Cache Based GI

Prof. Dr. Stefan Gumhold

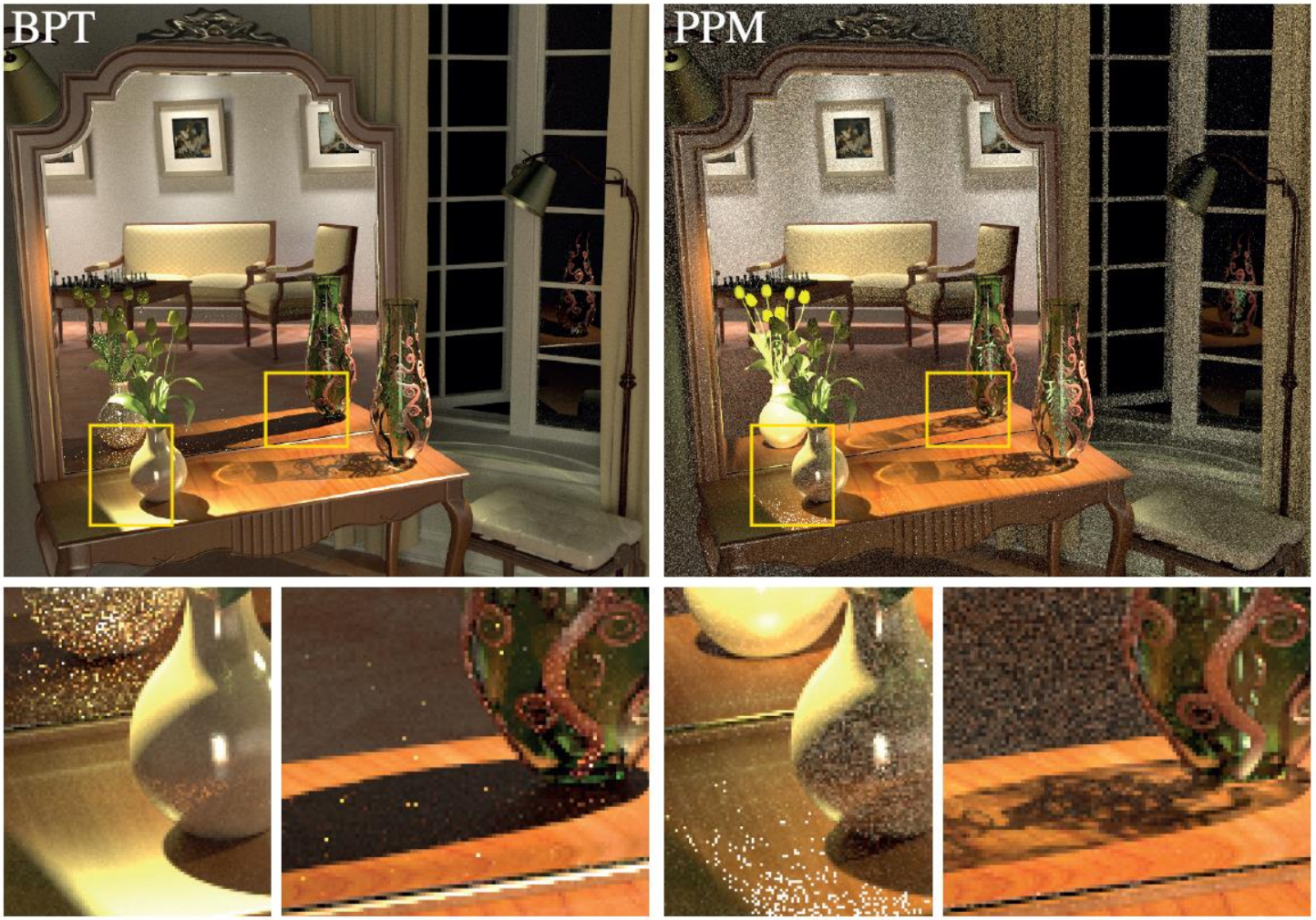
Chair of Computer Graphics and Visualization, TU Dresden

Content

MOTIVATION

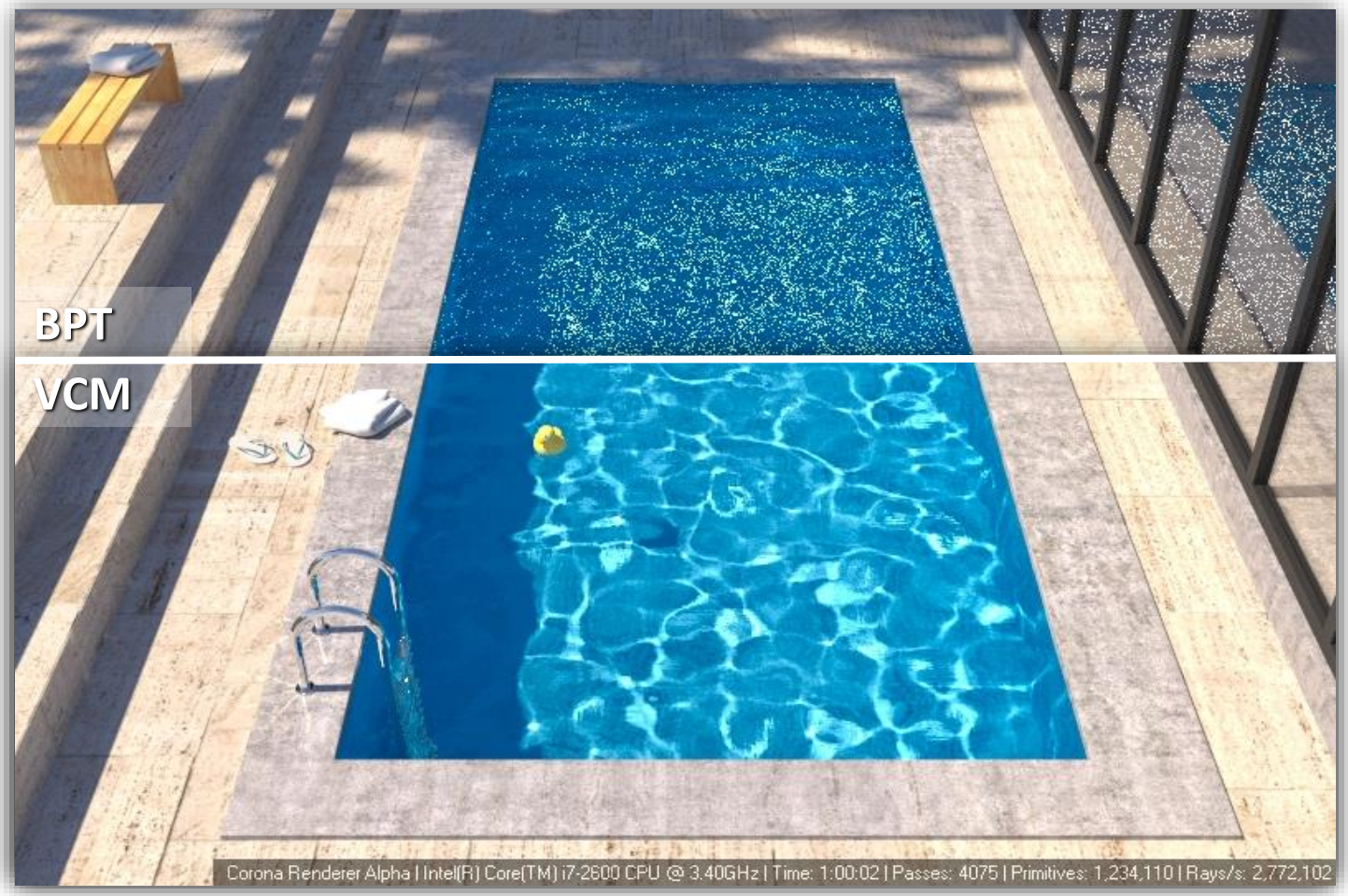
Motivation

- Bidirectional Path Tracing is very bad for caustics in mirror



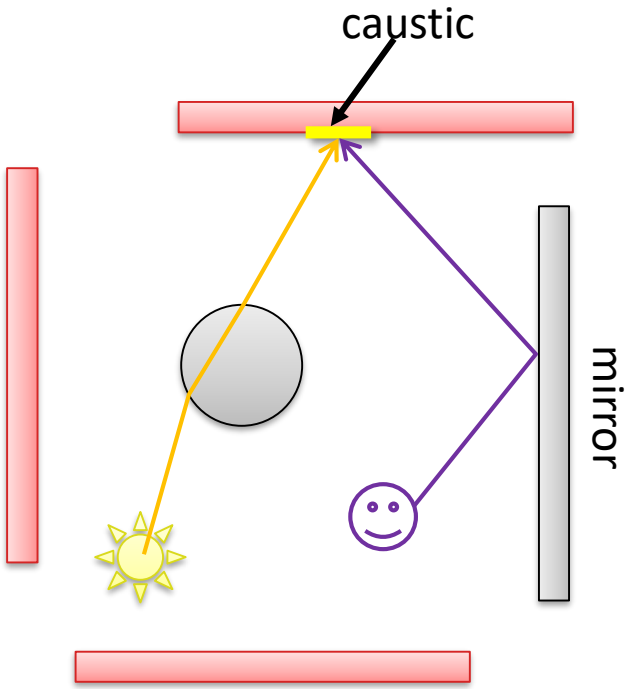
Motivation

- Bidirectional Path Tracing is very bad for caustics in water



Developed by Ondra Karlík *http://www.corona-renderer.com*

Motivation



- probability that correct light and eye paths are chosen in BPT together is extremely low
- Photon mapping: generate large number of light paths first, cache them and find for each eye path the matching ones

PHOTON MAPPING (PM)

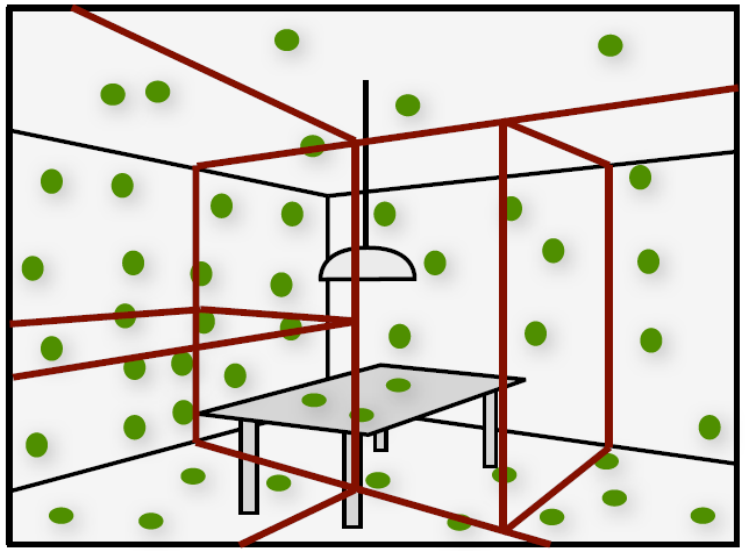
Photon Mapping Idea

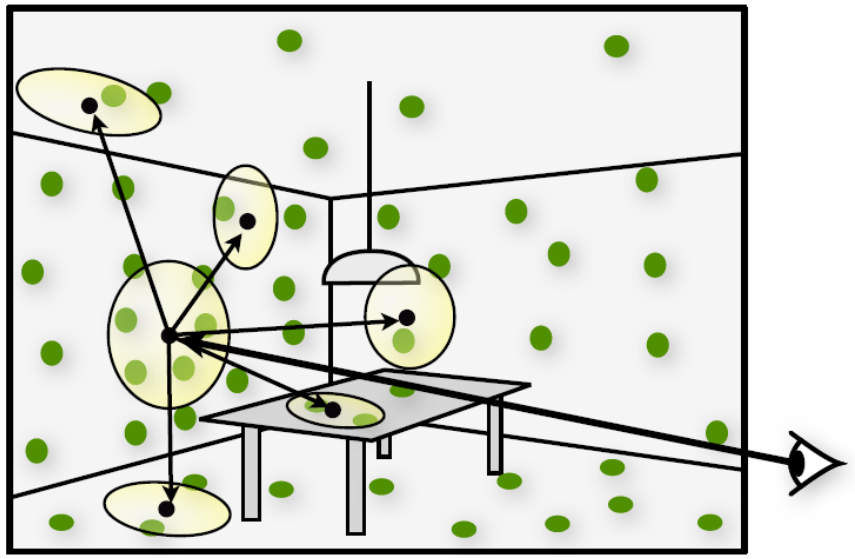
1st Pass

- emit large number of photons corresponding to light paths
- trace light paths trough purely mirroring or refracting surfaces with Russian Roulette on diffuse or glossy surfaces
- store photons on diffuse or glossy surfaces in a kd-tree

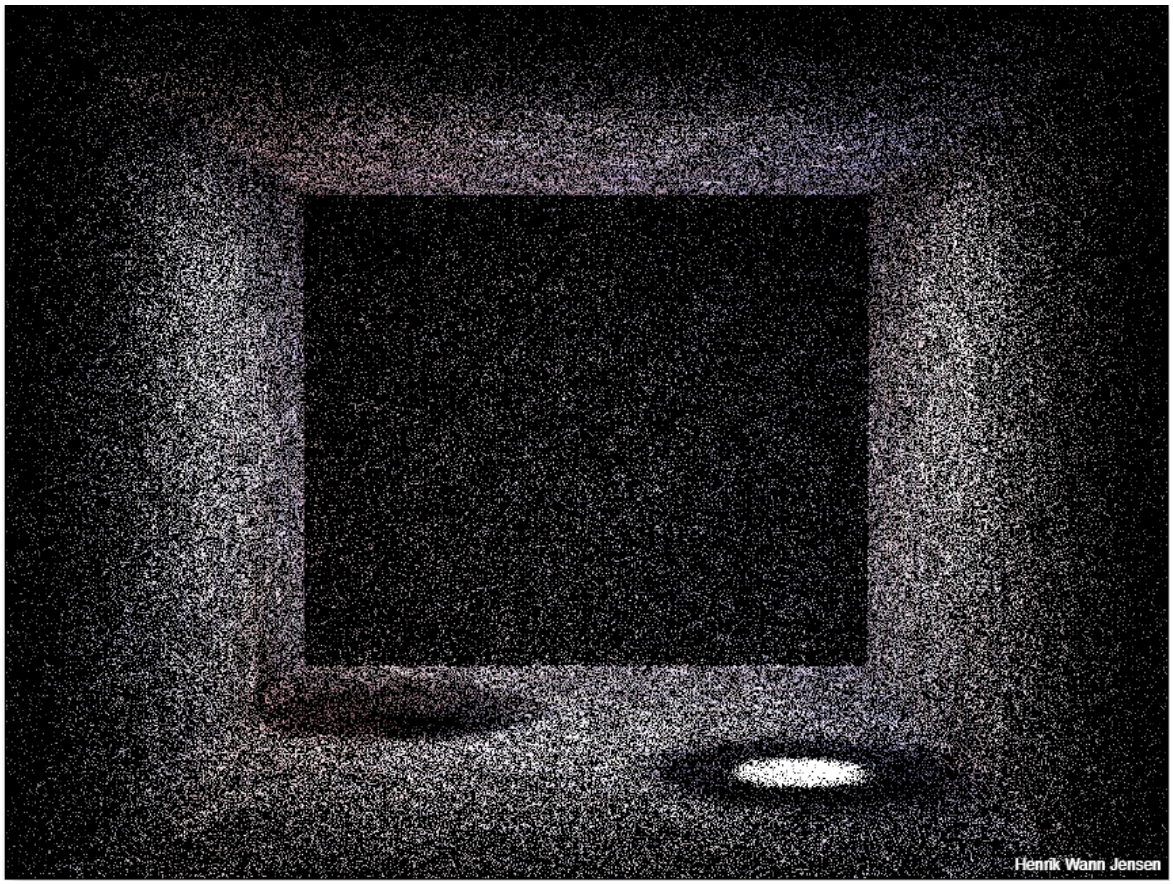
2nd Pass

- generate per pixel one or several eye paths
- trace through purely mirroring or refracting surfaces and collect at diffuse or glossy surfaces close by photons to estimate reflected radiance

