Practical Global Illumination with Irradiance Caching

SIGGRAPH 2007 Course 16

Organizers

Jaroslav Křivánek
Czech Technical University in Prague

Pascal Gautron
France Telecom R&D

Lecturers

Greg Ward
Anyhere Software

Okan Arikan
University of Texas at Austin

Henrik Wann Jensen
University of California, San Diego

The course provides a practical guide to implementing irradiance caching algorithm to efficiently render flawless images featuring global illumination. In addition, recent research advances related to irradiance caching are presented: caching on glossy surfaces, temporal caching, acceleration through GPU implementation and irradiance decomposition.

Course abstract

Since its invention 20 years ago, irradiance caching has successfully been used to accelerate the global illumination computation in the *Radiance* lighting simulation system. Its widespread use had to wait until computers became fast enough to consider global illumination in production rendering. Since then, its use is ubiquitous. Virtually all commercial and open-source rendering software base the global illumination computation upon irradiance caching. Irradiance caching is essential for making photon mapping efficient.

The objective of the course is twofold. The first objective is to expose the irradiance caching algorithm along with all the details and tricks upon which the success of a practical implementation is dependent. Various image artifacts that the basic algorithm can produce will be shown along with a recipe to suppress them.

The second objective is to acquaint the audience with the recent research results that increase the speed and extend the functionality of irradiance caching. Those include: exploiting temporal coherence to suppress temporal flickering, extending the caching mechanism to rendering glossy surfaces; accelerating the algorithm by porting it to the GPU or by decomposing the indirect illumination field by distance. Advantages and disadvantages of those methods will be clearly exposed.

Course prerequisites

A basic understanding of rendering, ray tracing in particular, is expected. Familiarity with global illumination concepts is useful.

Level of difficulty

Intermediate

Intended audience

Anyone interested in making an irradiance caching implementation that reliably renders artifact-free images. Researchers and industrial developers interested in recent speed and quality improvements of global illumination with irradiance caching.

Course syllabus

1. Introduction (*Křivánek*)
   (8:30 – 8:35 / 5 min)
   - What is global illumination, why computing it?
   - Course overview

2. Stochastic Ray Tracing (*Křivánek*)
   (8:35 – 8:55 / 20 min)
   - Diffuse interreflections
   - Gathering approach to computing diffuse interreflections
   - Final gathering for photon mapping

3. Irradiance Caching Algorithm (*Ward*)
   (8:55 – 9:20 / 25 min)
   - Spatial coherence of indirect illumination
   - Caching strategy
   - Weighting function
   - Data structure
   - Irradiance Gradients

4. Irradiance Caching in RADIANCE (*Ward*)
   (9:20 – 9:35 / 15 min)
   - Computation of ambient "constant"
   - Adaptive super-sampling on hemisphere
   - Maximum and minimum record spacing
   - Gradient limit on record spacing
   - Bump maps using rotation gradient
   - Options for excluding surfaces/materials
   - Record sharing for multiprocessors

5. Implementation Details (*Křivánek*)
   (9:35 – 9:55 / 20 min)
   - Minimum record spacing
   - Missing small geometry
   - Neighbor clamping
   - Ray leaking
   - Weighting function revisited
   - Image sampling

6. Irradiance Caching and Photon Maps (*Jensen*)
   (9:55 – 10:15 / 20 min)

Break
(10:15 – 10:30 / 15 min)

7. Extension to Glossy Surfaces: Radiance Caching (*Křivánek*)
   (10:30 – 10:50 / 20 min)
   - Indirect Illumination on Glossy Surfaces
   - Incoming radiance representation, spherical harmonics
   - Radiance interpolation
   - Adaptive radiance caching

8. Hardware Implementation (*Gautron*)
   (10:50 – 11:10 / 20 min)
   - Irradiance caching: spatial data structure and ray tracing
   - Reformulation for efficient GPU implementation
   - Irradiance splatting
   - GPU-Based hemisphere sampling

9. Temporal Coherence (*Gautron*)
   (11:10 – 11:35 / 25 min)
   - Irradiance caching leads to flickering in animations
   - Temporal coherence of indirect lighting
   - Temporal weighting function
   - Temporal gradients

10. Irradiance Decomposition (*Arikan*)
    (11:35 – 12:00 / 25 min)
    - The problems with too many samples near geometric detail
    - Decomposing irradiance into nearby and distant terms
    - Sparse sampling for distant illumination
    - Efficient approximation for near illumination
    - Faking global illumination using only nearby objects

11. Discussion (*All*)
    (12:00 – 12:15 / 15 min)

Course presenter information

Jaroslav Křivánek, Czech Technical University in Prague
xkrivanj@fel.cvut.cz

Jaroslav Křivánek is an assistant professor at the Czech Technical University in Prague. He received his Ph.D. from IRISA/INRIA Rennes and the Czech Technical University (joint degree) in 2005 for extending the irradiance caching algorithm to rendering global illumination on glossy surfaces. In 2003 and 2004 he was a research associate at the University of Central Florida. He received a Masters in computer science from the Czech Technical University in Prague in 2001.

Pascal Gautron, France Telecom R&D
pascal.gautron@orange-ftgroup.com

Pascal Gautron is a post-doctoral researcher at France Telecom R&D Rennes, France. During his Ph.D. in collaboration with the IRISA/INRIA Rennes and the University of Central Florida, his research focused on fast, GPU-based global illumination computation using the irradiance caching algorithm. His current research work aims at physically-based rendering of organic materials in real-time.

Greg Ward, Anyhere Software
gward@lmi.net

Greg Ward has been involved with computer graphics since 1985, when he began development of the RADIANCE lighting simulation and rendering system at the Lawrence Berkeley National Lab. Since then, he has worked on rendering algorithms, reflectance models and measurement systems, tone reproduction operators, and HDR image processing and display techniques. His past employers include the Lawrence Berkeley National Laboratory, EPFL Switzerland, SGI, Shutterfly, and Exponent. Greg holds a bachelor's in Physics from UC Berkeley and a master's in Computer Science from SF State University. He is currently working as an independent consultant in Albany, California <www.anyhere.com>.

Okan Arikan, University of Texas at Austin
okan@cs.utexas.edu

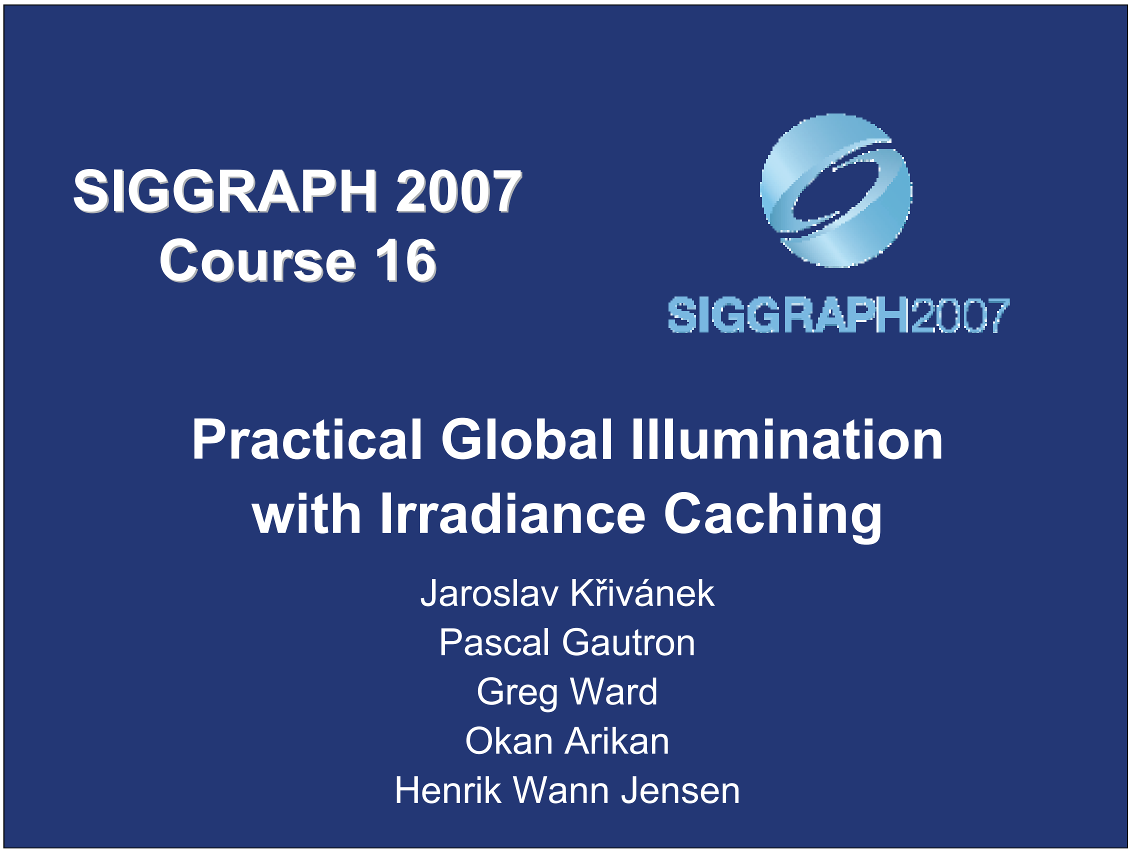
Okan Arikan has been an assistant professor at University of Texas, Austin since 2005. Prior to joining UT, he got his Ph.D. and masters from University of California, Berkeley. His research interests include character animation, physically based modeling, rendering and computer assisted digital content creation. He is also the lead developer of Pixie – open source RenderMan renderer –, in which he has implemented his irradiance decomposition algorithm that dramatically speeds up irradiance caching.

Henrik Wann Jensen, University of California, San Diego
henrik@cs.ucsd.edu

Dr. Henrik Wann Jensen is an assistant professor at the University of California at San Diego, where he is establishing a computer graphics lab. with a research focus on realistic image synthesis, global illumination, rendering of natural phenomena, and appearance modeling. His contributions to computer graphics include the photon mapping algorithm for global illumination, and the first technique for efficiently simulating subsurface scattering in translucent materials. He is the author of ”Realistic Image Synthesis using Photon Mapping,” AK Peters 2001. Prior to coming to UCSD in 2002, he was a research associate at Stanford University from 1999-2002, a postdoctoral researcher at the Massachusetts Institute of Technology (MIT) from 1998-1999, and a research scientist in industry working on commercial rendering software from 1996-1998. He received his M.Sc. and Ph.D. in Computer Science from the Technical University of Denmark in 1996 for developing the photon mapping method.

In 2004, Professor Jensen received an Academy Award (Technical Achievement Award) from the Academy of Motion Picture Arts and Sciences for pioneering research in rendering translucent materials, and he became a Sloan Fellow.

SIGGRAPH2007



SIGGRAPH 2007
Course 16

SIGGRAPH2007

Practical Global Illumination with Irradiance Caching

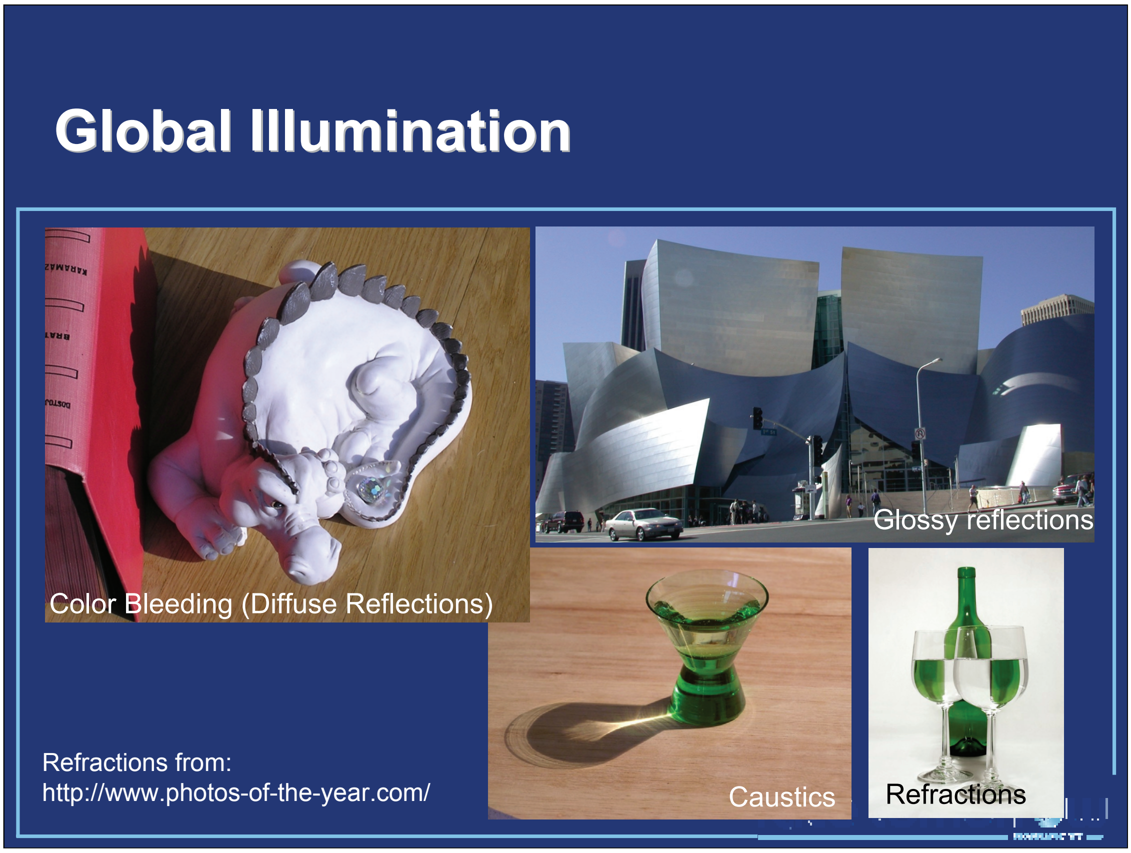
Jaroslav Křivánek
Pascal Gautron
Greg Ward
Okan Arikan
Henrik Wann Jensen

Introduction

SIGGRAPH2007

Jaroslav Křivánek

ČVUT v Praze – CTU Prague



Global Illumination

Color Bleeding (Diffuse Reflections)

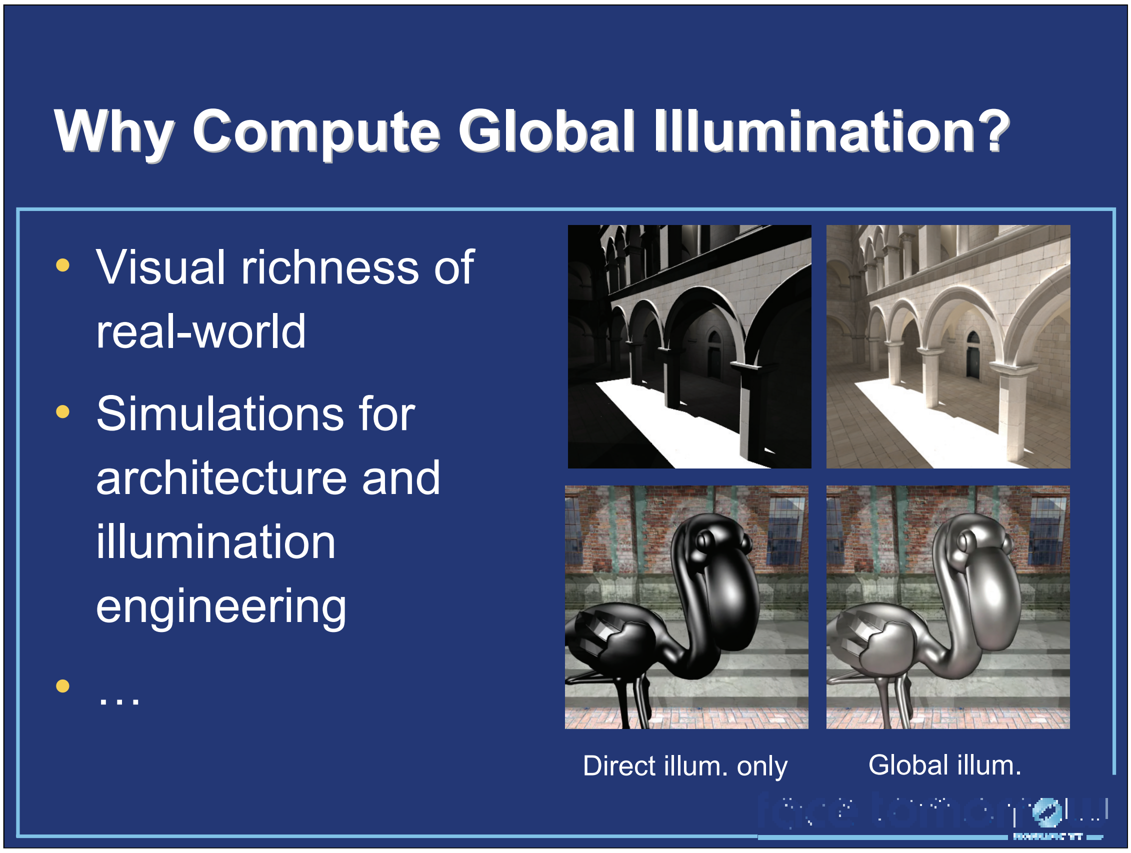
Glossy reflections

Caustics

Refractions

Refractions from:
http://www.photos-of-the-year.com/

The term “global illumination” embraces many lighting effects encountered in real world. Some of them are shown on this slide.



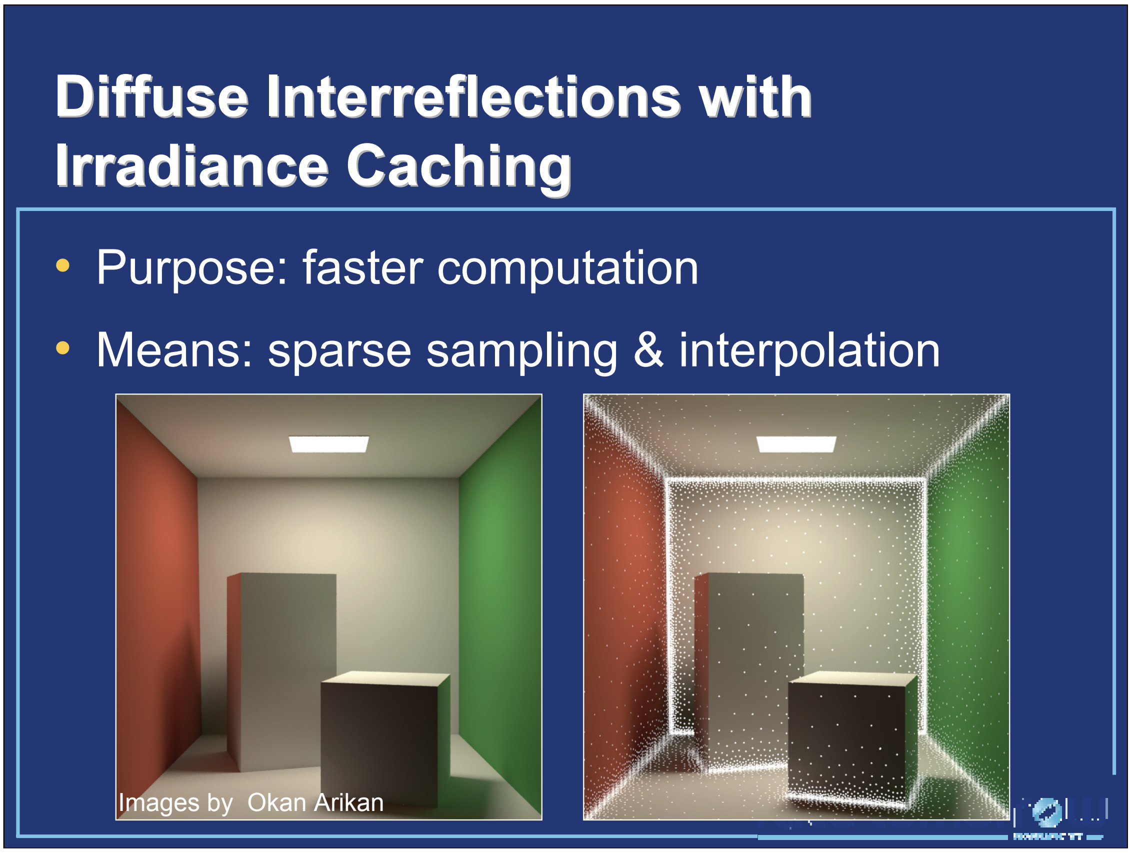
Why Compute Global Illumination?

- Visual richness of real-world
- Simulations for architecture and illumination engineering
- …

Direct illum. only

Global illum.

Simulating global illumination can reproduce the visual richness of real world. It is also useful for predictive rendering.

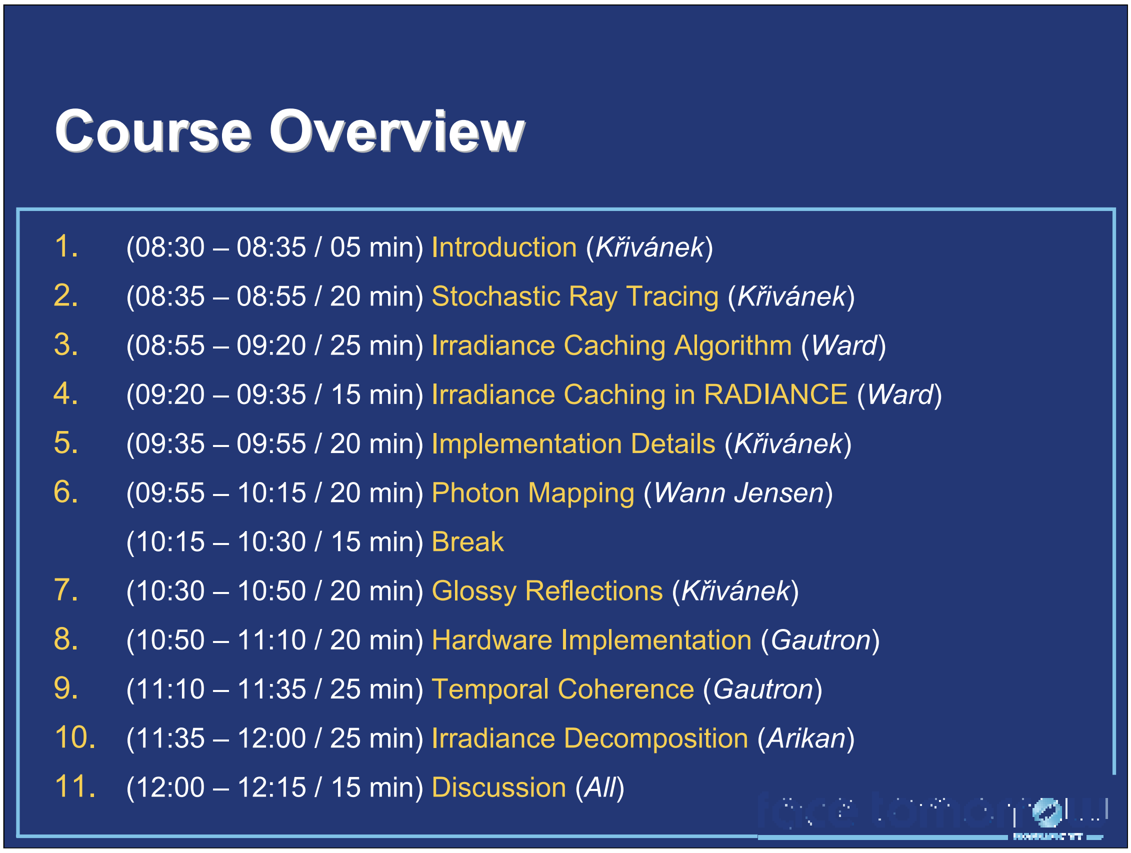


Diffuse Interreflections with Irradiance Caching

- Purpose: faster computation
- Means: sparse sampling & interpolation

Images by Okan Arikan

The focus of this course is on irradiance caching – an algorithm for fast computation of global illumination on diffuse surfaces. The algorithm gains its efficiency by sparse sampling of indirect illumination and its interpolation.



Course Overview

1. (08:30 – 08:35 / 05 min) Introduction (*Křivánek*)
2. (08:35 – 08:55 / 20 min) Stochastic Ray Tracing (*Křivánek*)
3. (08:55 – 09:20 / 25 min) Irradiance Caching Algorithm (*Ward*)
4. (09:20 – 09:35 / 15 min) Irradiance Caching in RADIANCE (*Ward*)
5. (09:35 – 09:55 / 20 min) Implementation Details (*Křivánek*)
6. (09:55 – 10:15 / 20 min) Photon Mapping (*Wann Jensen*)

   (10:15 – 10:30 / 15 min) Break

7. (10:30 – 10:50 / 20 min) Glossy Reflections (*Křivánek*)
8. (10:50 – 11:10 / 20 min) Hardware Implementation (*Gautron*)
9. (11:10 – 11:35 / 25 min) Temporal Coherence (*Gautron*)
10. (11:35 – 12:00 / 25 min) Irradiance Decomposition (*Arikan*)
11. (12:00 – 12:15 / 15 min) Discussion (*All*)

SIGGRAPH2007

Stochastic Ray Tracing

SIGGRAPH2007

Jaroslav Křivánek

ČVUT v Praze – CTU Prague

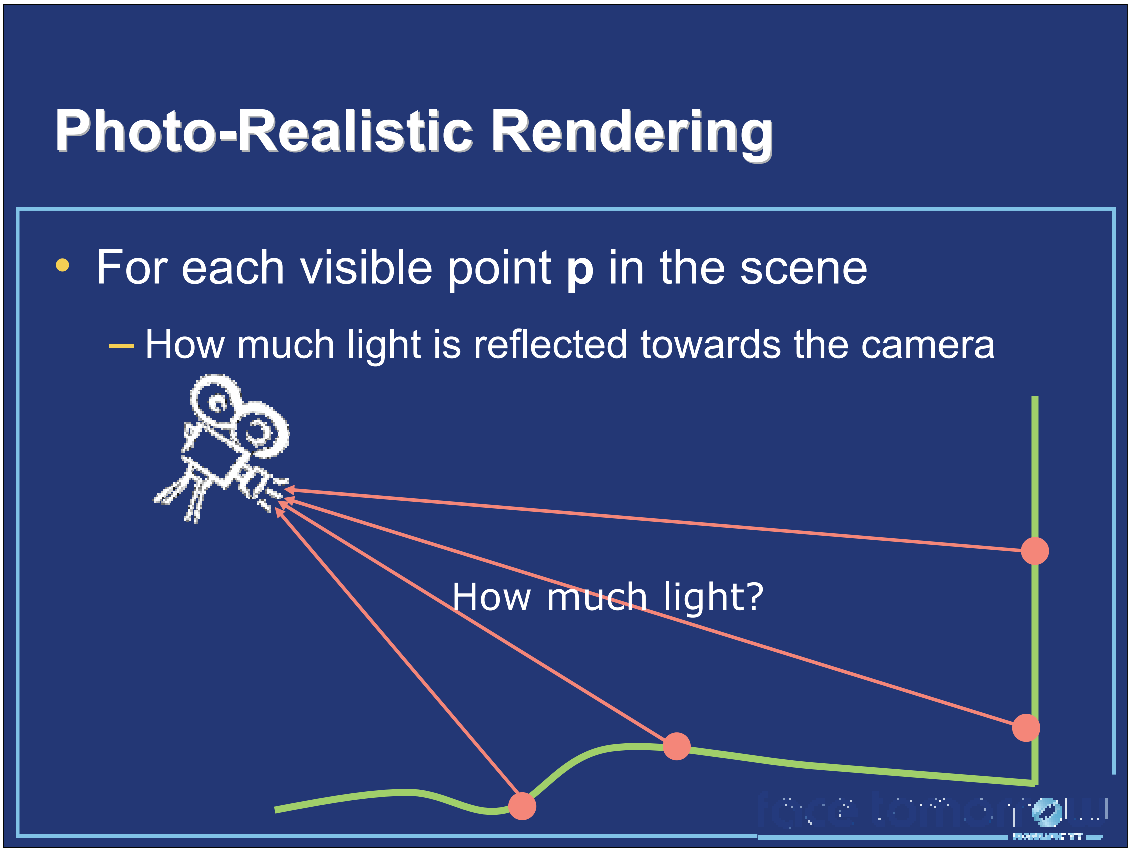
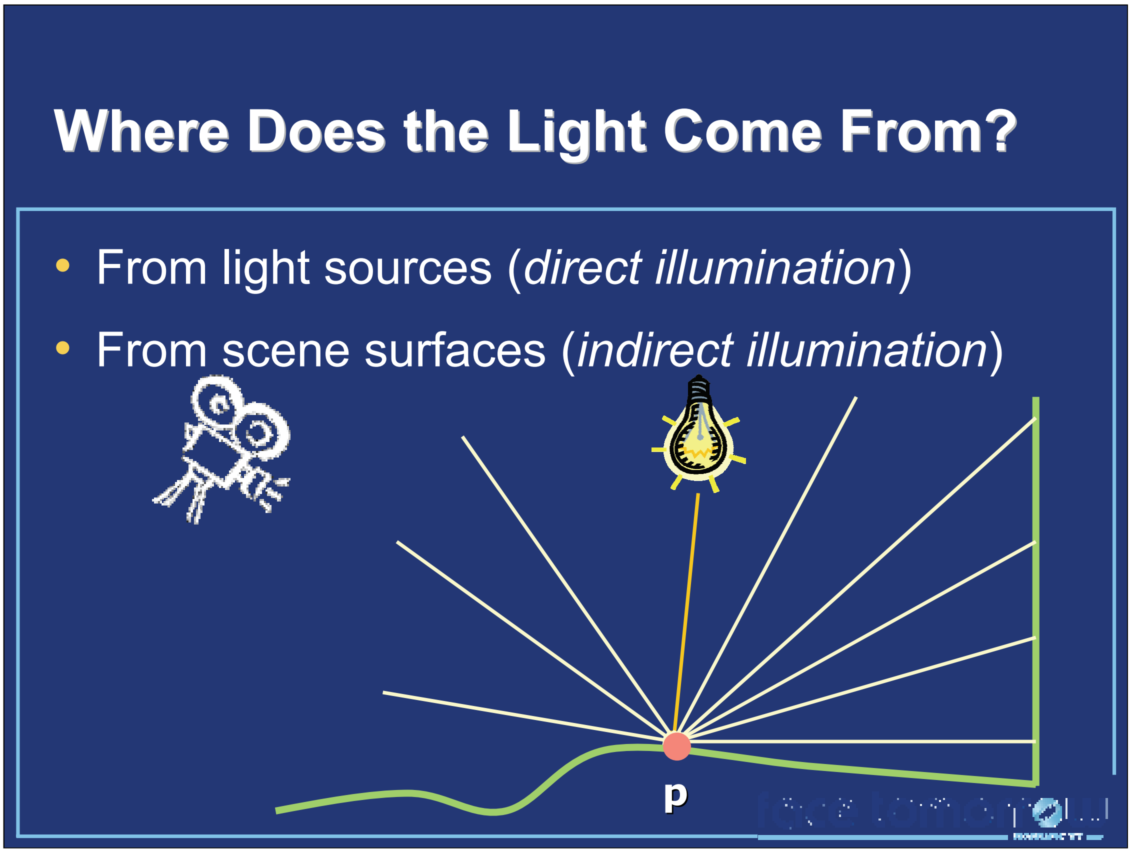


Photo-Realistic Rendering

- For each visible point **p** in the scene
  - How much light is reflected towards the camera

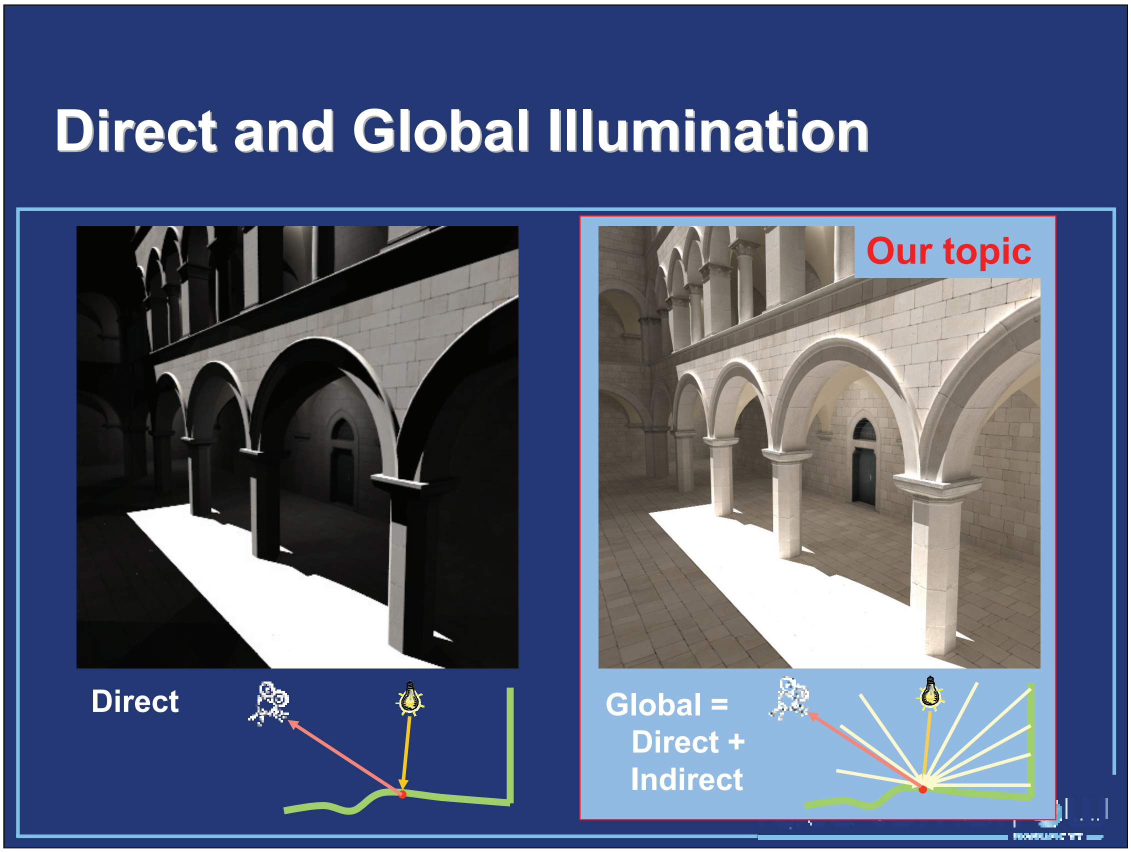
Given the description of a scene, the goal of photo-realistic rendering is to compute the amount of light reflected from visible scene surfaces that arrives to the virtual camera through image pixels. This light determines the color of image pixels.



Where Does the Light Come From?

- From light sources (*direct illumination*)
- From scene surfaces (*indirect illumination*)

Focusing on one such point, where does the light come from? Surely, it comes directly from the light sources placed in the scene – this is called the *direct illumination*. But light also comes after being reflected from other scene surfaces. This is the *indirect illumination*.



