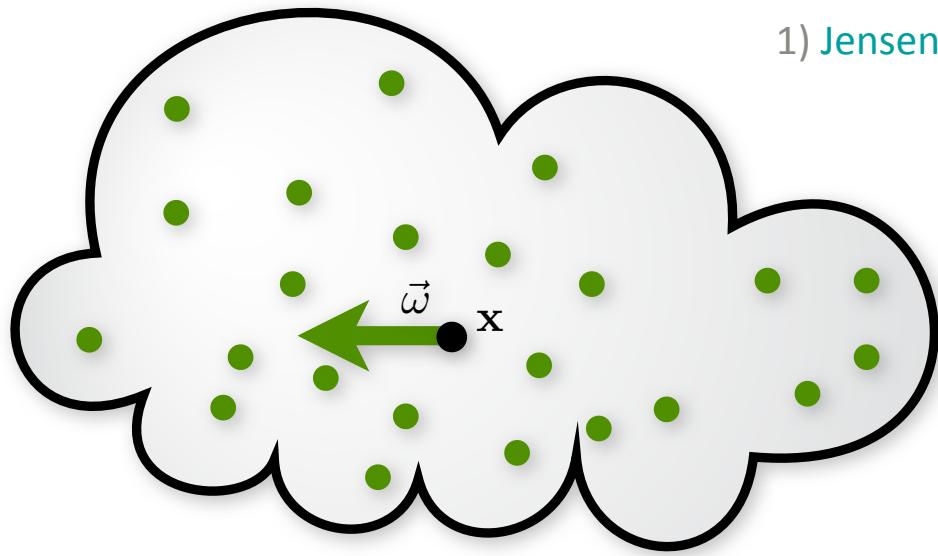
Thursday, August 23, 12

- And the main quantity that is integrated, L_i , is inscattered radiance
- L_i itself is an integral. it represents the amount of light that reaches some point in the volume (from any other location in the scene), and then subsequently scatters towards the eye.
- This is similar to the reflection equation on surfaces, but using the phase function
- Computing this inscattered radiance is really what makes volume rendering expensive



Volumetric Radiance Estimate

1) Jensen & Christensen 98



$$L_i(\mathbf{x}_t, \vec{\omega}) = \int_{\Omega_{4\pi}} p(\mathbf{x}_t, \vec{\omega}_t, \vec{\omega}) L(\mathbf{x}_t, \vec{\omega}_t) d\omega_t$$

25

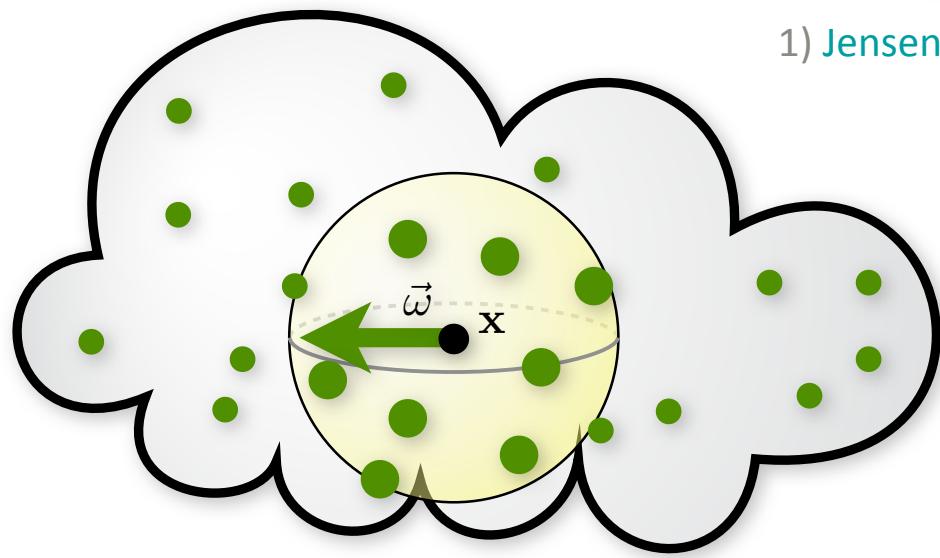
Thursday, August 23, 12

- But we have the photon map, and we can simply find photons within the local region, just like we did at surfaces
- However, instead of computing a density on a surface, we compute a volume density, so we divide by the volume of the sphere



Volumetric Radiance Estimate

1) Jensen & Christensen 98



$$L_i(\mathbf{x}, \vec{\omega}) \approx \sum_{i=1}^k p(\mathbf{x}, \vec{\omega}_i, \vec{\omega}) \frac{\Phi_i(\mathbf{x}, \vec{\omega}_i)}{\mathcal{V}(\mathbf{x})}$$

25

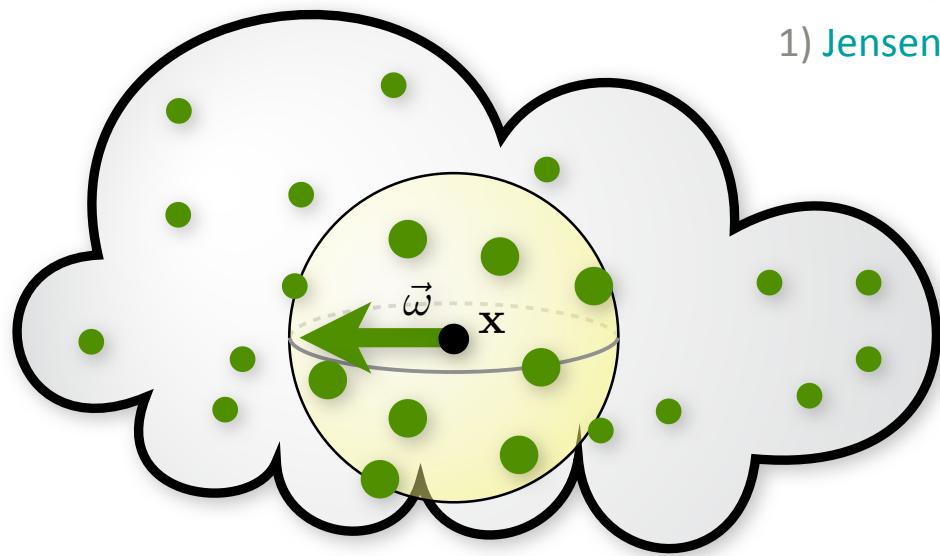
Thursday, August 23, 12

- But we have the photon map, and we can simply find photons within the local region, just like we did at surfaces
- However, instead of computing a density on a surface, we compute a volume density, so we divide by the volume of the sphere



Volumetric Radiance Estimate

1) Jensen & Christensen 98



$$L_i(\mathbf{x}, \vec{\omega}) \approx \sum_{i=1}^k p(\mathbf{x}, \vec{\omega}_i, \vec{\omega}) \frac{\Phi_i(\mathbf{x}, \vec{\omega}_i)}{\frac{4}{3}\pi r(\mathbf{x})^3}$$

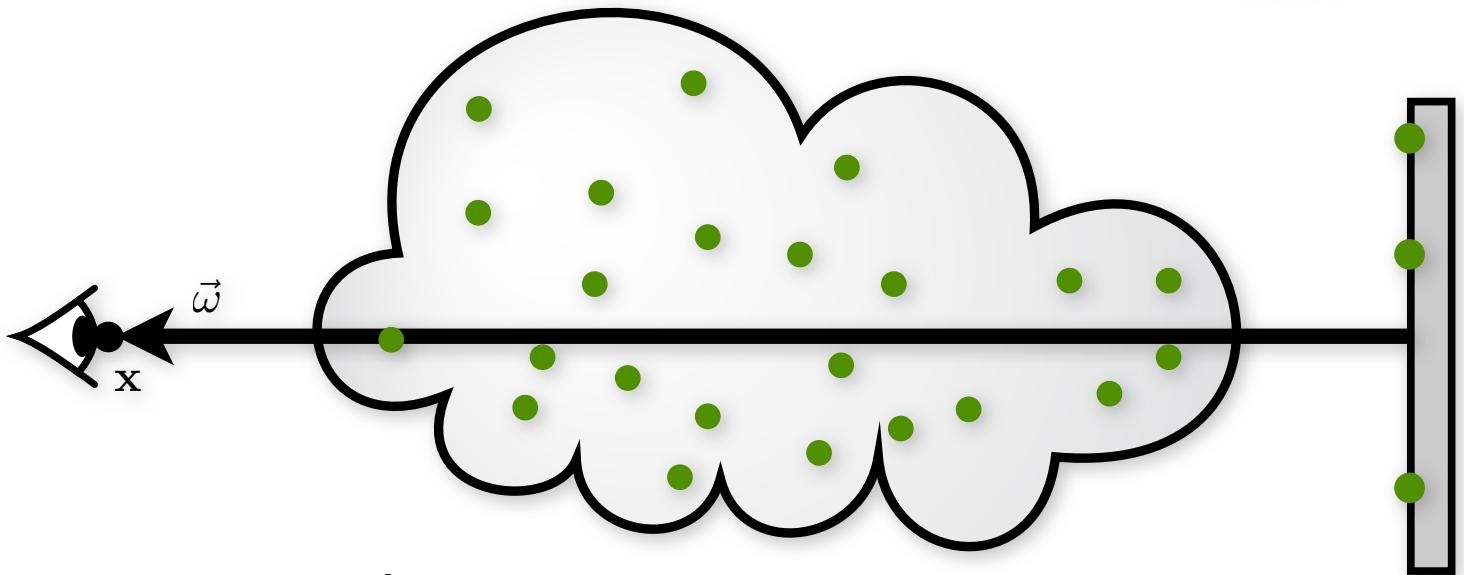
25

Thursday, August 23, 12

- But we have the photon map, and we can simply find photons within the local region, just like we did at surfaces
- However, instead of computing a density on a surface, we compute a volume density, so we divide by the volume of the sphere



Volume Rendering Equation



$$L(\mathbf{x}, \vec{\omega}) = \int_0^s T_r(\mathbf{x} \leftrightarrow \mathbf{x}_t) \sigma_s(\mathbf{x}_t) L_i(\mathbf{x}_t, \vec{\omega}) dt + T_r(\mathbf{x} \leftrightarrow \mathbf{x}_s) L(\mathbf{x}_s, \vec{\omega})$$

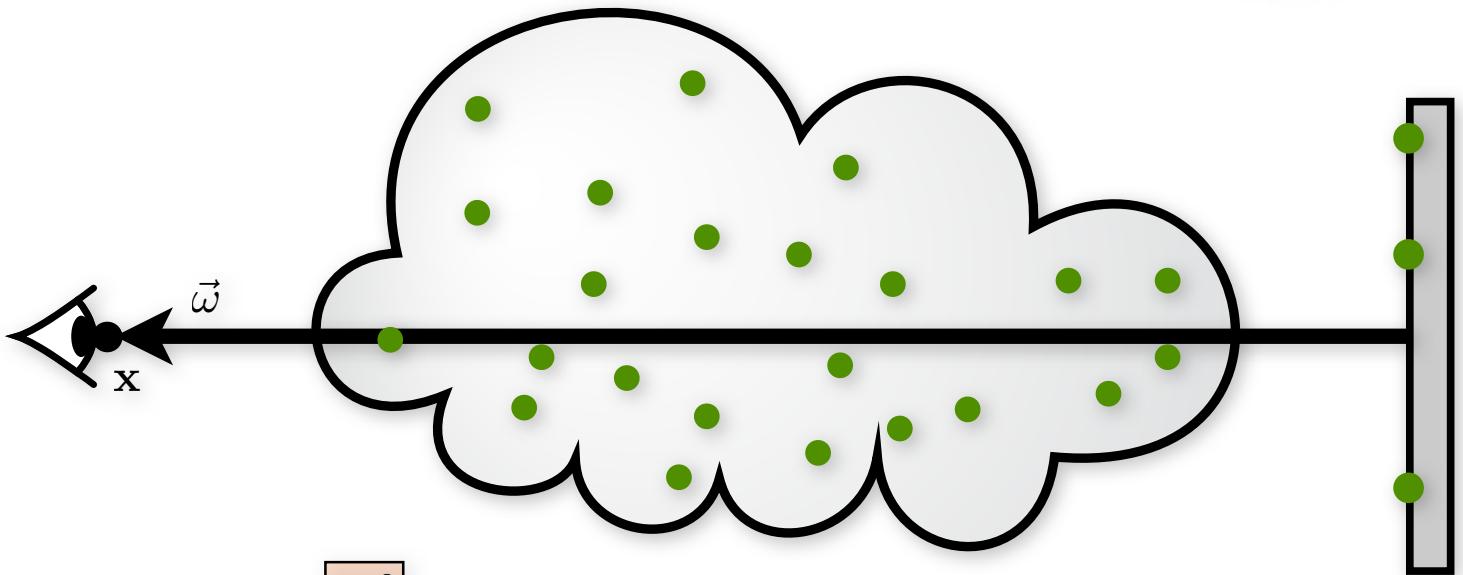
26

Thursday, August 23, 12

- However, we still need to compute the outer integral
- And we can do this by using a simple Riemann sum, which results in ray marching



Volume Rendering Equation



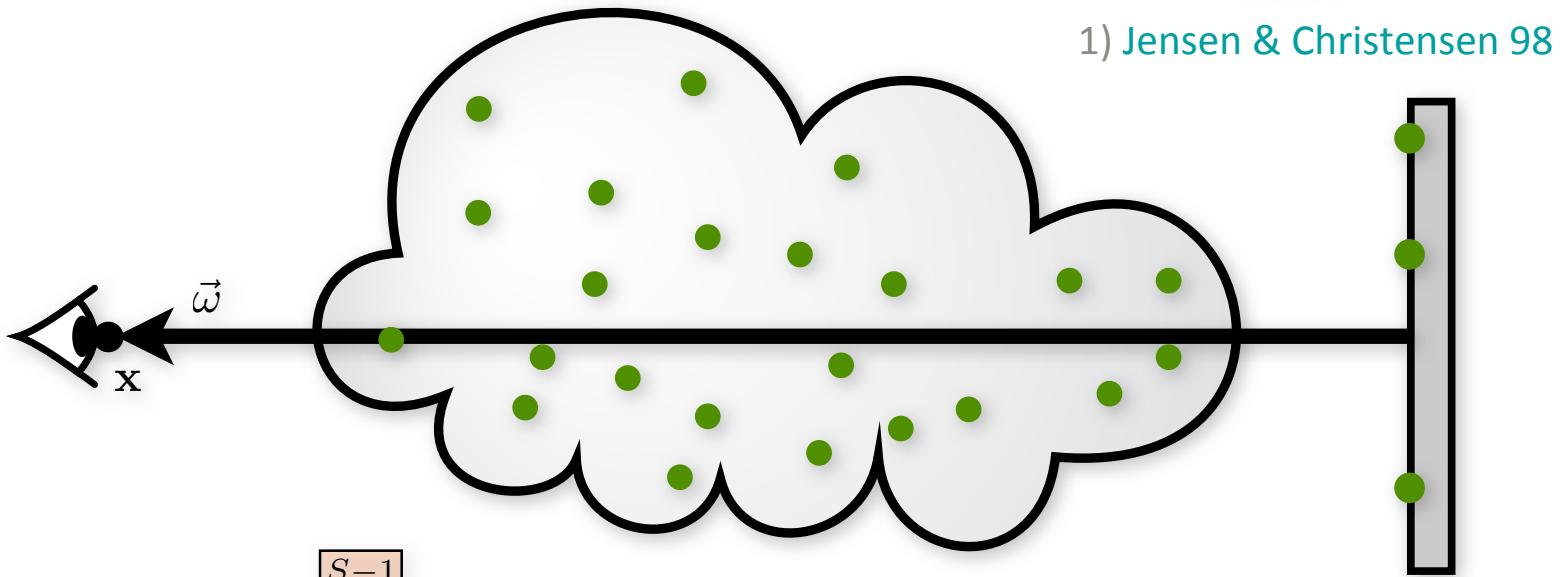
$$L(\mathbf{x}, \vec{\omega}) = \int_0^s T_r(\mathbf{x} \leftrightarrow \mathbf{x}_t) \sigma_s(\mathbf{x}_t) L_i(\mathbf{x}_t, \vec{\omega}) dt + T_r(\mathbf{x} \leftrightarrow \mathbf{x}_s) L(\mathbf{x}_s, \vec{\omega})$$

Approximate/compute using Riemann sum

- However, we still need to compute the outer integral
- And we can do this by using a simple Riemann sum, which results in ray marching



Ray Marching



$$L(\mathbf{x}, \vec{\omega}) \approx \sum_{t=0}^{S-1} T_r(\mathbf{x} \leftrightarrow \mathbf{x}_t) \sigma_s(\mathbf{x}_t) L_i(\mathbf{x}_t, \vec{\omega}) \Delta_t + T_r(\mathbf{x} \leftrightarrow \mathbf{x}_s) L(\mathbf{x}_s, \vec{\omega})$$

Approximate/compute using Riemann sum

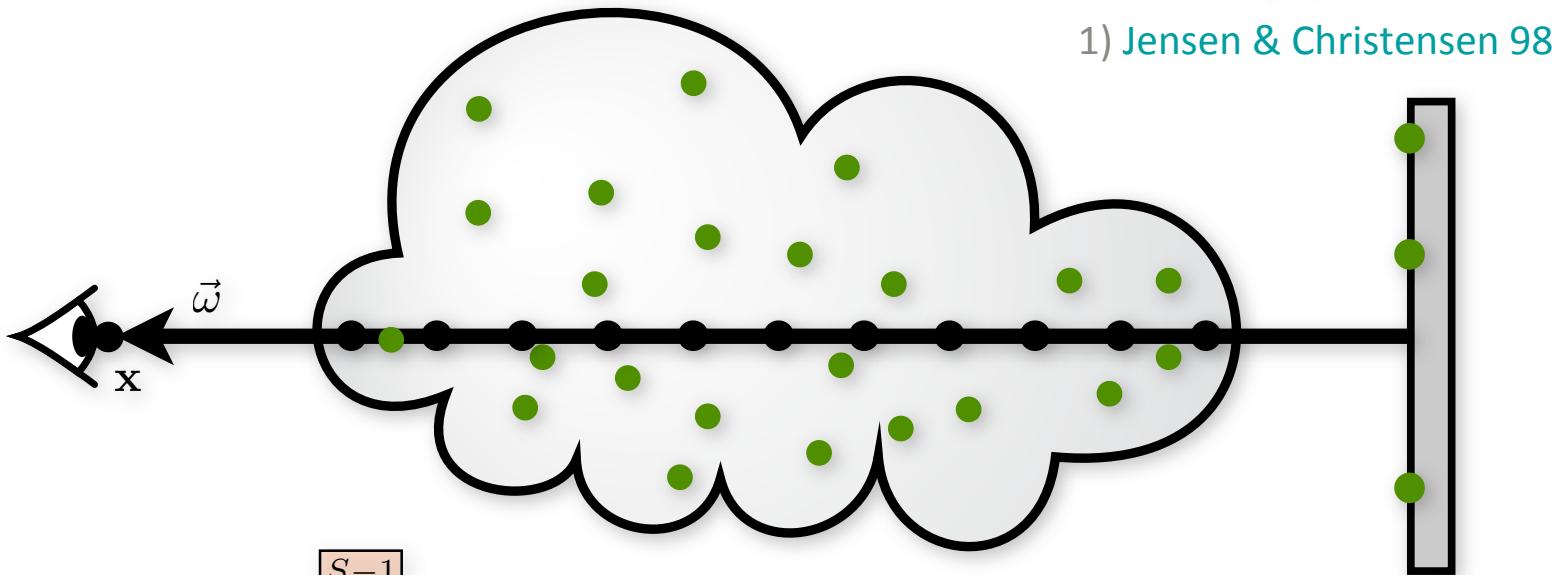
27

Thursday, August 23, 12

- However, we still need to compute the outer integral
- And we can do this by using a simple Riemann sum, which results in ray marching
- Where we effectively march along the ray, and at each point we will estimate the integrand, and just add all these evaluations together.



Ray Marching



$$L(\mathbf{x}, \vec{\omega}) \approx \sum_{t=0}^{S-1} T_r(\mathbf{x} \leftrightarrow \mathbf{x}_t) \sigma_s(\mathbf{x}_t) L_i(\mathbf{x}_t, \vec{\omega}) \Delta_t + T_r(\mathbf{x} \leftrightarrow \mathbf{x}_s) L(\mathbf{x}_s, \vec{\omega})$$

Approximate/compute using Riemann sum

27

Thursday, August 23, 12

- However, we still need to compute the outer integral
- And we can do this by using a simple Riemann sum, which results in ray marching
- Where we effectively march along the ray, and at each point we will estimate the integrand, and just add all these evaluations together.



Ray Marching

1) Jensen & Christensen 98



$$L(\mathbf{x}, \vec{\omega}) \approx \sum_{t=0}^{S-1} T_r(\mathbf{x} \leftrightarrow \mathbf{x}_t) \sigma_s(\mathbf{x}_t) L_i(\mathbf{x}_t, \vec{\omega}) \Delta_t + T_r(\mathbf{x} \leftrightarrow \mathbf{x}_s) L(\mathbf{x}_s, \vec{\omega})$$

Approximate/compute using Riemann sum

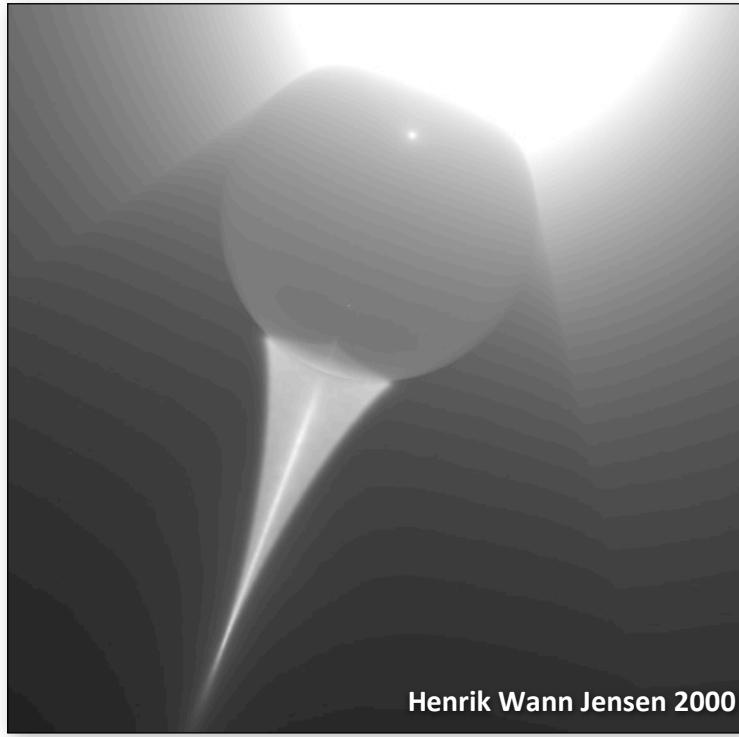
27

Thursday, August 23, 12

- However, we still need to compute the outer integral
- And we can do this by using a simple Riemann sum, which results in ray marching
- Where we effectively march along the ray, and at each point we will estimate the integrand, and just add all these evaluations together.



Volume Caustics



- And by doing this, we can get really complex lighting effects like this volume caustic



Subsurface Scattering



Henrik Wann Jensen

29

Thursday, August 23, 12

- Or, if we imagine the surface of this statue to just be the boundary of some participating medium, we get soft natural subsurface scattering within this marble bust

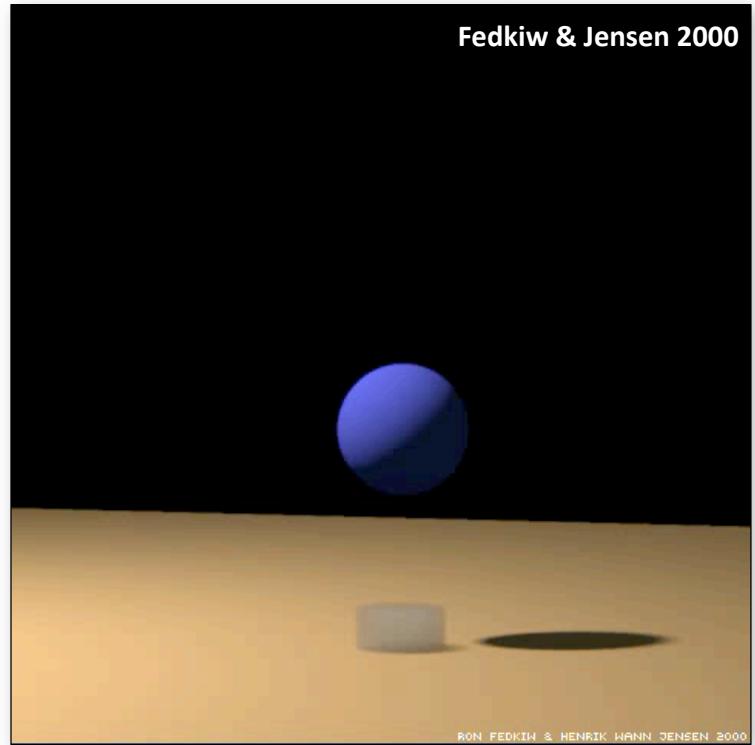


Rising Smoke

Fedkiw & Jensen 2000



Fedkiw & Jensen 2000

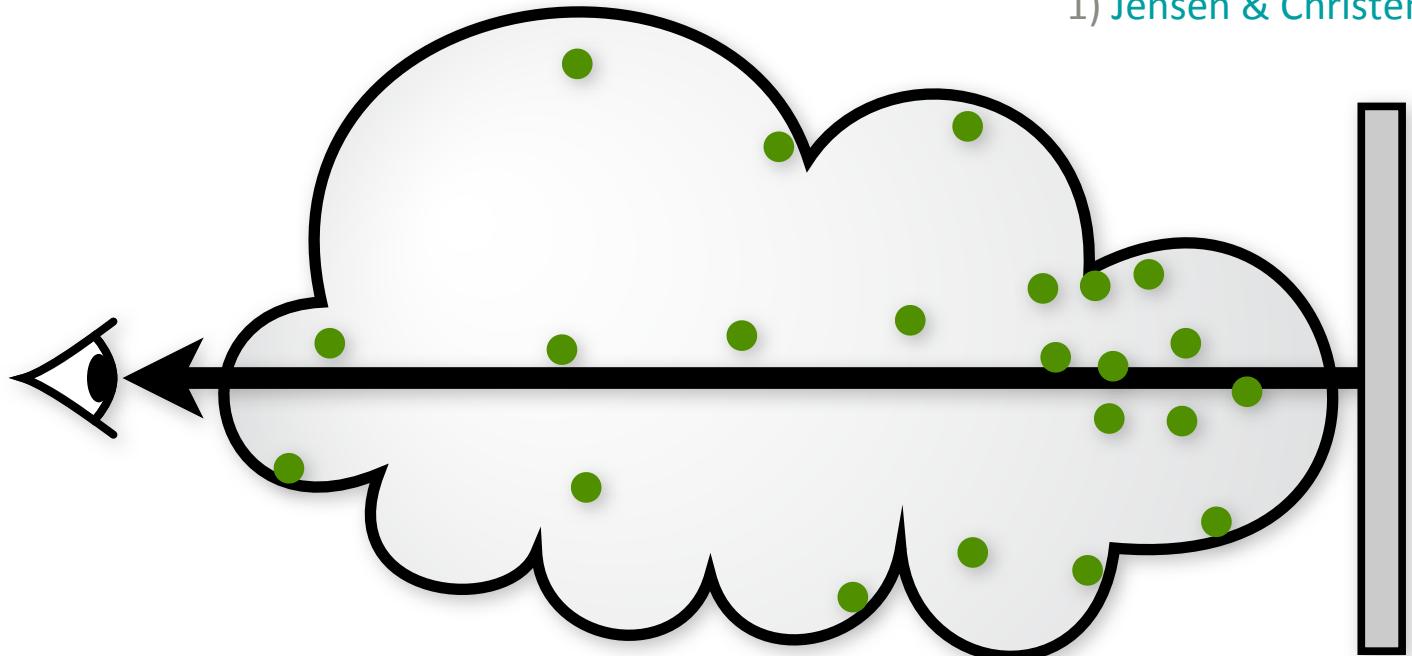


- We can also simulate smoke, and we can see that



Volumetric Photon Mapping

1) Jensen & Christensen 98





Volumetric Photon Mapping

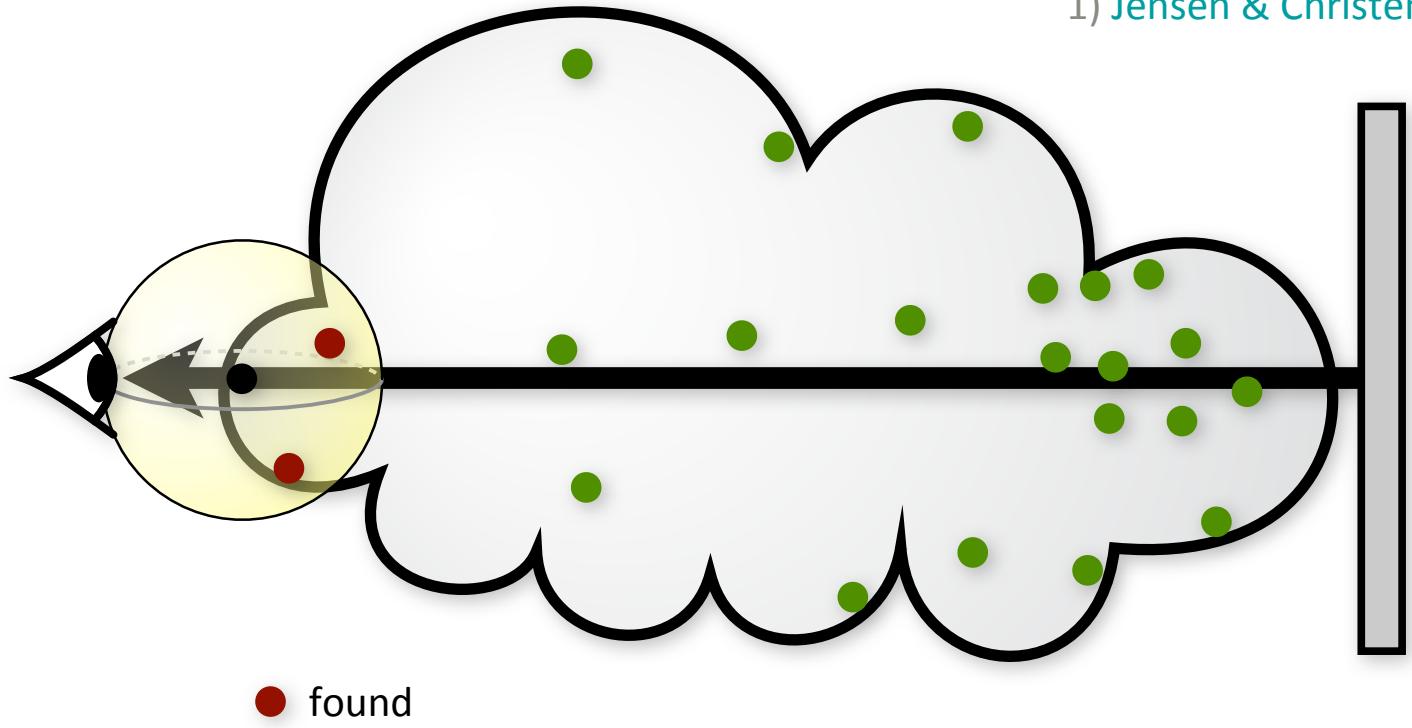
1) Jensen & Christensen 98





Volumetric Photon Mapping

1) Jensen & Christensen 98

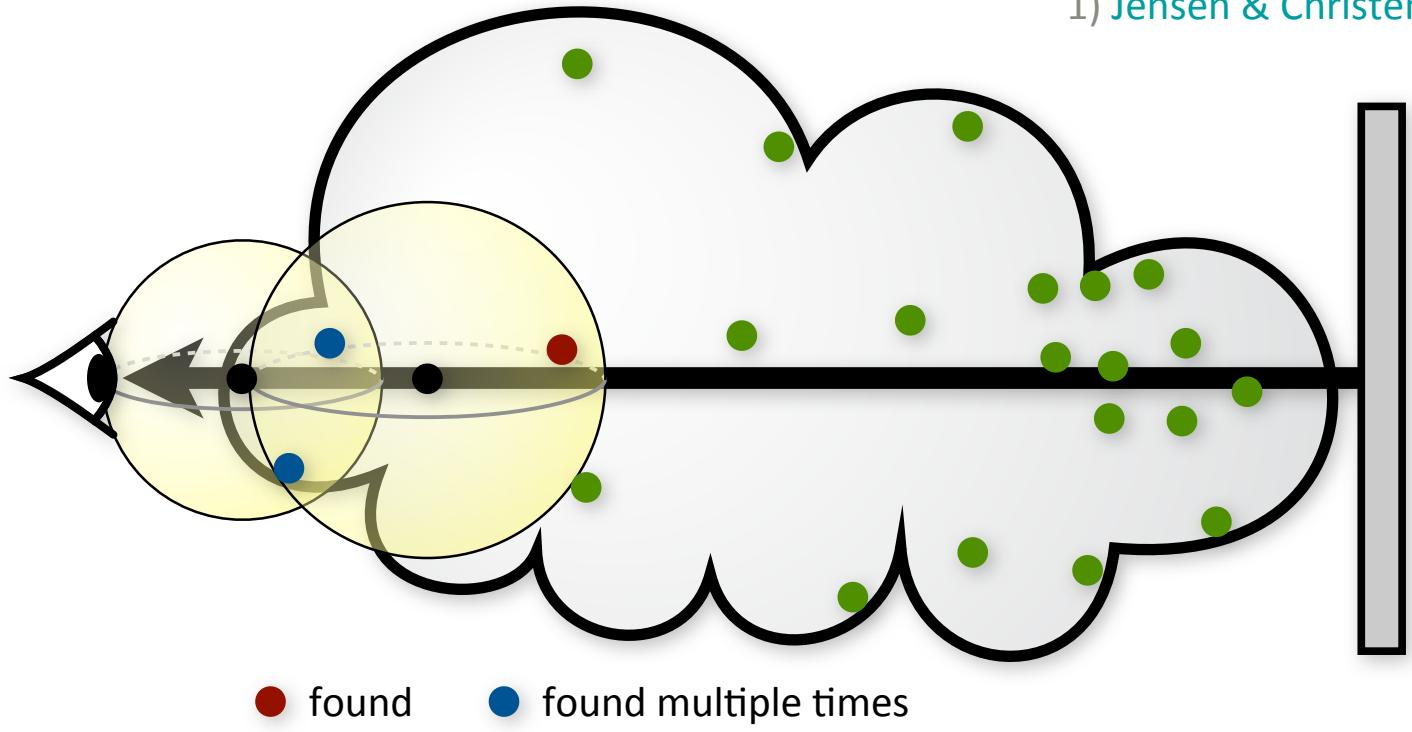


● found



Volumetric Photon Mapping

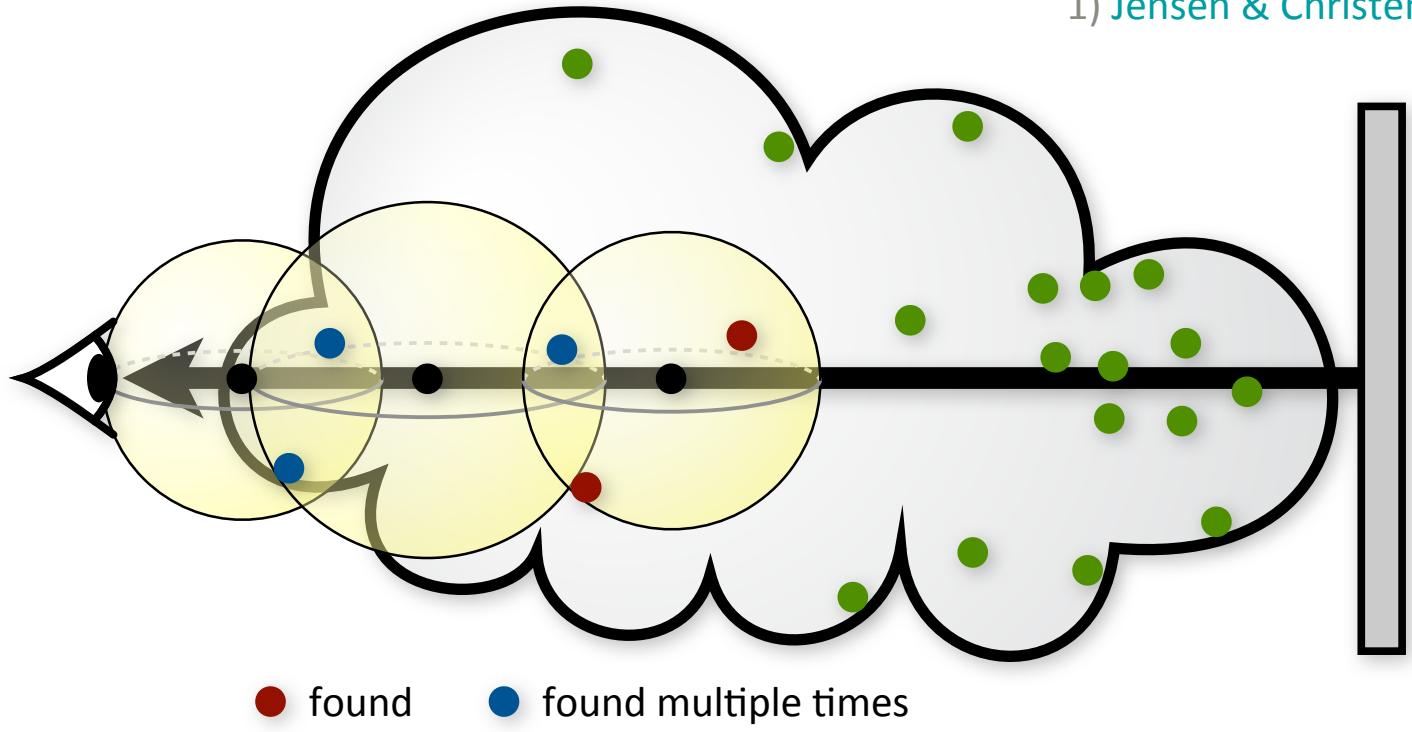
1) Jensen & Christensen 98





Volumetric Photon Mapping

1) Jensen & Christensen 98

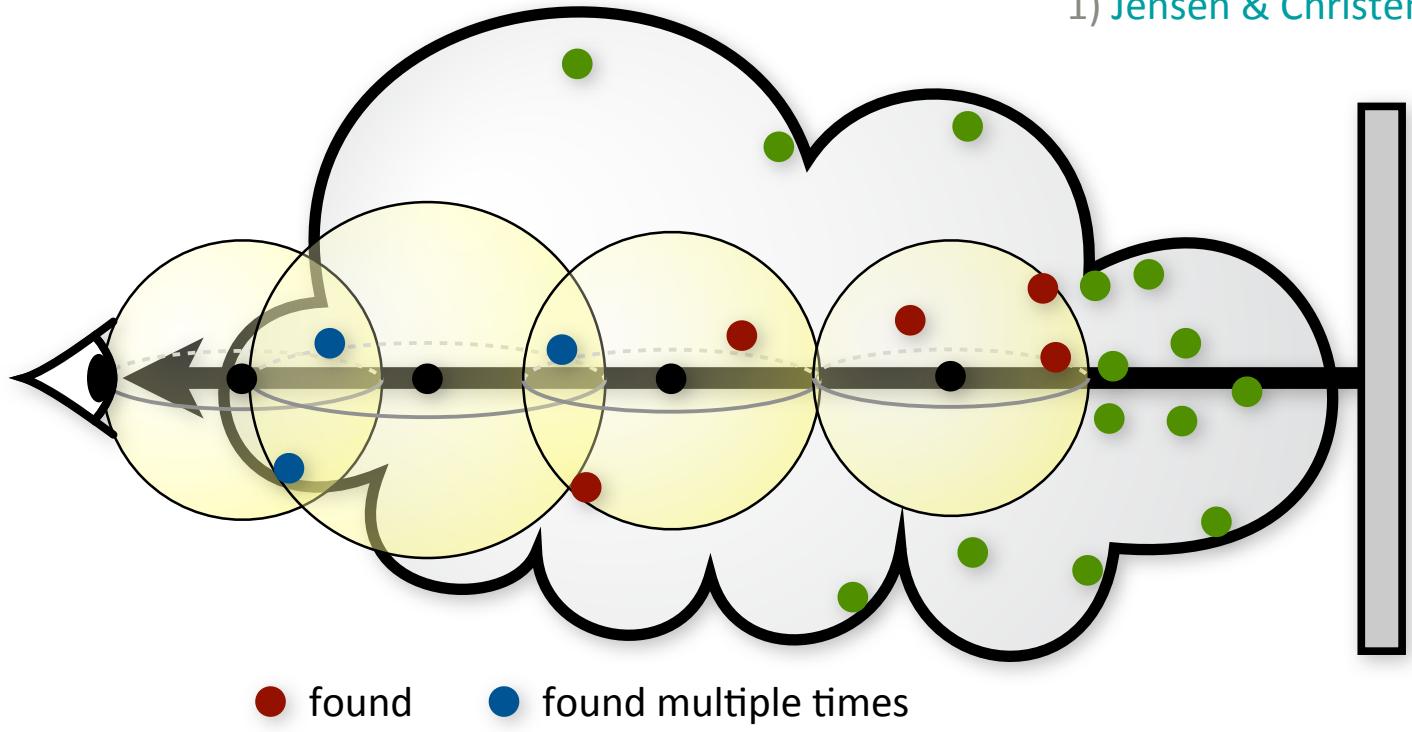


● found ● found multiple times



Volumetric Photon Mapping

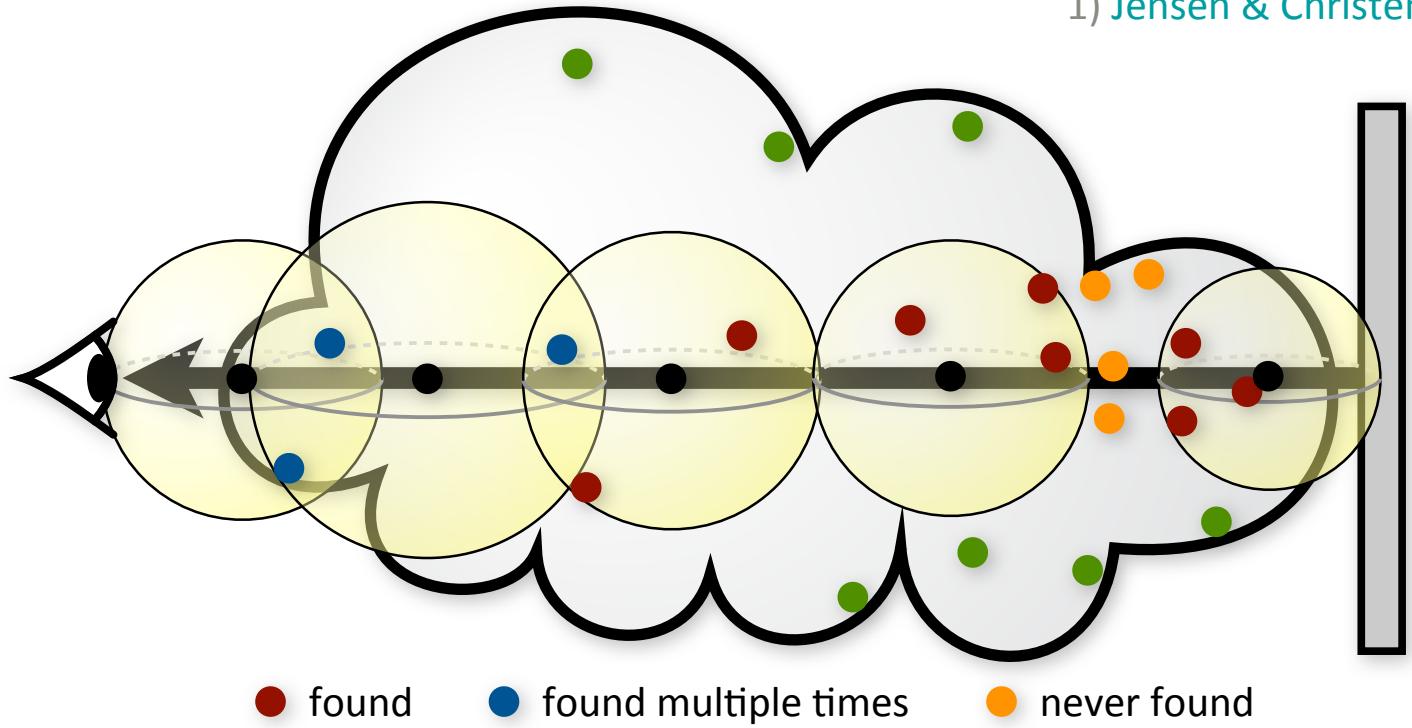
1) Jensen & Christensen 98





Volumetric Photon Mapping

1) Jensen & Christensen 98

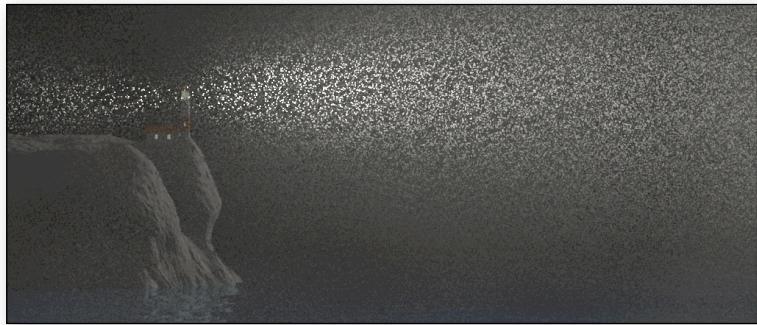




Drawbacks

Large Step-size

1) Jensen & Christensen 98



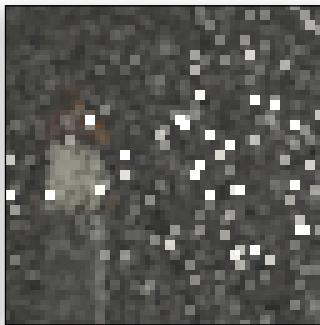


Drawbacks

Large Step-size



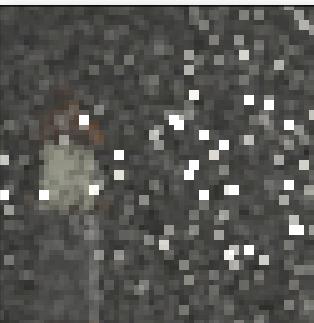
1) Jensen & Christensen 98



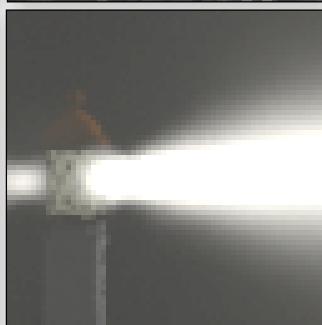
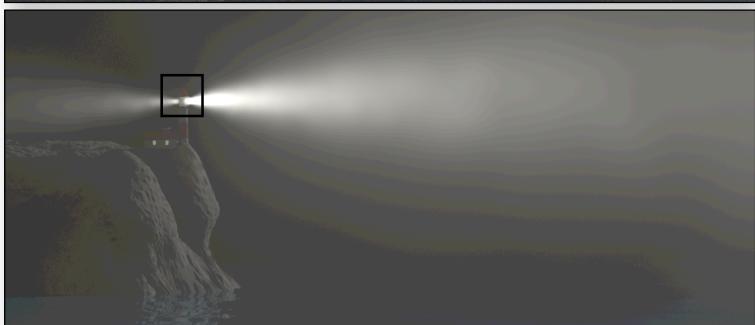


Drawbacks

Large Step-size



1) Jensen & Christensen 98

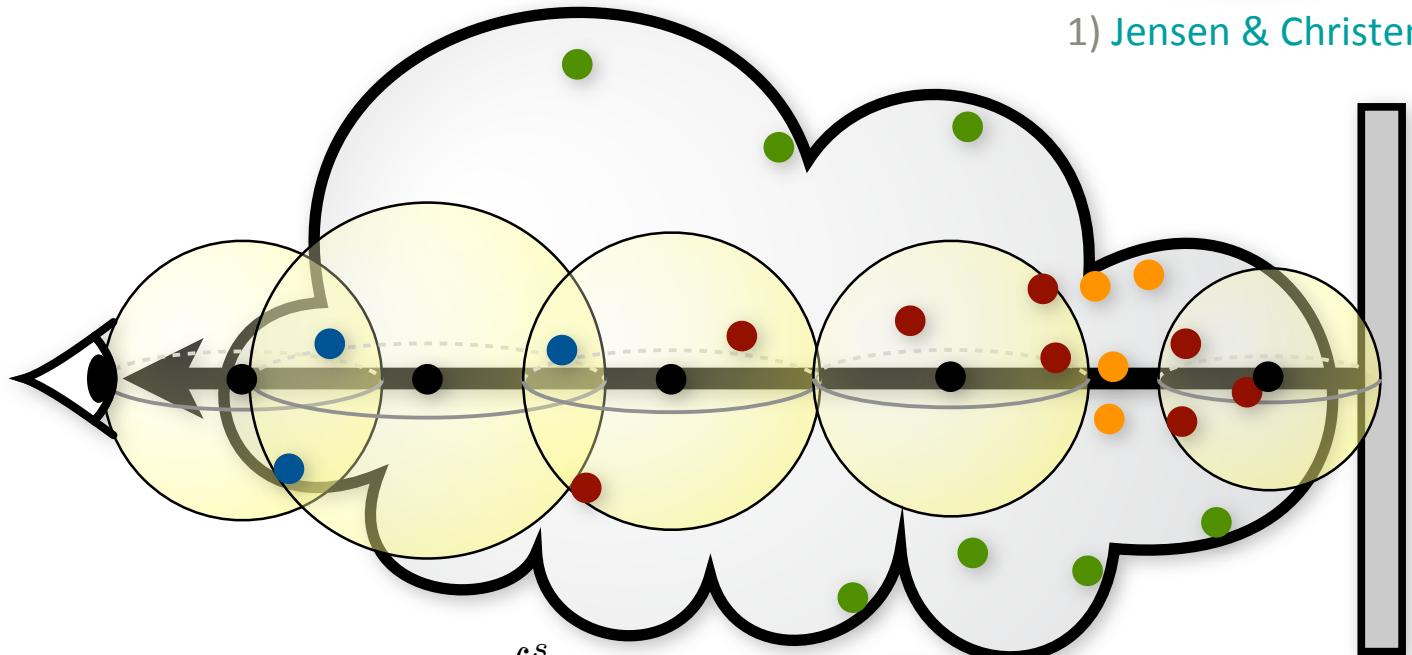


Very Small Step-size



Volumetric Photon Mapping

1) Jensen & Christensen 98

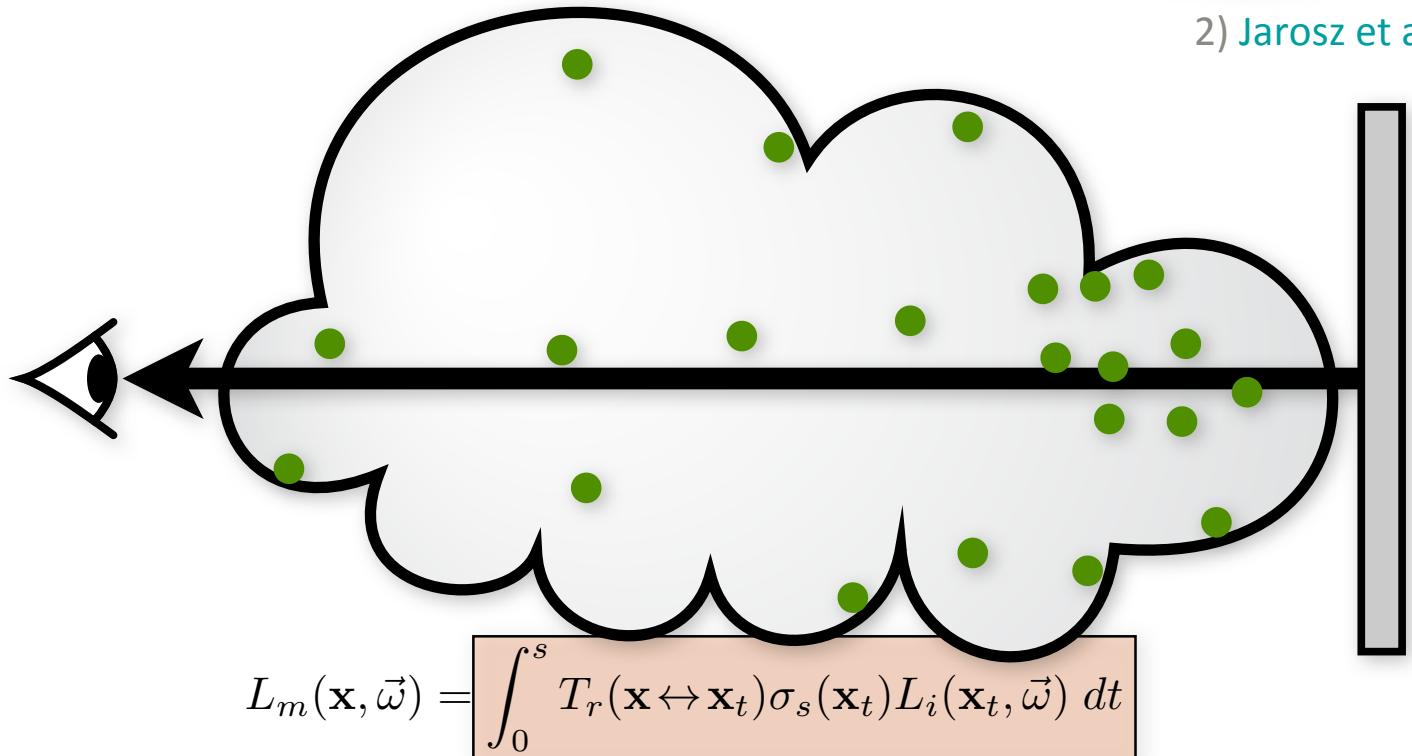


$$L_m(\mathbf{x}, \vec{\omega}) = \int_0^s T_r(\mathbf{x} \leftrightarrow \mathbf{x}_t) \sigma_s(\mathbf{x}_t) L_i(\mathbf{x}_t, \vec{\omega}) dt$$