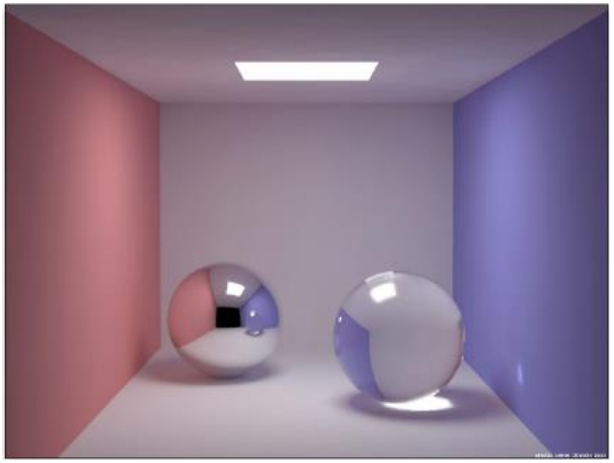




Photon Map Example



photon map



scene

photon map visualization

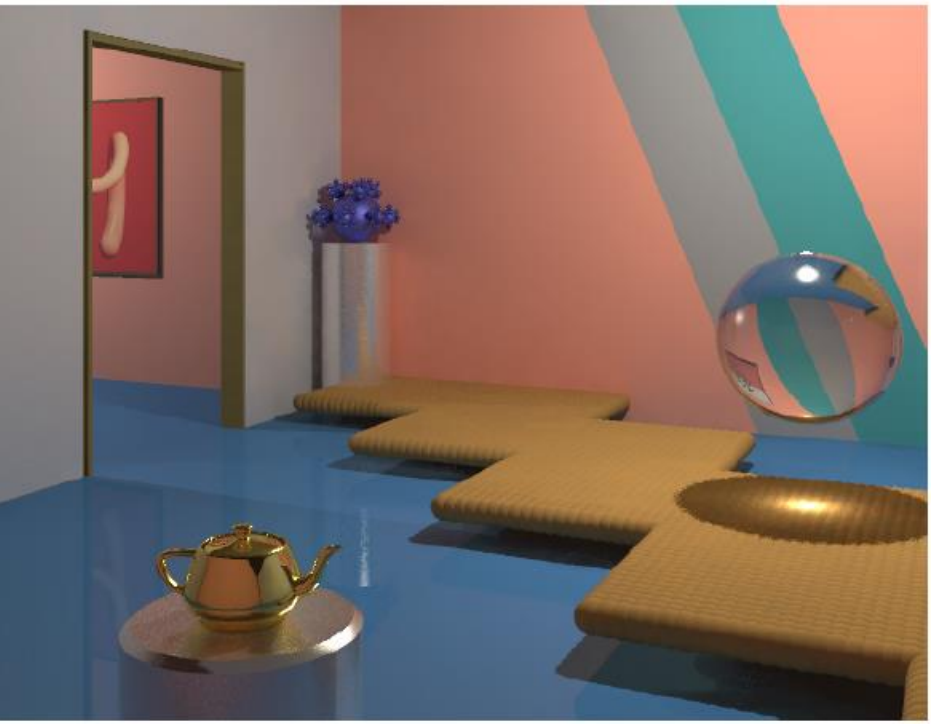
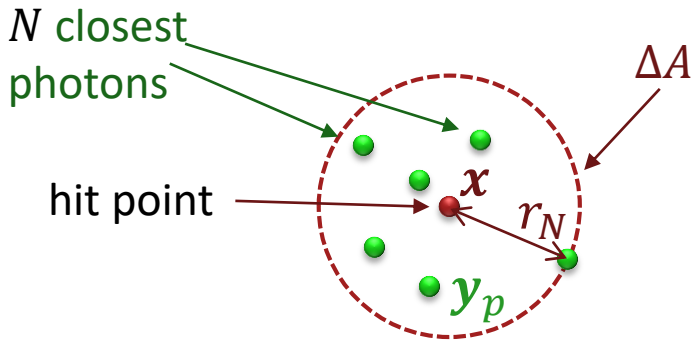


Figure 3: The Museum scene

Figure 4: Direct visualization of the global photon map in the Museum scene

Photon Mapping
Pass 2 – Radiance Estimate (a)



$$L_\lambda^{\text{reflect}}(\boldsymbol{x}, \boldsymbol{\omega}^{\text{out}}) = \iint_{\Omega^{\text{in}}} \rho\left(\boldsymbol{x}, \boldsymbol{\omega}^{\text{in}}, \boldsymbol{\omega}^{\text{out}}\right) \overbrace{L_\lambda^{\text{in}}\left(\boldsymbol{x}, \boldsymbol{\omega}^{\text{in}}\right)}^{\frac{d^2\Phi_\lambda^{\text{in}}}{dA_\perp d\Omega^{\text{in}}}} \cos\theta^{\text{in}} d\Omega^{\text{in}}$$

- collect N nearest photons with r_N being distance to farthest
- discretize once per hit point: $dA_\perp \rightarrow \Delta A = \pi r_N^2$
- discretize per photon p: $d^2\Phi_\lambda^{\text{in}} \rightarrow \Delta\Phi_p$, $\boldsymbol{\omega}^{\text{in}} \rightarrow \boldsymbol{\omega}_p^{\text{in}}$
- combine with cosine term: $\Delta\widetilde{\Phi}_p = \Delta\Phi_p \cdot \cos\theta_p^{\text{in}}$
- radiance estimate with radial filter $\lambda(r)$ (simplest approach: $\lambda(r) \equiv 1$)

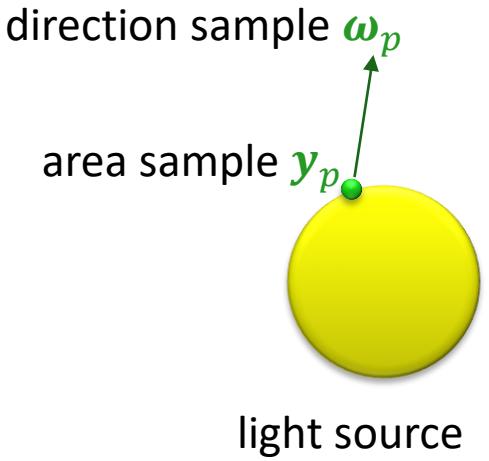
$$L_\lambda^{\text{reflect}}(\boldsymbol{x}, \boldsymbol{\omega}^{\text{out}}) = \sum_p \lambda\left(\|\boldsymbol{y}_p - \boldsymbol{x}\|\right) \rho\left(\boldsymbol{x}, \boldsymbol{\omega}_p^{\text{in}}, \boldsymbol{\omega}^{\text{out}}\right) \frac{\Delta\widetilde{\Phi}_p}{\Delta A}$$

Photon Mapping
Pass 1 – Photon Emission (a)

given pdfs $p_{\boldsymbol{y}}(.)$ and $p_{\boldsymbol{\omega}|\boldsymbol{y}}(.)$ sample light source location $\boldsymbol{y}_p$ and afterwards emission direction $\boldsymbol{\omega}_p$:

- location is sampled over light source area such that $\left[p_{\boldsymbol{y}}(.)\right] = \frac{1}{m^2}$
- direction is sampled over hemisphere such that $\left[p_{\boldsymbol{\omega}|\boldsymbol{y}}(.)\right] = \frac{1}{\mathrm{sr}}$
- to account for pdfs as in Monte Carlo integration we can compute photon power

$$\Delta\Phi_p = \frac{L_{\mathrm{emit}}(\boldsymbol{y}_p, \boldsymbol{\omega}_p)}{p_{\boldsymbol{y}}(\boldsymbol{y}_p)\, p_{\boldsymbol{\omega}|\boldsymbol{y}}(\boldsymbol{\omega}_p)}$$



Photon Mapping
Pass 1 – Photon Emission (b)

- Each $\Delta\Phi_p$ measures the total light emitted from the light source, i.e. each photon distributes the complete light into the scene.
- In the end the pixel estimates $I_j = \int_{x\in\Omega} f_j(x)d\mu(x)$ need to be normalized by dividing through the total number N_{total} of photons:

$$I_j \leftarrow \frac{I_j}{N_{\text{total}}}$$

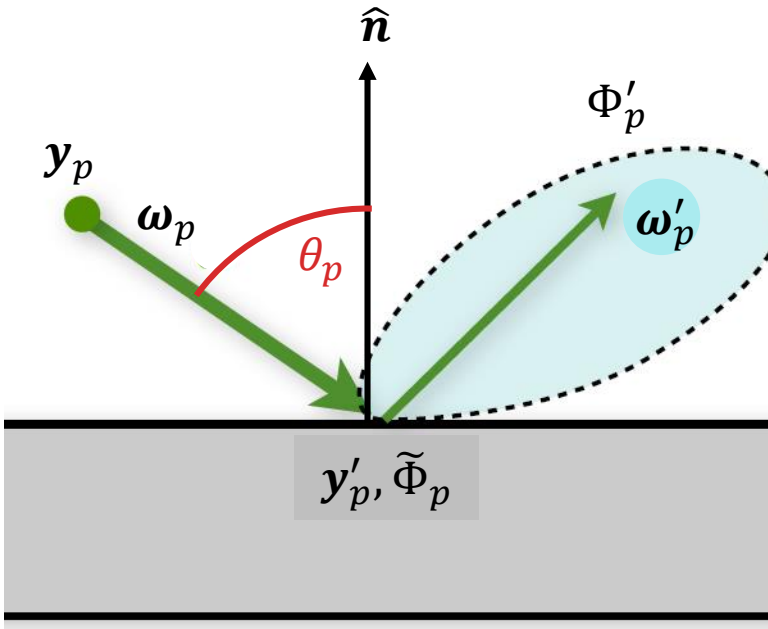
- to support several light sources, we prepend the sampling of a light source index l according to $p_l(.)$ and compute initial photon power according to

$$\Delta\Phi_p = \frac{L_{\text{emit},l}(\boldsymbol{y}_p, \boldsymbol{\omega}_p)}{p_l(l)p_{\boldsymbol{y}|l}(\boldsymbol{y}_p)p_{\boldsymbol{\omega}|\boldsymbol{y},l}(\boldsymbol{\omega}_p)}$$

Photon Mapping
Pass 1 – Photon Tracing

`TracePhoton`$(\boldsymbol{y}_p, \boldsymbol{\omega}_p, \Phi_p)$

- find surface intersection $\boldsymbol{y}'_p$
- set $\tilde{\Phi}_p = \cos\theta_p \cdot \Phi_p$
- if surface BRDF has diffuse or glossy part, store photon in photon map
- terminate if Russian Roulette with success probability s fails
- if not, sample reflected direction $\boldsymbol{\omega}'_p$ with pdf $p_{\boldsymbol{\omega}'}(.)$ according to BRDF ρ and recurs with $\boldsymbol{y}'_p$, $\boldsymbol{\omega}'_p$ and $\Phi'_p = \frac{\rho(\boldsymbol{y}_p, -\boldsymbol{\omega}_p, \boldsymbol{\omega}'_p)}{s \cdot p_{\boldsymbol{\omega}'}(\boldsymbol{\omega}'_p)} \tilde{\Phi}_p$



Photon Mapping
Pass 1 – Photon Storage

- during photon tracing photons are stored in a list with the information necessary for later radiance estimates:
  - position after surface hit $\boldsymbol{y}'_p$
  - cosine weighted power $\widetilde{\Phi}_p$ also referred to as energy
  - incoming direction $\boldsymbol{\omega}_p^{\text{in}} = -\boldsymbol{\omega}_p$
- some optimizations need storage of surface normal, previous photon on light path, light source index, and further flags
- for accelerated collection of photons during radiance estimate a kd-tree proved most simple and flexible over surface subdivision or texture maps
- kd-tree can be built inplace to avoid storage of child pointers

Photon Mapping
Pass 2 – Radiance Estimate (b)

- Jensen proposes in 1996 (Global illumination using photon maps) to use cone filter with parameter k in radiance estimate:

$$\lambda(r) = \frac{1}{1 - \frac{2}{3k}} \max\left\{0{,}1 - \frac{1}{k} \cdot \frac{r}{r_N}\right\}$$

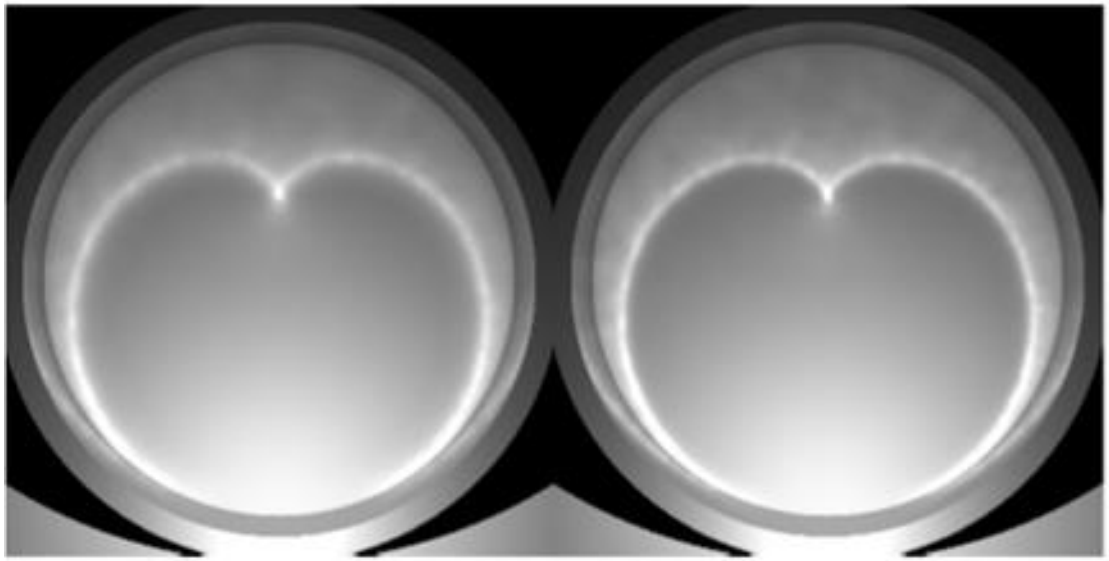
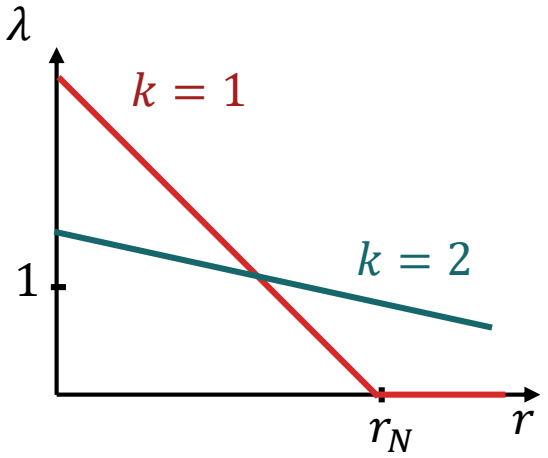


Figure 2: The effect of the cone filter. The left image is unfiltered and the right image is filtered using the cone with 1 as the filter constant



Photon Mapping
Rendering pass

Illumination splitting

- based on BRDF f_r in diffuse $f_{r,d}$ and specular $f_{r,s}$ parts, and
- based on incoming radiance L_i:
  - $L_{i,l}$ … direct lighting
  - $L_{i,c}$ … light from caustic
  - $L_{i,d}$ … indirect light after diffuse reflection

$$L_r = \int_\Omega f_r L_{i,l} \cos\theta_i \, d\omega_i + \text{A}$$
$$\int_\Omega f_{r,s}(L_{i,c} + L_{i,d}) \cos\theta_i \, d\omega_i + \text{B}$$
$$\int_\Omega f_{r,d} L_{i,c} \cos\theta_i \, d\omega_i + \text{C}$$
$$\int_\Omega f_{r,d} L_{i,d} \cos\theta_i \, d\omega_i \quad \text{D}$$

Different Approximations

$$f_r = f_{r,s} + f_{r,d} \quad \text{and} \quad L_i = L_{i,l} + L_{i,c} + L_{i,d}$$

A
- direct light sampling

B
- Monte-Carlo Ray Tracing with multiple importance sampling

C
- Radiance estimate from caustic photon map

D
- Radiance estimate from global photon map

Photon Mapping
Two Photon Maps

Caustic Photon Map

- direct use for eye rays to estimate reflected radiance
- large number of photons needed
- importance sampling of emitted photons via projection maps:
  - Sample light source positions
  - for each position generate projection map that stores object type (diffuse, specular, none) seen for bins of hemispherical directions (use (ϕ, θ) grid or hemi-cubemap)
  - project objects (or some bounding volume) onto projection map and perform depth-buffer algorithm
  - define pdf for directions such that specular objects have very high probability
- terminate paths at diffuse or glossy surface

Global Photon Maps

- only used to estimate radiance in indirect illumination (approximation of L_{in} in diffuse reflections)
- fewer photon density needed but
- support for very long paths necessary (use Russian Roulette for termination)
- Importance sampling only based on emission characteristics of light sources and BRDFs of light path points.
- do not store first diffuse surface hit photon as accounted for by direct light sampling and caustic photon map

Photon Mapping Discussion

Pros

- knn-query allows to find light paths to find LSDSE paths, which are hard to sample for BPT
- cached light paths can be reused leading to performance increase
- generalization to volumetric effects simple
- additional optimization potential
  - speed up light source visibility test with shadow photons
  - use photon map for importance sampling according to

$$\rho \cdot L_{\text{in}} \cdot \cos\theta$$

Cons

- density estimation introduces bias through interop-lation (visible artefacts avoided by final gathering)