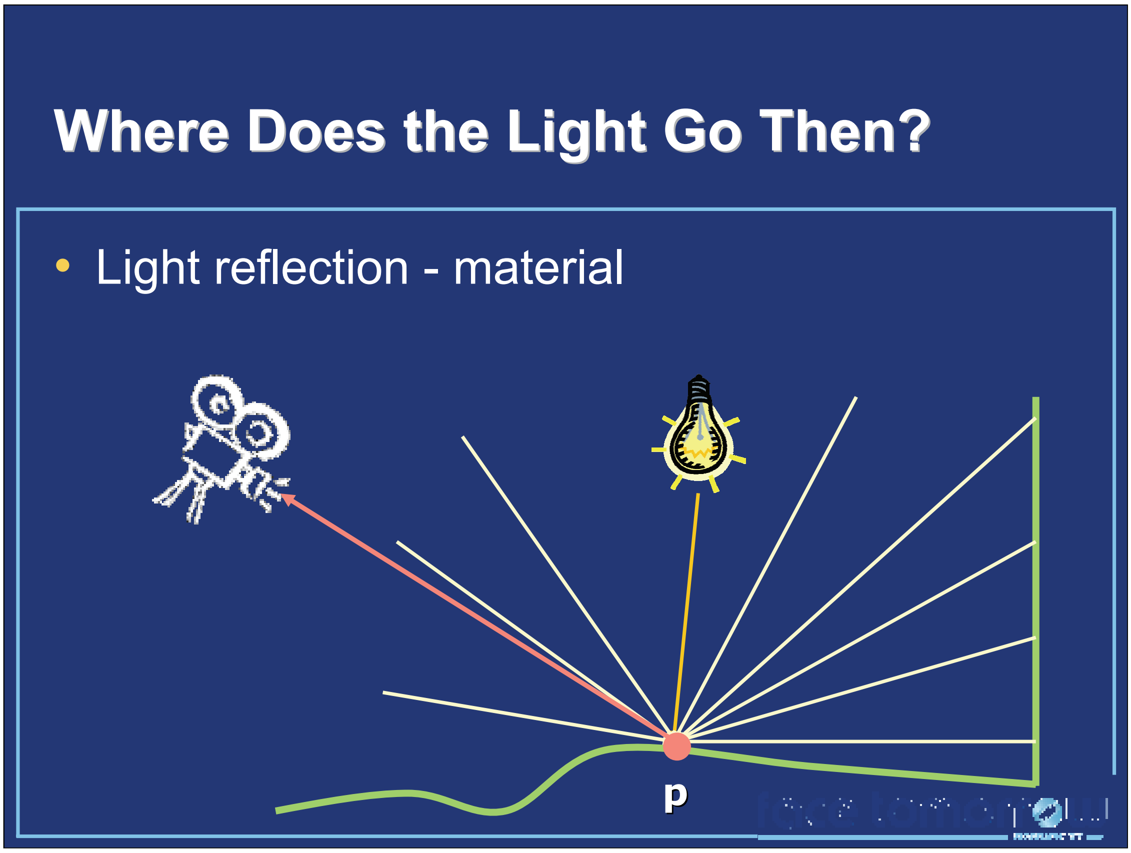
Direct and Global Illumination

Our topic

Direct

Global = Direct + Indirect

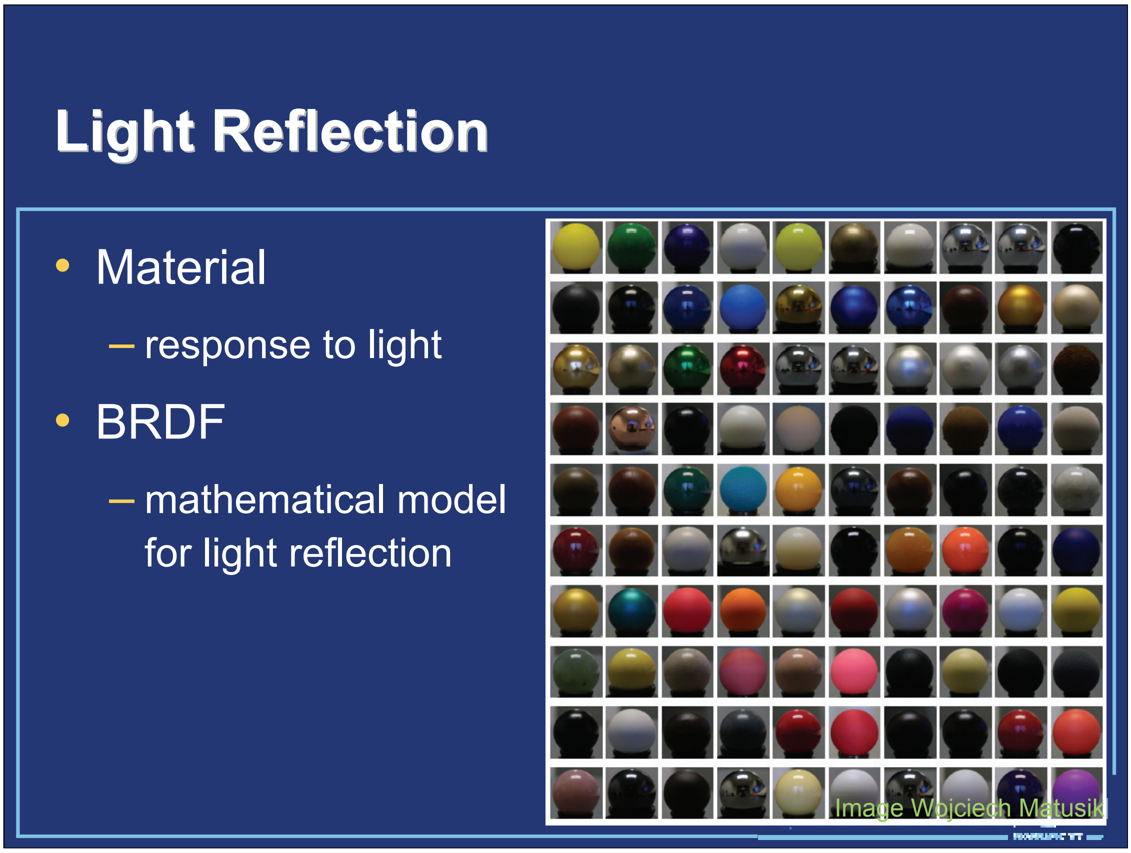
On the left is an image generated by taking into account only direct illumination. Shadows are completely black because, obviously, there is no direct illumination in the shadowed areas. On the right is the result of global illumination computation where light is allowed to bounce around in the scene before it is finally reflected towards the camera.



Where Does the Light Go Then?

- Light reflection - material

So far, we know where the light comes from at **p**. But we want to compute how much of it is reflected towards the camera. This is determined by the surface material.



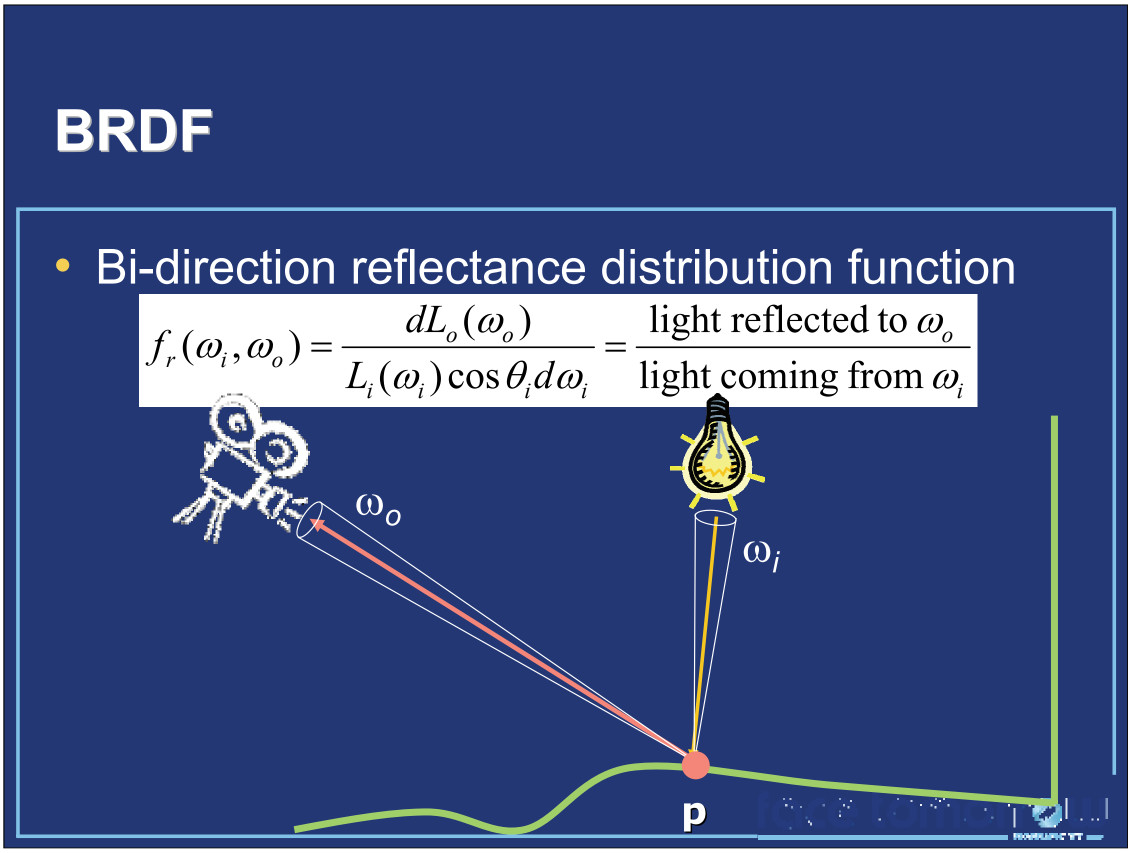
Light Reflection

- Material
  - response to light
- BRDF
  - mathematical model for light reflection

Image Wojciech Matusik

From the point of view of light simulation, the material can be looked upon as the surface’s response to incoming light. The spheres on the right are all illuminated by the same light – their varying appearance is only determined by their different material properties.

In computer graphics, the most common way to mathematically describe material reflectance characteristics is the BRDF – bidirectional reflectance distribution function.

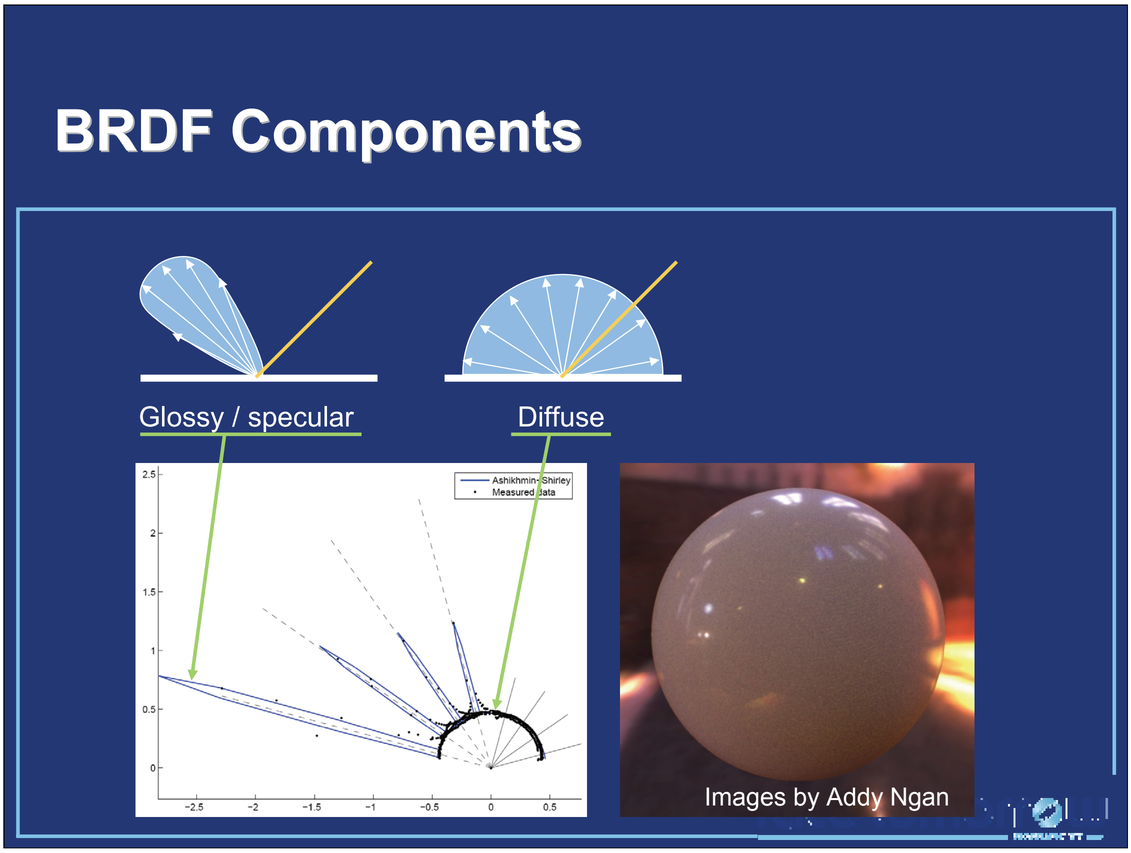


BRDF

- Bi-direction reflectance distribution function

$$f_r(\omega_i,\omega_o) = \frac{dL_o(\omega_o)}{L_i(\omega_i)\cos\theta_i d\omega_i} = \frac{\text{light reflected to }\omega_o}{\text{light coming from }\omega_i}$$

The BRDF at a single point has two parameters: the incoming and the outgoing light directions, w_i and w_o, respectively. For our purposes, it is sufficient to say that the BRDF value is the ratio of the light reflected in the outgoing direction to the light coming from the incoming direction. In radiometry, the quantity associated with the amount of light reaching a point **p** along a direction w is the radiance $L(\mathbf{p},w)$.



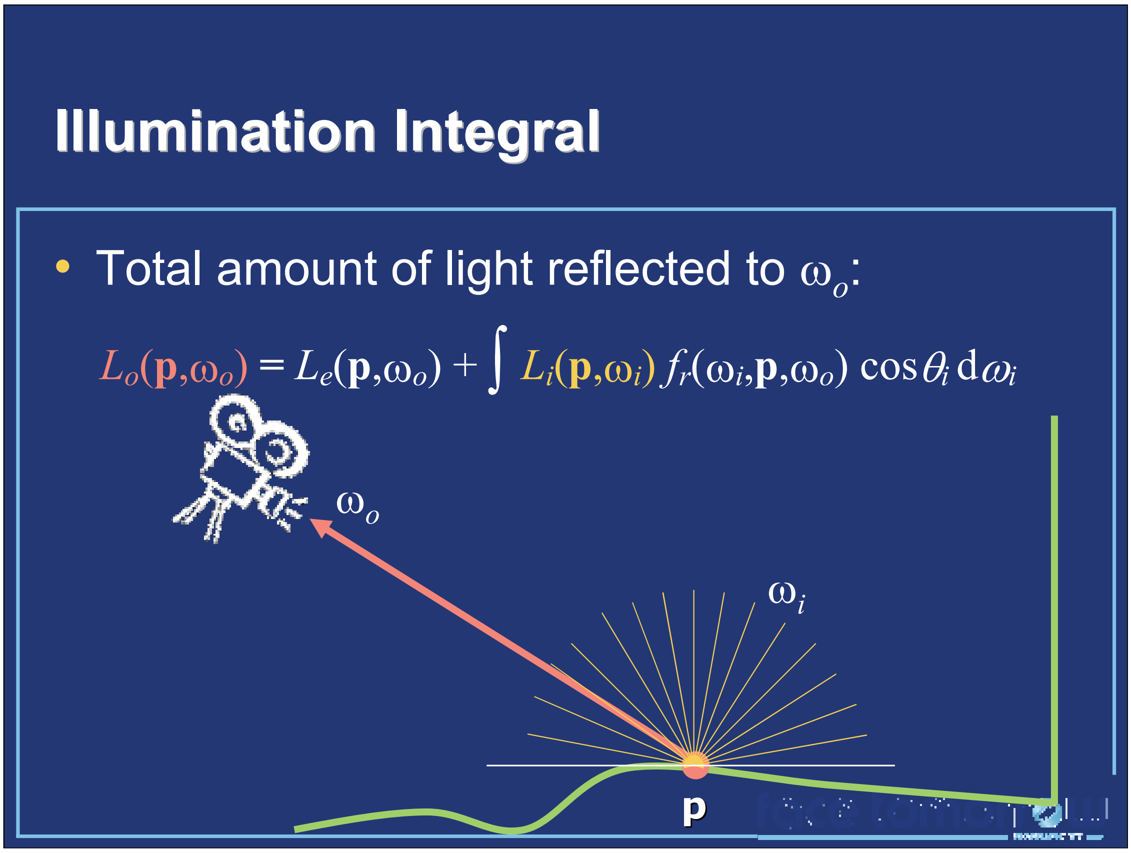
BRDF Components

Glossy / specular

Diffuse

Images by Addy Ngan

For practical purposes, we express a general BRDF as a sum of several terms, or components. On this slide you see a plot of the BRDF of an acrylic white paint (from the supplemental material for [Ngan 2002]). The BRDF is plotted for four different outgoing directions. For each fixed outgoing direction, the BRDF is a function of only the incoming direction – this is the BRDF lobe. There is a prominent diffuse component and clearly visible specular component added to it. The diffuse BRDF component is a constant function, independent of the incoming and outgoing direction. It corresponds to the 'color' of an obejct. The specular BRDF component has significant values only in a narrow cone. The specular component adds the highlight on the sphere.

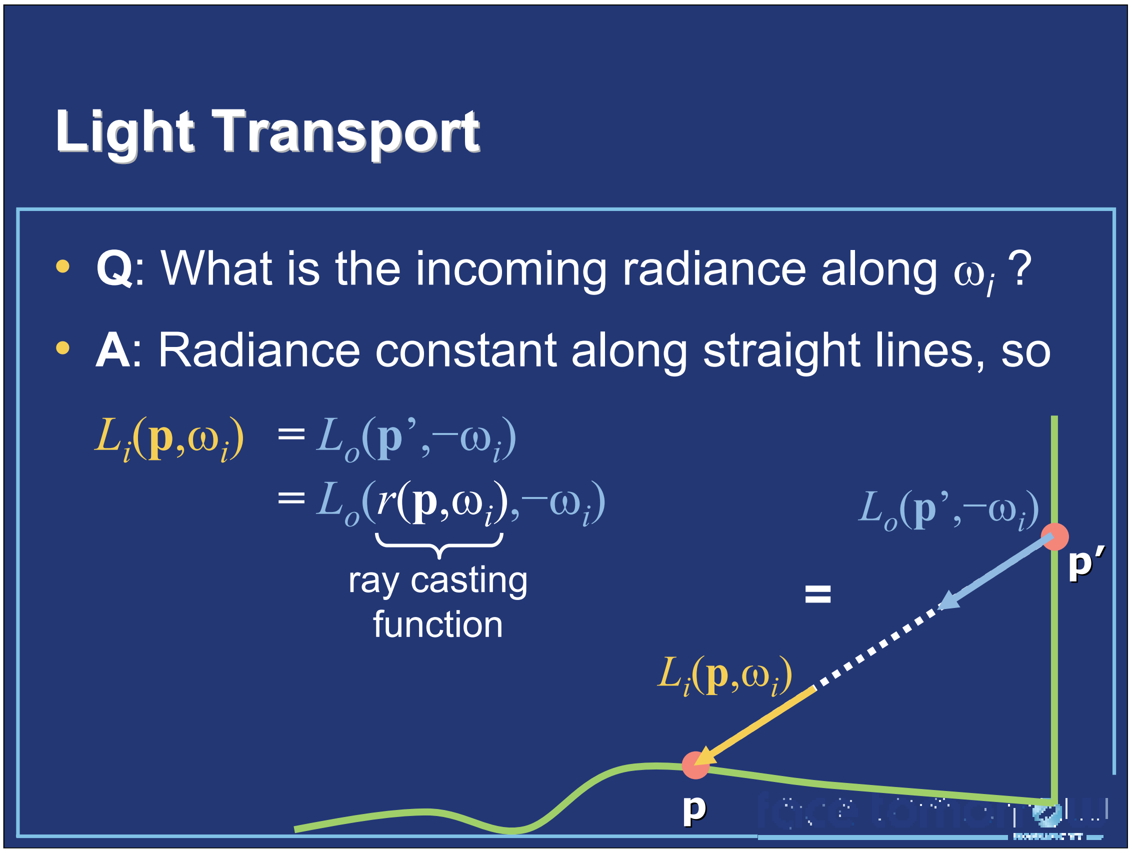


Illumination Integral

- Total amount of light reflected to ω_o:

$$L_o(\mathbf{p},\omega_o) = L_e(\mathbf{p},\omega_o) + \int L_i(\mathbf{p},\omega_i)\, f_r(\omega_i,\mathbf{p},\omega_o) \cos\theta_i \, d\omega_i$$

Let’s go back to image rendering. We want to compute the amount of light going from **p** towards the camera. Assuming we know how much light is coming to **p** from all possible directions (the incoming hemisphere), the reflected light can be computed by evaluating the illumination integral: For each incoming direction, we multiply the radiance from that direction by the BRDF and the cosine term. To get the total amount of reflected light, we sum (integrate) these contributions over all incoming directions. Then we add the self emitted light at **p** (in case **p** is on the light source) and we have the total amount of light going from **p** towards the camera.

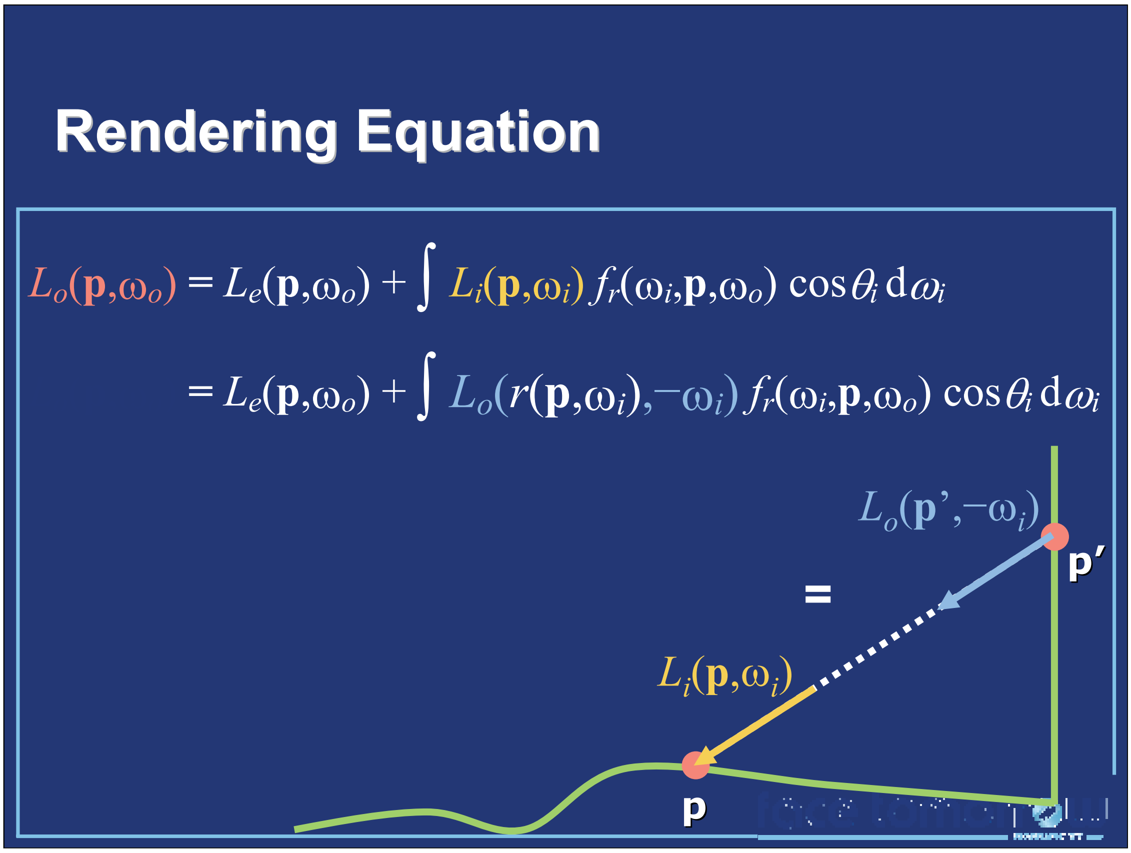


Light Transport

- **Q**: What is the incoming radiance along ω_i ?
- **A**: Radiance constant along straight lines, so

$$\begin{aligned} L_i(\mathbf{p},\omega_i) &= L_o(\mathbf{p}',-\omega_i) \\ &= L_o(\underbrace{r(\mathbf{p},\omega_i)}_{\text{ray casting function}},-\omega_i) \end{aligned}$$

We made an important assumption in computing the illumination integral. We assumed to know how much light is coming to **p** from each direction. But how do we find this out? Taking advantage of the fact that *radiance does not change along straight lines*, we see that the incoming radiance from direction w_i is equal to the outgoing radiance at a point **p**’ visible from **p** along w_i.

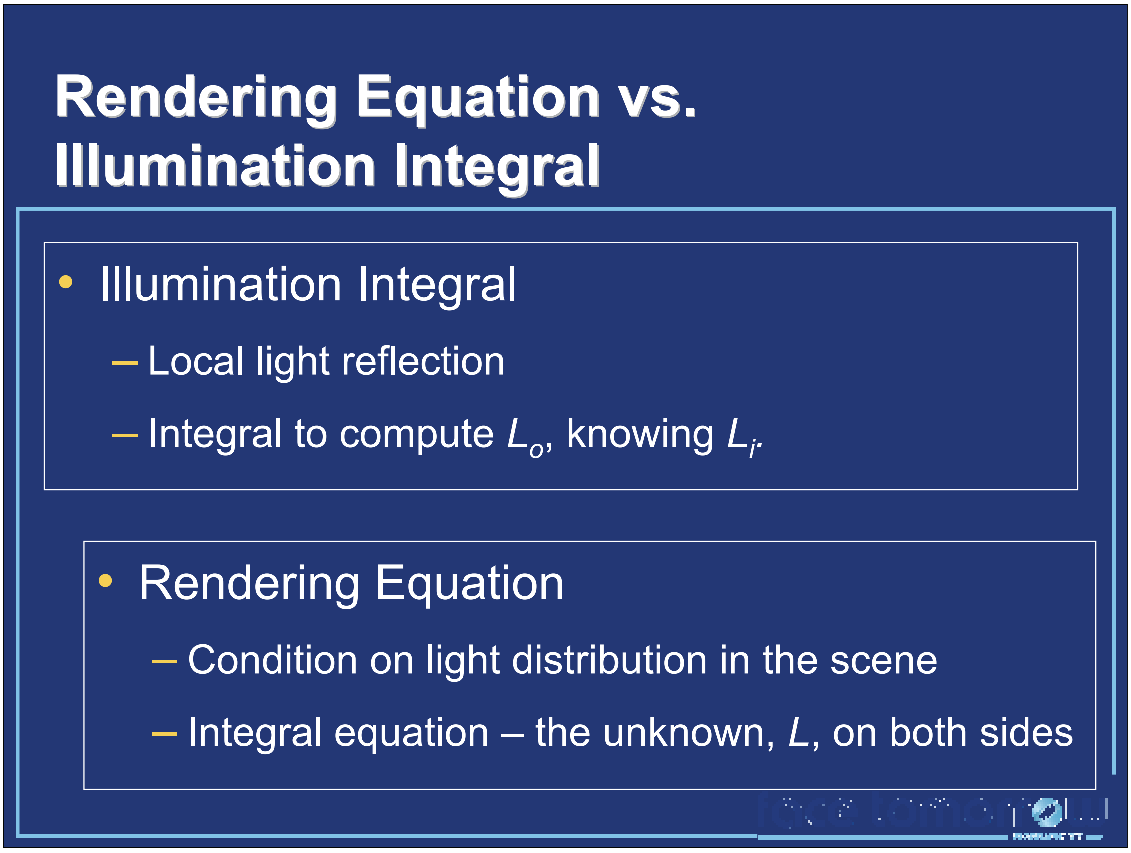


Rendering Equation

$$L_o(\mathbf{p},\omega_o) = L_e(\mathbf{p},\omega_o) + \int L_i(\mathbf{p},\omega_i)\, f_r(\omega_i,\mathbf{p},\omega_o)\cos\theta_i\, d\omega_i$$

$$= L_e(\mathbf{p},\omega_o) + \int L_o(r(\mathbf{p},\omega_i),-\omega_i)\, f_r(\omega_i,\mathbf{p},\omega_o)\cos\theta_i\, d\omega_i$$

Plugging our knowledge of light transport to the illumination integral yields *the rendering equation* [Kajiya 1986].

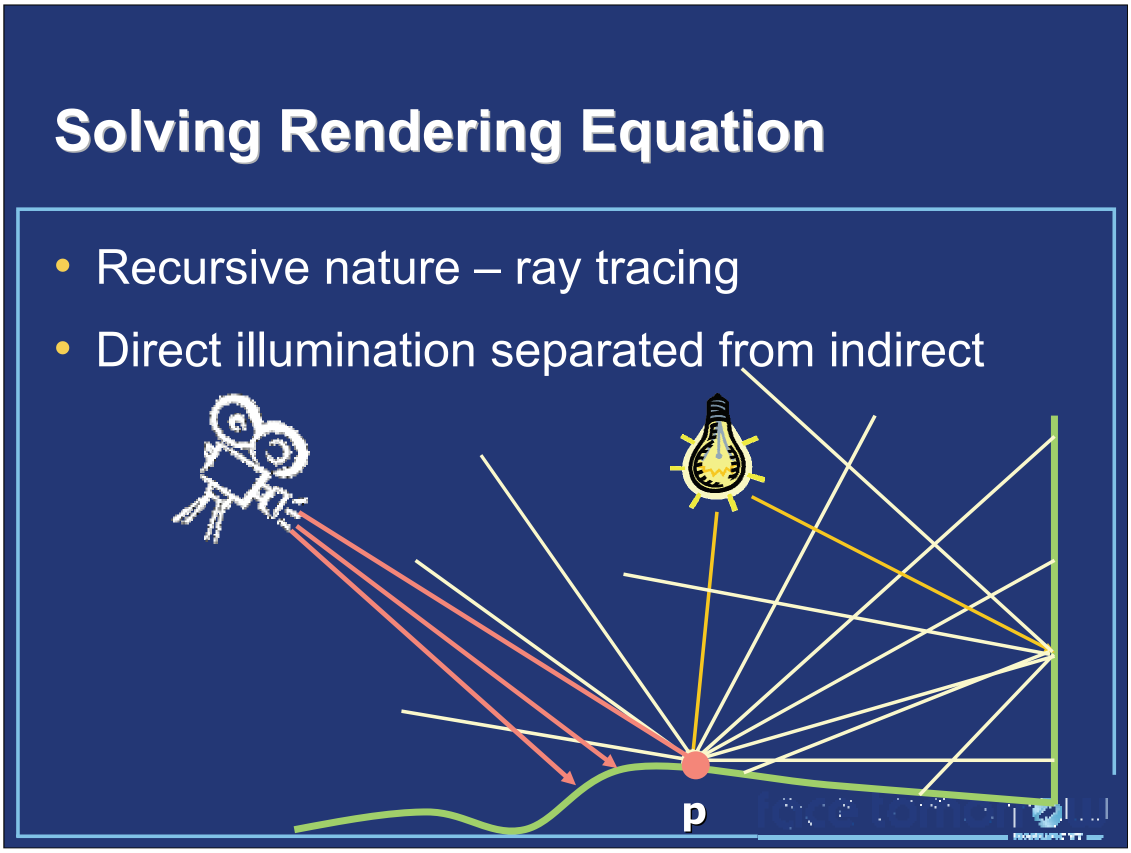


Rendering Equation vs. Illumination Integral

- Illumination Integral
  - Local light reflection
  - Integral to compute L_o, knowing L_i.

- Rendering Equation
  - Condition on light distribution in the scene
  - Integral equation – the unknown, L, on both sides

Unlike the illumination integral, which is simply a formula that computes reflected light from incident light and therefore has local character, the rendering equation has a more profound meaning. It is truly an ‘equation’ – the unknown quantity, radiance L, – appears on both sides. The rendering equation expresses a condition on equilibrium light distribution in the whole scene.

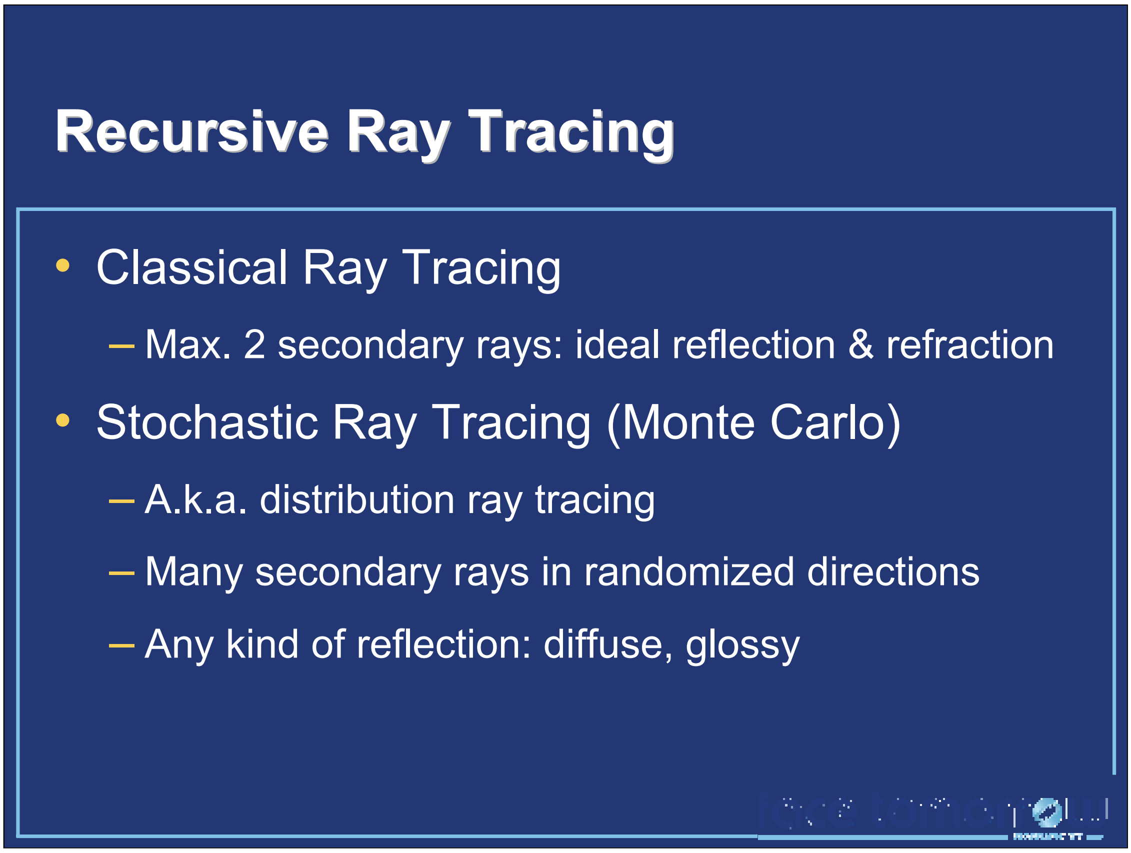


Solving Rendering Equation

- Recursive nature – ray tracing
- Direct illumination separated from indirect

Although the rendering equation is at the heart of all theoretical analyses of global illumination, our solution strategy – recursive ray tracing – is much more reminiscent of the illumination integral. At each point of the scene, we want to estimate the outgoing light. In order to do this, we want to numerically evaluate the illumination integral. So we shoot secondary rays, which are ‘samples’ of the incoming radiance. These samples are then multiplied by the BRDF and averaged to numerically estimate the outgoing radiance.

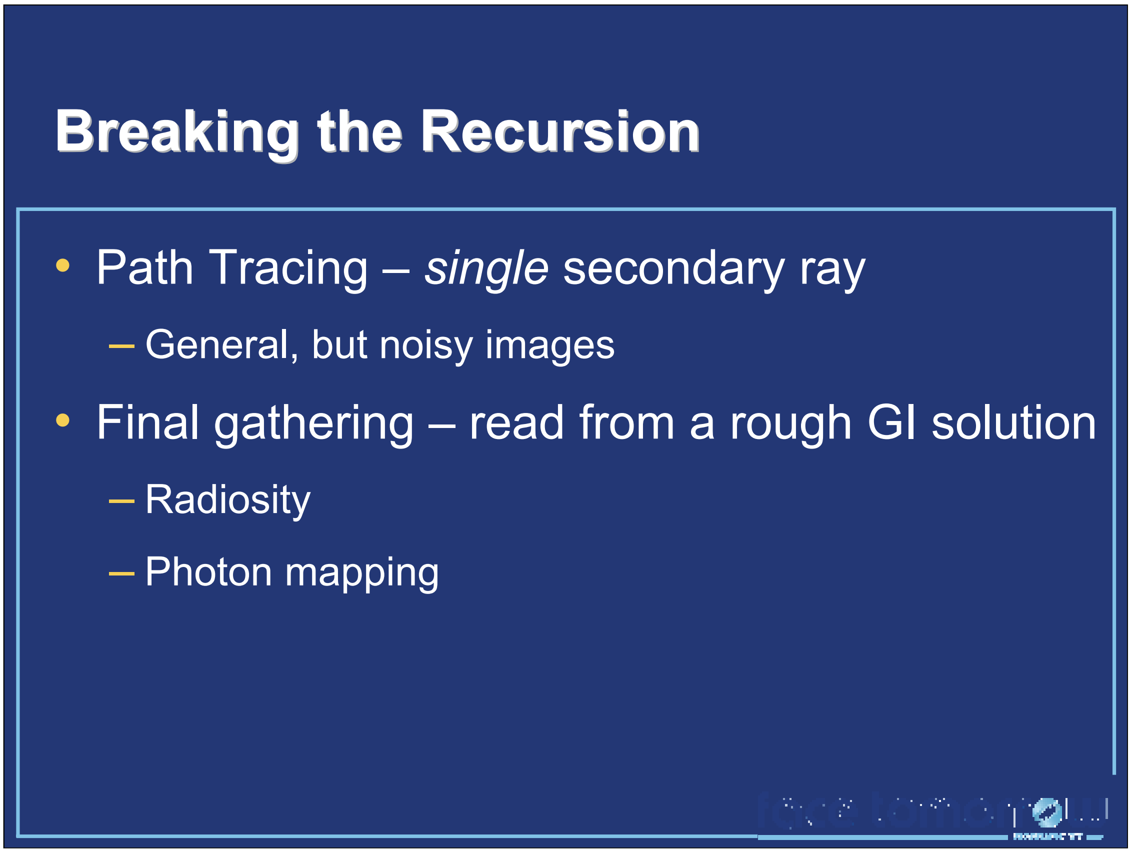
For each of these secondary rays, we need to evaluated the outgoing radiance at a point where they intersect the scene. So we use the same procedure recursively again and again. The recursion can be limited since each light reflection takes away some of the transported light energy. So after couple of reflections, the contribution of light to the image becomes insignificant.



Recursive Ray Tracing

- Classical Ray Tracing
  - Max. 2 secondary rays: ideal reflection & refraction
- Stochastic Ray Tracing (Monte Carlo)
  - A.k.a. distribution ray tracing
  - Many secondary rays in randomized directions
  - Any kind of reflection: diffuse, glossy

In the classical ray tracing, we shoot at most two secondary rays – in the direction of ideal specular reflection and refraction. That means we disregard indirect illumination on anything but ideal specular reflectors or refractors (such as water, glass etc.) If we want to add indirect light on surfaces of arbitrary reflective characteristics (glossy, diffuse), we need to send many rays to estimate the illumination integral – this is most often referred to as the *distribution ray tracing*.