Beam Radiance Integral

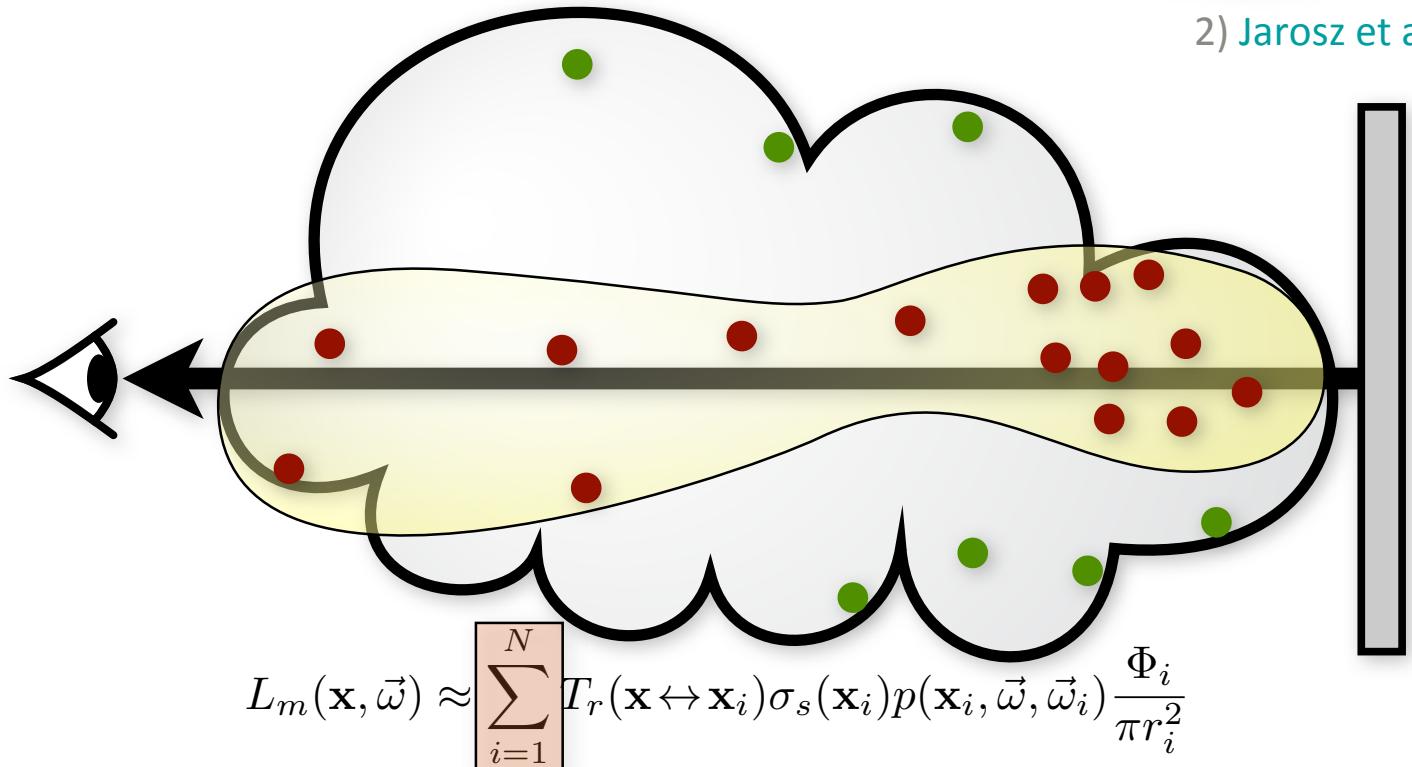
2) Jarosz et al. 2008





Beam Radiance Estimation

2) Jarosz et al. 2008

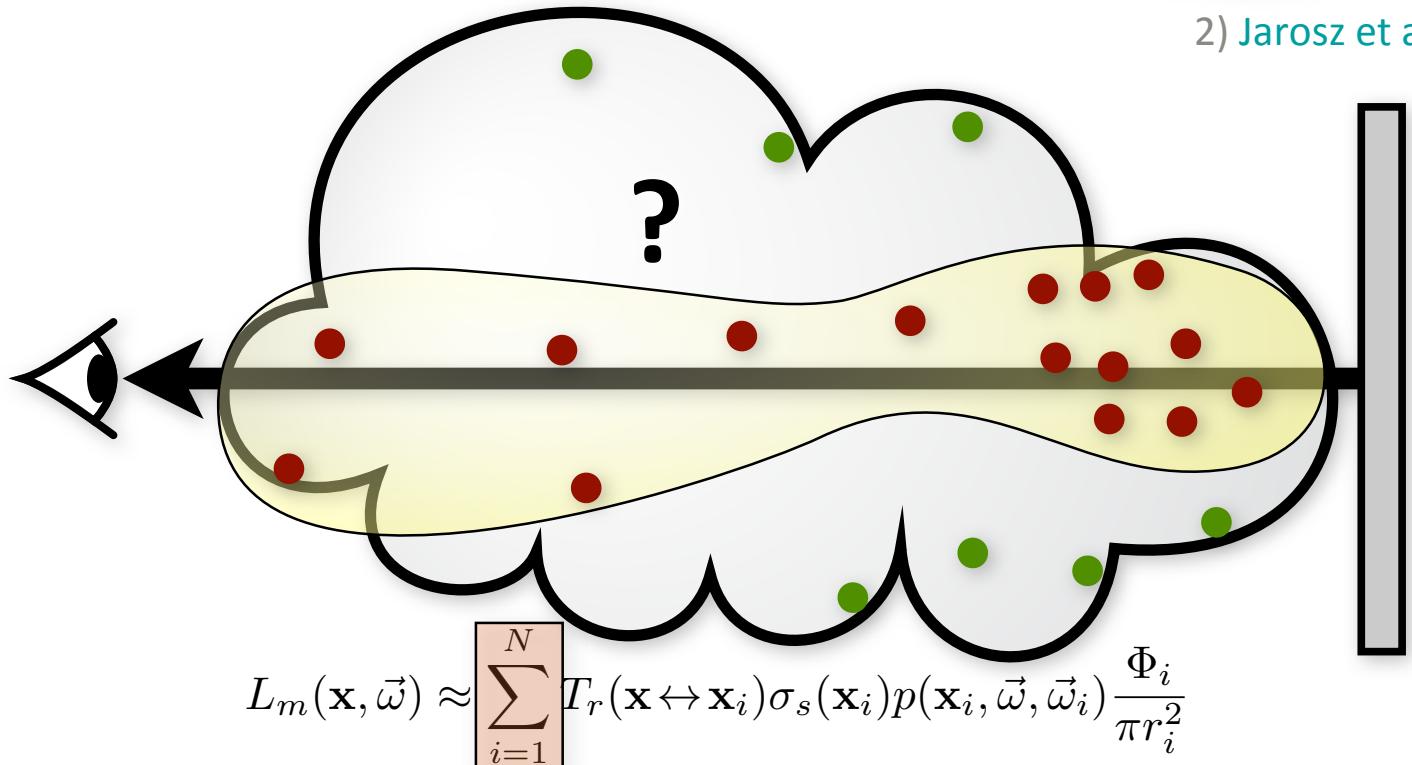


$$L_m(\mathbf{x}, \vec{\omega}) \approx \sum_{i=1}^N T_r(\mathbf{x} \leftrightarrow \mathbf{x}_i) \sigma_s(\mathbf{x}_i) p(\mathbf{x}_i, \vec{\omega}, \vec{\omega}_i) \frac{\Phi_i}{\pi r_i^2}$$



Beam Radiance Estimation

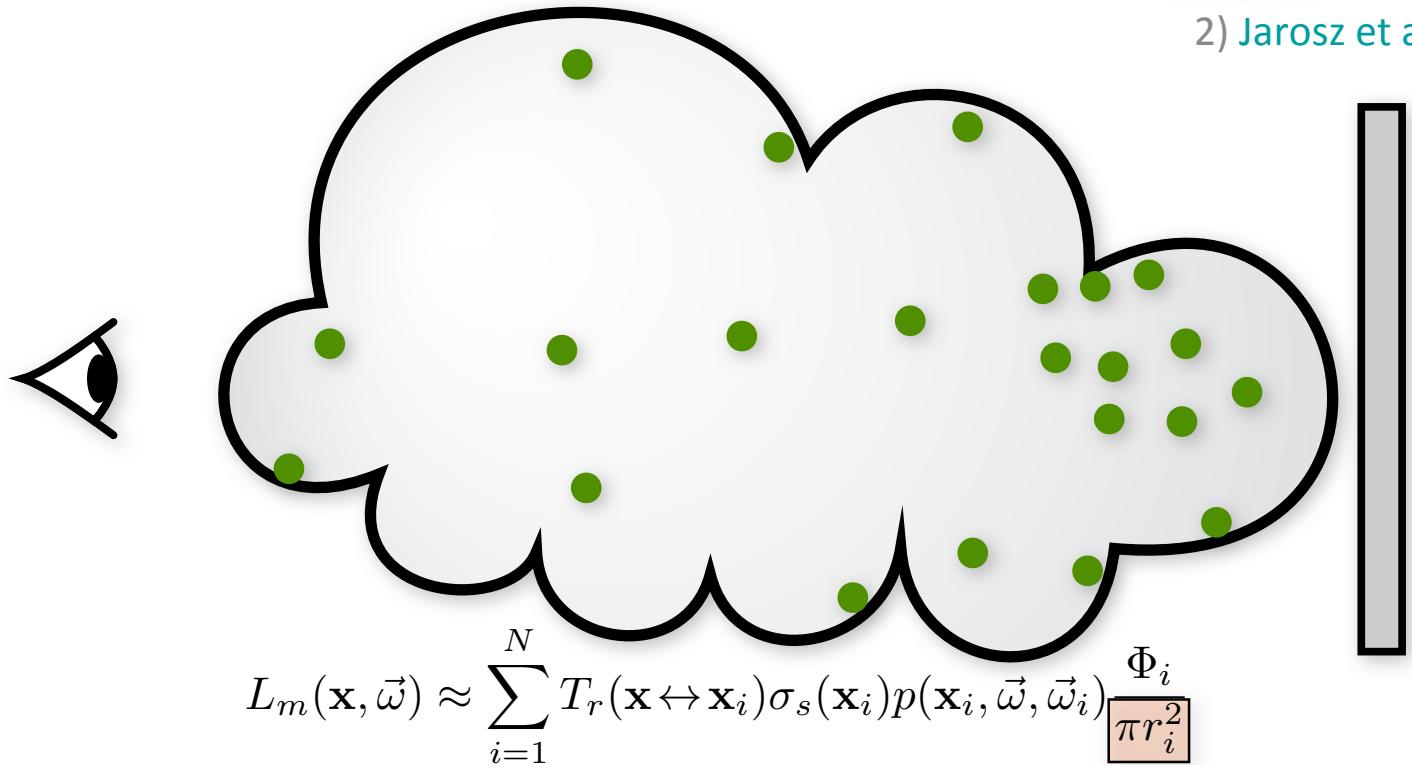
2) Jarosz et al. 2008





Photon Radius Estimation

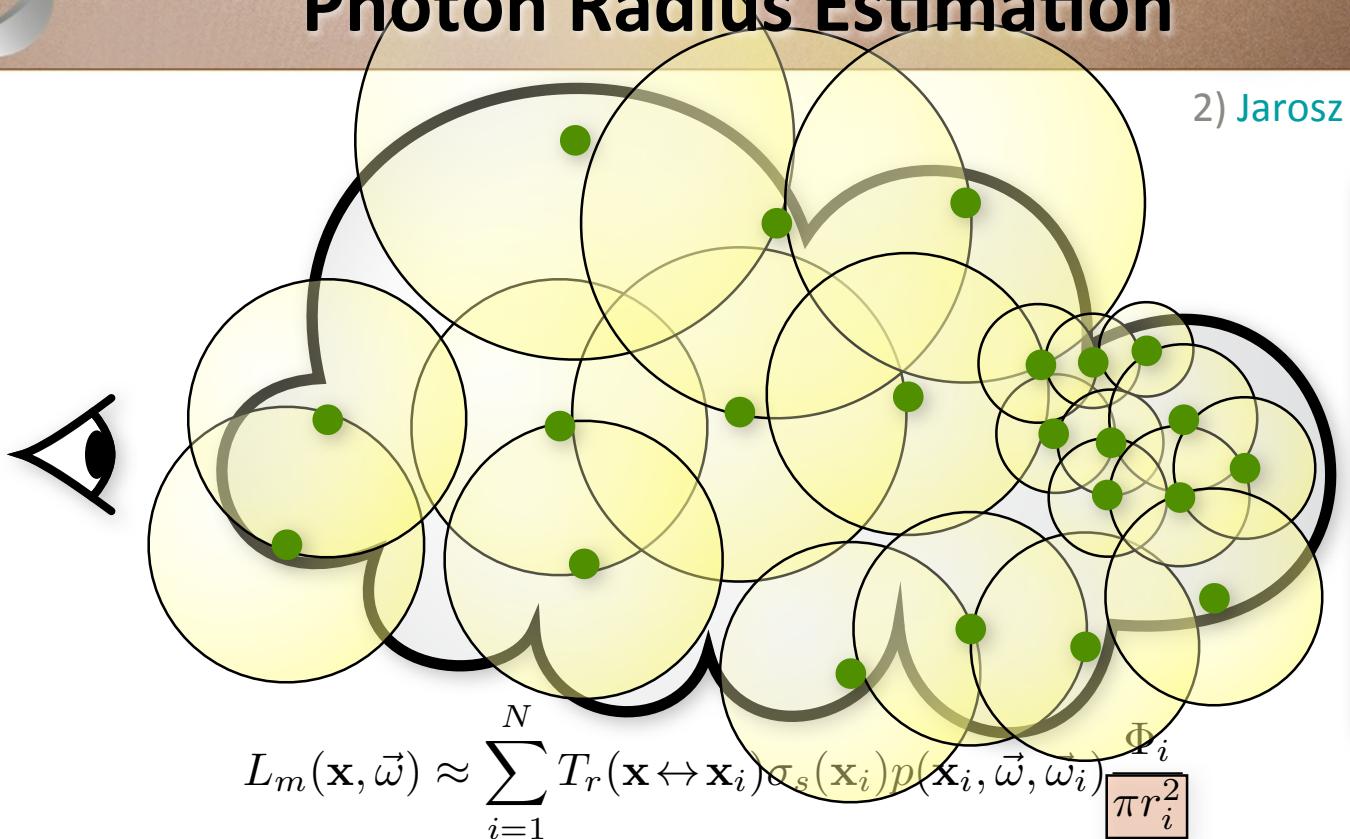
2) Jarosz et al. 2008





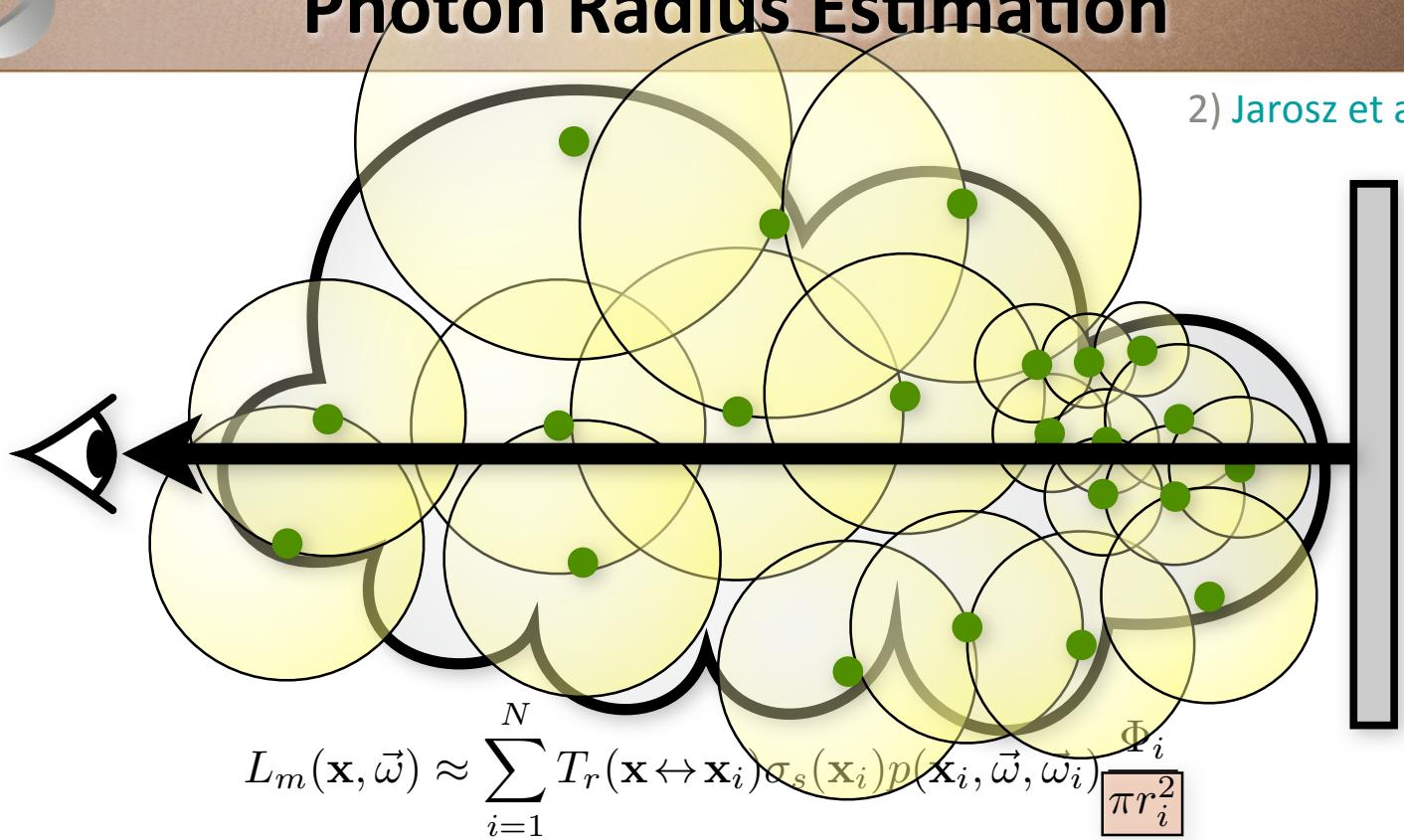
Photon Radius Estimation

2) Jarosz et al. 2008



Photon Radius Estimation

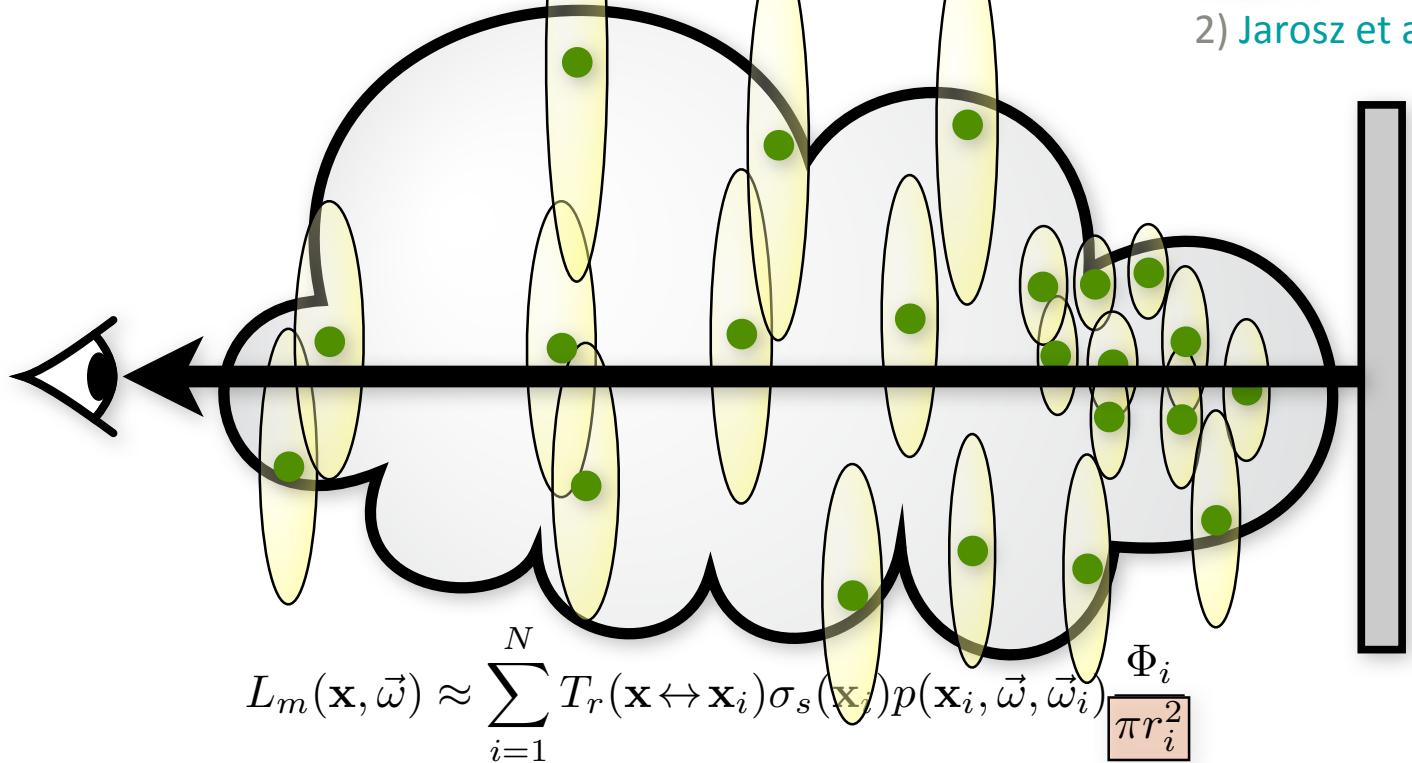
2) Jarosz et al. 2008





Photon Radius Estimation

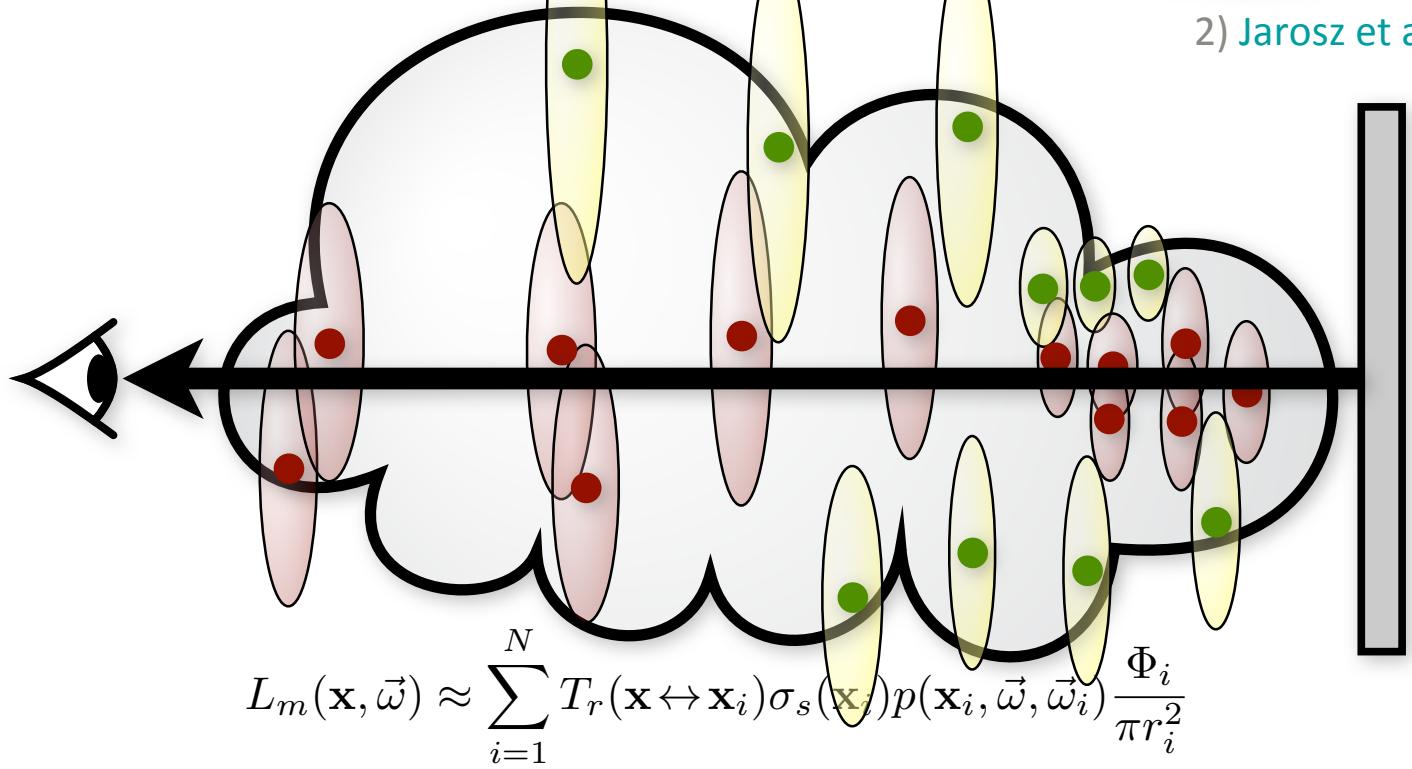
2) Jarosz et al. 2008





Photon Radius Estimation

2) Jarosz et al. 2008

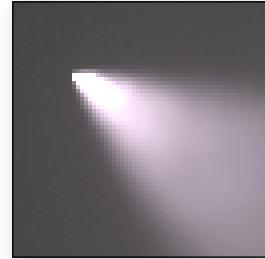
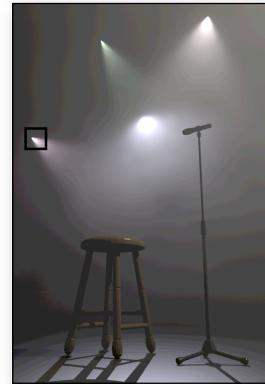


$$L_m(\mathbf{x}, \vec{\omega}) \approx \sum_{i=1}^N T_r(\mathbf{x} \leftrightarrow \mathbf{x}_i) \sigma_s(\mathbf{x}_i) p(\mathbf{x}_i, \vec{\omega}, \vec{\omega}_i) \frac{\Phi_i}{\pi r_i^2}$$



Concert Stage

Beam Estimate

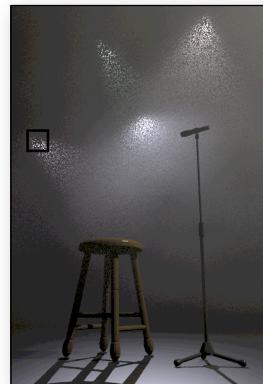


(6:22)



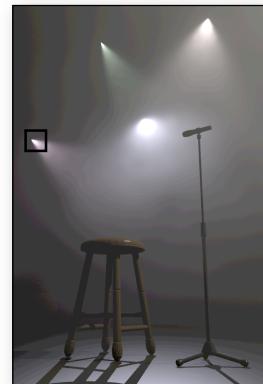
Concert Stage

Trad. Estimate



(6:38)

Beam Estimate

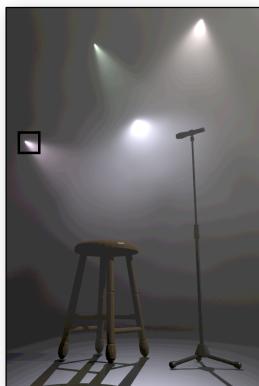


(6:22)



Concert Stage

Trad. Estimate



(∞)

Trad. Estimate



(6:38)

Beam Estimate



(6:22)

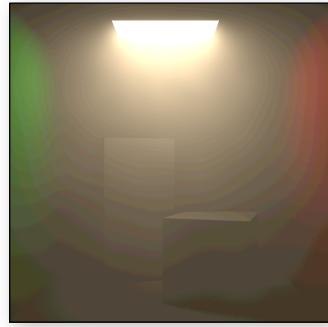


Smoky Cornell Box

Trad. Estimate



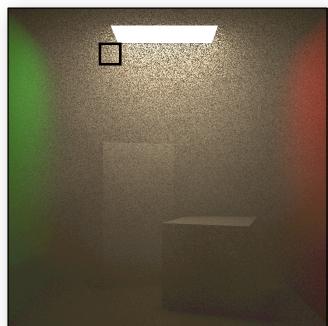
Beam Estimate





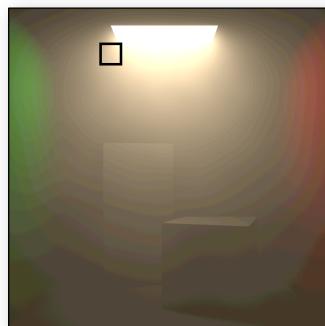
Smoky Cornell Box

Trad. Estimate



(4:03)

Beam Estimate



(3:35)



Cars on Foggy Street

Beam Estimate



Traditional Estimate

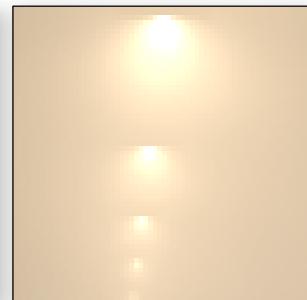


Cars on Foggy Street

Beam Estimate



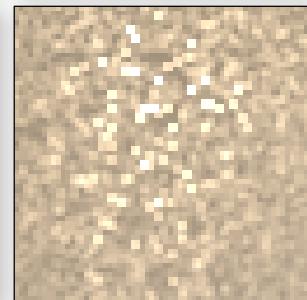
(1:53)



Traditional Estimate



(2:02)



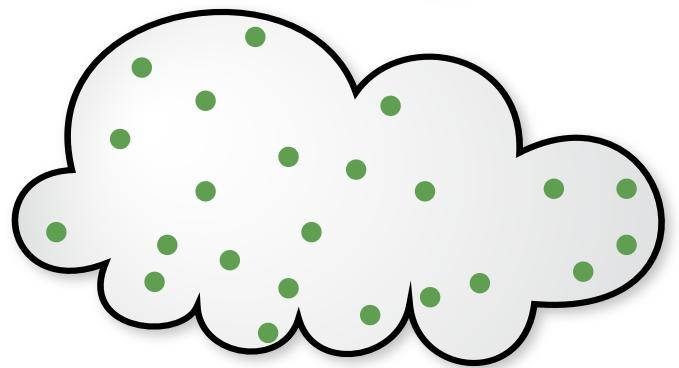
Questions?

From Photons to Beams



So Far...

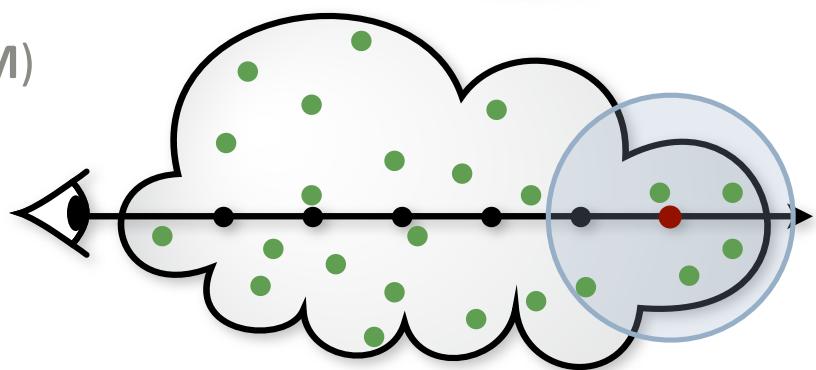
- Volumetric Photon Mapping (**VPM**)
[Jensen & Christensen 98]
- The Beam Radiance Estimate (**BRE**)
[Jarosz et al. 08]





So Far...

- Volumetric Photon Mapping (**VPM**)
[Jensen & Christensen 98]

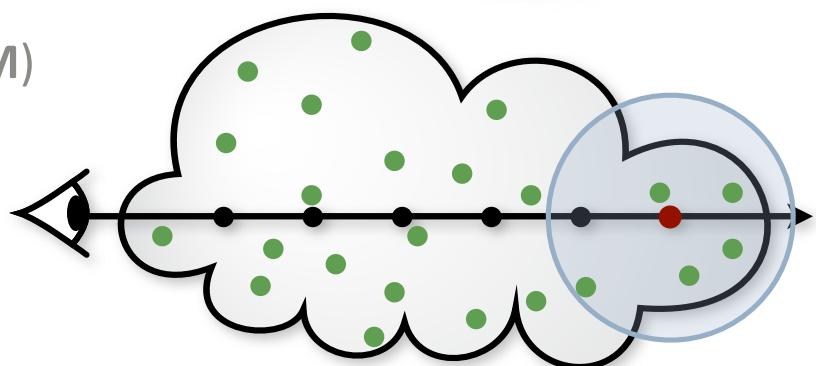


- The Beam Radiance Estimate (**BRE**)
[Jarosz et al. 08]

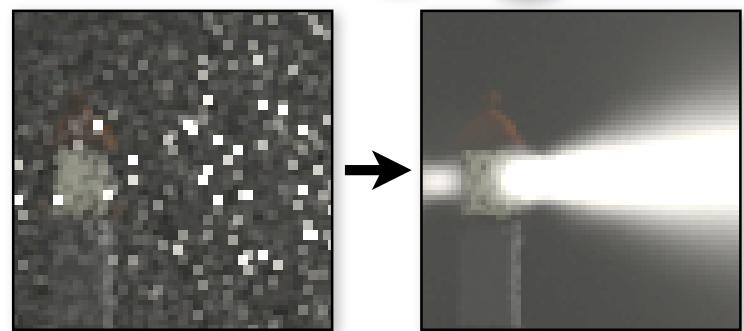


So Far...

- Volumetric Photon Mapping (**VPM**)
[Jensen & Christensen 98]



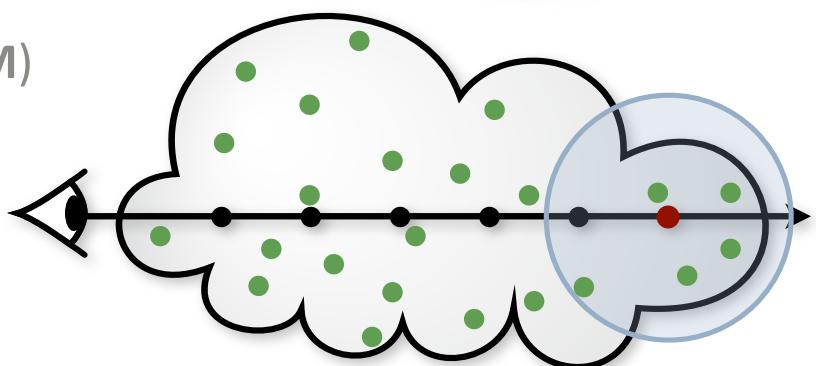
- The Beam Radiance Estimate (**BRE**)
[Jarosz et al. 08]



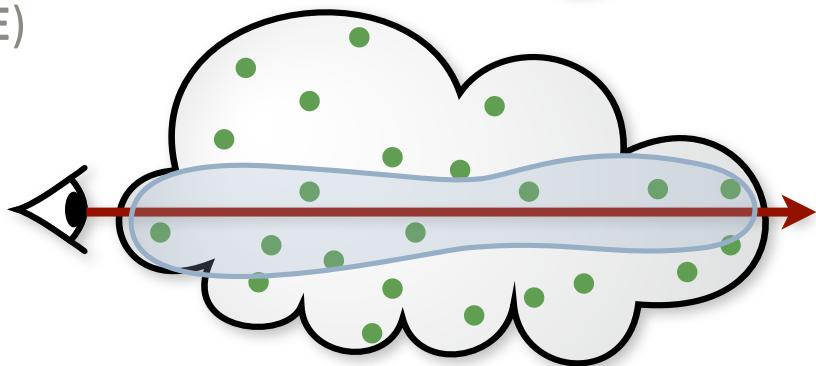


So Far...

- Volumetric Photon Mapping (**VPM**)
[Jensen & Christensen 98]



- The Beam Radiance Estimate (**BRE**)
[Jarosz et al. 08]

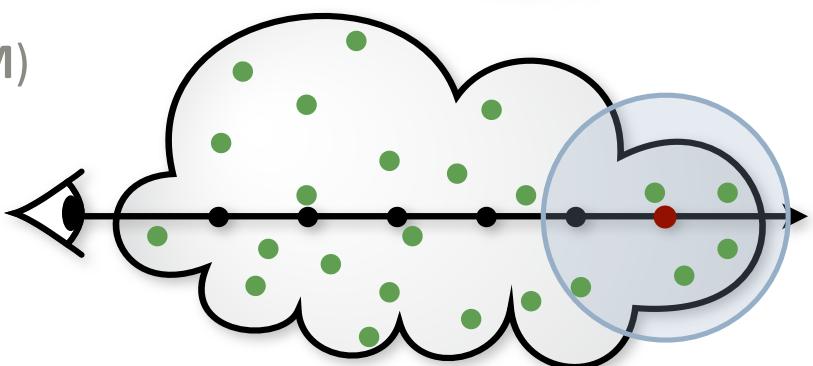




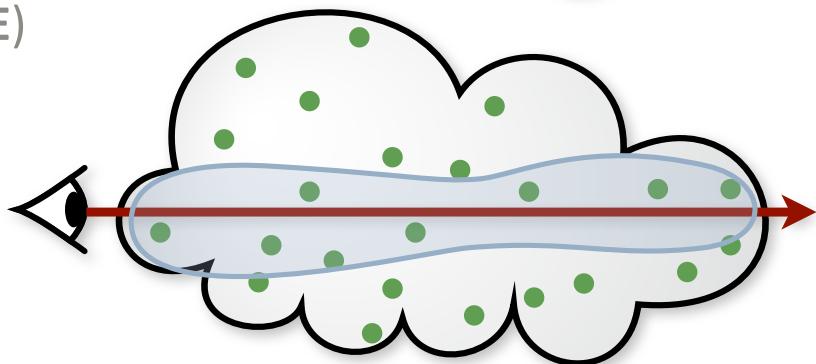
So Far...

- Volumetric Photon Mapping (**VPM**)
[Jensen & Christensen 98]

Query
Point



- The Beam Radiance Estimate (**BRE**)
[Jarosz et al. 08]

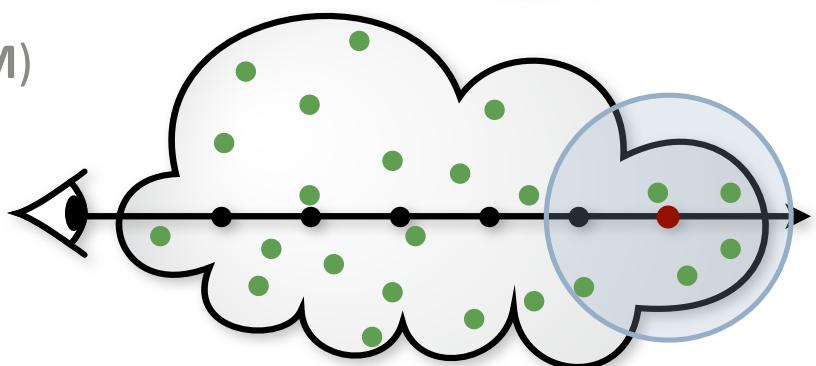




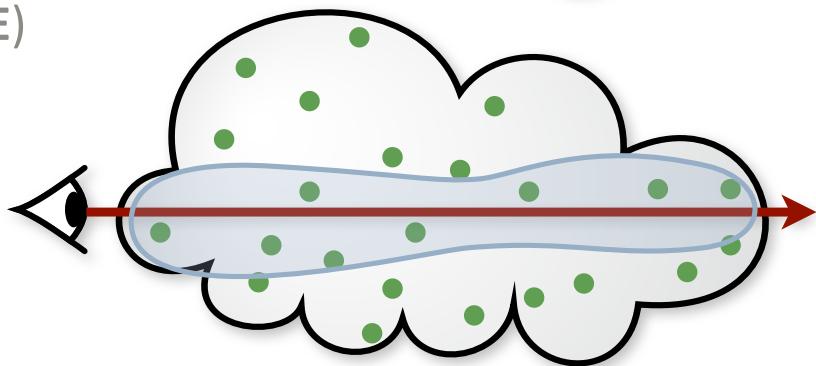
So Far...

- Volumetric Photon Mapping (**VPM**)
[Jensen & Christensen 98]

Query	x	Data
Point	x	Point



- The Beam Radiance Estimate (**BRE**)
[Jarosz et al. 08]

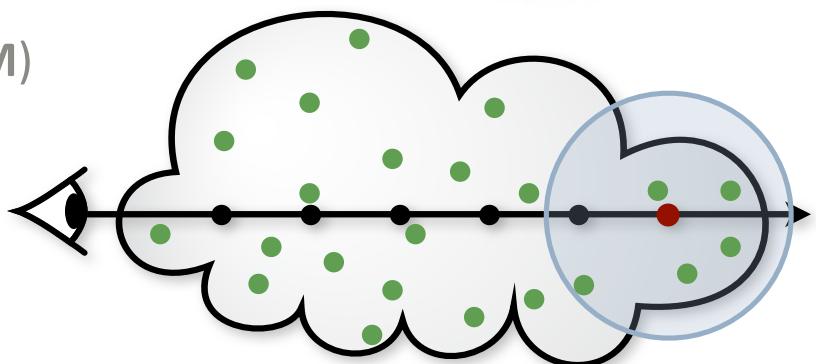




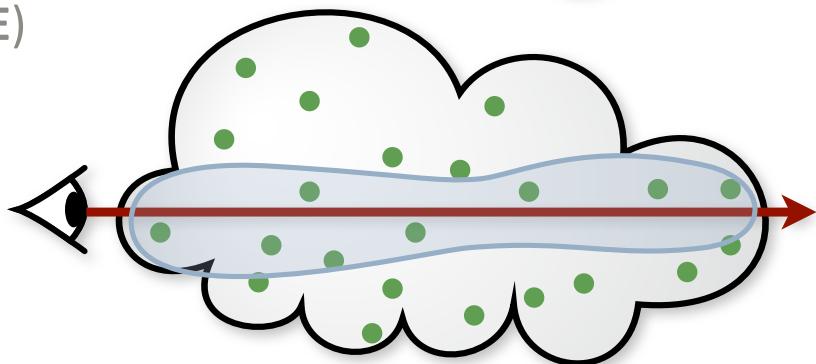
So Far...

- Volumetric Photon Mapping (**VPM**)
[Jensen & Christensen 98]

Query	x	Data	Blur
Point	x	Point	(3D)



- The Beam Radiance Estimate (**BRE**)
[Jarosz et al. 08]

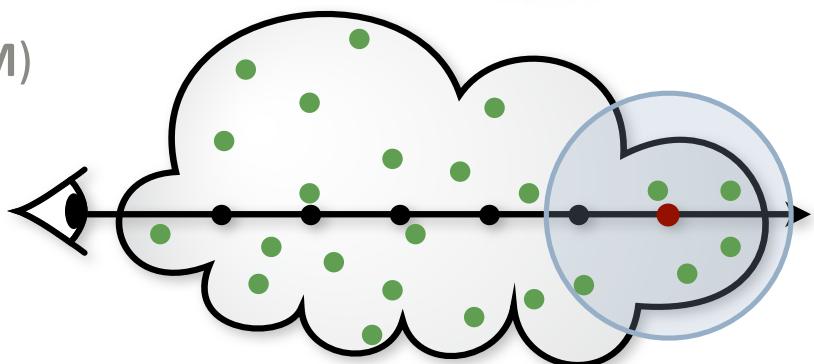




So Far...

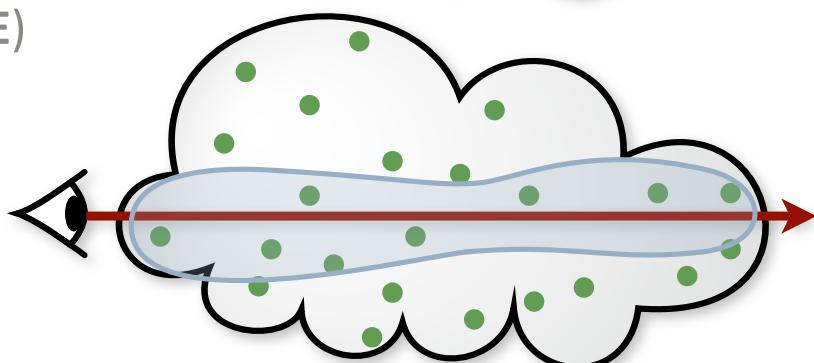
- Volumetric Photon Mapping (**VPM**)
[Jensen & Christensen 98]

Query	x	Data	Blur
Point	x	Point	(3D)



- The Beam Radiance Estimate (**BRE**)
[Jarosz et al. 08]

Query
Beam

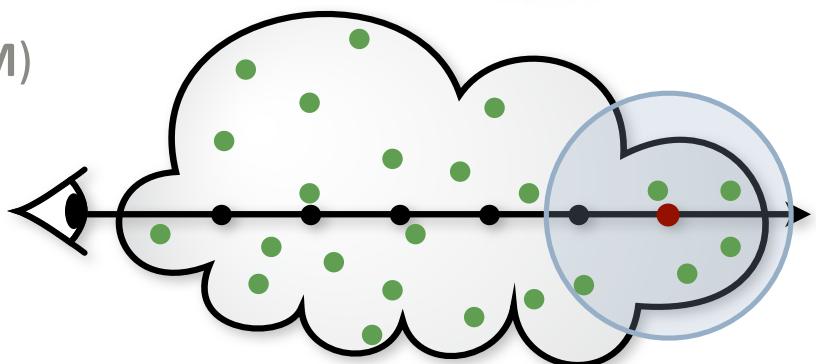




So Far...

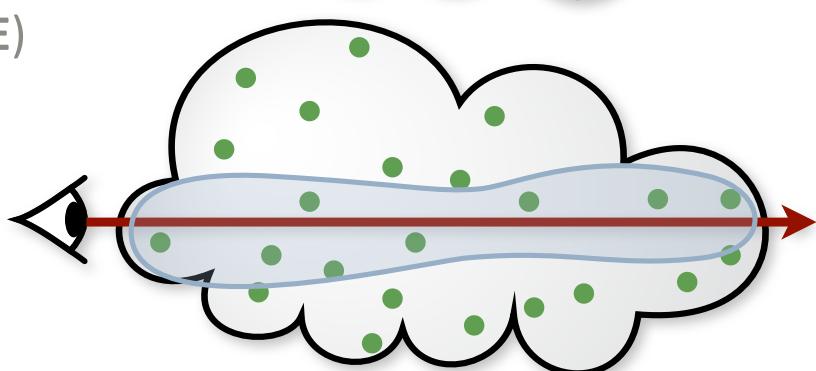
- Volumetric Photon Mapping (**VPM**)
[Jensen & Christensen 98]

Query	x	Data	Blur
Point	x	Point	(3D)



- The Beam Radiance Estimate (**BRE**)
[Jarosz et al. 08]

Query	x	Data
Beam	x	Point

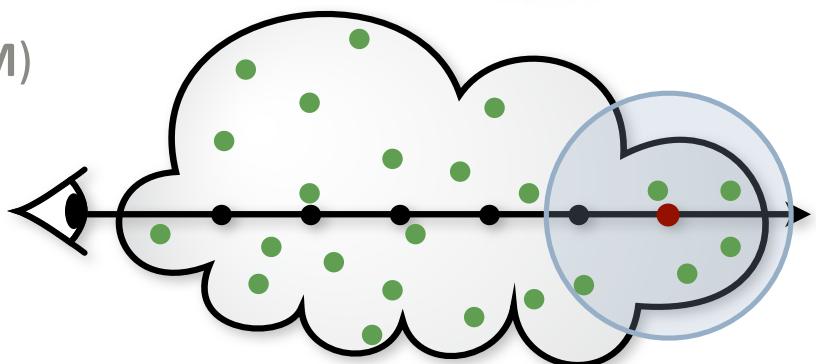




So Far...

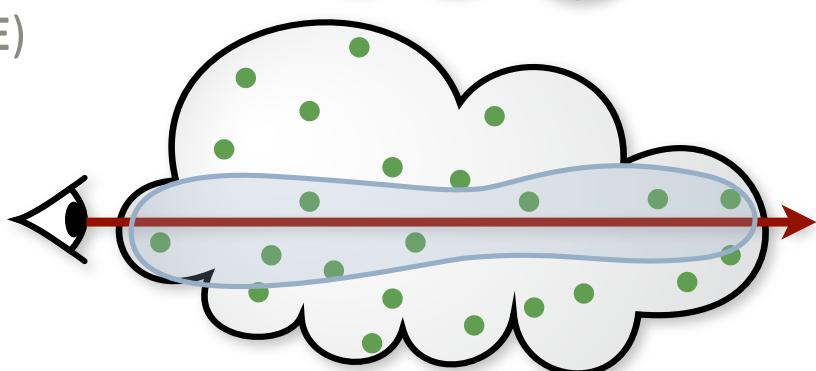
- Volumetric Photon Mapping (**VPM**)
[Jensen & Christensen 98]

Query	x	Data	Blur
Point	x	Point	(3D)



- The Beam Radiance Estimate (**BRE**)
[Jarosz et al. 08]

Query	x	Data	Blur
Beam	x	Point	(2D)





So Far...

- Volumetric Photon Mapping (**VPM**) ■ Beyond Photon Points:
[Jensen & Christensen 98]

Query	x	Data	Blur
Point	x	Point	(3D)

- The Beam Radiance Estimate (**BRE**)
[Jarosz et al. 08]

Query	x	Data	Blur
Beam	x	Point	(2D)



So Far...

- Volumetric Photon Mapping (**VPM**)
[Jensen & Christensen 98]
- Beyond Photon Points:

Query	x	Data	Blur
Point	x	Point	(3D)

Query
Point/Beam

- The Beam Radiance Estimate (**BRE**)
[Jarosz et al. 08]

Query	x	Data	Blur
Beam	x	Point	(2D)



So Far...

- Volumetric Photon Mapping (**VPM**)
[Jensen & Christensen 98]
- Beyond Photon Points:

Query	x	Data	Blur
Point	x	Point	(3D)

Query	Data
Point/Beam	Point/Beam

- The Beam Radiance Estimate (**BRE**)
[Jarosz et al. 08]

Query	x	Data	Blur
Beam	x	Point	(2D)



So Far...