Figure 4: Direct visualization of the global photon map in the Museum scene

1996 Jensen: Global illumination using photon maps

Scene	Resolution	Caustic photons	Global photons	Pass 1	Rendering
Diffuse Cornell Box	1280x960	21.162	286.489	67 sec.	8 min
Diffuse Cornell Box [**R**]	1280x960	-	-	-	60 min
Glossy Cornell Box	2560x1920	0	382.598	56 sec.	50 min
Glossy Cornell Box [**R**]	5120x3840	-	-	-	360 min
The Museum	1280x960	389.755	165.791	298 sec.	51 min
The Cognac Glass	1280x960	224.316	3095	27 min.	65 min

Table 1: Rendering statistics. [**R**] indicates the images rendered with Radiance

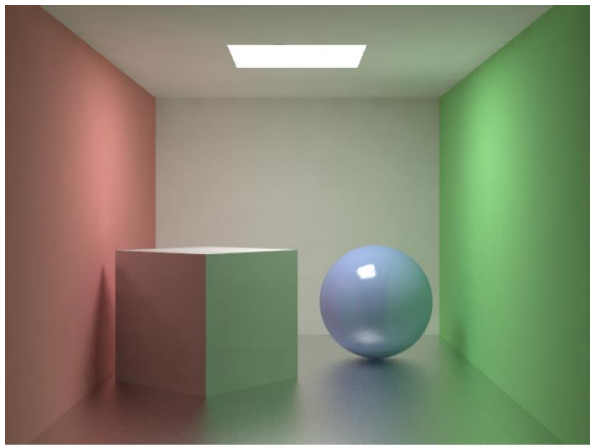


Figure 6: The glossy Cornell box rendered with photon maps

Figure 3: The Museum scene

Figure 5: A Cognac glass on a fractal surface

REVERSE PHOTON MAPPING

Reverse photon mapping

Observation: in high quality GI there are many more final gathering rays needed than photons

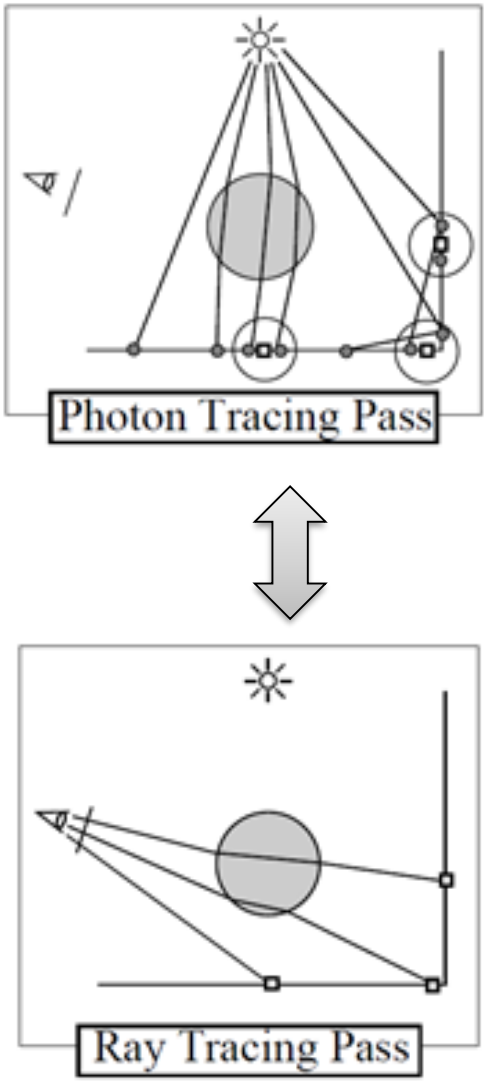
- number of final gathering rays: $\#_{\mathrm{FGR}}$
- number of photons: $\#_{\mathrm{PH}}$

- Photon mapping runtime T_{PM} is dominated by knn-query for photon based radiance estimation:

$$T_{\mathrm{PM}} = O(\#_{\mathrm{FGR}} \log \#_{\mathrm{PH}})$$

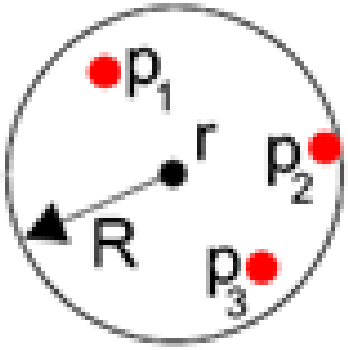
- A speedup of a factor up to 6 can be achieved by first tracing eye paths to diffuse or glossy surfaces and storing them as so called reverse photons:

$$T_{\mathrm{PM}} = O(\#_{\mathrm{PH}} \log \#_{\mathrm{FGR}})$$

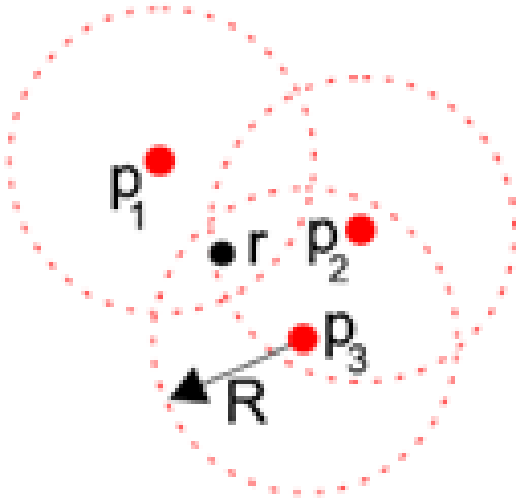


Reverse photon mapping

- For speedup we need to collected reverse photons per photon and for this we need to estimate influence radii per photon
- Havran proposed 2005 to estimate photon influence radii by first distributing all photons and using one knn query per photon with runtime: $O(\#_{\mathrm{PH}} \log \#_{\mathrm{PH}})$
- Photons are iterated again in order to distribute photon power on reverse photons within influence radii in $O(\#_{\mathrm{PH}} \log \#_{\mathrm{FGR}})$



3 nearest neighbor photons define density estimation radius in forward photon mapping

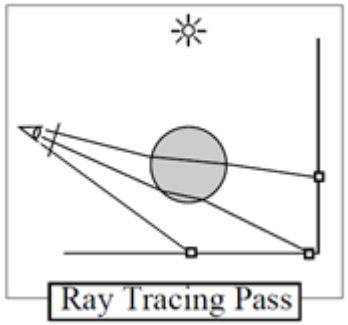


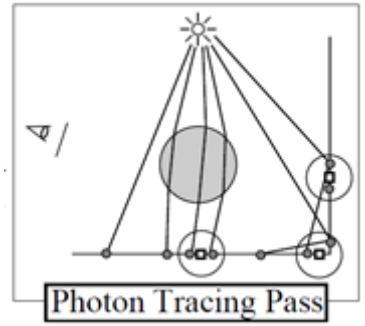
Each photon influences reverse photons that are within influence radius

Reverse photon mapping

Reverse photon mapping algorithm:

- Eye pass
  - for each pixel generate eye paths and store reverse photons
  - build kd-search tree over reverse photons
- Light pass
  - generate photons
  - build kd-tree over photons and estimate influence radii from leaf diagonal and leaf photon count
  - Iterate photons memory coherently and distribute energy to reverse photons within photon influence radius (kernel density estimation)

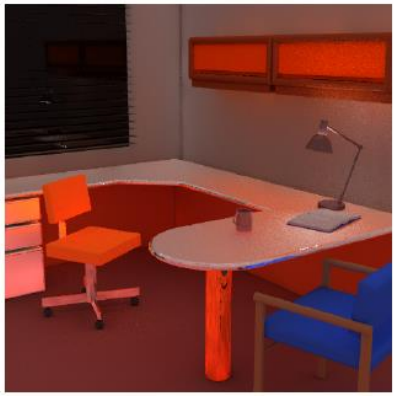




Reverse photon mapping

Cornell Box

Corner Room



MGF Office

Gallery

Sponza

Apartement

Sibenik

Aizu Atrium

Example scenes used in Havran et al., *Fast final gathering via reverse photon mapping*, 2005 where reverse photon mapping achieved speedups of factors up to 6

PROGRESSIVE PHOTON MAPPING (PPM)

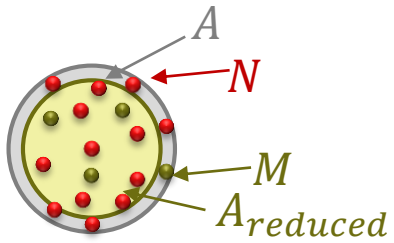
Progressive Photon Mapping (PPM)

- Photon mapping (PM) is biased through kernel density estimation (KDE) step resulting in an error in the result even for infinitely many photons
- In limit for infinitely many photons **and** zero KDE radius method becomes unbiased. Such convergent methods are called *consistent*
- Progressive Photon Mapping iterates PM with decreasing density estimation radius such that error converges to zero in limit
- The central questions are
  - How to update the radius after each iteration?
  - How to store KDE radius that can be updated per iteration?
  - How to combine the results of photon mapping iterations to final result?

Progressive Photon Mapping (PPM)

- There are three important PPM approaches:
  - 2008 Hachisuka: Progressive Photon Mapping (PPM)
  - 2009 Hachisuka: Stochastic PPM (SPPM)
  - 2011 Knaus: PPM - A Probabilistic Approach (probPPM)
- All methods work with the same radius update:
  - Let N be the total number of photons generated in all previous iterations and M the number of photons generated in current iteration (constant over all iterations i such that $N = i \cdot M$)
  - One defines a user adjustable fraction $\alpha \in [0,1]$ of photons to be considered from the M new ones in the estimation with reduced radius
  - Assuming a low varying photon density the radius can be updated according to

$$\frac{r_{i+1}^2}{r_i^2} = \frac{A_{reduced}}{A} = \frac{N + \alpha M}{N + M} \left(= \frac{iM + \alpha M}{iM + M} = \frac{i + \alpha}{i + 1} \right)$$



Progressive Photon Mapping (PPM)

- **Idea:** use reverse photons to store KDE radius & accumulate radiance estimates. Multiply with throughput of reverse photon to get pixel contribution

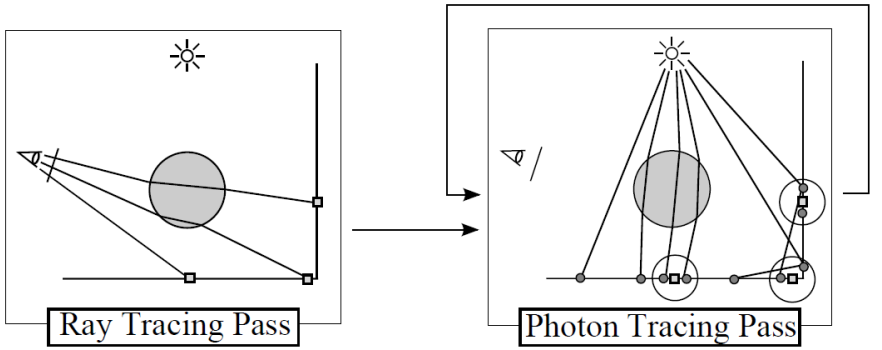
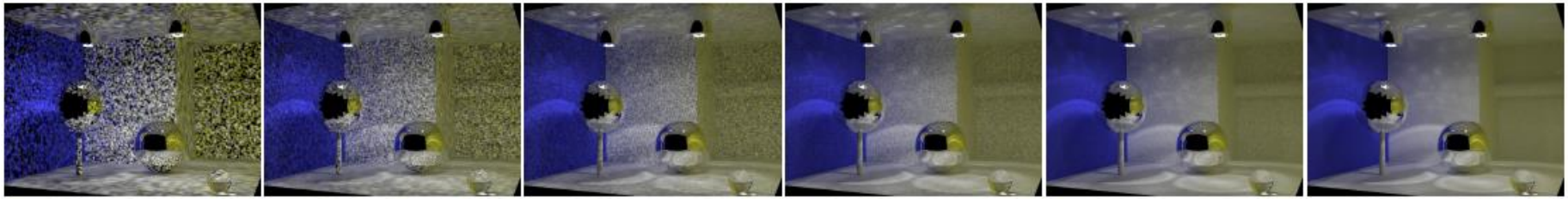


Figure 2: *Progressive photon mapping uses ray tracing in the first pass followed by one or more photon tracing passes.*

Approach

- generate reverse photon map (also store throughputs)
- compute photon map with M photons & estimate KDE radius r_q per reverse photon q (iteration $i = 1$)
- iterate $i = 2, \ldots$
  - compute M photons & update KDE radii $r_q \leftarrow r_q \sqrt{\frac{(i-1)+\alpha}{i}}$
  - per photon estimate influence radius from KDE radius of closest reverse photons and add photon contribution to accumulated radiances in influenced reverse photons
  - to compute current estimate multiply accumulated radiances with throughput of reverse photon and divide summed pixel contributions by $i \cdot M$

PPM results



0.1M photons | 0.4M photons | 1.6M photons | 6.4M photons | 25.6M photons | 102.4M photons

Figure 5: *Sequence of images with increasing number of photons. The images show a direct visualization the progressive radiance estimate after 1, 16, 64, 256, and 1024 photon tracing passes. Note, how the illumination is quite good already after 0.1 million photons. Adding more photons makes the image sharper and reduces low frequency noise.*

	PT Samples	BDPT Samples	MLT Mutations	PPM Iterations	Time Hours
Lamp	840	80	82	1651	22
Box	1428	155	132	2134	4
Torus	1050	550	359	520	2
Bathroom	675	66	66	6126	16

Table 1: *Rendering statistics for the different scenes. The samples and mutations numbers are the average number per pixel. PPM uses 100000 photons per iteration.*

PPM results

Figure 8: *Lighting simulation in a bathroom. The scene is illuminated by a small lighting fixture consisting of a light source embedded in glass. The illumination in the mirror cannot be resolved using Monte Carlo ray tracing. Photon mapping with 20 million photons results in a noisy and blurry image, while progressive photon mapping is able to resolve the details in the mirror and in the illumination without noise.*

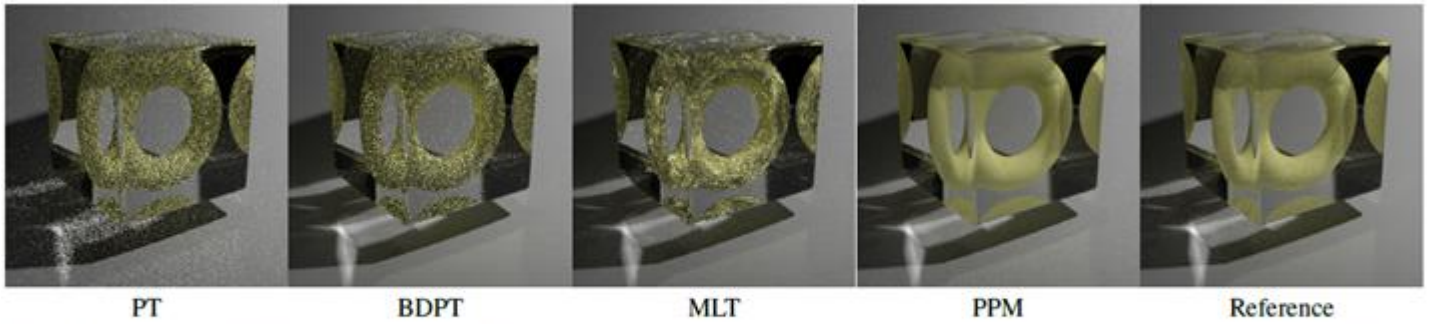


Figure 7: *Torus embedded in a glass cube. The reference image on the far right have been rendered using path tracing with 51500 samples per pixel. The Monte Carlo ray tracing methods fail to capture the lighting within the glass cube, while progressive photon mapping provides a smooth result using the same rendering time.*