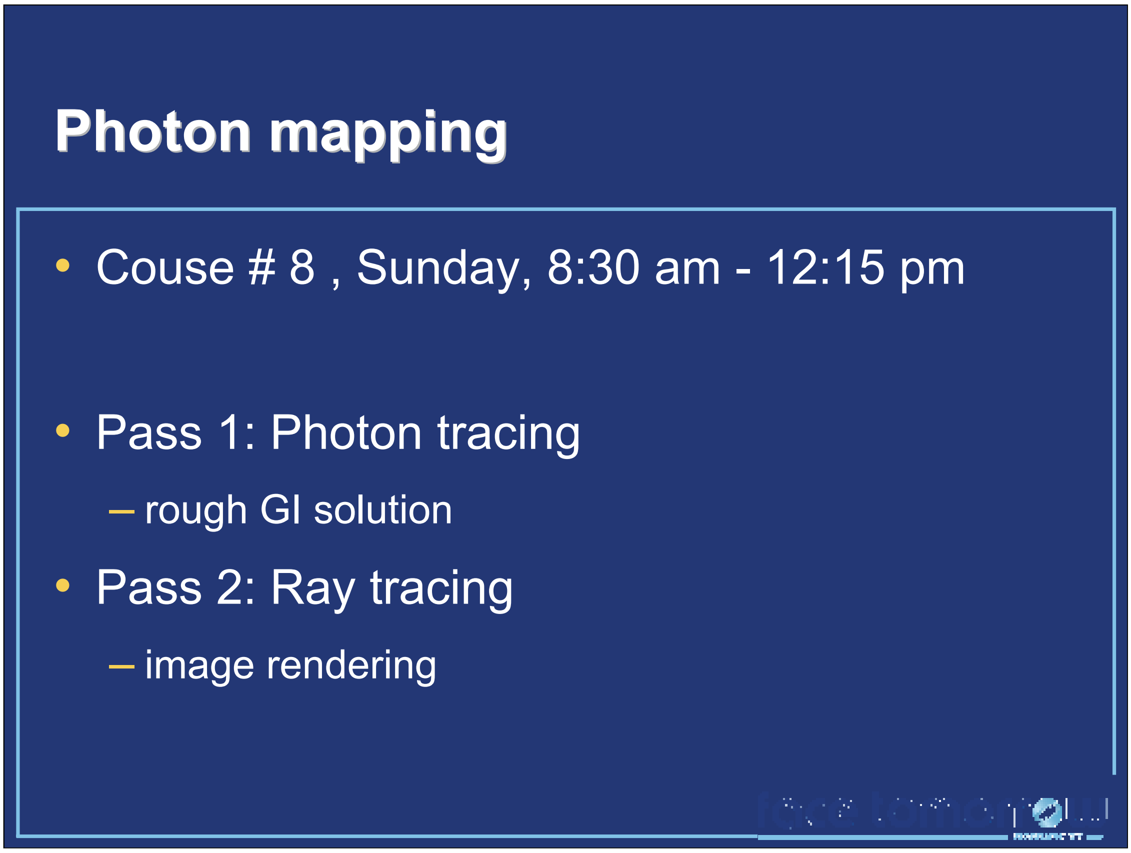


Breaking the Recursion

- Path Tracing – *single* secondary ray
  - General, but noisy images
- Final gathering – read from a rough GI solution
  - Radiosity
  - Photon mapping

One serious problem of distribution ray tracing is the explosion of the number of rays due to the recursion, where each ray is split into many rays upon each reflection. One way to combat this problem is path tracing where only one single ray is generated at each reflection. The rays then form linear ‘paths’ instead of the ray tree in distribution ray tracing. Path tracing is the original strategy that Jim Kajiya [1986] proposed for solving the rendering equation. It is very general and simple to implement. However, thousands of paths have to be traced through each image pixel in order to compute images without visible noise, which is time-consuming.

Another way to break the recursion is to compute a rough global illumination solution in a pre-process and store it in the scene. Then, as we trace the camera rays, we can read the rough GI solution instead of continuing the recursion. In practice, the most common way of doing this is to use photon mapping.

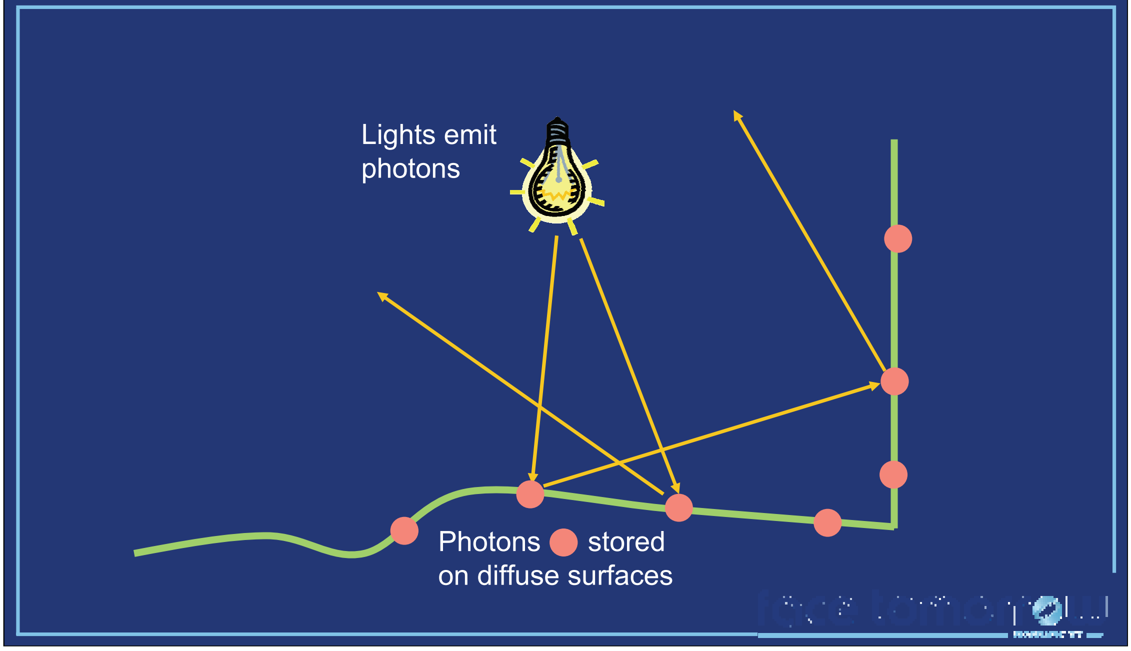


Photon mapping

- Couse # 8 , Sunday, 8:30 am - 12:15 pm
- Pass 1: Photon tracing
  - rough GI solution
- Pass 2: Ray tracing
  - image rendering

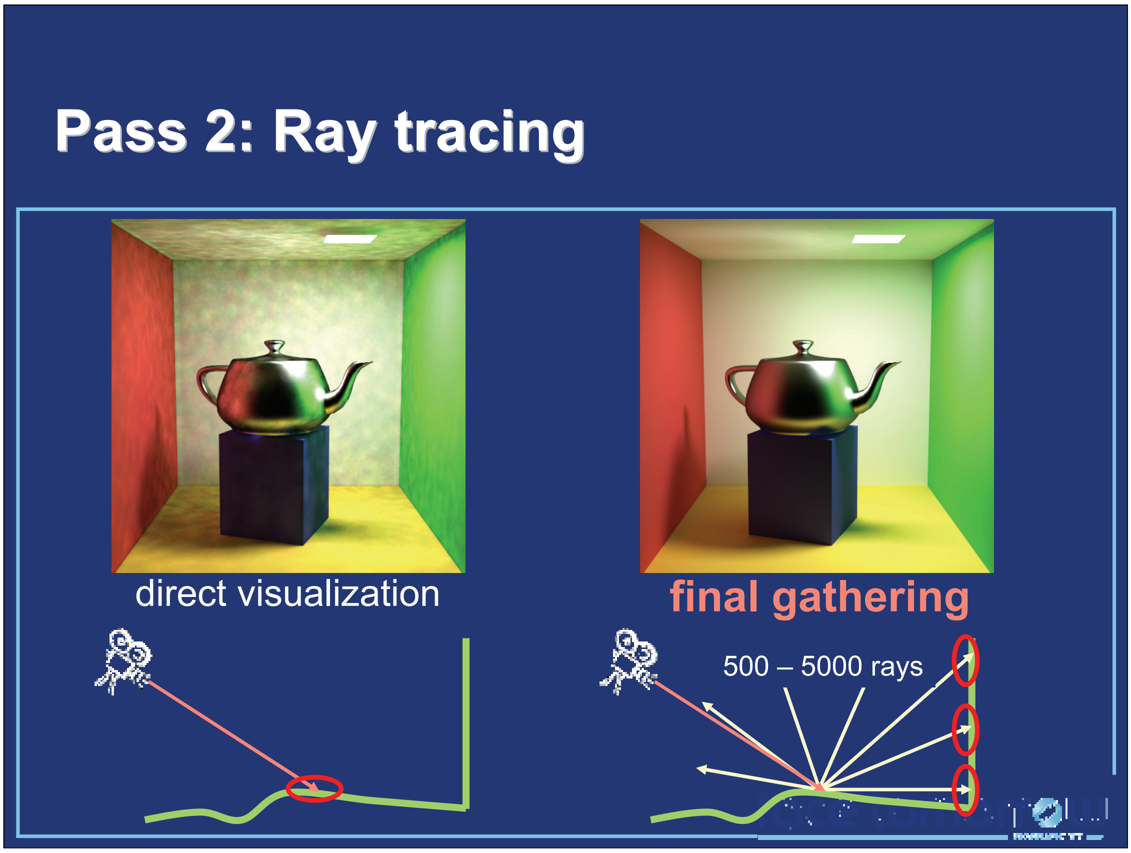
Photon mapping works in two passes. In the first pass – photon tracing – a rough GI solution is created in form of ‘photons’ distributed in the scene. The second pass is distribution ray tracing as described previously, where recursion is limited by reading the rough GI solution from the photon map.

Pass 1: Photon tracing



Lights emit photons

Photons ● stored on diffuse surfaces



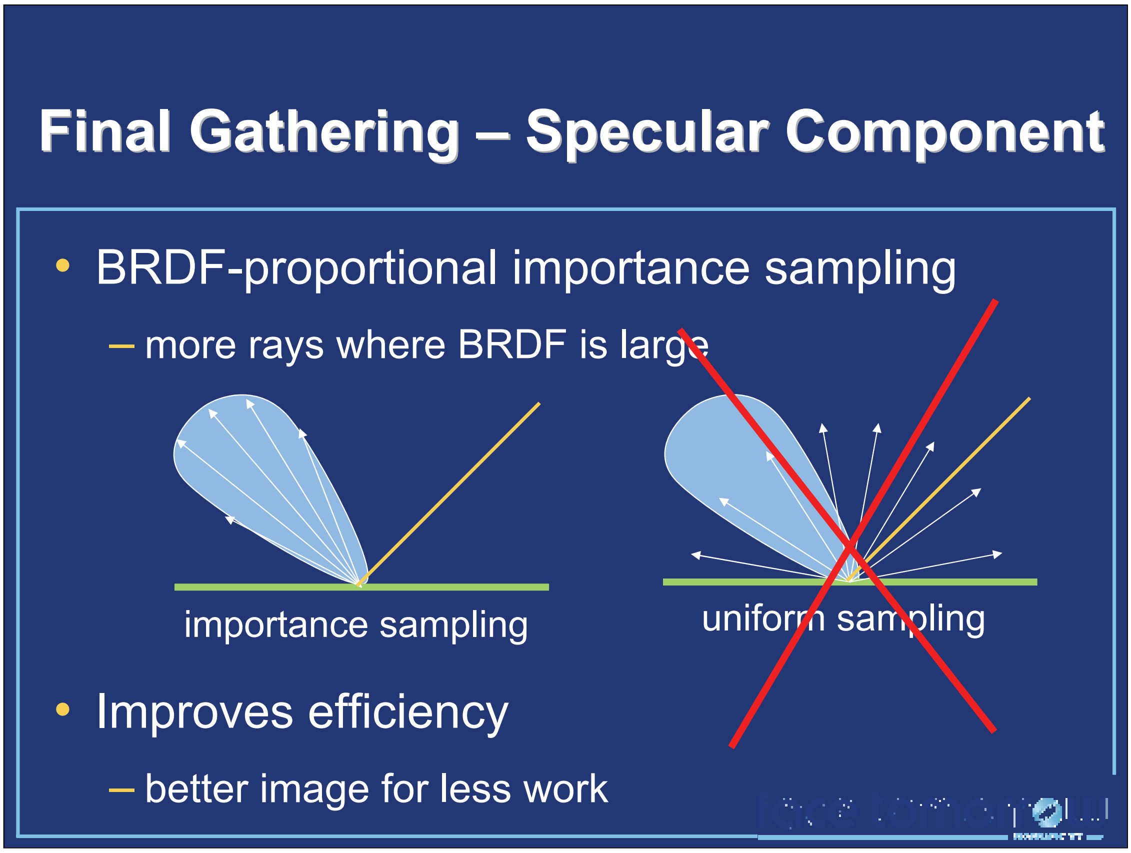
Pass 2: Ray tracing

direct visualization

final gathering

In ray tracing with photon maps, one possibility is to read the rough GI solution from the photon map immediately as a primary ray hits a surface. However, this gives a ‘splotchy’ artifacts in the images, precisely because the GI solution in the photon map is very rough.

A better way is to perform one level of distribution ray tracing and read the photon map after the first reflection. Shooting many secondary rays averages out any artifacts and the result is a clean image with global illumination. This technique is called *final gathering*. Final gathering as described so far is *slow*: for each of the thousands or millions primary rays, hundreds or thousand gather rays have to be shot.



Final Gathering – Specular Component

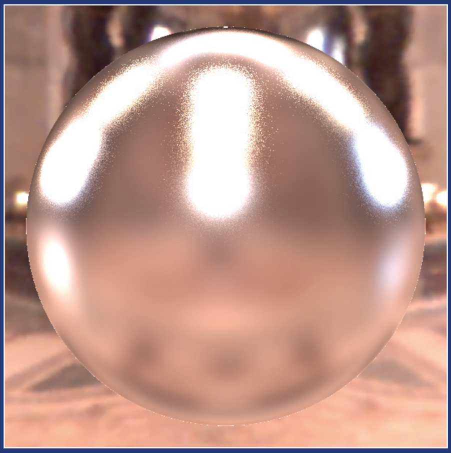
- BRDF-proportional importance sampling
  - more rays where BRDF is large

- Improves efficiency
  - better image for less work

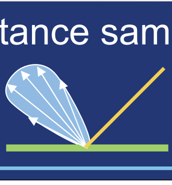
On specular surfaces, an extremely powerful way to reduce the cost of distribution ray tracing – and final gathering – is *importance sampling*. Since the BRDF lobe corresponding to the (known) outgoing direction has significant values only in a narrow cone, there is no use wasting time in sampling the whole hemisphere uniformly. It is much better to concentrate the rays within that cone.

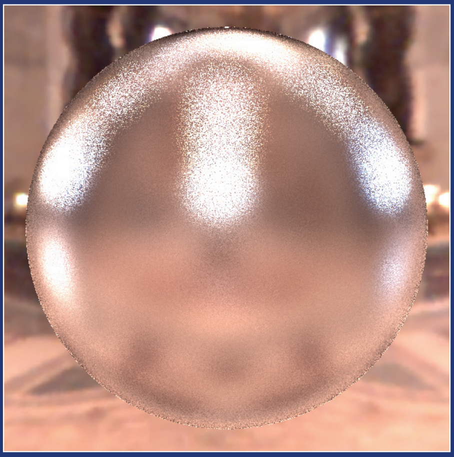
Importance sampling improves efficiency – we get the same quality results with much fewer rays (or much better quality with the same number of rays). The more specular surface, the more effective importance sampling.

Final Gathering – Specular Component

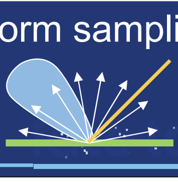


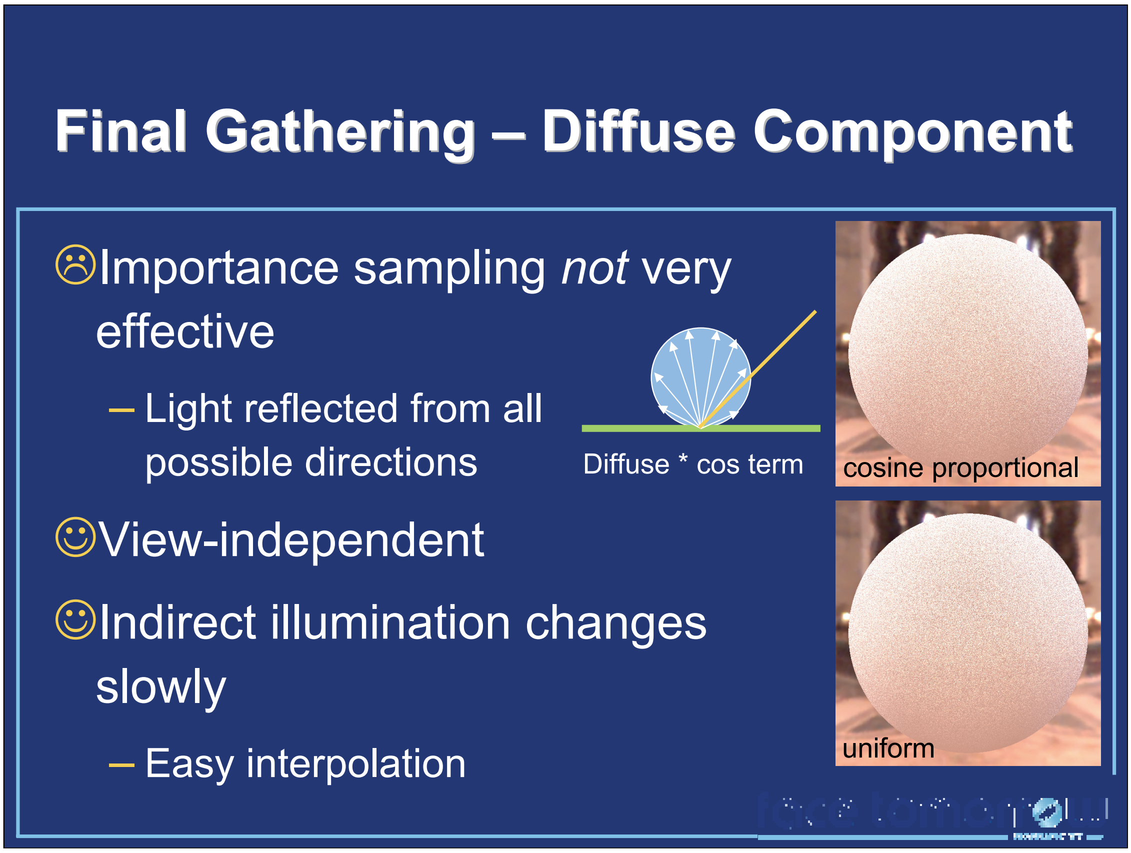
importance sampling





uniform sampling



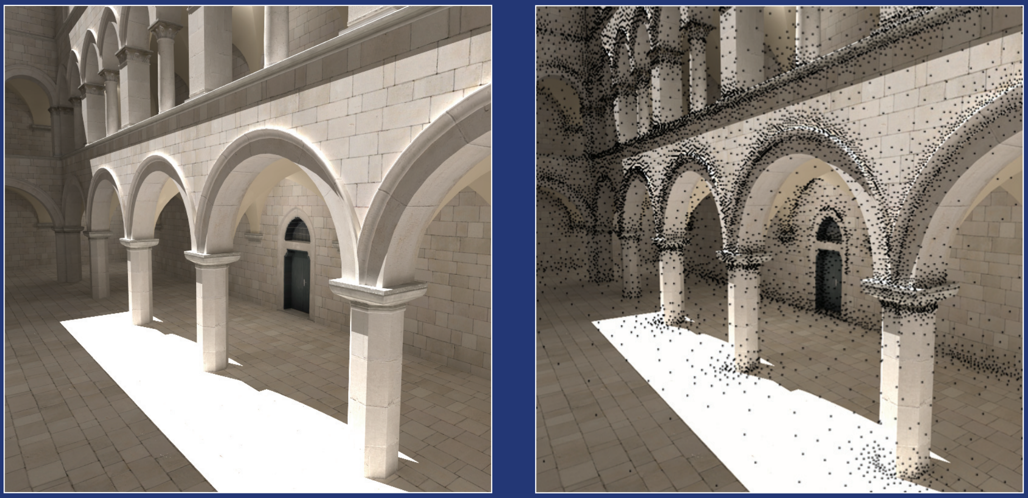


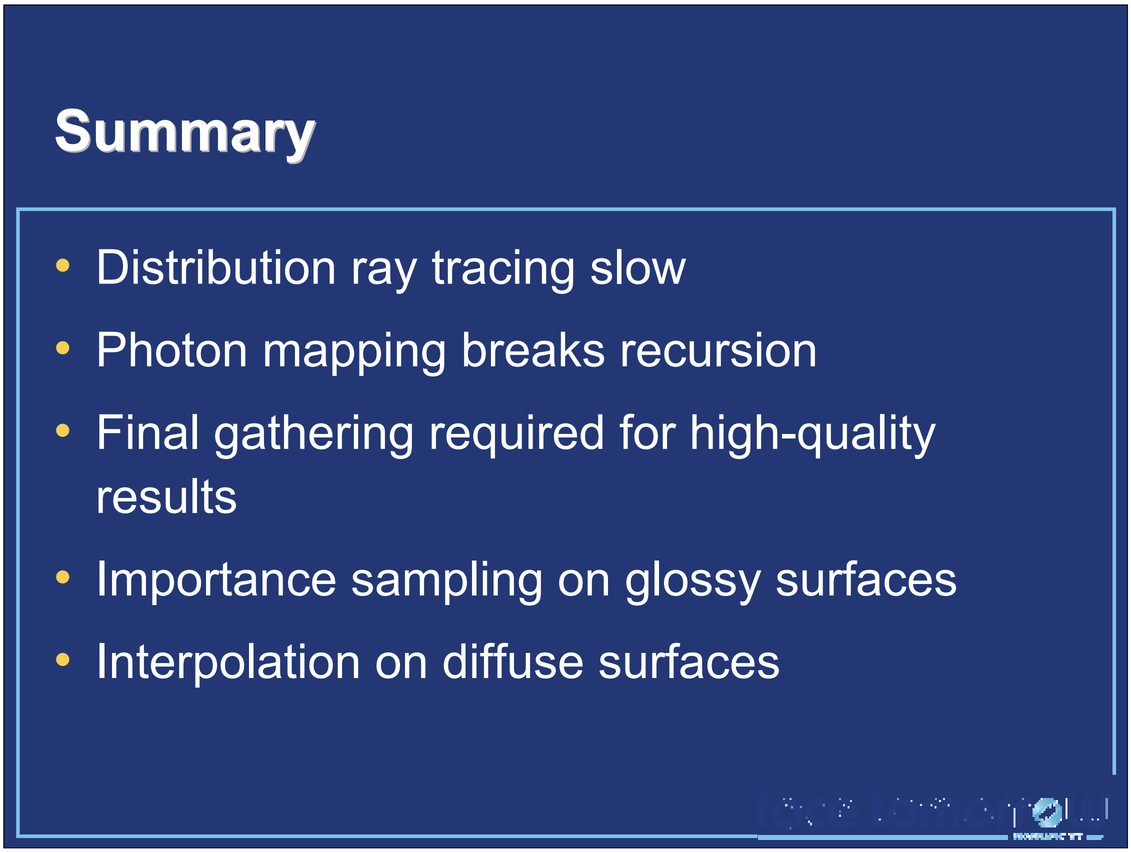
Final Gathering – Diffuse Component

- ☹Importance sampling *not* very effective
  - Light reflected from all possible directions
- ☺View-independent
- ☺Indirect illumination changes slowly
  - Easy interpolation

Unfortunately, importance sampling isn’t very effective on diffuse surfaces, since light is reflected nearly equally from all incoming directions. Two good news help us to make distribution ray tracing on glossy surface efficient: 1) indirect illumination is view-independent and 2) it changes very slowly over surfaces. This makes it easy to compute indirect illumination sparsely in the scene and interpolate.

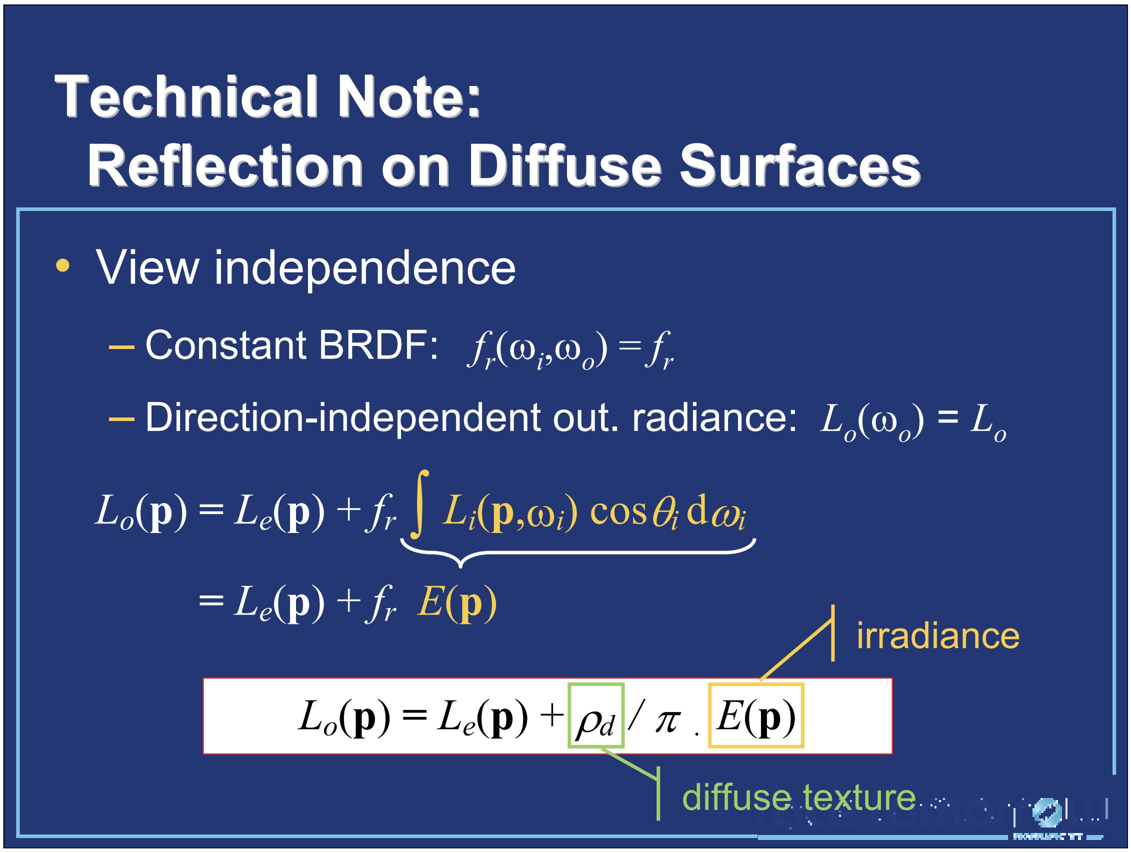
Final Gathering with Irradiance Caching





Summary

- Distribution ray tracing slow
- Photon mapping breaks recursion
- Final gathering required for high-quality results
- Importance sampling on glossy surfaces
- Interpolation on diffuse surfaces



Technical Note:
Reflection on Diffuse Surfaces

- View independence
  - Constant BRDF: $f_r(\omega_i,\omega_o) = f_r$
  - Direction-independent out. radiance: $L_o(\omega_o) = L_o$

$$L_o(\mathbf{p}) = L_e(\mathbf{p}) + f_r \underbrace{\int L_i(\mathbf{p},\omega_i)\cos\theta_i\,\mathrm{d}\omega_i}$$

$$= L_e(\mathbf{p}) + f_r\ E(\mathbf{p})$$

irradiance

$$\boxed{L_o(\mathbf{p}) = L_e(\mathbf{p}) + \boxed{\rho_d} / \pi \ .\ \boxed{E(\mathbf{p})}}$$

diffuse texture

The view independence of diffuse materials means that the outgoing radiance $L_o(w_o)$ is independent of the direction, w_o. This implies that the BRDF is constant in both the incoming and the outgoing directions and, in the illumination integral, it can be taken out of the integration. What remains is the integral of the incoming radiance multiplied by the cosine term – this is the irradiance *E*. Irradiance expresses the area density of light energy and is measured in Watts per square meter.

The BRDF of the diffuse surface is uniquely determined by one color triple – the reflectance rho_d. Reflectance of a real surface must be between zero and one so that reflection conserves energy. Reflectance is the quantity that corresponds to the diffuse texture. Beware! It must be divided by pi to get the BRDF value.

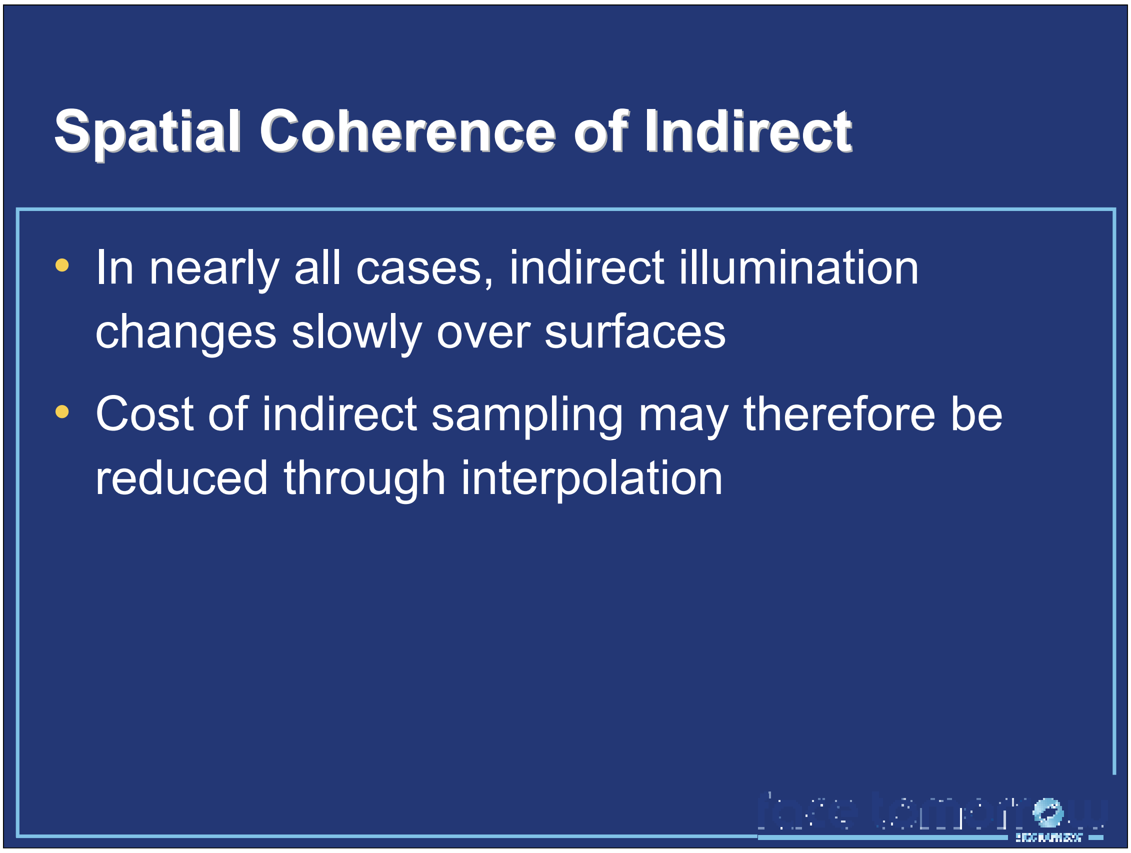
SIGGRAPH2007

Irradiance Caching Algorithm

SIGGRAPH2007

Greg Ward

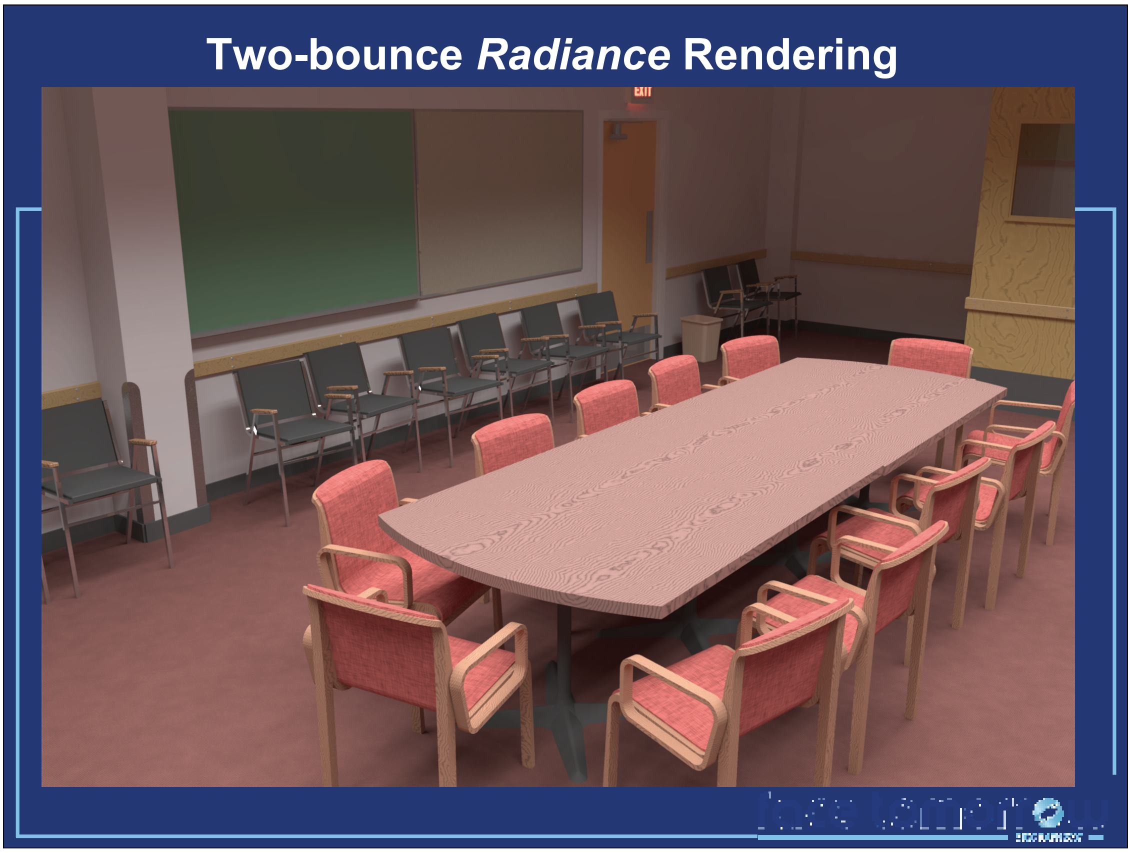
Anyhere Software



Spatial Coherence of Indirect

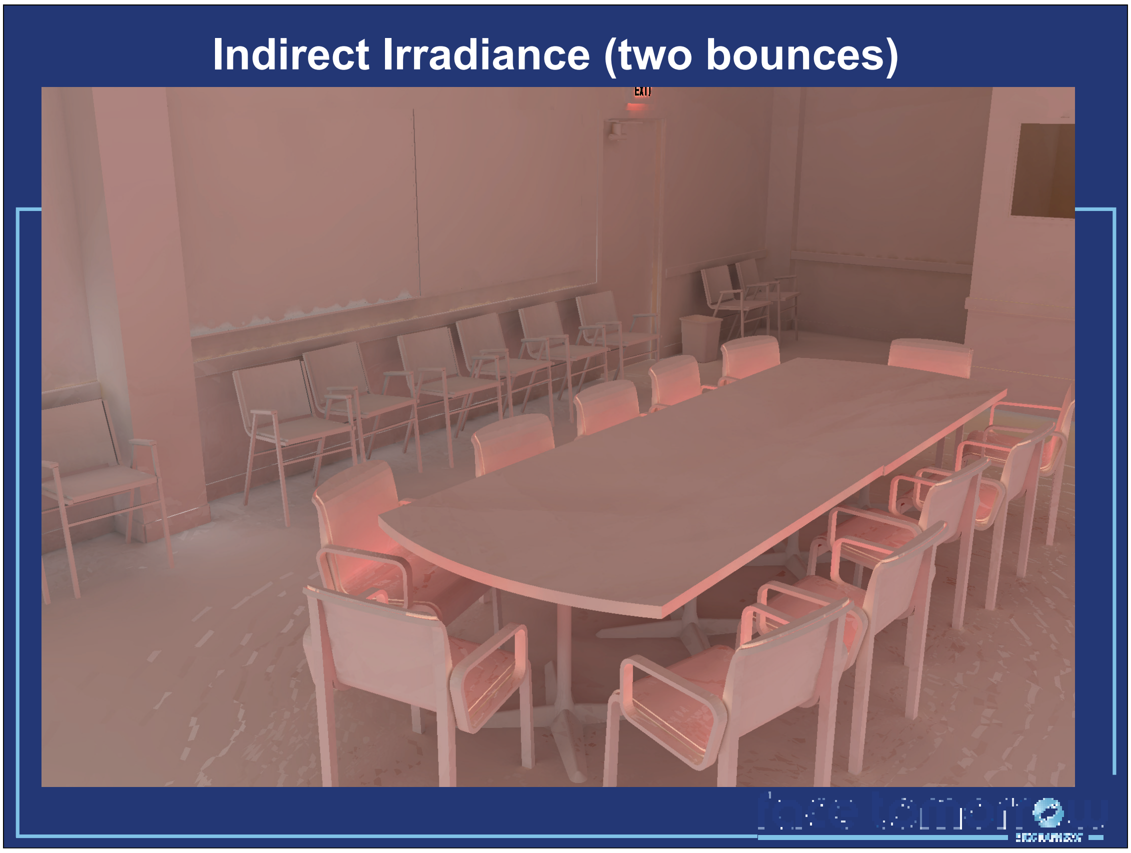
- In nearly all cases, indirect illumination changes slowly over surfaces
- Cost of indirect sampling may therefore be reduced through interpolation

Radiosity techniques work for the same reason -- and have trouble with point light sources because direct illumination does *not* change slowly over surfaces.