- Volumetric Photon Mapping (**VPM**)
[Jensen & Christensen 98]
- Beyond Photon Points:

Query	x	Data	Blur
Point	x	Point	(3D)

Query	Data	Blur
Point/Beam	Point/Beam	1D/2D/3D

- The Beam Radiance Estimate (**BRE**)
[Jarosz et al. 08]

Query	x	Data	Blur
Beam	x	Point	(2D)



Density Estimator Options

Query	x	Data	Blur
Point	x	Point	(3D)
Beam	x	Point	(2D)

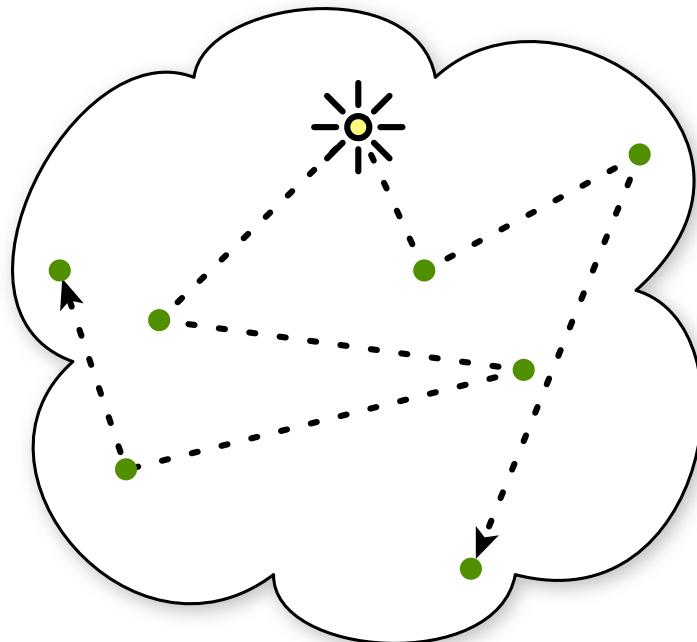


Density Estimator Options

Query	x	Data	Blur
Point	x	Point	(3D)
Beam	x	Point	(2D)
Beam	x	Point	(3D)
Point	x	Beam	(3D)
Point	x	Beam	(2D)
Beam	x	Beam	(3D)
Beam	x	Beam	(2D) ₁
Beam	x	Beam	(2D) ₂
Beam	x	Beam	(1D)



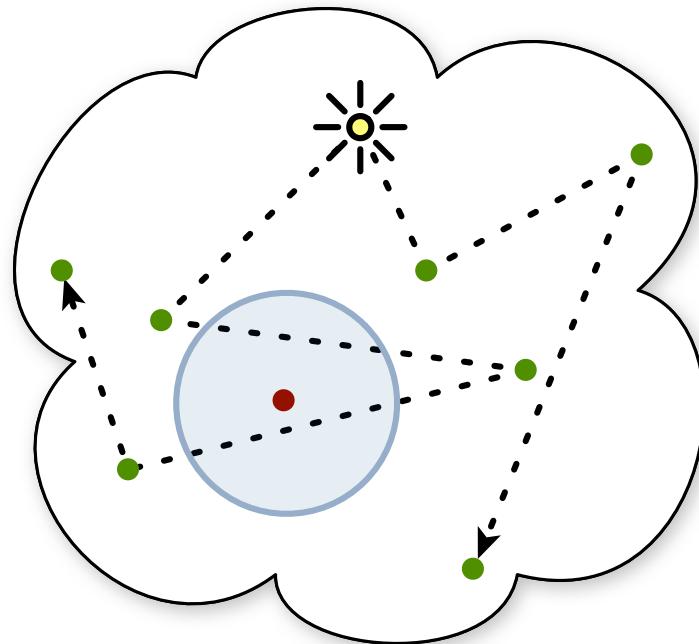
Volumetric Photon Mapping



Photon Points



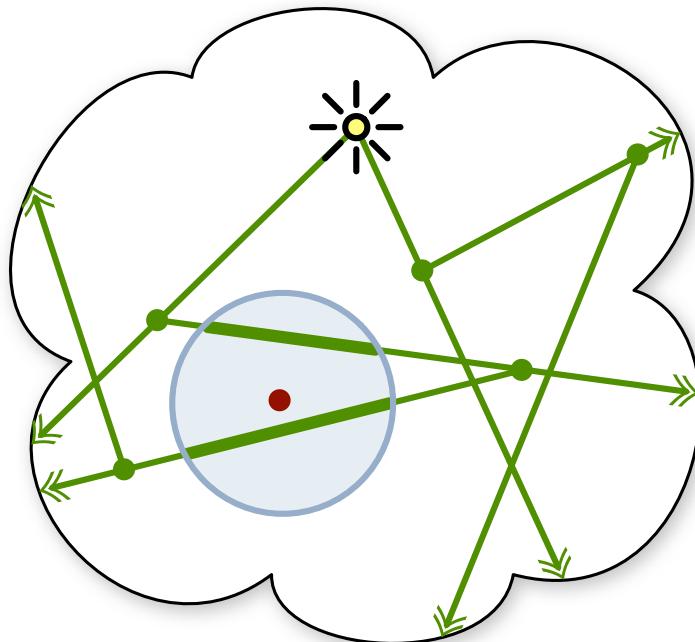
Volumetric Photon Mapping



Photon Points

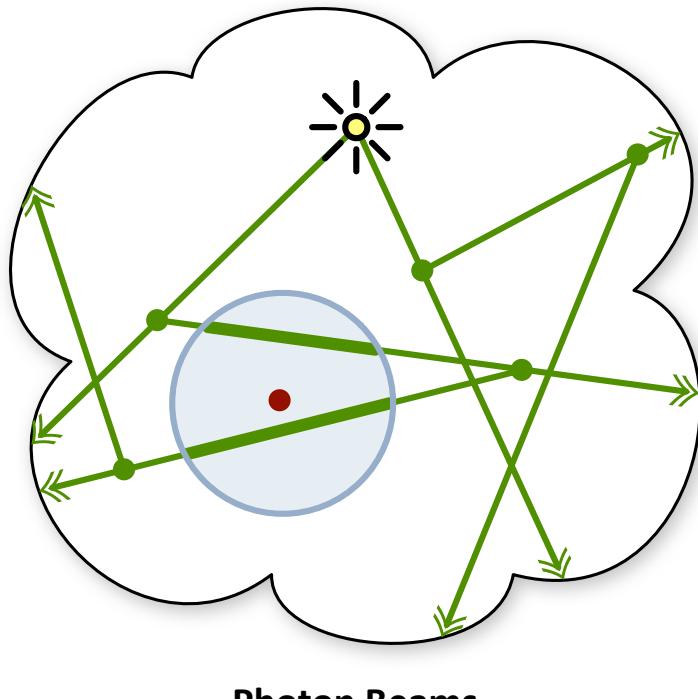


Volumetric Photon Mapping





Volumetric Photon Mapping

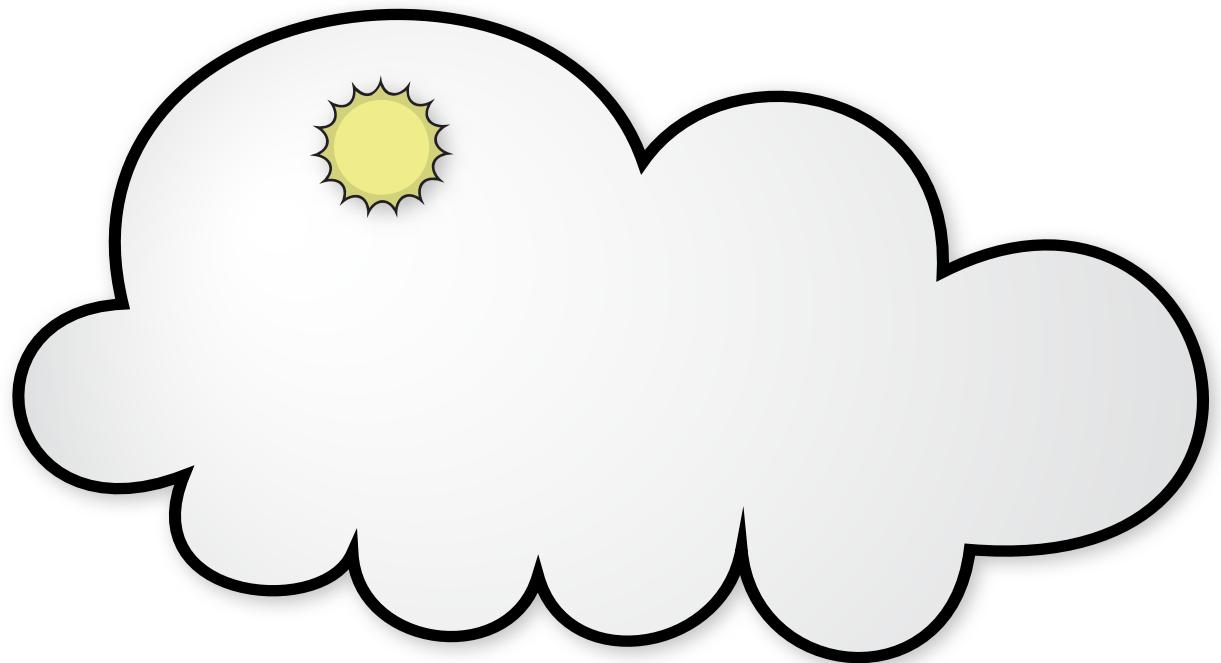




Photon Beams



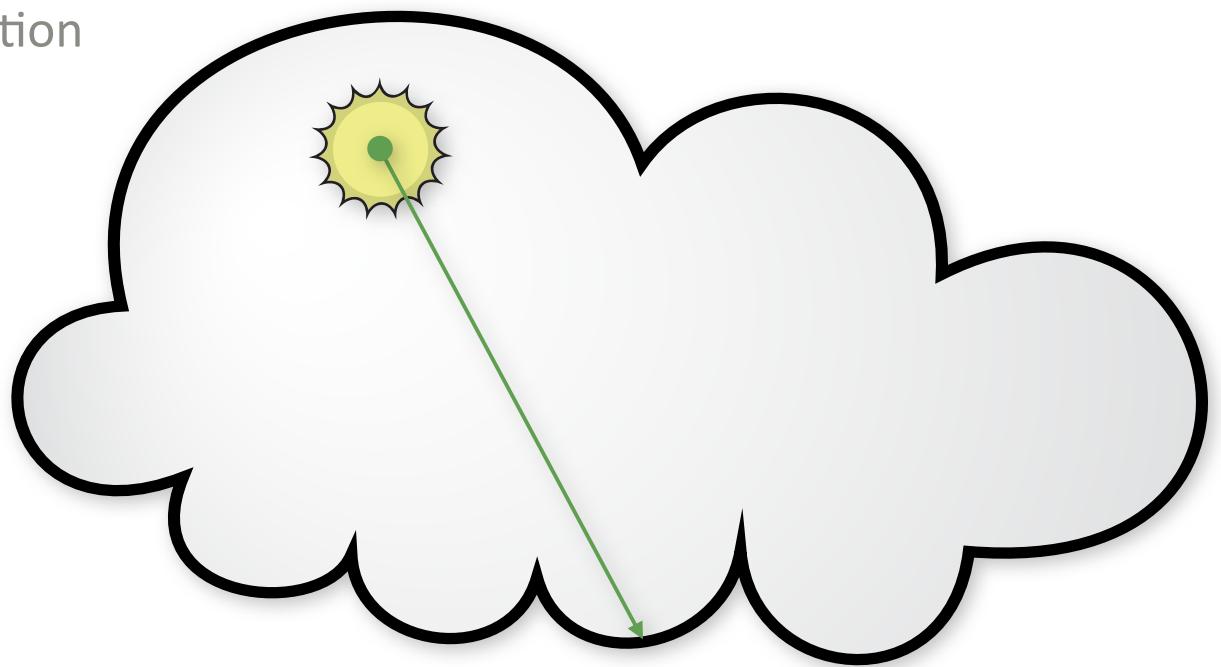
Traditional Photon Tracing





Traditional Photon Tracing

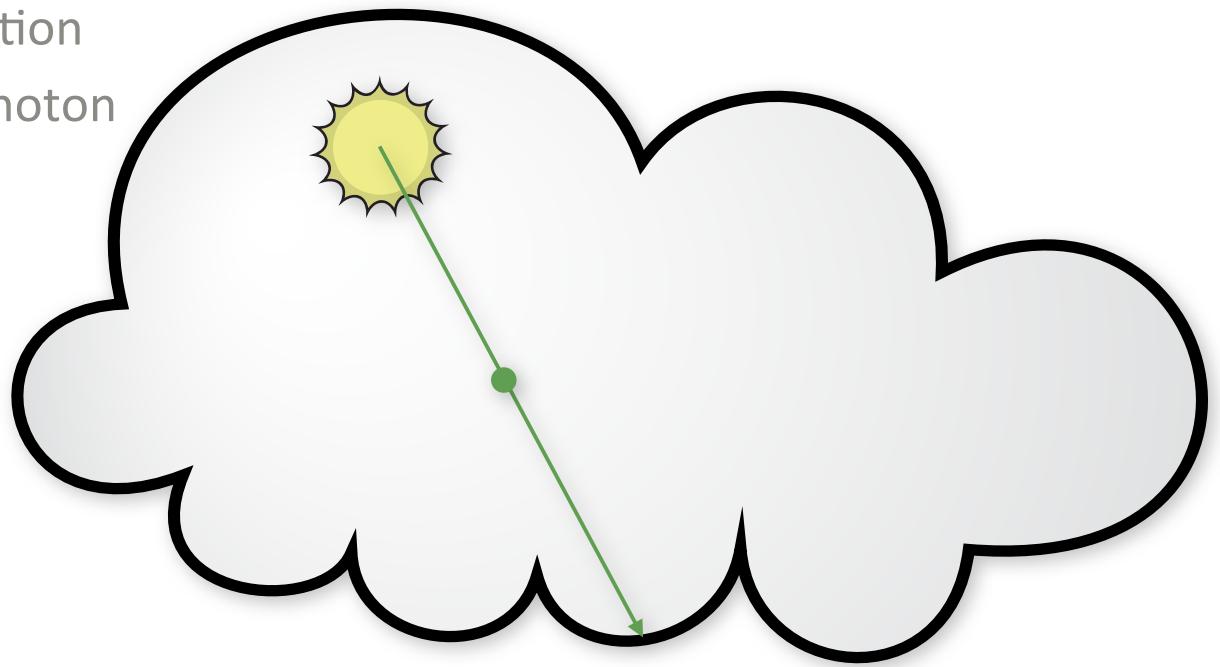
1) choose direction





Traditional Photon Tracing

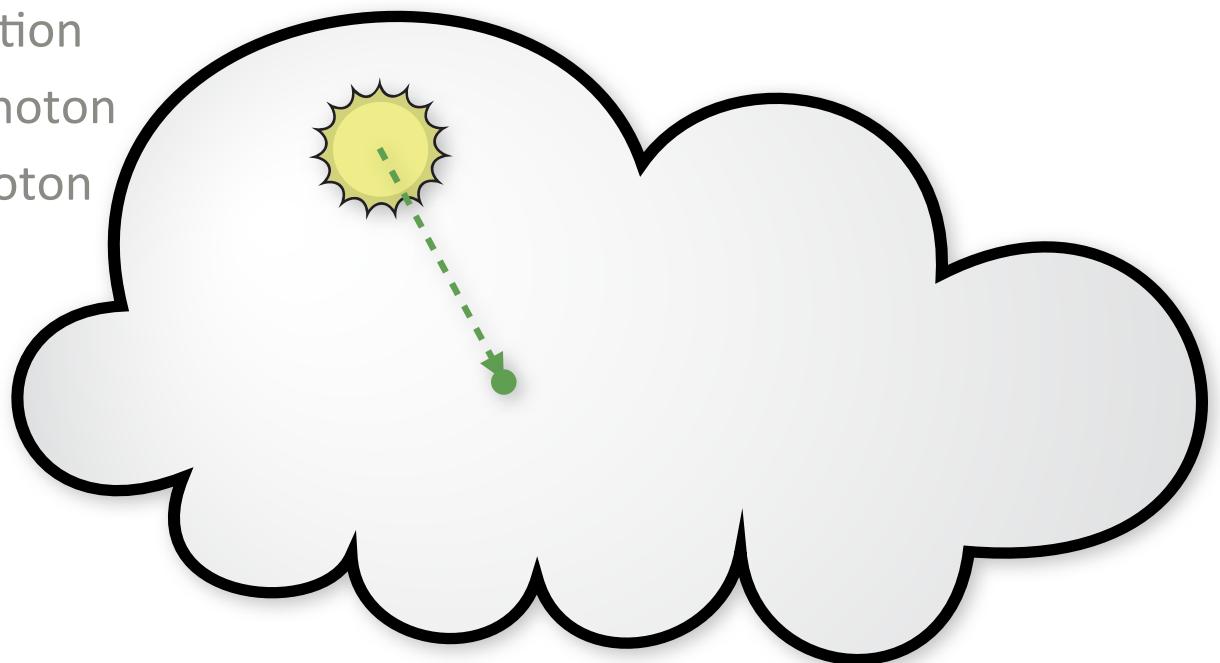
- 1) choose direction
- 2) propagate photon





Traditional Photon Tracing

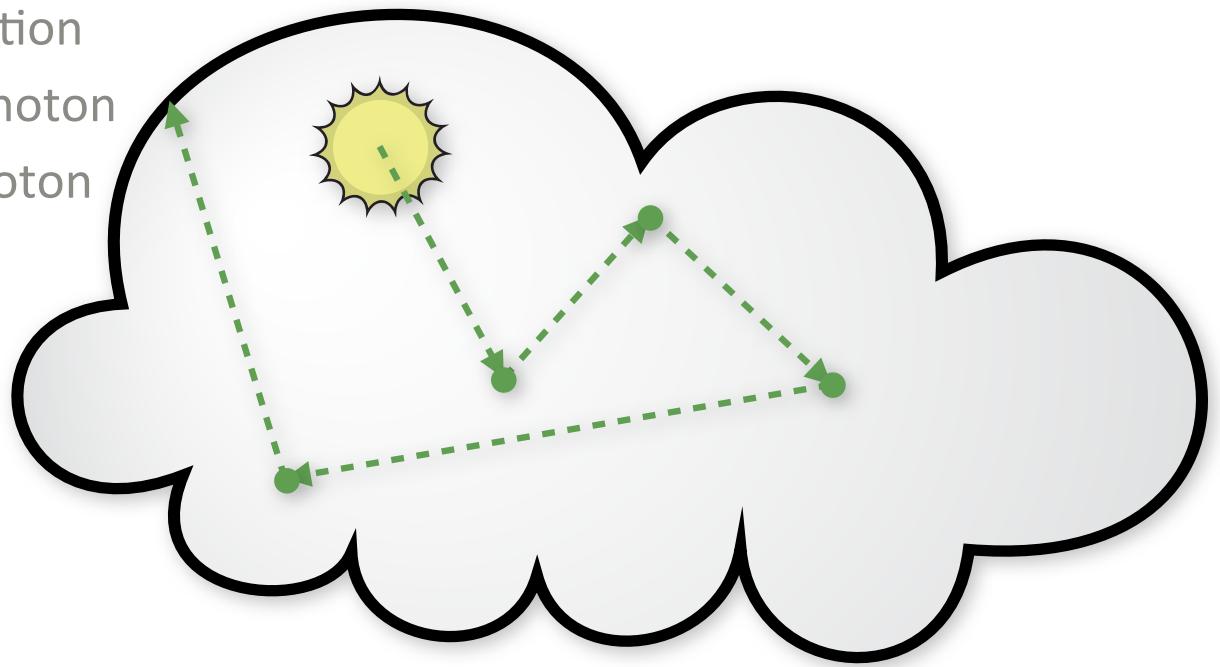
- 1) choose direction
- 2) propagate photon
- 3) deposit a photon





Traditional Photon Tracing

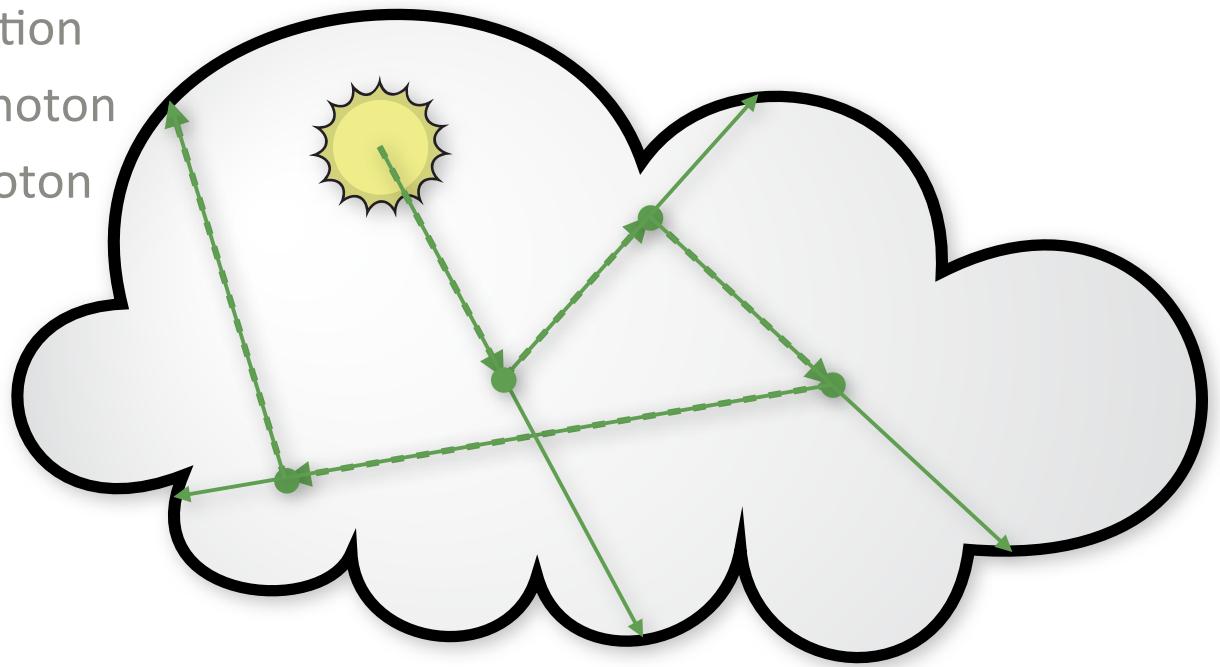
- 1) choose direction
- 2) propagate photon
- 3) deposit a photon
- 4) repeat





Traditional Photon Tracing

- 1) choose direction
- 2) propagate photon
- 3) deposit a photon
- 4) repeat

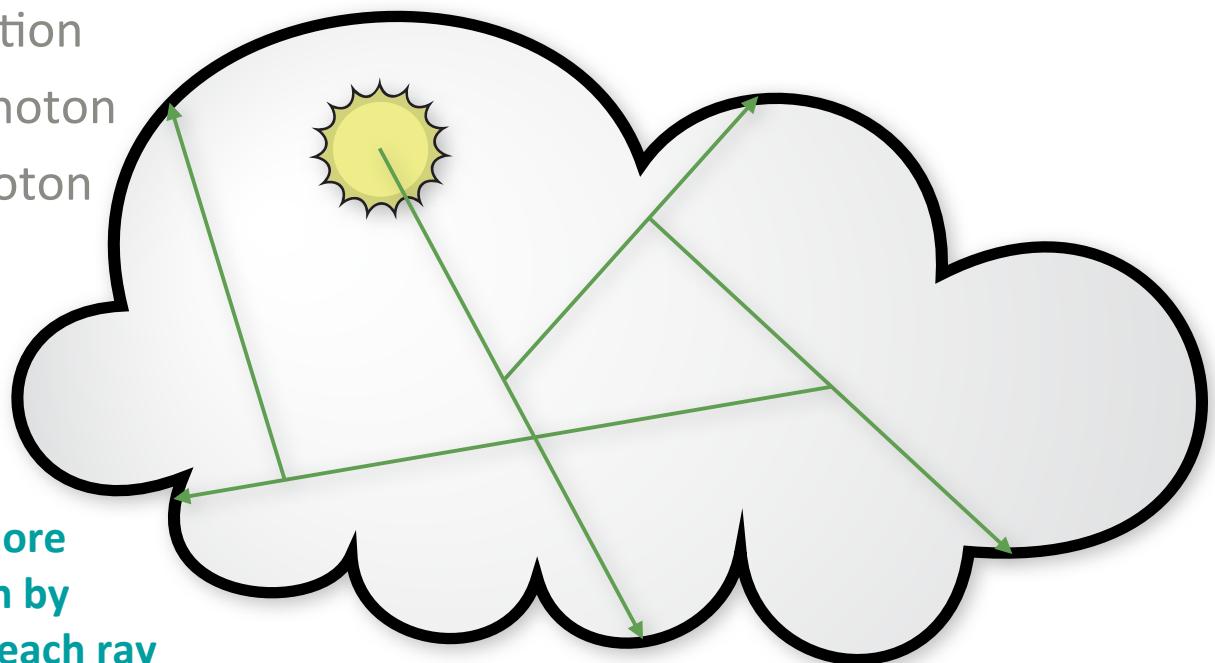




“Photon Marching”

- 1) choose direction
- 2) propagate photon
- 3) deposit a photon
- 4) repeat

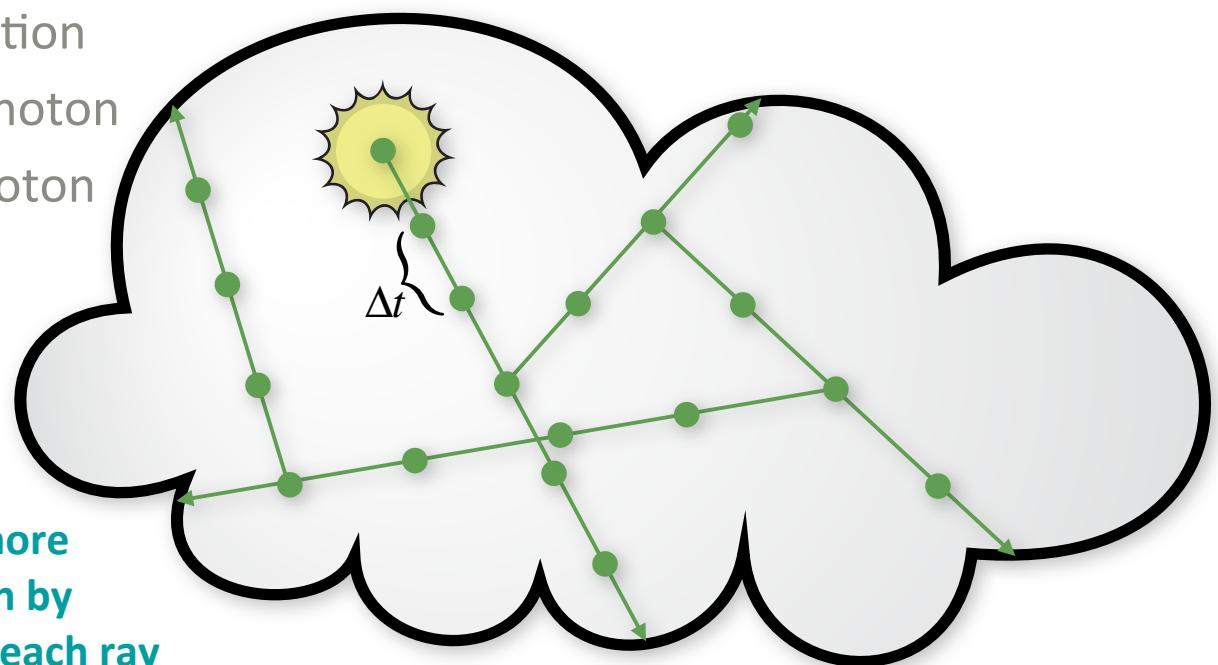
Could deposit more than one photon by marching along each ray





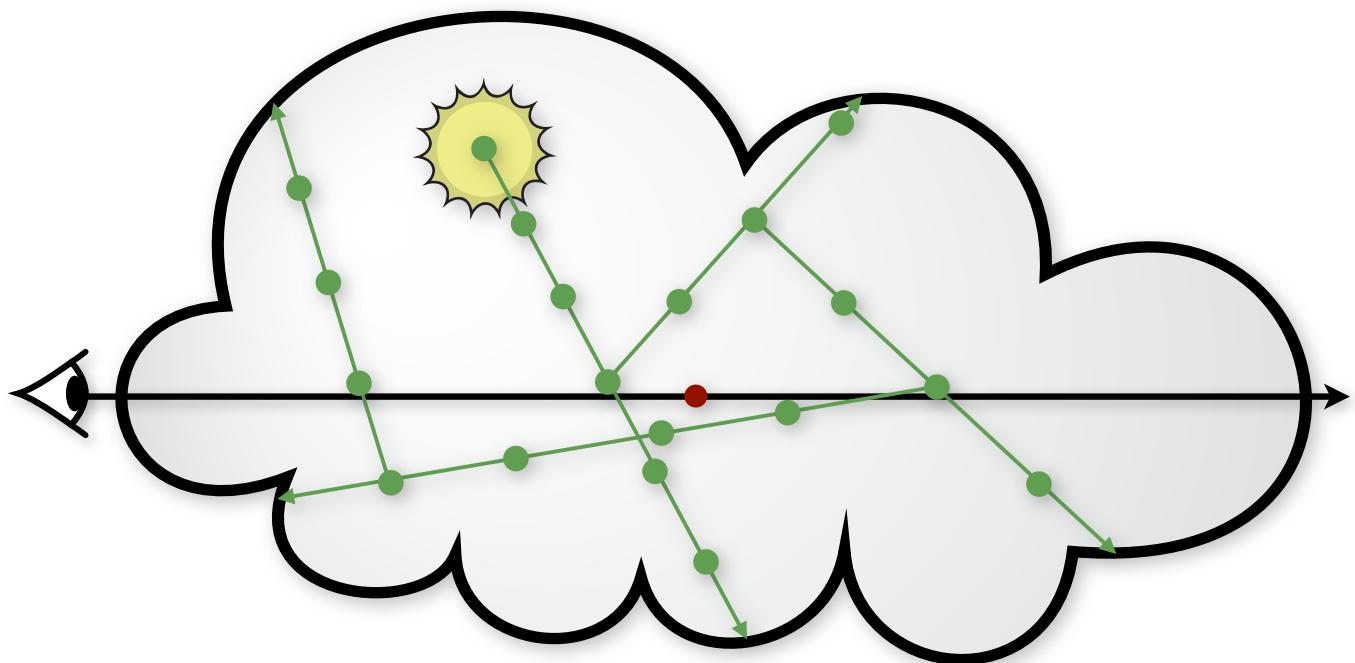
“Photon Marching”

- 1) choose direction
- 2) propagate photon
- 3) deposit a photon
- 4) repeat





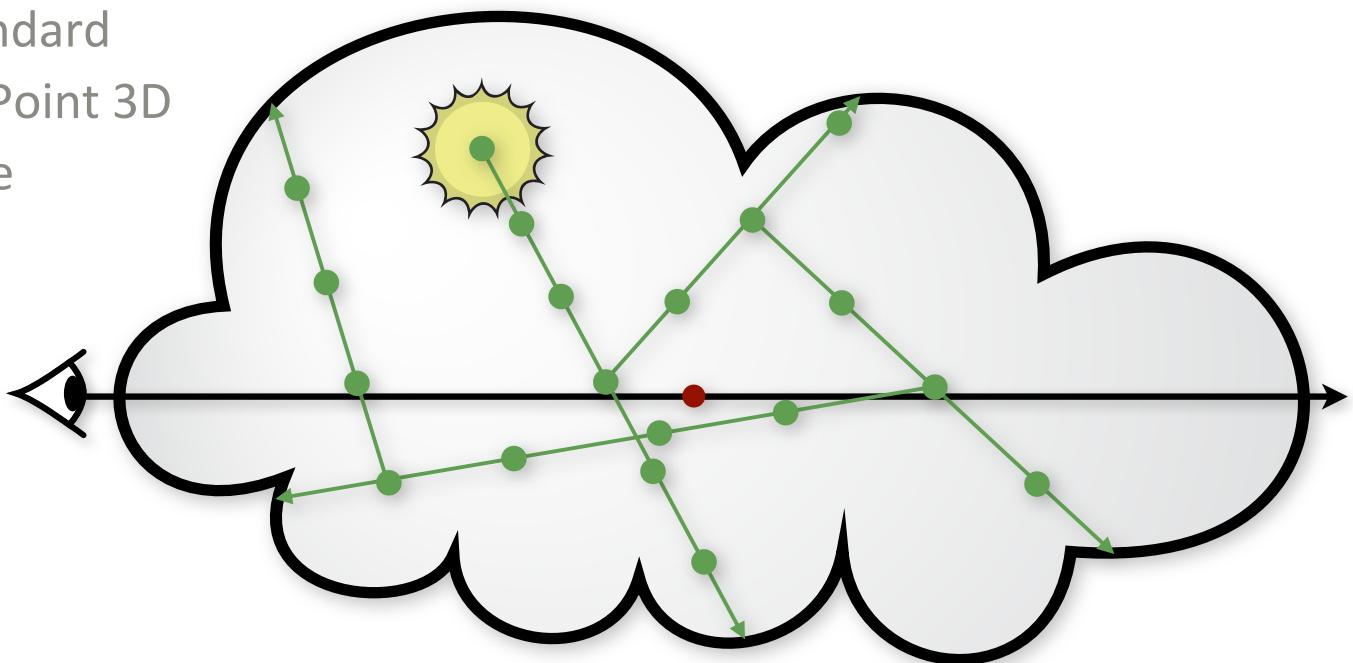
Radiance Estimation using “Discrete Photon Beams”





Radiance Estimation using “Discrete Photon Beams”

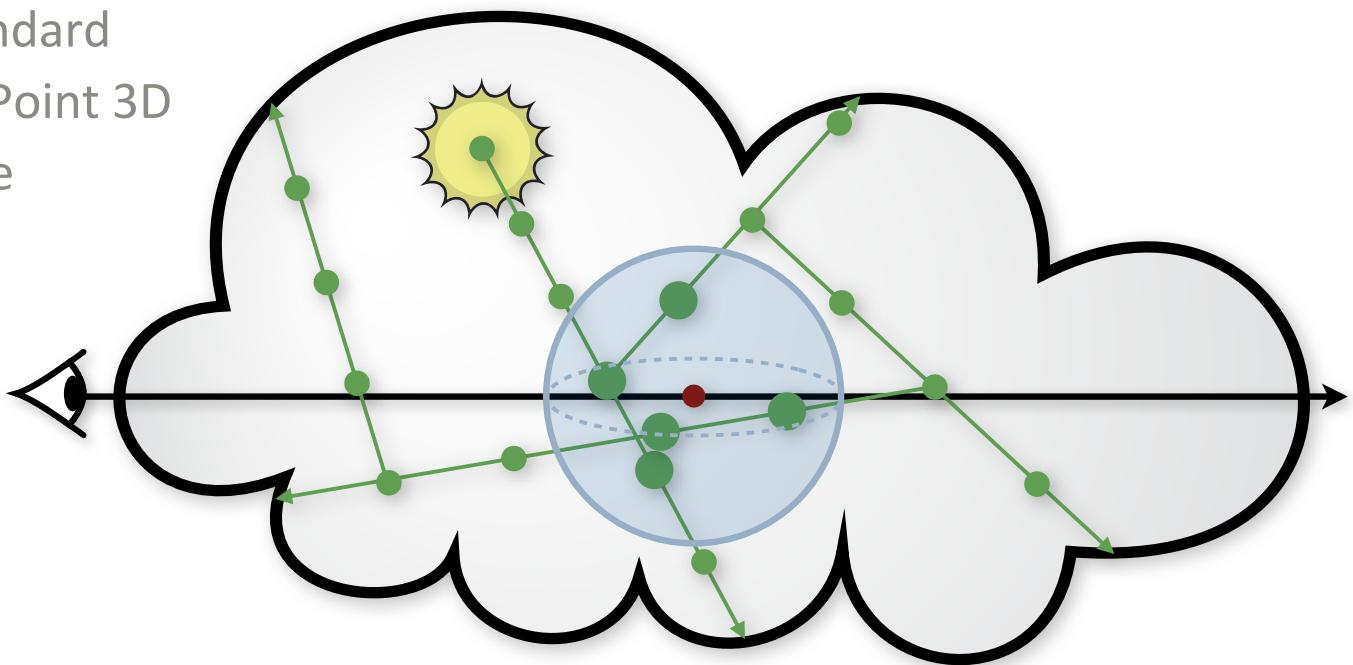
- Use standard
Point x Point 3D
estimate





Radiance Estimation using “Discrete Photon Beams”

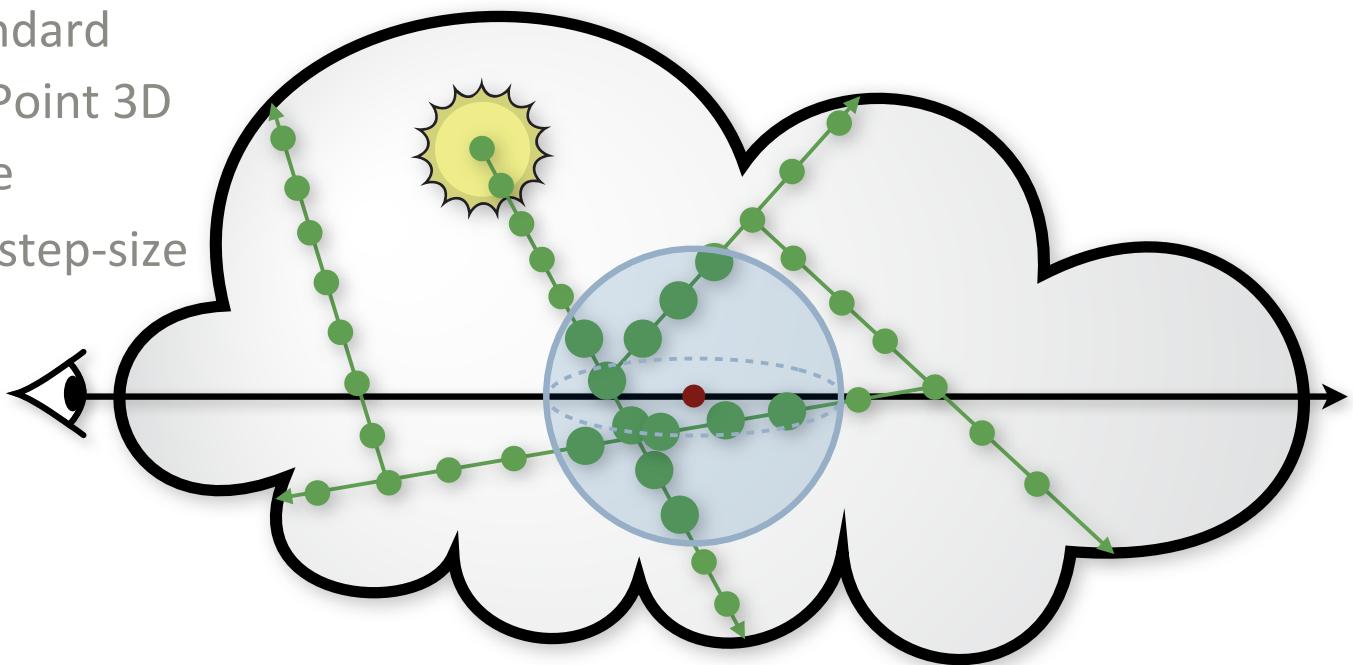
- Use standard
Point x Point 3D
estimate





Radiance Estimation using “Discrete Photon Beams”

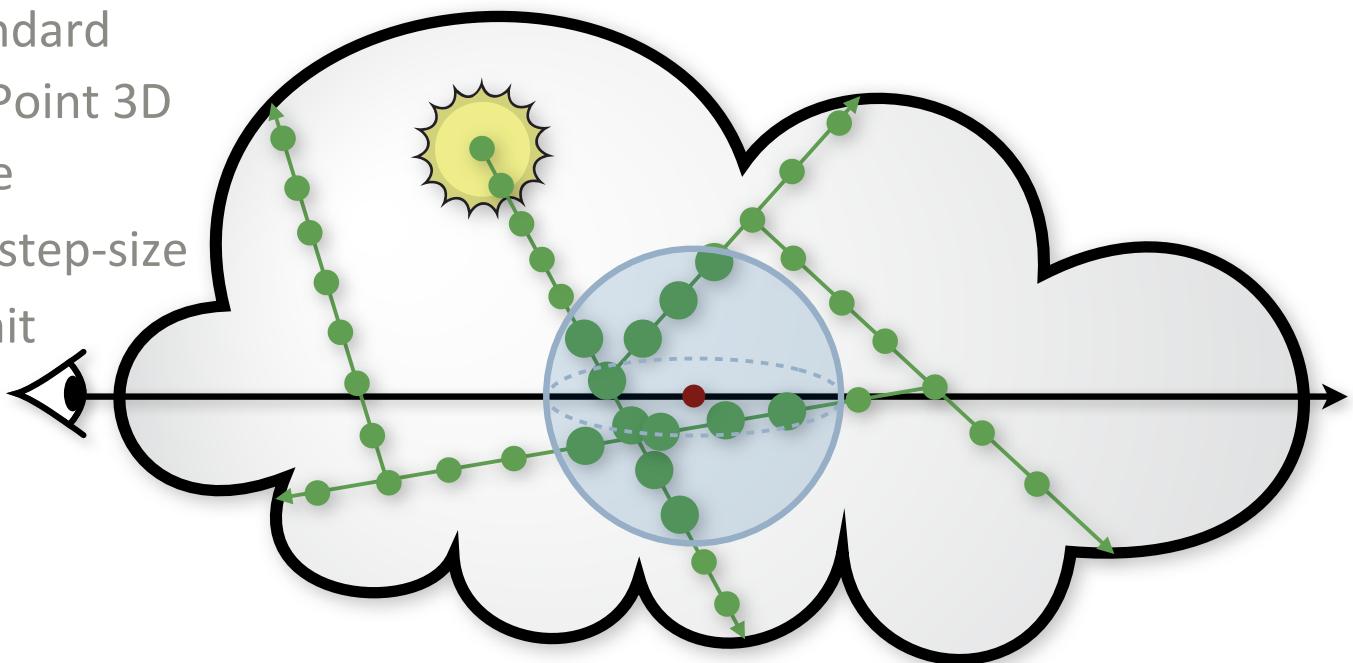
- Use standard Point x Point 3D estimate
- Reduce step-size





Radiance Estimation using “Discrete Photon Beams”

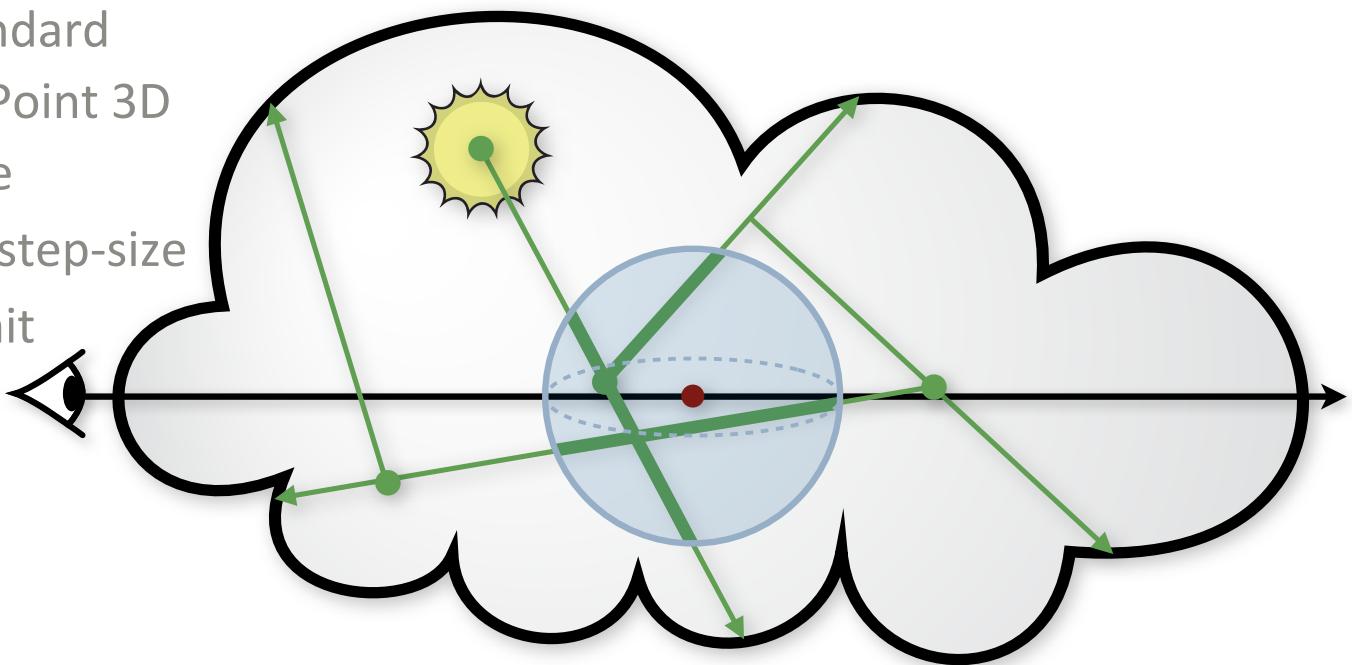
- Use standard Point x Point 3D estimate
- Reduce step-size
- Take limit





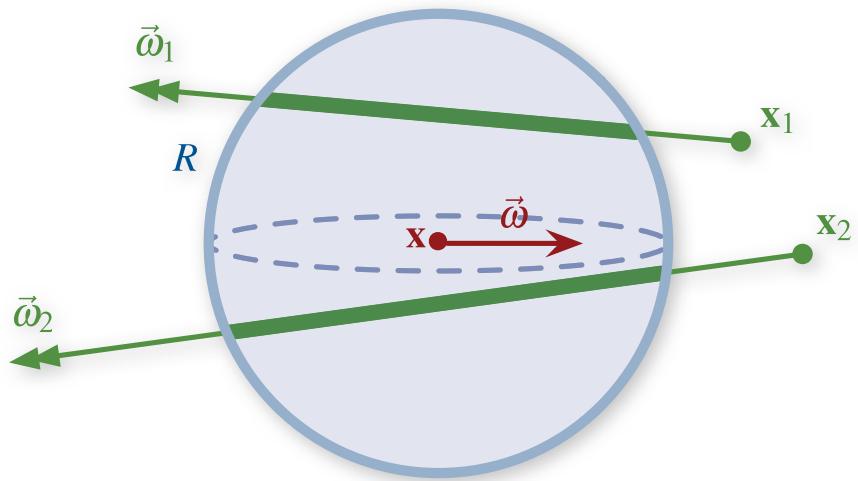
Radiance Estimation using “Discrete Photon Beams”

- Use standard Point x Point 3D estimate
- Reduce step-size
- Take limit





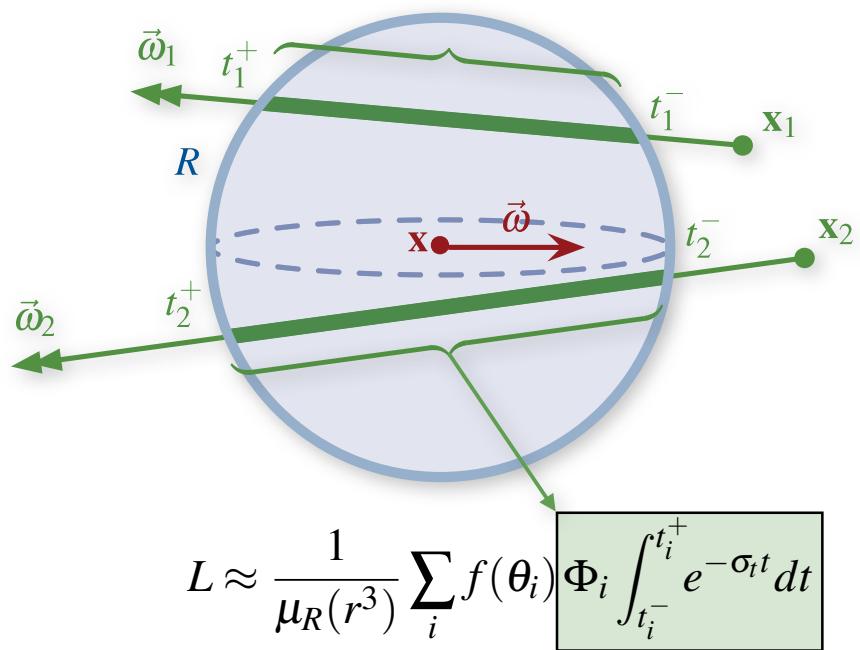
Point Query × Beam Data (3D blur)



$$L \approx \frac{1}{\mu_R(r^3)} \sum_i f(\theta_i) \Phi_i \int_{t_i^-}^{t_i^+} e^{-\sigma_t t} dt$$

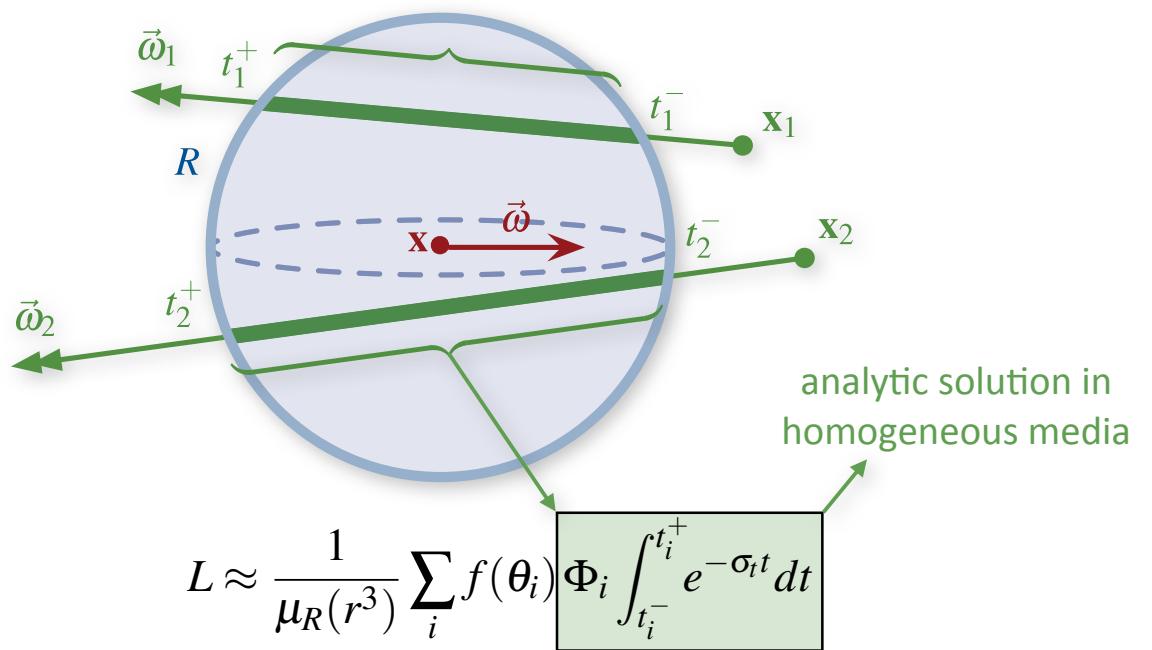


Point Query × Beam Data (3D blur)



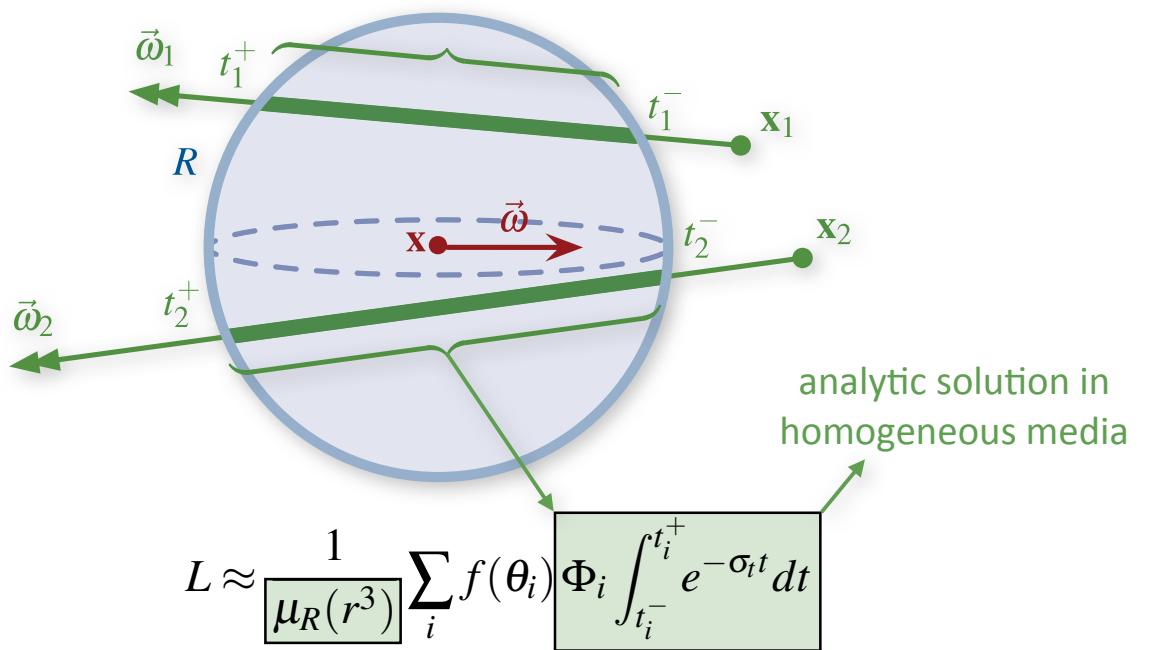


Point Query × Beam Data (3D blur)



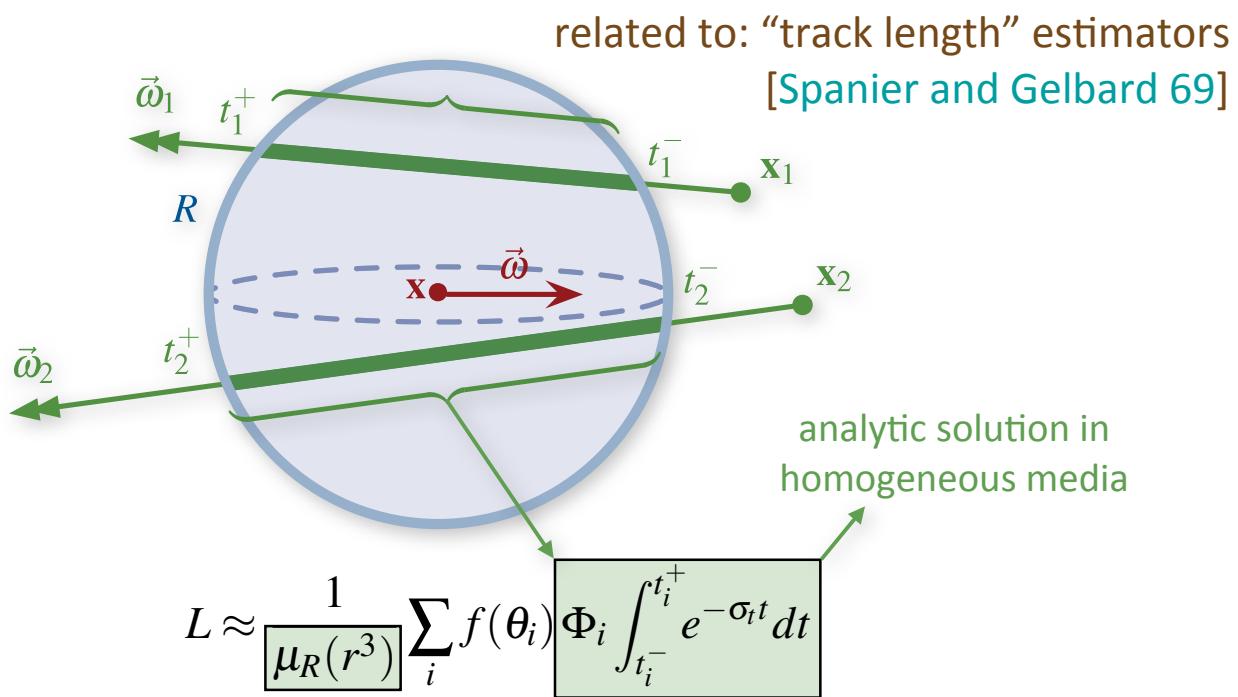


Point Query × Beam Data (3D blur)





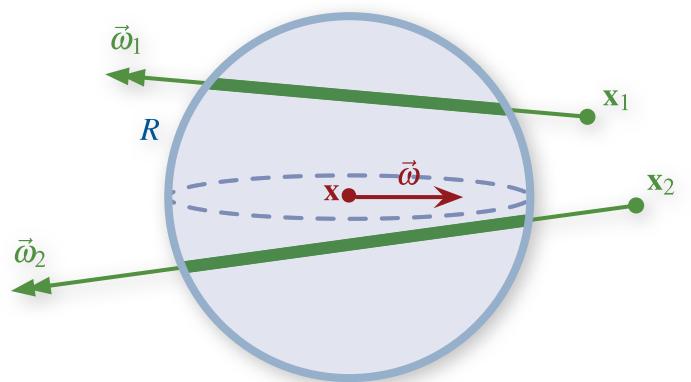
Point Query × Beam Data (3D blur)





Reducing Blur Dimensionality

Point Query x Beam Data (3D blur)