Stochastic PPM results

Source code available at
www.luxrender.net/wiki/SPPM

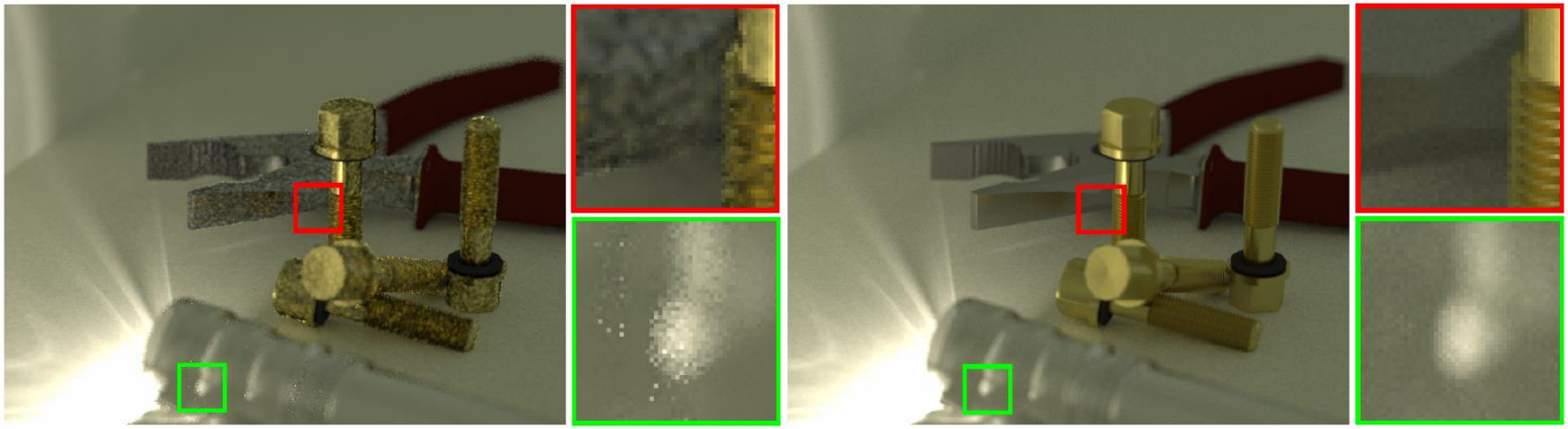


Figure 1: *Tools with a flashlight. The scene is illuminated by caustics from the flashlight, which cause SDS paths on the flashlight and highly glossy reflections of caustics on the bolts and plier. The flashlight and the plier are out of focus. Using the same rendering time, our method (right) robustly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

Progressive Photon Mapping

SPPM

$r_i \rightarrow r_i(S)$

- store distribution averages of attributes like KDE radius and accumulated radiance over *unknown scene area S*
- start with generation of one reverse photon map
- compute first photon map to estimate initial KDE radius avg.
- iterate photon map generation & distributed ray tracing pass sampling reverse photon positions in *unknown scene area S*
- same update rules

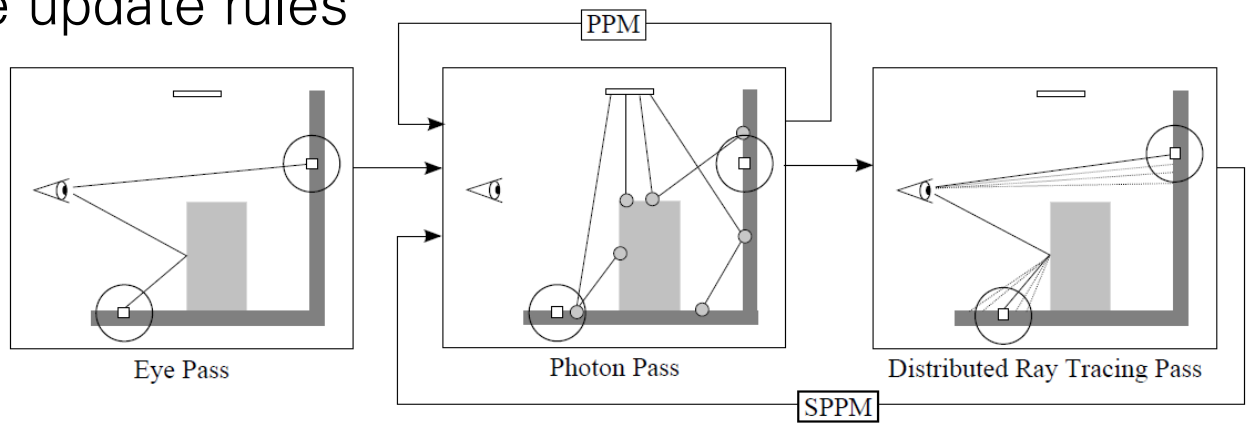


Figure 2: *Difference between the algorithms of progressive photon mapping (PPM) and stochastic progressive photon mapping (SPPM). In order to compute the average radiance values, SPPM adds a new distributed ray tracing pass after each photon tracing pass. The photon tracing algorithm itself stays the same, but PPM uses a fixed set of hit points (shown as squares), whereas SPPM uses randomly generated hit points by the distributed ray tracing pass.*

Stochastic PPM results

	Triangles	Rendering time [min]	PPM passes	SPPM passes
Cornell Box (Figure 4)	38	50	899	742
Cornell Box with Wall lights (Figure 4)	7660	50	234	220
Furry Bunny (Figure 6)	371247	132	1722	1197
Transparent Dices (Figure 6)	370860	110	287	195
Alarm clocks (Figure 7)	119856	480	1101	1202
Tools (Figure 1)	56486	200	353	484

Table 1: *Rendering statistics of our experiments. We show a single rendering time for each scene because both PPM and SPPM used the same rendering time. PPM passes are the numbers of photon tracing passes, and SPPM passes are the numbers of photon tracing passes as well as the numbers of distributed ray tracing passes. We used 500,000 emitted photons per pass in both methods. SPPM usually performs a less number of passes in the same rendering time because of the additional cost of the distributed ray tracing pass. However, SPPM can perform more passes in the alarm clocks scene and the tools scene, where PPM needed to use multiple hit points per pixel (16 hit points per pixel) to achieve distributed ray tracing effects.*

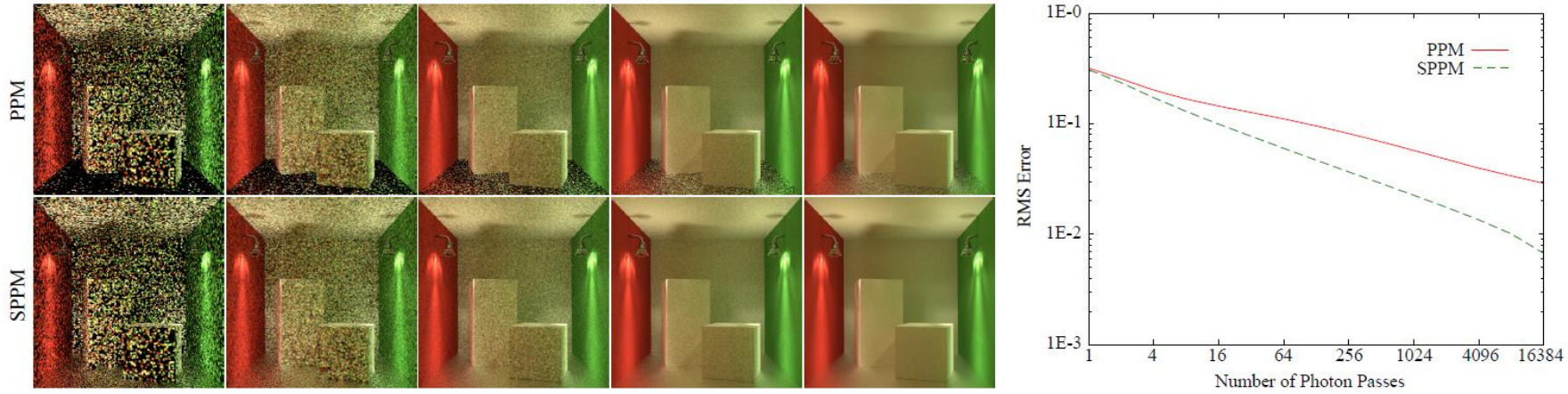
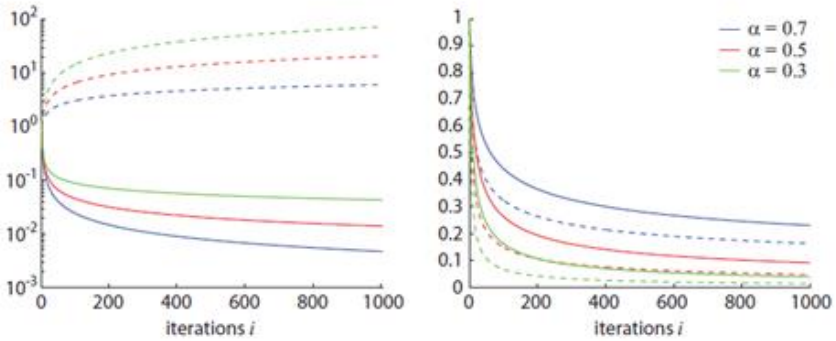


Figure 5: *Progressive sequences of rendering of Cornell box and the RMS errors. The number of photon passes of the images is 1, 8, 64, 512, and 4096 correspondingly. The graph shows the RMS errors from the converged result with PPM. The result with SPPM converges visually pleasing results with a smaller number of photon passes, and the RMS errors are consistently lower than the result with PPM. The rendering time of SPPM is approximately 10 percent longer than that of PPM for the same number of photon passes.*

PPM – A Probabilistic Approach (2011 Knaus)

probPPM

- Use globally const or per pixel KDE radius stored with pixel
- Iterate standard photon mapping with the stored KDE radii and update the radii after each iteration
- combine per iteration results by adding up per pixel radiances and dividing by total number of photons
- major new contribution: proof of consistency (convergence)



(a) Variance of error Var[ε_i] (dotted), and variance of its average Var[$\bar{\varepsilon}_i$]

(b) Expected error E[ε_i] (dotted), and expected value of its average E[$\bar{\varepsilon}_i$]

Fig. 2: *This figure summarizes (a) the behavior of the variance of each iteration* $\mathrm{Var}[\epsilon_i]$ *and the variance of the average* $\mathrm{Var}[\bar{\epsilon}_i]$. *Note that* $\mathrm{Var}[\bar{\epsilon}_i]$ *denotes the variance of the average over the first* i *iterations. Figure (b) shows the expected error of each iteration* $\mathrm{E}[\epsilon_i]$ *and the expected value of the average error* $\mathrm{E}[\bar{\epsilon}_i]$ *(again over the first* i *iterations). We plot the relative change compared to an initial variance* $\mathrm{Var}[\epsilon_1]$ *and expected error* $\mathrm{E}[\epsilon_1]$.

2011 Knaus: PPM - A Probabilistic Approach
error expectation and variance analysis

Computergraphik und Visualisierung

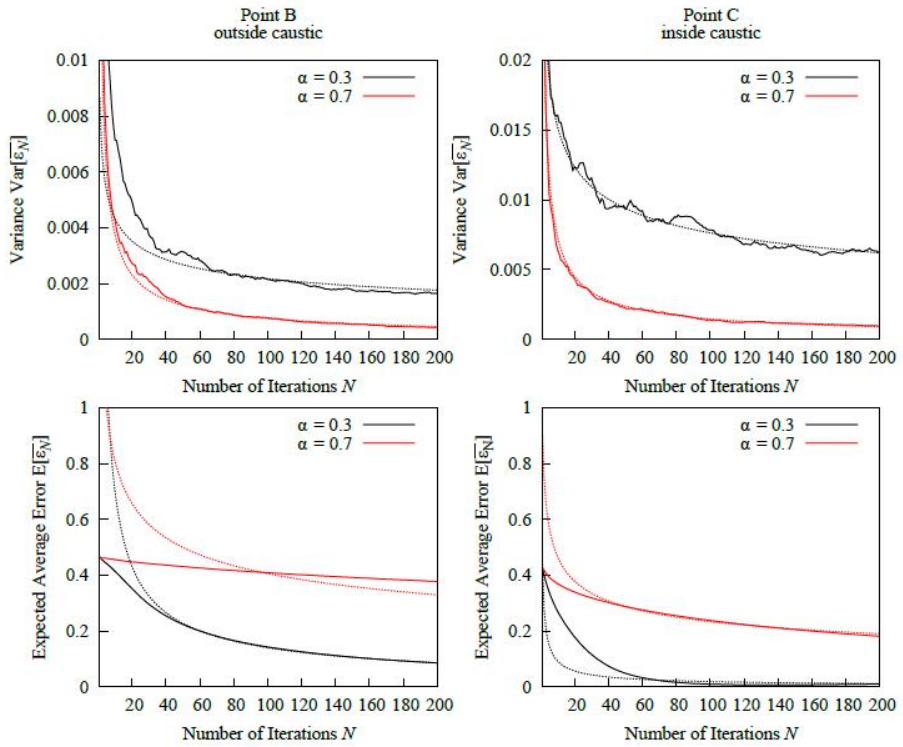


Fig. 4: *Vanishing behavior of variance and expected error for pixels B and C in Figure 5. The plots in the top row show the variance* $\mathrm{Var}[\bar{\epsilon}_N]$ *and the plots in the bottom row show the expected average error* $\mathrm{E}[\bar{\epsilon}_N]$ *as a function of the number of iterations* N*. We plot the empirically estimated variance and expected average error (solid lines) and the asymptotic behavior of the variance* $O(1/N^{\alpha})$ *and expected average error* $O(1/N^{1-\alpha})$ *(dotted lines). We show results for* $\alpha = 0.3$ *and* $\alpha = 0.7$*, using 100'000 photons in each iteration step. The reference radius is determined as the* kNN *distance in the photon maps with* $k = 100$*.*

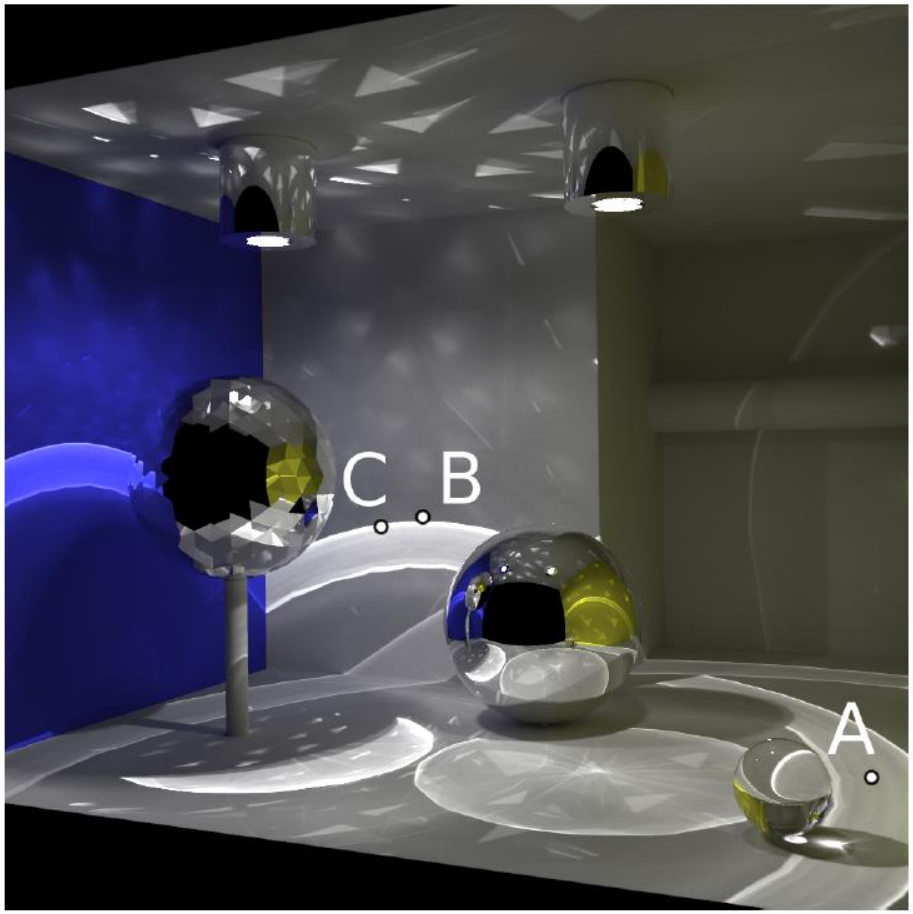


Fig. 5: *This figure shows three characteristic points sampled for the statistics in Figure 4 and 6. Point A lies in a homogenous region, points B and C lie just outside and just inside the caustic.*