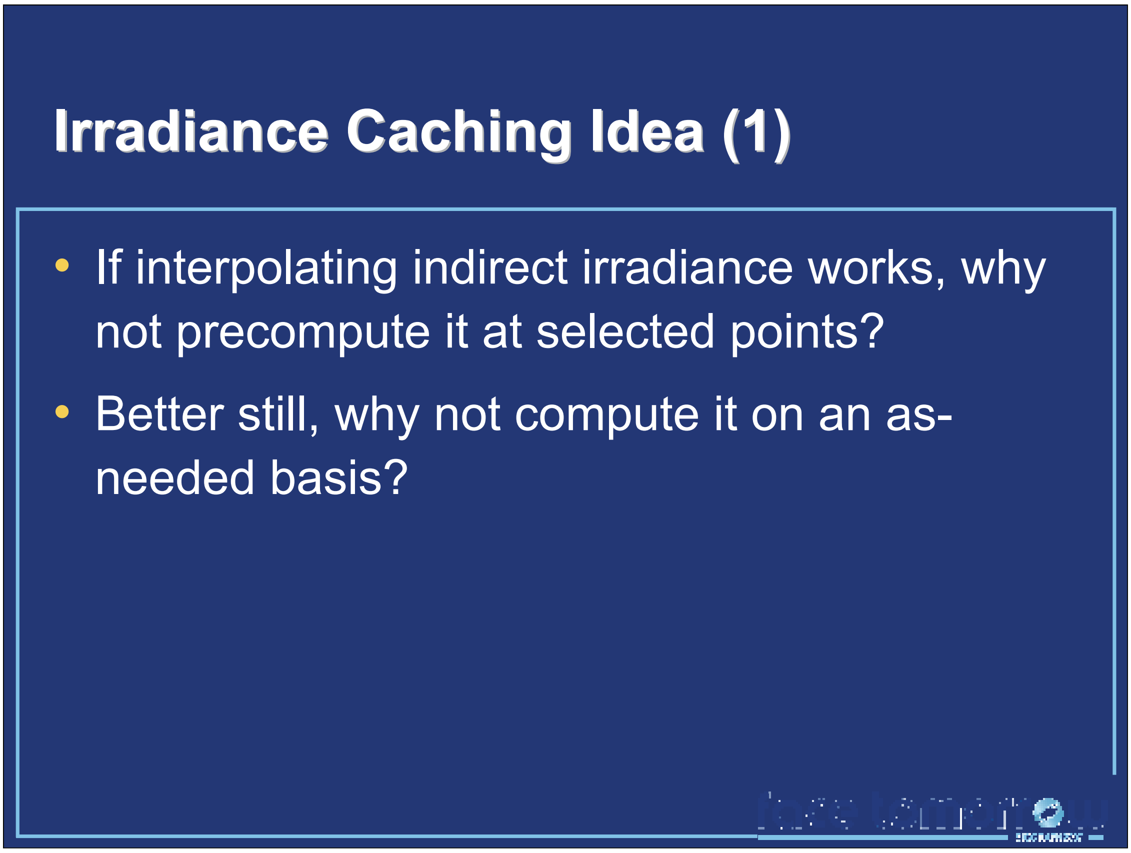
Two-bounce *Radiance* Rendering

Radiance is a physically-based renderer that has been around for over 20 years, and the irradiance cache was in the first public release in 1989.



Indirect Irradiance (two bounces)

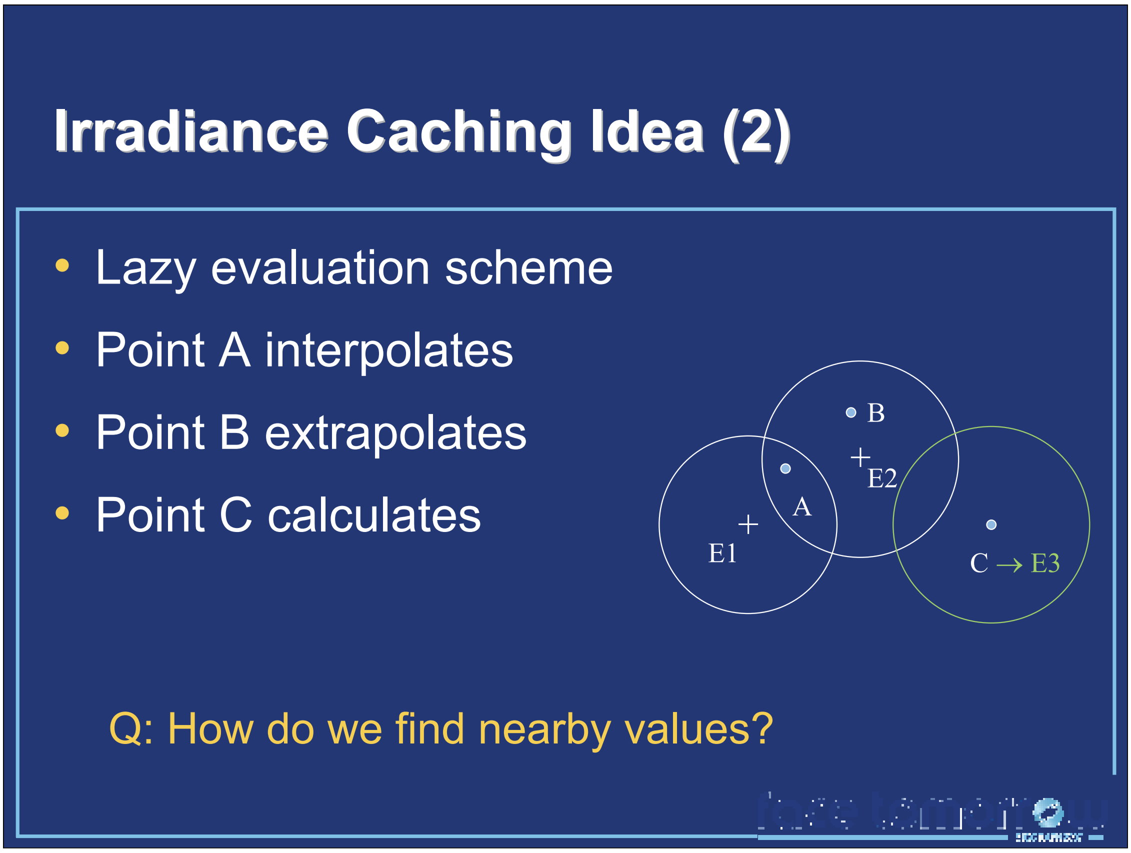
Shown here is the indirect (RGB) irradiance, which changes slowly over flat surface areas, and more rapidly over curved regions. Notice the saturation of red near the chairs, though their final color is not shown. Depicted is only the light arriving at surfaces after one or more bounces.



Irradiance Caching Idea (1)

- If interpolating indirect irradiance works, why not precompute it at selected points?
- Better still, why not compute it on an as-needed basis?

I didn’t coin the term “irradiance caching” -- that was someone else’s good idea. I just called it “lazy evaluation,” being too lazy myself to think up a good name.

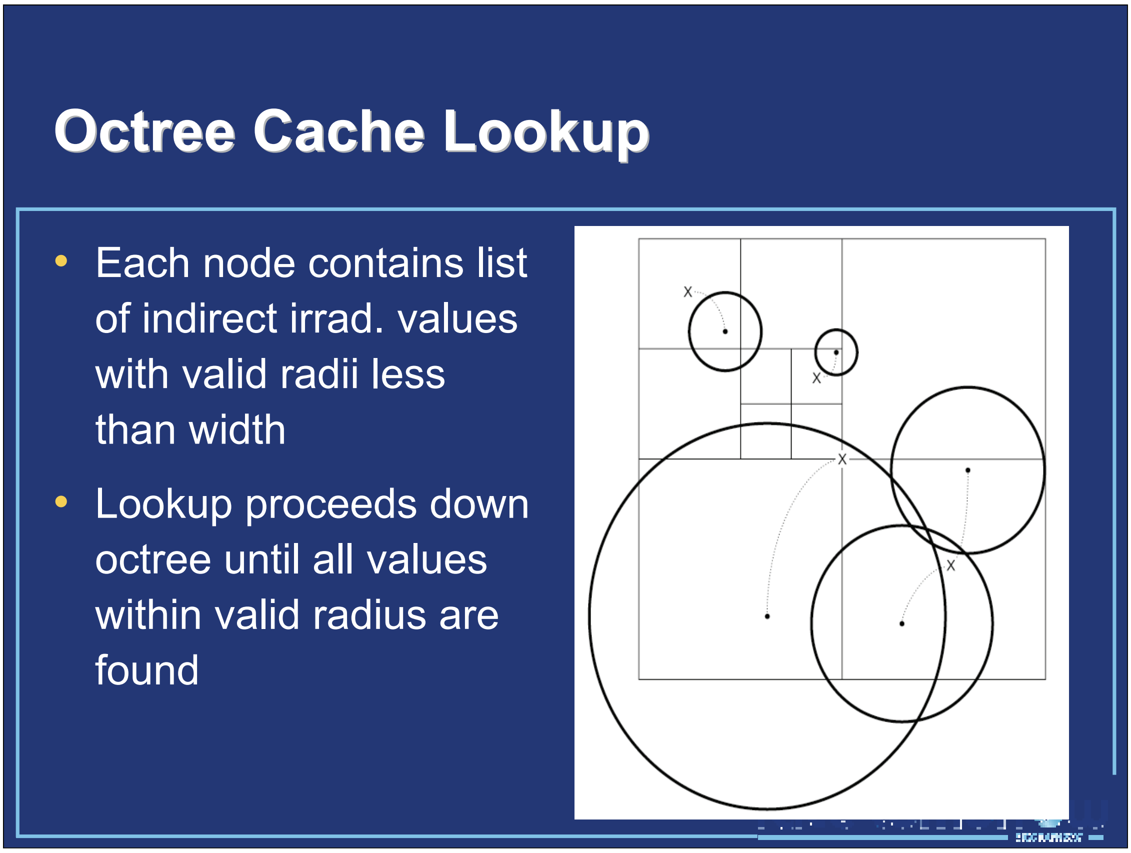


Irradiance Caching Idea (2)

- Lazy evaluation scheme
- Point A interpolates
- Point B extrapolates
- Point C calculates

Q: How do we find nearby values?

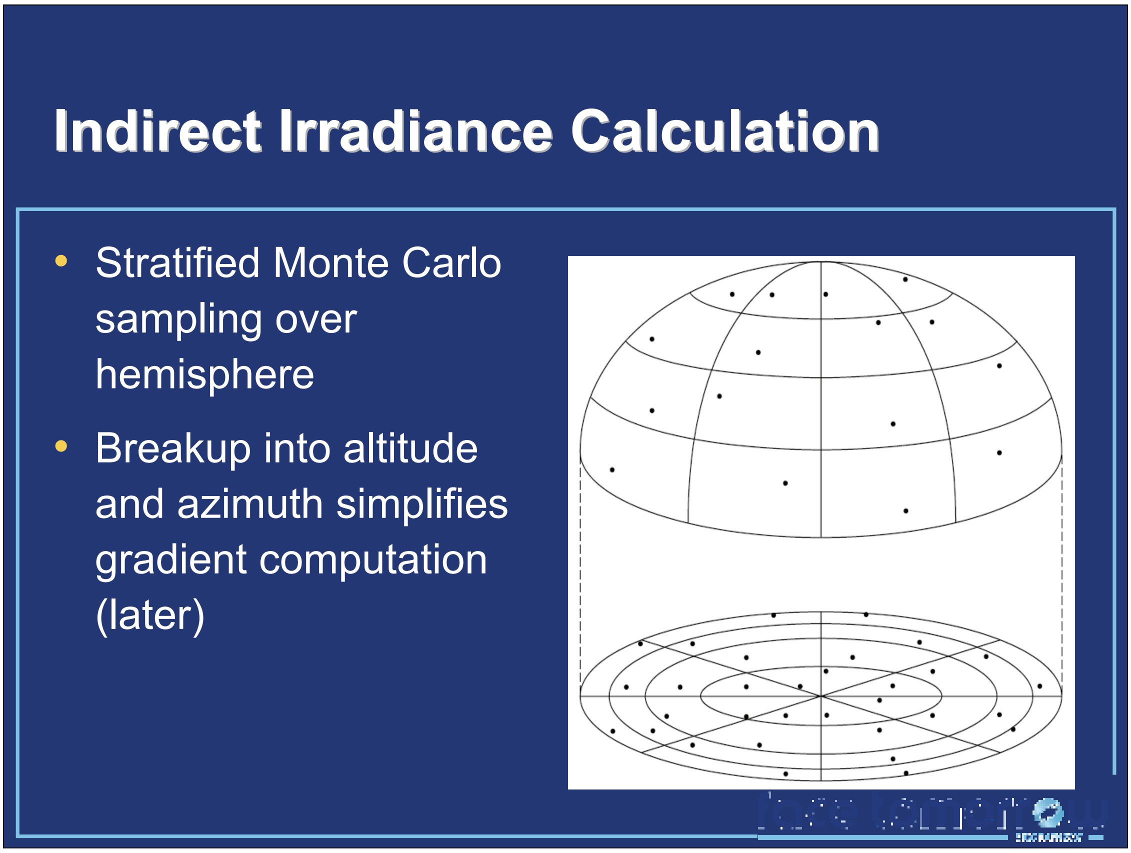
The E’s are evaluation points. E1 and E2 shown here are previous evalulations, where E3 is a new evaluation triggered at query position C.



Octree Cache Lookup

- Each node contains list of indirect irrad. values with valid radii less than width
- Lookup proceeds down octree until all values within valid radius are found

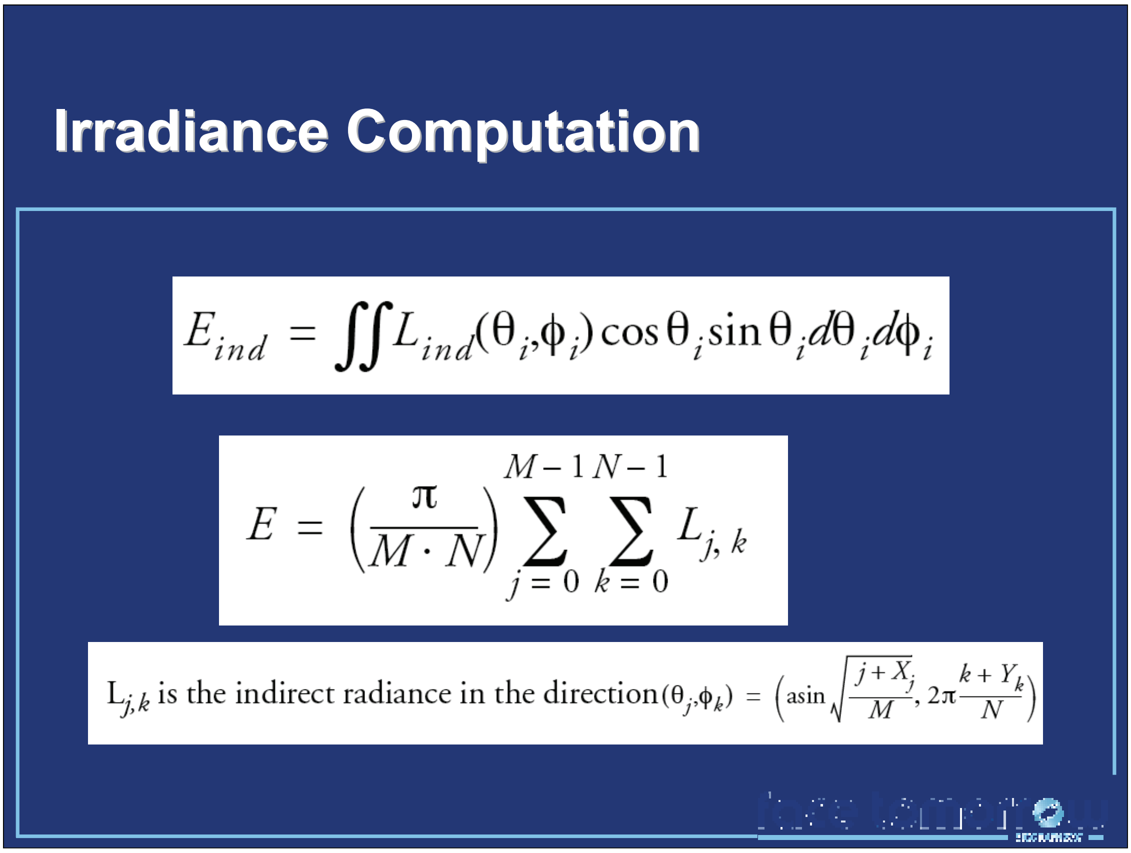
Using an octree that is independent of surfaces avoids placing restrictions on the geometric representation. It even works for volume data, sort of…



Indirect Irradiance Calculation

- Stratified Monte Carlo sampling over hemisphere
- Breakup into altitude and azimuth simplifies gradient computation (later)

Stratified sampling reduces noise in the results at no extra cost, without adding bias. Divisions are placed to maintain constant projected areas on unit circle (Nusselt analog).



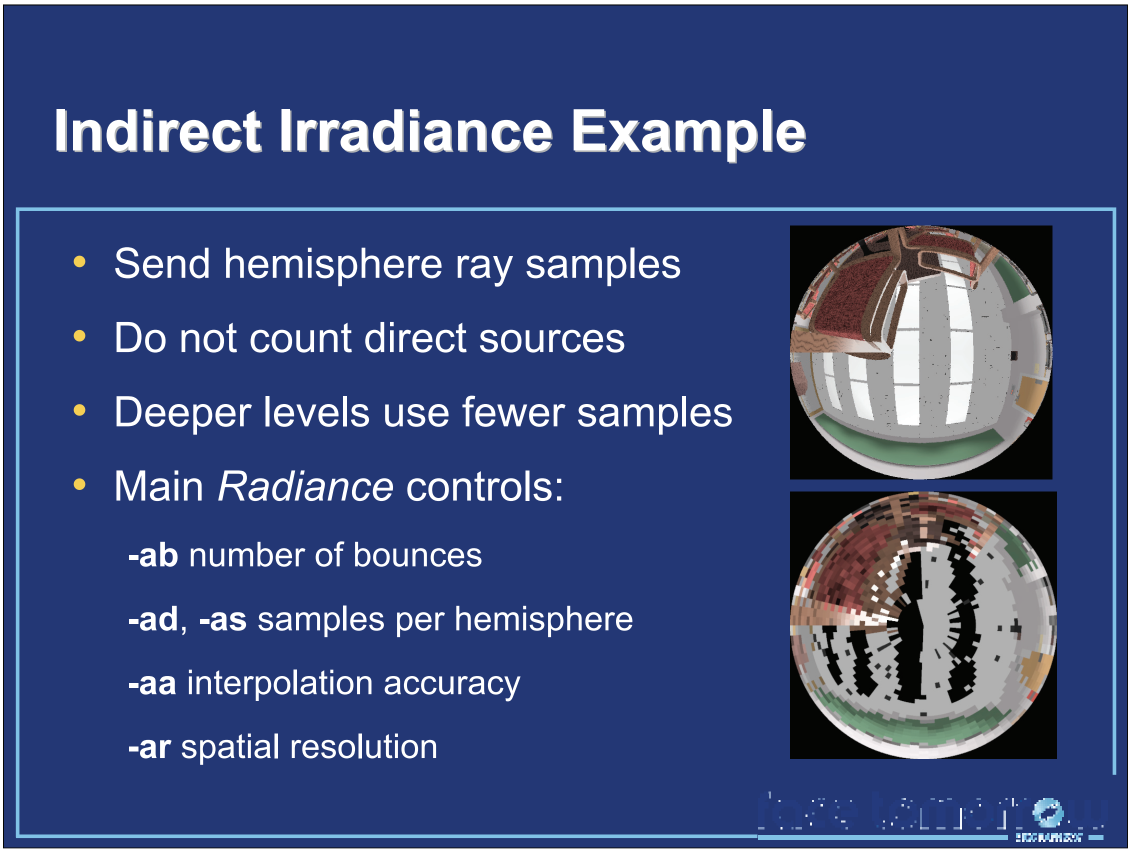
Irradiance Computation

$$E_{ind} = \iint L_{ind}(\theta_i, \phi_i) \cos\theta_i \sin\theta_i d\theta_i d\phi_i$$

$$E = \left(\frac{\pi}{M \cdot N}\right) \sum_{j=0}^{M-1} \sum_{k=0}^{N-1} L_{j,k}$$

$L_{j,k}$ is the indirect radiance in the direction $(\theta_j, \phi_k) = \left(\operatorname{asin}\sqrt{\frac{j+X_j}{M}}, 2\pi\frac{k+Y_k}{N}\right)$

Integral equation may be converted to equal-weighted sum of selected “measures” using standard Monte Carlo integration techniques (inverting cosine projection). X_j and Y_k are uniformly distributed random variables between 0 and 1. To compute indirect irradiance, we count as zero any samples that intersect a light source.



Indirect Irradiance Example

- Send hemisphere ray samples
- Do not count direct sources
- Deeper levels use fewer samples
- Main *Radiance* controls:

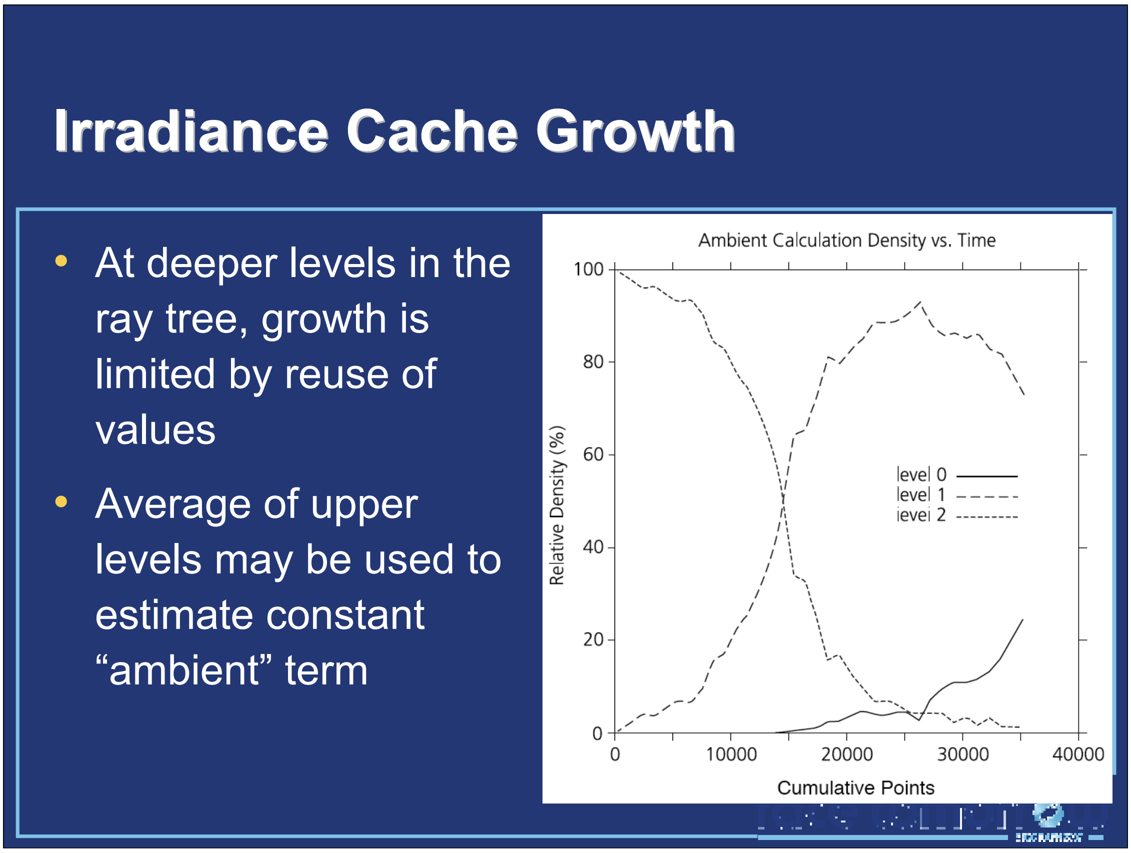
  -ab number of bounces

  -ad, **-as** samples per hemisphere

  -aa interpolation accuracy

  -ar spatial resolution

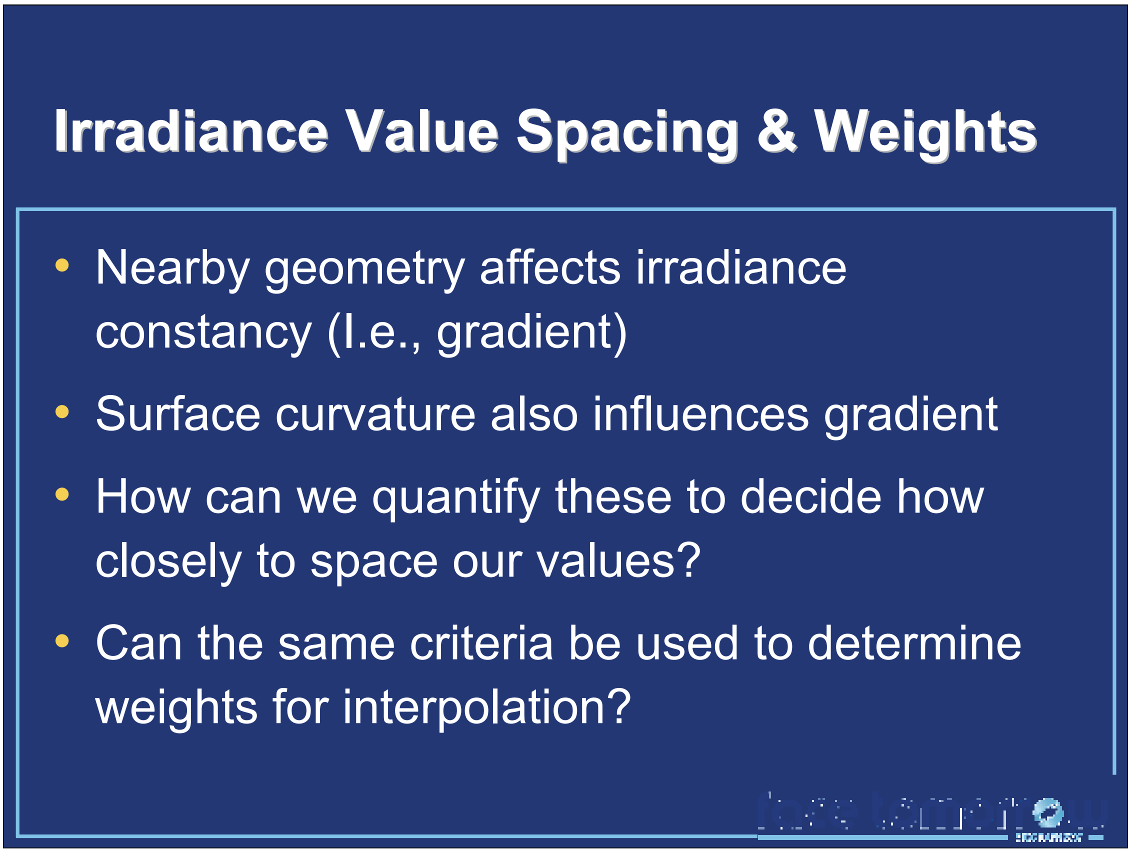
The top image shows a hemispherical equal-contribution projection from the floor near one of the chairs in the conference room model. The bottom image shows the actual shape and number of samples sent for a particular indirect irradiance computation. Note that the sources are black, and the sampling areas follow altitude and azimuth divisions. Actual sample placement within each area is random.



Irradiance Cache Growth

- At deeper levels in the ray tree, growth is limited by reuse of values
- Average of upper levels may be used to estimate constant “ambient” term

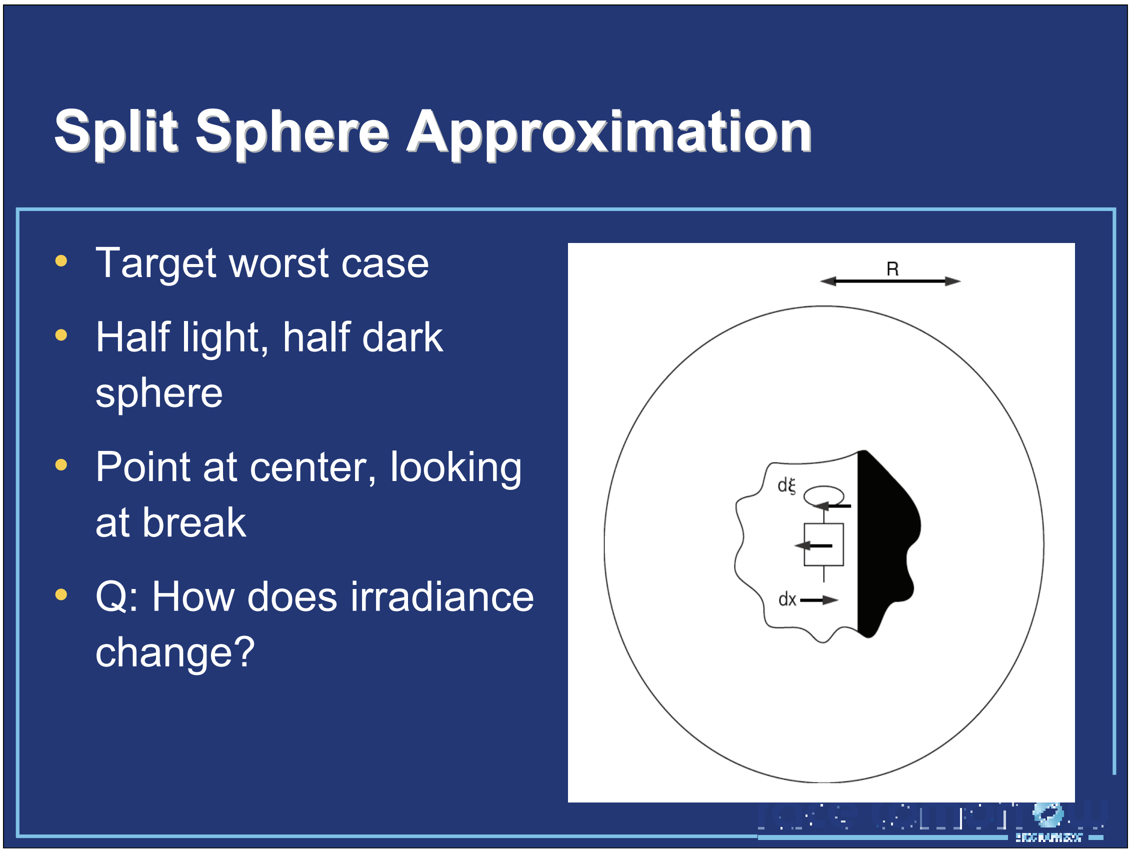
In a multi-bounce calculation, the deepest level is filled first, as the initial sample starts a hemisphere calculation, which begets another at the next level, and so on. The plot shows a three-bounce calculation, which starts by filling in the “level 2” cache, then spends time on the “level 1” cache before finally reaching the “level 0” cache -- the final indirect irradiance.



Irradiance Value Spacing & Weights

- Nearby geometry affects irradiance constancy (I.e., gradient)
- Surface curvature also influences gradient
- How can we quantify these to decide how closely to space our values?
- Can the same criteria be used to determine weights for interpolation?

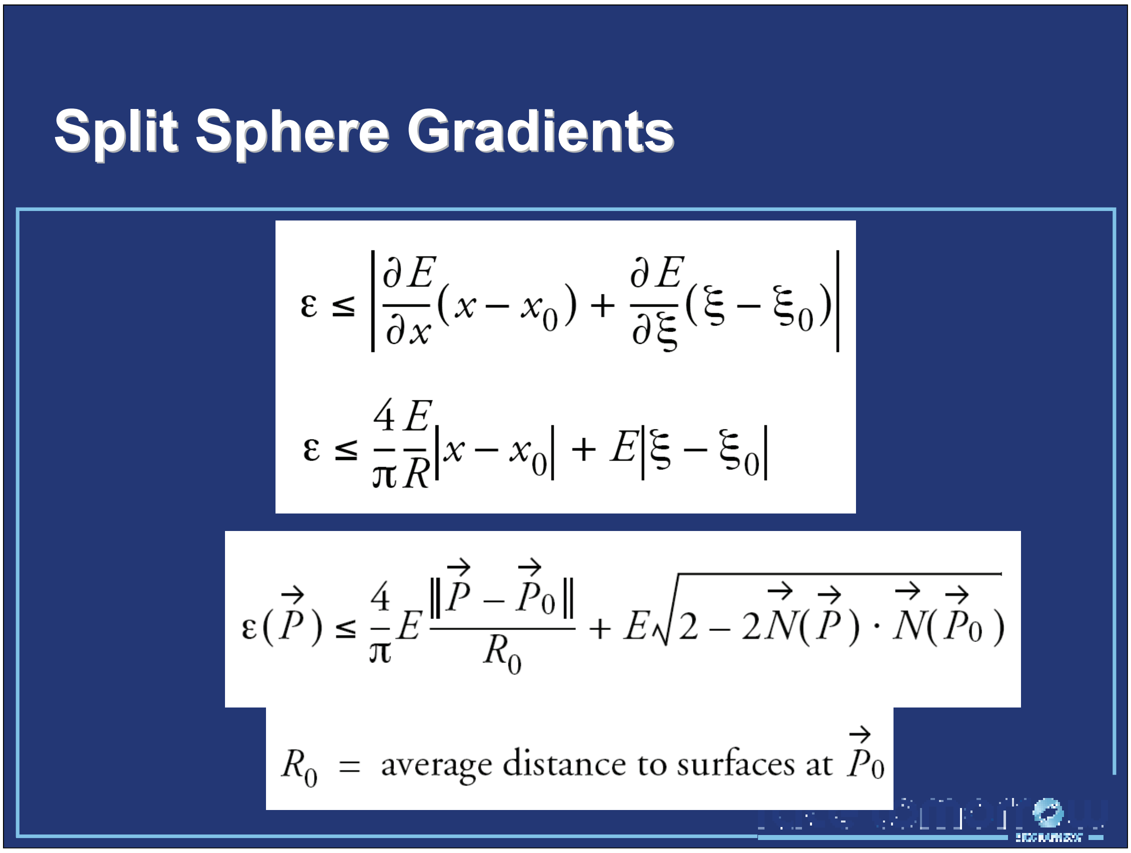
We have an intuition about the behavior of indirect irradiance -- we need to translate this into an algorithm that is robust and reliable.



Split Sphere Approximation

- Target worst case
- Half light, half dark sphere
- Point at center, looking at break
- Q: How does irradiance change?

The “split sphere” is not technically a “worst case” -- it’s just a “pretty bad case.” It is better to qualify this as an assumption. I.e., we assume our actual environment will behave no worse than the split sphere. If it does, our technique won’t necessarily break, but it won’t be as accurate as we hoped, either.



Split Sphere Gradients

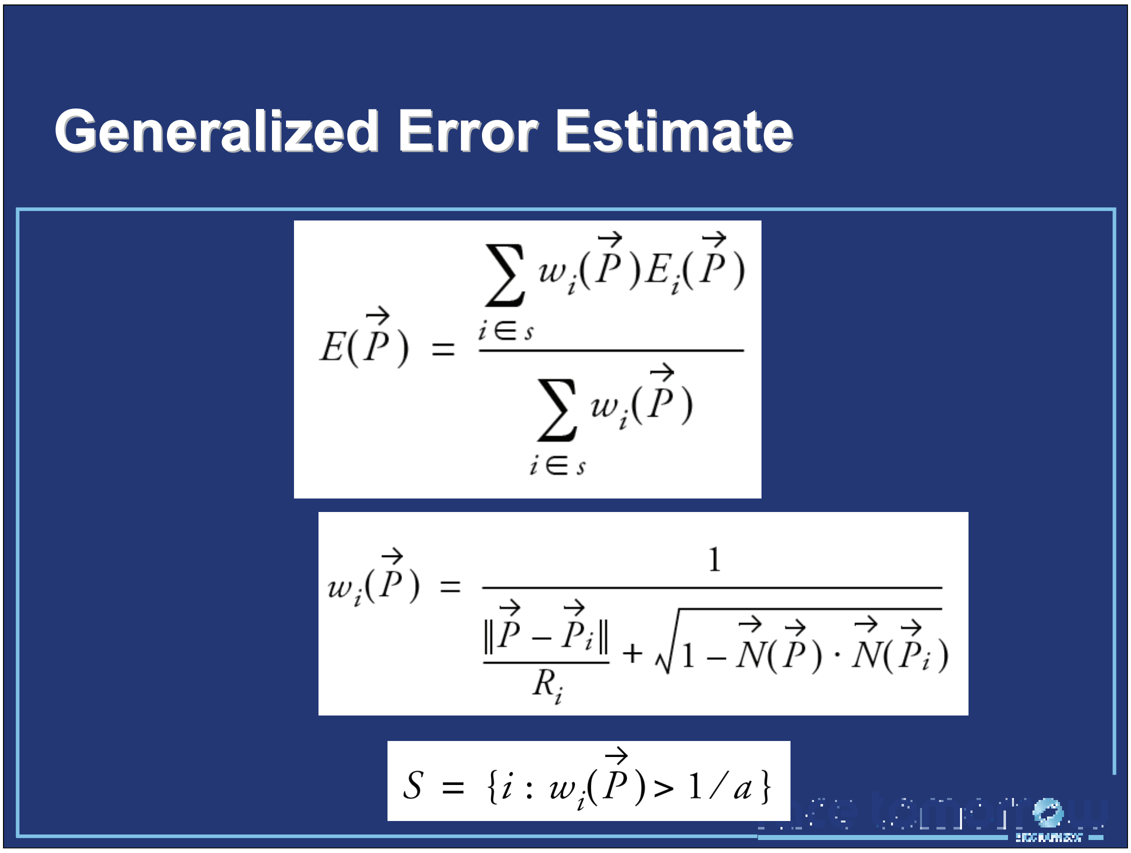
$$\varepsilon \le \left| \frac{\partial E}{\partial x}(x - x_0) + \frac{\partial E}{\partial \xi}(\xi - \xi_0) \right|$$

$$\varepsilon \le \frac{4}{\pi}\frac{E}{R}|x - x_0| + E|\xi - \xi_0|$$

$$\varepsilon(\vec{P}) \le \frac{4}{\pi}E\frac{\|\vec{P} - \vec{P}_0\|}{R_0} + E\sqrt{2 - 2\vec{N}(\vec{P}) \cdot \vec{N}(\vec{P}_0)}$$

$$R_0 = \text{average distance to surfaces at } \vec{P}_0$$

The first two inequalities bound the split sphere partial derivatives. The final equation uses these bounds to derive a “pretty bad case” error estimate for movement and rotation based on this first-order analysis. The calculation of R0 uses a harmonic mean to neighboring surfaces as measured by our ray samples. A harmonic mean is preferred since the radius appears in the denominator of our equation. Obviously, zero ray lengths are forbidden.