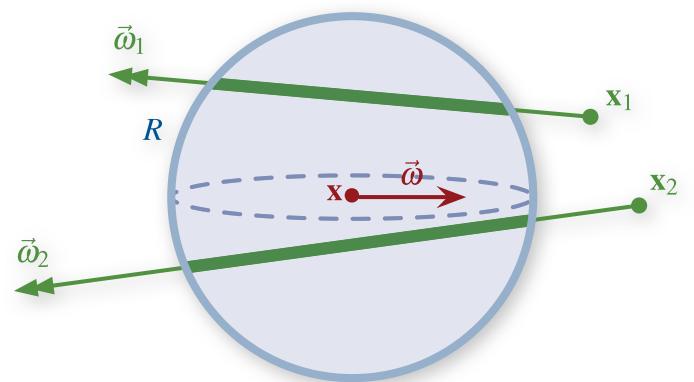


$$L \approx \frac{1}{\mu_R(r^3)} \sum_i f(\theta_i) \Phi_i \int_{t_i^-}^{t_i^+} e^{-\sigma_t t} dt$$



Reducing Blur Dimensionality

Point Query x Beam Data (3D blur)

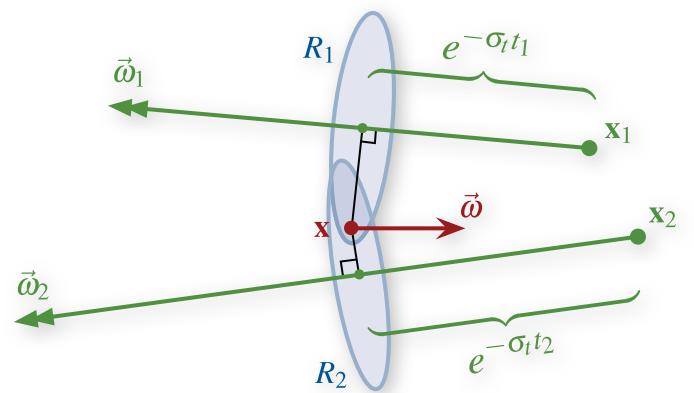


$$L \approx \frac{1}{\mu_R(r^3)} \sum_i f(\theta_i) \boxed{\Phi_i \int_{t_i^-}^{t_i^+} e^{-\sigma_t t} dt}$$



Reducing Blur Dimensionality

Point Query x Beam Data (2D blur)

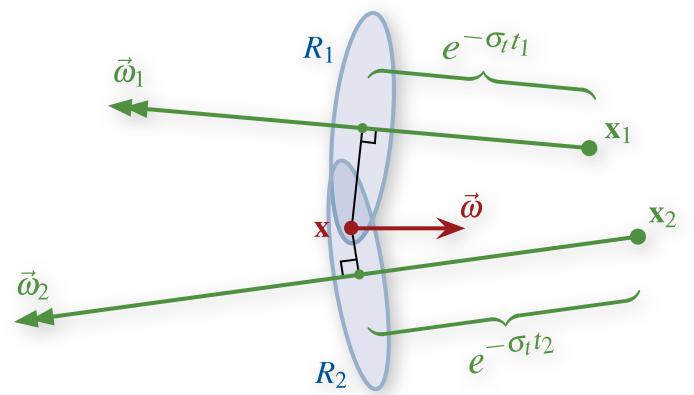


$$L \approx \frac{1}{\mu_R(r^2)} \sum_i f(\theta_i) \boxed{\Phi_i e^{-\sigma_t t_i}}$$



Reducing Blur Dimensionality

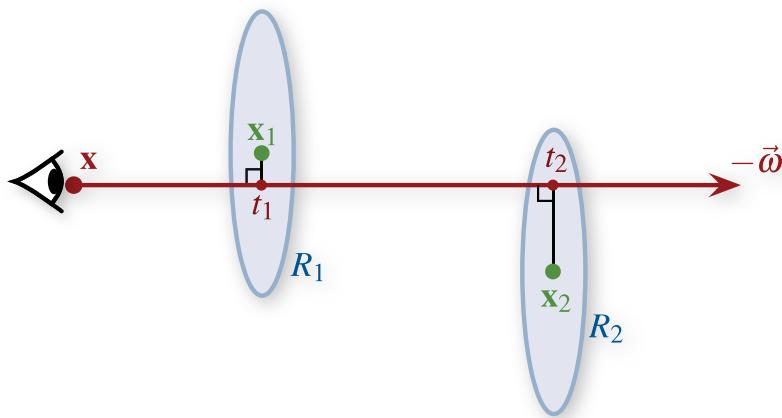
Point Query x Beam Data (2D blur)



$$L \approx \frac{1}{\mu_R(r^2)} \sum_i f(\theta_i) \Phi_i e^{-\sigma_t t_i}$$

Radiometric Duality

Beam Query x Point Data (2D blur)

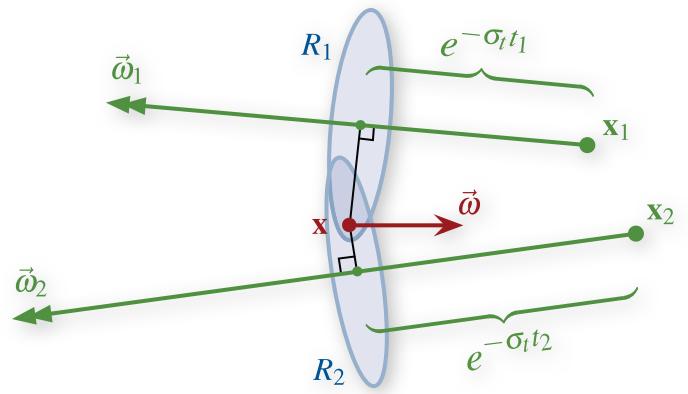


$$L \approx \frac{1}{\mu_R(r^2)} \sum_i f(\theta_i) \Phi_i e^{-\sigma_t t_i}$$

“Beam Radiance Estimate”

[Jarosz et al. 08]

Point Query x Beam Data (2D blur)

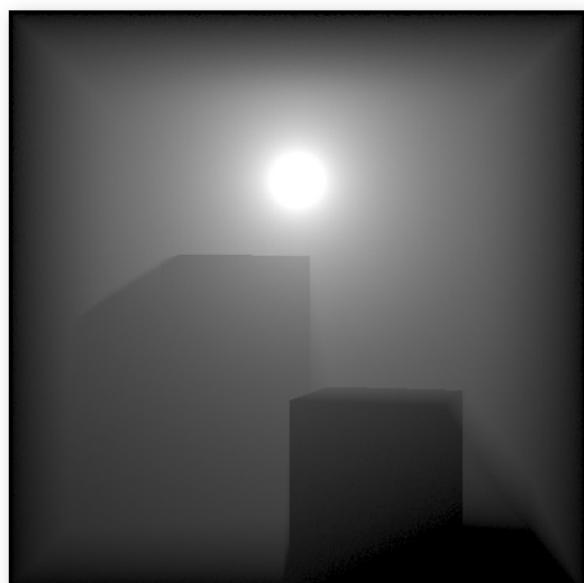


$$L \approx \frac{1}{\mu_R(r^2)} \sum_i f(\theta_i) \Phi_i e^{-\sigma_t t_i}$$



Photon Points vs. Photon Beams

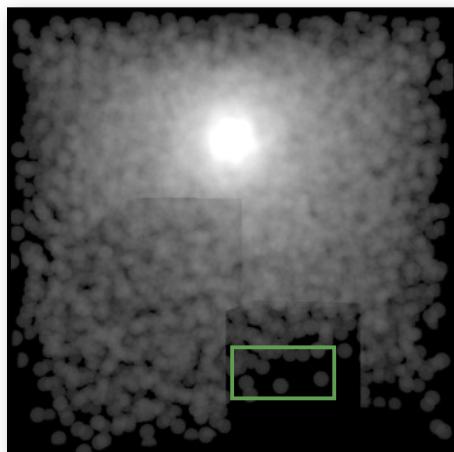
Ground Truth



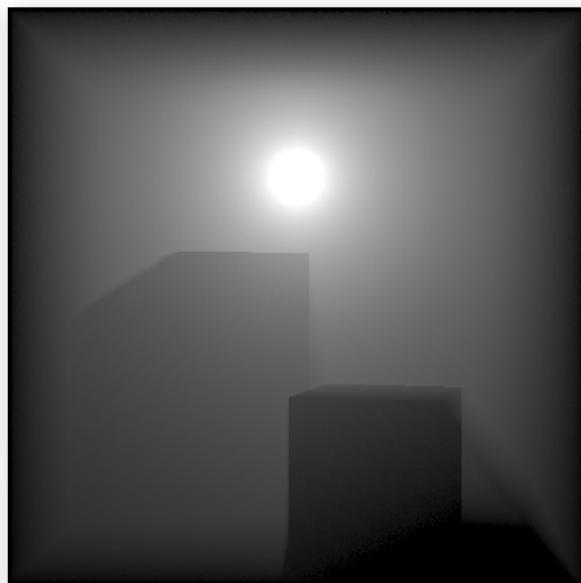


Photon Points vs. Photon Beams

100k Photon Points



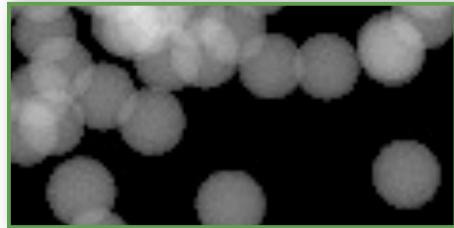
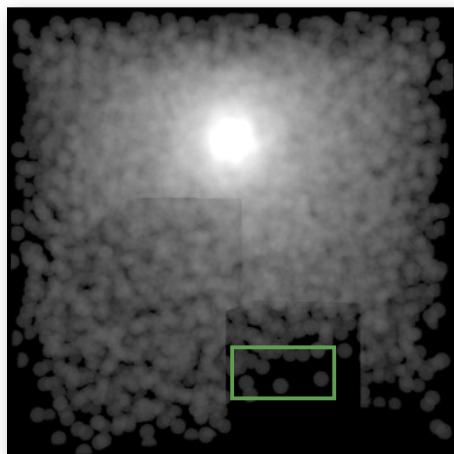
Ground Truth



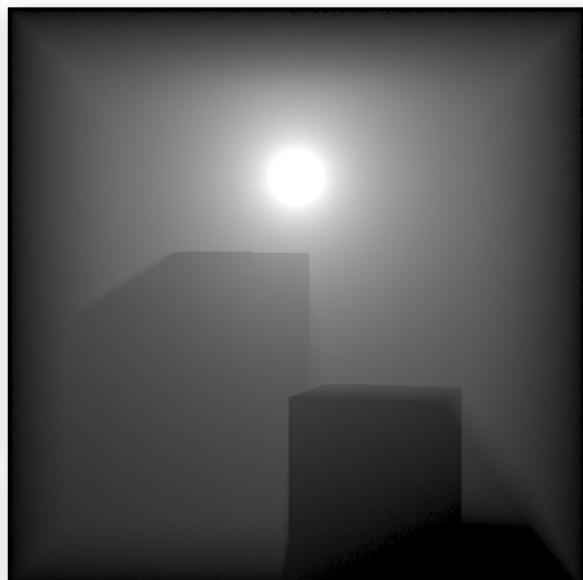


Photon Points vs. Photon Beams

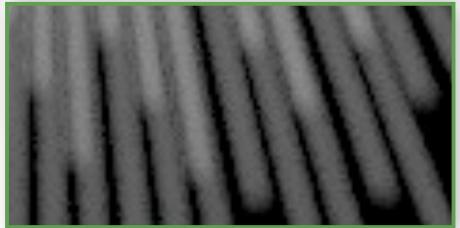
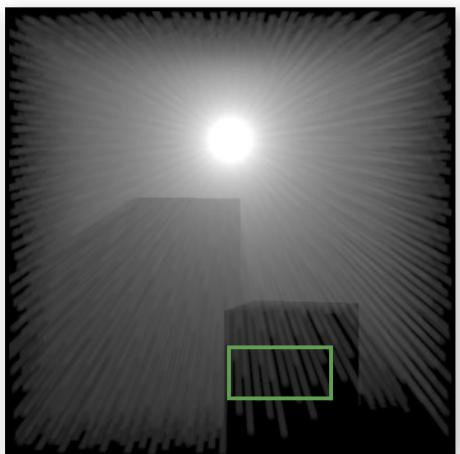
100k Photon Points



Ground Truth



5k Photon Beams





Beam Queries with Photon Beams

- Beam Query x Beam Data (3D)
- Beam Query x Beam Data (2D)₁
- Beam Query x Beam Data (2D)₂
- Beam Query x Beam Data (1D)

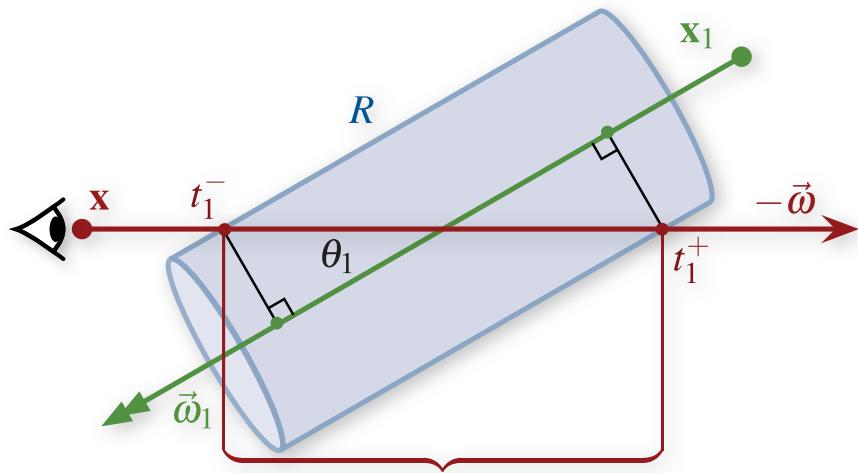


Beam Query \times Beam Data (2D blur)





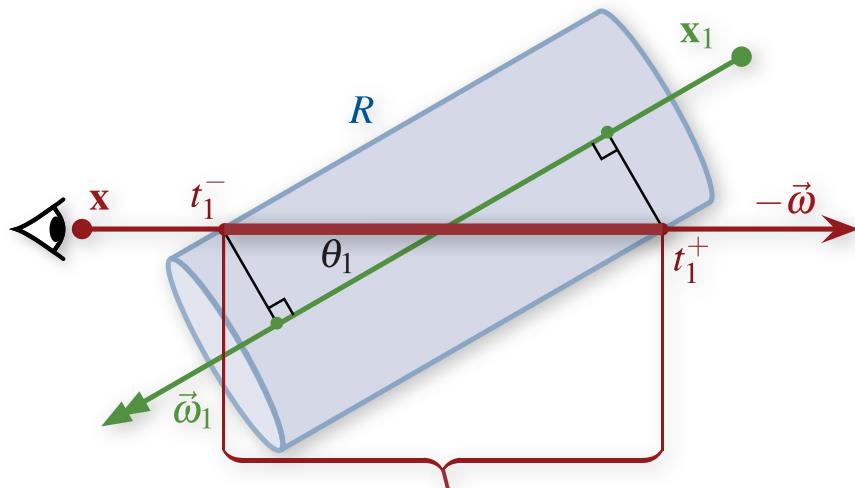
Beam Query \times Beam Data (2D blur)



$$L \approx \frac{\sigma_s}{\mu_R(r^2)} \sum_i f(\theta_i) \Phi_i \int_{t_i^-}^{t_i^+} e^{-\sigma_t t_c} e^{-\sigma_t t_b} dt_c$$



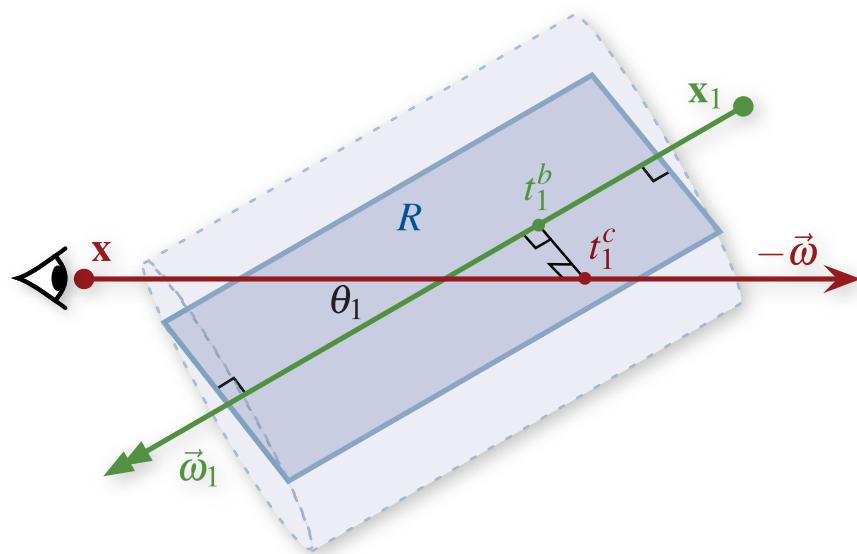
Beam Query \times Beam Data (2D blur)



$$L \approx \frac{\sigma_s}{\mu_R(r^2)} \sum_i f(\theta_i) \Phi_i \int_{t_i^-}^{t_i^+} e^{-\sigma_t t_c} e^{-\sigma_t t_b} dt_c$$



Beam Query \times Beam Data (1D blur)



$$L \approx \frac{\sigma_s}{\mu_R(r)} \sum_i \frac{f(\theta_i) [\Phi_i e^{-\sigma_t t_i^c} e^{-\sigma_t t_i^b}]}{\sin \theta_i}$$



Radiance Estimator Summary



Radiance Estimator Summary

- Beam queries remove ray marching



Radiance Estimator Summary

- Beam queries remove ray marching
- Beam data increases data density



Radiance Estimator Summary

- Beam queries remove ray marching
- Beam data increases data density
- Lower blur dimension reduces bias and computation



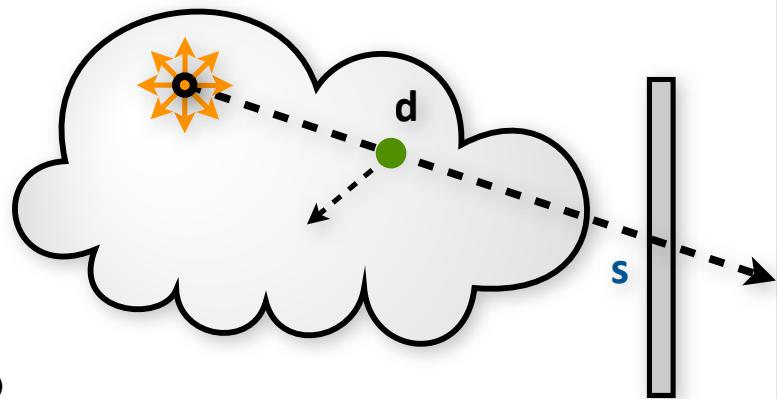
Radiance Estimator Summary

- Beam queries remove ray marching
- Beam data increases data density
- Lower blur dimension reduces bias and computation
- **use: Beam Query x Beam Data (1D)**



Basic Volumetric Photon Tracer

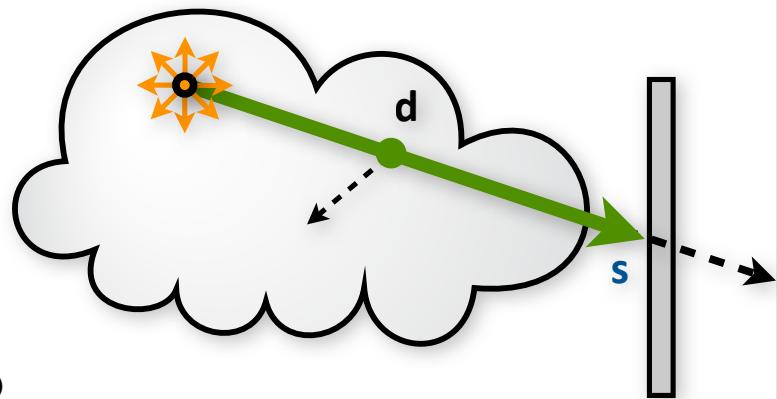
```
void vPT(o, ω, Φ)
    s = nearestSurfaceHit(o, ω)
    d = freeFlightDistance(o, ω)
    if (d < s)          // media scattering
        o += d*ω        // propagate photon
        storeVolumePhoton(o, ω, Φ)
    return vPT(o, samplePF(), Φ * σs / σt)
else                      // surface scattering
    o += s*ω        // propagate photon
    storeSurfacePhoton(o, ω, Φ)
    (ωi, pdfi) = sampleBRDF(o, ω)
    return vPT(o, ωi, Φ * BRDF(o, ω, ωi) / pdfi)
```





Basic Volumetric Photon Tracer

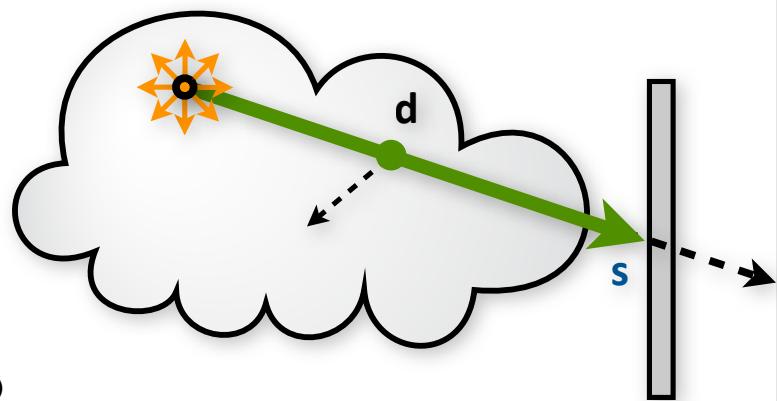
```
void vPT(o, ω, Φ)
    s = nearestSurfaceHit(o, ω)
    storeVolumePhoton(o, ω, Φ)
    d = freeFlightDistance(o, ω)
    if (d < s)          // media scattering
        o += d*ω        // propagate photon
        return vPT(o, samplePFC(), Φ * σs / σt)
    else                // surface scattering
        o += s*ω        // propagate photon
        storeSurfacePhoton(o, ω, Φ)
        (ωi, pdfi) = sampleBRDF(o, ω)
        return vPT(o, ωi, Φ * BRDF(o, ω, ωi) / pdfi)
```

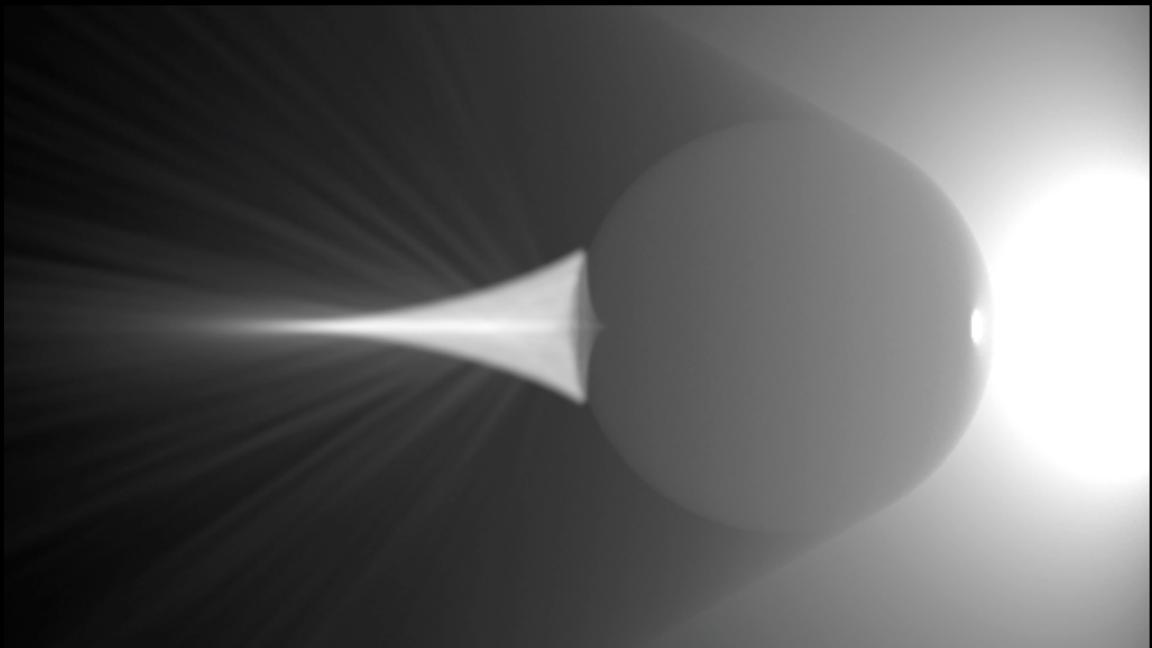


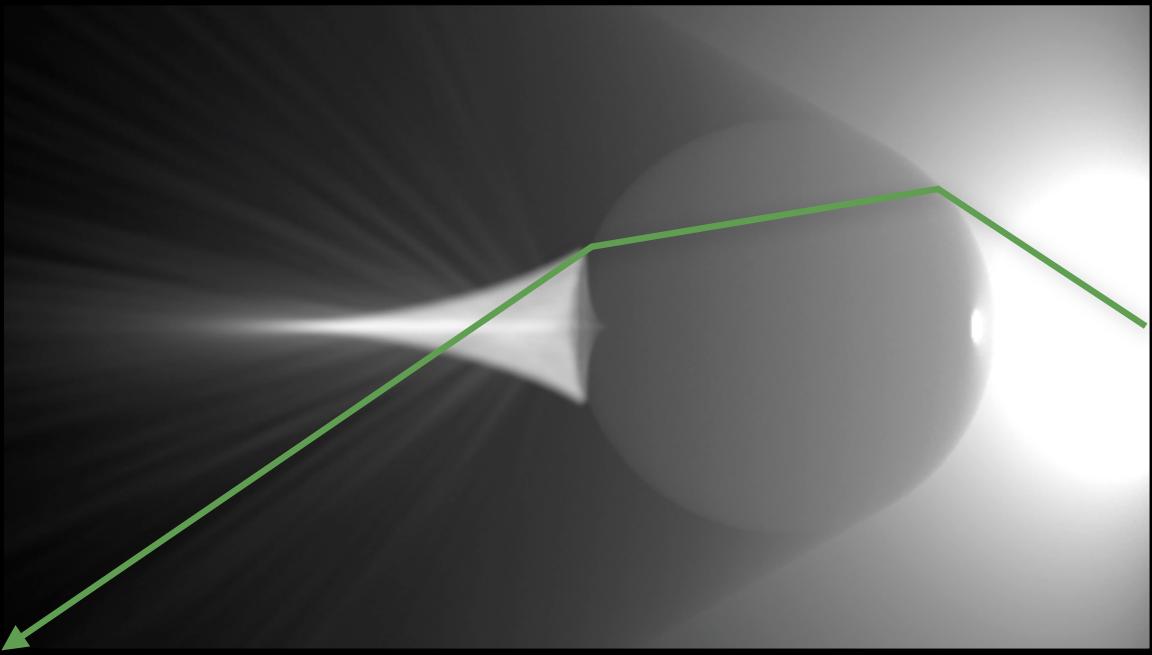


Basic Volumetric Photon Tracer

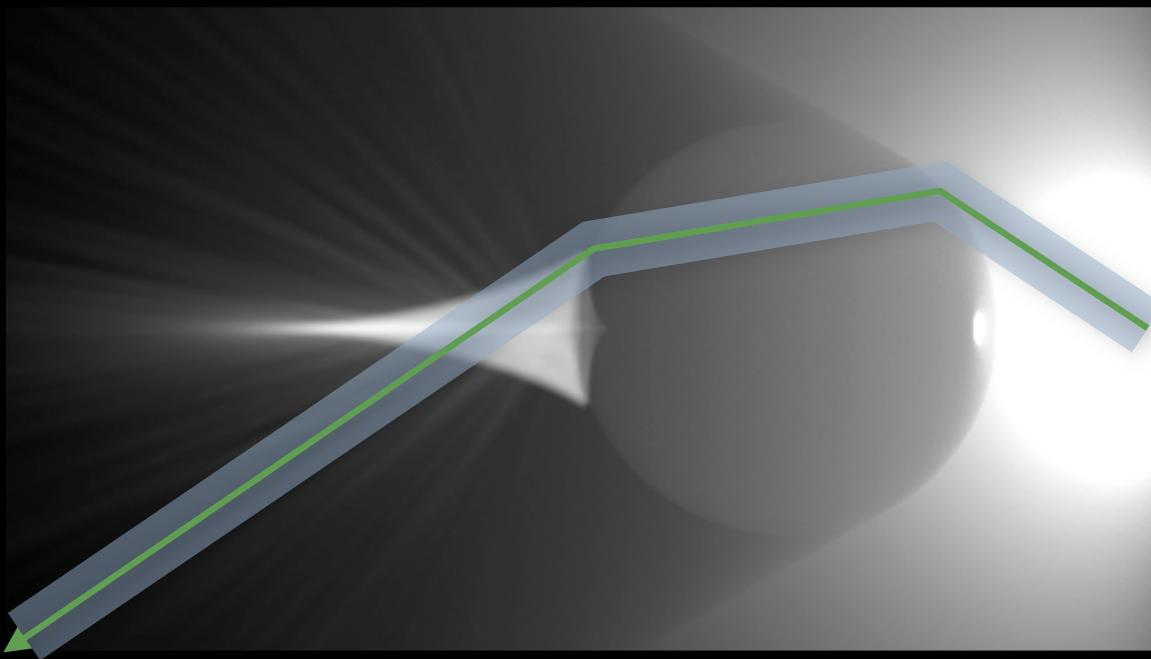
```
void vPT(o, ω, φ)
    s = nearestSurfaceHit(o, ω)
    storePhotonBeam(o, ω, s, φ)
    d = freeFlightDistance(o, ω)
    if (d < s)          // media scattering
        o += d*ω        // propagate photon
        return vPT(o, samplePFC(), φ * σs / σt)
    else                // surface scattering
        o += s*ω        // propagate photon
        storeSurfacePhoton(o, ω, φ)
        (ωi, pdfi) = sampleBRDF(o, ω)
        return vPT(o, ωi, φ * BRDF(o, ω, ωi) / pdfi)
```



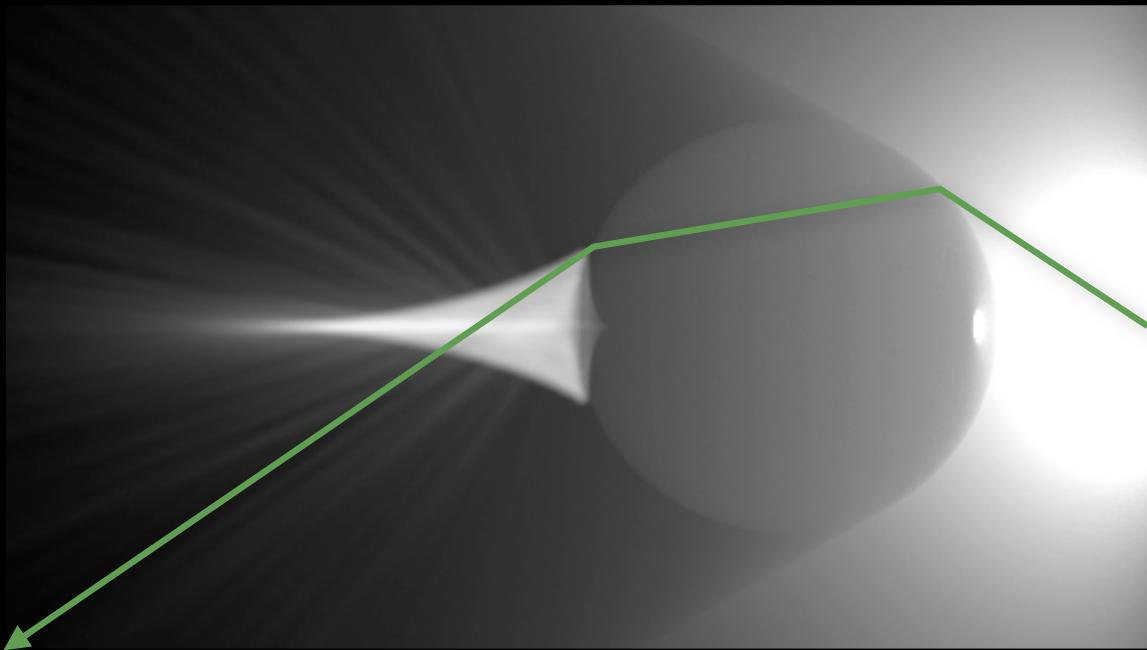




Fixed-width Beams

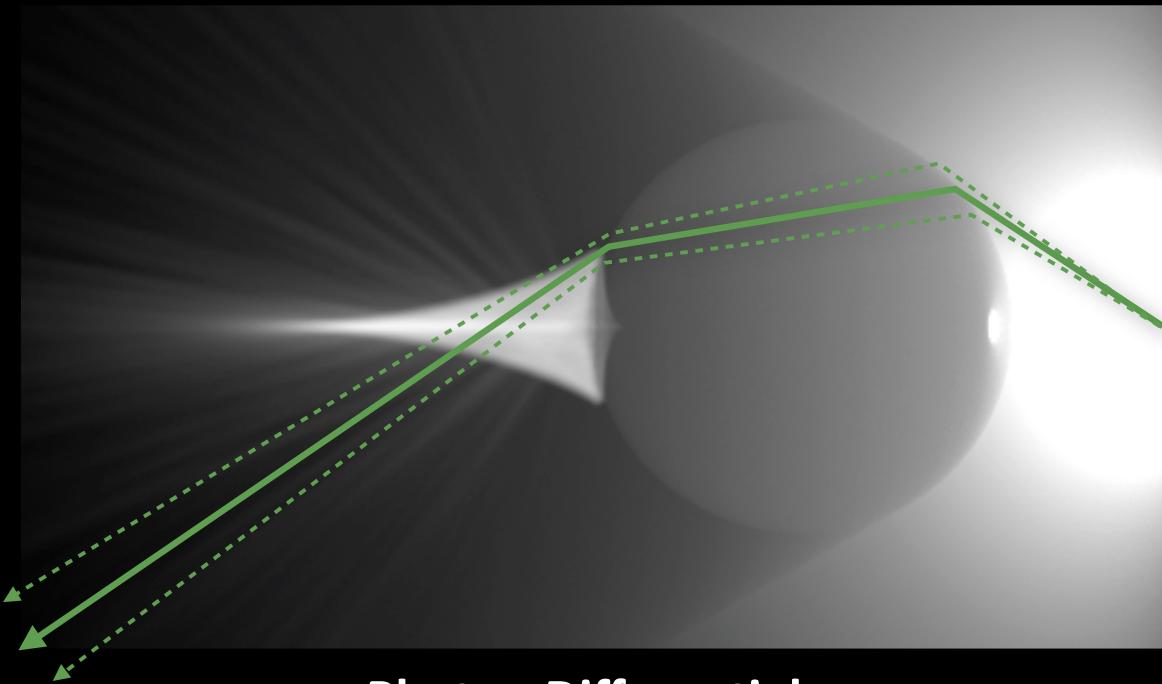


Fixed-width Beams



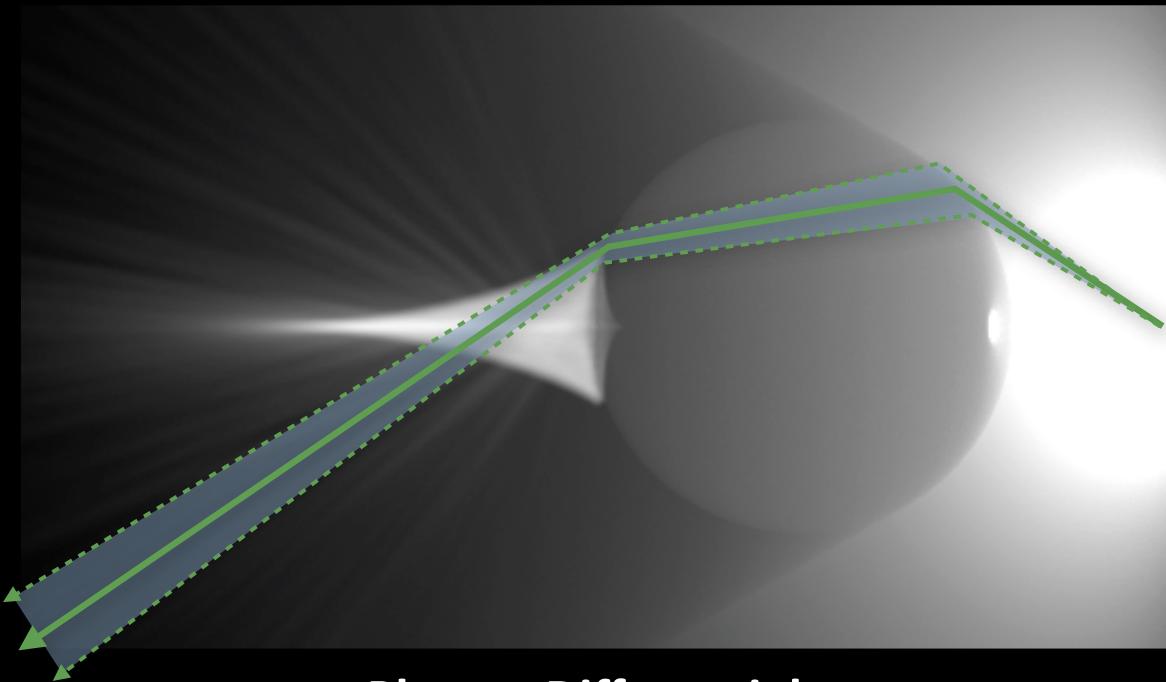
Photon Differentials
[Igehy 99, Schjøth et al. 07]

Fixed-width Beams



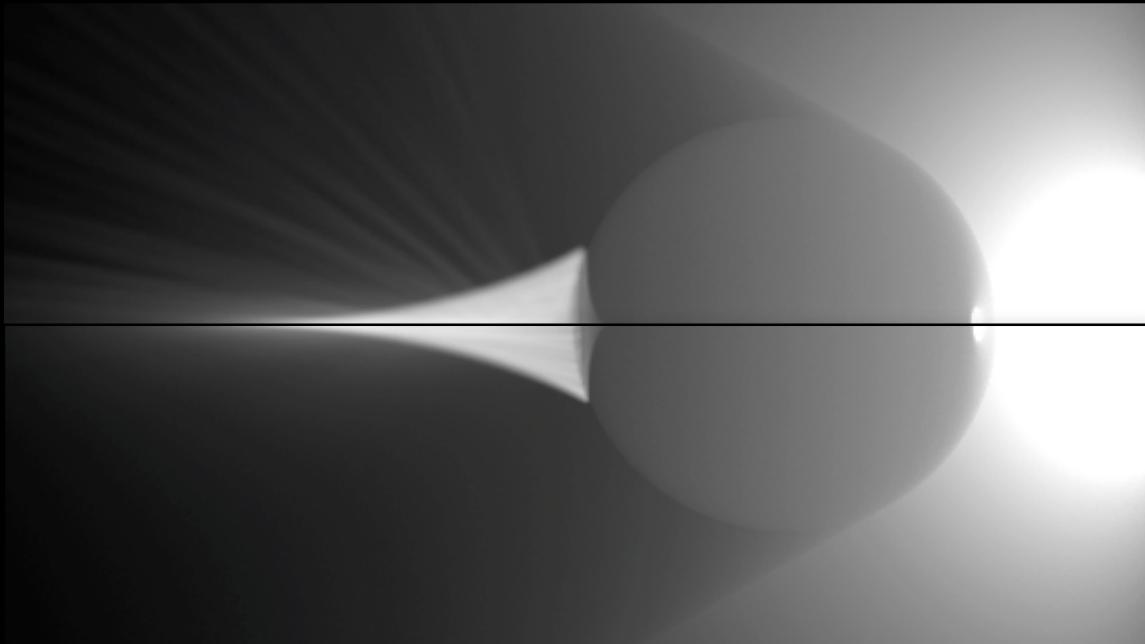
Photon Differentials
[Igehy 99, Schjøth et al. 07]

Fixed-width Beams



Photon Differentials
[Igehy 99, Schjøth et al. 07]

Fixed-width Beams



Adaptive-width Beams



Rendering



Rendering

- Need to intersect each ray with all photon beams
(expensive!)



Rendering

- Need to intersect each ray with all photon beams (expensive!)
- Place photon beams in an acceleration structure



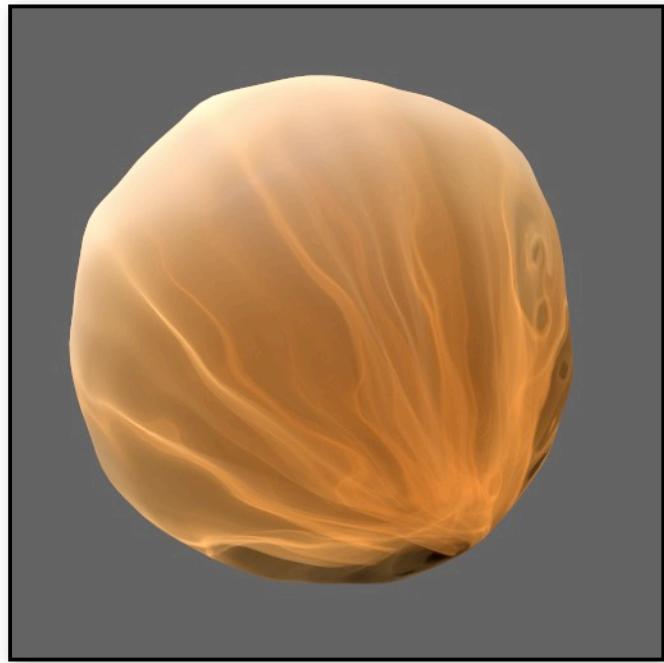
Rendering

- Need to intersect each ray with all photon beams (expensive!)
- Place photon beams in an acceleration structure
- Rasterization (beams are just axial billboards!)



Bumpy Sphere

courtesy of Bruce Walter

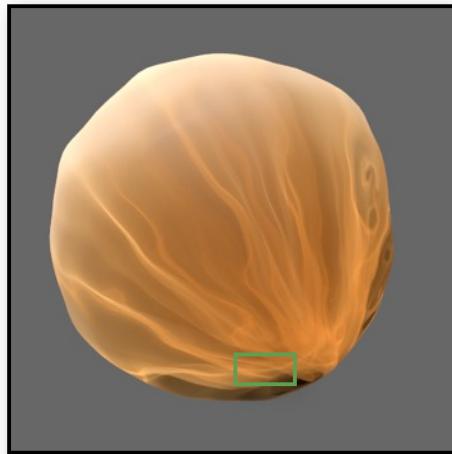




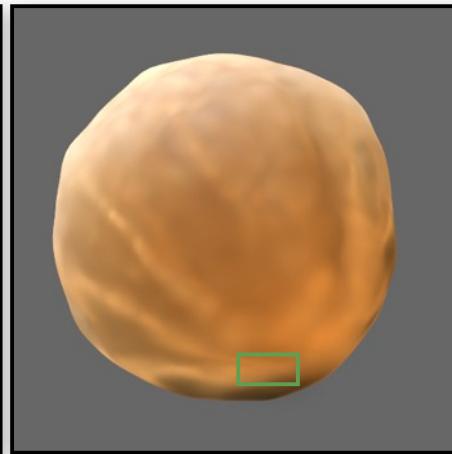
Bumpy Sphere

courtesy of Bruce Walter

Ground Truth



90k Photon Points

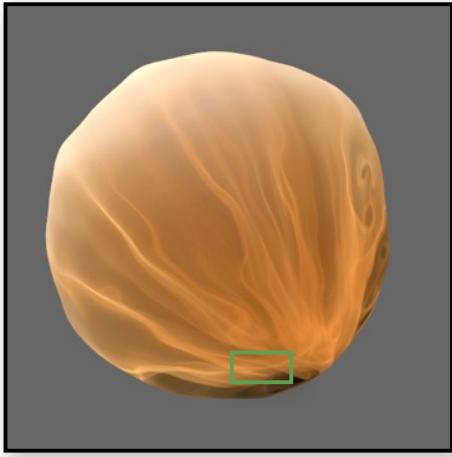




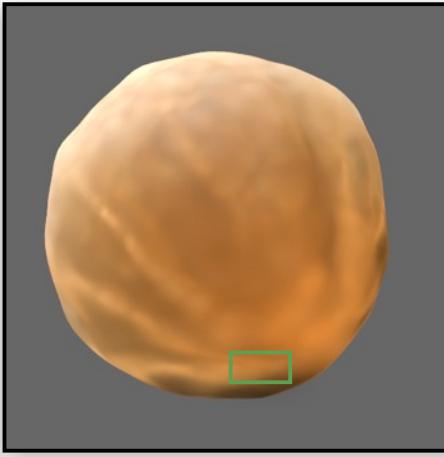
Bumpy Sphere

courtesy of Bruce Walter

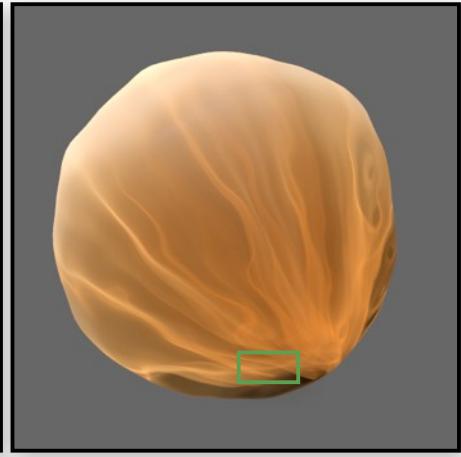
Ground Truth



90k Photon Points



90k Photon Beams



Bumpy Sphere

Rendered at 512x512 with up to 16 samples/pixel

Equal Photon Count

Photon Points

Photon Beams



90K Photon **Points**
~ 40 seconds/frame



90K Photon **Beams**
~ 103 seconds/frame

Equal Render Time

Photon Points

Photon Beams



1.3M Photon **Points**
~ 101 seconds/frame



90K Photon **Beams**
~ 103 seconds/frame

Lighthouse

Photon Points



10K Photon Points
~ 31 seconds/frame

Roughly Equal Time

Photon Beams



700 Photon Beams
~ 25 seconds/frame

Lighthouse

Underwater Sun Beams

Rendered at 1024x576 with up to 16 samples/pixel

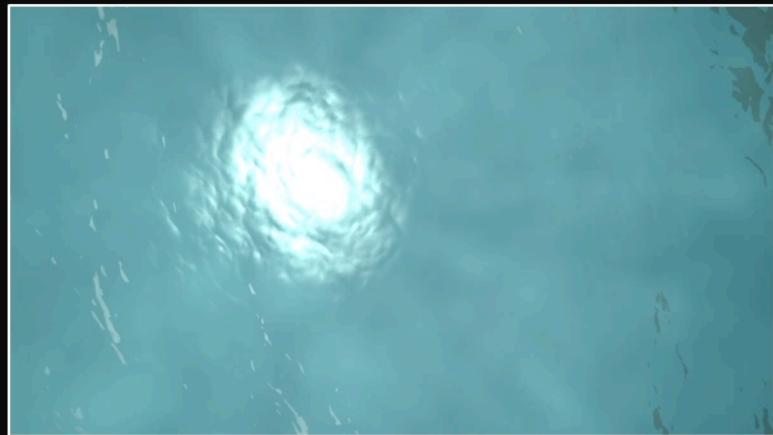
1M Photon Points
~ 226 seconds/frame

9x Render Time

700 Photon Beams
~ 25 seconds/frame

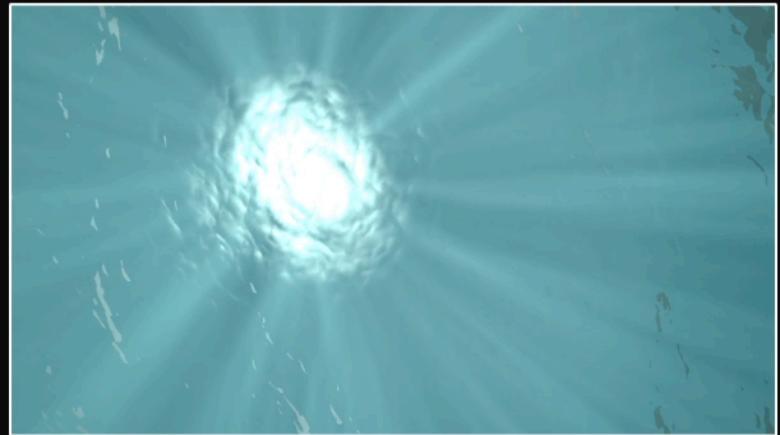
Underwater Sun Beams

Photon Points



100K Photon Points
~ 204 seconds/frame

Photon Beams



25K Photon Beams
~ 200 seconds/frame

Roughly Equal Time



Progressive Photon Beams



Progressive Photon Beams

- Combine benefits of:



Progressive Photon Beams

- Combine benefits of:
 - photon beams



Progressive Photon Beams

- Combine benefits of:
 - photon beams
 - progressive photon mapping



Statistical Convergence

- Previous derivations not directly applicable
 - beam density vs. point density



Statistical Convergence

- Previous derivations not directly applicable
 - beam density vs. point density
- Reduction factor: $f_i = \frac{i + \alpha}{i + 1}$



Statistical Convergence

- Previous derivations not directly applicable
 - beam density vs. point density
- Reduction factor: $f_i = \frac{i + \alpha}{i + 1}$
- *Application* of factor depends on blur dimensionality
 - Surfaces (2D): $r_{i+1}^2 = f_i \cdot r_i^2$



Statistical Convergence

- Previous derivations not directly applicable
 - beam density vs. point density
- Reduction factor: $f_i = \frac{i + \alpha}{i + 1}$
- *Application* of factor depends on blur dimensionality
 - Surfaces (2D): $r_{i+1}^2 = f_i \cdot r_i^2$
 - Volumetric photon mapping (3D): $r_{i+1}^3 = f_i \cdot r_i^3$



Statistical Convergence

- Previous derivations not directly applicable
 - beam density vs. point density
- Reduction factor: $f_i = \frac{i + \alpha}{i + 1}$
- *Application* of factor depends on blur dimensionality
 - Surfaces (2D): $r_{i+1}^2 = f_i \cdot r_i^2$
 - Volumetric photon mapping (3D): $r_{i+1}^3 = f_i \cdot r_i^3$
 - Beam × Beam (1D): $r_{i+1} = f_i \cdot r_i$



Algorithm



Algorithm

Step 1:

- Photon tracing: emit, scatter, store beams
- Scale beam widths by global factor r_i



Algorithm

Step 1:

- Photon tracing: emit, scatter, store beams
- Scale beam widths by global factor r_i

Step 2:

- Trace random camera path, evaluate radiance estimate along each ray using beams



Algorithm

Step 1:

- Photon tracing: emit, scatter, store beams
- Scale beam widths by global factor r_i

Step 2:

- Trace random camera path, evaluate radiance estimate along each ray using beams
- Display running average



Algorithm

Step 1:

- Photon tracing: emit, scatter, store beams
- Scale beam widths by global factor r_i

Step 2:

- Trace random camera path, evaluate radiance estimate along each ray using beams
- Display running average
- **Reduce** global factor r_i and **repeat**



Algorithm

Step 1:

- Photon tracing: emit, scatter, store beams
- Scale beam widths by global factor r_i

Step 2:

Trivially Parallelizable

- Trace random camera path, evaluate radiance estimate along each ray using beams
- Display running average
- *Reduce global factor r_i and repeat*



Evaluating the Transmittance

- Need to compute transmittance: along photon beam, along camera ray



Evaluating the Transmittance

- Need to compute transmittance: along photon beam, along camera ray
- Homogeneous: analytic



Evaluating the Transmittance

- Need to compute transmittance: along photon beam, along camera ray
- Homogeneous: analytic
- Heterogeneous: use progressive deep shadow maps



Results & Implementation

- 3 implementations:
 - GPU-only OptiX ray-tracer
 - GPU-only rasterization
 - General: Hybrid CPU/GPU



Results & Implementation

- 3 implementations:
 - GPU-only OptiX ray-tracer
 - GPU-only rasterization
 - General: Hybrid CPU/GPU

scene courtesy of Bruce Walter

BUMPYSHERE

OPTIX
IMPLEMENTATION



2x speed

alpha: 0.60 beams per pass: 1024 pass number: 1 render time per pass: 132.97 ms

Thursday, August 23, 12



Results & Implementation

- 3 implementations:
 - GPU-only OptiX ray-tracer
 - GPU-only **rasterization**
 - General: Hybrid CPU/GPU

```
alpha = 0.5  
P = 0.037895  
Shadow map resolution: 64 x 64  
pass number: 14  
average render time per pass: 33 ms
```

www.fraps.com

OCEAN

OPENGL
RASTERIZATION-ONLY
IMPLEMENTATION

$\alpha = 0.5$

2x speed

Thursday, August 23, 12



Results & Implementation

- 3 implementations:
 - GPU-only OptiX ray-tracer
 - GPU-only rasterization
 - General: Hybrid CPU/GPU