ProbPPM Comparison of Kernels

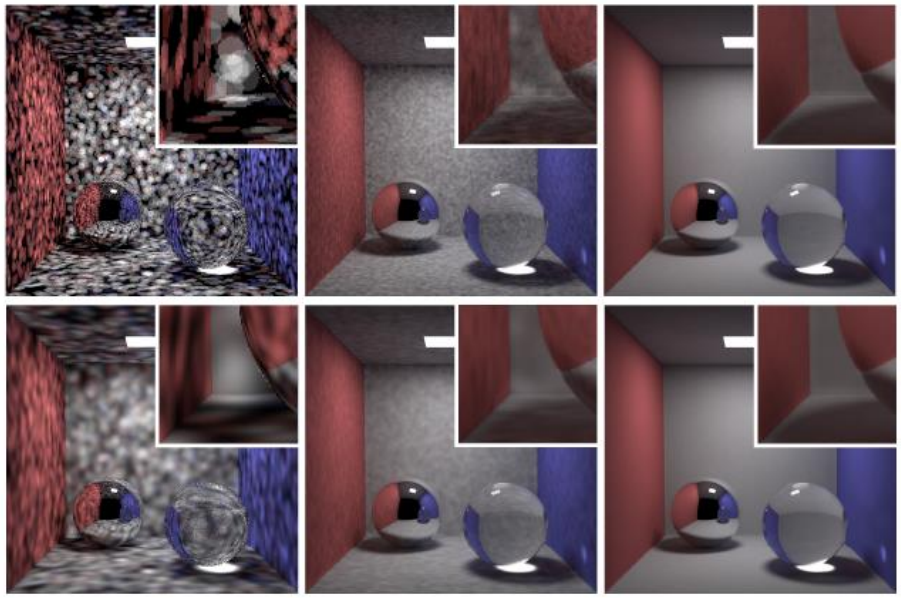


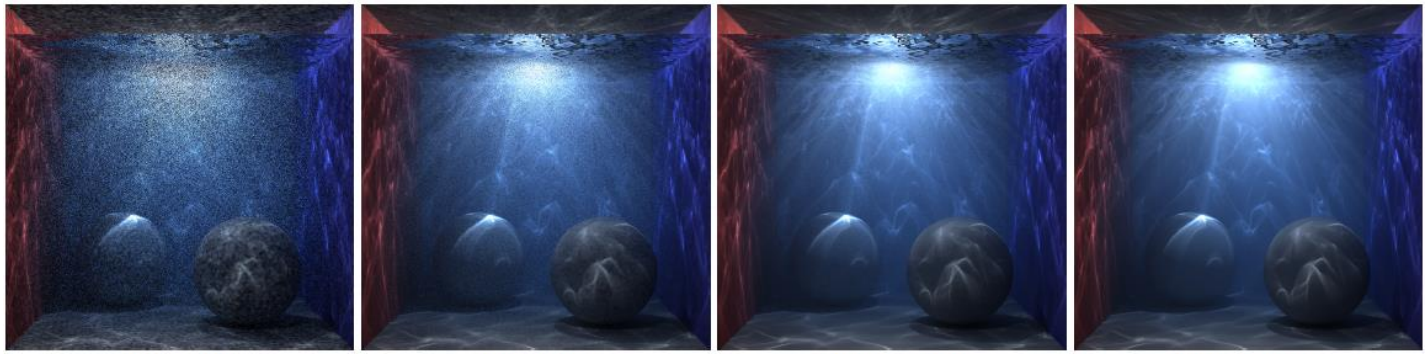
Fig. 8: *We compare rendering results using a box kernel (top row) and a Gaussian kernel (bottom row). From left to right, the images were rendered using 1, 100, and 10'000 iterations. We used only 10'000 photons per iteration to emphasize the difference between the kernels. The standard deviation of the Gaussian kernel is equal to the radius of the box kernel.*

Fig. 9: *We show the rendering progress using radiance estimates on a glossy surface. The images were rendered using 1, 10, 100, 1000, 10'000, and 100'000 iterations, with 1 million photons per iteration. The last image was rendered using 100 billion photons.*

ProbPPM results

Fig. 10: *The diamonds are rendered using 10 billion photons with dispersion and depth of field.*

Fig. 11: *The clocks are rendered using 200 million photons on a glossy surface and with depth of field.*



1 iteration
2 million photons

10 iterations
20 million photons

100 iterations
200 million photons

1000 iterations
2 billion photons

Fig. 12: *We demonstrate our approach with volumetric photon mapping. We trace two million photons per iteration. We obtain a result with little noise and sharp caustics after several hundred iterations and tracing more than a billion photons in total.*

VERTEX CONNECTION AND MERGING (VCM)

2012 Georgiev: Light Transport Simulation with Vertex Connection and Merging

Source code available at
www.smallvcm.com

Slides available at
http://cgg.mff.uni.cz/~jaroslav/papers/2012-vcm/

- combines stochastic PPM with bidirectional path tracing such that advantages of both methods are exploited
- first PPM is reformulated in formalism of BPT and then integrated via multiple importance sampling

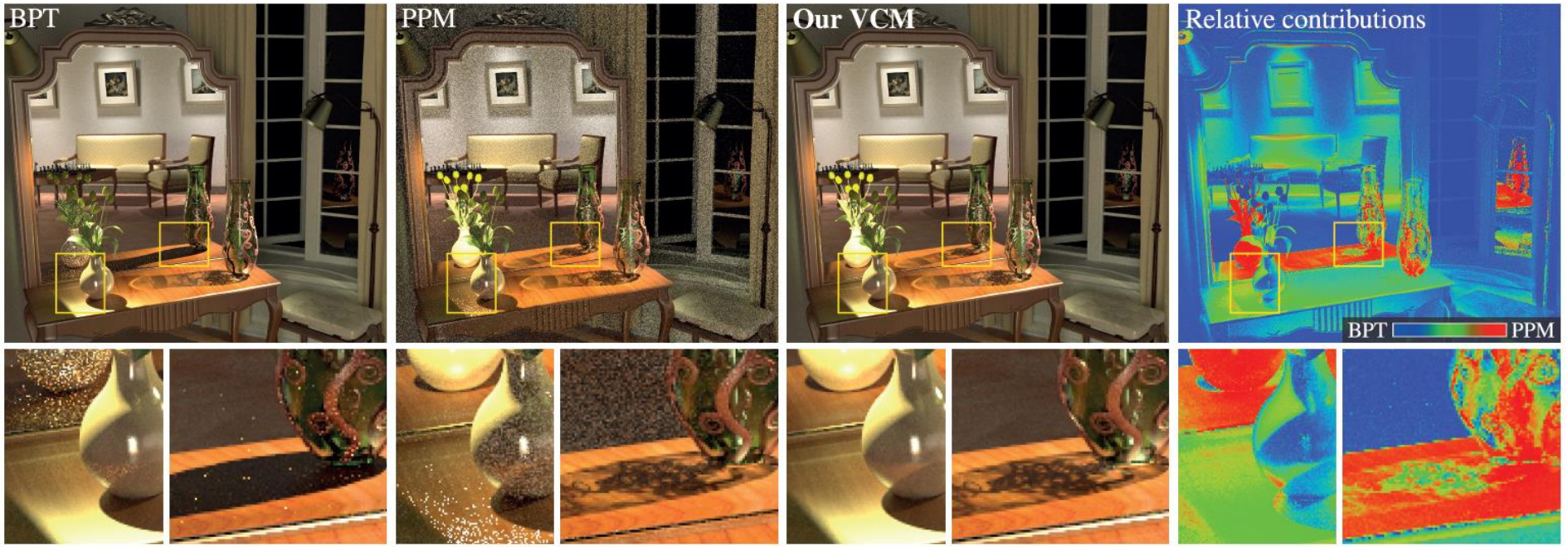
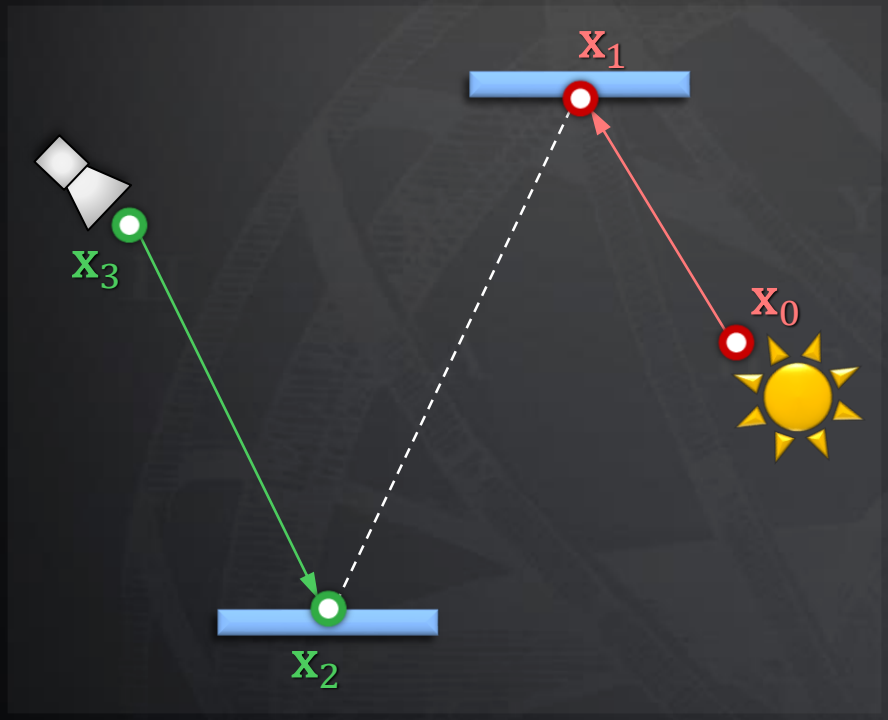


Figure 1: *A comparison of our new progressive vertex connection and merging (VCM) algorithm against bidirectional path tracing (BPT) and stochastic progressive photon mapping (PPM) after 30 minutes of rendering. BPT fails to reproduce the reflected caustics produced by the vase, while PPM has difficulties in handling the illumination coming from the room seen in the mirror. Our new VCM algorithm automatically computes a good mixture of sampling techniques from BPT and PPM to robustly capture the entire illumination in the scene. The rightmost column shows the relative contributions of the BPT and PPM techniques to the VCM image in false color.*

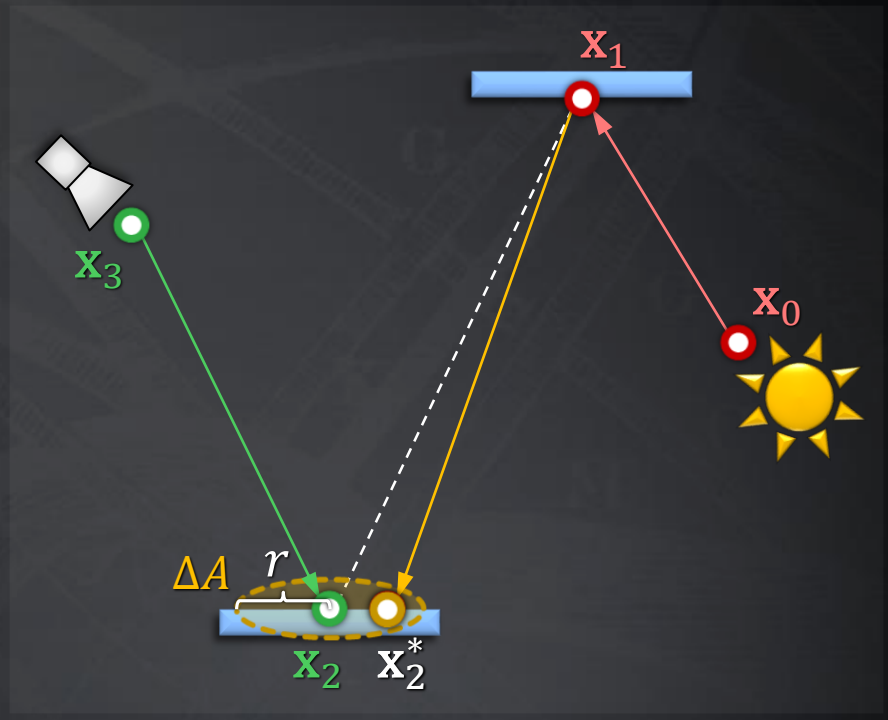
MIS Formulation of Vertex Merging

Light vertex
Camera vertex



BPT → vertex Connection (VC)

$$p_{VC}(\bar{\mathbf{x}}) = p(\mathbf{x}_0)p(\mathbf{x}_0 \to \mathbf{x}_1)\, p(\mathbf{x}_3)p(\mathbf{x}_3 \to \mathbf{x}_2)$$

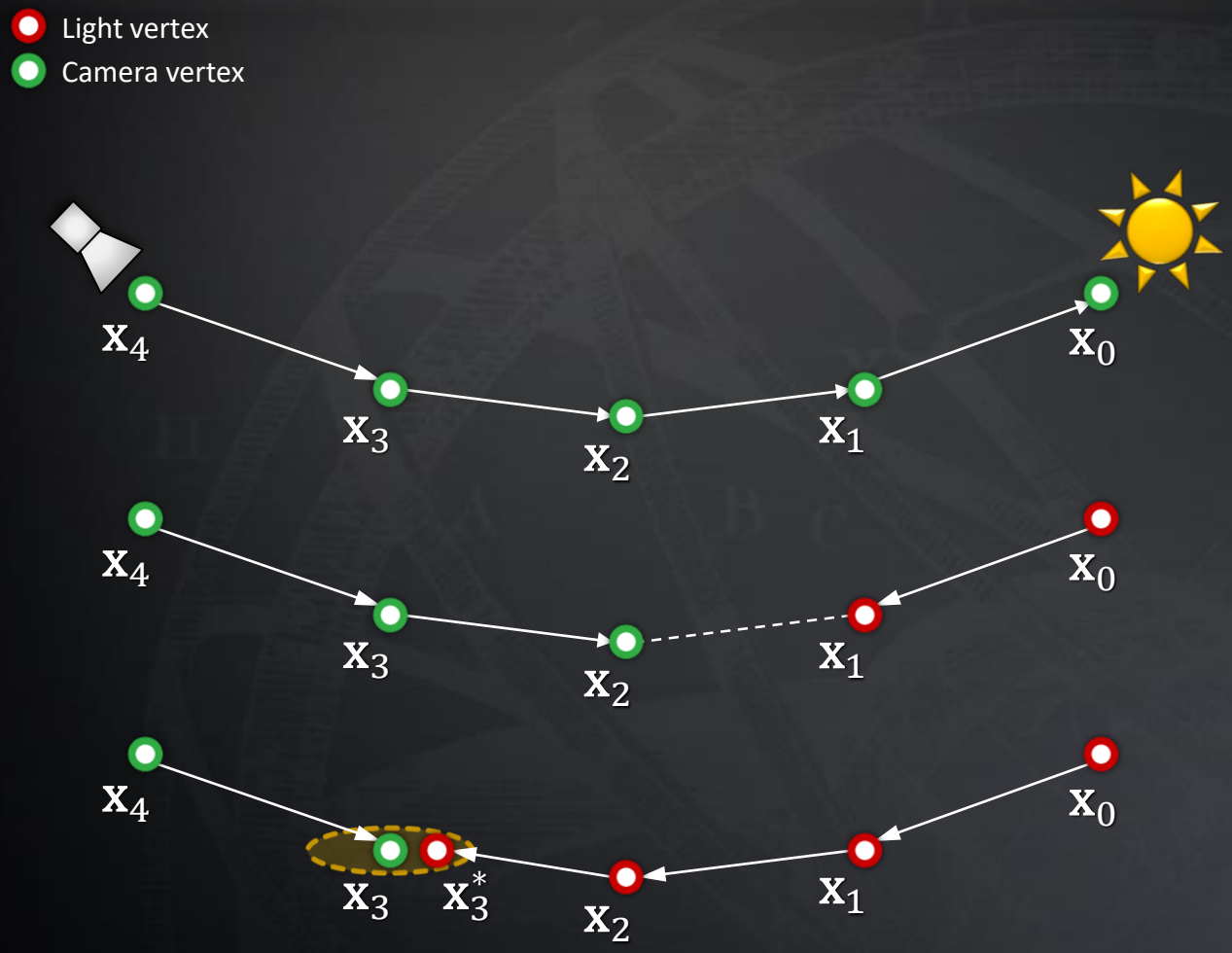


PM → Vertex Merging (VM)

$$p_{VM}(\bar{\mathbf{x}}) \approx p(\mathbf{x}_0)p(\mathbf{x}_0 \to \mathbf{x}_1)\, p(\mathbf{x}_3)p(\mathbf{x}_3 \to \mathbf{x}_2)\, \underbrace{p(\mathbf{x}_1 \to \mathbf{x}_2^*) \cdot \pi r^2}$$

merging probability that is constant in ΔA and 0 outside

MIS Sampling techniques



Unidirectional	2 ways
Vertex connection	4 ways
Vertex merging	**5 ways**
Total	**11 ways**


Bidirectional path tracing (30 min)


Stochastic progressive photon mapping (30 min)


Vertex connection and merging (30 min)


VM
VC
Relative technique contributions

Real-Time Techniques

IMAGE SPACE PHOTON MAPPING (ISPM)

Image Space Photon Mapping

McGuire, M., & Luebke, D. (2009, August). Hardware-accelerated global illumination by image space photon mapping. In Proceedings of the Conference on High Performance Graphics 2009 (pp. 77-89), DOI

Direct Illumination Only | Direct + Constant Ambient | Image Space Photon Mapping

Source Code inside of G3D engine: https://casual-effects.com/g3d
Lots of Scenes in Repo: http://svn.code.sf.net/p/g3d/code/

Image Space Photon Mapping

key observations:

1. The **initial bounce is expensive**: the initial photon-tracing segment from the emitter to the first bounce must trace the most photons.
2. The **final bounce is expensive**: the final segment to the camera requires the most computation because the algorithm must sum over many photons at each pixel.