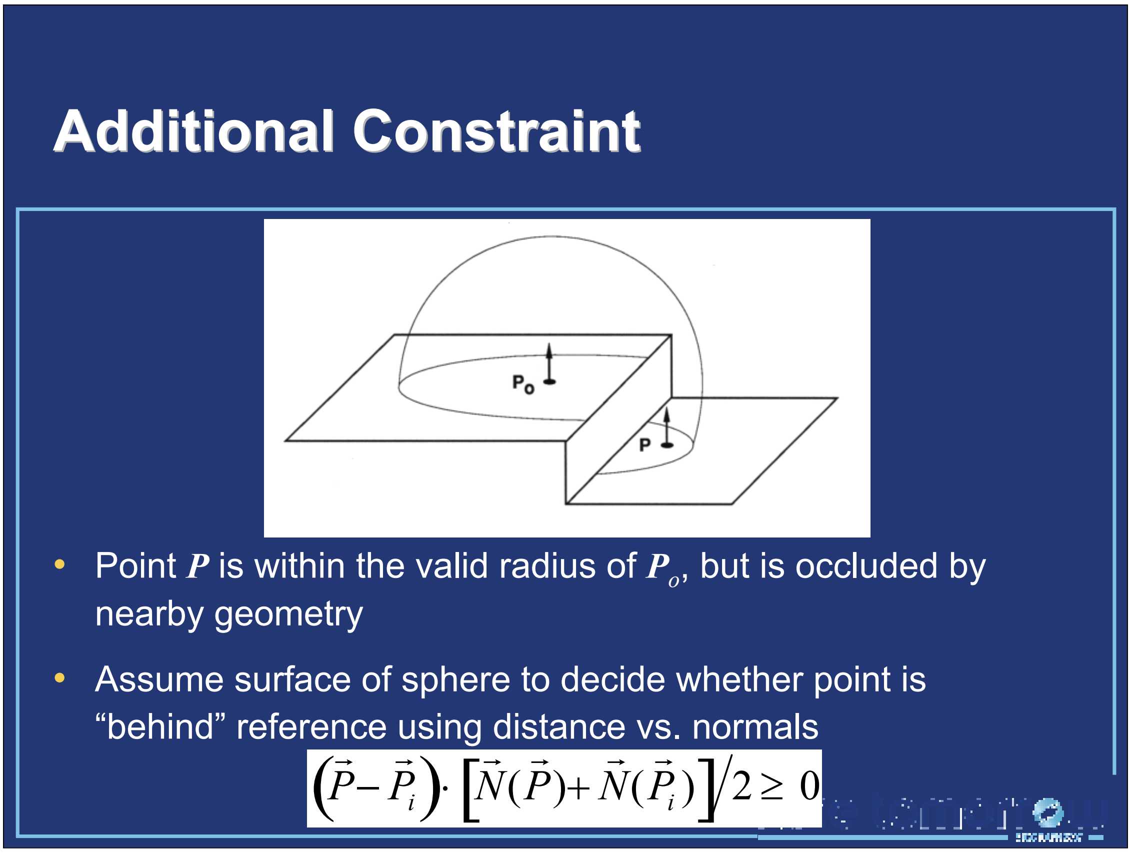
Generalized Error Estimate

$$E(\vec{P}) = \frac{\sum_{i \in s} w_i(\vec{P}) E_i(\vec{P})}{\sum_{i \in s} w_i(\vec{P})}$$

$$w_i(\vec{P}) = \frac{1}{\frac{\|\vec{P} - \vec{P}_i\|}{R_i} + \sqrt{1 - \vec{N}(\vec{P}) \cdot \vec{N}(\vec{P}_i)}}$$

$$S = \{i : w_i(\vec{P}) > 1/a\}$$

Ignoring the constants from our previous equation, we can derive a weighting function based on the split sphere approximation. Since our weights correlate to 1/error, applying them in a weighted average distributes error uniformly between the interpolated irradiance values. We hope to maintain final accuracy by restricting our set of values to ones whose estimated error is below some user tolerance, “a”.

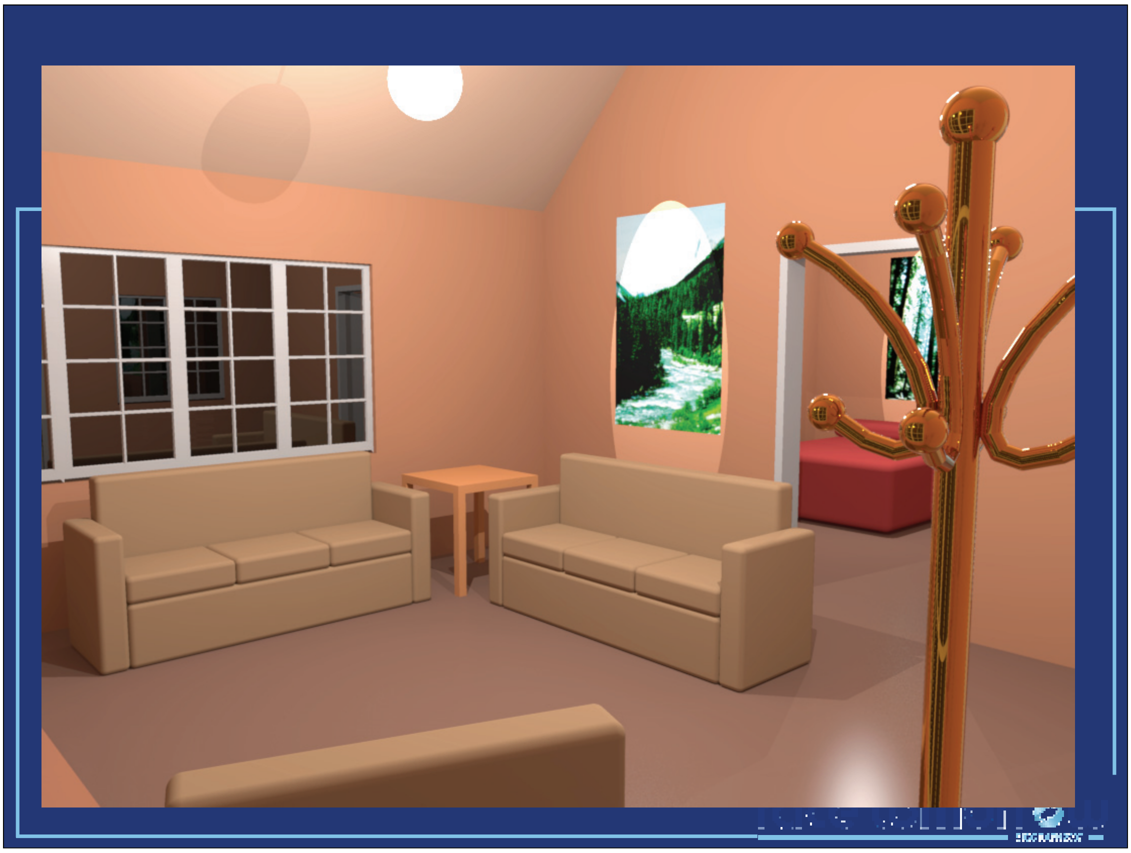


Additional Constraint

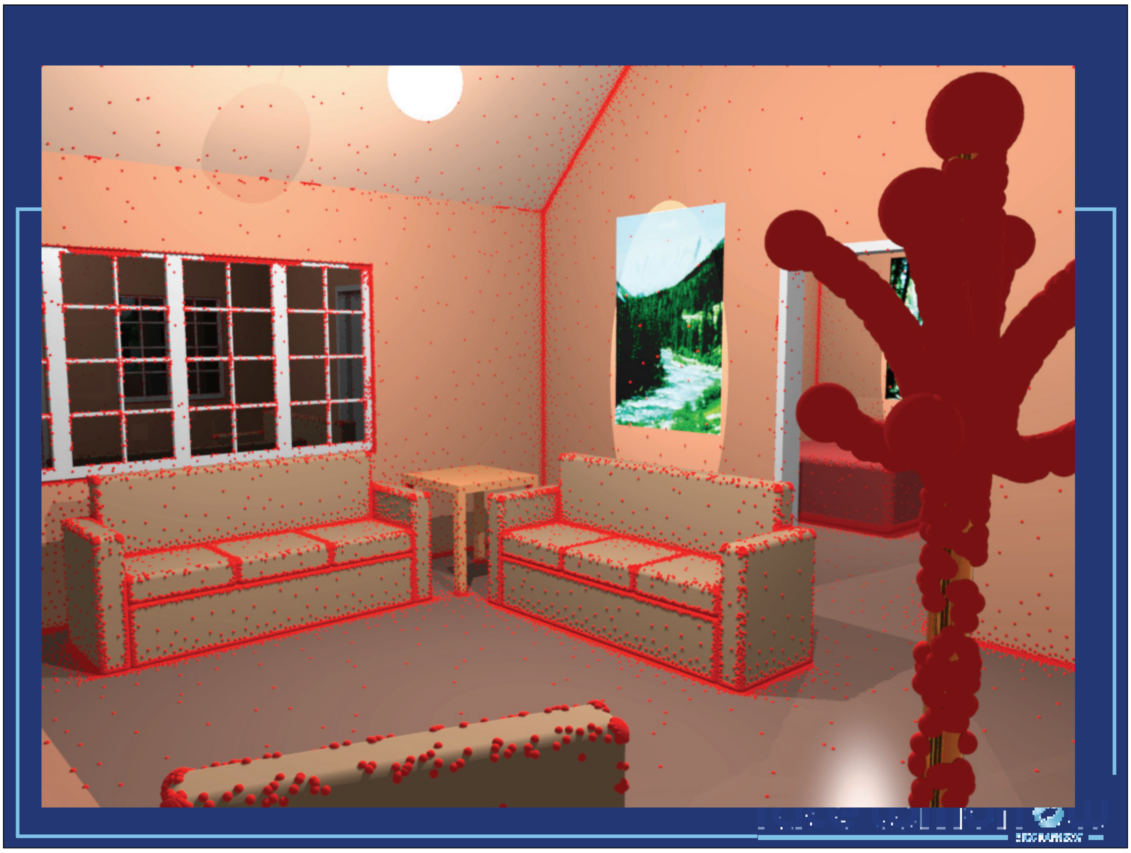
- Point P is within the valid radius of P_o, but is occluded by nearby geometry
- Assume surface of sphere to decide whether point is “behind” reference using distance vs. normals

$$\left(\vec{P}-\vec{P}_i\right)\cdot\left[\vec{N}(\vec{P})+\vec{N}(\vec{P}_i)\right]\Big/2 \geq 0$$

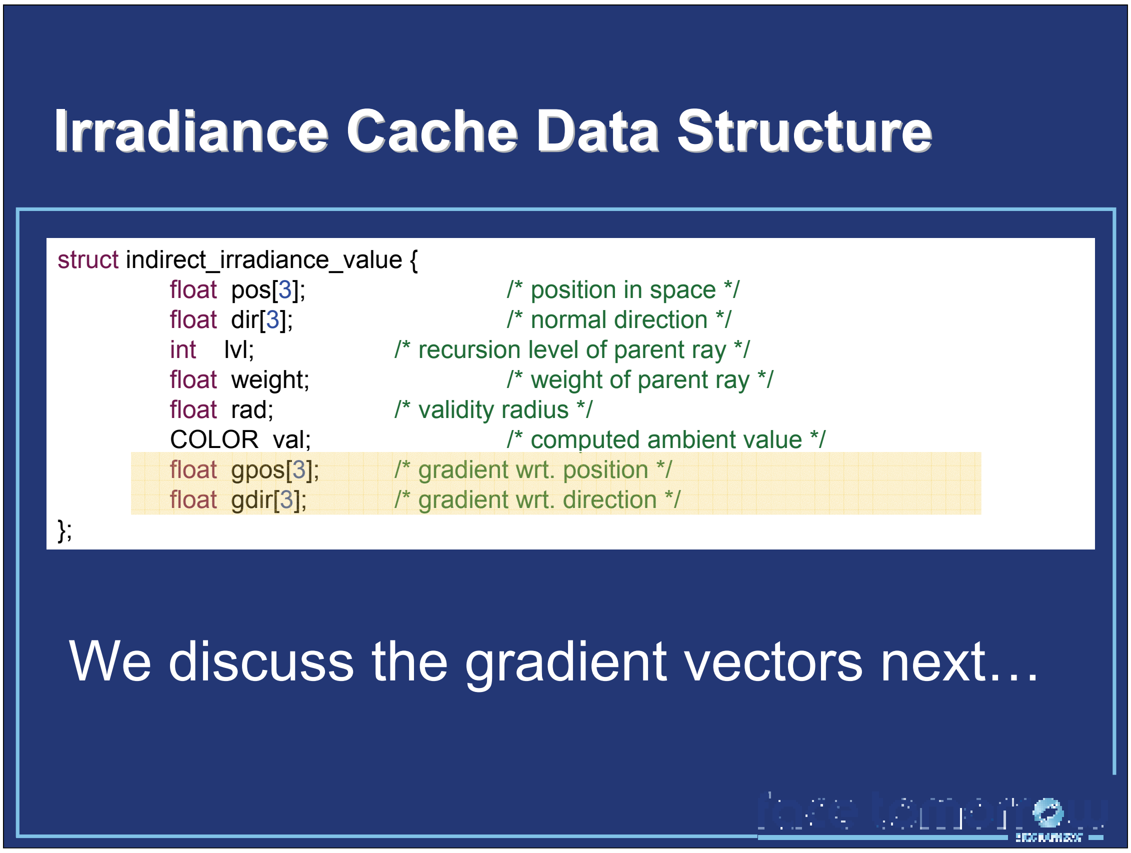
One special case we need to consider occurs when an irradiance value *thinks* it is valid over a larger region than it actually is. This case is depicted in the figure, where we are thinking about using an irradiance value at P0 at the new query position P. We can avoid this problem condition by adding a “behind test” to our criteria, which via the weighting function already incorporates a curvature test and a relative distance test.



Cabin model showing single-bounce calculation.



Indirect irradiance record placement shown as red spheres. Note how record spacing is large on flat surfaces far from neighboring geometry, then bunches up on outside curves and (especially) inside corners.

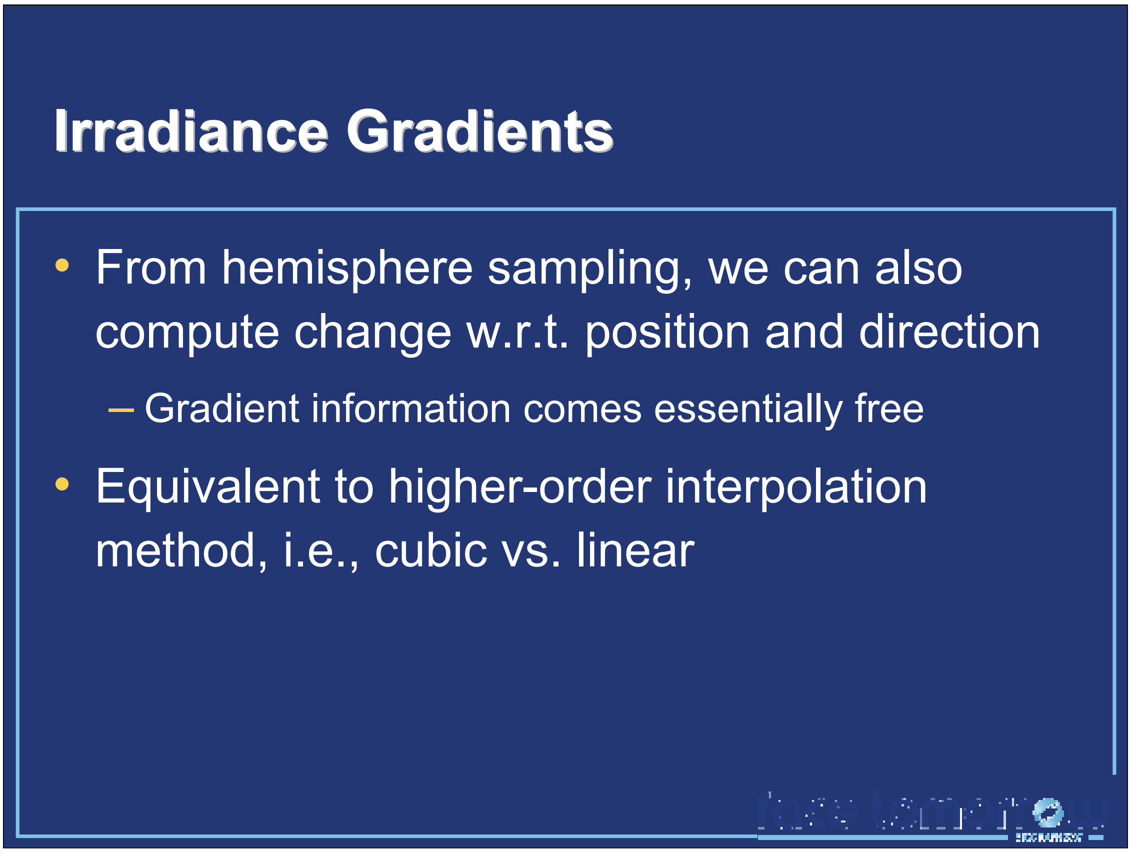


Irradiance Cache Data Structure

```
struct indirect_irradiance_value {
          float  pos[3];                 /* position in space */
          float  dir[3];                 /* normal direction */
          int    lvl;           /* recursion level of parent ray */
          float  weight;                 /* weight of parent ray */
          float  rad;           /* validity radius */
          COLOR  val;                    /* computed ambient value */
          float  gpos[3];       /* gradient wrt. position */
          float  gdir[3];       /* gradient wrt. direction */
};
```

We discuss the gradient vectors next…

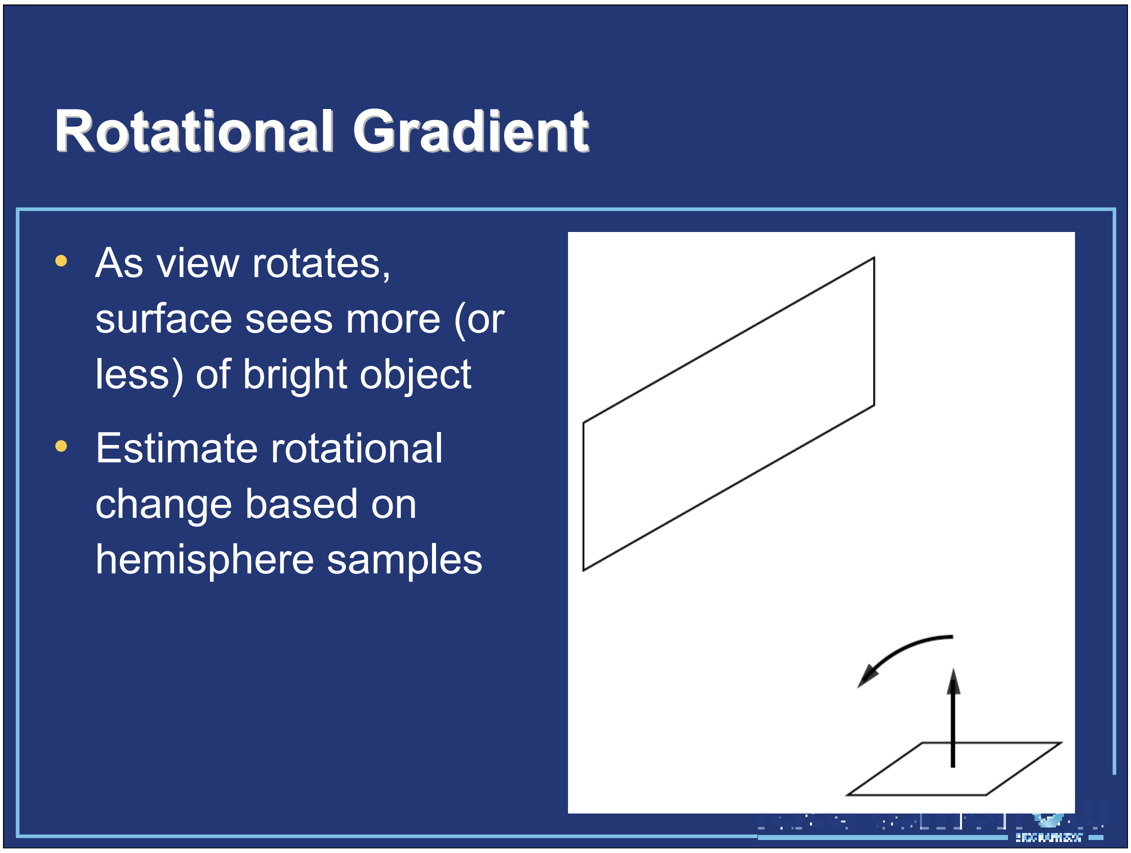
The ray level and weight are used to determine when to truncate the ray tree. The position and normal vectors are used in the weight/threshold computation, together with the “validity radius.”



Irradiance Gradients

- From hemisphere sampling, we can also compute change w.r.t. position and direction
  - Gradient information comes essentially free
- Equivalent to higher-order interpolation method, i.e., cubic vs. linear

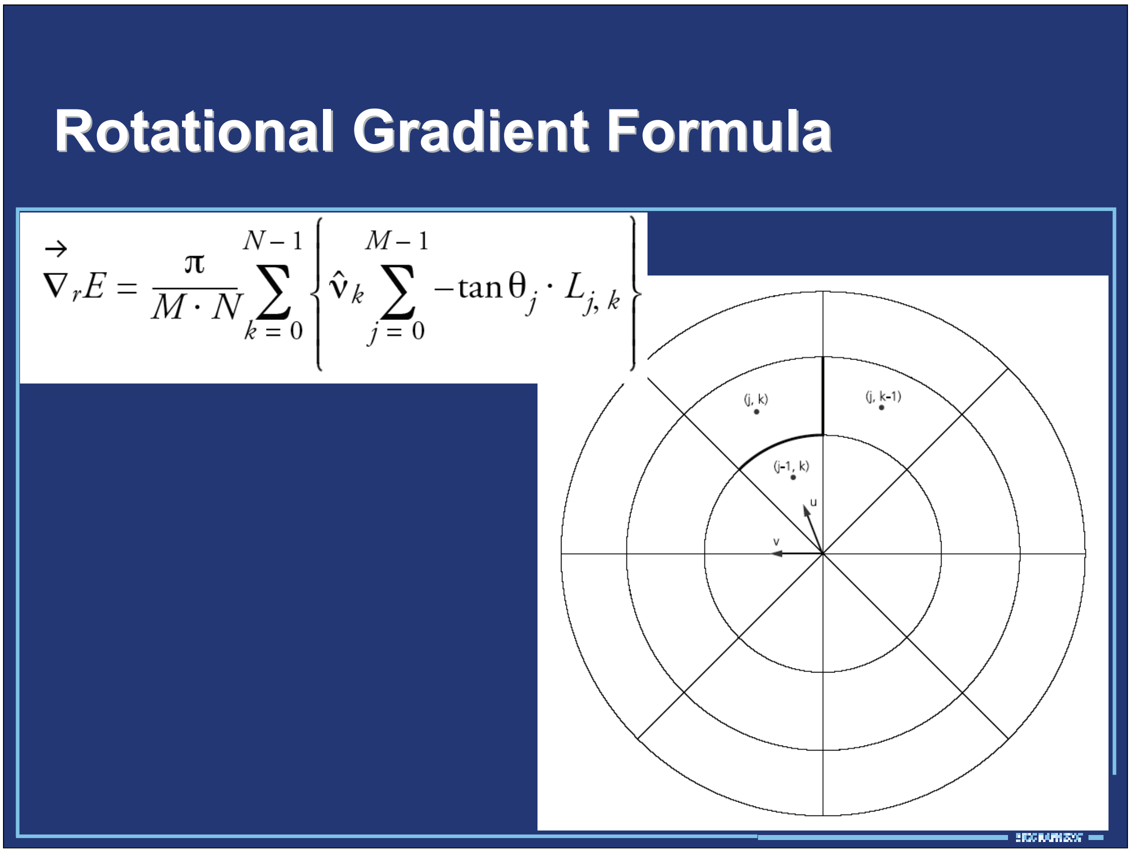
Initially, Paul Heckbert and I were thinking a gradient calculation could inform better value spacing. A year went by before we realized that it was better to apply them directly during interpolation to reduce discontinuities. This also avoids bias issues.



Rotational Gradient

- As view rotates, surface sees more (or less) of bright object
- Estimate rotational change based on hemisphere samples

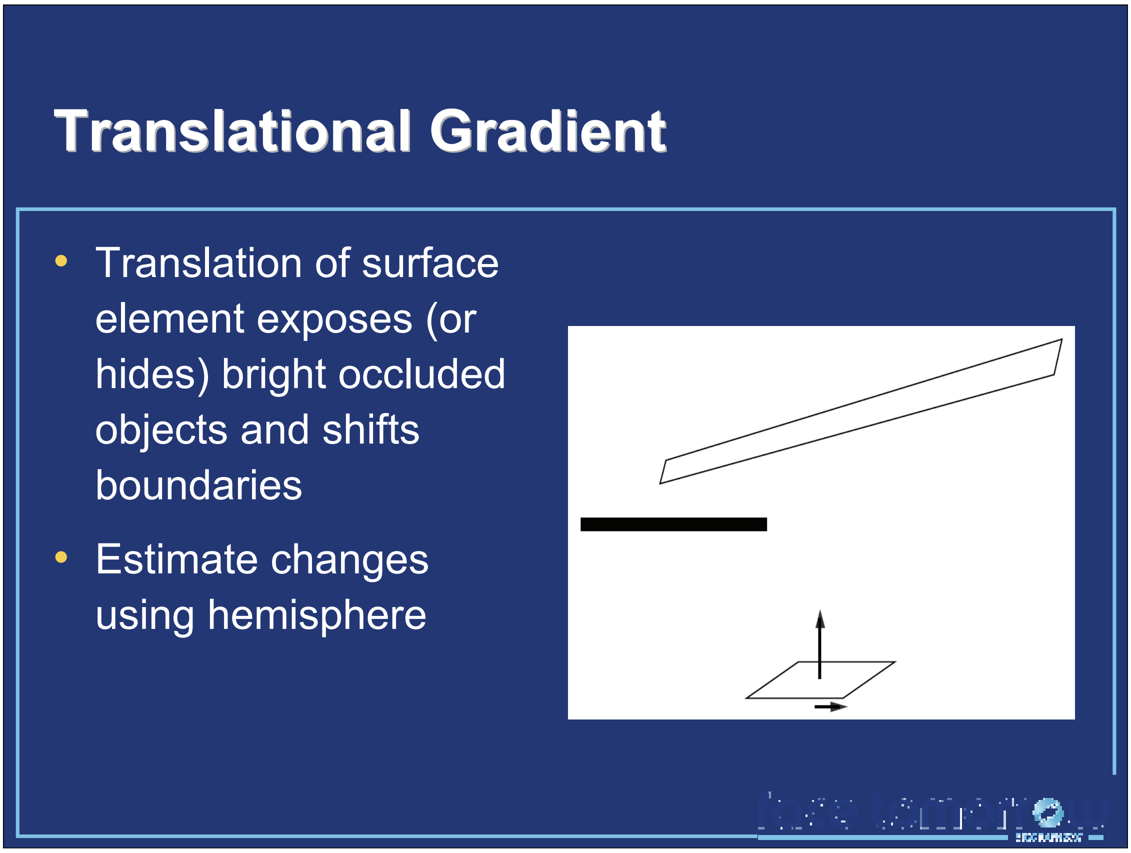
Caveat: we cannot know what will appear over the horizon.



Rotational Gradient Formula

$$\vec{\nabla}_r E = \frac{\pi}{M \cdot N} \sum_{k=0}^{N-1} \left\{ \hat{v}_k \sum_{j=0}^{M-1} -\tan\theta_j \cdot L_{j,k} \right\}$$

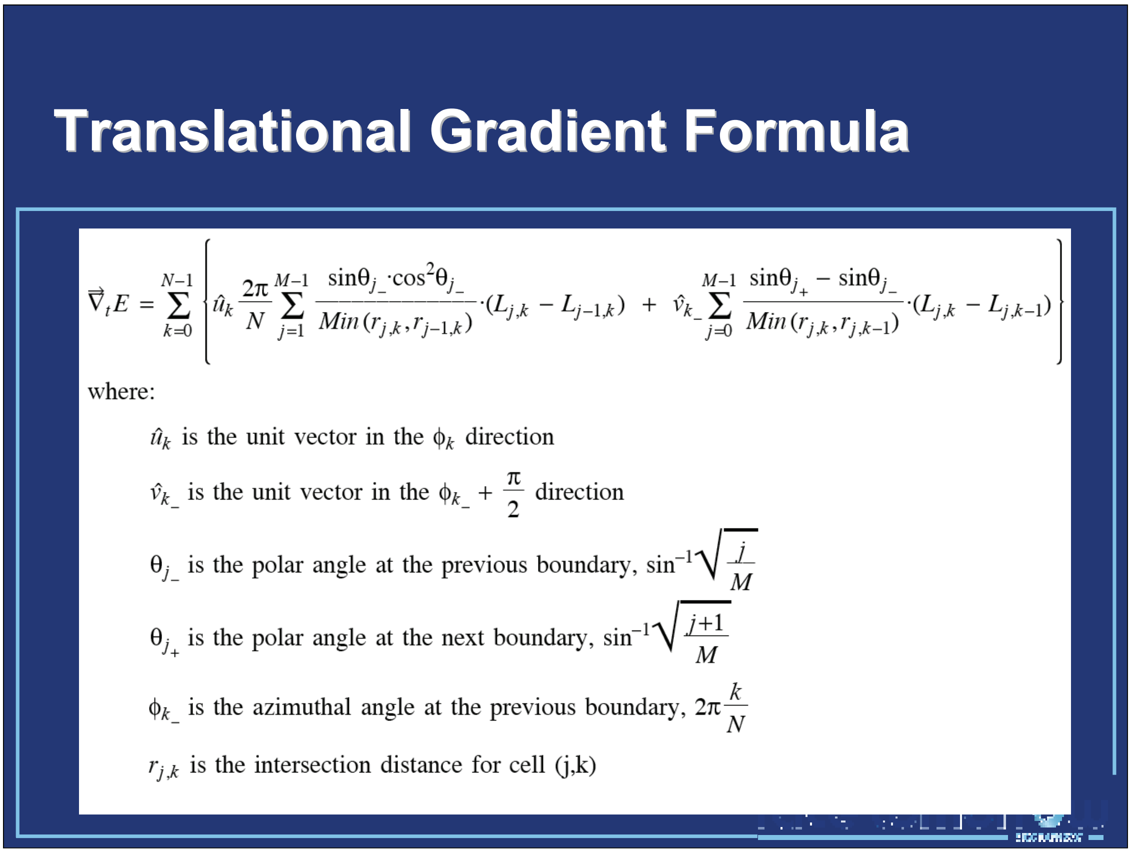
The rotational gradient is zero at the zenith because the derivative of the cosine is zero, hence small rotations don’t affect contribution for samples looking straight up.



Translational Gradient

- Translation of surface element exposes (or hides) bright occluded objects and shifts boundaries
- Estimate changes using hemisphere

The distance to our sampled geometry is important, and we have to assume that disocclusions look like what we already see of the background object. Hence, the boundary moves as the nearer edge moves rather than the further surface.



Translational Gradient Formula

$$\vec{\nabla}_t E = \sum_{k=0}^{N-1} \left\{ \hat{u}_k \frac{2\pi}{N} \sum_{j=1}^{M-1} \frac{\sin\theta_{j\_} \cdot \cos^2\theta_{j\_}}{Min\,(r_{j,k}, r_{j-1,k})} \cdot (L_{j,k} - L_{j-1,k}) \;+\; \hat{v}_{k\_} \sum_{j=0}^{M-1} \frac{\sin\theta_{j_+} - \sin\theta_{j\_}}{Min\,(r_{j,k}, r_{j,k-1})} \cdot (L_{j,k} - L_{j,k-1}) \right\}$$

where:

$\hat{u}_k$ is the unit vector in the ϕ_k direction

$\hat{v}_{k\_}$ is the unit vector in the $\phi_{k\_} + \frac{\pi}{2}$ direction

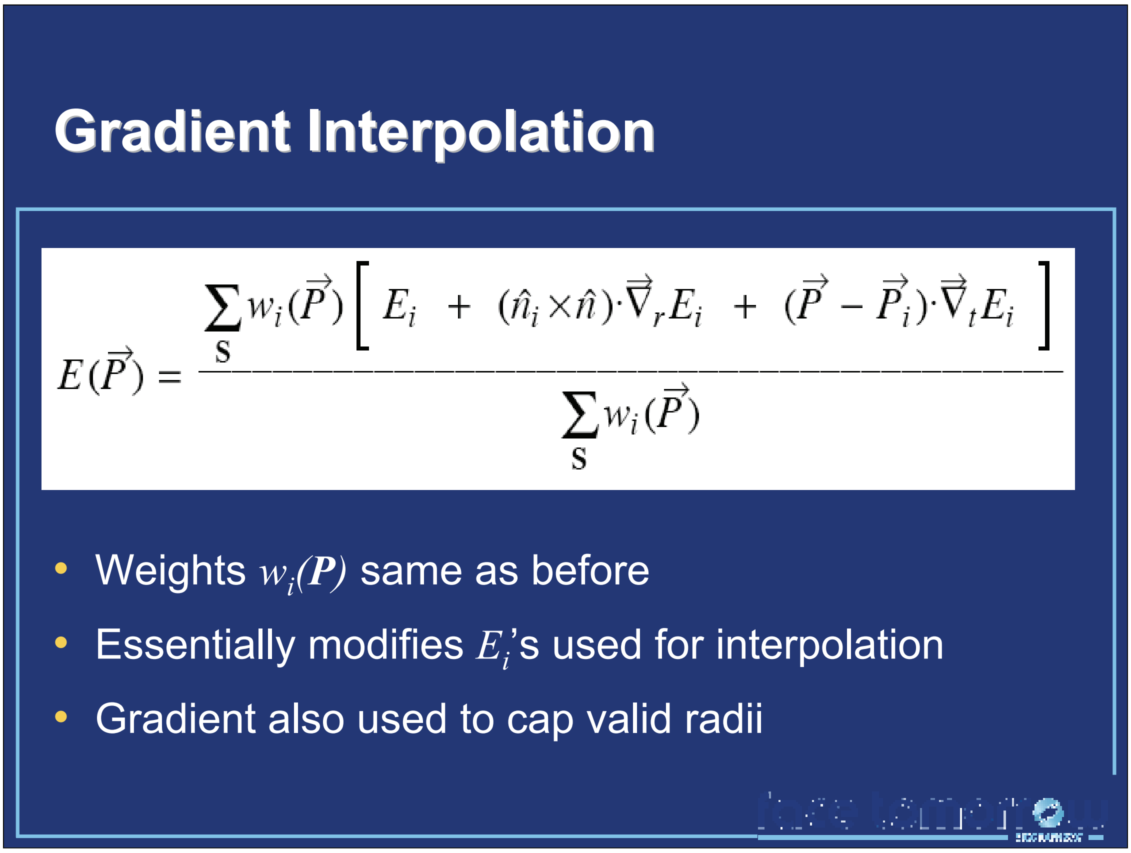
$\theta_{j\_}$ is the polar angle at the previous boundary, $\sin^{-1}\sqrt{\frac{j}{M}}$

θ_{j_+} is the polar angle at the next boundary, $\sin^{-1}\sqrt{\frac{j+1}{M}}$

$\phi_{k\_}$ is the azimuthal angle at the previous boundary, $2\pi\frac{k}{N}$

$r_{j,k}$ is the intersection distance for cell (j,k)

Refer to the vector diagram on the rotational gradient slide for variable definitions. Here, the '+' and '-' suffixes refer to the vectors (not drawn) corresponding to the leading and trailing edges of each cell, respectively. The Min(a,b) function is used to determine the closer of two neighboring surface samples.