CARS

1280x720, Depth-of-Field

Pass 1



Homogeneous



Heterogeneous

Pass 1



Average of Passes 1..1



Thursday, August 23, 12

Pass 2



Average of Passes 1..2



Thursday, August 23, 12

Pass 4



Average of Passes 1..4



Thursday, August 23, 12

Pass 8



Average of Passes 1..8



Thursday, August 23, 12

Pass 16



Average of Passes 1..16



Thursday, August 23, 12

Pass 32



Average of Passes 1..32



Thursday, August 23, 12

Pass 64



Average of Passes 1..64



Thursday, August 23, 12

Pass 128



Average of Passes 1..128



Thursday, August 23, 12

Pass 256



Average of Passes 1..256



Thursday, August 23, 12

Pass 512



Average of Passes 1..512



Thursday, August 23, 12

Pass 1024



Average of Passes 1..1024



Thursday, August 23, 12

CARS

1280x720, Depth-of-Field



Homogeneous

14.55M Photon Beams
9.5 minutes



Heterogeneous

15.04M Photon Beams
16.8 minutes

CARS

1280x720, Depth-of-Field

Homogeneous
14.55M Photon Beams
9.5 minutes



Heterogeneous
15.04M Photon Beams
16.8 minutes



FLASHLIGHTS

1280x720, Depth-of-Field

Pass 1



Average of Passes 1..1

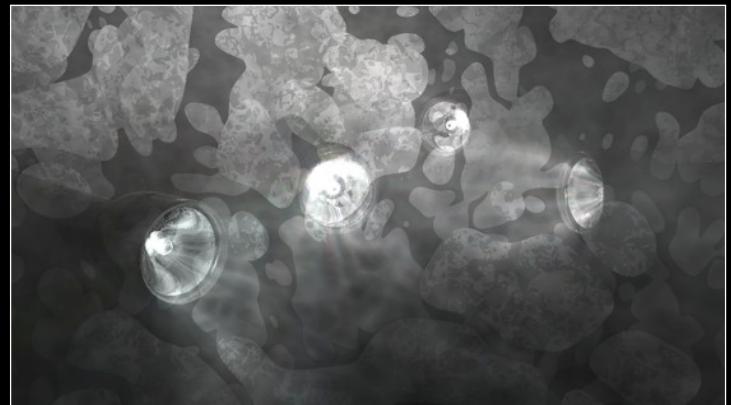
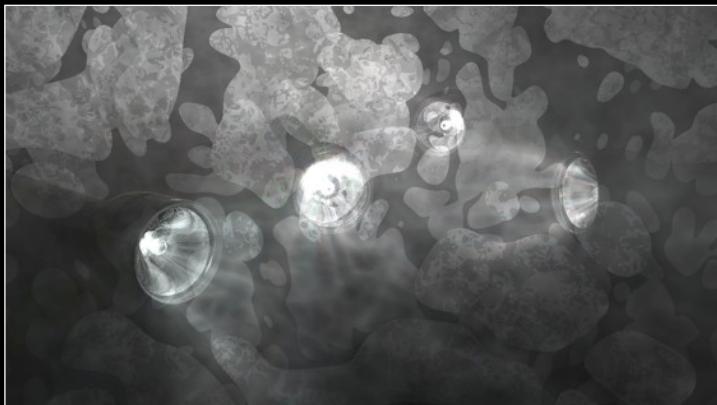


Thursday, August 23, 12

Pass 1



Average of Passes 1..1

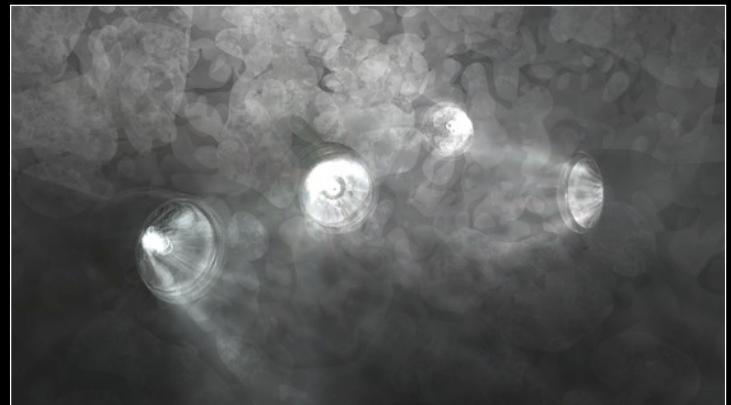
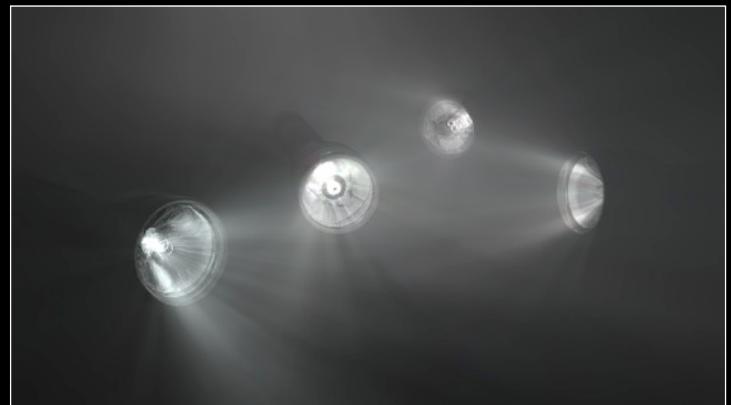


Thursday, August 23, 12

Pass 2



Average of Passes 1..2

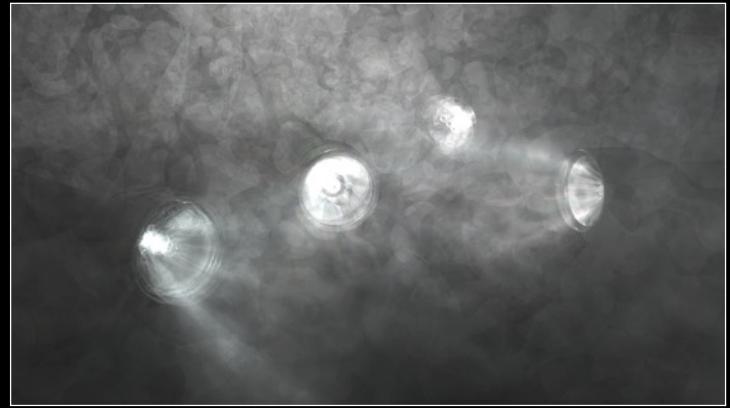
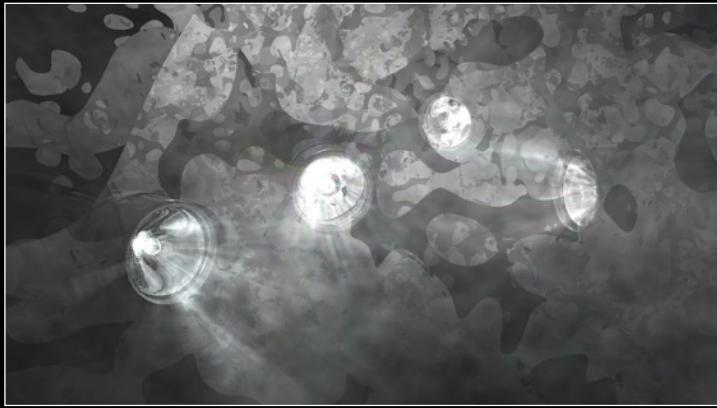
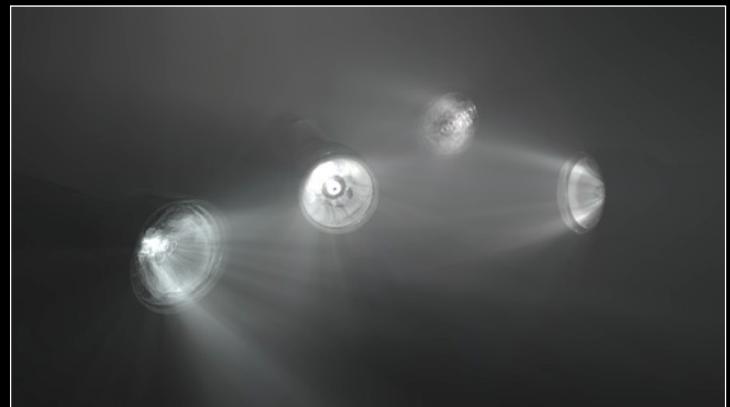


Thursday, August 23, 12

Pass 4

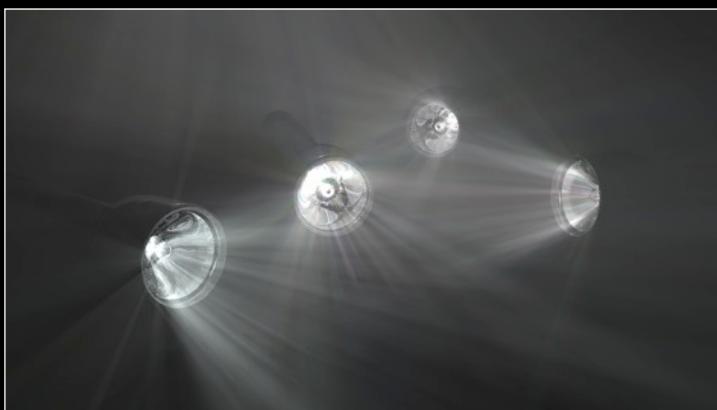


Average of Passes 1..4

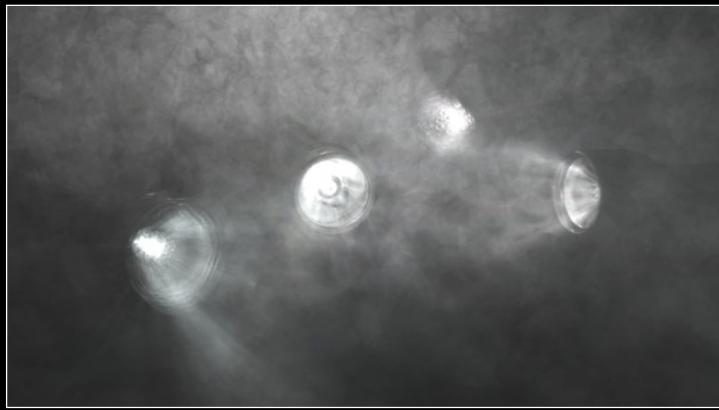
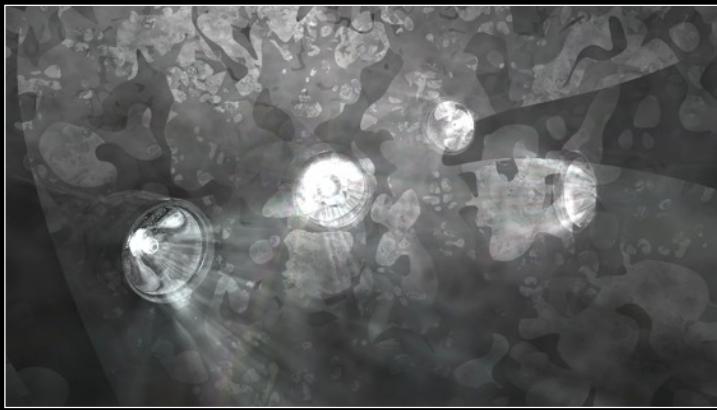
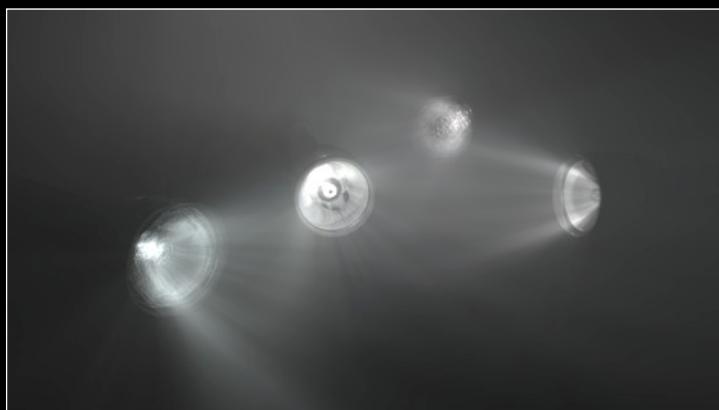


Thursday, August 23, 12

Pass 8

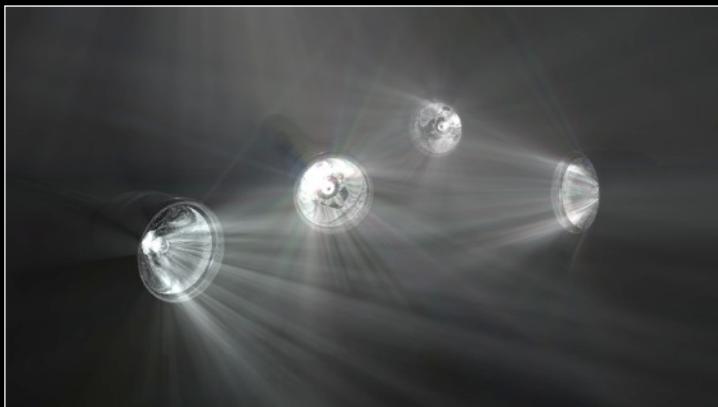


Average of Passes 1..8

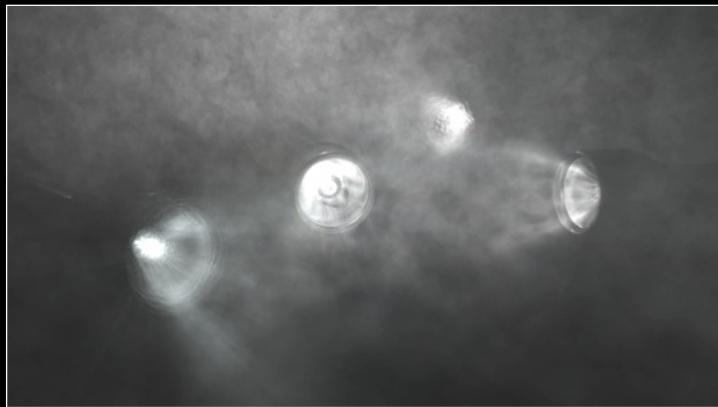
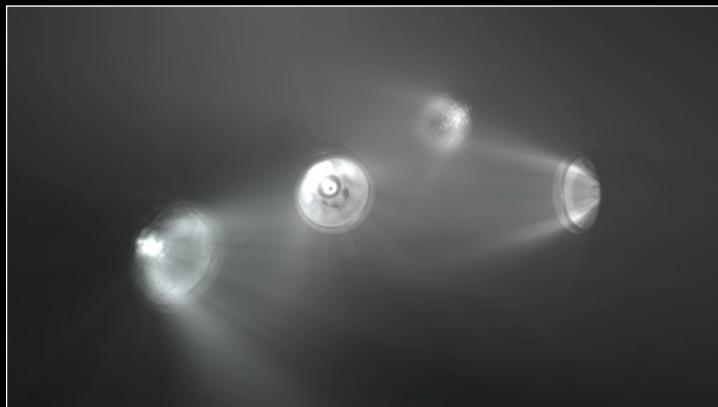


Thursday, August 23, 12

Pass 16

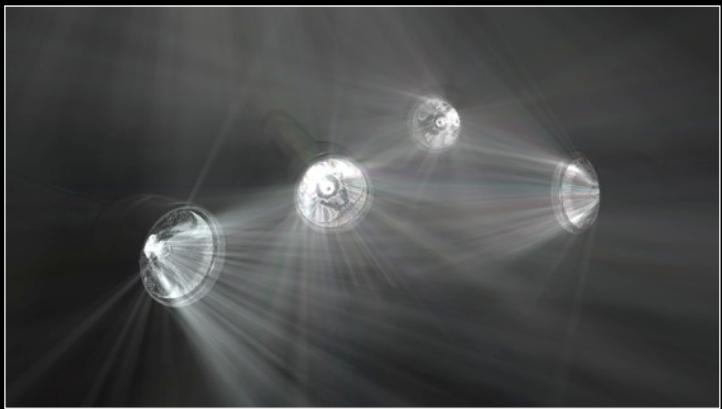


Average of Passes 1..16

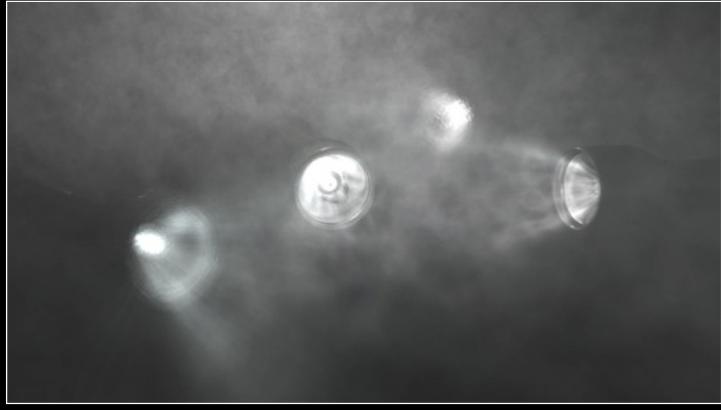
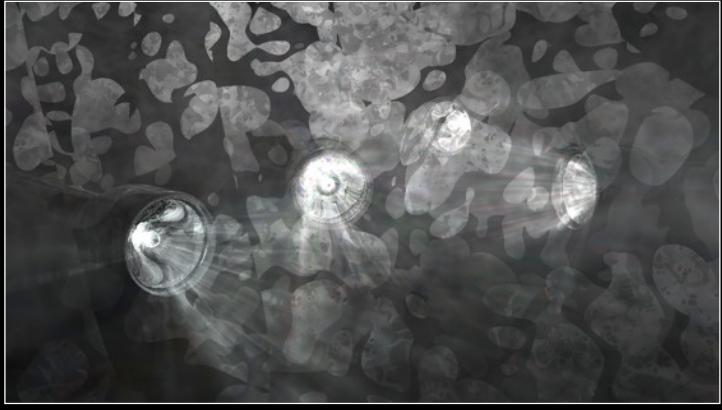
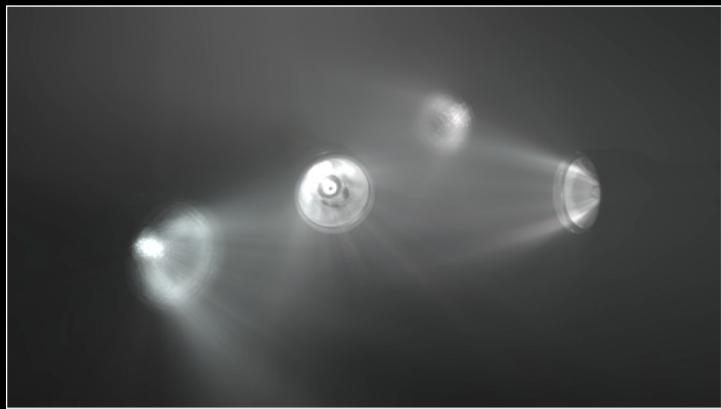


Thursday, August 23, 12

Pass 32

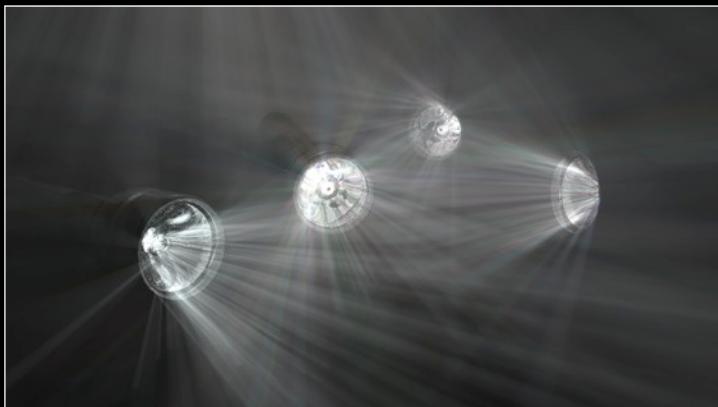


Average of Passes 1..32

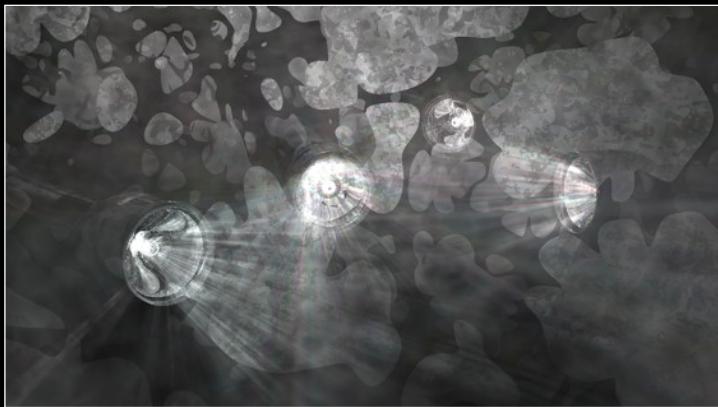


Thursday, August 23, 12

Pass 64

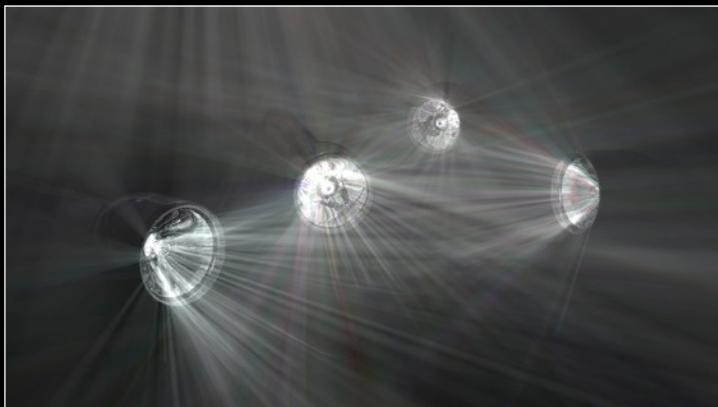


Average of Passes 1..64

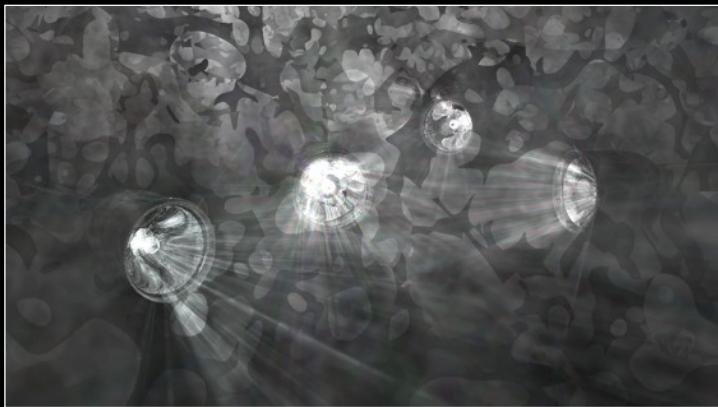
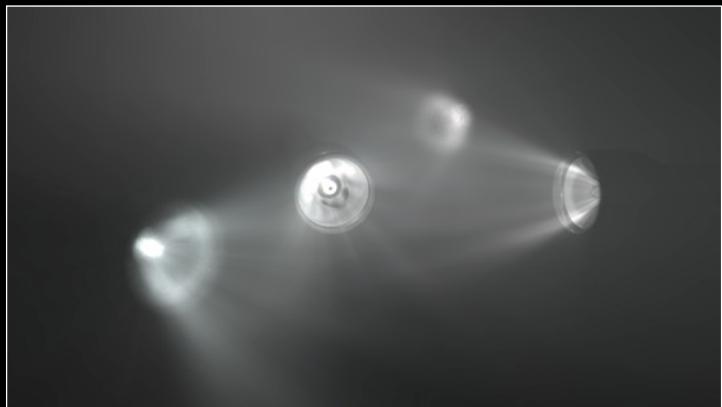


Thursday, August 23, 12

Pass 128



Average of Passes 1..128

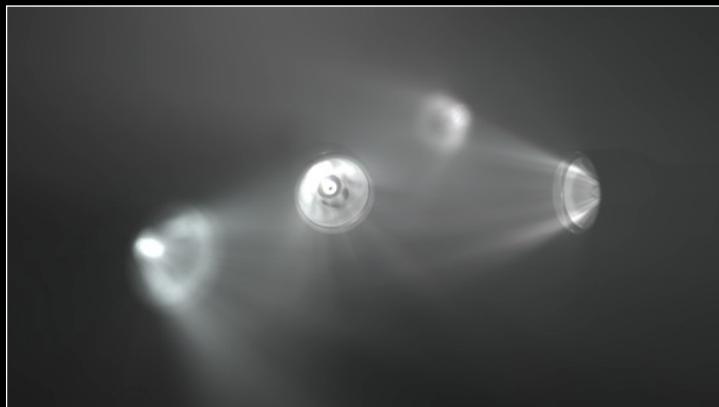


Thursday, August 23, 12

Pass 256



Average of Passes 1..256

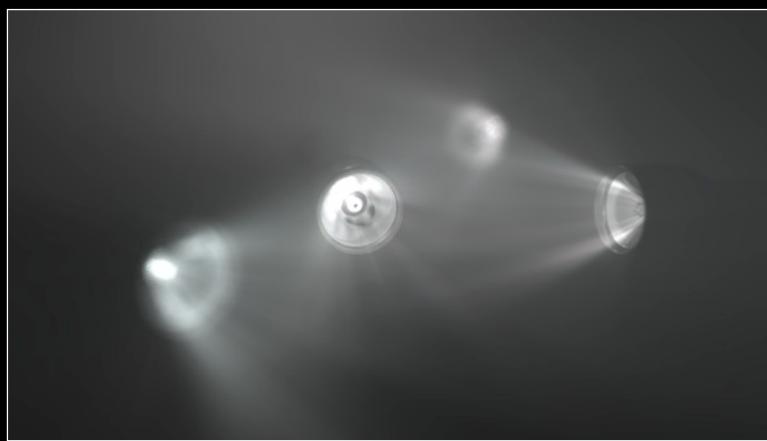


Thursday, August 23, 12

FLASHLIGHTS

1280x720, Depth-of-Field

Homogeneous
2.1M Photon Beams
8 minutes



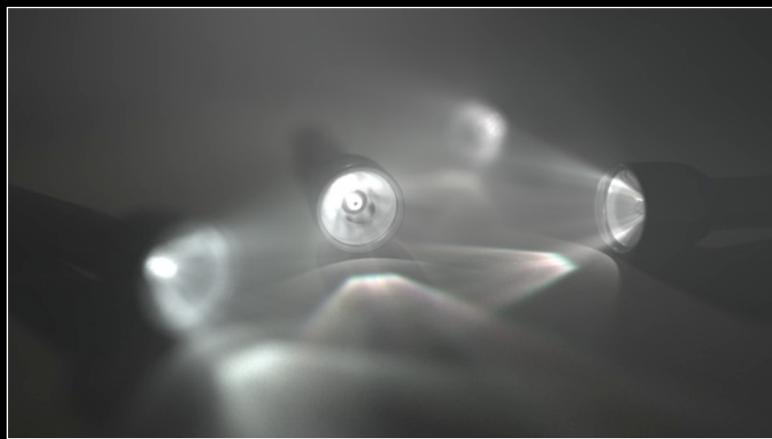
Heterogeneous
2.1M Photon Beams
10.8 minutes



FLASHLIGHTS

1280x720, Depth-of-Field

Homogeneous
2.1M Photon Beams
8 minutes



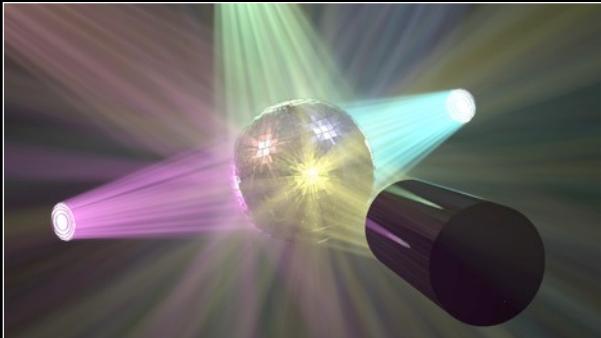
Heterogeneous
2.1M Photon Beams
10.8 minutes



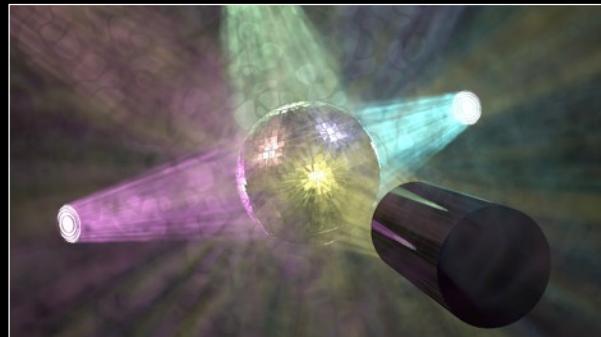
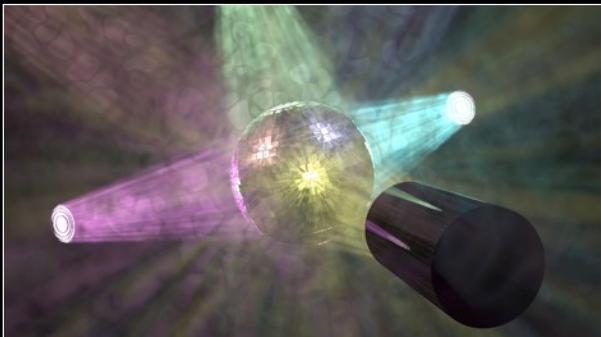
DISCO

1280x720, Depth-of-Field

Pass 1



Average of Passes 1..1

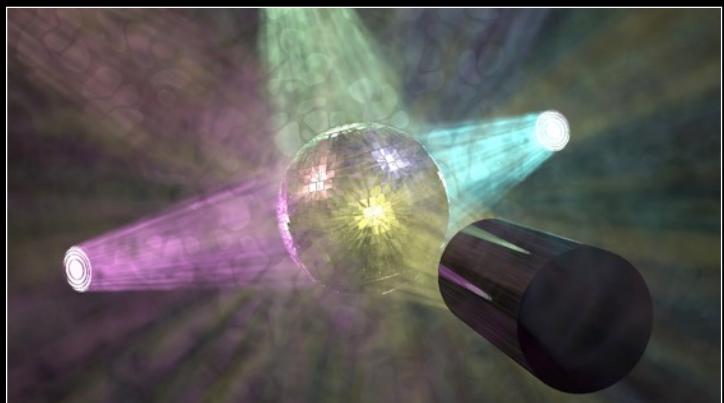
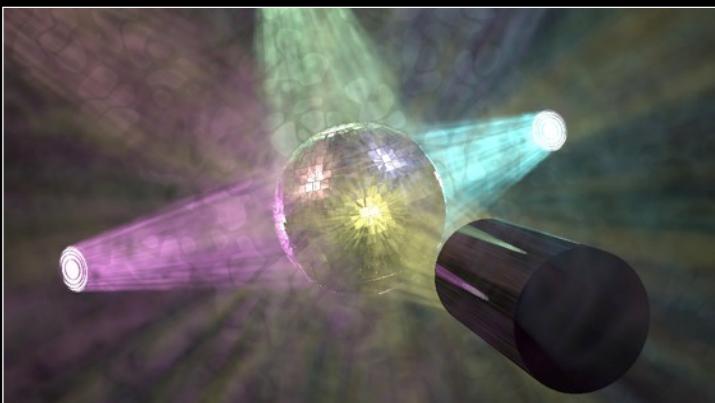


Thursday, August 23, 12

Pass 1



Average of Passes 1..1

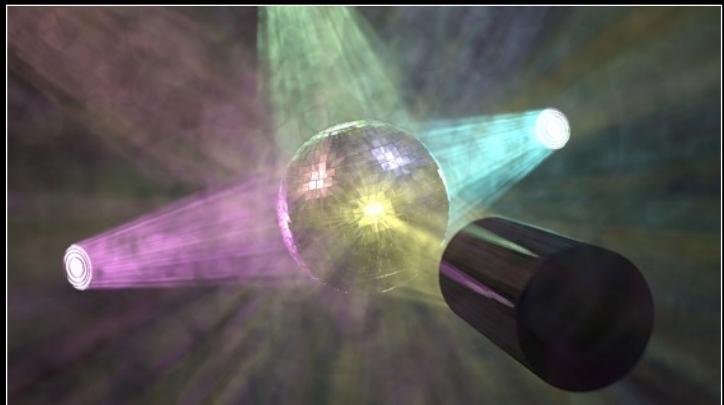


Thursday, August 23, 12

Pass 2

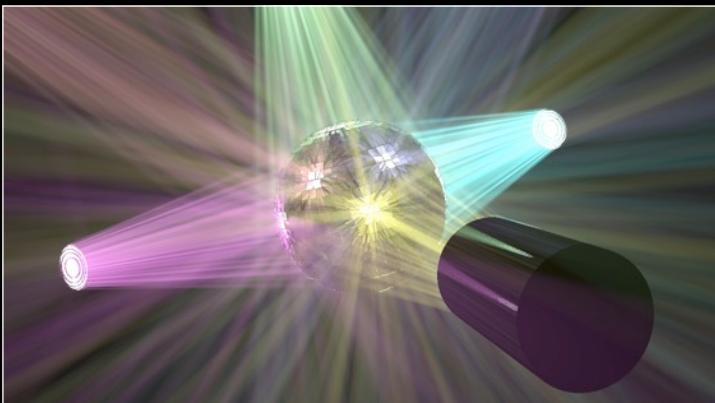


Average of Passes 1..2

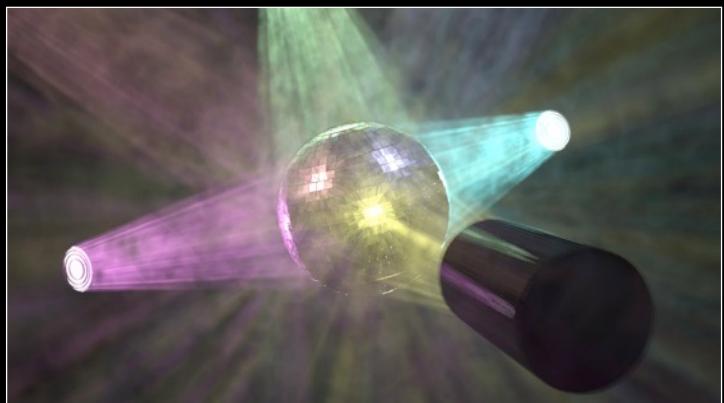


Thursday, August 23, 12

Pass 4



Average of Passes 1..4

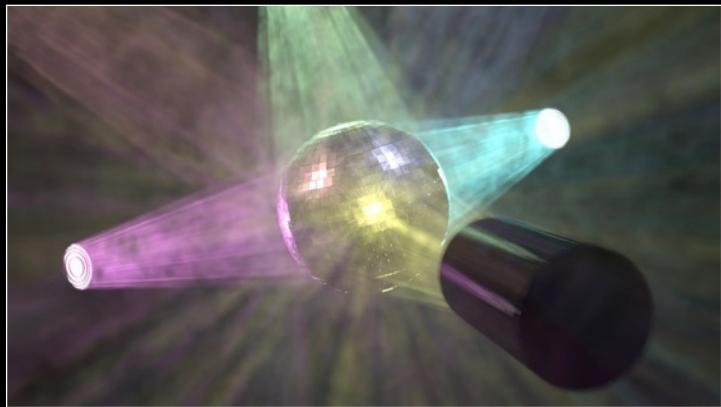
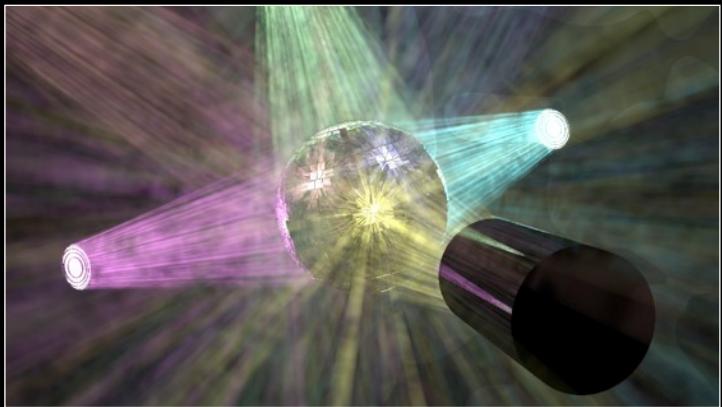


Thursday, August 23, 12

Pass 8



Average of Passes 1..8

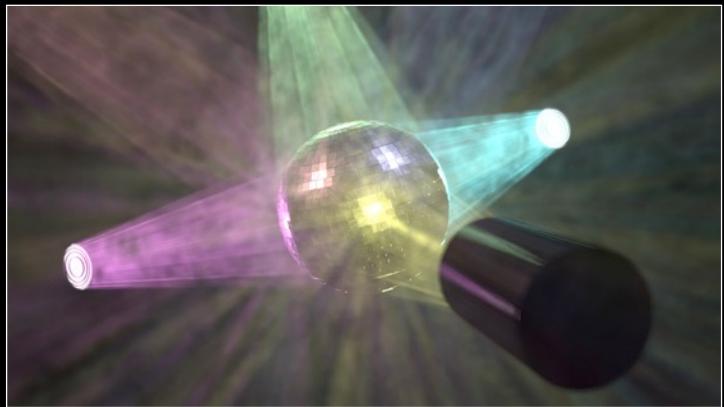
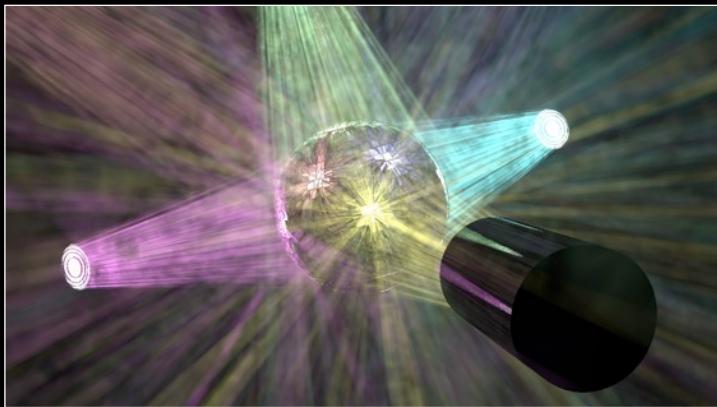


Thursday, August 23, 12

Pass 16

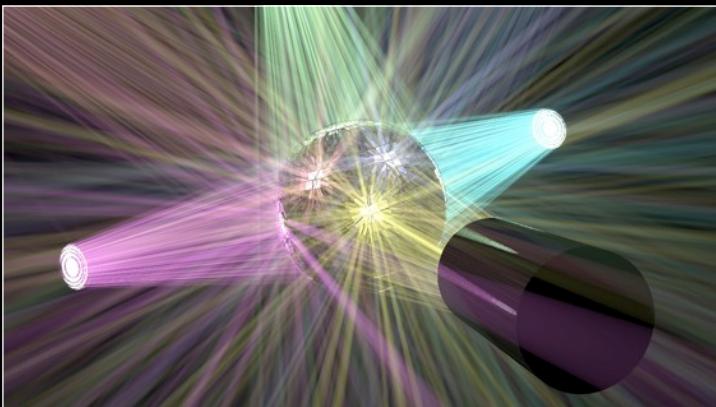


Average of Passes 1..16

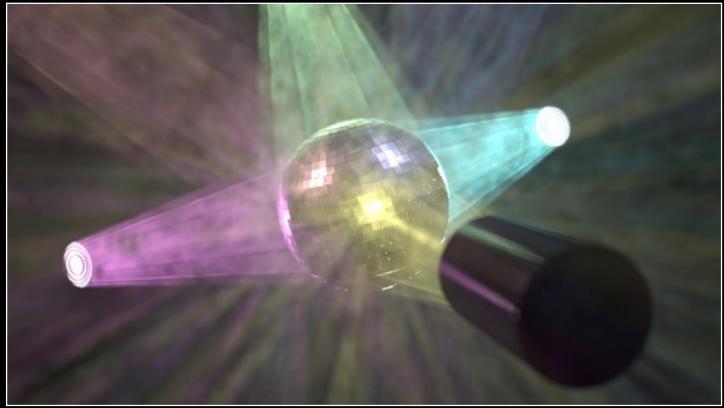


Thursday, August 23, 12

Pass 32

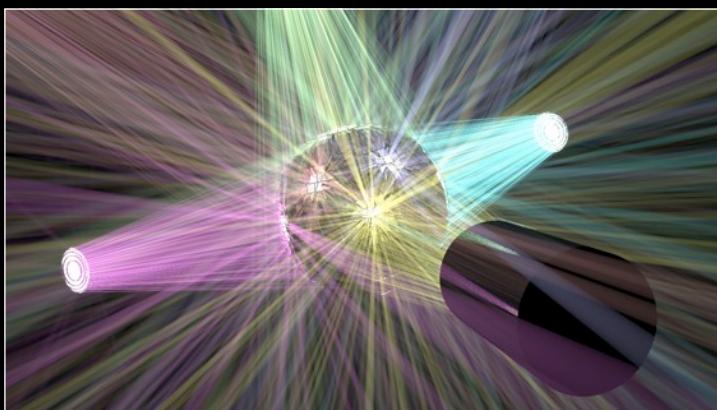


Average of Passes 1..32

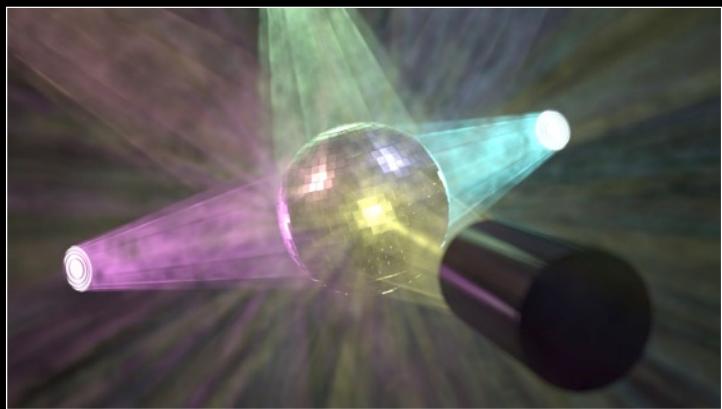
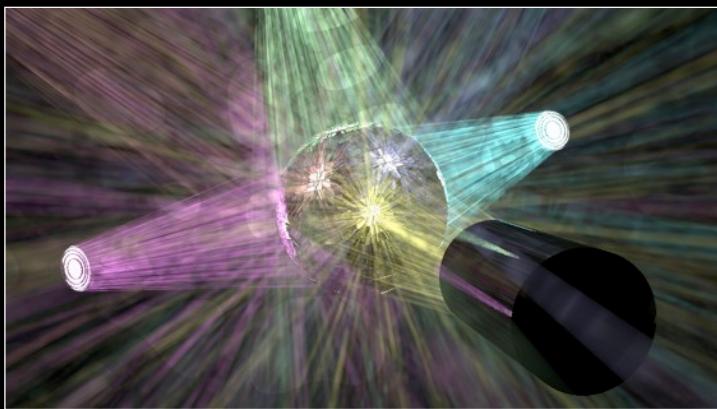


Thursday, August 23, 12

Pass 64

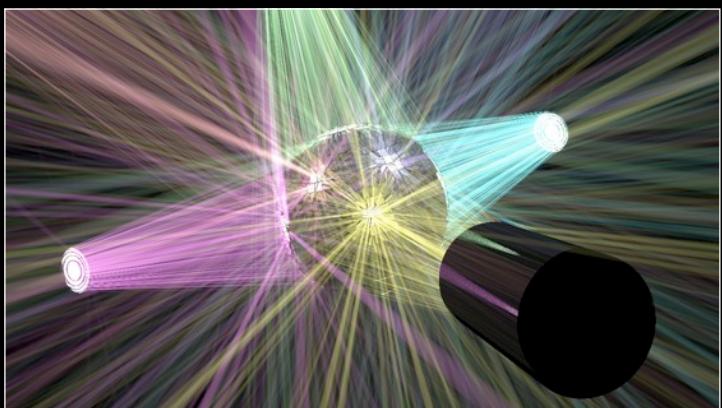


Average of Passes 1..64

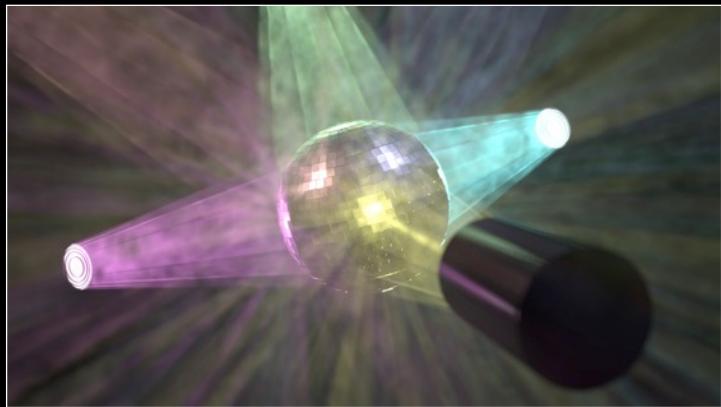
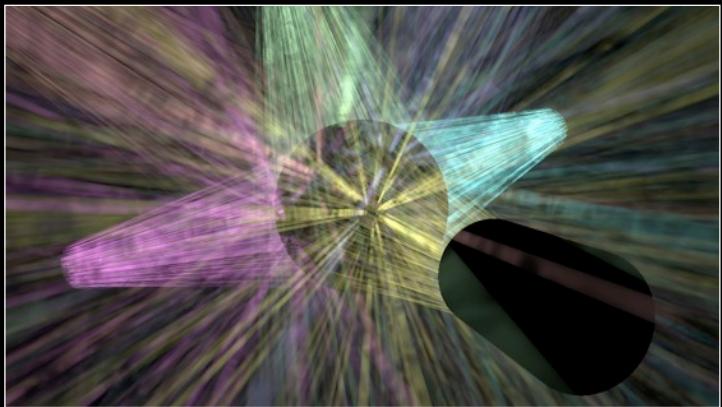
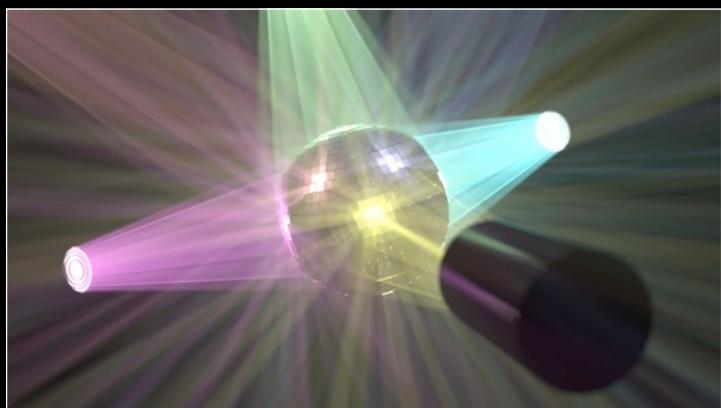


Thursday, August 23, 12

Pass 128

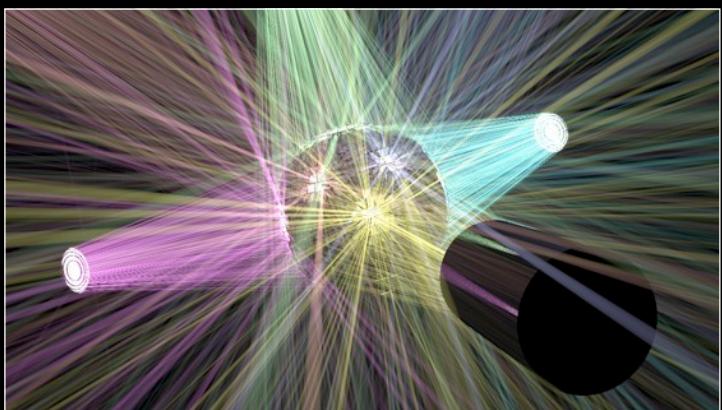


Average of Passes 1..128

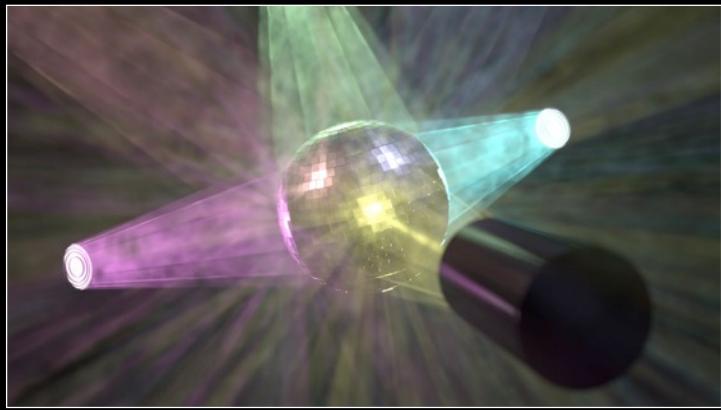
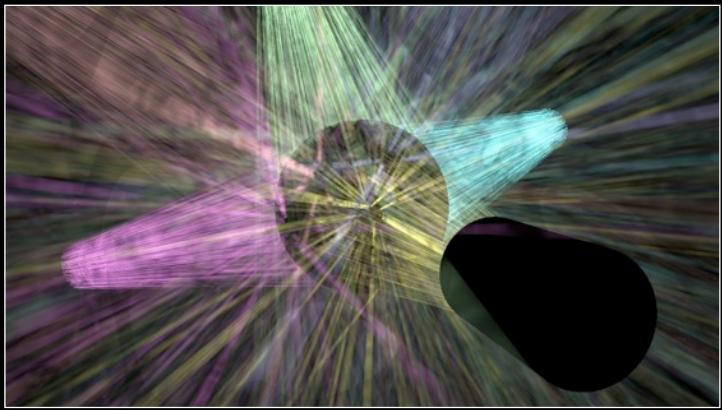
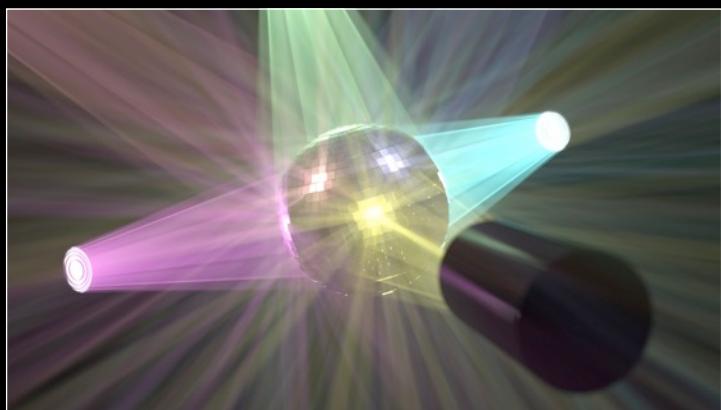


Thursday, August 23, 12

Pass 256

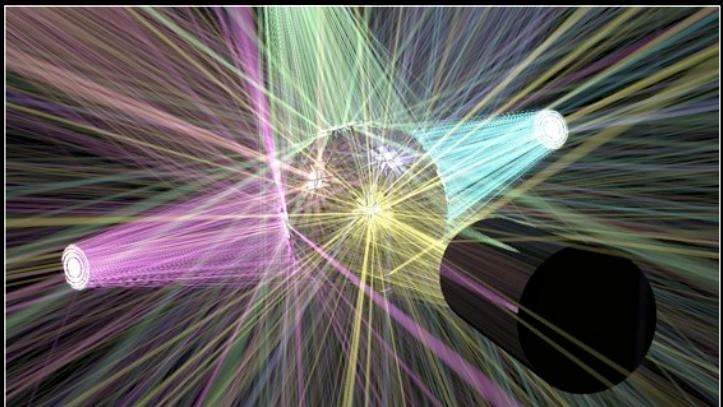


Average of Passes 1..256

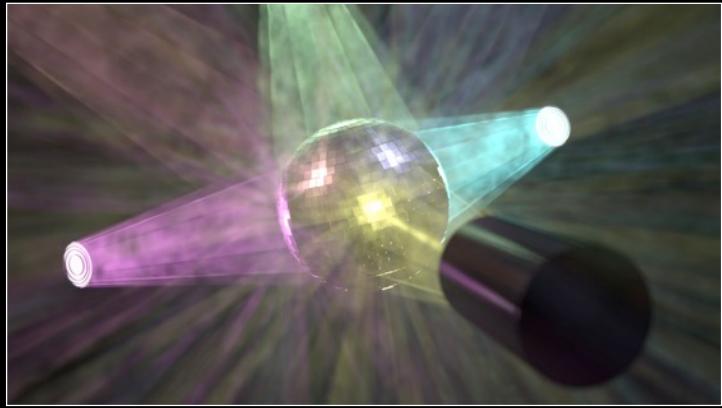
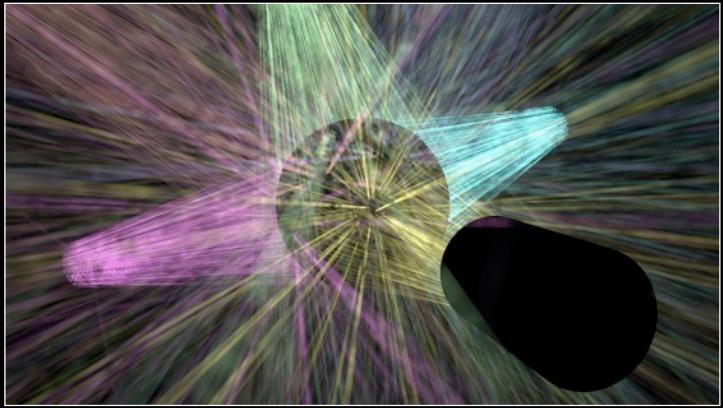
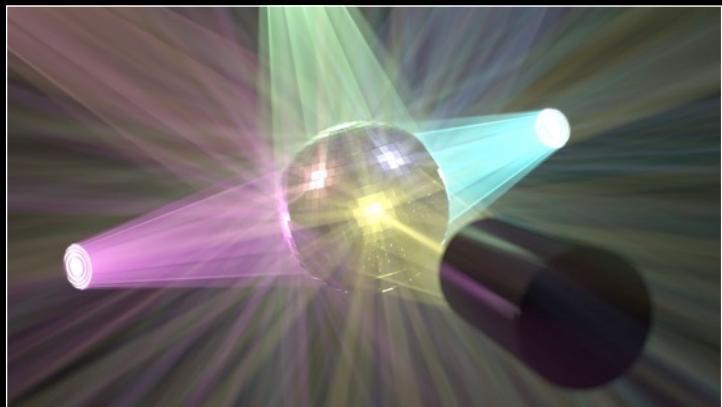