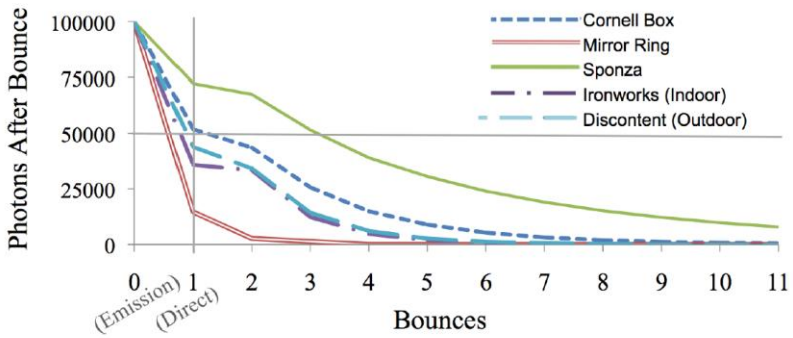
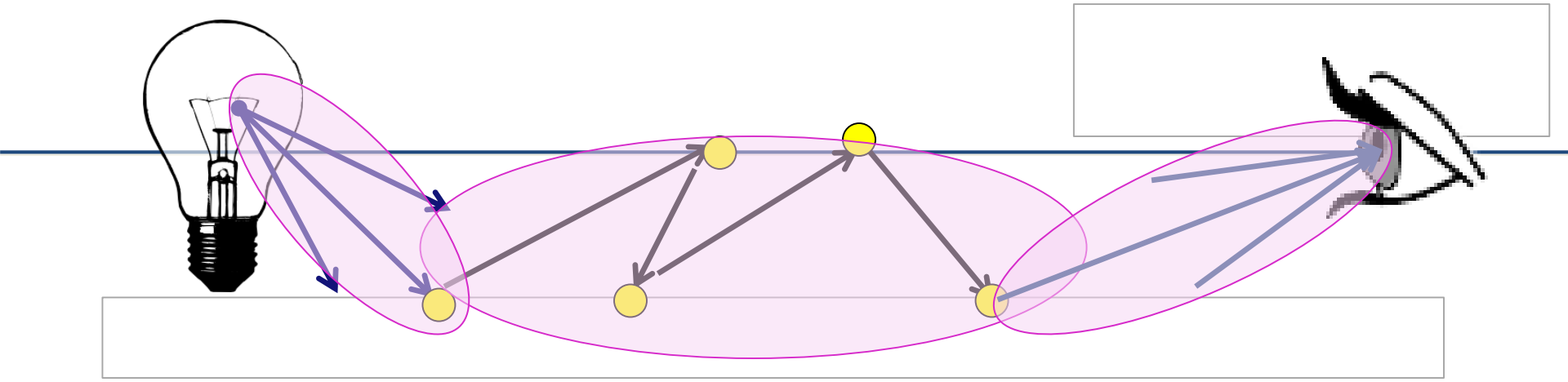


Figure 10: Photon attrition due to Russian roulette sampling for the example scenes. The results support our claim that rasterizing first-bounce photons via the bounce map significantly reduces the number of photons to be simulated; in most scenes, by at least 50%.

3. These expensive bounces **each have a single center** of projection (assuming point emitters and pinhole cameras), and thus can be computed with rasterization.



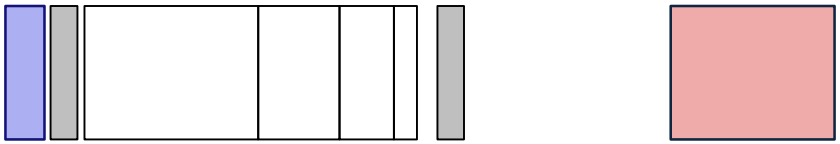
Photon Mapping Time (seconds)

e.g., [Jensen 96, 01, Pharr and Humphreys 04]

1st Bounce 2nd 3rd 4th... Build Tree Last Bounce

("Estimate Radiance")

Image Space Photon Mapping (*milliseconds*)



GPU Bounce Map CPU Trace GPU Photon Volumes

Data Transfer

Image Space Photon Mapping

ISPM algorithm overview:

- **Rasterize** the **scene** from the eye to create a screen space deferred-shading G-buffer.
- For each light, rasterize a **bounce map** in light space that stores photons directly after first bounce in textures or cubemaps.
- Advance bounced photons by **world-space ray tracing** (engineer fast acceleration data structure for dynamic scene)
- Scatter the final photons in screen space, invoking illumination on the G-buffer by rasterizing **photon volumes** with enabled depth test.

Image Space Photon Mapping

Computergraphik und Visualisierung

ISPM algorithm is implemented on GPU and CPU

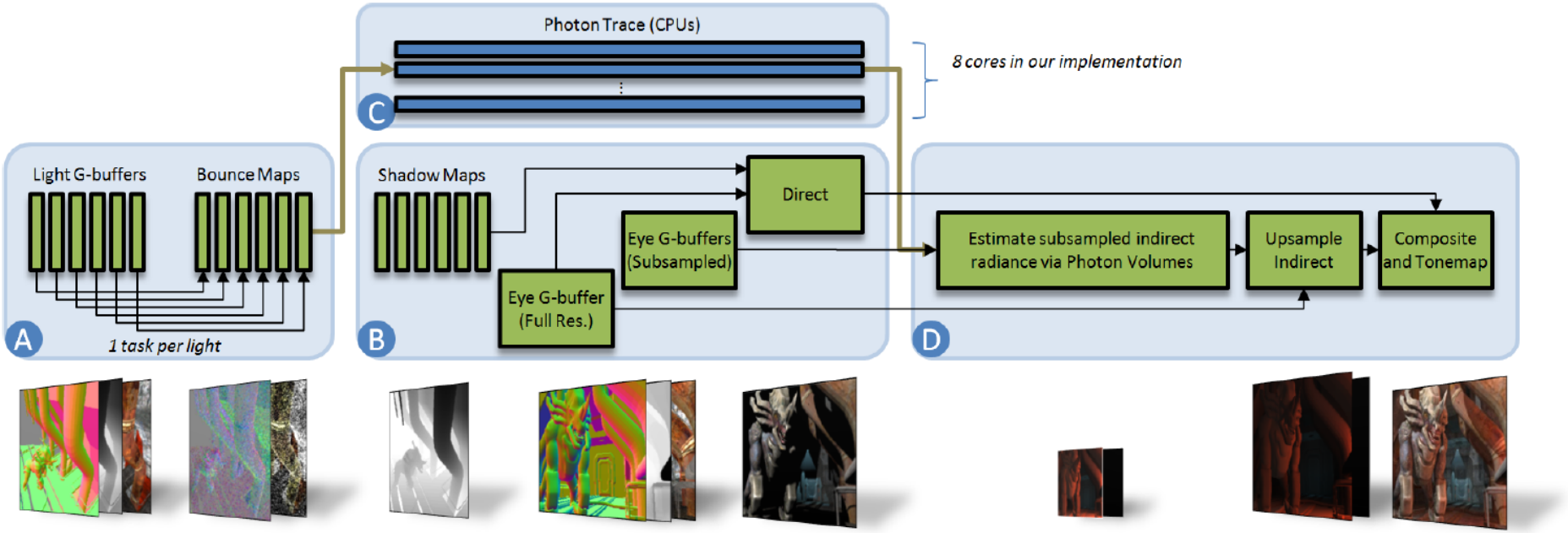
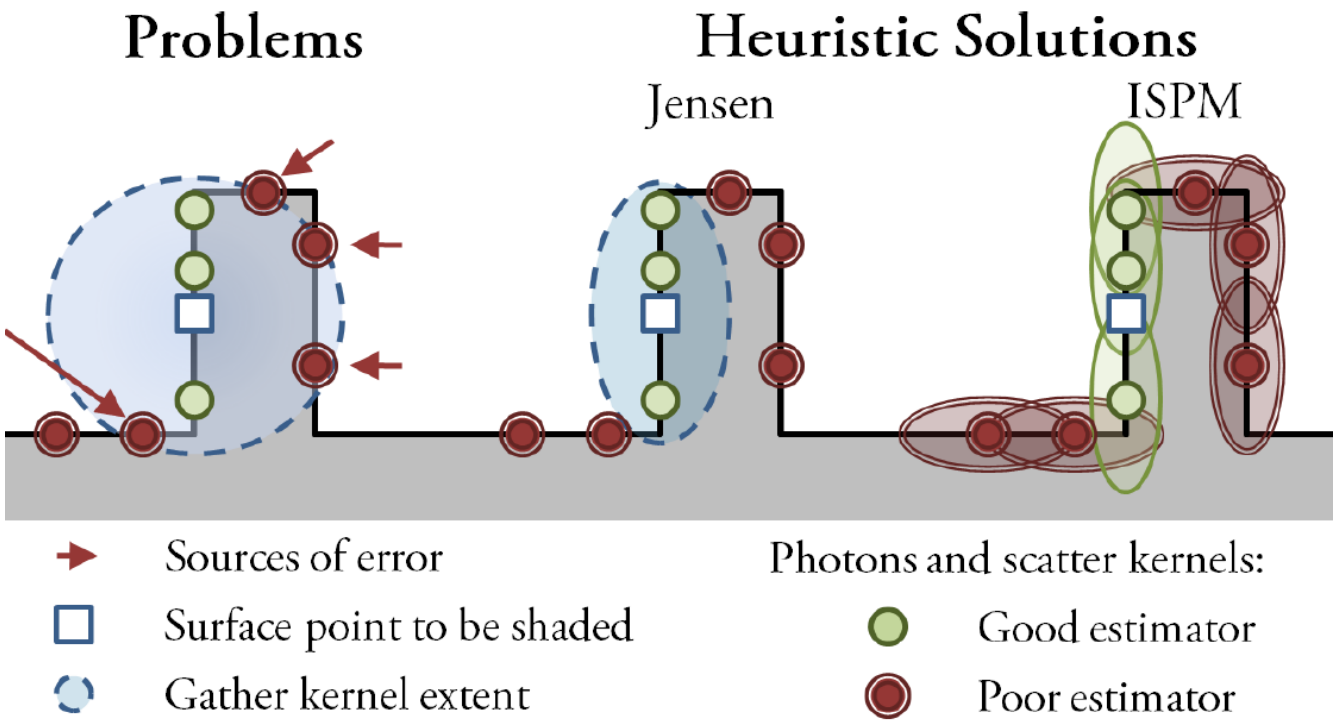


Figure 3: ISPM algorithm overview and task scheduling to address architecture constraints: GPUs have high latency, so a task should not use its immediate upstream neighbor's output; CPU photon tracing and GPU radiance estimation tasks require the most time; thick arrows are barrier synchronization points. Images represent the buffers for the NS2 scene computed by each box and widths are proportional to runtimes.

Image Space Photon Mapping

Gathering is implemented by splatting reconstruction kernels as spherical volumes around photons in screen space. Errors are mitigated by compressing spheres into ellipsoids:



ISPM Results

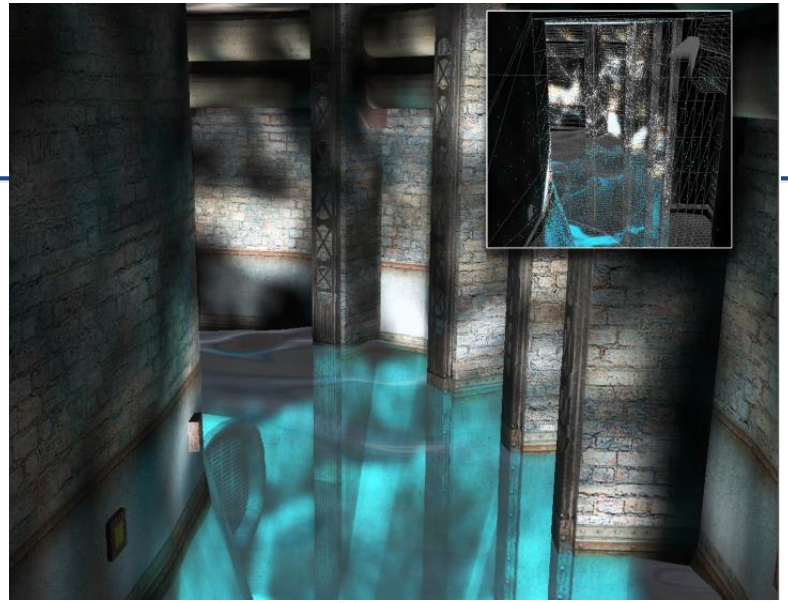


Figure 13: ISPM applied to the NS2 scene. *Left top:* direct illumination only. *Left bottom:* direct plus constant "ambient" term. *Middle top:* photons drawn as large spots on wireframe to illustrate the sampling and geometry. *Middle bottom:* ISPM's 3-bounce indirect illumination radiance estimate. *Right:* composite direct+indirect image rendered by ISPM; compare to the direct+ambient on the lower left. The potentially visible portion of the scene (Figure 9, Cargo Bay only) contains 58k polys on static objects and 15k on dynamic ones, and 4 lights that emit 20k photons total. We import all assets directly from games. This view renders at 21.4 Hz at 1920×1080.

Hardware-Accelerated Global Illumination by

Image Space Photon Mapping

Morgan McGuire
Williams College

David Luebke
NVIDIA Corporation

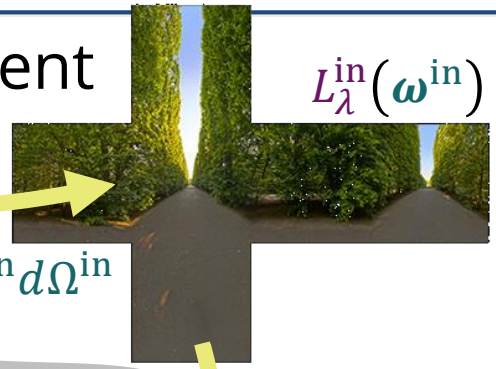
ACM SIGGRAPH / EuroGraphics High-Performance Graphics 2009

Real-Time Techniques

IMAGE BASED LIGHTING (IBL)

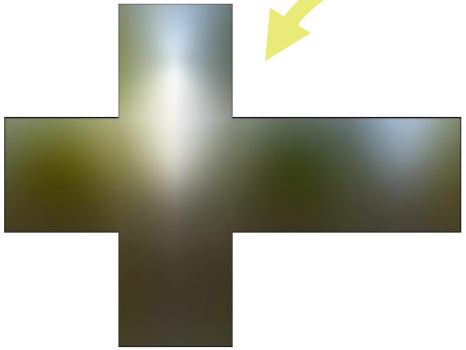
Image Based Lighting

- support lighting from cubemap environment by pre-integration of reflectance integral



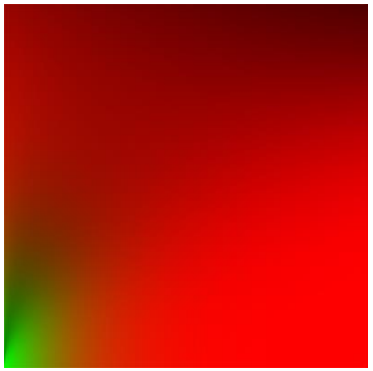
$$L_\lambda^{\text{reflect}}(\boldsymbol{x}, \boldsymbol{\omega}^{\text{out}}) = \iint_{\Omega^{\text{in}}} L_\lambda^{\text{in}}(\boldsymbol{x}, \boldsymbol{\omega}^{\text{in}}) \rho_\lambda(\boldsymbol{x}, \boldsymbol{\omega}^{\text{in}}, \boldsymbol{\omega}^{\text{out}}) \cos\theta^{\text{in}} d\Omega^{\text{in}}$$

- split brdf in diffuse & specular (Cook-Torrance):
  $\ddot{\rho}(\boldsymbol{x}, \boldsymbol{\omega}^{\text{in}}, \boldsymbol{\omega}^{\text{out}}) = \ddot{r}_d f_d + \ddot{r}_s f_{s,\text{Cook–Torrance}}$, with
  $f_d = \frac{1}{\pi}$ and $f_{\text{s,Cook Torrance}} = \frac{F_r(\hat{\boldsymbol{\omega}}_{\text{out}}) G(\hat{\boldsymbol{\omega}}_{\text{in}}, \hat{\boldsymbol{\omega}}_{\text{out}}) D(\hat{\boldsymbol{\omega}}_{\text{h}})}{4 \cdot \cos\theta_{\text{out}} \cos\theta_{\text{in}}}$



Irrandiance Map (IM)

Prefiltered Environment Map (PEM)



Environment BRDF

Image Based Lighting

Computergraphik und Visualisierung

general assumptions:

- there are no shadowing objects that obstruct parts of the environment illumination
- $\boldsymbol{x}$ can be ignored and assumed to be in cubemap center.