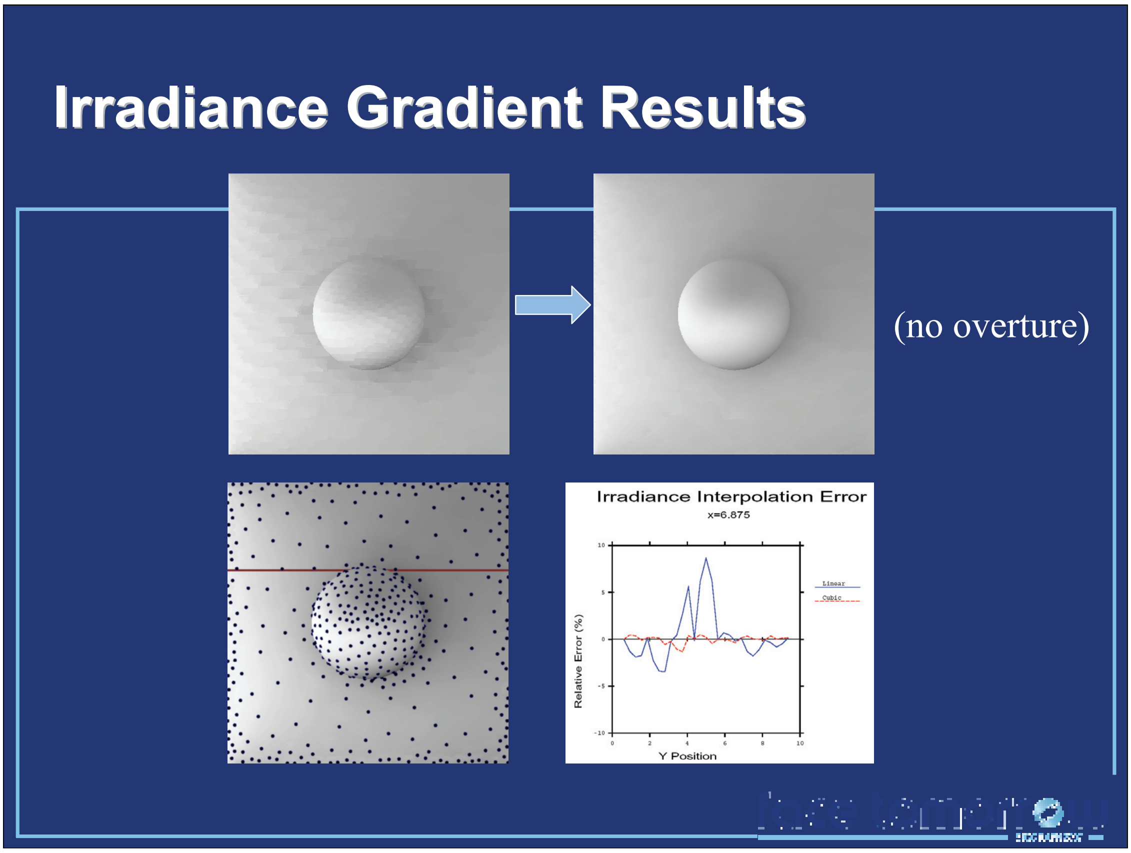


Gradient Interpolation

$$E(\vec{P}) = \frac{\sum_{S} w_i(\vec{P}) \left[E_i \ + \ (\hat{n}_i \times \hat{n}) \cdot \vec{\nabla}_r E_i \ + \ (\vec{P} - \vec{P}_i) \cdot \vec{\nabla}_t E_i \right]}{\sum_{S} w_i(\vec{P})}$$

- Weights $w_i(\boldsymbol{P})$ same as before
- Essentially modifies E_i's used for interpolation
- Gradient also used to cap valid radii

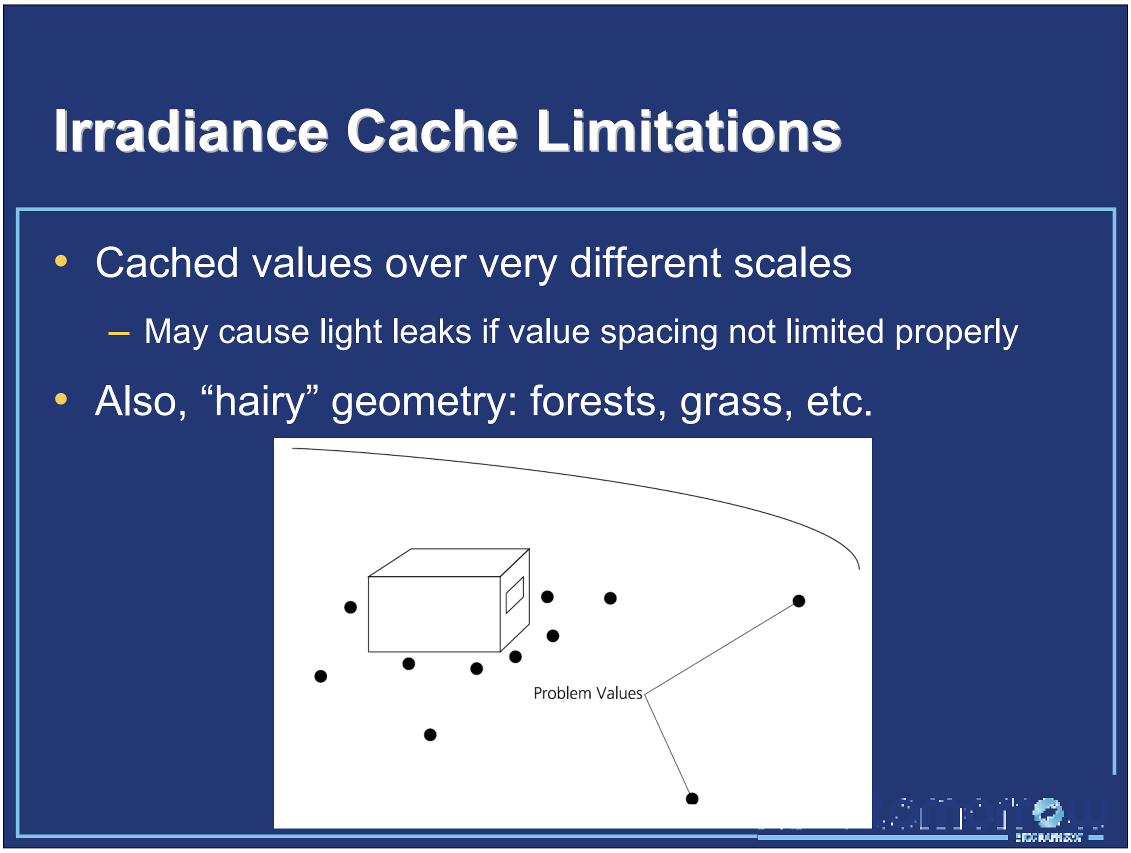
Once we have our rotational and translational gradients corresponding to each irradiance value, we can apply them in a first-order, weighted interpolation as shown.



Irradiance Gradient Results

(no overture)

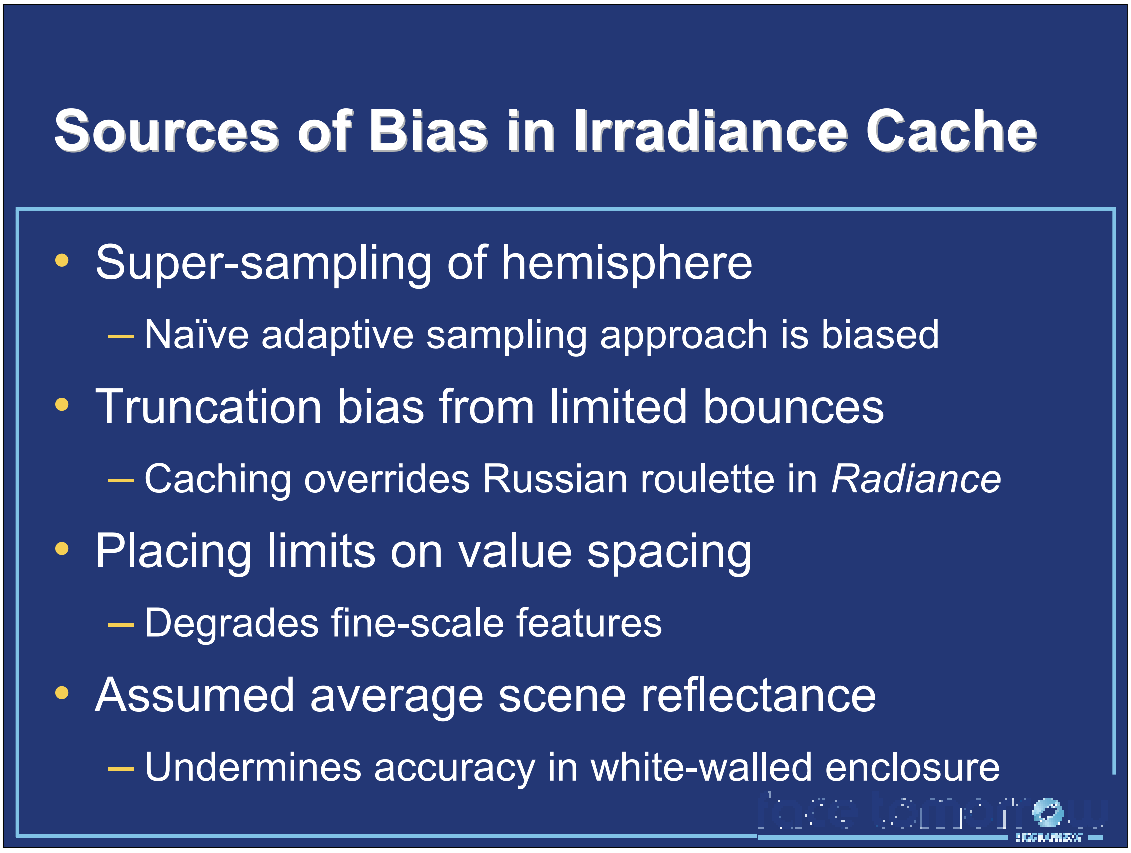
The effect of applying the irradiance gradient technique is highlighted in this extrapolated calculation, where lazy evaluation is shown in its “full glory.” Despite discontinuities introduced by adding values to the cache during scanline rendering, our first-order extrapolation makes these artifacts all but disappear. Value placement is shown together with the comparative error when applying gradients to interpolation (I.e., adding an overture calculation).



Irradiance Cache Limitations

- Cached values over very different scales
  - May cause light leaks if value spacing not limited properly
- Also, “hairy” geometry: forests, grass, etc.

Jaroslav has a solution called “neighbor clamping” for this problem. Hairy geometry is best dealt with by switching to a noisy MCPT method for busy topologies.

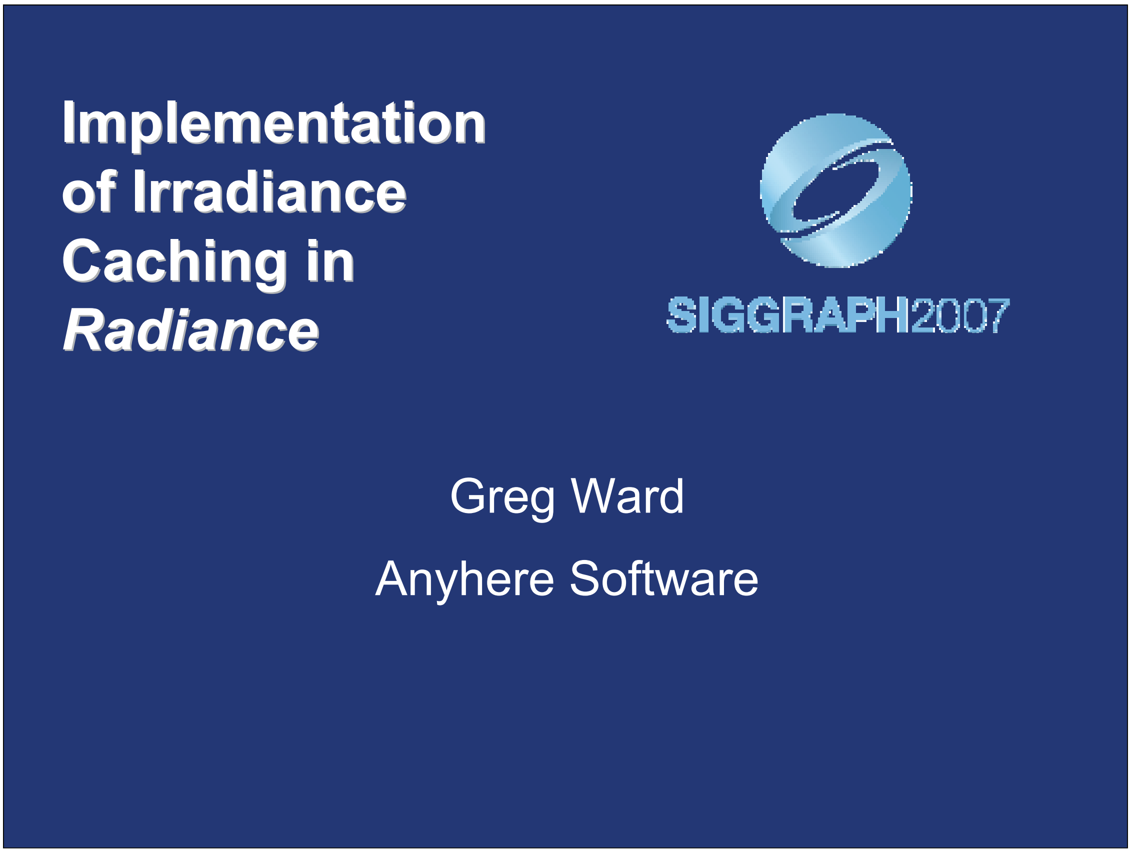


Sources of Bias in Irradiance Cache

- Super-sampling of hemisphere
  - Naïve adaptive sampling approach is biased
- Truncation bias from limited bounces
  - Caching overrides Russian roulette in *Radiance*
- Placing limits on value spacing
  - Degrades fine-scale features
- Assumed average scene reflectance
  - Undermines accuracy in white-walled enclosure

To eliminate bias, don’t use adaptive super-sampling, add Russian roulette to final bounce via path tracing, eliminate minimum value spacing (expensive) and use actual surface reflectances (undermines sharing). Decreasing number of hemisphere samples (by reflectance) and increasing sample spacing (by reflectance^-0.5) distributes errors evenly between levels and speeds convergence.

SIGGRAPH2007

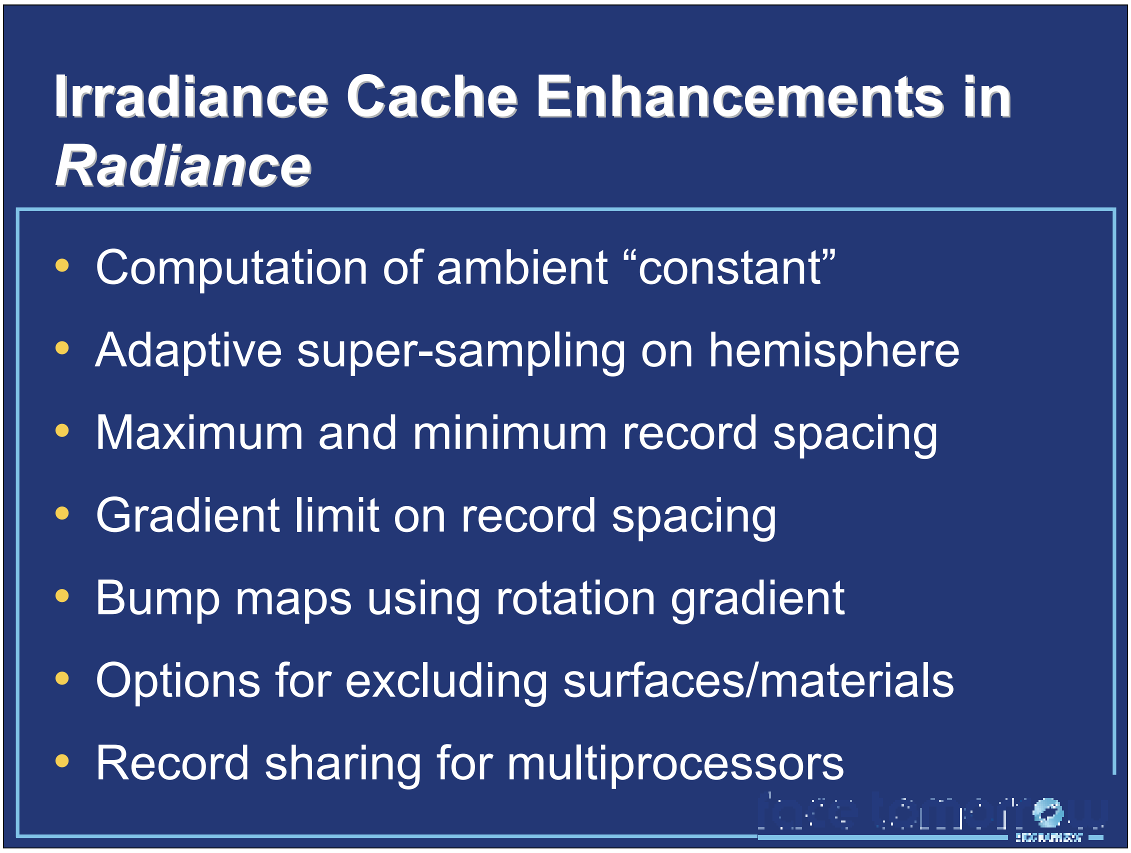


Implementation of Irradiance Caching in *Radiance*

SIGGRAPH2007

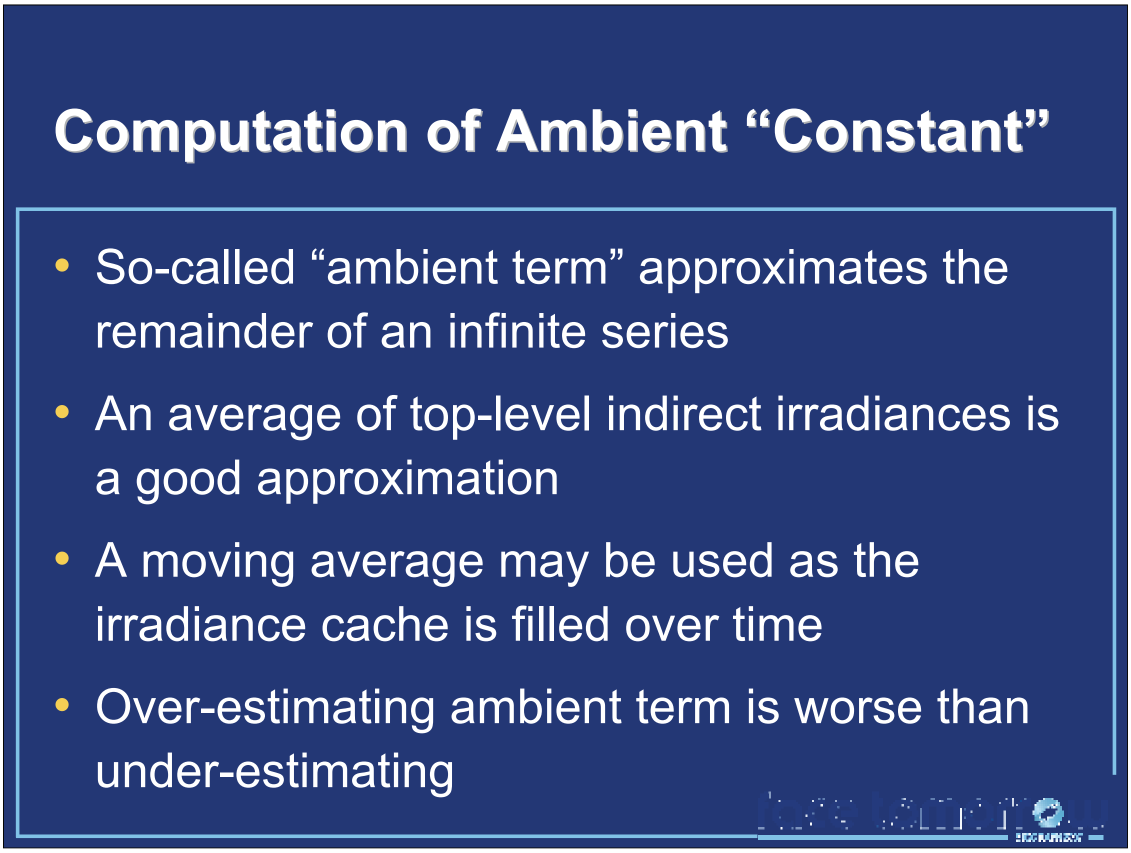
Greg Ward

Anyhere Software



Irradiance Cache Enhancements in *Radiance*

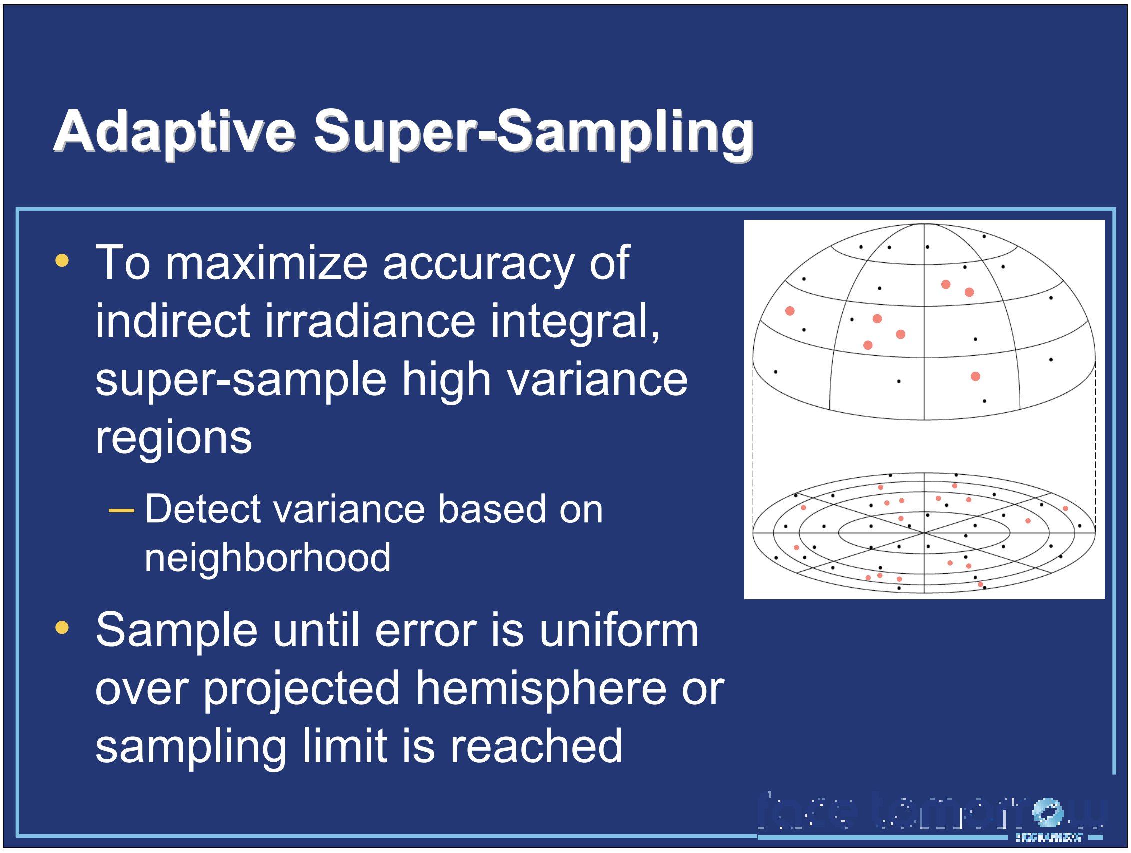
- Computation of ambient “constant”
- Adaptive super-sampling on hemisphere
- Maximum and minimum record spacing
- Gradient limit on record spacing
- Bump maps using rotation gradient
- Options for excluding surfaces/materials
- Record sharing for multiprocessors



Computation of Ambient "Constant"

- So-called "ambient term" approximates the remainder of an infinite series
- An average of top-level indirect irradiances is a good approximation
- A moving average may be used as the irradiance cache is filled over time
- Over-estimating ambient term is worse than under-estimating

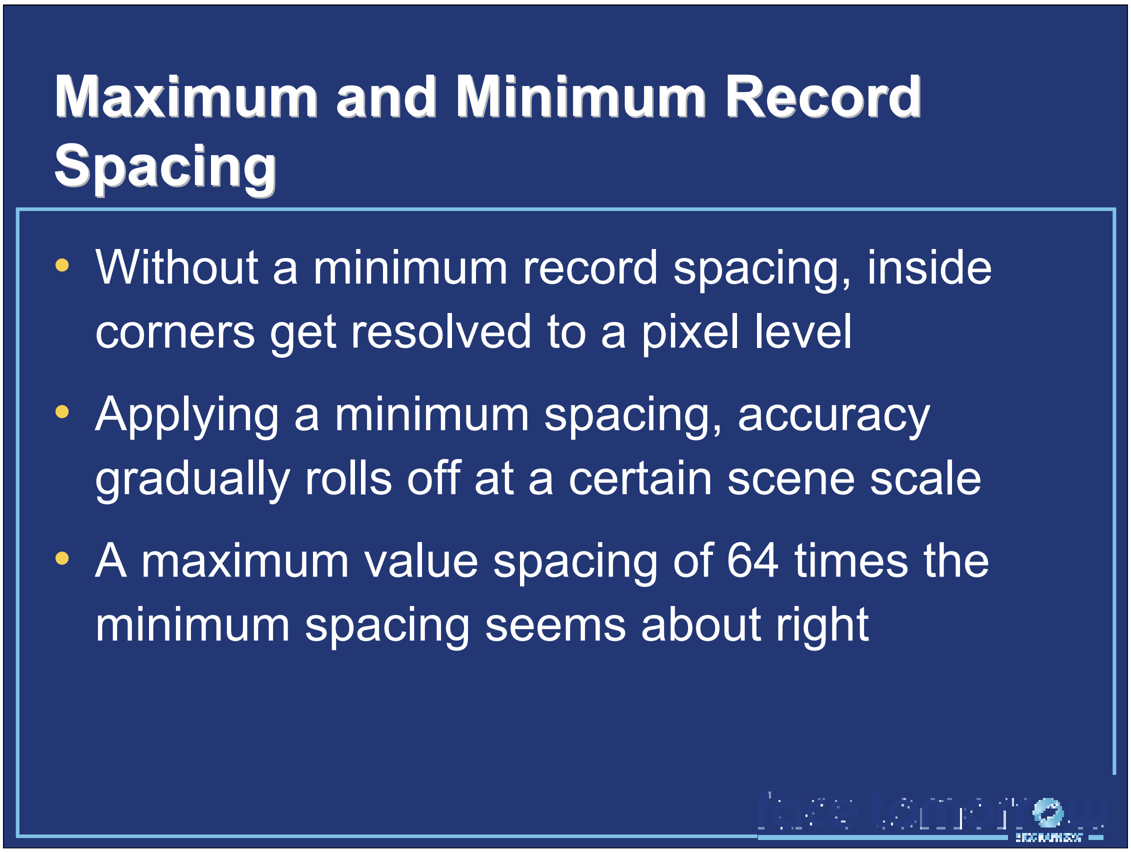
The -aw option in Radiance controls this function. As indirect values are collected, they overtake the initial, user-specified ambient term. It's one of those features that seemed like a good idea at the time, but in practice it doesn't get much use.



Adaptive Super-Sampling

- To maximize accuracy of indirect irradiance integral, super-sample high variance regions
  - Detect variance based on neighborhood
- Sample until error is uniform over projected hemisphere or sampling limit is reached

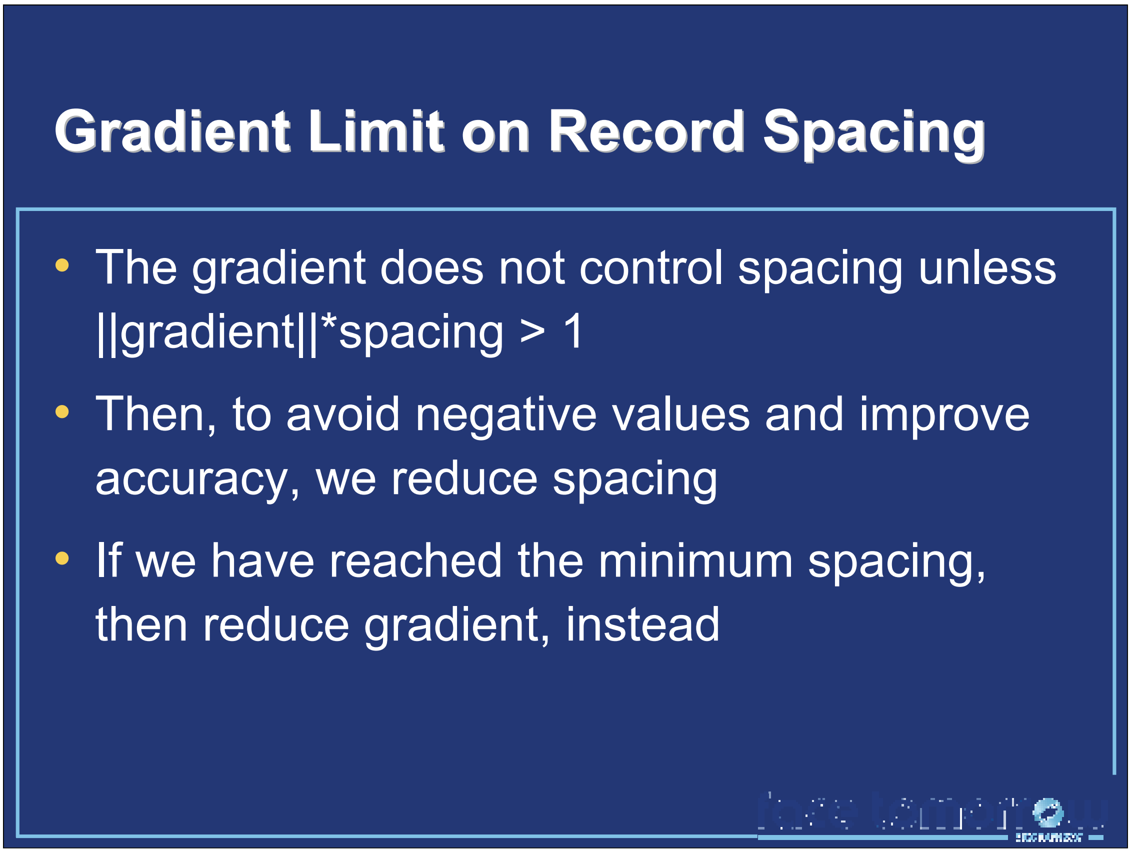
Controlled by the -as option in Radiance, this often-used optimization is especially effective in bright, daylighted interiors.



Maximum and Minimum Record Spacing

- Without a minimum record spacing, inside corners get resolved to a pixel level
- Applying a minimum spacing, accuracy gradually rolls off at a certain scene scale
- A maximum value spacing of 64 times the minimum spacing seems about right

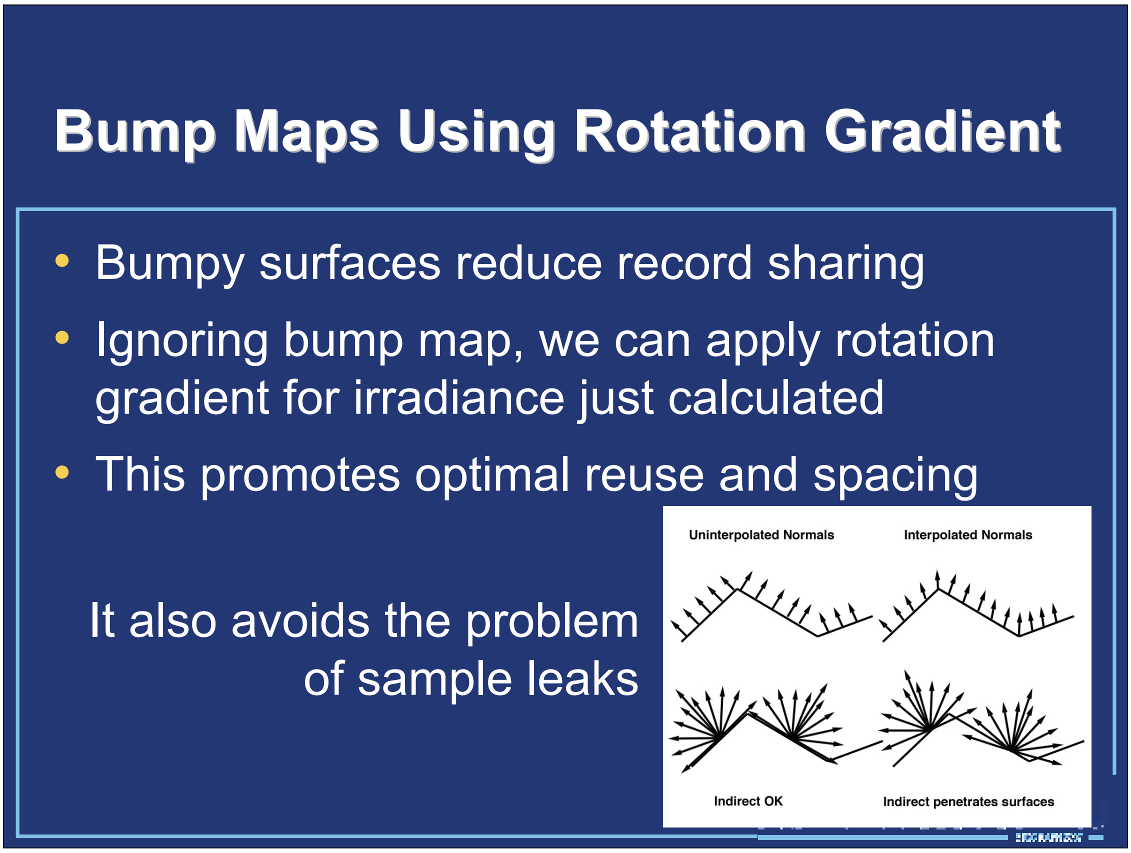
If available, ray pixel size may also be used to adjust record spacing, though this would tend to undermine view independence.



Gradient Limit on Record Spacing

- The gradient does not control spacing unless ||gradient||*spacing > 1
- Then, to avoid negative values and improve accuracy, we reduce spacing
- If we have reached the minimum spacing, then reduce gradient, instead

It is important *not* to use the gradient to determine record spacing in general, as we cannot know what the gradient is before sampling, and we don’t know how often to sample if the gradient dictates spacing. It is better to use a conservative metric that doesn’t depend on actual scene radiances, which may or may not behave as expected.



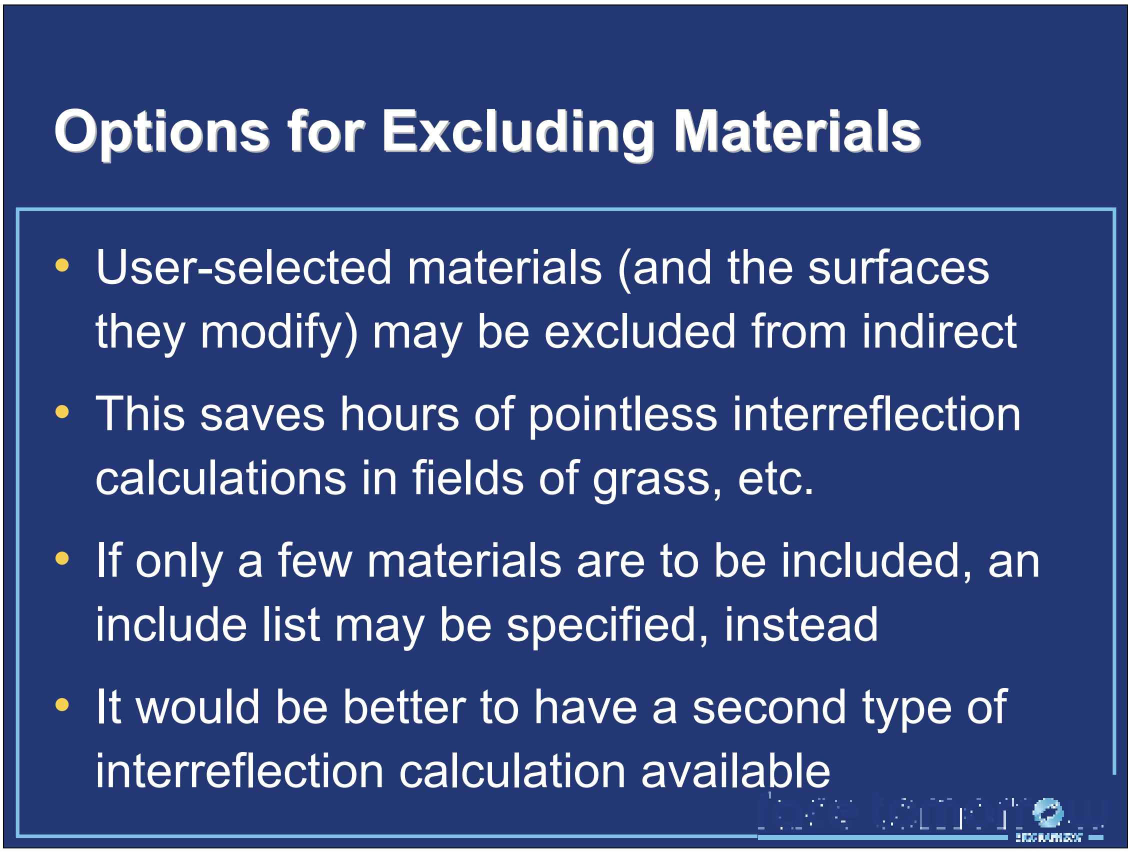
Bump Maps Using Rotation Gradient

- Bumpy surfaces reduce record sharing
- Ignoring bump map, we can apply rotation gradient for irradiance just calculated
- This promotes optimal reuse and spacing

It also avoids the problem of sample leaks

Uninterpolated Normals
Interpolated Normals
Indirect OK
Indirect penetrates surfaces

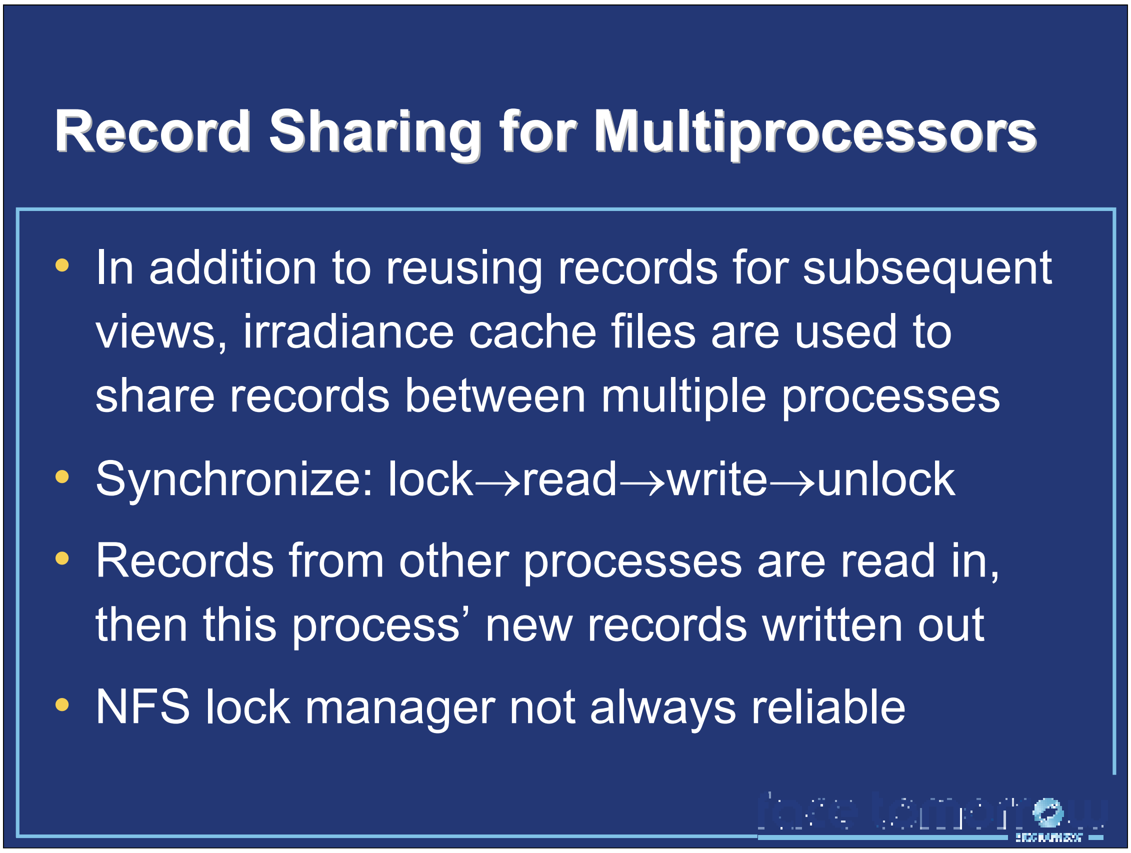
No-cost addition to irradiance gradient calculation -- just apply the perturbed surface normals on the final interpolation.