Thursday, August 23, 12

Pass 512

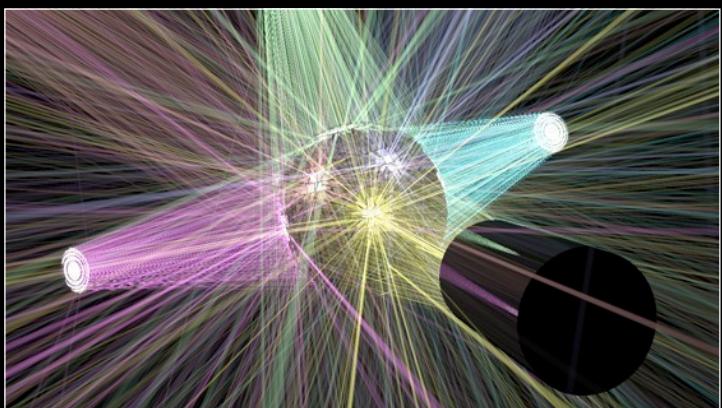


Average of Passes 1..512

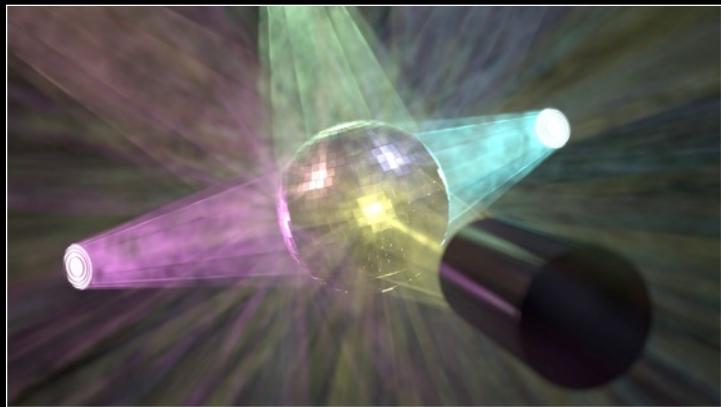
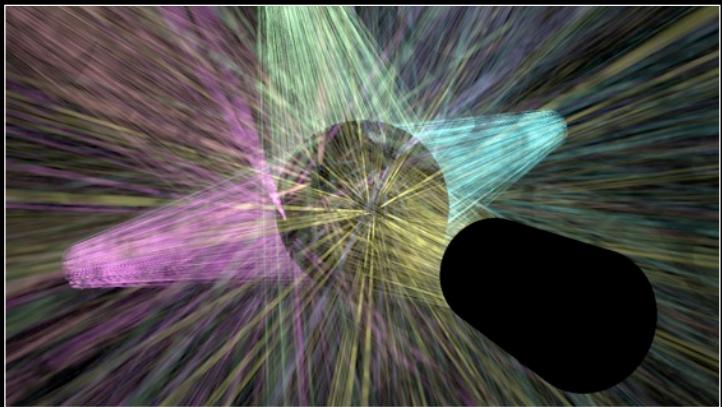
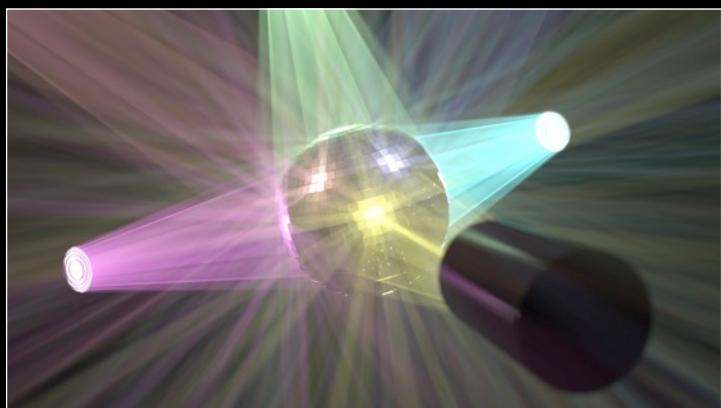


Thursday, August 23, 12

Pass 1024



Average of Passes 1..1024

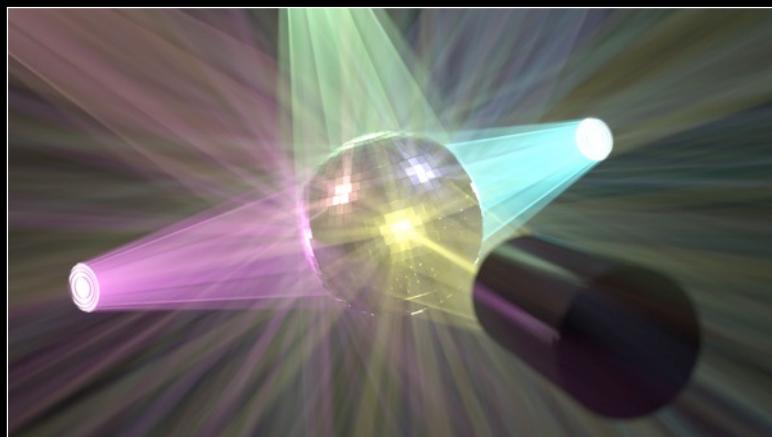


Thursday, August 23, 12

DISCO

1280x720, Depth-of-Field

Homogeneous
19.67M Photon Beams
3 minutes



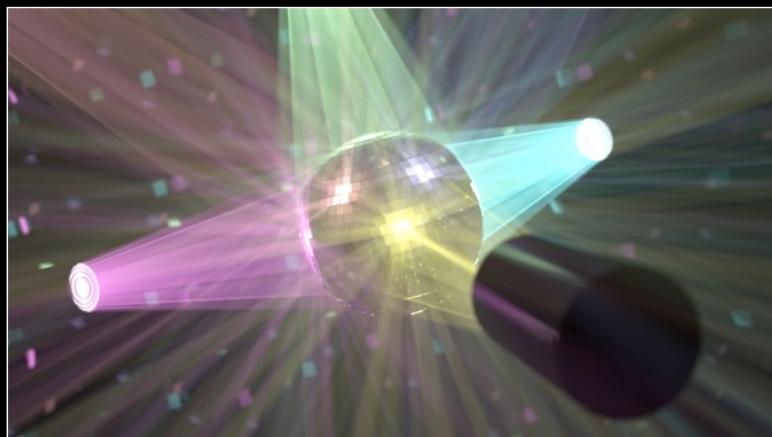
Heterogeneous
16.19M Photon Beams
5.7 minutes



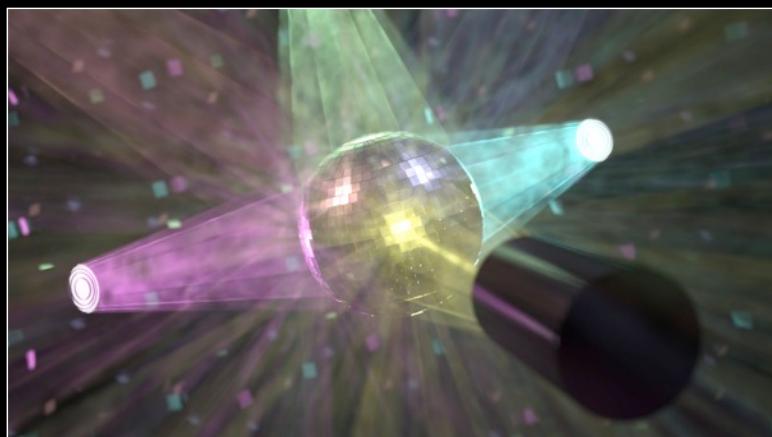
DISCO

1280x720, Depth-of-Field

Homogeneous
19.67M Photon Beams
3 minutes



Heterogeneous
16.19M Photon Beams
5.7 minutes



USER INTERACTION

Hybrid CPU/GPU Implementation

```
Pass number (GPU): 5
Pass number (CPU): 0
Photons per pass: 10000
Total render time: 0.36 s
12 fps : 82 ms (Total time between redraws)
11 fps : 60 ms (OpenGL rendering)
53 fps : 19 ms (CPU beam shooting)
876 fps : 1 ms (CPU camera tracing)
```



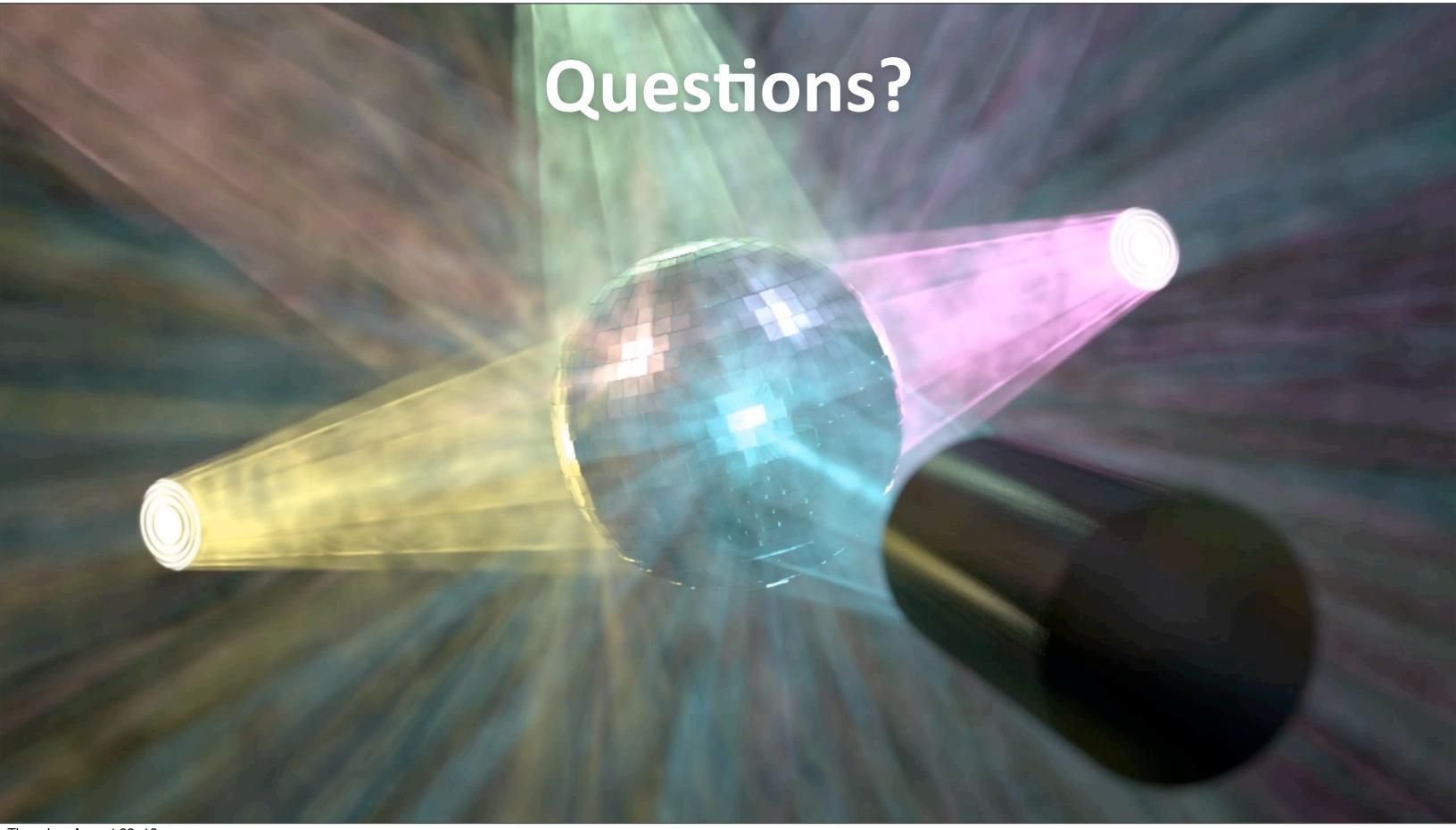
Homogeneous

```
Exposure: -0.250
Gamma: 1.414
Pass number (GPU): 22
Pass number (CPU): 22
Total render time: 5.27 s
Photons per pass: 1000
4 fps : 230 ms (Total time between redraws)
23 fps : 43 ms (OpenGL rendering)
176 fps : 6 ms (CPU beam shooting)
6 fps : 180 ms (CPU camera tracing)
```



Heterogeneous

Real-time capture



A disco ball with a reflective surface is positioned in the center, surrounded by several bright, colorful beams of light in shades of yellow, green, blue, and pink. The background is dark, making the light rays stand out.

Questions?

Beyond Photon Density Estimation

Photon Relaxation



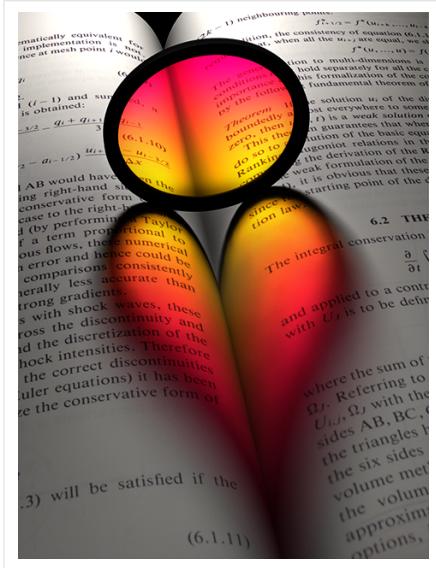
Photon Relaxation

Ben Spencer

Visual and Interactive Computing Group
Swansea University



Introduction



Photon relaxation is a contribution to the area of error minimization in photon density estimation:

- Estimate error is the sum of bias and noise.
- Goal is to reduce noise without increasing apparent bias.
- Problem often addressed at the kernel level with filters/intelligent bandwidth selection.
- Photon relaxation is different in that it directly manipulates the underlying point dataset.

Background

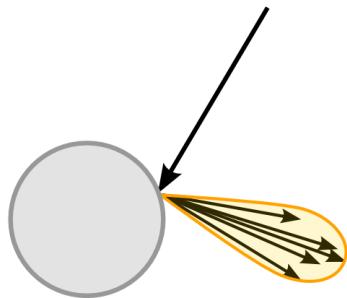
Challenges facing kernel-based noise removal:

- Tricky to preserve high-frequency detail – particularly at sub-kernel scales – while ensuring adequate smoothing; noise removal and bias often correlated.
- Wide bandwidths required to effectively filter all-frequency noise - increases rendering cost.

Photon relaxation addresses both of these points. Salient features of caustics can be preserved on a fine scale while allowing noise removal on a broader scale due to diffusion.

Furthermore, the relaxed distribution allows the use of very low-bandwidth kernels.

Background

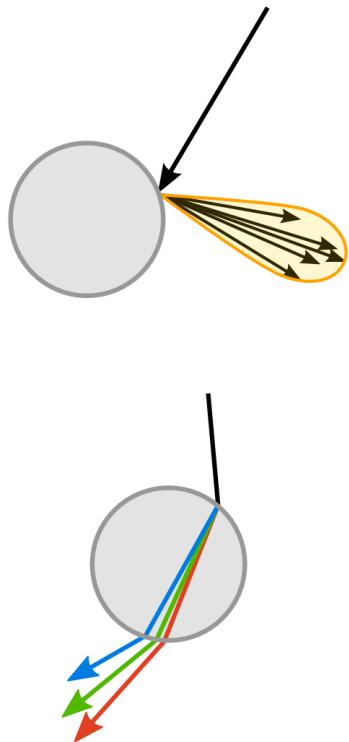


What causes noise?

Two factors:

- Point discrepancy. Caused by stochastic processes (e.g. scattering) and photon decoherence (geometry) during the particle tracing step.

Background

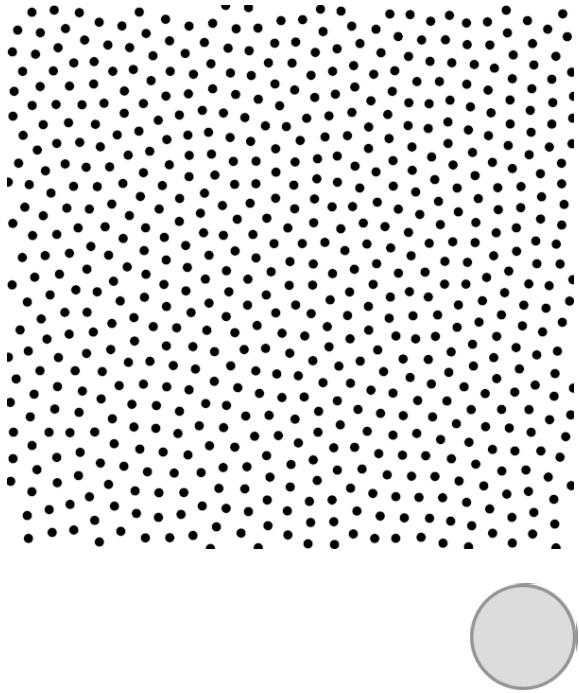


What causes noise?

Two factors:

- Point discrepancy. Caused by stochastic processes (e.g. scattering) and photon decoherence (geometry) during the particle tracing step.
- Variance in photon flux. This can be caused by absorption, attenuation, dispersion through dielectric media, etc.

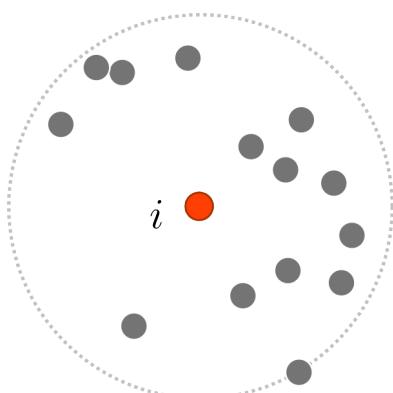
Photon Relaxation



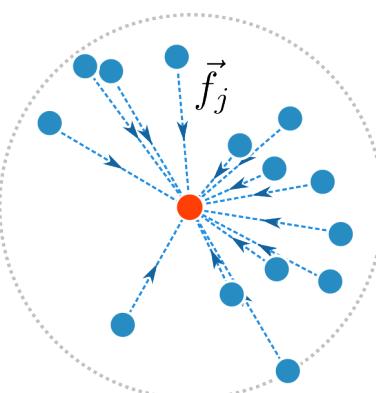
Basic principles:

- Use point repulsion to minimize local discrepancy.

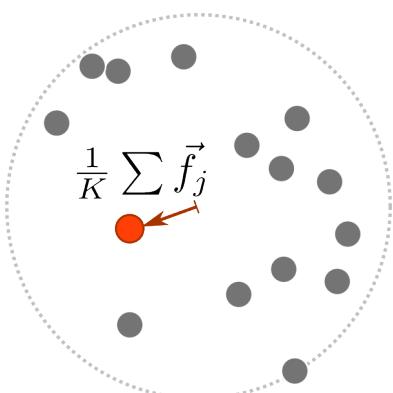
Photon Relaxation



1. For each photon, x ,
gather K-nearest neighbours
to x .



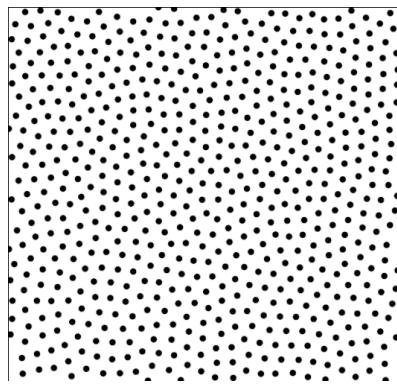
2. Compute individual
repulsive forces on x from
members of K .



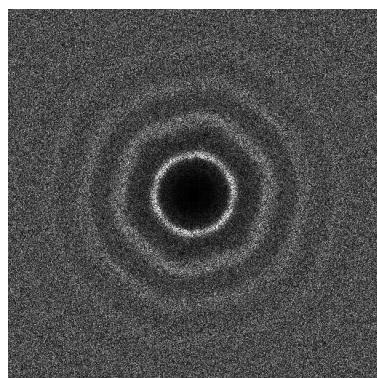
3. Apply mean of forces to
position of x .

4. Repeat.

Photon Relaxation



Spatial domain

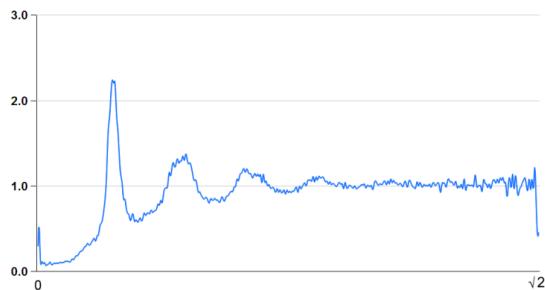


Frequency domain

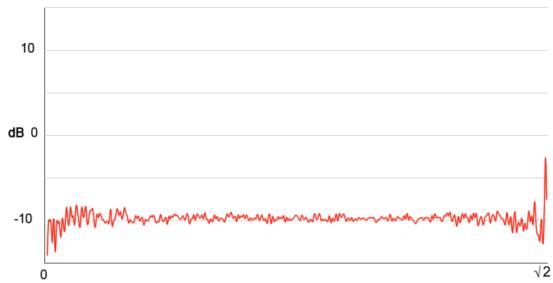
Basic principles:

- Use point repulsion to minimize local discrepancy.
- Aim to relax distribution so it exhibits a blue noise spectral signature and low angular anisotropy.

Photon Relaxation



Radially-averaged power spectrum

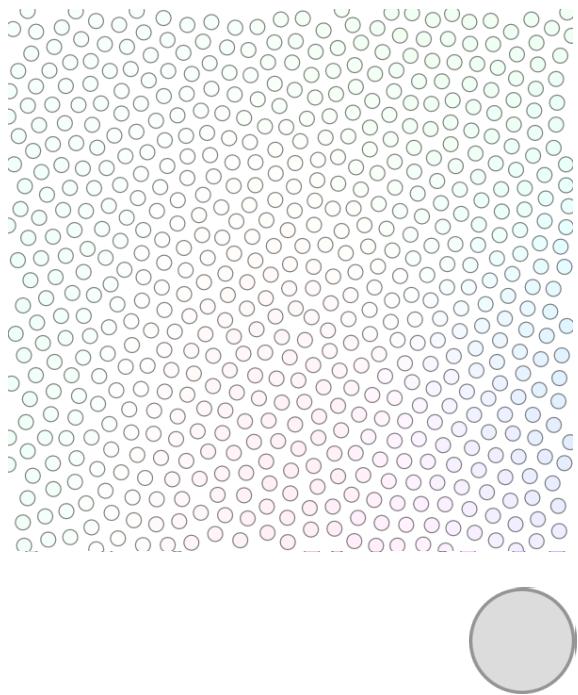


Angular anisotropy

Basic principles:

- Use point repulsion to minimize local discrepancy.
- Aim to relax distribution so it exhibits a blue noise spectral signature and low angular anisotropy.

Photon Relaxation



Basic principles:

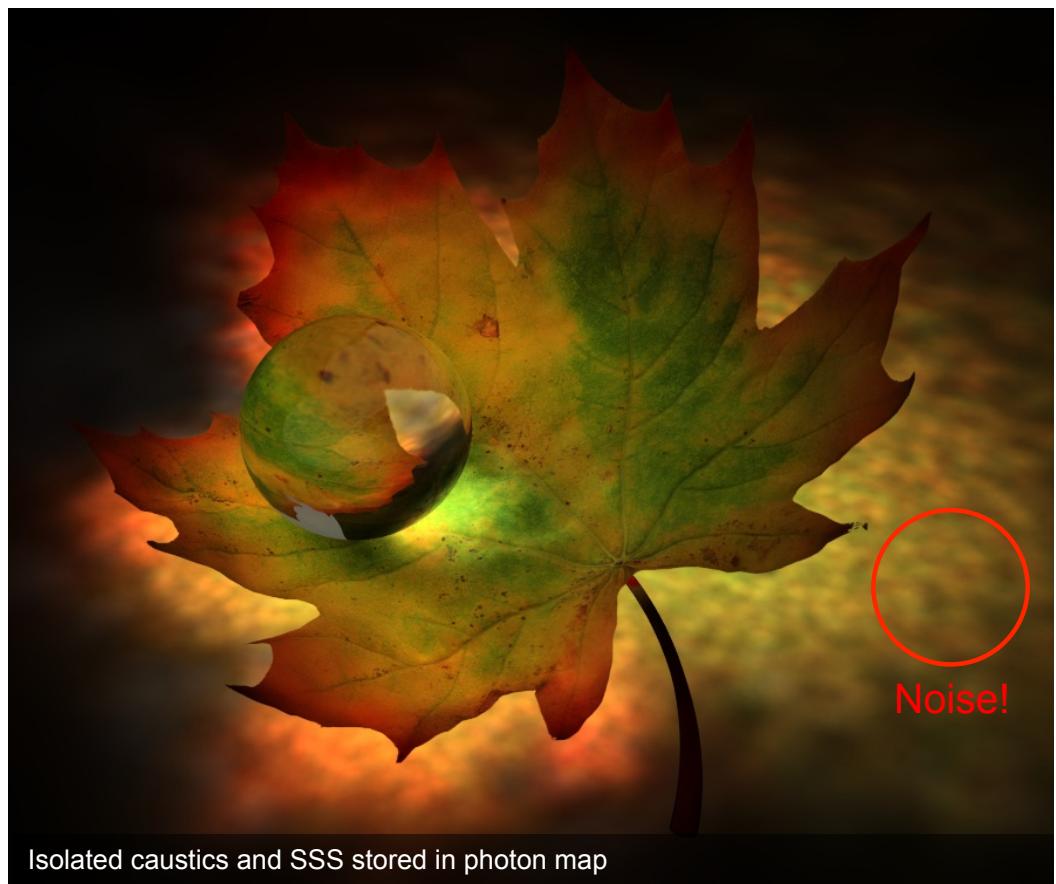
- Use point repulsion to minimize local discrepancy.
- Aim to relax distribution so it exhibits a blue noise spectral signature and low angular anisotropy.
- Use diffusion to homogenize flux between photons.

Photon Relaxation



No caustics or sub-surface scattering

Photon Relaxation



Photon Relaxation



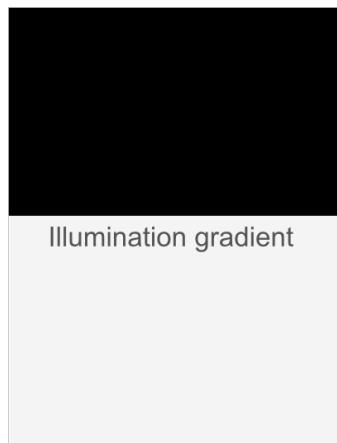
Photon map relaxed by 20 iterations

Photon Relaxation

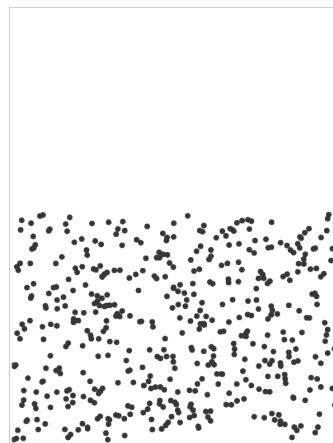


Feature Detection and Preservation

Relaxation removes noise effectively, but photon diffusion also degrades larger-scale features of the distribution.



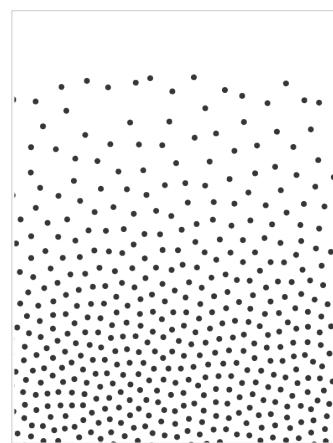
Sample PDF



Stochastically seeded
photon distribution



Relax



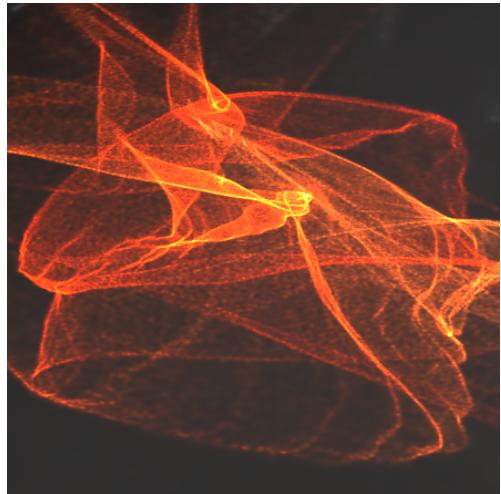
Diffusion results in
degraded
reconstruction

Feature Detection and Preservation

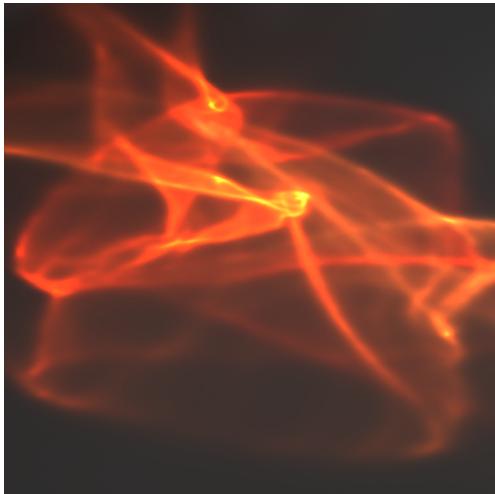


Cognac Redux

Feature Detection and Preservation



Before relaxation



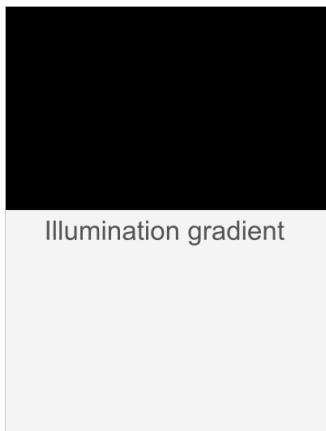
After relaxation.
High-frequency detail has been
lost due to diffusion.



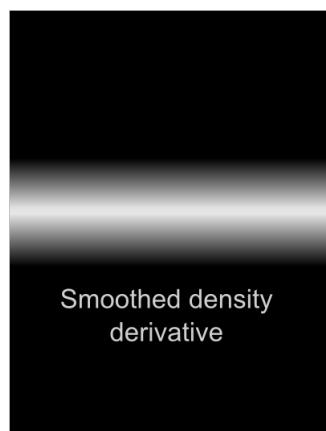
Can we do better?

Feature Detection and Preservation

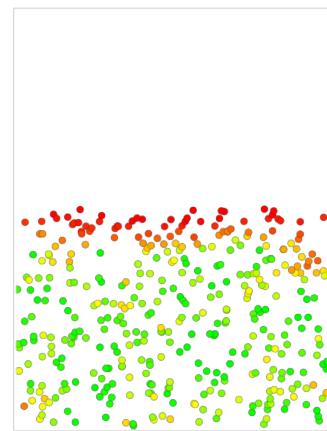
Feature detection aims to preserve these features by detecting and inhibiting motion in the direction of migration.



Illumination gradient

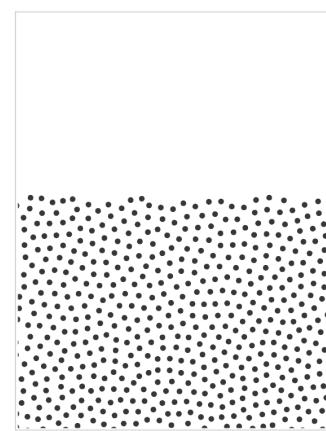


Smoothed density derivative



Sample PDF

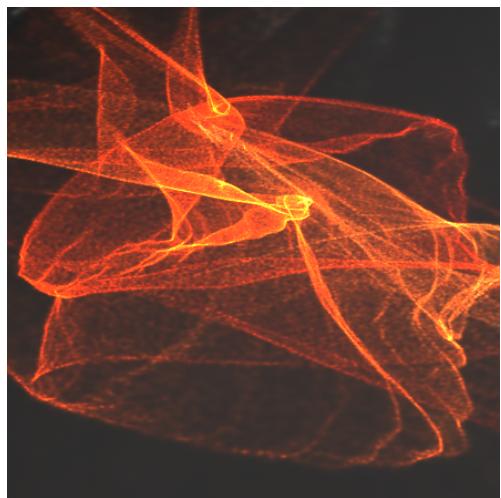
Photons migrate along direction of density derivative.



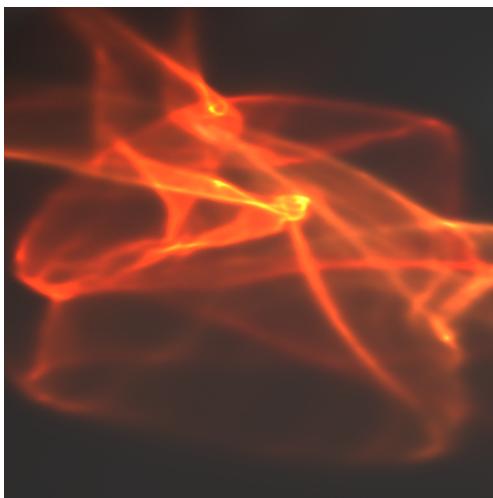
Reconstructed distribution better preserves PDF.

Feature detection constrains photons in the direction of migration (red = constrained).

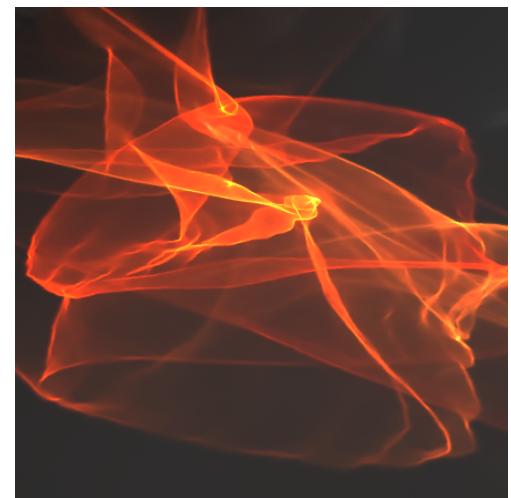
Feature Detection and Preservation



Before relaxation



Relaxation with *no* constraints



Relaxation *with* constraints



Thank you!

Progressive Expectation–Maximization

Progressive Expectation–Maximization for Hierarchical Volumetric Photon Mapping

Wenzel Jakob^{1,2} Christian Regg^{1,3} Wojciech Jarosz¹



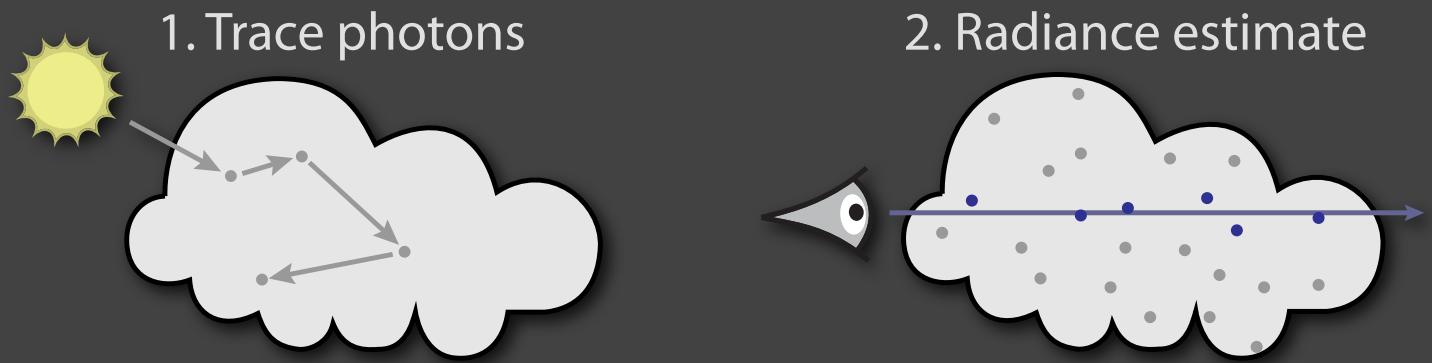
¹ Disney Research, Zürich

² Cornell University

³ ETH Zürich

Motivation

Volumetric photon mapping



Issues

- high-frequency illumination requires *many* photons
- time spent on photons that contribute very little
- prone to temporal flickering

Motivation



Beam radiance estimate : 917K photons



Per-pixel
render time

Motivation

Beam radiance estimate : 917K photons



Render time: 281 s



Per-pixel
render time

Our method: 4K Gaussians



Render time: 125 s



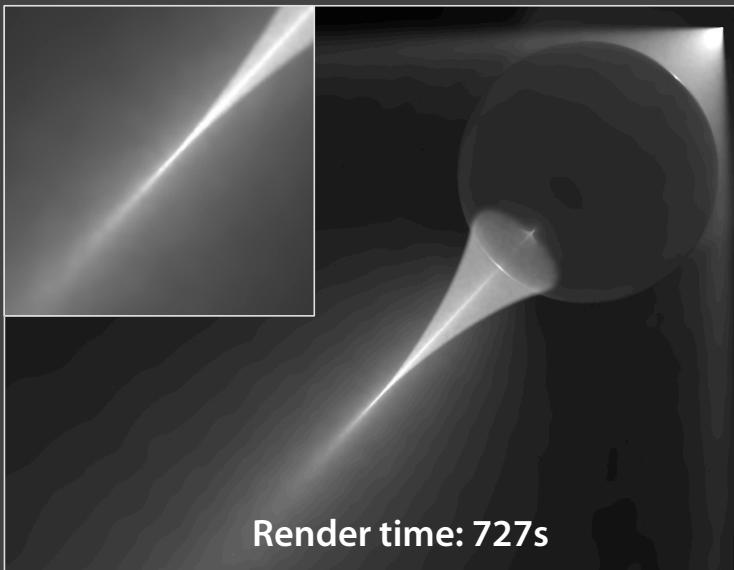
Per-pixel
render time

Our approach:

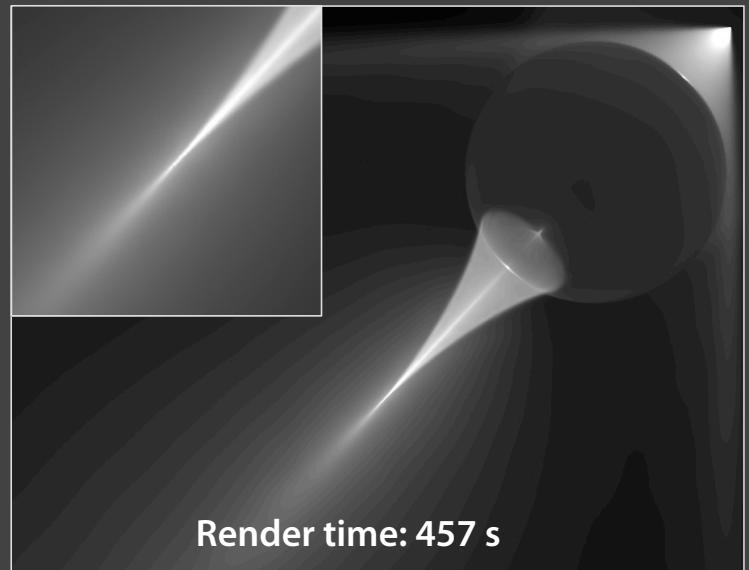
- represent radiance using a Gaussian mixture model (**GMM**)
- fit using progressive expectation maximization (**EM**)
- render with multiple levels of detail

Motivation

Beam radiance estimate : 4M photons



Our method: 16K Gaussians



Our approach:

- represent radiance using a Gaussian mixture model (**GMM**)
- fit using progressive expectation maximization (**EM**)
- render with multiple levels of detail

Related work

- Diffusion based photon mapping
[Schjøth et al. 08]



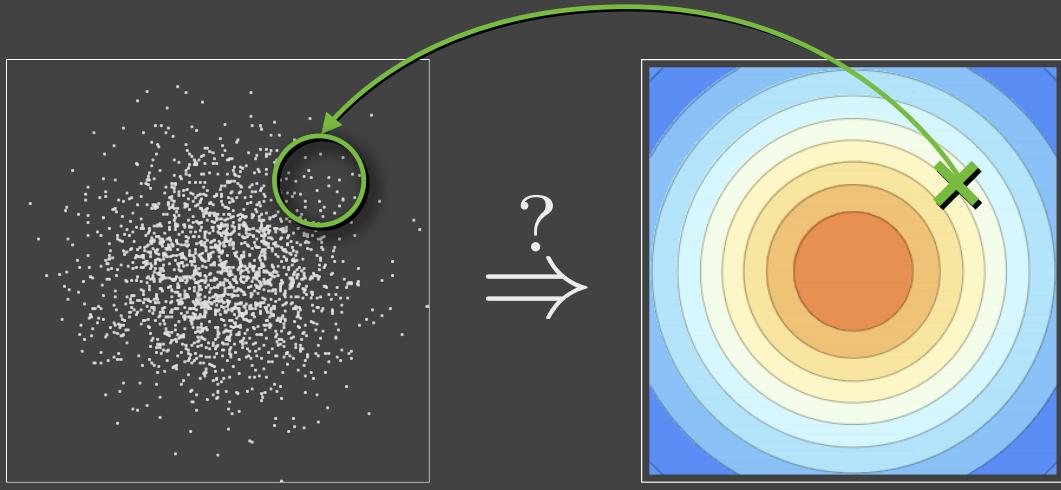
- Photon relaxation
[Spencer et al. 09]



- Hierarchical photon mapping
[Spencer et al. 09]



Density estimation



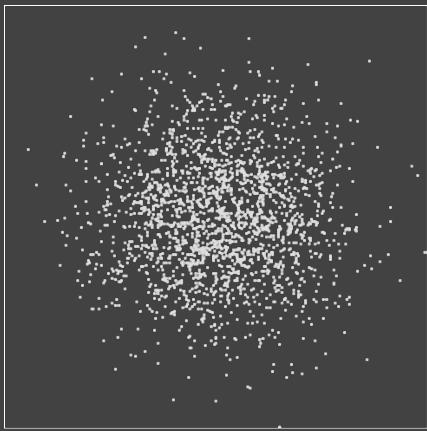
Given photons
 x_1, x_2, \dots

approximately determine
their density f

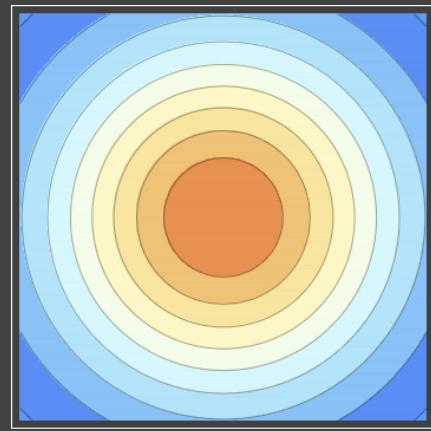
Nonparametric:

- Count the number of photons within a small region

Density estimation



Given photons
 x_1, x_2, \dots



approximately determine
their density f

Nonparametric:

- Count the number of photons within a small region

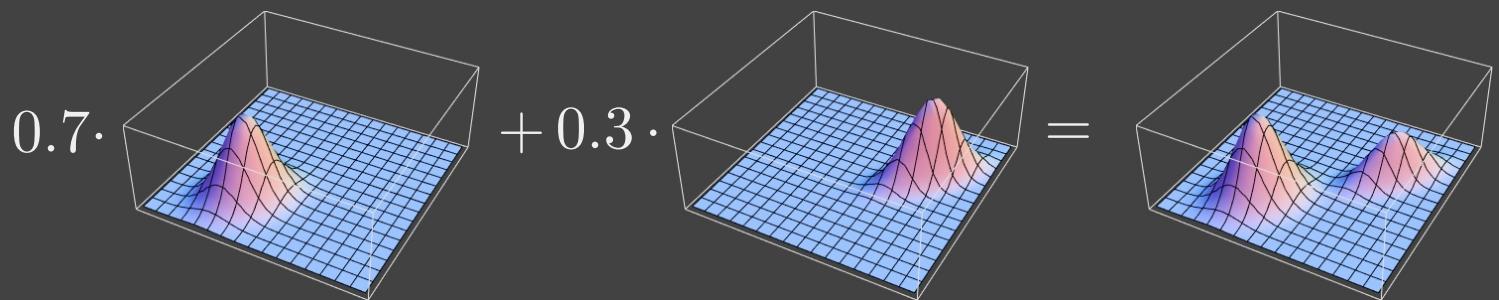
Parametric:

- Find suitable parameters for a **known** distribution

Gaussian mixture models

- Photon density modeled as a weighted sum of Gaussians:

$$f(\mathbf{x} \mid \Theta) = \sum_{i=1}^k w_i g(\mathbf{x} \mid \Theta_i)$$

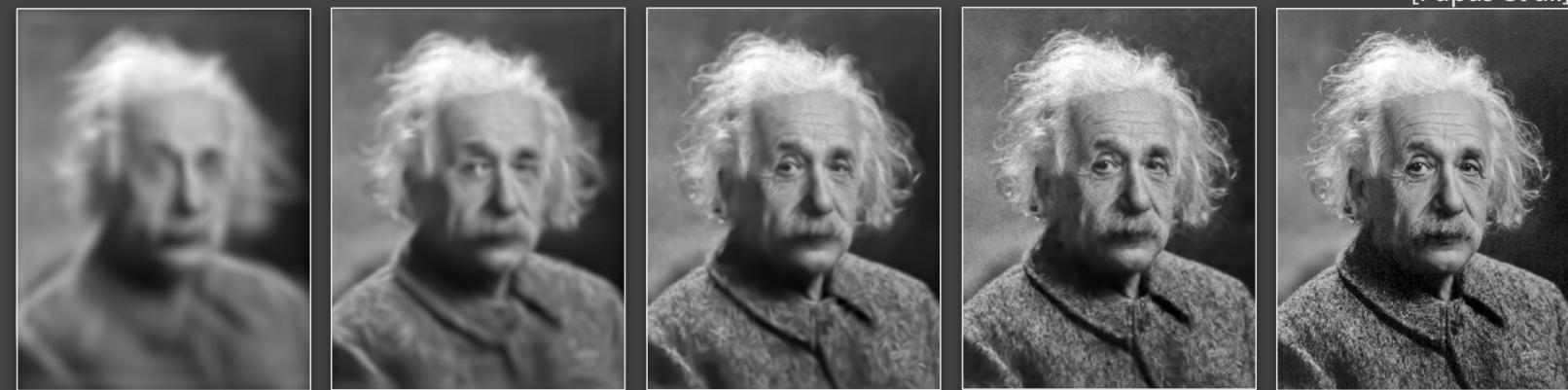


Gaussian mixture models

- Photon density modeled as a weighted sum of Gaussians:

$$f(\mathbf{x} \mid \Theta) = \sum_{i=1}^k w_i g(\mathbf{x} \mid \Theta_i)$$

[Papas et al.]



256 Gaussians

1024 Gaussians

4096 Gaussians

16384 Gaussians

Target density

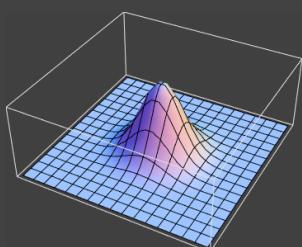
Gaussian mixture models

- Photon density modeled as a weighted sum of Gaussians:

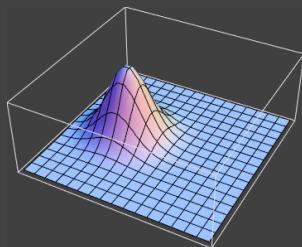
$$f(\mathbf{x} \mid \Theta) = \sum_{i=1}^k w_i g(\mathbf{x} \mid \Theta_i)$$

Unknown parameters Θ :

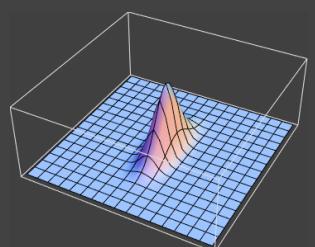
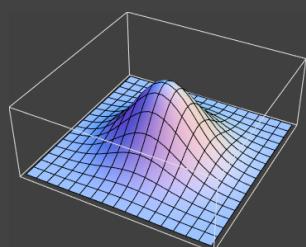
1. Weights



2. Means



3. Covariance matrices



Maximum likelihood estimation

Approach: find the “most likely” parameters, i.e.

$$\Theta^* := \underset{\Theta}{\operatorname{argmax}} \prod_{i=1}^n f(x_i \mid \Theta)$$

↑ ↓
Estimated parameters Mixture model
 Photon locations

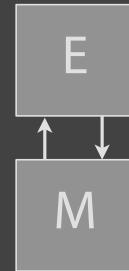
→ **Expectation maximization**

Expectation maximization

- Two components:

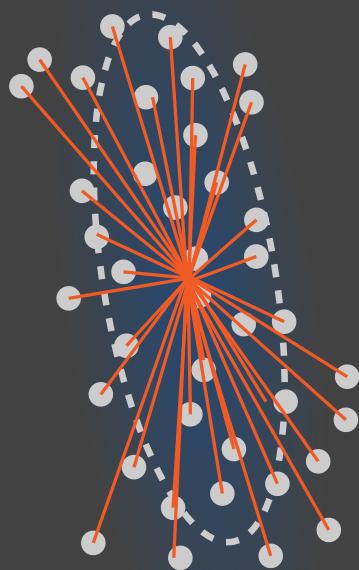
E-Step: establish soft assignment between photons and Gaussians

M-Step: maximize the expected likelihood



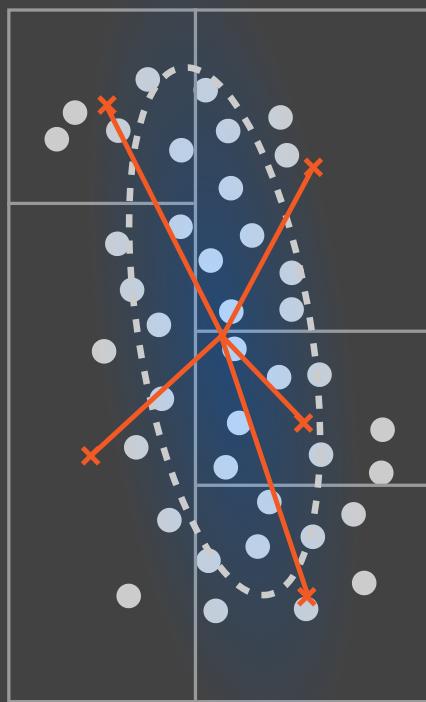
- Finds a locally optimal solution
→ good starting guess needed!
- Slow and scales poorly — $\mathcal{O}(n^2)$
(where n : photon count)

Expectation maximization



Accelerated EM by [Verbeek et al. 06]

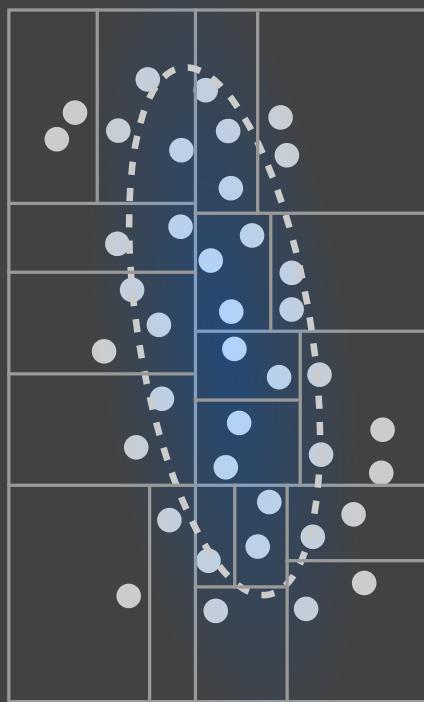
Accelerated EM



Stored cell statistics:

- photon count
- mean position
- average outer product

Progressive EM



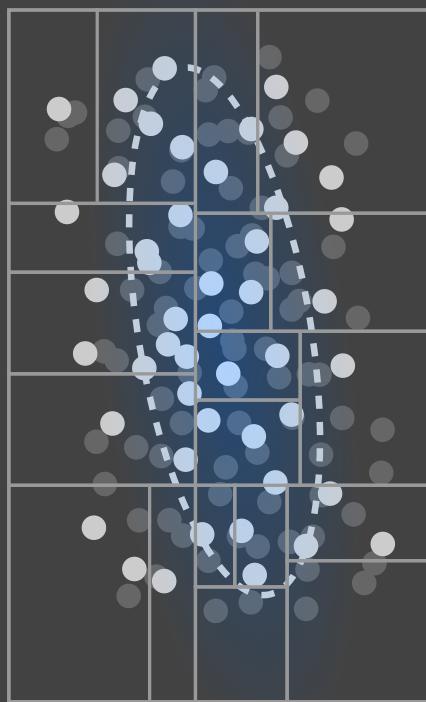
Stored cell statistics:

- photon count
- mean position
- average outer product

Our modifications:

- better cell refinement

Progressive EM



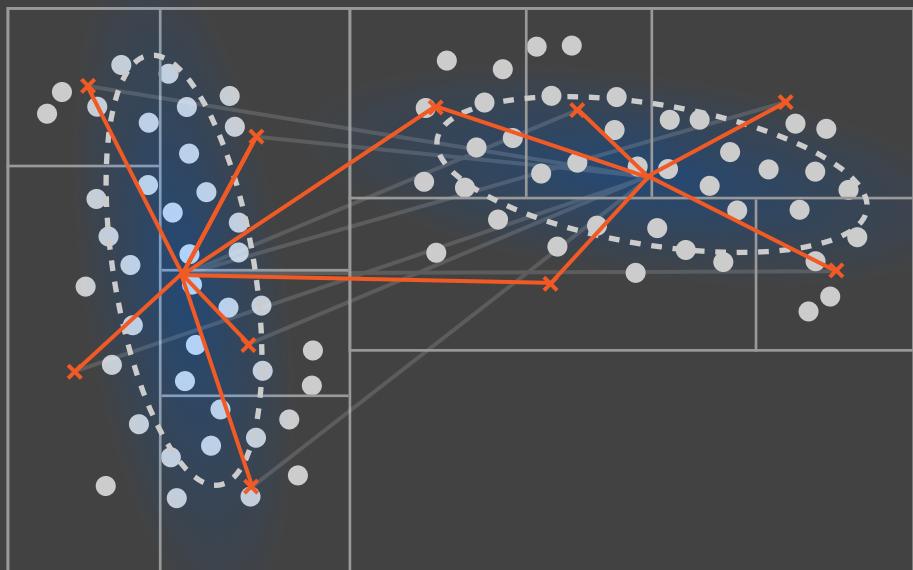
Stored cell statistics:

- photon count
- mean position
- average outer product

Our modifications:

- better cell refinement
- progressive photons shooting passes

Progressive EM



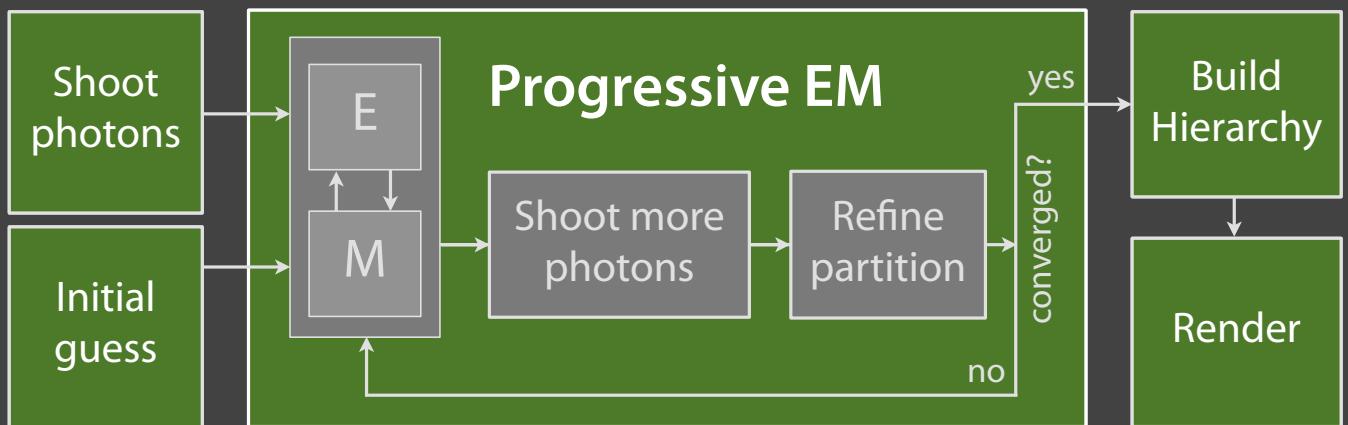
Stored cell statistics:

- photon count
- mean position
- average outer product

Our modifications:

- better cell refinement
- progressive photons shooting passes
- reduced complexity
 $\mathcal{O}(n^2) \rightarrow \mathcal{O}(n \log n)$

Pipeline overview



Rendering



$$\text{pixel value} = \sum_{i=1}^k \text{contrib}(i)$$

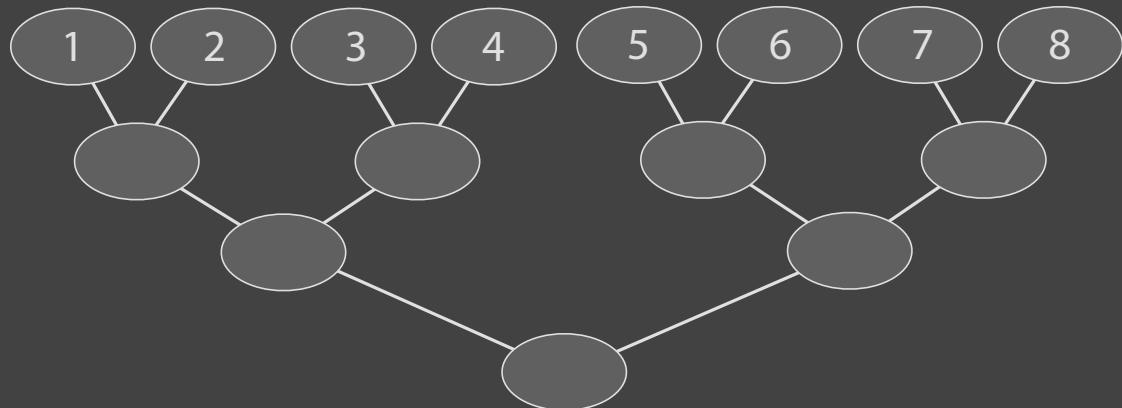
$$\text{contrib}(i) = \int_a^b g(\mathbf{r}(t) | \bar{\Theta}_i) e^{-\sigma_t t} dt = C_0 \left[\text{erf} \left(\frac{C_3 + 2C_2 b}{2\sqrt{C_2}} \right) - \text{erf} \left(\frac{C_3 + 2C_2 a}{2\sqrt{C_2}} \right) \right]$$

...