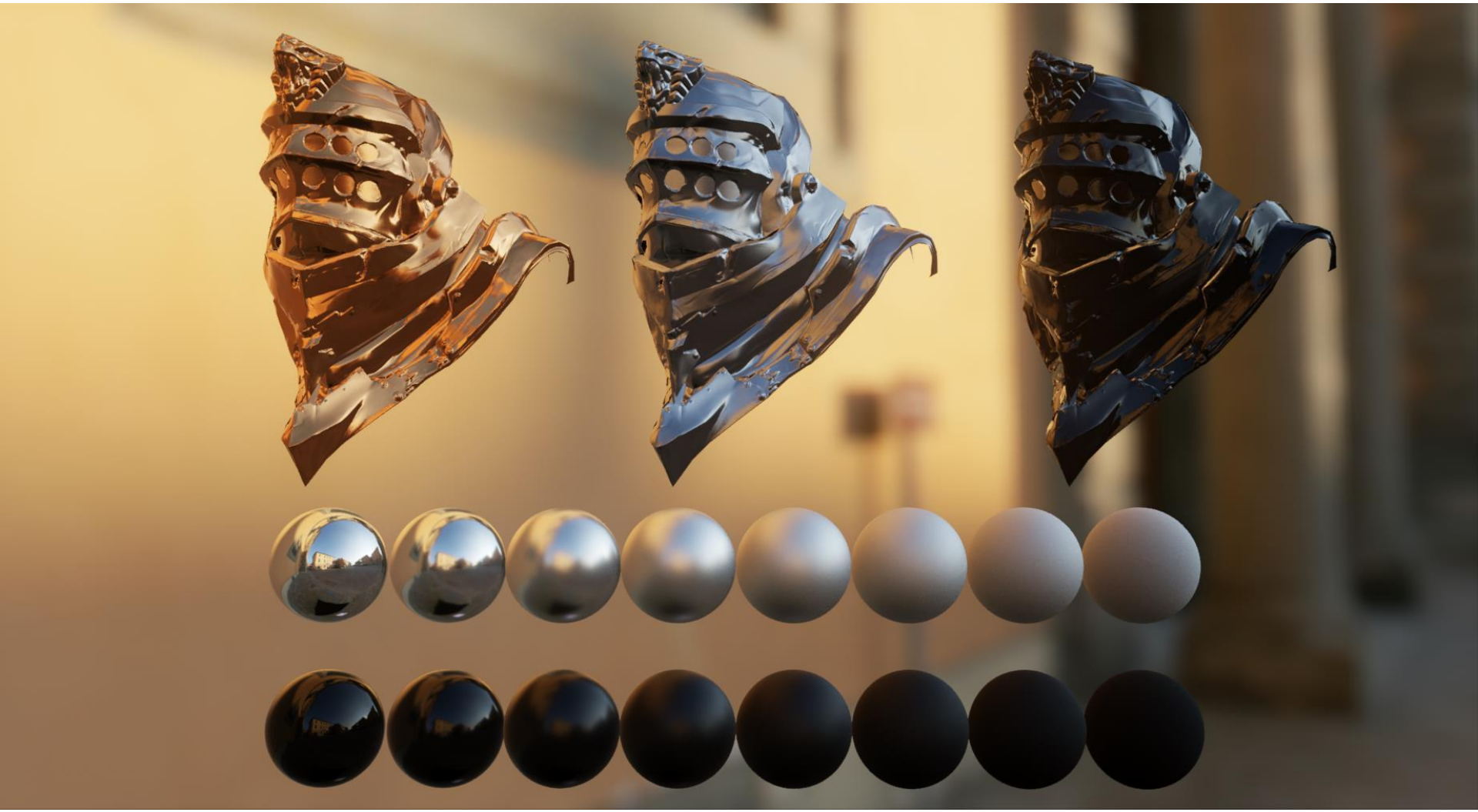
diffuse part

- only depends on hemisphere orientation given by normal direction $\hat{\boldsymbol{n}}$. Integration yields irradiance. Pre-integration computes irrandiance map (IM):

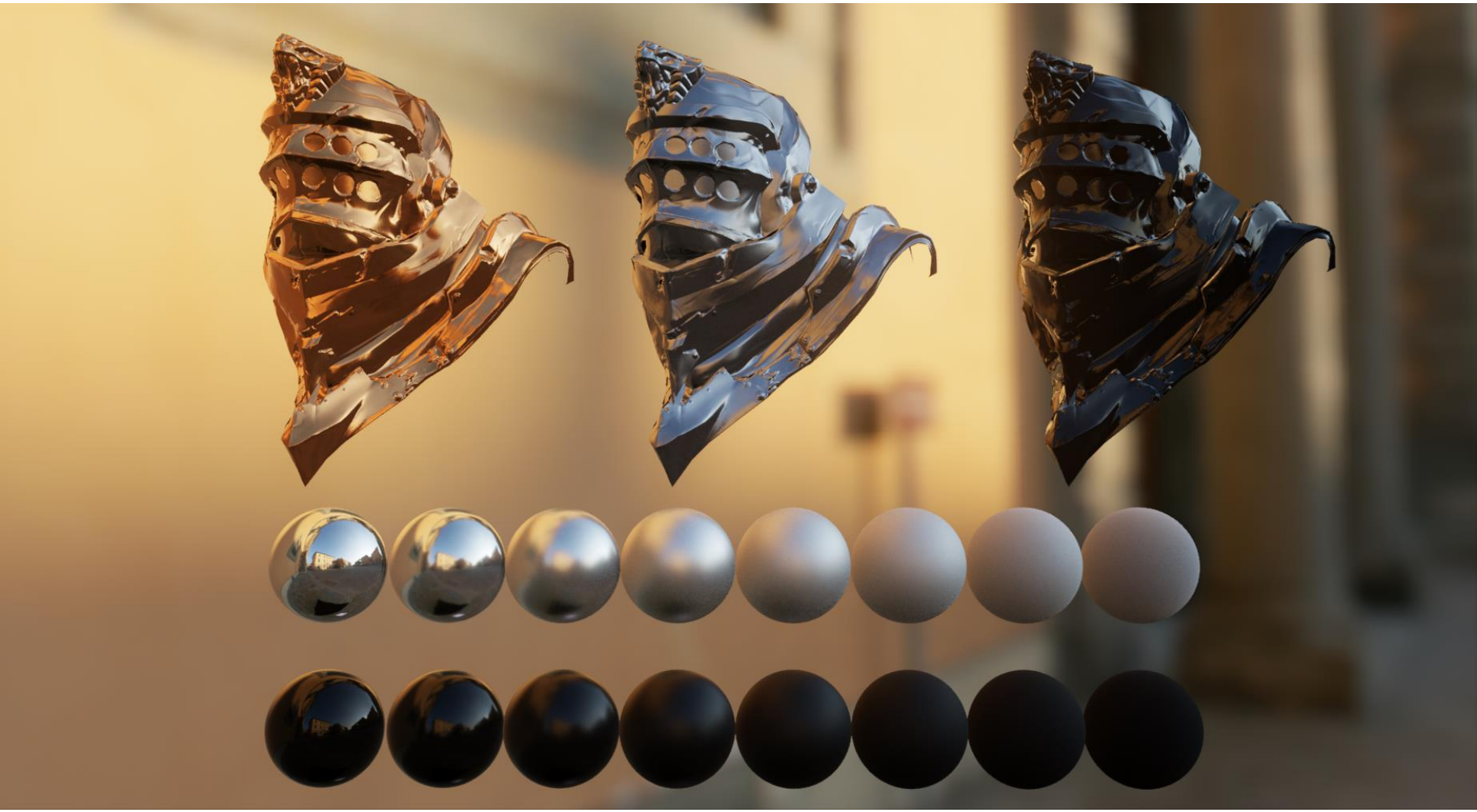
$$\ddot{\mathbf{I}}\ddot{\mathbf{M}}(\hat{\boldsymbol{n}}) = \iint_{\Omega^{\text{in}}(\hat{\boldsymbol{n}})} \frac{1}{\pi} \ddot{\boldsymbol{L}}^{\text{in}}(\boldsymbol{\omega}^{\text{in}}) \cos\theta^{\text{in}} d\Omega^{\text{in}}$$

- real-time access in shader: $\ddot{\boldsymbol{L}}_d^{\text{reflect}}(\boldsymbol{\omega}^{\text{out}}) = \ddot{\boldsymbol{r}}_d \otimes \ddot{\mathbf{I}}\ddot{\mathbf{M}}(\hat{\boldsymbol{n}})$

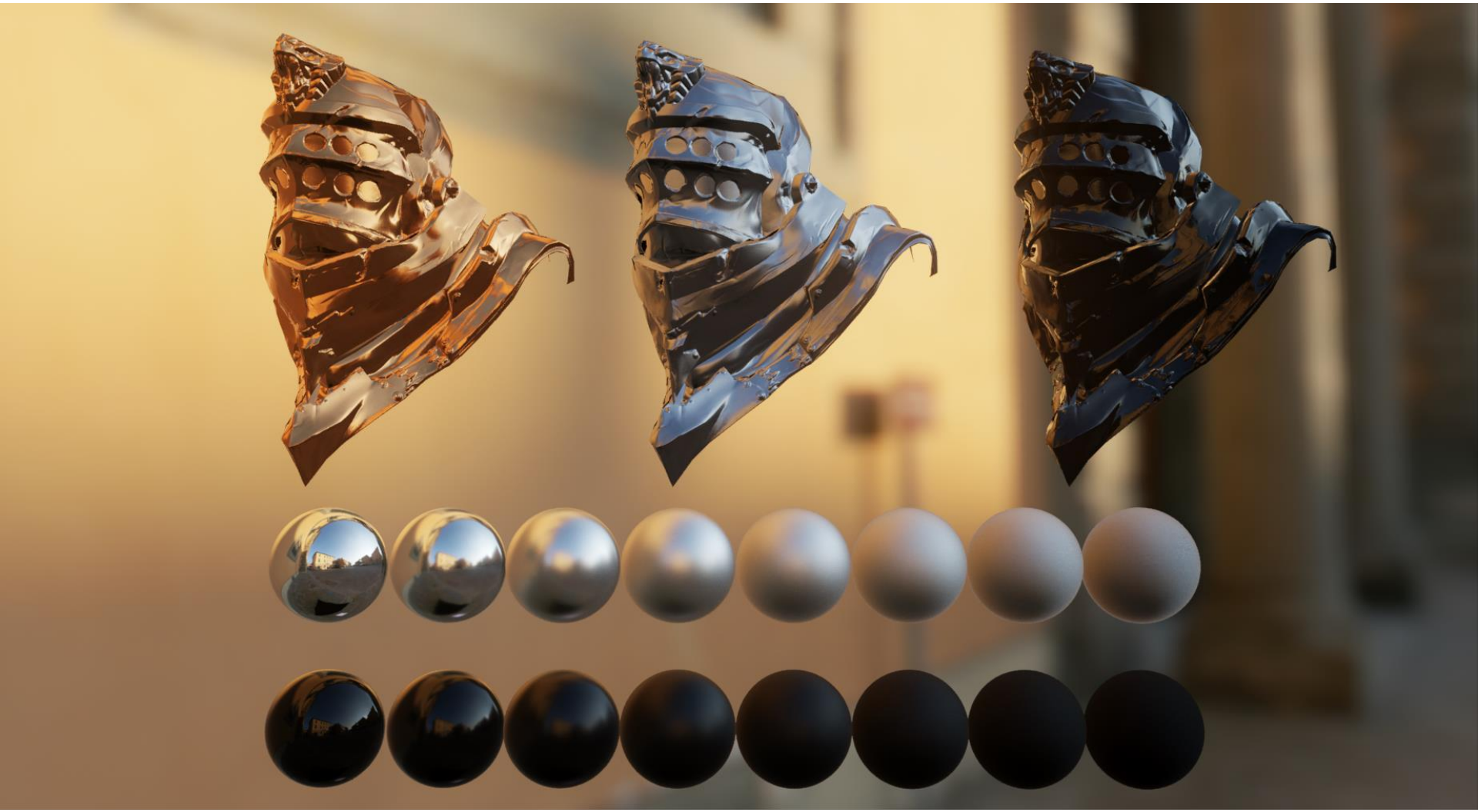
Split sum approximation



Importance-sampled reference



Complete approximation (n=v)



IBL Foundations

Walter, B., Marschner, S. R., Li, H., & Torrance, K. E. (2007). Microfacet Models for Refraction through Rough Surfaces. *Rendering techniques*, *2007*, 18th.

- paper provides self contained intro to microfacet based surface scattering (bsdf) unifying reflection (brdf) and refraction (btdf) with derivations, evaluation and sampling formula.
- It also introduces new normal distribution function (NDF) $D(\hat{\boldsymbol{\omega}}_{\mathrm{h}})$ called GGX and a corresponding GGX shadowing term $G(\hat{\boldsymbol{\omega}}_{\mathrm{in}}, \hat{\boldsymbol{\omega}}_{\mathrm{out}})$
- Furthermore, it explains how to transform pdf of half-vector sampling to pdf of $\hat{\boldsymbol{\omega}}_{\mathrm{in}}$ or $\hat{\boldsymbol{\omega}}_{\mathrm{out}}$.

IBL Foundations – GGX Dist&Shad

Walter, B., Marschner, S. R., Li, H., & Torrance, K. E. (2007). Microfacet Models for Refraction through Rough Surfaces. *Rendering techniques, 2007*, 18th.

$$D(\mathbf{m}) = \frac{\alpha_g^2\ \chi^+(\mathbf{m}\cdot\mathbf{n})}{\pi\cos^4\theta_m\ (\alpha_g^2+\tan^2\theta_m)^2}$$

$$G(\mathbf{i},\mathbf{o},\mathbf{m}) \approx G_1(\mathbf{i},\mathbf{m})G_1(\mathbf{o},\mathbf{m})$$

$$G_1(\mathbf{v},\mathbf{m}) = \chi^+\left(\frac{\mathbf{v}\cdot\mathbf{m}}{\mathbf{v}\cdot\mathbf{n}}\right)\ \frac{2}{1+\sqrt{1+\alpha_g^2\tan^2\theta_v}}$$

typically: $\alpha_g = \varrho^2$ ← roughness

sampling:

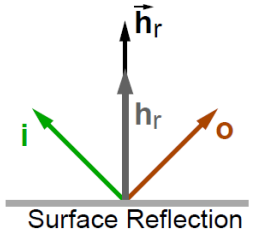
$$\begin{aligned}\theta_m &= \arctan\left(\frac{\alpha_g\sqrt{\xi_1}}{\sqrt{1-\xi_1}}\right)\\ \phi_m &= 2\pi\xi_2\end{aligned}$$

$$\vec{\boldsymbol{\omega}}_\mathrm{h} = \hat{\boldsymbol{\omega}}_\mathrm{in} + \hat{\boldsymbol{\omega}}_\mathrm{out} \qquad \hat{\boldsymbol{\omega}}_\mathrm{h} = \frac{\vec{\boldsymbol{\omega}}_\mathrm{h}}{\|\vec{\boldsymbol{\omega}}_\mathrm{h}\|}$$

$$d\Omega^\mathrm{h} = \frac{d\Omega^\mathrm{out}}{4|\hat{\boldsymbol{\omega}}_\mathrm{out}\cdot\hat{\boldsymbol{\omega}}_\mathrm{h}|} = \frac{d\Omega^\mathrm{in}}{4|\hat{\boldsymbol{\omega}}_\mathrm{in}\cdot\hat{\boldsymbol{\omega}}_\mathrm{h}|}$$

$$\frac{|\hat{\boldsymbol{\omega}}_\mathrm{out}\cdot\hat{\boldsymbol{\omega}}_\mathrm{h}|}{\|\vec{\boldsymbol{\omega}}_\mathrm{h}\|^2} = \frac{|\hat{\boldsymbol{\omega}}_\mathrm{out}\cdot\hat{\boldsymbol{\omega}}_\mathrm{h}|}{4|\hat{\boldsymbol{\omega}}_\mathrm{out}\cdot\hat{\boldsymbol{\omega}}_\mathrm{h}|^2} = \frac{1}{4|\hat{\boldsymbol{\omega}}_\mathrm{out}\cdot\hat{\boldsymbol{\omega}}_\mathrm{h}|}$$

$$\|\vec{\boldsymbol{\omega}}_\mathrm{h}\| = |\vec{\boldsymbol{\omega}}_\mathrm{h}\cdot\hat{\boldsymbol{\omega}}_\mathrm{h}| = |(\hat{\boldsymbol{\omega}}_\mathrm{in}+\hat{\boldsymbol{\omega}}_\mathrm{out})\cdot\hat{\boldsymbol{\omega}}_\mathrm{h}| = |2\hat{\boldsymbol{\omega}}_{\mathrm{out|in}}\cdot\hat{\boldsymbol{\omega}}_\mathrm{h}|$$