Options for Excluding Materials

- User-selected materials (and the surfaces they modify) may be excluded from indirect
- This saves hours of pointless interreflection calculations in fields of grass, etc.
- If only a few materials are to be included, an include list may be specified, instead
- It would be better to have a second type of interreflection calculation available

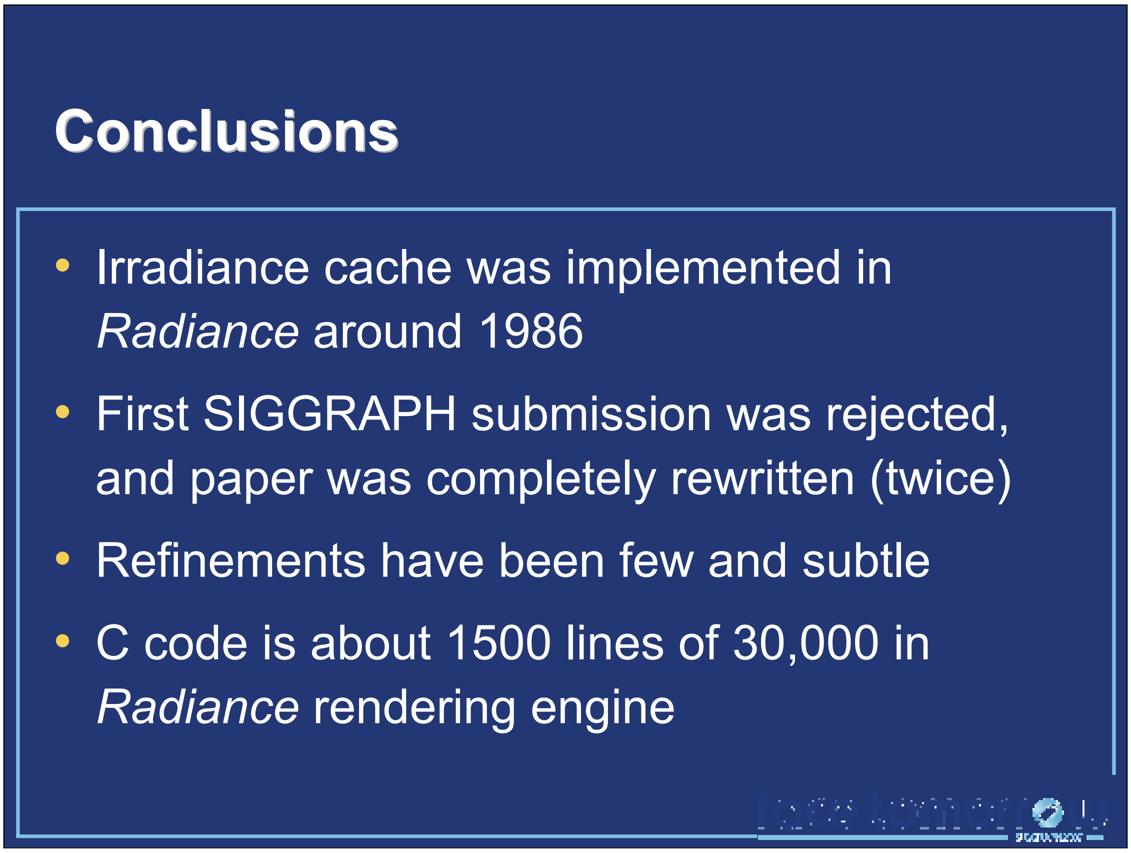
The -ae option excludes a single material (and its assigns), whereas -aE excludes all materials listed in a file. The -ai and -aI options may be used to specify included materials, instead.



Record Sharing for Multiprocessors

- In addition to reusing records for subsequent views, irradiance cache files are used to share records between multiple processes
- Synchronize: lock→read→write→unlock
- Records from other processes are read in, then this process’ new records written out
- NFS lock manager not always reliable

This technique works with a fixed buffer size of about 13 records up to 10 processors or so, then a larger buffer works better. At some point, I would like to implement a client-server or broadcast model to reduce overhead and avoid problems with NFS.



Conclusions

- Irradiance cache was implemented in *Radiance* around 1986
- First SIGGRAPH submission was rejected, and paper was completely rewritten (twice)
- Refinements have been few and subtle
- C code is about 1500 lines of 30,000 in *Radiance* rendering engine

SIGGRAPH2007

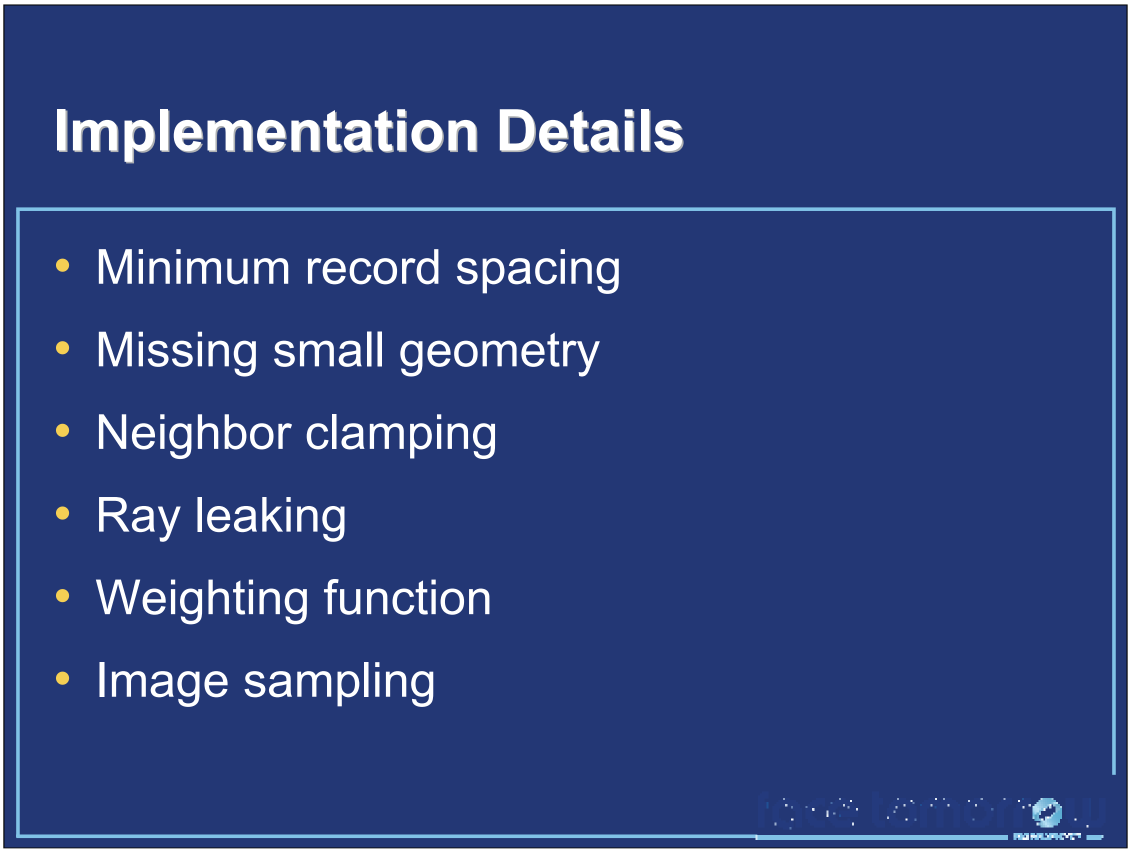
Implementation Details

SIGGRAPH2007

Jaroslav Křivánek

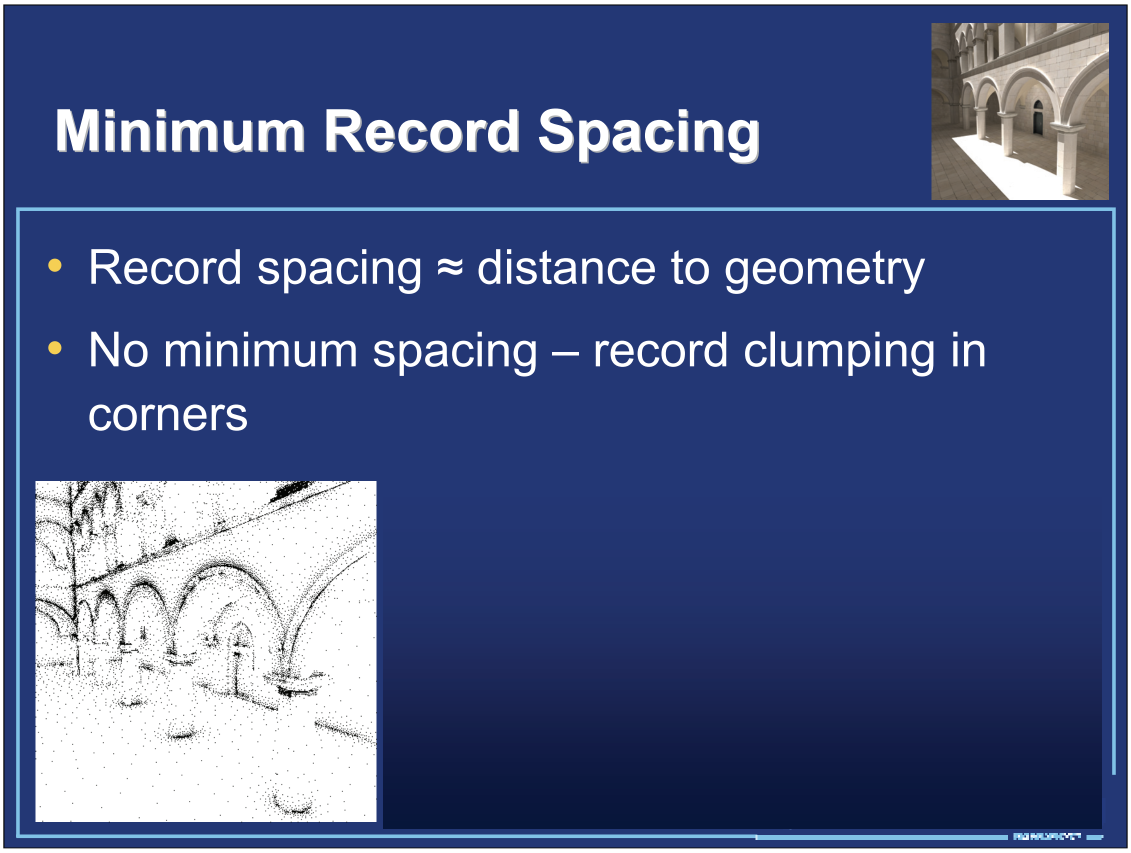
ČVUT v Praze – CTU Prague

In this part of the course, we will discuss a number of tricks that make irradiance caching a reliable algorithm. At the first sight, they might not make much sense, but it is quite difficult to get irradiance caching produce artifact-free images without using these tricks.



Implementation Details

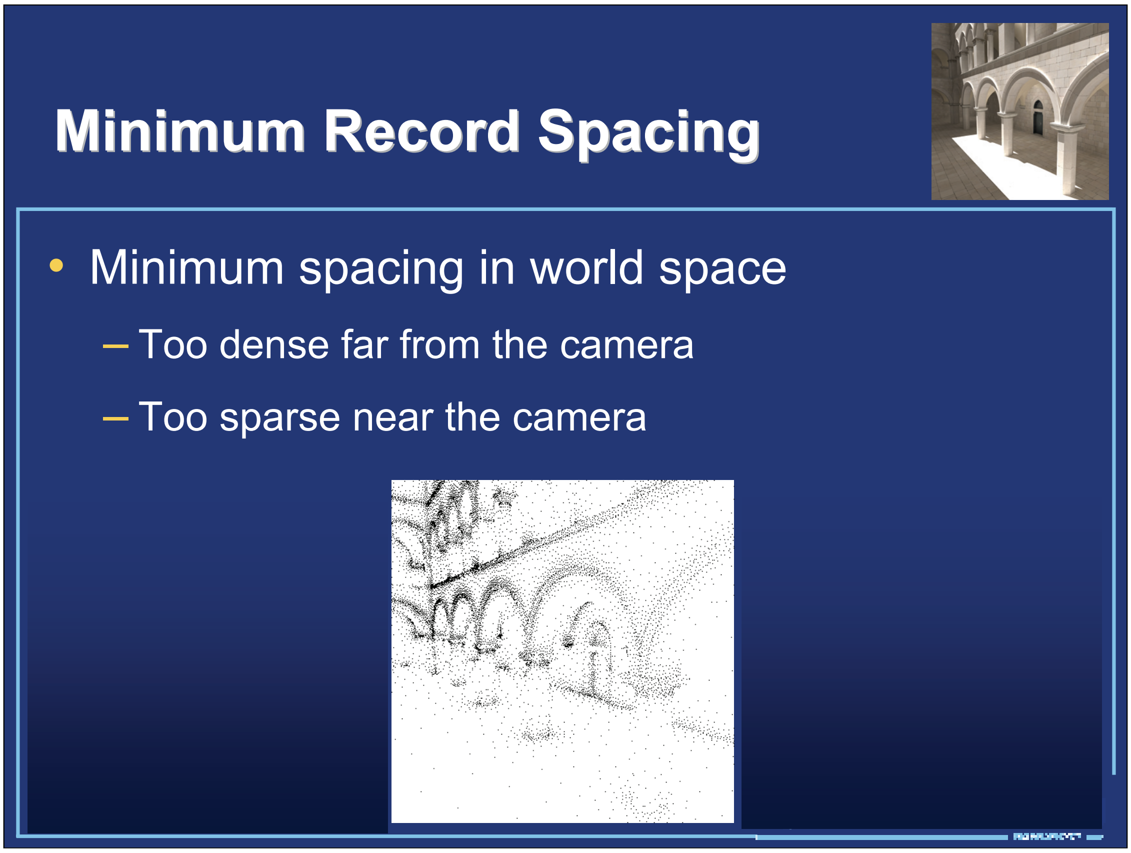
- Minimum record spacing
- Missing small geometry
- Neighbor clamping
- Ray leaking
- Weighting function
- Image sampling



Minimum Record Spacing

- Record spacing ≈ distance to geometry
- No minimum spacing – record clumping in corners

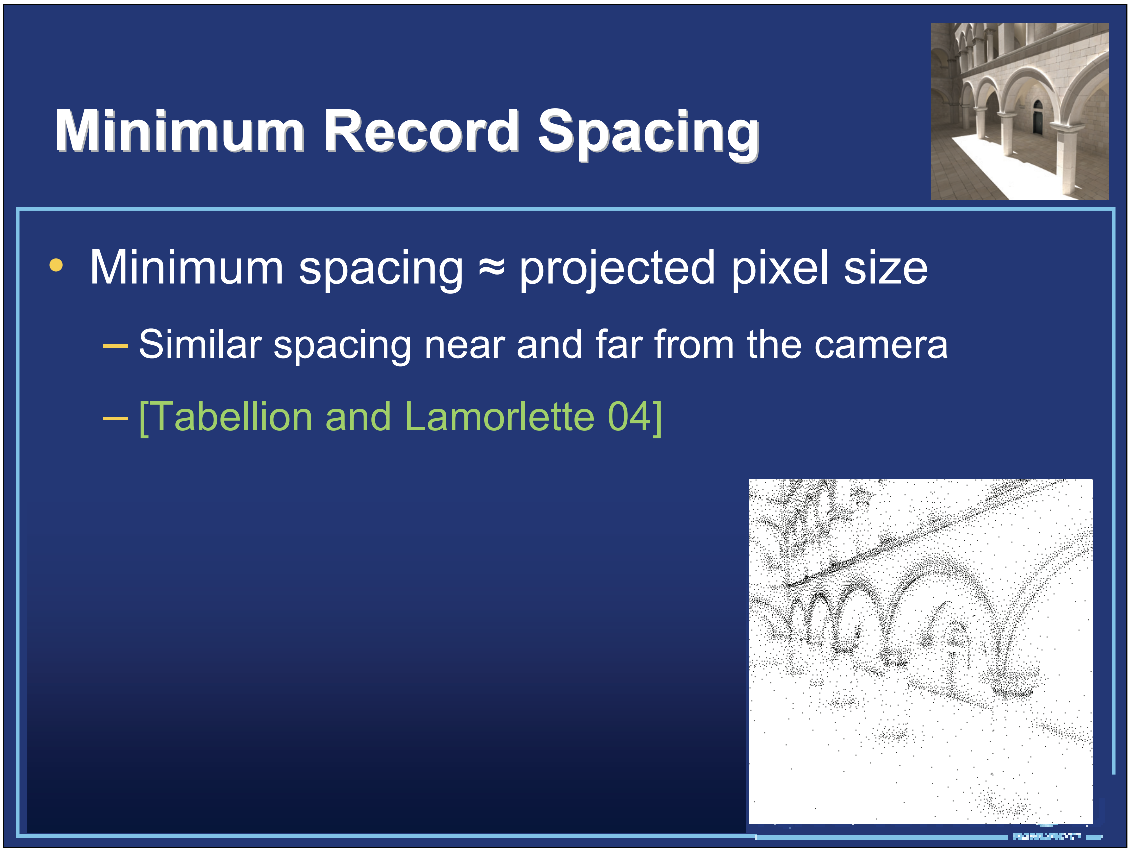
Remember that the spacing of irradiance records is given by the mean distance to the neighboring geometry (and also by the object curvature, which we disregard in this discussion). If you do not impose any minimum limit on the spacing, irradiance caching will spend most of the time generating too many records around edges and corners. To avoid this problem, it is a good idea to impose a minimum distance between the records. This can be done by setting some minimum threshold R_{min} on the R_i value of a record.



Minimum Record Spacing

- Minimum spacing in world space
  - Too dense far from the camera
  - Too sparse near the camera

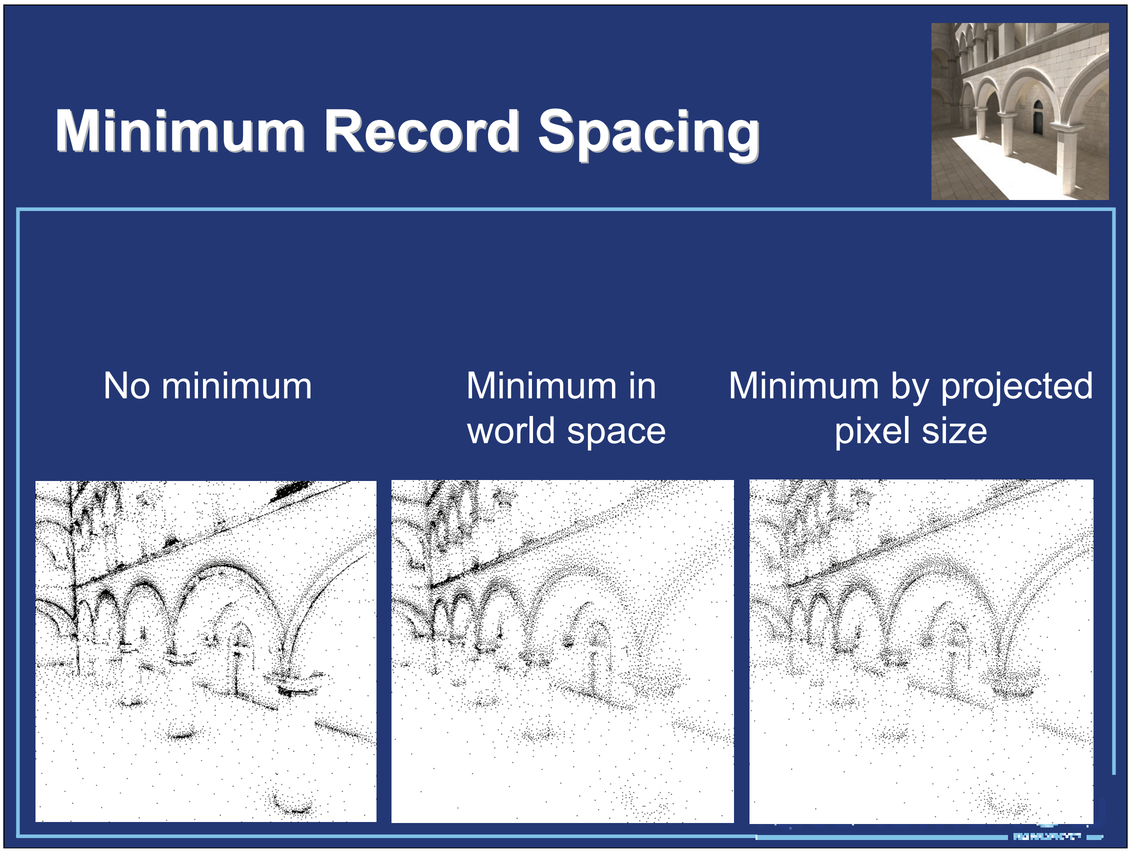
One possibility is to limit the spacing in world space, by fixing the threshold, R_{min}, to the same value all over the scene. In *Radiance*, R_{min} is specified as a fraction of the scene size. This way of limiting the record spacing tends to generate too few radiance cache records near the camera and too many records far away.



Minimum Record Spacing

- Minimum spacing ≈ projected pixel size
  - Similar spacing near and far from the camera
  - [Tabellion and Lamorlette 04]

A better idea, proposed by Tabellion and Lamorlette [2004] is to use a multiple of the projected pixel size for the threshold R_{min}. Good values for R_{min} range between 1.5x and 3x the projected pixel size.



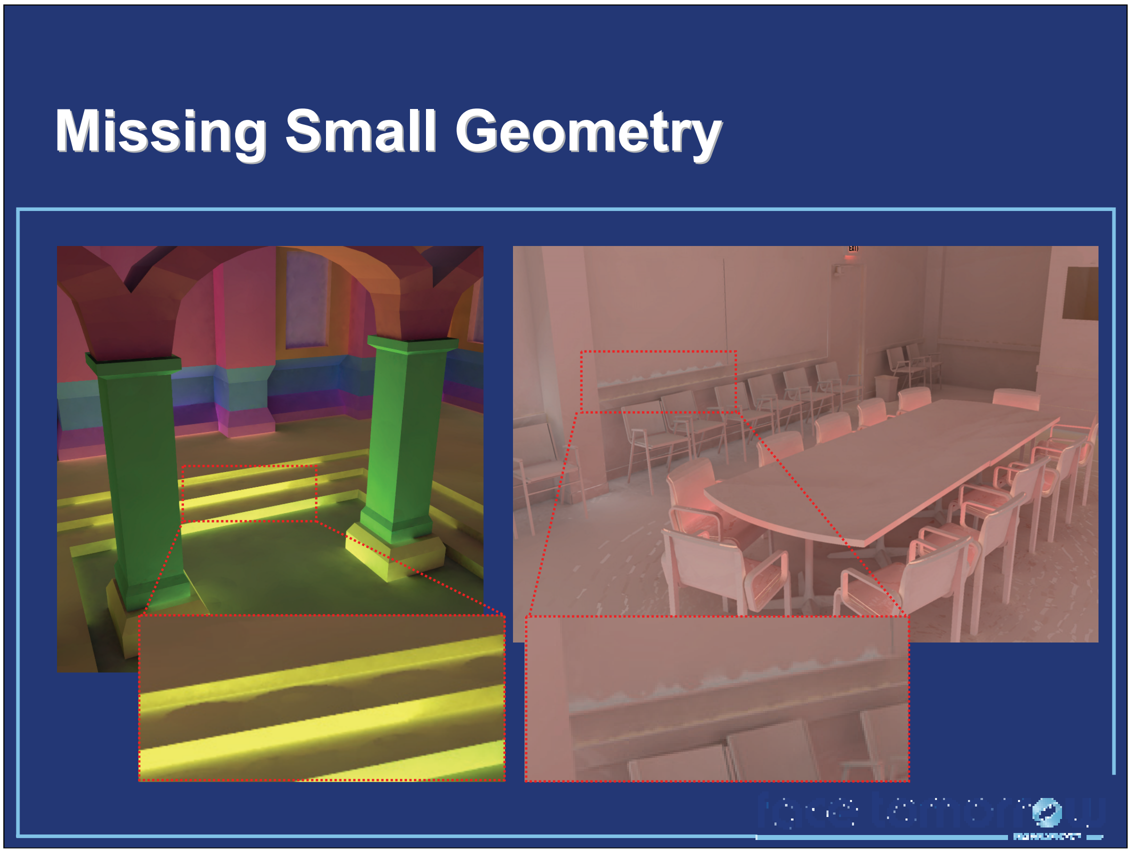
Minimum Record Spacing

No minimum

Minimum in world space

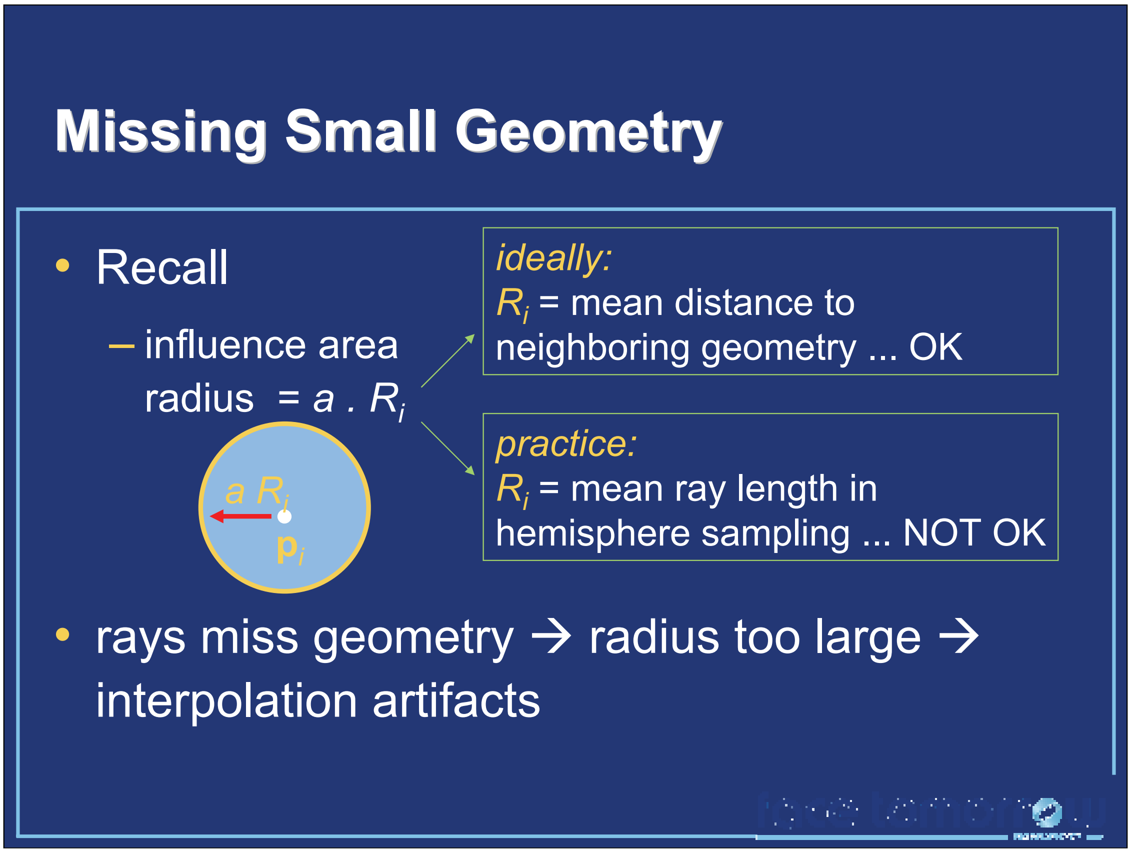
Minimum by projected pixel size

Especially for exterior scenes, it is also important to limit the maximum value of R_i. In *Radiance*, this maximum is 64 times the minimum. Tabellion and Lamorlette [2004] use the maximum of 10x the projected pixel size.



Missing Small Geometry

A common problem in irradiance caching is that rays in the stochastic hemisphere sampling miss geometry features in the scene. This can produce clearly visible image artifacts.



Missing Small Geometry

- Recall
  - influence area radius = $a \cdot R_i$

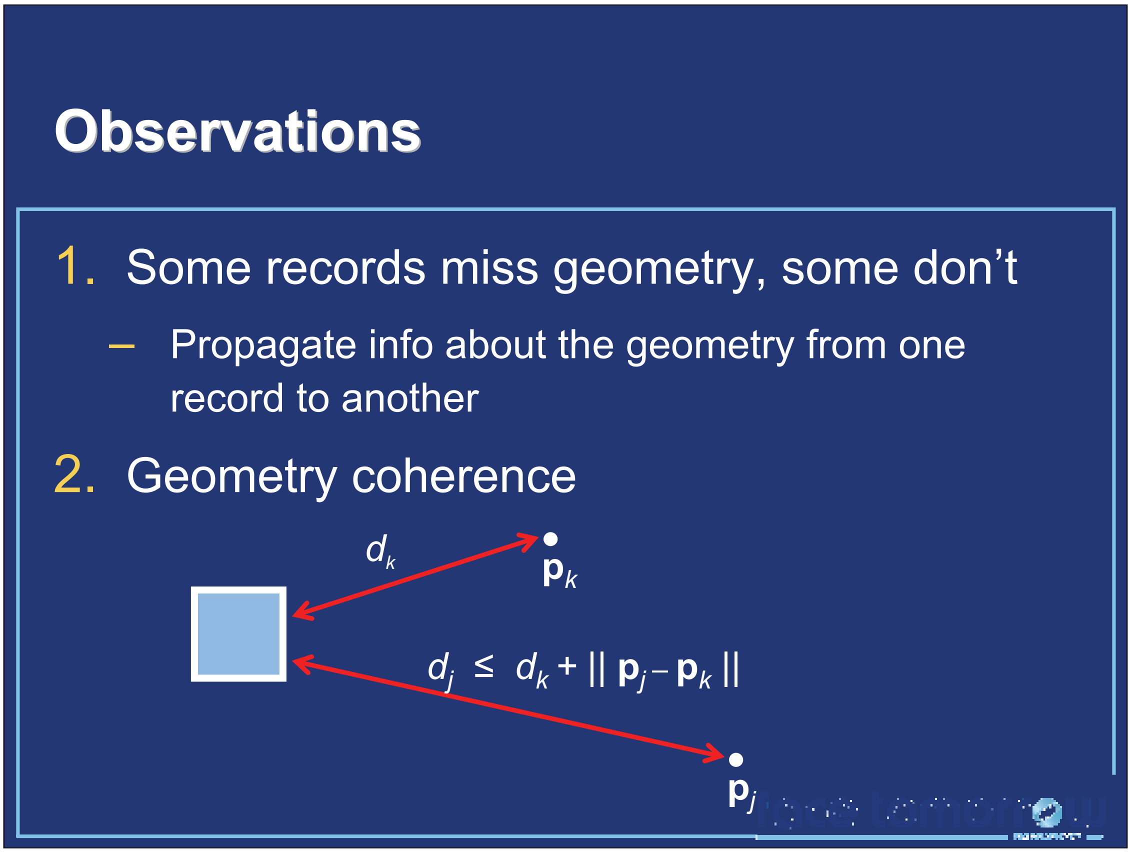
$a\ R_i$
$\mathbf{p}_i$

ideally:
R_i = mean distance to neighboring geometry ... OK

practice:
R_i = mean ray length in hemisphere sampling ... NOT OK

- rays miss geometry → radius too large → interpolation artifacts

Ideally, record spacing would be determined by the mean distance to the neighboring geometry. In practice, this distance is determined as the mean of the ray lengths in hemisphere sampling. If rays miss some geometry, the resulting mean distance is overestimated and we can see discontinuities in the resulting images due to interpolation.



Observations

1. Some records miss geometry, some don't
   - Propagate info about the geometry from one record to another
2. Geometry coherence

d_k

$\mathbf{p}_k$

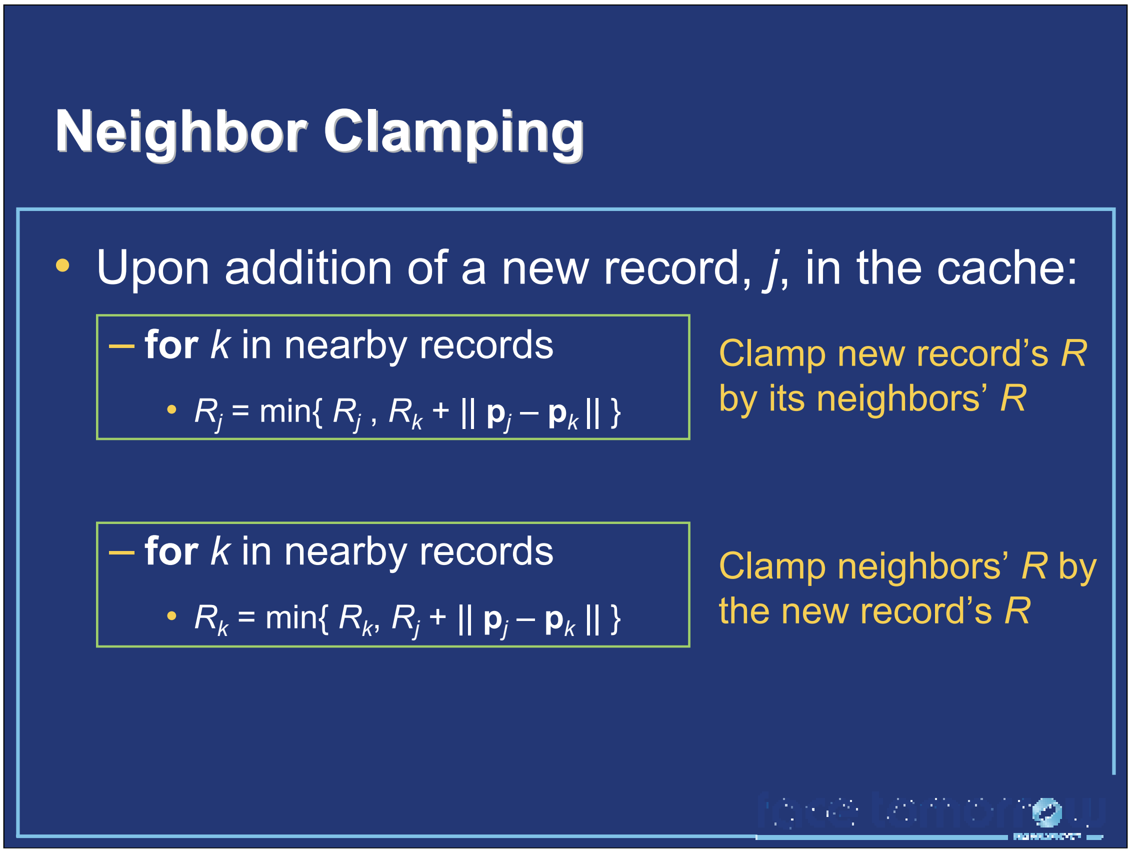
$d_j \le d_k + \| \mathbf{p}_j - \mathbf{p}_k \|$

$\mathbf{p}_j$

A reliable way of detecting the over-estimated mean distance due to missing geometry in hemisphere sampling is based on two observations.