Level of detail hierarchy

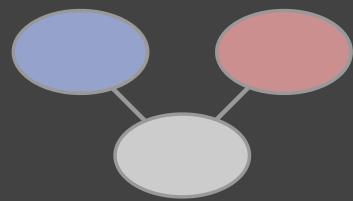
Agglomerative construction:

- Repeatedly merge nearby Gaussians based on their Kullback-Leibler divergence

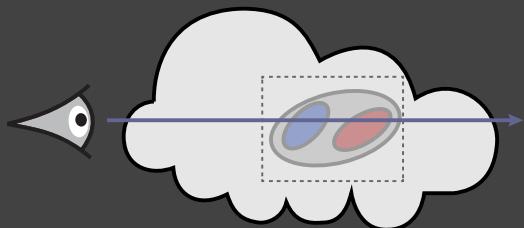


Rendering

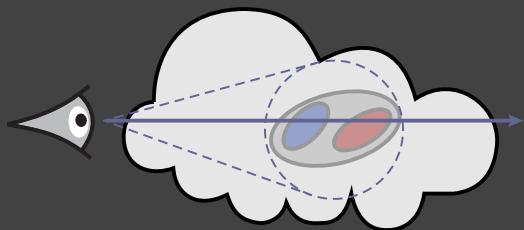
Example hierarchy:



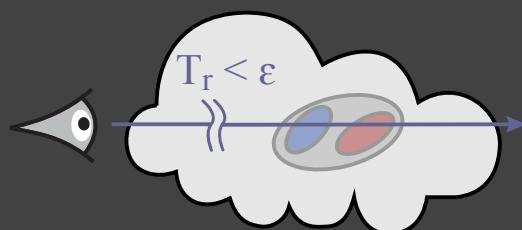
Criterion 1: bounding box intersected?



Criterion 2: solid angle large enough?



Criterion 3: attenuation low enough?





BRE: 1M Photons

$23+192 = 215 \text{ s}$

23

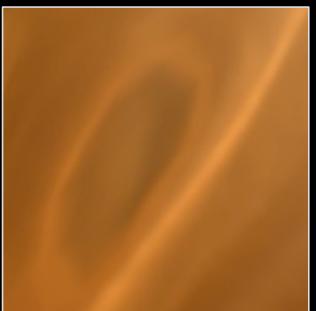
Saturday, August 4, 12



Our method: 4K Gaussians
(fit to 1M photons)

$35+24 = 59$ s
(3.6x)

24



BRE: 18M Photons

$507+609 = 1116$ s

25

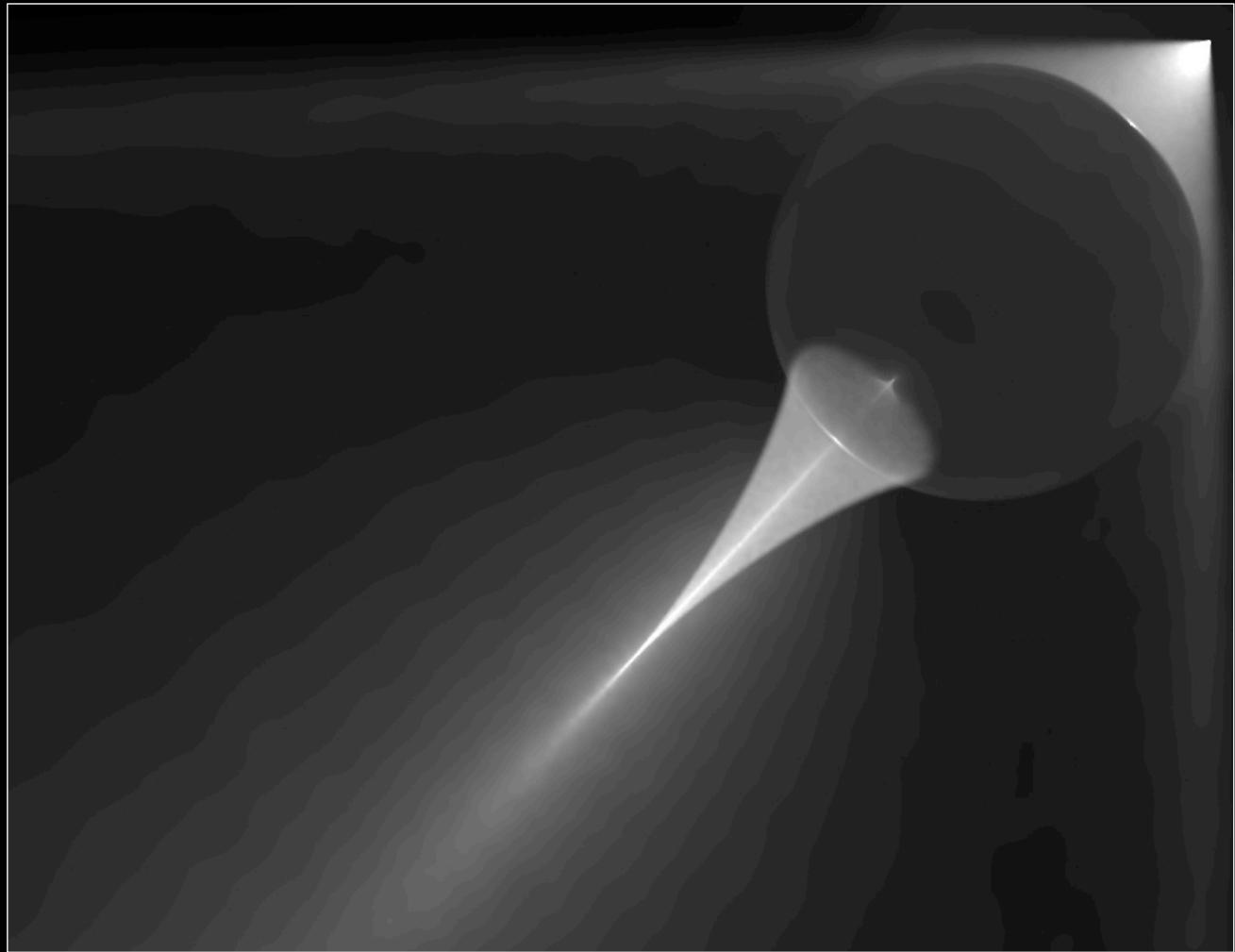
Saturday, August 4, 12



Our method: 64K Gaussians
(fit to 18M photons)

$868+66 = 934$ s
(1.2x)

26

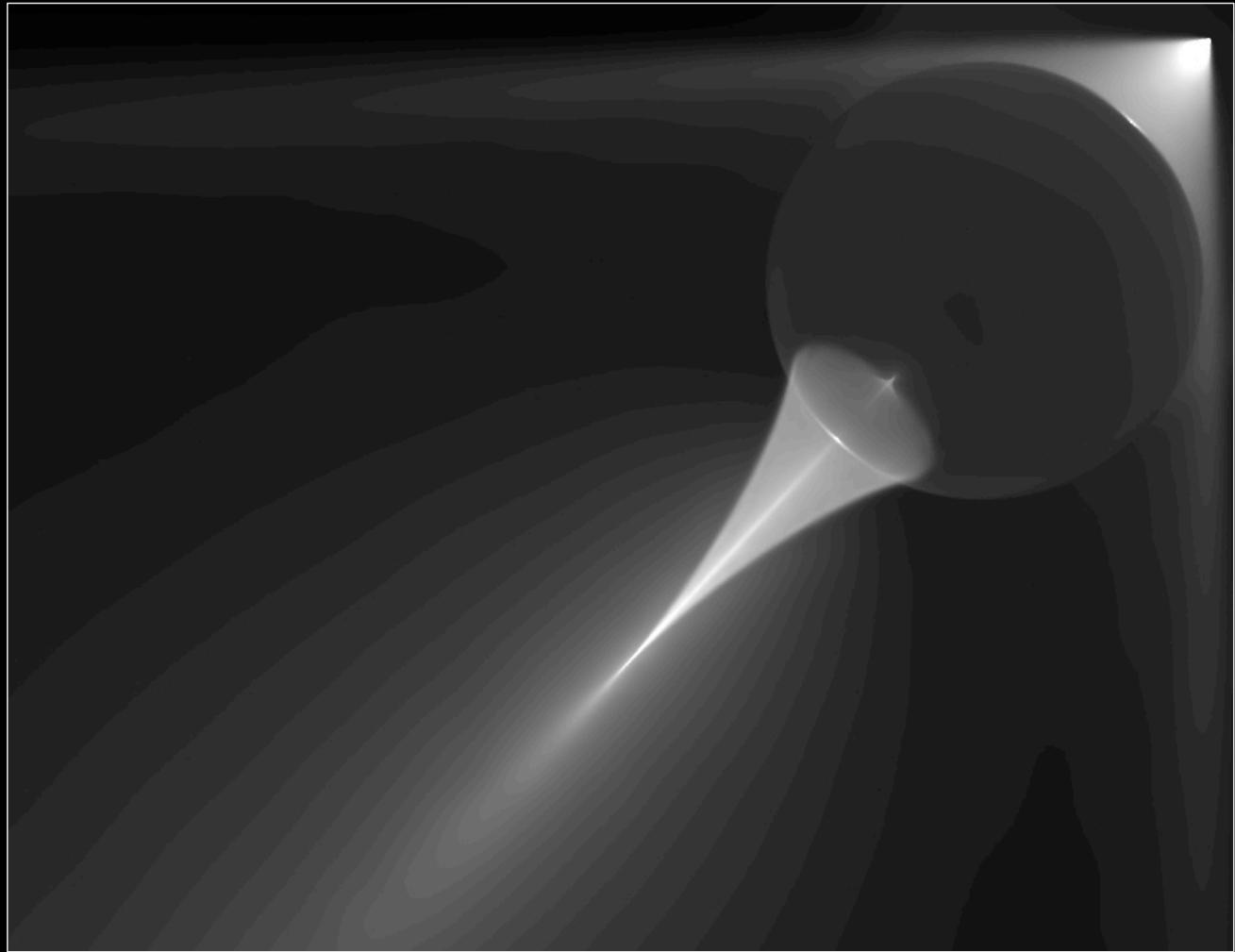


BRE: 4M Photons

$$89 + 638 = 727 \text{ s}$$

27

Saturday, August 4, 12

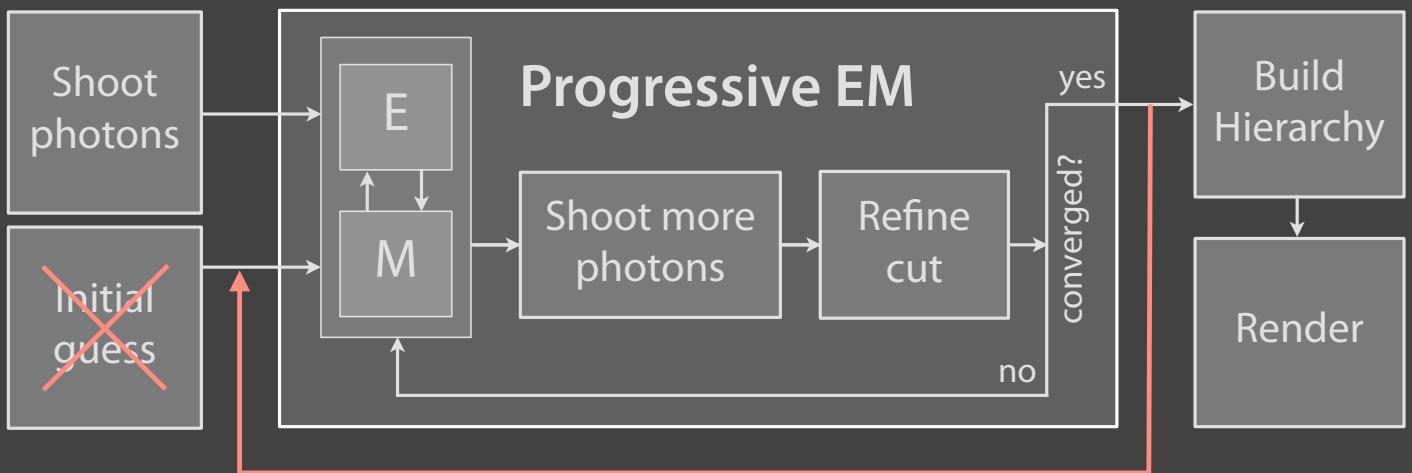


Our method: 16K Gaussians

$330 + 127 = 457 \text{ s}$
(1.6x)

28

Temporal Coherence



- Feed the result of the current frame into the next one
→ Faster fitting, no temporal noise

[Video]

[Video]

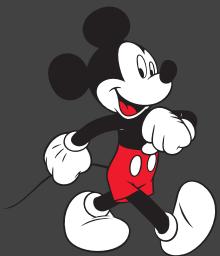
GPU-based rasterizer:

- Anisotropic Gaussian splat shader: 30 lines of GLSL
- Gaussian representation is very compact
(4096-term GMM requires only ~240KB of storage)

Conclusion

Contributions

- Rendering technique based on parametric density estimation
- Uses a progressive and optimized variant of accelerated EM
- Compact & hierarchical representation of volumetric radiance
- Extensions for temporal coherence and real-time visualization



Combination with MC integration and Path Space Regularization



SIGGRAPH ASIA 2013

Combining photon mapping and bidirectional path tracing

Iliyan Georgiev

In this talk I will discuss how to efficiently combine bidirectional path tracing and photon mapping into a unified algorithm via multiple importance sampling.



Bidirectional path tracing (30 min)

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 result 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.

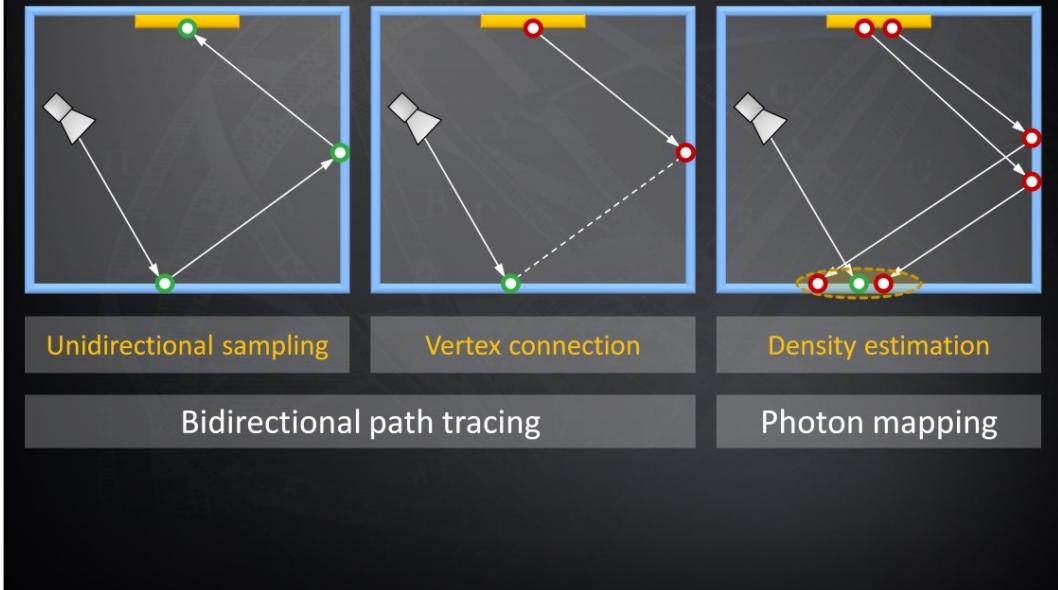


Vertex connection and merging (30 min)

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

SIGGRAPH
ASIA 2013



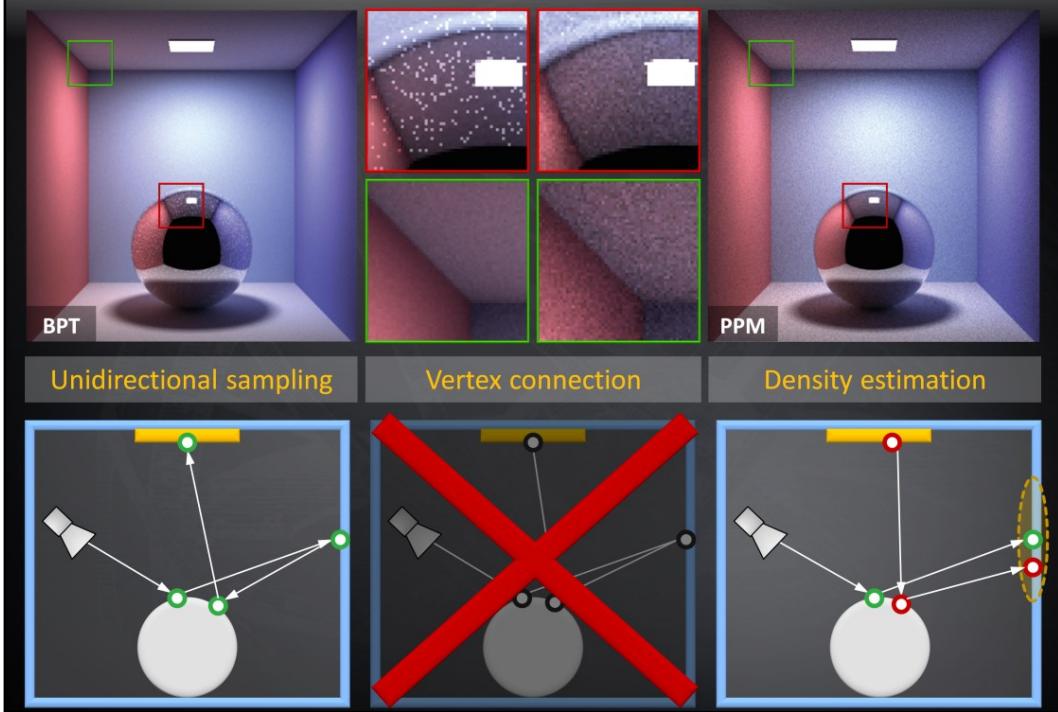
Let us start by reviewing how bidirectional path tracing (BPT) and photon mapping (PM) sample light transport paths that connect the light sources to the camera:

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

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

BPT vs PM

SIGGRAPH
ASIA 2013



BPT can efficiently capture directly visible caustics, as it can connect light subpath vertices to the camera. However, for sampling specular-diffuse-specular paths, BPT can only rely on unidirectional sampling, as VC cannot perform connections with specular vertices. Since the probability of randomly hitting the light source is often very low, the resulting images suffer from excessive noise.

On the other hand, photon mapping handles both direct and reflected caustics pretty much the same way – by loosely connecting nearby eye and light sub-path vertices. But as can be seen in the images above, it is less efficient than BPT for diffuse illumination.

⌚ Problem: different mathematical frameworks

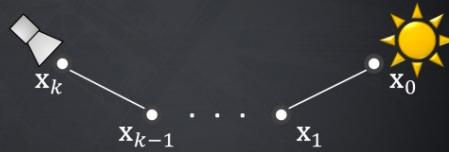
- BPT: Monte Carlo integration
- PM: Density estimation

👉 Key idea: Reformulate photon mapping in Veach's path integral framework

- 1) Formalize as path sampling technique
- 2) Derive path probability density

* The path integral

- $I_j = \int_{\Omega} f_j(\bar{x}) d\mu(\bar{x})$
- $\langle I_j \rangle = \frac{f_j(\bar{x})}{p(\bar{x})}$ ▪ $p(\bar{x}) = p(x_0, x_1, \dots, x_k)$



It has been long recognized that bidirectional path tracing (BPT) and photon mapping (PM) complement each other in terms of the light transport effects they can efficiently handle. However, even though both methods have been published more than 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.

The first step toward combining these two methods is to put them in the same mathematical framework. We choose Veach's path integral formulation of light transport due to its versatility and simplicity, and also because BPT is already naturally defined in this framework.

We need two key ingredients: (1) express PM as a sampling technique that constructs light transport paths that connect the light sources to the camera, and (2) derive the probability densities for paths sampled with this technique. This will give us a basis for reasoning about the relative efficiency of BPT and PM. And more importantly, it will lay the ground for combining their corresponding estimators via multiple importance sampling.

Bidirectional MC path sampling

SIGGRAPH
ASIA 2013

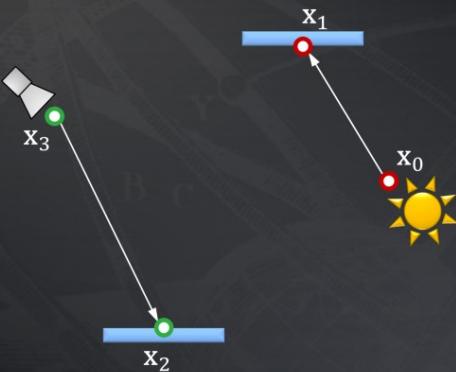


Let us start by taking a simple length-3 path and see how it can be constructed bidirectionally.

Bidirectional MC path sampling

SIGGRAPH
ASIA 2013

- Light vertex
- Camera vertex

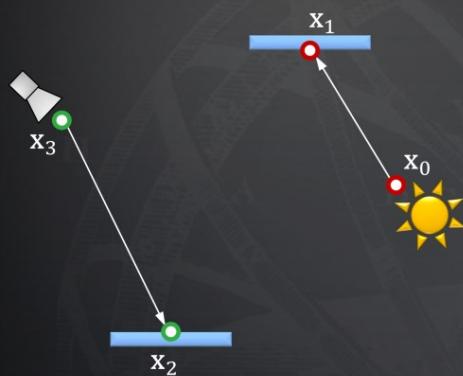


We first trace one subpath from the camera and another one from a light source.

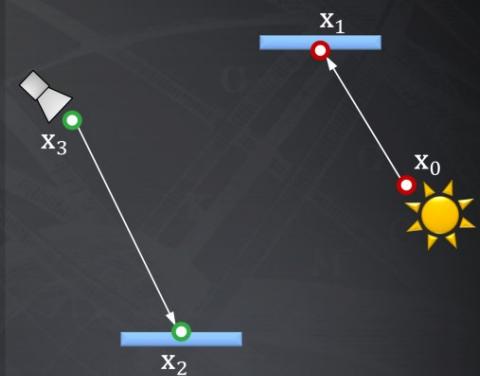
Bidirectional MC path sampling

SIGGRAPH
ASIA 2013

- Light vertex
- Camera vertex



Bidirectional path tracing

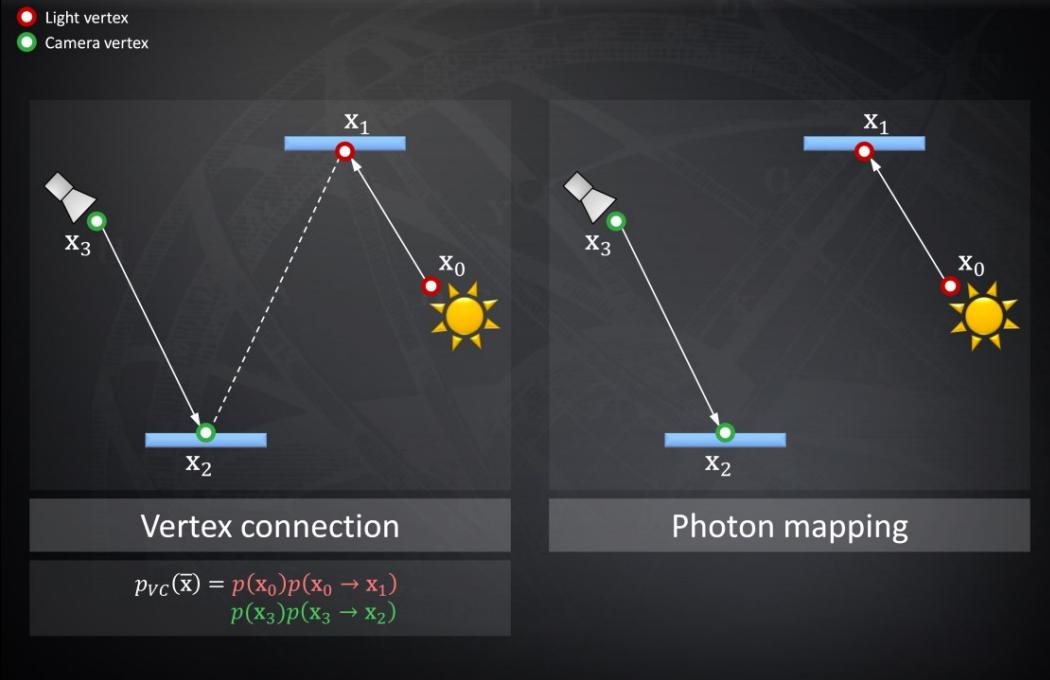


Photon mapping

Now let's see how we complete a full path in BPT and PM.

Bidirectional MC path sampling

SIGGRAPH
ASIA 2013

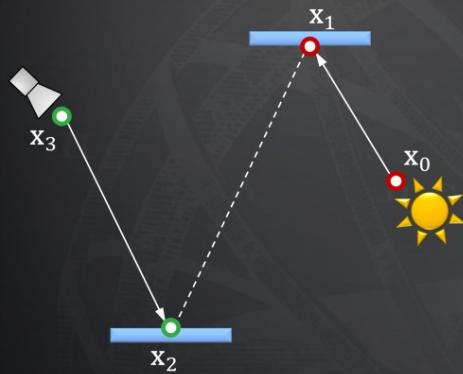


Bidirectional path tracing connects the subpath endpoints deterministically. We call this technique *vertex connection*. The PDF of the resulting full path is well known, and is simply the product of the PDFs of two subpaths, which have been sampled independently.

Bidirectional MC path sampling

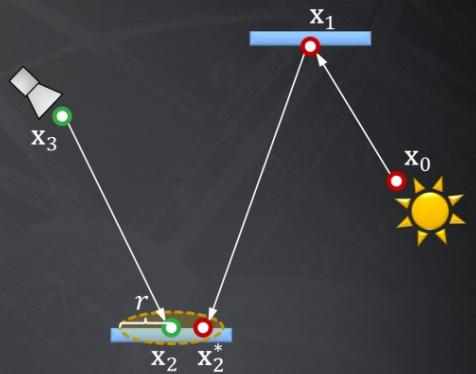
SIGGRAPH
ASIA 2013

● Light vertex
● Camera vertex



Vertex connection

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



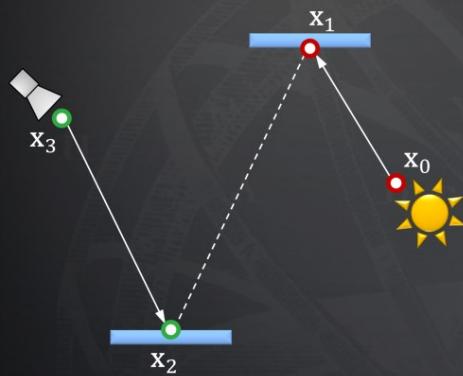
Photon mapping

Photon mapping, on the other hand, will extend the light subpath by sampling one more vertex from x_1 , and will concatenate the two subpaths only if the “photon” hit-point x_2^* lies within a distance r from x_2 .

Bidirectional MC path sampling

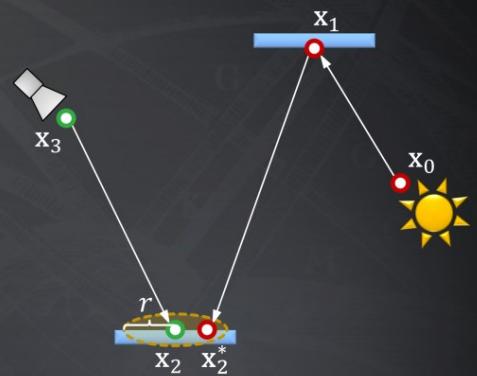
SIGGRAPH
ASIA 2013

● Light vertex
● Camera vertex



Vertex connection

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

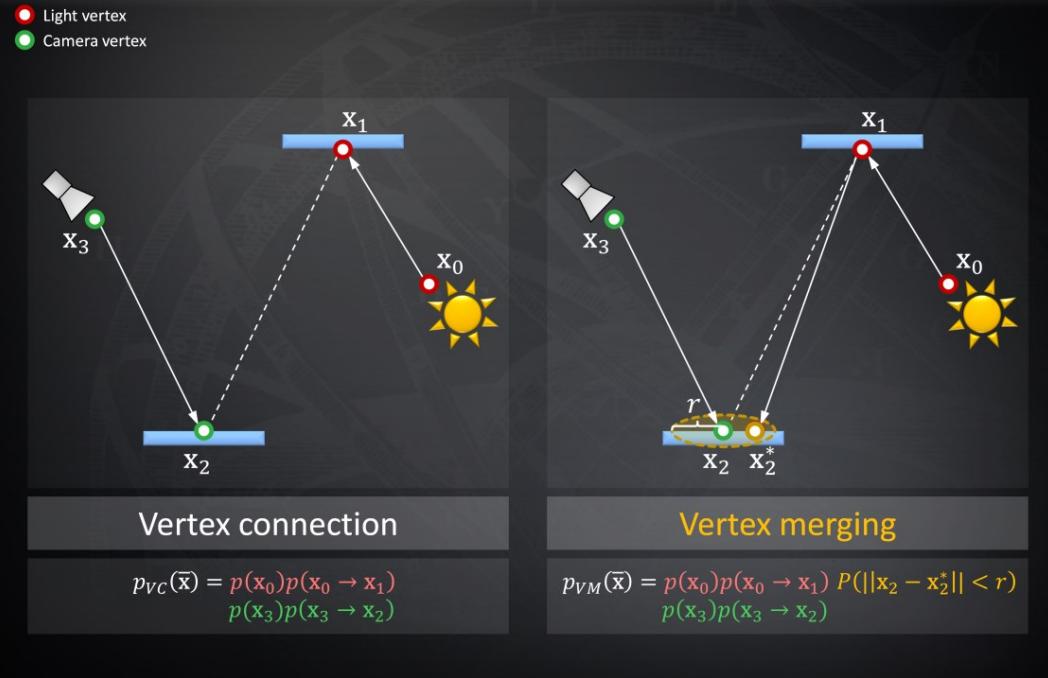


Vertex merging

We label this technique *vertex merging*, as it can be intuitively thought to weld the endpoints of the two subpaths if they lie close to each other.

Bidirectional MC path sampling

SIGGRAPH
ASIA 2013

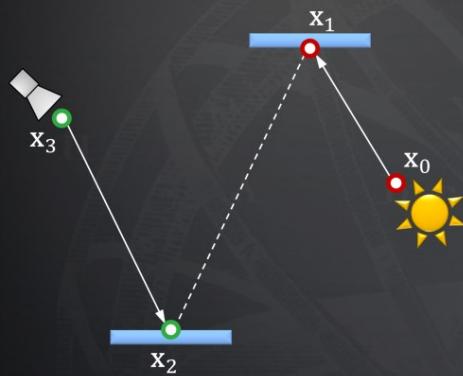


What remains is to derive the PDF of the resulting full path. To do this, we can interpret the last step as establishing a regular vertex connection between x_1 and x_2 , but conditioning its acceptance on the random event that a vertex x_2^* sampled from x_1 lands within a distance r to x_2 . This probabilistic acceptance is nothing more than a Russian roulette decision. The full path PDF is then again the product of the subpath PDFs, but in addition multiplied by the probability of sampling the point x_2^* within a distance r of x_2 . This acceptance probability is equal to the integral of the PDF of x_2^* over the r -neighborhood of x_1 .

Bidirectional MC path sampling

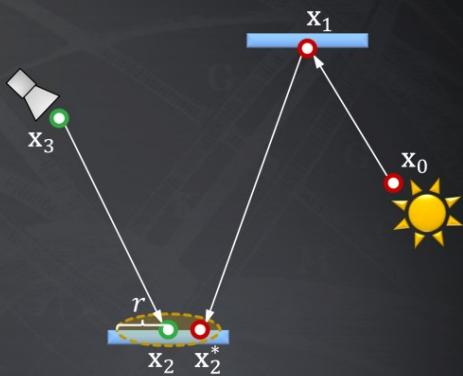
SIGGRAPH
ASIA 2013

● Light vertex
● Camera vertex



Vertex connection

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



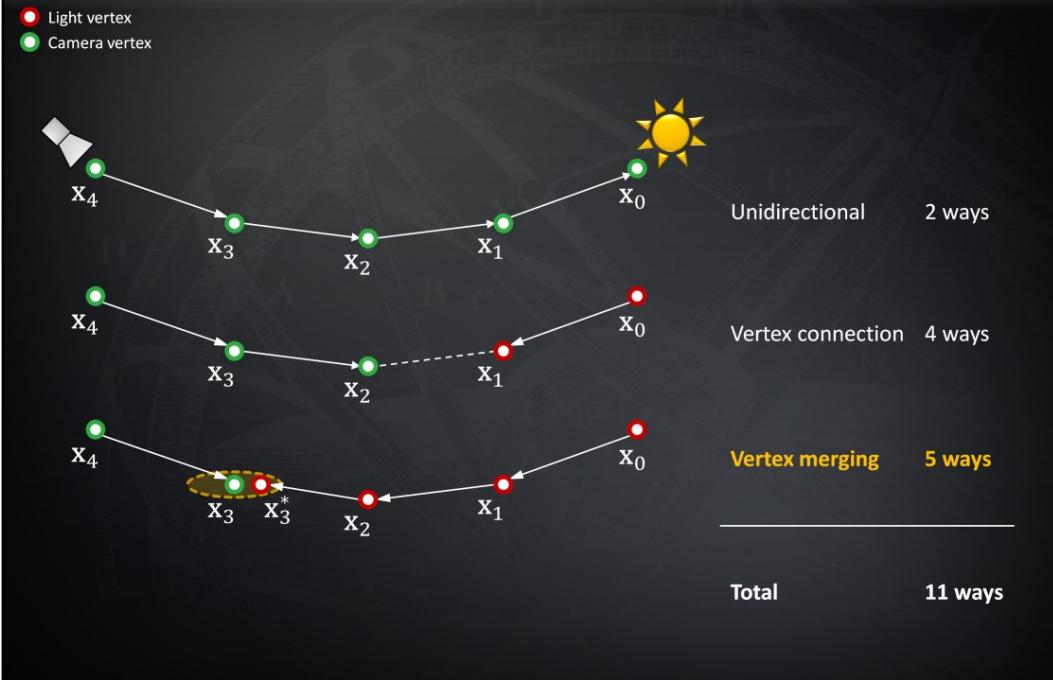
Vertex merging

$$p_{VM}(\bar{x}) \approx \frac{p(x_0)p(x_0 \rightarrow x_1)p(x_1 \rightarrow x_2^*)\pi r^2}{p(x_3)p(x_3 \rightarrow x_2)}$$

Under the reasonable assumptions that the surface around x_1 is locally flat, i.e. that this neighborhood is a disk, and that the density of x_2^* is constant inside this disc, the integral can be well approximated by the PDF of the actual point x_2^* we have sampled, multiplied by the disc area πr^2 .

Sampling techniques

SIGGRAPH
ASIA 2013



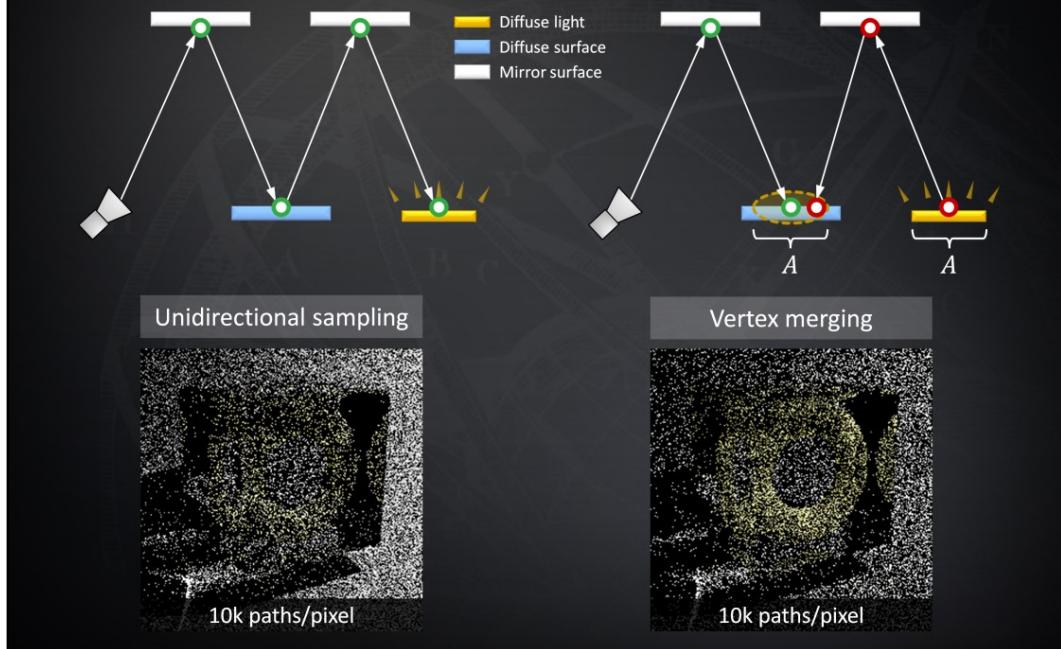
Now that we have formulated the vertex merging path sampling technique, we can put it side by side with the already available techniques in BPT. There are two ways to sample a length-4 path unidirectionally, and four ways to sample it via vertex connection. Vertex merging adds five new ways to sample 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.

But with so many ways to sample the same light transport path, a question naturally arises in the mind of the curious: which technique is the most efficient for what types of paths?

Technique comparison

SIGGRAPH ASIA 2013

SDS paths



To answer this question, let us first take a look at specular-diffuse-specular (SDS) paths. Here, bidirectional path tracing 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.

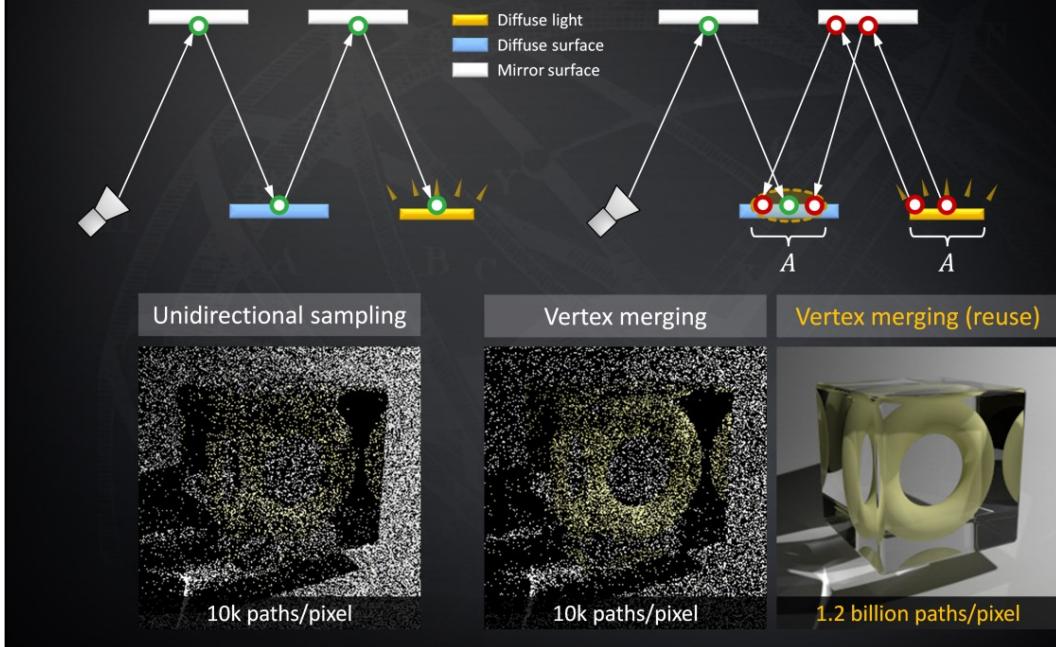
It can be shown that if the light source and the merging disk have the same area A , then unidirectional sampling and vertex merging sample paths with roughly the same probability density. This means that we should expect the two techniques to perform similarly in terms of rendering quality.

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 they hit the light. For image on the right, we trace subpaths from both ends, and merge their endpoints if they lie within a distance $r = \sqrt{A/\pi}$ from each other. Both images look equally noisy, even after sampling 10,000 paths per pixel. This confirms that vertex merging, and thus photon mapping, is *not* an intrinsically more robust sampling technique for SDS paths than unidirectional sampling.

Technique comparison

SIGGRAPH
ASIA 2013

SDS paths

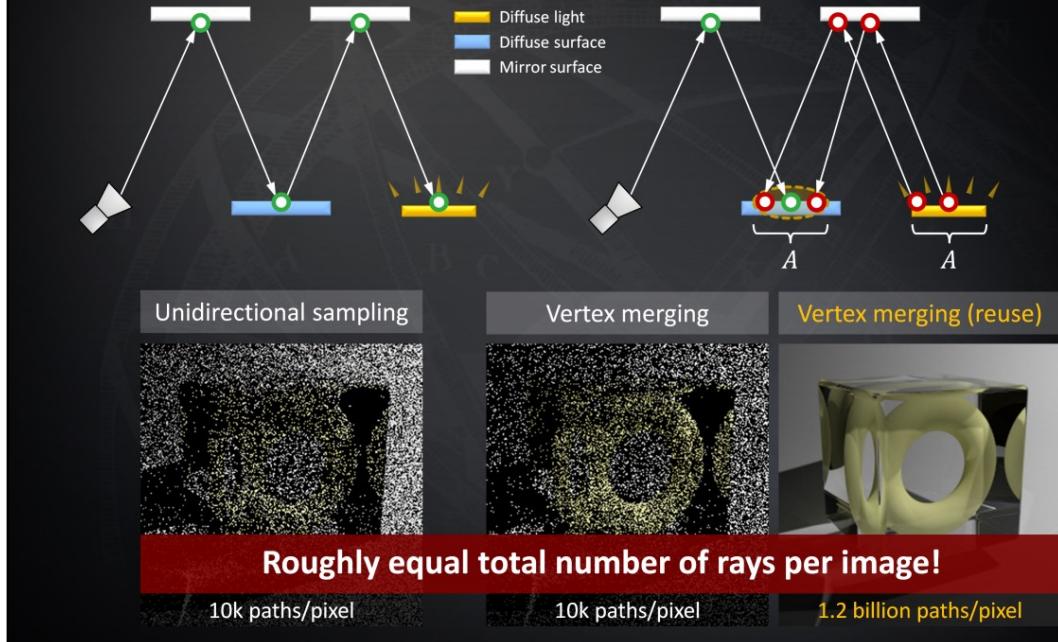


However, the strength of vertex merging is computational efficiency – we can very efficiently reuse the light subpaths traced for *all* pixels at the cost of a single range search query. This allows us to quickly construct orders of magnitude more light transport estimators from the same sampling data,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

Technique comparison

SIGGRAPH
ASIA 2013

SDS paths

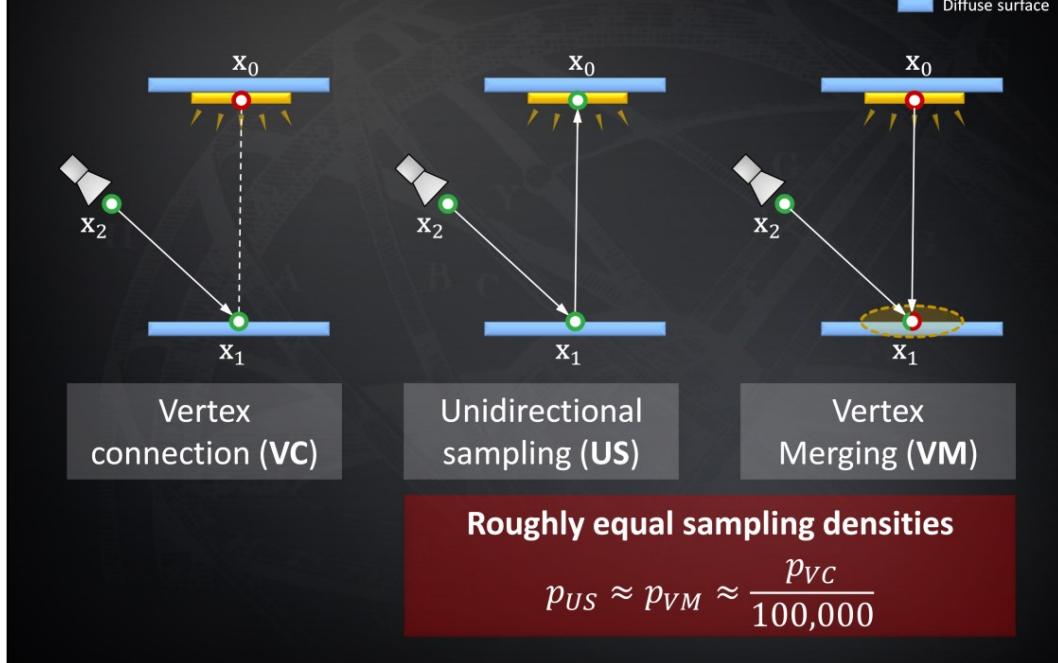


For all these three images we have traced roughly the same number of rays, and the only difference between the one in the center and the one on the right is that the for right image we have enabled path reuse, by storing, and looking up, the light subpath vertices in a photon map at every rendering iteration.

Technique comparison

SIGGRAPH
ASIA 2013

Diffuse illumination



Now let's look at another extreme example – diffuse illumination. Note that vertex connection (VC) constructs the edge between x_1 and x_2 deterministically, while unidirectional sampling (US) and vertex merging (VM) both rely on random sampling.

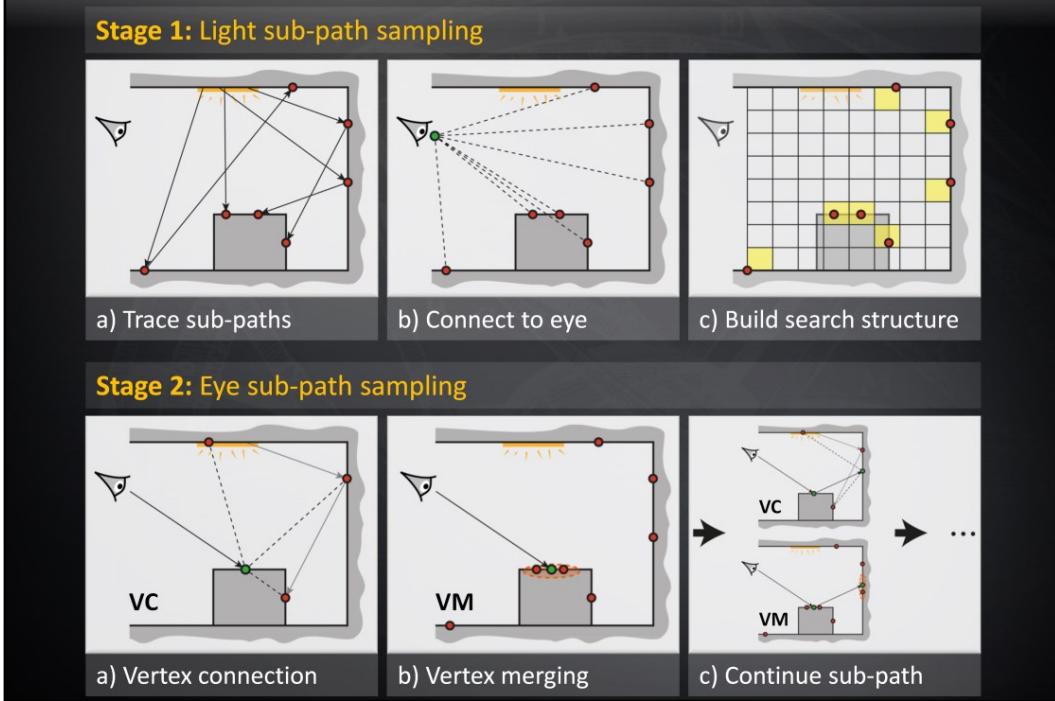
Once again, it can be shown that if the light source and the merging disk have the same area, then US and VM sample this path with roughly the same probability density.

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, vertex merging, and thus photon mapping, gains a lot of efficiency over unidirectional sampling.

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

Vertex connection & merging (VCM)

SIGGRAPH ASIA 2013



Even more usefully, 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, which we call *vertex connection and merging* (VCM), operates in two stages.

1. In the first stage, we
 - a) trace the light subpaths for all pixels,
 - b) connect them to the camera, and
 - c) store them in a range search acceleration data structure (e.g. a kd-tree or a hashed grid).
2. In the second stage, we trace a camera subpath for every pixel.
 - a) 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) merged with the vertices of *all* light subpaths.
 - c) We then sample the next vertex and do the same.

In a progressive rendering setup, we perform these steps at each rendering iteration, progressively reducing the vertex merging radius . For details on this, please refer to the cited papers below for details.



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)



Vertex connection and merging (30 min)



Here, we visualize the relative contributions of VM and VC techniques to the VCM image from the previous slide. This directly corresponds to the weights that VCM assigned to these techniques.