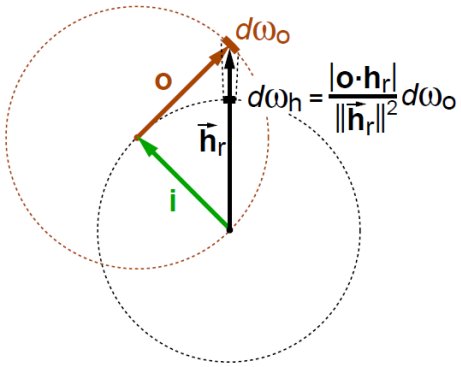


Figure 6: *Geometry for ideal reflection with half-vector* $\vec{\mathbf{h}}_\mathrm{r} = \mathbf{i}+\mathbf{o}$ *and normalized half-direction* $\mathbf{h}_\mathrm{r} = \vec{\mathbf{h}}_\mathrm{r}/\|\vec{\mathbf{h}}_\mathrm{r}\|$. *To compute the Jacobian we compute the solid angle perturbation in the normalized half vector,* $d\omega_h$, *induced by an infinitesimal solid angle perturbation,* $d\omega_o$, *in* $\mathbf{o}$. *Solid angle is directly proportional to area on the corresponding unit spheres. Only the 2D incidence plane slice through the full 3D space is shown.*

Image Based Lighting

specular part

- transform reflection integral to half-vector:

$$\ddot{L}_s^{\text{reflect}}(\hat{\omega}^{\text{out}}) = \ddot{r}_s \otimes \iint_{\Omega^{\text{in}}} f_s(\hat{\omega}^{\text{in}},\hat{\omega}^{\text{out}})\ddot{L}^{\text{in}}(\hat{\omega}^{\text{in}})\cos\theta^{\text{in}}\,d\Omega^{\text{in}}$$

$$\ddot{L}_s^{\text{reflect}}(\hat{\omega}^{\text{out}}) = \ddot{r}_s \otimes \iint_{\Omega^{\text{h}}} f_s(\hat{\omega}^{\text{in}},\hat{\omega}^{\text{out}})\ddot{L}^{\text{in}}(\hat{\omega}^{\text{in}})\cos_+\theta^{\text{in}}\,4\left|\hat{\omega}^{\text{in}}\cdot\hat{\omega}^{\text{h}}\right|d\Omega^{\text{h}}$$

- Monte Carlo estimate with importance sampling according to NDF pdf $p(\hat{\omega}_{\text{h}}) = D(\hat{\omega}_{\text{h}})|\hat{\omega}_{\text{h}}\cdot\hat{n}|$ with $\hat{\omega}_i^{\text{in}} = 2\left(\hat{\omega}^{\text{out}}\cdot\hat{\omega}_i^{\text{h}}\right)\hat{\omega}_i^{\text{h}} - \hat{\omega}^{\text{out}}$

$$\hat{\ddot{L}}_s^{\text{reflect}}(\hat{\omega}^{\text{out}}) = \frac{4\ddot{r}_s}{N}\otimes\sum_i \frac{f_s(\hat{\omega}_i^{\text{in}},\hat{\omega}^{\text{out}})\ddot{L}^{\text{in}}(\hat{\omega}_i^{\text{in}})\cos_+\theta_i^{\text{in}}\left|\hat{\omega}_i^{\text{in}}\cdot\hat{\omega}_i^{\text{h}}\right|}{p(\hat{\omega}_{\text{h}})}$$

$$\hat{\ddot{L}}_s^{\text{reflect}}(\hat{\omega}^{\text{out}}) = \frac{4\ddot{r}_s}{N}\otimes\sum_i \frac{F_r(\hat{\omega}_{\text{out}})G(\hat{\omega}_i^{\text{in}},\hat{\omega}^{\text{out}})\cancel{D(\hat{\omega}_{\text{h}})}\cos_+\theta_i^{\text{in}}\left|\hat{\omega}_i^{\text{in}}\cdot\hat{\omega}_i^{\text{h}}\right|}{4\cdot\cos\theta^{\text{out}}\cos\theta_i^{\text{in}}\,\cancel{D(\hat{\omega}_{\text{h}})}\left|\hat{n}\cdot\hat{\omega}_i^{\text{h}}\right|}\ddot{L}^{\text{in}}(\omega_i^{\text{in}})$$

$$\hat{\ddot{L}}_s^{\text{reflect}}(\hat{\omega}^{\text{out}}) = \frac{\ddot{r}_s}{N}\otimes\sum_i \underbrace{\chi^+\left(\hat{n}\cdot\hat{\omega}_i^{\text{in}}\right)}_{\chi^+(x)=\begin{cases}1 & x>0\\ 0 & x\le 0\end{cases}} F_r(\hat{\omega}_{\text{out}})G(\hat{\omega}_i^{\text{in}},\hat{\omega}_{\text{out}})\frac{\left|\hat{\omega}^{\text{out}}\cdot\hat{\omega}_i^{\text{h}}\right|}{\left|\hat{n}\cdot\hat{\omega}^{\text{out}}\right|\left|\hat{n}\cdot\hat{\omega}_i^{\text{h}}\right|}\ddot{L}^{\text{in}}(\omega_i^{\text{in}})$$

Image Based Lighting

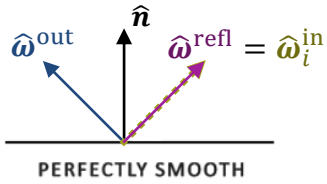
specular part continued

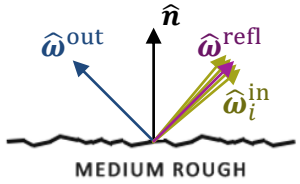
- split sum approximation:

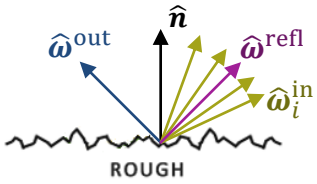
$$\hat{L}_s^{\text{reflect}}(\hat{\omega}^{\text{out}}) = \frac{\ddot{r}_s}{N} \otimes \sum_i \chi^+(\hat{n}\cdot\hat{\omega}_i^{\text{in}}) F_r(\hat{\omega}_{\text{out}}) G(\hat{\omega}_i^{\text{in}}, \hat{\omega}_{\text{out}}) \frac{|\hat{\omega}^{\text{out}}\cdot\hat{\omega}_i^{\text{h}}|}{|\hat{n}\cdot\hat{\omega}^{\text{out}}||\hat{n}\cdot\hat{\omega}_i^{\text{h}}|} \ddot{L}^{\text{in}}(\omega_i^{\text{in}})$$

$$\approx \underbrace{\left(\frac{1}{N}\sum_i \ddot{L}^{\text{in}}(\omega_i^{\text{in}})\right)}_{\text{PrefilterEnvMap}(\cancel{\hat{\omega}^{\text{out}}},\hat{n},\varrho)} \otimes \ddot{r}_s \underbrace{\left(\frac{1}{N}\sum_i \chi^+(\hat{n}\cdot\hat{\omega}_i^{\text{in}}) F_r(\hat{\omega}_{\text{out}}) G(\hat{\omega}_i^{\text{in}}, \hat{\omega}_{\text{out}}) \frac{|\hat{\omega}^{\text{out}}\cdot\hat{\omega}_i^{\text{h}}|}{|\hat{n}\cdot\hat{\omega}^{\text{out}}||\hat{n}\cdot\hat{\omega}_i^{\text{h}}|}\right)}_{\text{EnvironmentBRDF}(\hat{\omega}^{\text{out}},\hat{n},\varrho)}$$

- Note that split sum approximation separates environment from brdf
- Note that $\text{PrefilterEnvMap}(\hat{n}, \varrho)$ depends also on normal and roughness as sampling depends on these parameters.







specular part – Prefiltered Environment Map

$$\mathbf{PËM}(\hat{\boldsymbol{n}}, \varrho) = \frac{1}{N}\sum_i \ddot{\boldsymbol{L}}^{\text{in}}\left(\hat{\boldsymbol{\omega}}_i^{\text{in}}\right)$$

- To eliminate dependence on two directions, one additionally assumes that $\hat{\boldsymbol{\omega}}^{\text{out}} = \hat{\boldsymbol{n}} = \hat{\boldsymbol{\omega}}^{\text{refl}}$ and uses $\hat{\boldsymbol{\omega}}^{\text{refl}}$ to parametrize $\mathbf{PËM}$.
- Roughness is dealt with by constructing a mipmap with $\varrho \in [0,1]$ defining the mipmap level, such that $\mathbf{PËM}\left(\hat{\boldsymbol{\omega}}^{\text{refl}}, \varrho\right)$ is implemented in cube-mipmap pyramid:

Image Based Lighting

specular part – Environment BRDF

$$\mathrm{EBR}(\hat{\boldsymbol{\omega}}^{\mathrm{out}}, \hat{\boldsymbol{n}}, \varrho) = \frac{1}{N}\sum_i \chi^+\left(\hat{\boldsymbol{n}} \cdot \hat{\boldsymbol{\omega}}_i^{\mathrm{in}}\right) F_r(\hat{\boldsymbol{\omega}}_{\mathrm{out}}) G\left(\hat{\boldsymbol{\omega}}_i^{\mathrm{in}}, \hat{\boldsymbol{\omega}}_{\mathrm{out}}\right) \frac{\left|\hat{\boldsymbol{\omega}}^{\mathrm{out}} \cdot \hat{\boldsymbol{\omega}}_i^{\mathrm{h}}\right|}{\left|\hat{\boldsymbol{n}} \cdot \hat{\boldsymbol{\omega}}^{\mathrm{out}}\right| \left|\hat{\boldsymbol{n}} \cdot \hat{\boldsymbol{\omega}}_i^{\mathrm{h}}\right|}$$

- $\mathrm{EBR}(\hat{\boldsymbol{\omega}}^{\mathrm{out}}, \hat{\boldsymbol{n}}, \varrho)$ provides scalar reflection due to homogeneous illumination from all sides such that arbitrary coordinate system can be chosen
- Assuming an isotropic brdf and a default coordinate system with $\hat{\boldsymbol{n}} = \hat{\boldsymbol{z}}$ and $\boldsymbol{\omega}^{\mathrm{out}}$ with zero y-component, one can recover $\boldsymbol{\omega}^{\mathrm{out}}$ from scalar $\boldsymbol{\omega}^{\mathrm{out}} \cdot \hat{\boldsymbol{n}}$ and $\hat{\boldsymbol{\omega}}_i^{\mathrm{in}}$ can be computed from the sampled $\hat{\boldsymbol{\omega}}_i^{\mathrm{h}}$.
- Therefore $\mathrm{EBR}(\hat{\boldsymbol{\omega}}^{\mathrm{out}} \cdot \hat{\boldsymbol{n}}, \varrho)$ can be parameterized over two parameters both in the range $[0,1]$.

Image Based Lighting

specular part – Environment BRDF – continued

$$\mathrm{EBR}(\hat{\boldsymbol{\omega}}^{\mathrm{out}} \cdot \hat{\boldsymbol{n}}, \varrho) = \frac{1}{N} \sum_i F_r(\hat{\boldsymbol{\omega}}_{\mathrm{out}}) \cdot \underbrace{\phi(\hat{\boldsymbol{\omega}}_i^{\mathrm{in}}, \hat{\boldsymbol{\omega}}_{\mathrm{out}}, \hat{\boldsymbol{n}})}_{\chi^+(\hat{\boldsymbol{n}} \cdot \hat{\boldsymbol{\omega}}_i^{\mathrm{in}}) G(\hat{\boldsymbol{\omega}}_i^{\mathrm{in}}, \hat{\boldsymbol{\omega}}_{\mathrm{out}}) \frac{|\hat{\boldsymbol{\omega}}^{\mathrm{out}} \cdot \hat{\boldsymbol{\omega}}_i^{\mathrm{h}}|}{|\hat{\boldsymbol{n}} \cdot \hat{\boldsymbol{\omega}}^{\mathrm{out}}| |\hat{\boldsymbol{n}} \cdot \hat{\boldsymbol{\omega}}_i^{\mathrm{h}}|}}$$

- With Schlick's approximation to $F_r(\hat{\boldsymbol{\omega}}_{\mathrm{out}})$ one can furthermore take out the material specific orthogonal reflection factor F_0 of the integral:

$$F_r^{\mathrm{Schlick}}(\hat{\boldsymbol{\omega}}_{\mathrm{out}}) = F_0 + (1 - F_0) \underbrace{\left(1 - \hat{\boldsymbol{\omega}}^{\mathrm{out}} \cdot \hat{\boldsymbol{\omega}}_i^{\mathrm{h}}\right)^5}_{o_i} = F_0(1 - o_i) + o_i$$

- Representing $\mathbf{E\ddot{B}R}$ with two channels (x for F_0 scale and y for offset) allows real-time computation according to

$$\mathrm{EBR}(\hat{\boldsymbol{\omega}}^{\mathrm{out}} \cdot \hat{\boldsymbol{n}}, \varrho) = F_0 \cdot \underline{\mathbf{EBR}}(\hat{\boldsymbol{\omega}}^{\mathrm{out}} \cdot \hat{\boldsymbol{n}}, \varrho).\mathrm{x} + \underline{\mathbf{EBR}}(\hat{\boldsymbol{\omega}}^{\mathrm{out}} \cdot \hat{\boldsymbol{n}}, \varrho).\mathrm{y}$$

$$\underline{\mathbf{EBR}}(\hat{\boldsymbol{\omega}}^{\mathrm{out}} \cdot \hat{\boldsymbol{n}}, \varrho).\mathrm{x} = \frac{1}{N} \sum_i \phi(\hat{\boldsymbol{\omega}}_i^{\mathrm{in}}, \hat{\boldsymbol{\omega}}_{\mathrm{out}}, \hat{\boldsymbol{n}})(1 - o_i)$$

$$\underline{\mathbf{EBR}}(\hat{\boldsymbol{\omega}}^{\mathrm{out}} \cdot \hat{\boldsymbol{n}}, \varrho).\mathrm{y} = \frac{1}{N} \sum_i \phi(\hat{\boldsymbol{\omega}}_i^{\mathrm{in}}, \hat{\boldsymbol{\omega}}_{\mathrm{out}}, \hat{\boldsymbol{n}}) o_i$$

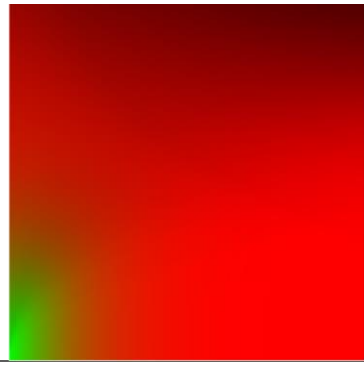


Image Based Lighting

source code can be found

- In course notes of Epic Games contribution to siggraph 2013 course on real-time rendering in games:

  https://cdn2.unrealengine.com/Resources/files/2013SiggraphPresentationsNotes-26915738.pdf

- in online book

  https://learnopengl.com/PBR/IBL

  together with explanations that are a bit buggy

Real-Time Techniques

VOXEL CONE TRACING (SOON)

References

- 1988 G. Ward: *Ray Tracing Solution for Diffuse Interreflection*
- 1991 G. Ward: *Real pixels*. Graphics Gems II, pp. 80-83
- 1994 H.W. Jensen: *Efficient Shadows using the Photon Map*
- 1995 H.W. Jensen: *Importance driven path tracing using the photon map*
- 1996 H.W. Jensen: *Global illumination using photon maps*.
- 1996 H.W. Jensen: *Rendering Caustics on Non-Lambertian Surfaces*
- 1997 E. Veach: *Robust Monte Carlo Methods for Light Transport Simulation*, chapter 10
- 1998 H.W. Jensen: *Efficient Simulation of Light Transport in Scenes with Participating Media using Photon Maps*.
- 2005 V. Havran: *Fast final gathering via reverse photon mapping*.
- 2007 B. Walter: *Microfacet Models for Refraction through Rough Surfaces*
- 2008 T. Hachisuka: *Progressive Photon Mapping*.
- 2009 T. Hachisuka: *Stochastic progressive photon mapping*. (code)
- 2009 M. McGuire: *Hardware-accelerated global illumination by image space photon mapping*
- 2011 C. Knaus: *Progressive photon mapping a probabilistic approach*.
- 2012 I. Georgiev: *Light Transport Simulation with Vertex Connection and Merging*.
- 2013 B. Karis: *Real Shading in Unreal Engine 4*.