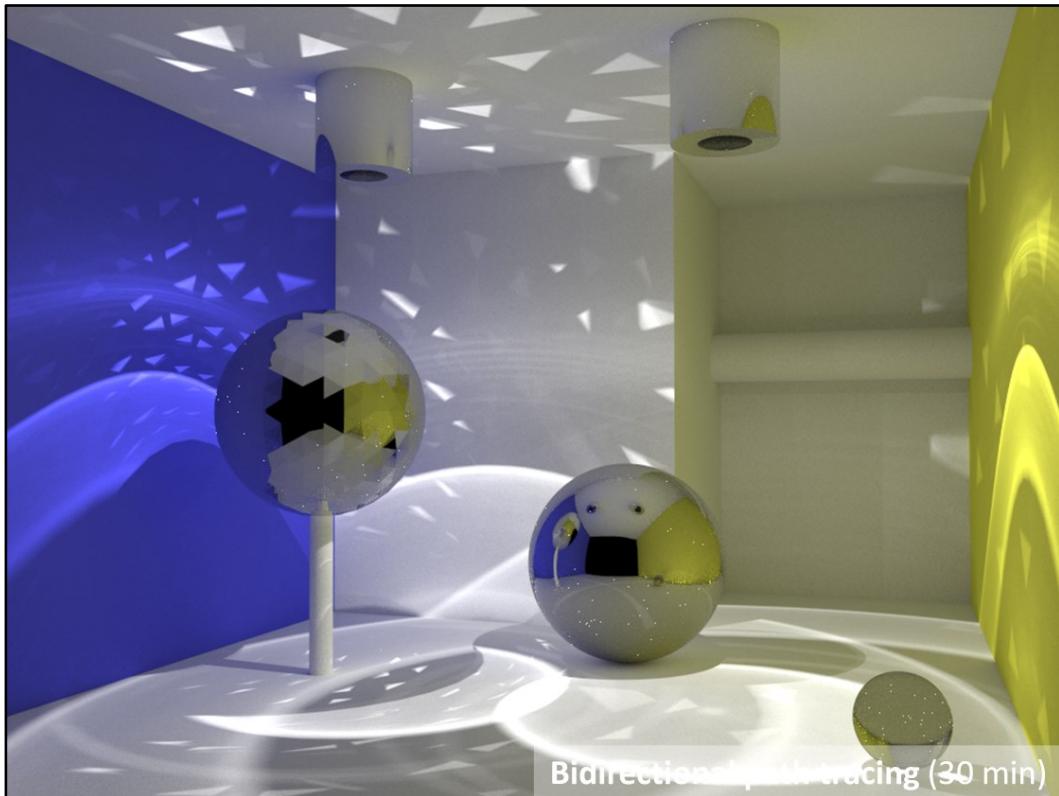
Bidirectional path tracing (30 min)

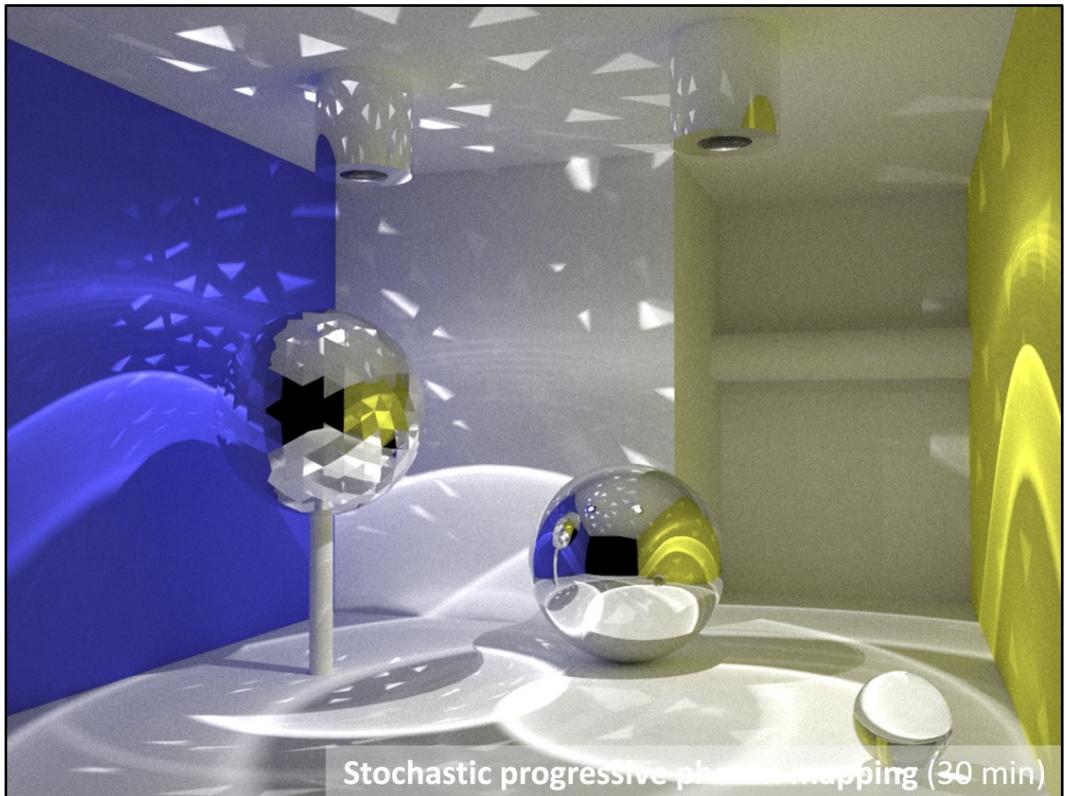


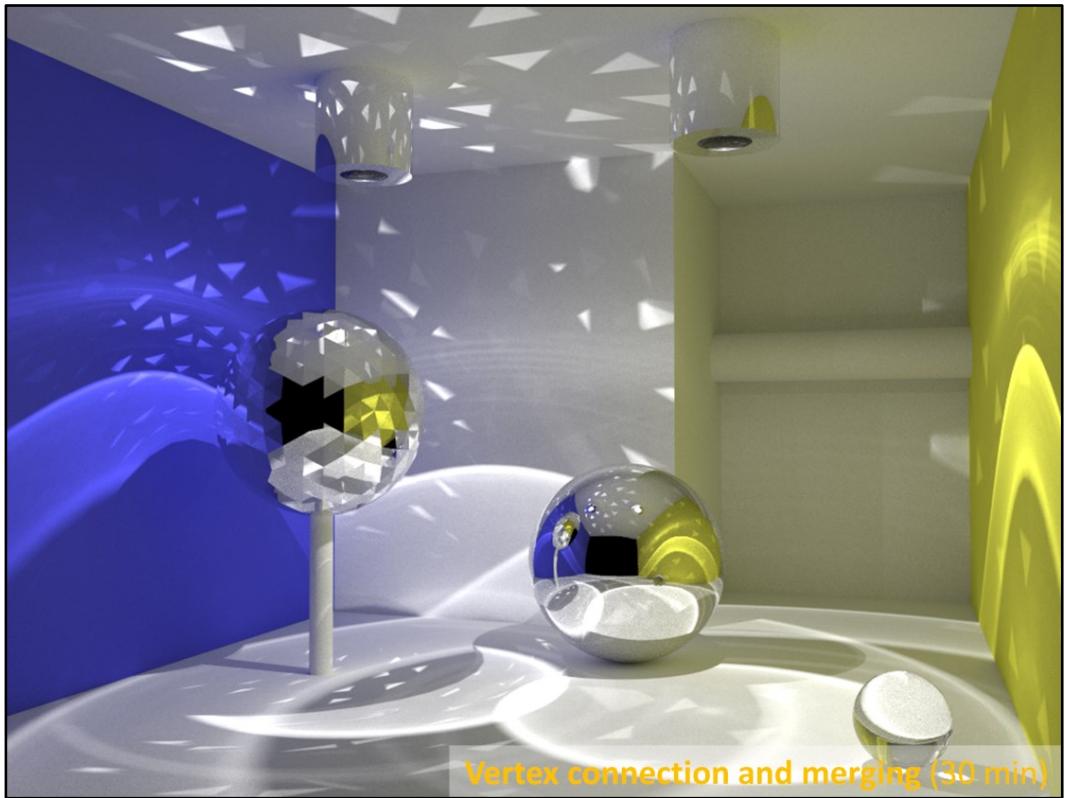


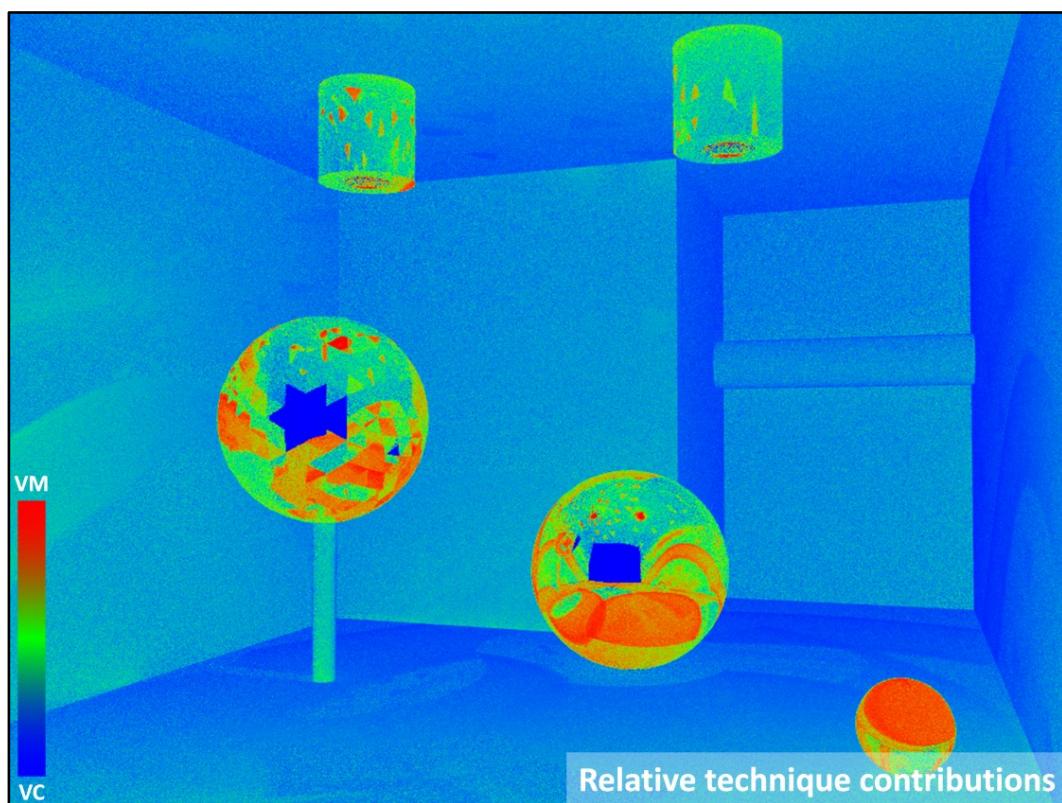
Vertex connection and merging (30 min)











Good practices



- ★ No vertex merging for
 - Direct illumination
 - Direct 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 VCM in a production renderer.

We can almost 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 VCM algorithm I just described has the practical issue that its memory footprint can be much larger than that of (progressive) photon mapping. The problem 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 VCM, 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.

* Merging radius

- Compute from pixel footprint (ray differentials)
- Don't reduce (or use $\alpha = 0.75$)

* MIS weights

- Efficient accumulation during sub-path sampling

* Spectral rendering, motion blur

- ❖ Merging in wavelength/time reduces efficiency
- Reuse photons over rendering iterations

The progressive VCM 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 VCM, 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.

And finally, a note on spectral rendering and motion blur. When a (sub)path can only carry a single randomly sampled wavelength and/or time instant, in order to merge two subpath endpoints, they not only need to be close to each other in Euclidean space, but also in wavelength and/or time. This can significantly reduce the efficiency of vertex merging, since the VM PDFs are then additionally multiplied by a corresponding wavelength/time acceptance probability, and can thus be much lower. A simple solution to ameliorate this problem is to accumulate and reuse light vertices (photons) over N iterations. This number N depends on the wavelength/time merging radius – the larger the radius, the smaller N can be. Note that efficient light vertex storage in this case becomes even more important.

* Papers

- “Light Transport Simulation with Vertex Connection and Merging”
[Georgiev et al. 2012]
- “A Path Space Extension for Robust Light Transport Simulation”
[Hachisuka et al. 2012]
- Same algorithm, different theoretical derivations!

* Additional material

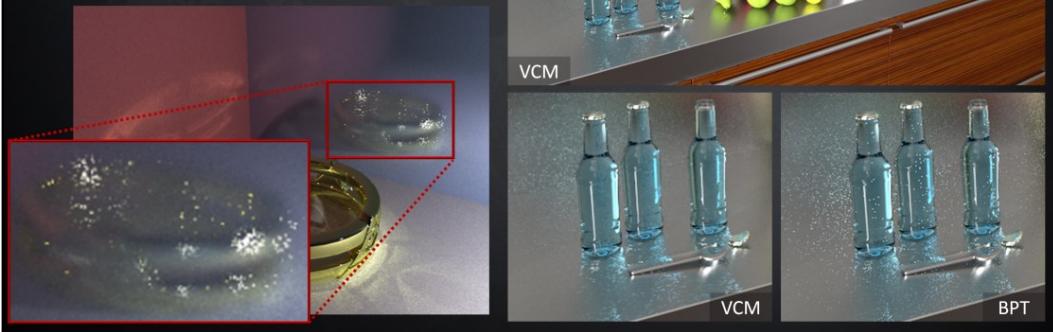
- **Implementation tech. report, image comparisons** [iliyan.com]
- **SmallVCM** – open-source VCM implementation [SmallVCM.com]

In this talk, I could only give a high-level description of VCM. For more details, please refer to these two papers, which derive the same practical algorithm using two different theoretical formulations. They were simultaneously published by two independent research groups at SIGGRAPH Asia 2012, which we believe only reaffirms the correctness of the approach. Lots of additional material can be found on my web site, and we have also released a reference open source implementation in the SmallVCM renderer.

* Error convergence

- ↳ BPT: $O(N^{-0.5})$
- ↖ PPM: $O(N^{-0.33})$
- ↳ VCM: $O(N^{-0.5})$

* Remaining challenges



In summary, vertex connection and merging tries to combine the best of bidirectional path tracing (BPT) and (progressive) photon mapping. An important property of the algorithm is that it retains the higher order of convergence 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.

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. A prominent example are caustics falling on a glossy surface. On the kitchen scene, even though VCM brings practical improvements over BPT, there is still a lot to be desired from the caustics on the glossy surface.

VCM in production

SIGGRAPH
ASIA 2013

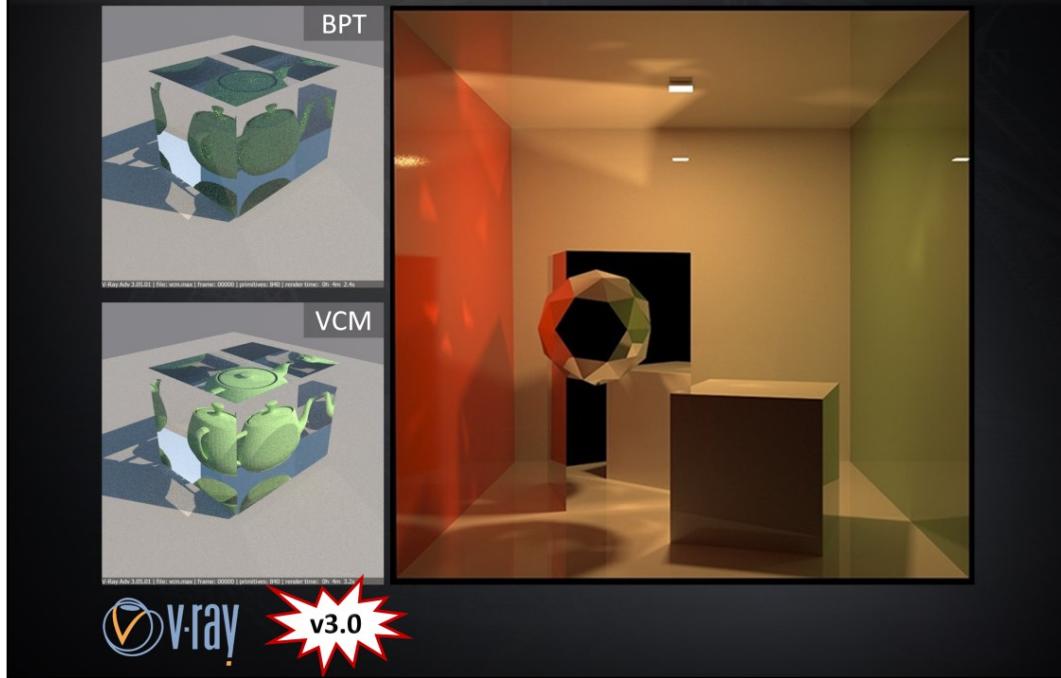


A number of companies have announced VCM integration in the upcoming releases of their commercial renderers.

For example, VCM is coming in Pixar's Photorealistic RenderMan v19.

VCM in production

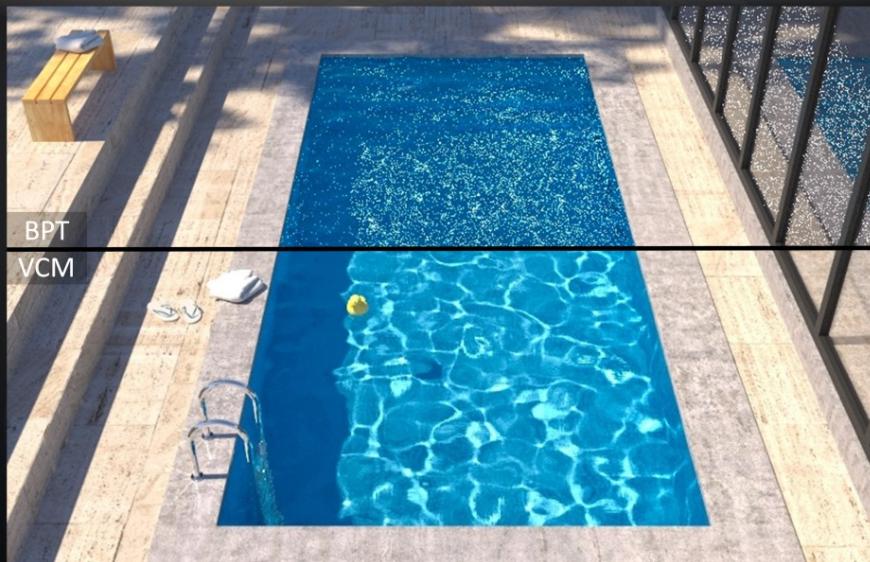
SIGGRAPH
ASIA 2013



Chaos Groups have also shown first images from V-Ray 3.0 with VCM support.

VCM in production

SIGGRAPH
ASIA 2013



Corona Renderer Alpha | Intel(R) Core(TM) i7-2600 CPU @ 3.40GHz | Time: 1:00:02 | Passes: 4075 | Primitives: 1,234,110 | Rays/s: 2,772,102



corona by Ondřej Karlík <http://www.corona-renderer.com>

VCM is also implemented in the public Alpha release of Corona renderer, which I encourage you to download and play around with.



That was all from me today. Thank you for your attention!

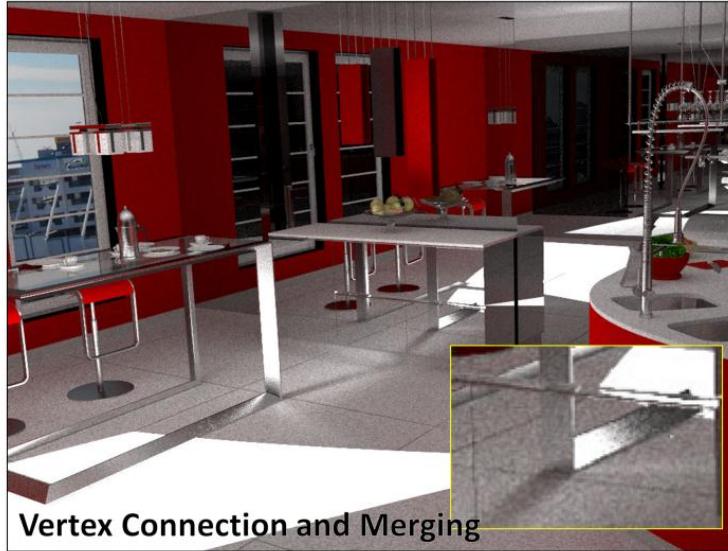
Path Space Regularization for Robust Light Transport

Anton S. Kaplanyan

Karlsruhe Institute of Technology, Germany

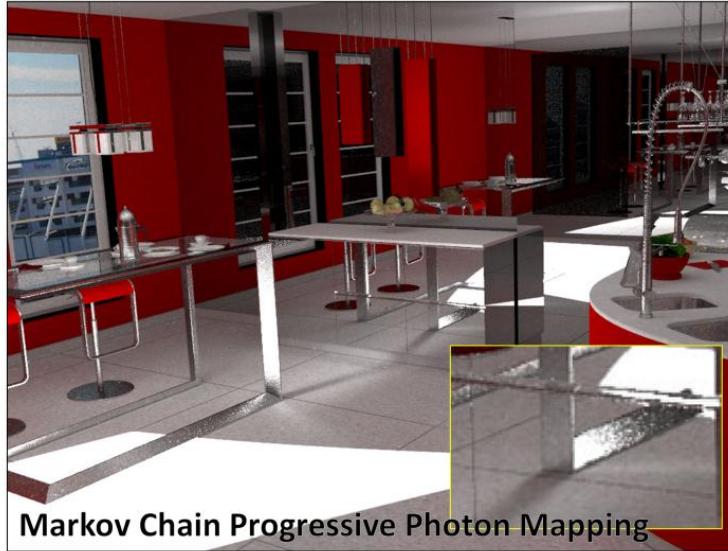


Motivation



Even the most recent existing methods are either not good **at** or not capable **of** handling complex illumination, such as reflected caustics on the floor.
In this part we will show how to combine the **strengths** of these methods together.

Motivation



Even the most recent existing methods are either not good **at** or not capable **of** handling complex illumination, such as reflected caustics on the floor.
In this part we will show how to combine the **strengths** of these methods together.

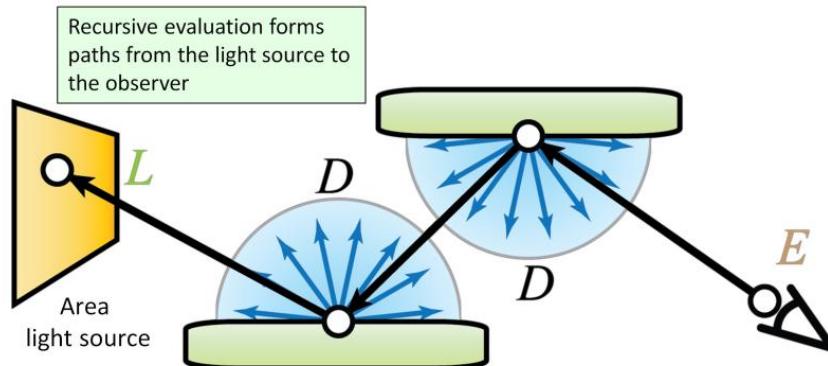
Motivation



Even the most recent existing methods are either not good **at** or not capable **of** handling complex illumination, such as reflected caustics on the floor.
In this part we will show how to combine the **strengths** of these methods together.

Path Tracing

- Multiple reflections are computed via recursion
- Full path constructed once hit the light source
- Monte Carlo samples are accumulated



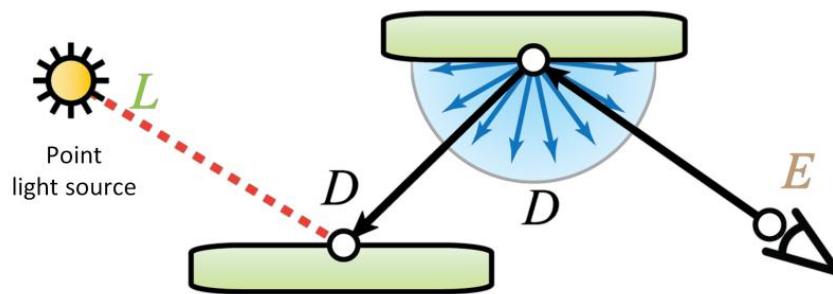
The first and the simplest method is called **path tracing**.

The rays are shot from the **camera** and bounced until **hit the light source**.

The complete paths are sampled **stochastically** and the rendering equation is solved using Monte Carlo integration.

Explicit Connection to Light

- Point light sources are impossible to hit
- Solution: Direct connection to the light source



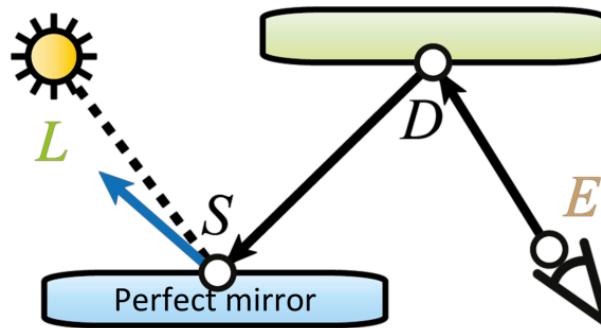
However it is impossible to directly **hit** a light source when rendering a scene in presence of **point lights** using path tracing.

The point light model is a common mathematical model used in graphics for many real-world emitters with tiny emissive area.

As a solution, such light sources are explicitly checked with **direct connection** at each interaction.

Non-Sampleable Paths

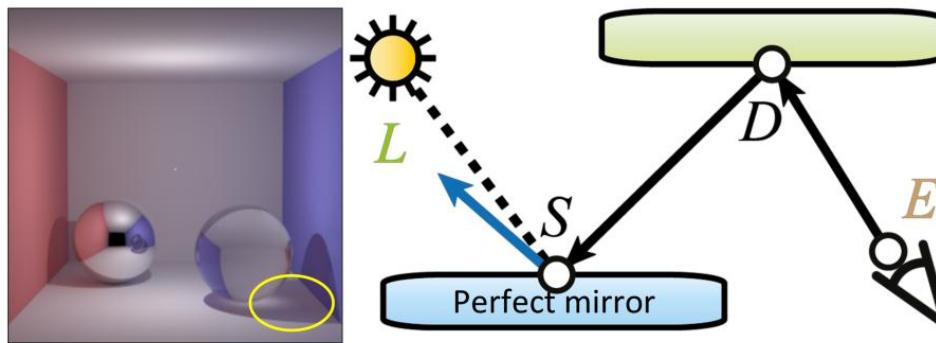
- Sometimes direct connection is not possible
 - Last edge can be fixed by specular reflection



However in some cases even direct connection **does not** solve the problem.
 Imagine a **caustic**. Typically in such case the interaction next to the light source is **perfectly specular**, such as mirror reflection or glass refraction.
 The direct connection **always fails**, since the connection direction should perfectly match the **stochastic** reflected direction.

Non-Sampleable Paths

- Path tracing: Any path with deterministic interaction to point light
 - Sampling probability is zero
 - Caustics from point lights are missing
- Impossible-to-sample paths appear due to singularities
 - Dirac deltas: Perfect reflections, refractions



So the paths with the deterministic last interaction, such as caustics from point lights, **cannot be constructed** with path tracing.

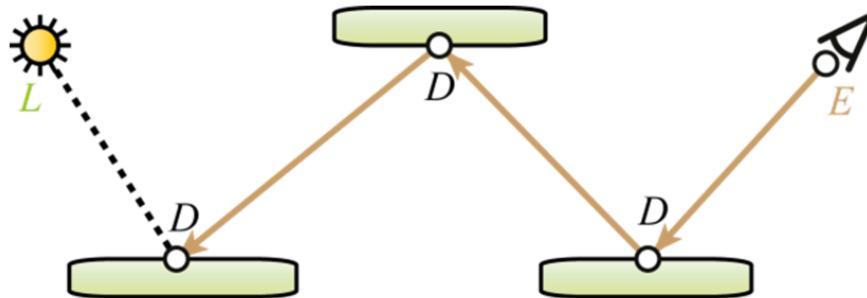
For example, in this simple image on the left, rendered with path tracing, the caustics on the floor are **missing**.

This happens because the deterministic interaction is mathematically defined as a Dirac delta distribution, causing a **singularity** in the integrand.

However there are more advanced methods that **can** sample caustics from point lights.

Sampling with Bidirectional Path Tracing

- Bidirectional path tracing (BDPT) samples subpaths from both ends
 - Constructs the path with all possible deterministic connections



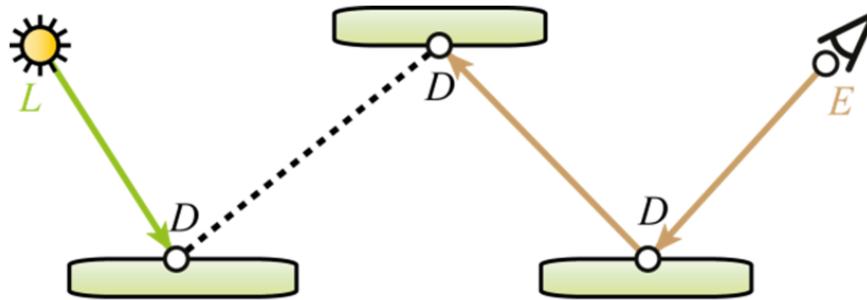
One of them is bidirectional path tracing.

It can potentially connect a given path at **every edge**, as illustrated on this path with four edges.

So the first option is to trace all the way to the light and then do a direct connection to the light source at the last edge.

Sampling with Bidirectional Path Tracing

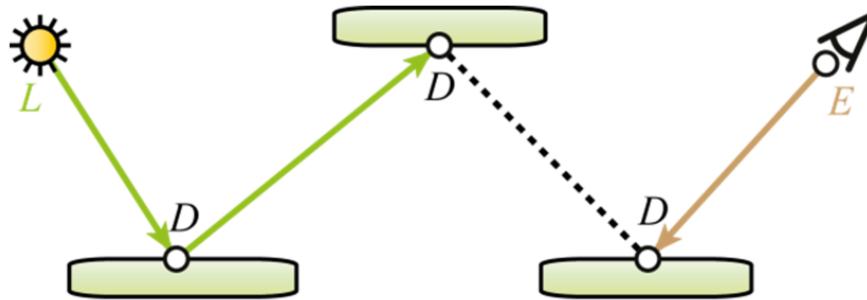
- Bidirectional path tracing (BDPT) samples subpaths from both ends
 - Constructs the path with all possible deterministic connections



Or trace it further from the light and do a connection in the middle.

Sampling with Bidirectional Path Tracing

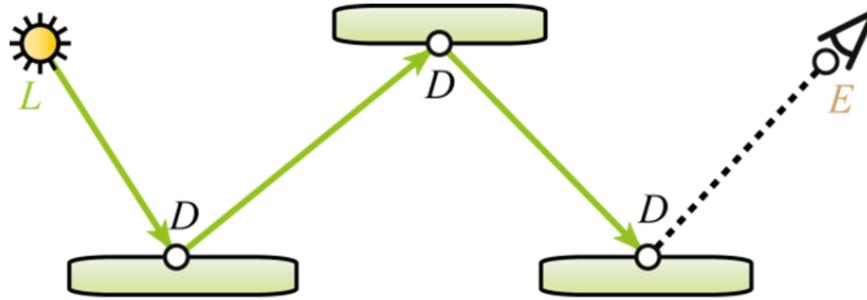
- Bidirectional path tracing (BDPT) samples subpaths from both ends
 - Constructs the path with all possible deterministic connections



Or trace it further from the light and do a connection in the middle.

Sampling with Bidirectional Path Tracing

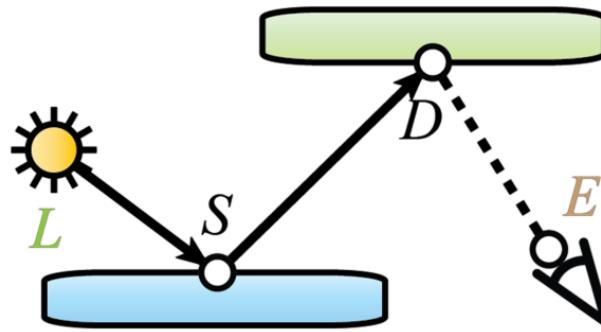
- Bidirectional path tracing (BDPT) samples subpaths from both ends
 - Constructs the path with all possible deterministic connections



Or trace it all the way from the light source and do a direct connection to the camera sensor.

Sampling with Bidirectional Path Tracing

- Bidirectional path tracing (BDPT) samples subpaths from both ends
 - Constructs the path with all possible deterministic connections
- Given the same example of caustics, BDPT can connect to the camera
- BDPT can sample caustics from point lights
 - Explicit evaluation of singularities



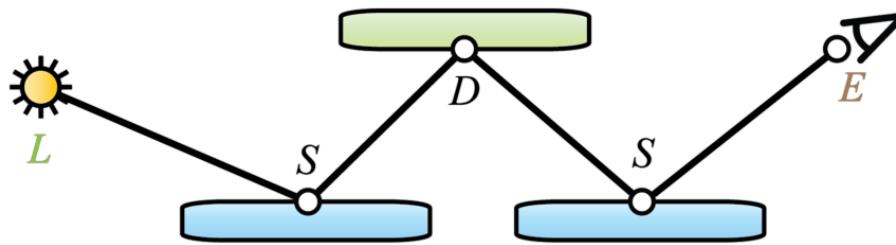
Consider the **previous example** of a caustic.

BDPT can construct such path starting from the light source by tracing up to the non-deterministic interaction, such as **diffuse surface**, and then connect to the camera.

The trick is that BDPT **bypasses** the singularities, instead of attempting to stochastically connect at them.

Non-Sampleable Paths, Part II

- What if all edges of the path contain singularities?
 - No place to perform a connection
 - Impossible to sample with any unbiased sampling [Veach97]
- Example: Reflected caustics from point light (L SDS E)



14 SIGGRAPH ASIA 2013

However, again, even such sophisticated methods like BDPT can **fail** to construct a path in some cases.

One of these cases is the **reflected caustics**.

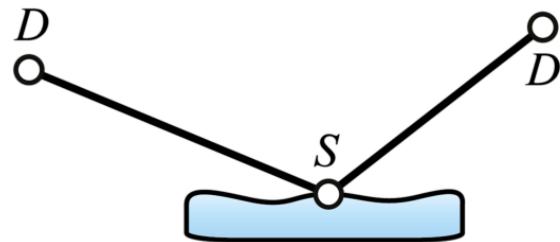
The problem here is that **every edge** adjoin a singularity at one of its ends.

Such paths with no connectible edges are showed to be **non-sampleable with local unbiased methods** by Eric Veach.

And there is a more fundamental reason for that. Let's take a brief **excursus**.

Excusus: Undecidability of Ray Tracing

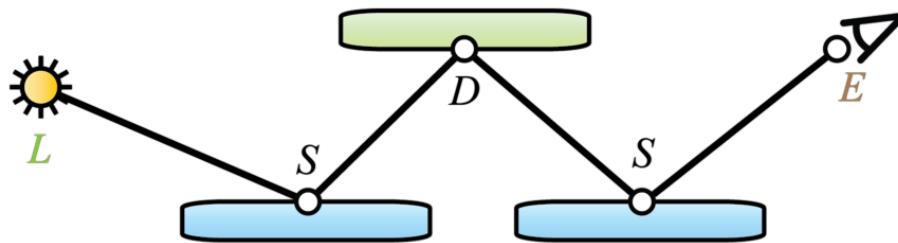
- Pure specular path tracing is *undecidable* [Reif et al. 1994]
 - Task: Find all specular paths from one fixed point to another (DS+D)
 - Impossible to find all such paths on a Turing machine
 - Given arbitrary specular geometry



Interestingly, it is **not possible** to find all specular paths from one fixed point to another on a Turing machine given an arbitrary configuration of specular surfaces.

Non-Sampleable Paths, Part II

- What if all edges of the path contain singularities?
 - No place to perform connection
 - Impossible to sample with any unbiased sampling
- Example: Reflected caustics from point light (LSDSE)
- Such paths cannot be found precisely
 - Reflected caustics: Two undecidable subpaths



16 SIGGRAPH ASIA 2013

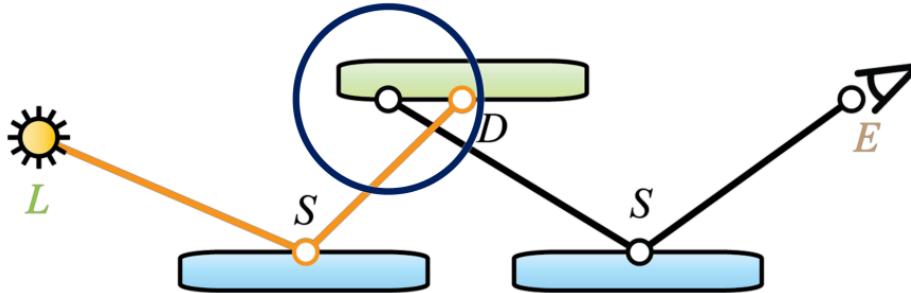
Coming back to an example of reflected caustics, we can see that this path consists of **two** undecidable subpaths.

Thus it does not matter from which direction to trace these subpaths, one of them will pose such problem during the connection.

Photon Mapping (a.k.a. Vertex Merging)

What can we do about this problem?

- Merge nearby vertices instead of connecting them
 - Imprecise path construction → causes bias (blurring) everywhere
- Can construct paths with only one diffuse interaction
- The chance two vertices are close is very low
 - Large cache (photon map) is required



Not a secret that photon mapping can sample such paths, considered non-sampleable by local unbiased methods.

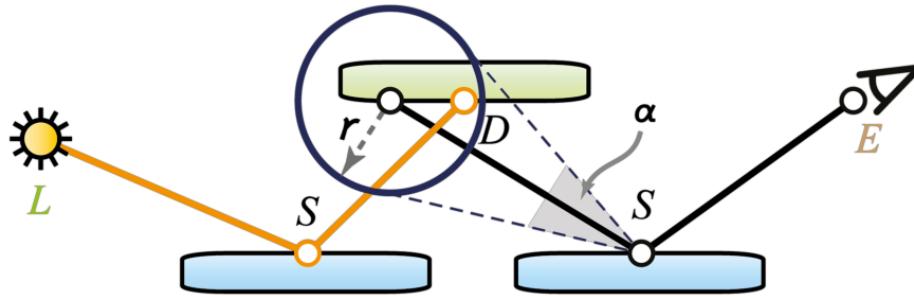
The reason is photon mapping constructs the full path by **merging** proximate vertices of different subpaths.

This is a **biased way**, causing blurring of image features. However this way **more difficult paths**, such as reflected caustics, can be sampled.

Note that the chance that two vertices happened to be located nearby is **very low**. Thus photon mapping requires a **large cache** of the light subpaths, called a photon map.

Photon Mapping: Under the Hood

- Photon mapping essentially regularizes specular interactions
- Regularization angle depends on the connection distance
 - On-surface radius is fixed for a given point
- It is a known mathematical procedure...



18 SIGGRAPH ASIA 2013

What photon mapping is going during density estimation is also known as regularization in mathematics.

Photon mapping **regularizes the interaction** by merging the path at the next vertex. The **regularization angle**, α , can be derived from the fixed photon mapping radius r and depends on the connection distance.

It appears to be a standard mathematical procedure. We also use this procedure, however in a more smart way.

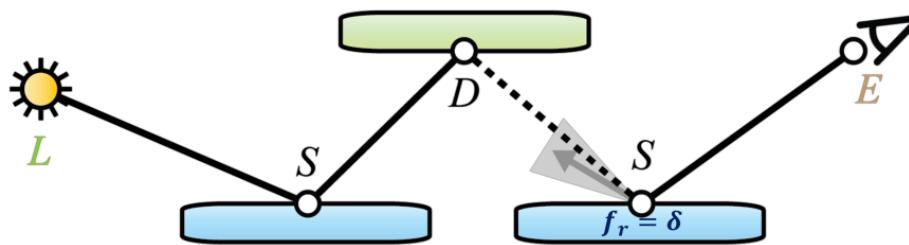
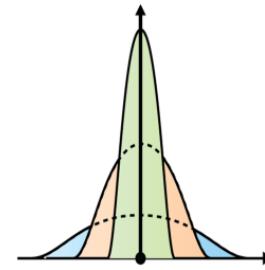
Regularization with Mollification

- Example: Dirac delta: $\delta(0) = +\infty$, $\delta(x) = 0$ for $x \neq 0$

$$\int_{\mathbb{R}} f(x)\delta(x)dx = f(0)$$

- Mollification: Approximate delta distribution with a sequence of functions $\{\varphi_\alpha\}$ with vanishing support

- Typically $\varphi_\alpha = \frac{1}{\alpha} \varphi_0(x/\alpha)$, regularization angle α
- Implicitly smoothes the integrand



This procedure is called **mollification**.

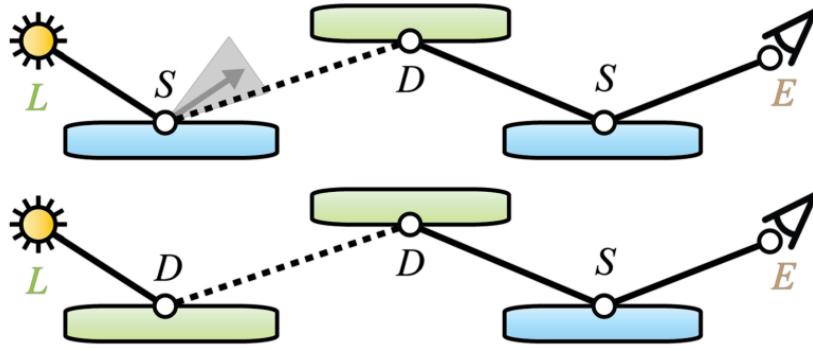
Given a singularity caused by a **delta distribution**, like the one at the right specular vertex;

we construct a **sequence** of integrable smooth functions, that approach delta distribution in the limit.

Then we shrink the regularization angle during the integration, making the integrand less and less smooth.

Selective Regularization of Path Space

- Goal: Minimize bias. Why smoothing a regular path?
 - Photon mapping causes uniform bias everywhere
- Regularize **only** non-sampleable paths!
 - Only if **all** edges join a singularity
 - Can only be detected once all interactions are known



20 SIGGRAPH ASIA 2013

So, now that we formulated what is going on, we can selectively regularize only when **necessary**.

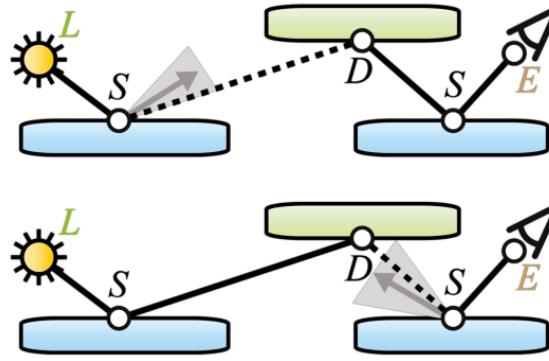
Instead of regularizing all paths, including regular ones, we regularize **only non-sampleable paths (irregular for the sampling method)**, thus **minimizing** the amount of bias.

In case of BDPT, non-sampleable paths are detected if **all edges** of the path adjoin at least one singularity.

This situation can be recognized only once **all subpaths are traced** and all interactions are known.

Combining Bidirectional Estimators

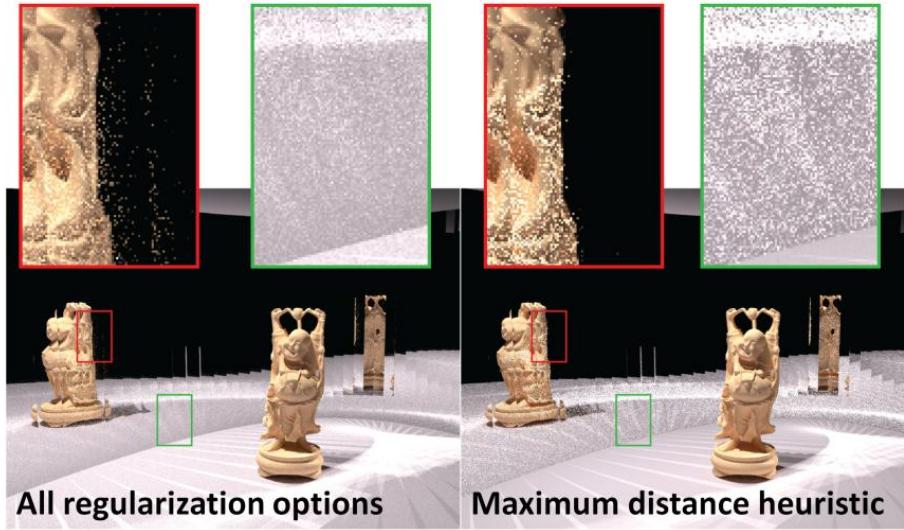
- Assume a non-sampleable path is detected
- Which regularization option to choose?
- We propose a *maximum connection distance* heuristic
 - Minimizes angular blurring
 - Simple to handle and implement



If the path generation method has multiple options of constructing the same path, such as a set of bidirectional estimators, then we need to choose between **several** different regularization options, as the two options for the same path in the bottom.

Here we propose to use the *maximum distance heuristic*: we regularize only if the connection edge is the longest among other path edges. Given that the on-surface radius is fixed, we minimize the angular smoothing caused by the regularization. Also it is very easy to practically handle.

Combining Bidirectional Estimators

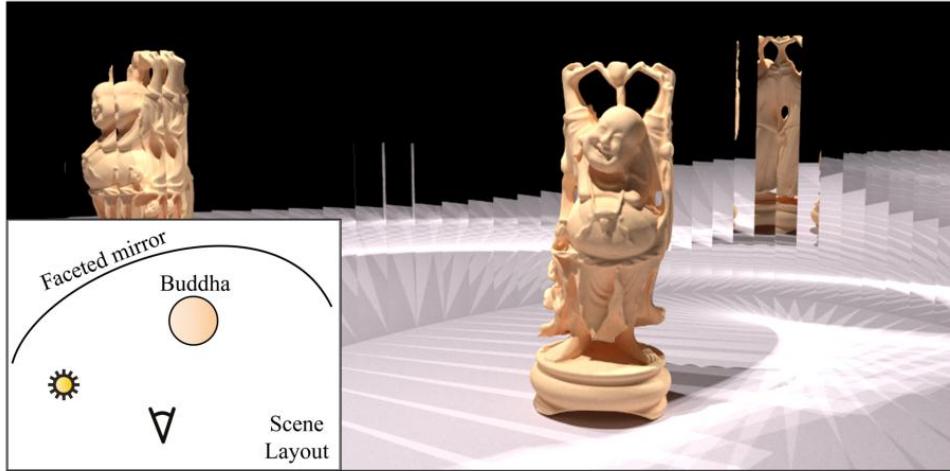


The example shows **all regularization options**, equally weighted, on the left; and the maximum distance heuristic on the right.

The bias caused by angular blurring is notably **smaller** on the right.

Comparison of Different Methods

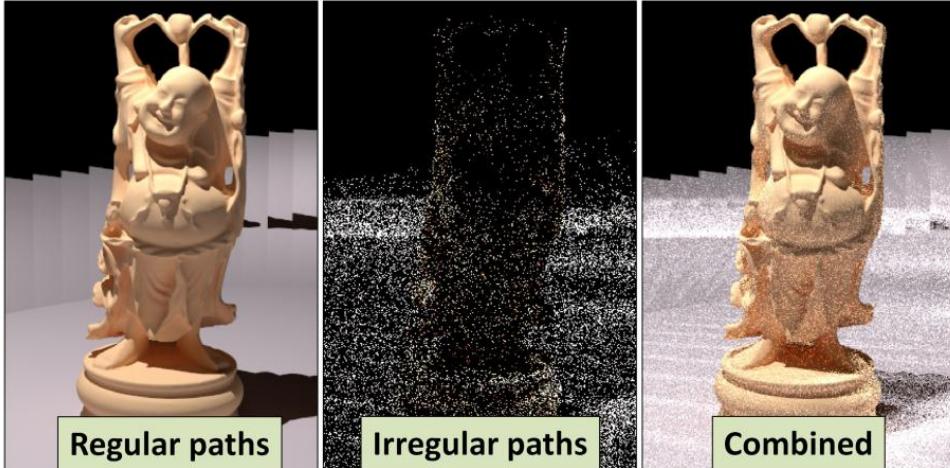
Reference



Now we will compare different popular methods with regularization.
This is a simple scene with caustics.

Comparison of Different Methods

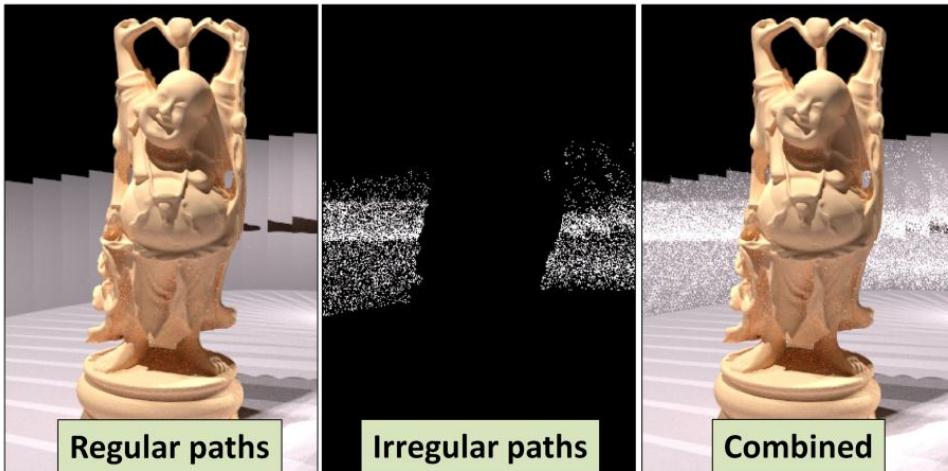
Path Tracing



PT cannot sample **both caustics and reflected caustics**, thus requiring a lot of regularization.

Comparison of Different Methods

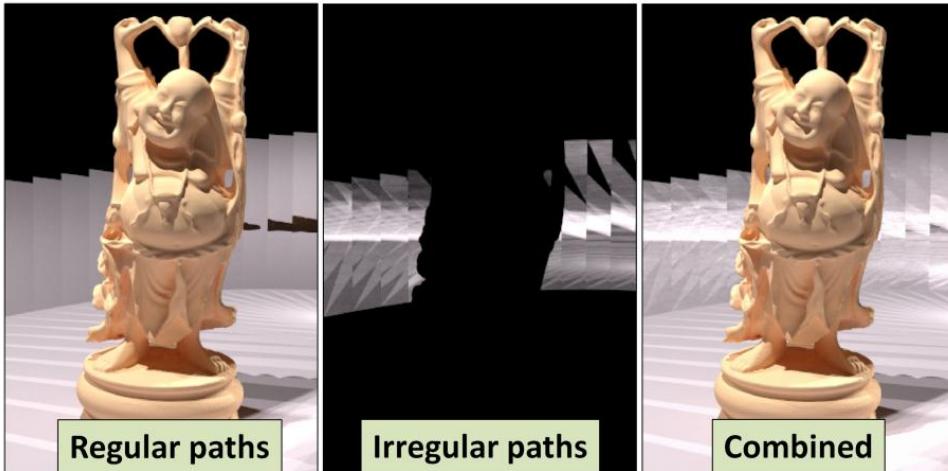
Bidirectional Path Tracing



BDPT only requires regularization when the path cannot be constructed with edge connection, thus requiring to regularize only **reflected caustics**. Note that the amount of noise caused by irregular paths is **high** because the regularization angle is small.

Comparison of Different Methods

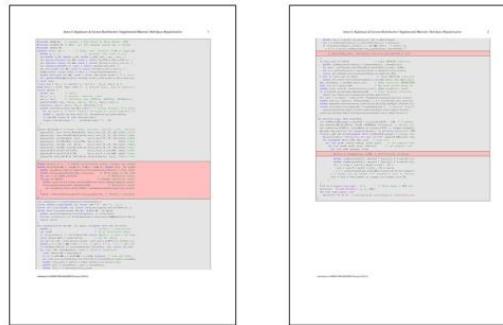
Metropolis Light Transport



Finally, MLT solves the noise problem by implicitly **caching** the path it has already found before as a current state of Markov chain.

Observations

- Ordinary MC methods (PT, BDPT, ...) need efficient caching
- Markov chain Monte Carlo (MLT, ...) resolves poor caching problem
 - Inherent caching with the current path of Markov chain
- Regularization is simple to integrate into existing renderer

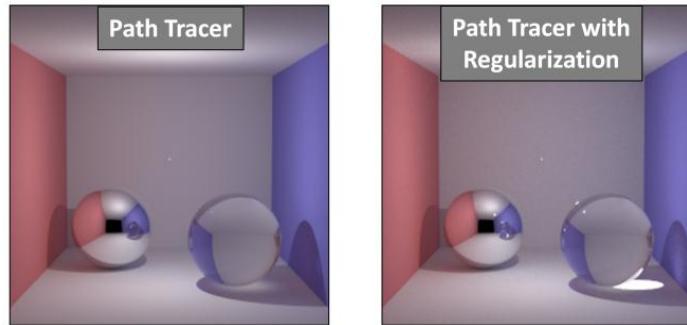


As we have seen, the ordinary Monte Carlo methods suffer from **noise**, thus are usually used with some efficient caching, such as photon map. However Markov chain based methods, such as Metropolis light transport, naturally **resolve** this issue by caching the last important path as a current state of Markov chain.

Regularization is also **simple** to implement. Here is a code of a minimalistic path tracing. Regularization requires some changes only to the **evaluation routine** for specular BRDF. The required additions are marked in **red**.

Observations

- Ordinary MC methods (PT, BDPT, ...) need efficient caching
- Markov chain Monte Carlo (MLT, ...) resolves poor caching problem
 - Inherent caching with the current path of Markov chain
- Regularization is simple to integrate into existing renderer



And you can see that with these small changes path tracing can already **handle caustics** of all kinds from point lights.

Consistency

- Converges to correct solution
 - Shrink regularization bandwidth
- Can be combined with all Monte Carlo methods (PT, BDPT, ...)
 - Consistent if bandwidth is $O(n^{-1/d}) < \alpha_n < O(1)$, details in the paper
- Can be combined with Markov chain MC (e.g., Metropolis light transport)
 - Consistent if $O(\gamma^n) < \alpha_n < O(1)$, details in the paper

In order to make sure the regularization converges to **correct solution** in the limit, we need to **decrease** the regularization angle throughout the integration.

The Monte Carlo methods have the **same shrinkage conditions** as progressive photon mapping, which is not surprising.

However the MCMC methods, such as MLT, require slightly **different rate**. Please see the **details** in the paper.

Combination with Manifold Exploration

- Enables unbiased sampling of non-sampleable paths!
 - Regularization provides local parameterization
 - Manifold exploration explores the surrounding paths
- Avoids the undecidability
 - By finding a local parameterization
 - Undecidable paths are sampled “almost surely”



Recent manifold exploration mutation can connect two vertices through a chain of specular interactions, given a **valid local parameterization** for the connection.

We deliver a method, which **almost surely** (that is, with probability one) provides the local parameterization for non-samplable paths, making the unbiased sampling of such paths **practical**.

This way we **avoid** stating undecidable problems.