1. Because only few record usually suffer from the overestimated mean distance, we could use the distance estimate at other records to rectify the overestimated distance.
2. Distances obey the triangle inequality. If one record, $\mathbf{p}_k$, is at the distance d_k from some geometry feature, then another record, $\mathbf{p}_j$, cannot be farther from this geometry feature than $d_k + \| \mathbf{p}_j - \mathbf{p}_k \|$.

If we replace d_k and d_j by the mean distance R_k, R_j, we can detect suspicious cases by comparing R_k, R_j to the distance between the two records to verify if the triangle inequality holds. Strictly speaking, this should only work if R_k, R_j, was the distance to the *nearest* geometry feature, but in practice this works fine even for *mean* distance. We call this heuristic 'neighbor clamping'.



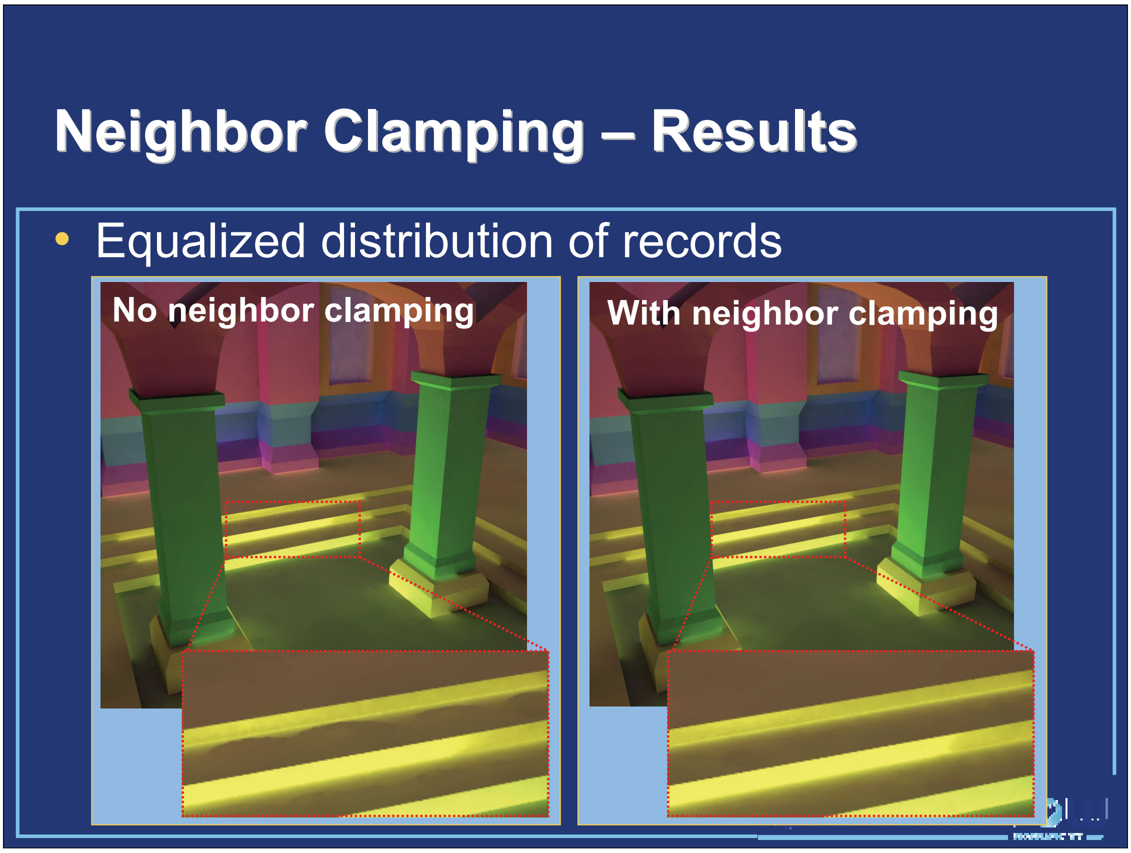
Neighbor Clamping

- Upon addition of a new record, *j*, in the cache:
  - **for** *k* in nearby records
    - $R_j = \min\{ R_j , R_k + \| \mathbf{p}_j - \mathbf{p}_k \| \}$
  - **for** *k* in nearby records
    - $R_k = \min\{ R_k, R_j + \| \mathbf{p}_j - \mathbf{p}_k \| \}$

Clamp new record's *R* by its neighbors' *R*

Clamp neighbors' *R* by the new record's *R*

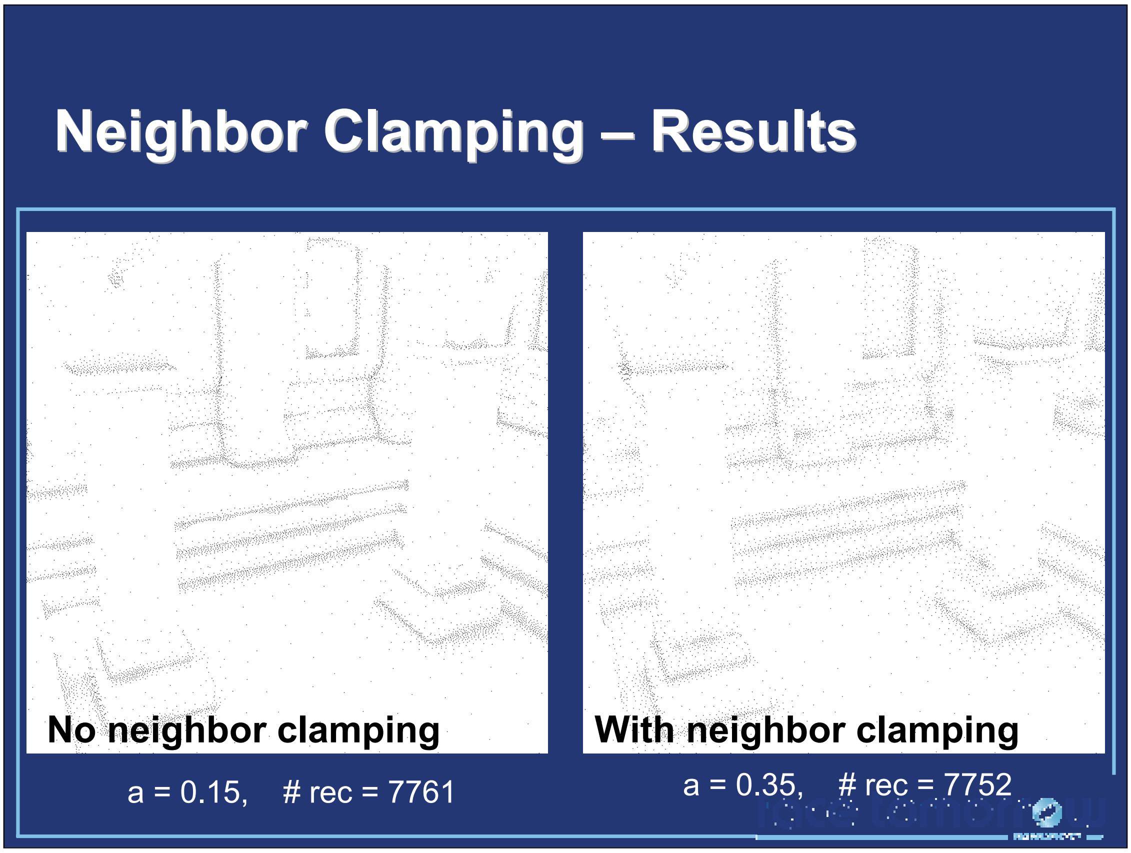
Here is how we proceed in practice. When a new record, *j*, is added to the cache, we first locate all existing records whose area of influence overlap with the area of influence of the record being added. (That is to say, all records *k*, such that $\| \mathbf{p}_j - \mathbf{p}_k \| < R_j + R_k$). Then for all those records, we clamp the R_j value of the new record: $R_j = \min\{ R_j , R_k + \| \mathbf{p}_j - \mathbf{p}_k \| \}$. This enforces the triangle inequality. After that, we use the clamped R_j value of the new record to clamp the R_k values of the nearby records. This second step enforces the *transitivity* of triangle inequality.



Neighbor Clamping – Results

- Equalized distribution of records

And voilà, the artifacts due to the over-estimated mean distance are gone.



Neighbor Clamping – Results

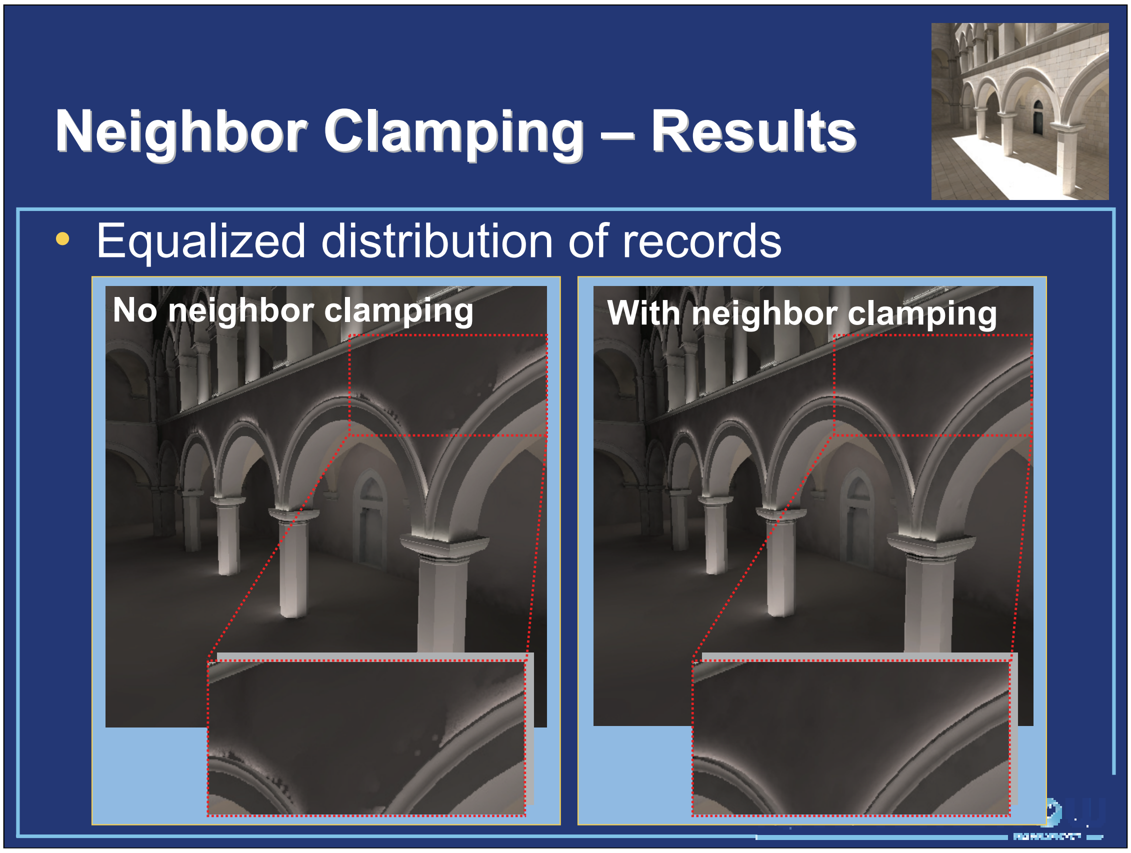
No neighbor clamping

a = 0.15,   # rec = 7761

With neighbor clamping

a = 0.35,   # rec = 7752

With neighbor clamping, the spacing between the records is equalized. It falls off gradually as we move away from the geometry features.



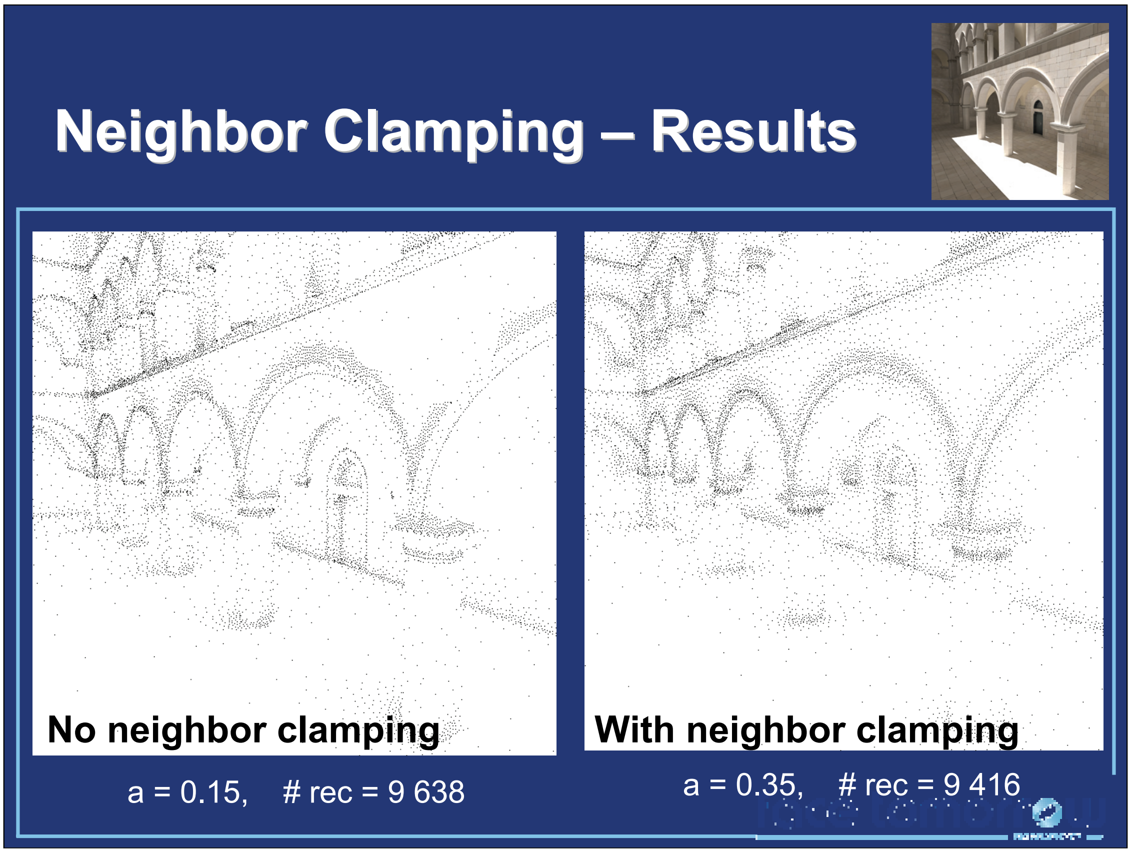
Neighbor Clamping – Results

- Equalized distribution of records

No neighbor clamping

With neighbor clamping

In the Sponza atrium scene, the benefit of neighbor clamping is even more apparent.



Neighbor Clamping – Results

No neighbor clamping

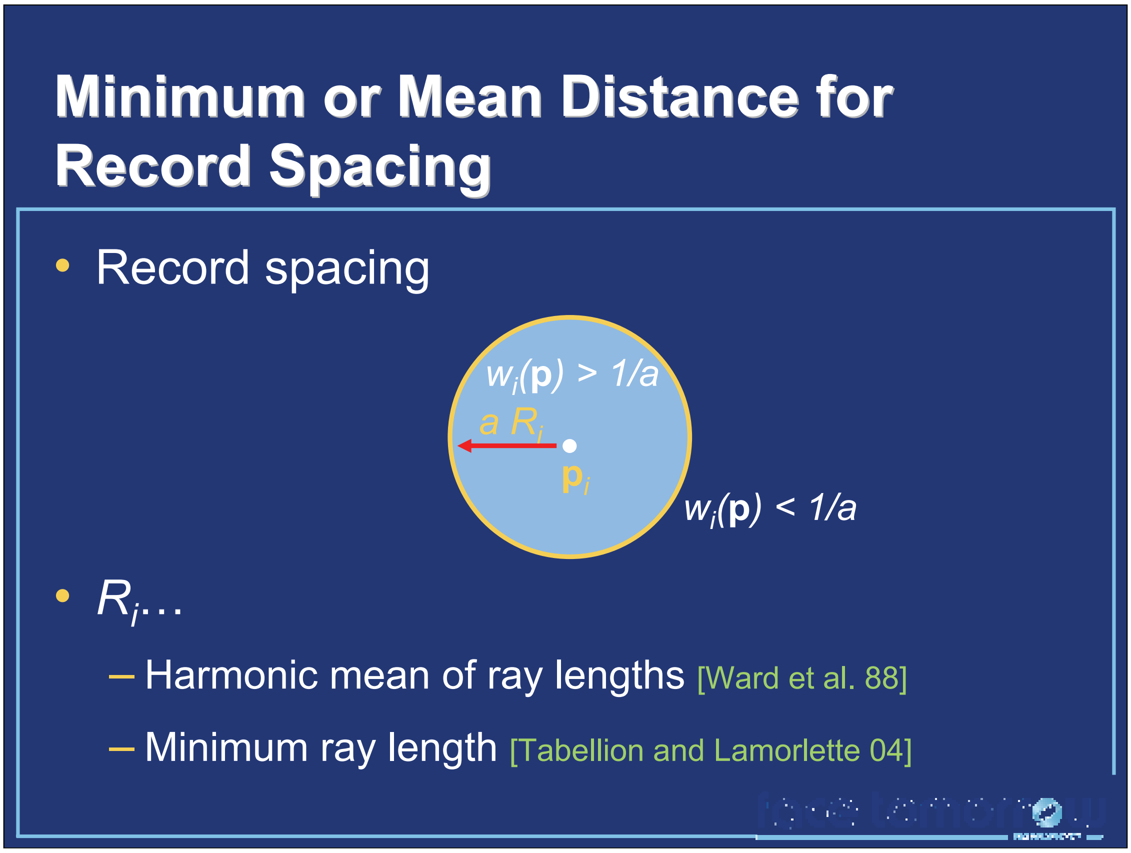
a = 0.15,   # rec = 9 638

With neighbor clamping

a = 0.35,   # rec = 9 416

Record spacing is equalized and indirect illumination is properly sampled around the cornices above the arches.

To conclude, irradiance caching produces image artifacts when the mean distance to geometry is overestimated. Neighbor clamping reliably detects and corrects the overestimated mean distance, thereby suppressing these artifacts.



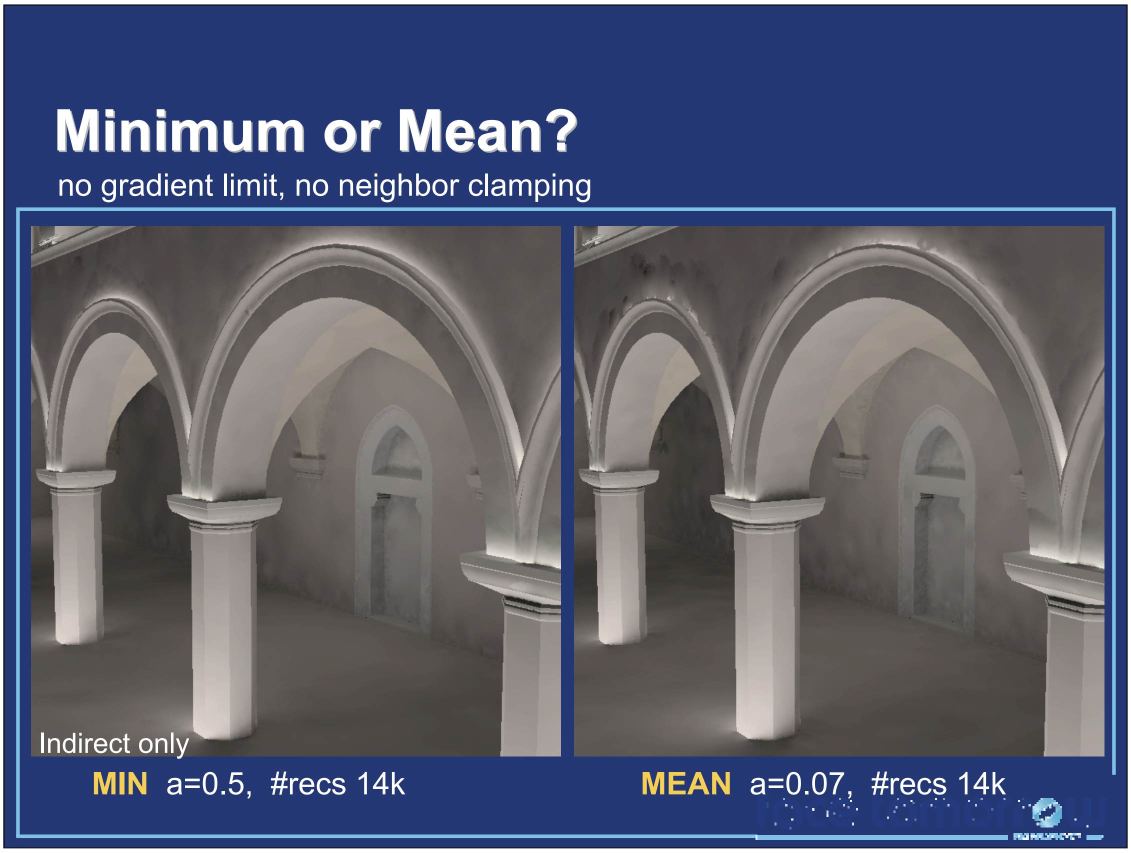
Minimum or Mean Distance for Record Spacing

- Record spacing

$w_i(\mathbf{p}) > 1/a$
$a\ R_i$
$\mathbf{p}_i$
$w_i(\mathbf{p}) < 1/a$

- R_i…
  - Harmonic mean of ray lengths [Ward et al. 88]
  - Minimum ray length [Tabellion and Lamorlette 04]

In the irradiance caching implementation in *Radiance* [Ward et al. 1988], the record spacing is based upon the *mean* distance to neighboring geometry. As shown on the previous slide, this tends to miss some geometry features. We have proposed neighbor clamping to resolve these problems at EGSR in 2006 [Křivánek et al. 2006]. In 2004, Tabellion and Lamorlette [2004] used the minimum distance instead of the mean distance to resolve these problems. Let us see how irradiance caching behaves when using the minimum and mean distance.



Minimum or Mean?

no gradient limit, no neighbor clamping

Indirect only

MIN a=0.5, #recs 14k

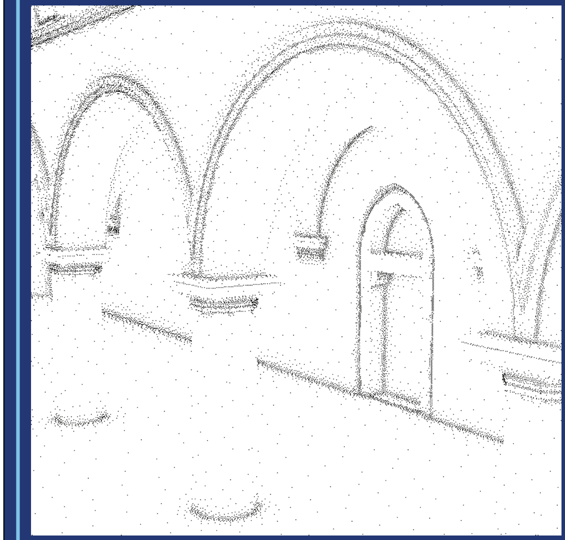
MEAN a=0.07, #recs 14k

First, when there is *no* gradient limit on record spacing and neighbor clamping is *not* used, then the minimum distance indeed produces much better images than the mean distance.

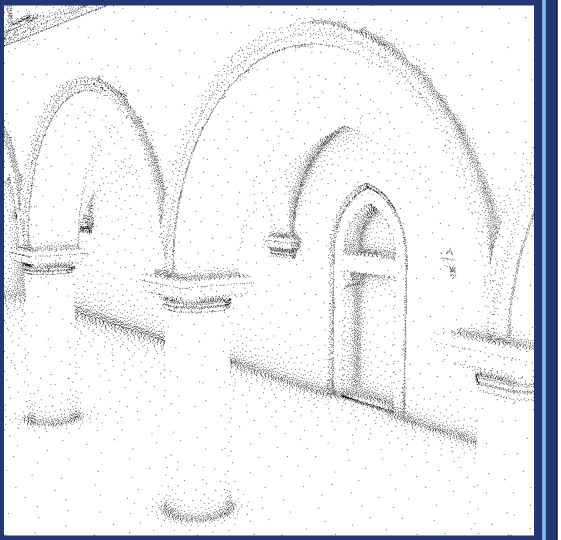
(See Greg Ward's slides on Radiance implementation for more information about the gradient limit on record spacing.)

Minimum or Mean?

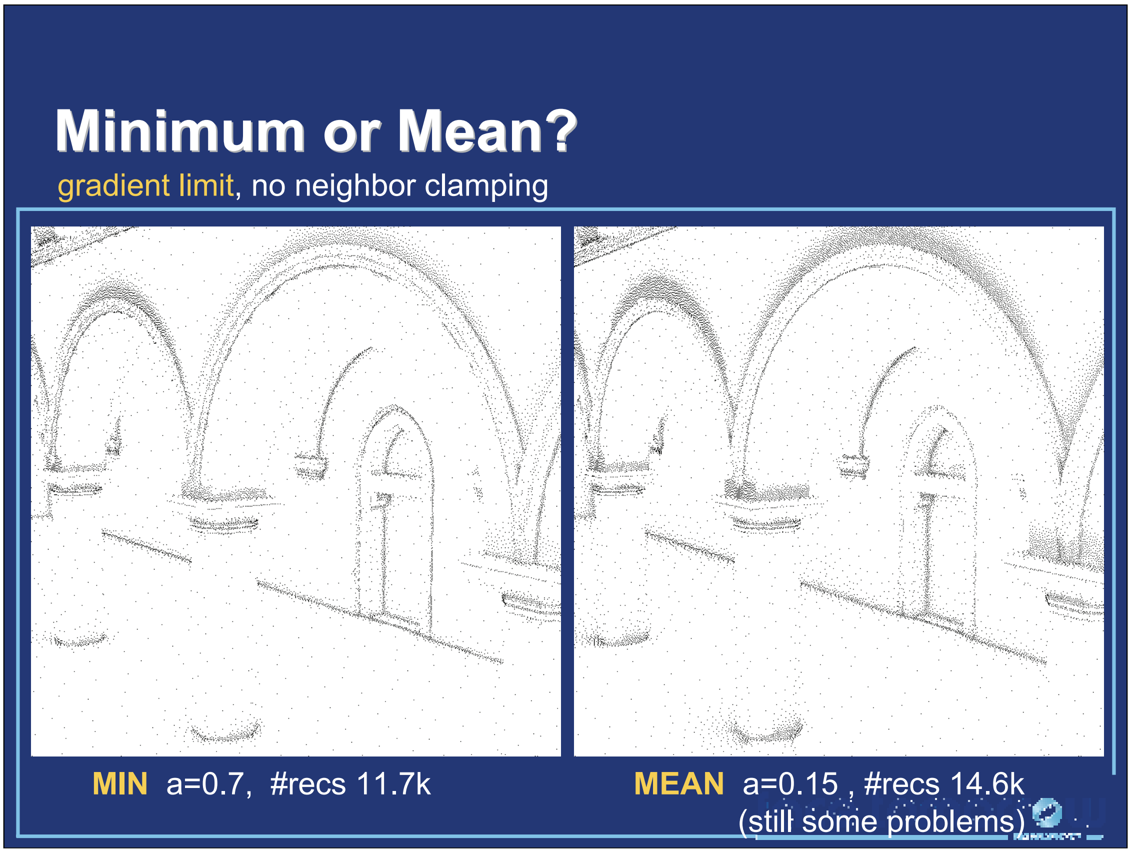
no gradient limit, no neighbor clamping



MIN a=0.5, #recs 14k



MEAN a=0.07, #recs 14k



Minimum or Mean?

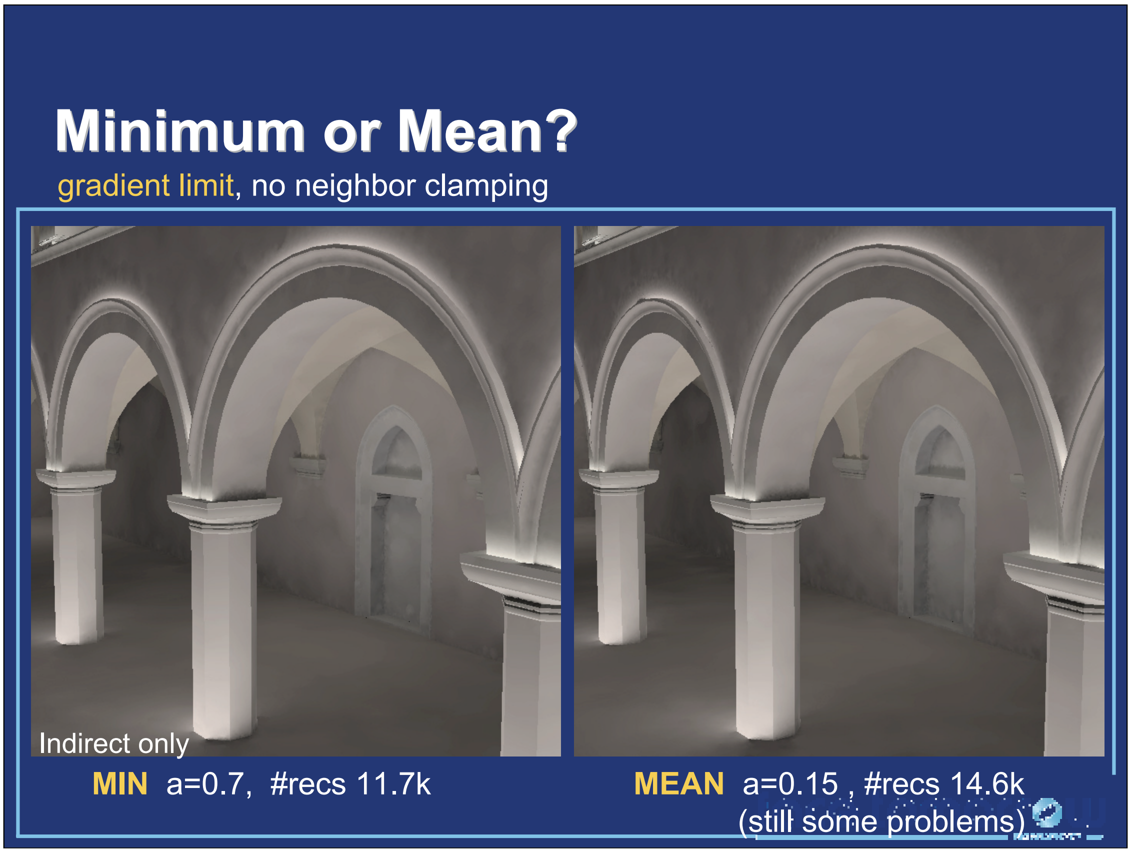
gradient limit, no neighbor clamping

MIN a=0.7, #recs 11.7k

MEAN a=0.15 , #recs 14.6k
(still some problems)

If we limit record spacing by the translational gradient, we get much more records in the high-gradient areas around the cornices.

This is also a nice example of the gradient limit on record spacing.



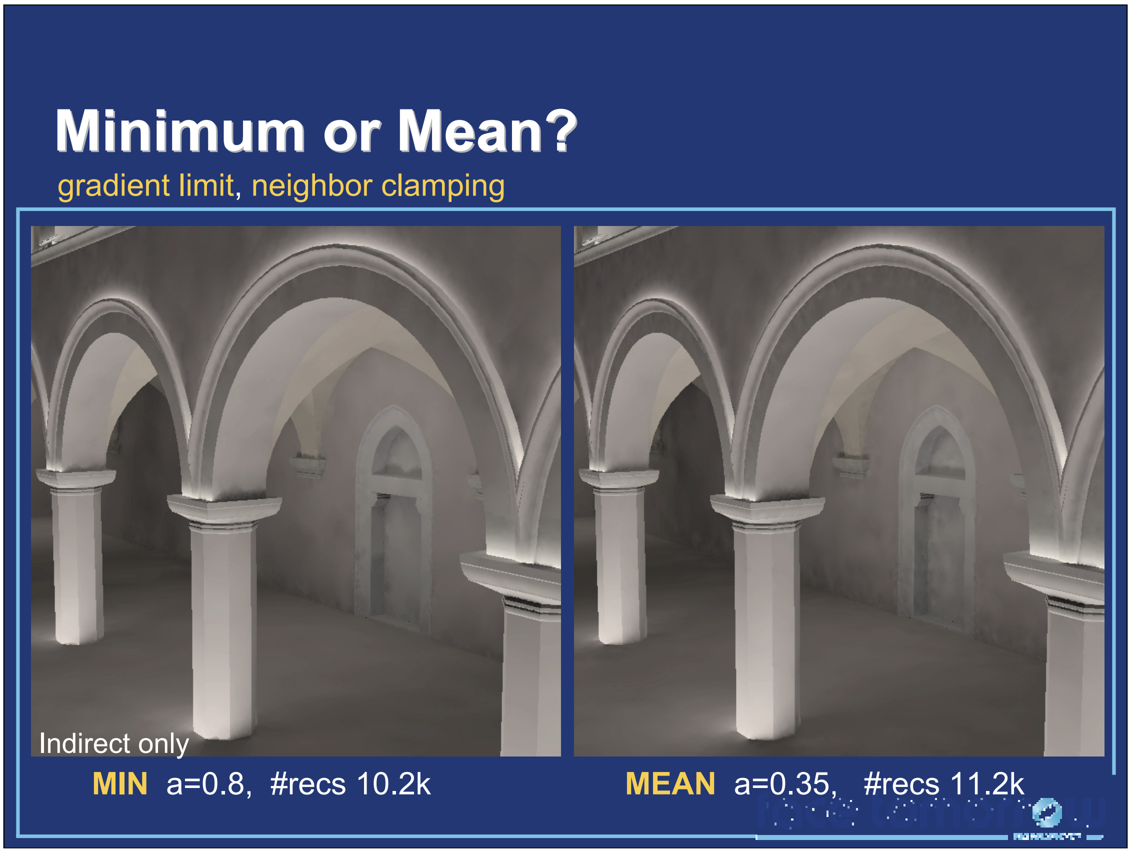
Minimum or Mean?

gradient limit, no neighbor clamping

MIN a=0.7, #recs 11.7k

MEAN a=0.15 , #recs 14.6k
(still some problems)

But still, even with the gradient limited spacing, using the mean distance leaves some artifacts around the cornices.



Minimum or Mean?

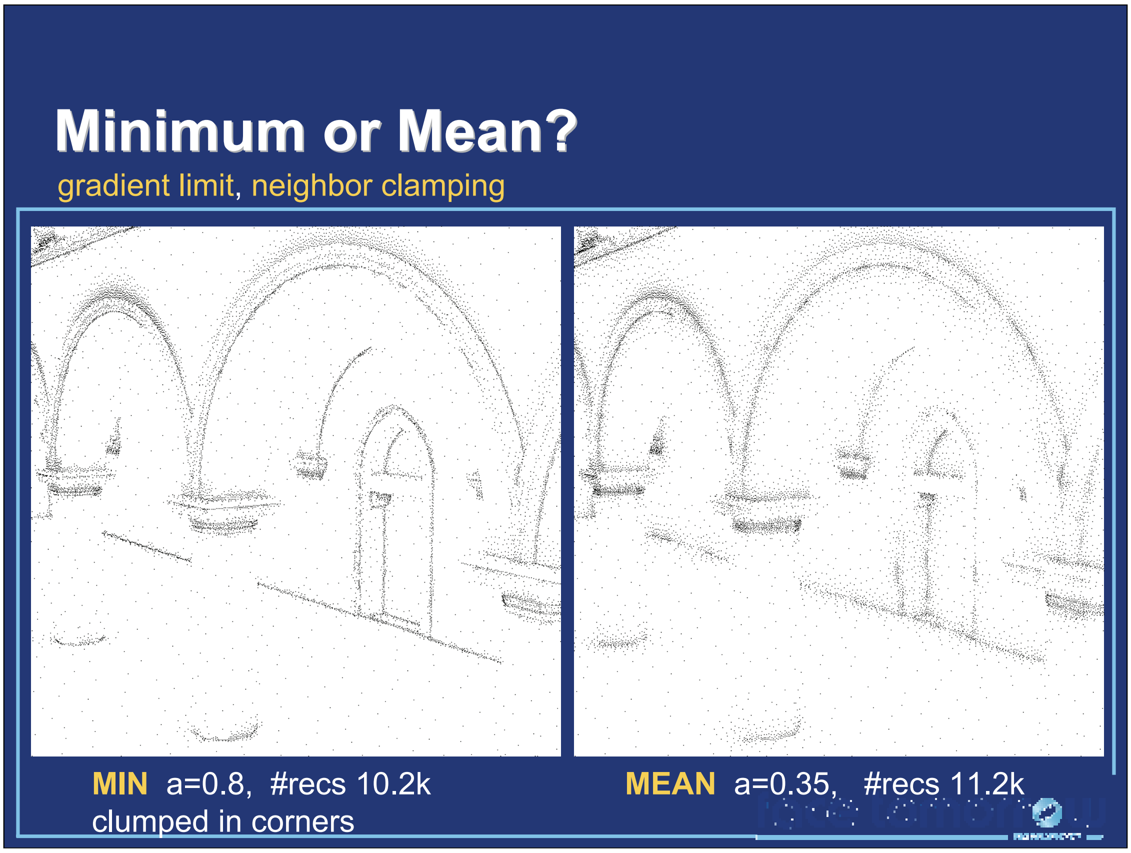
gradient limit, neighbor clamping

Indirect only

MIN a=0.8, #recs 10.2k

MEAN a=0.35, #recs 11.2k

However, when we turn on neighbor clamping, the artifacts are gone.



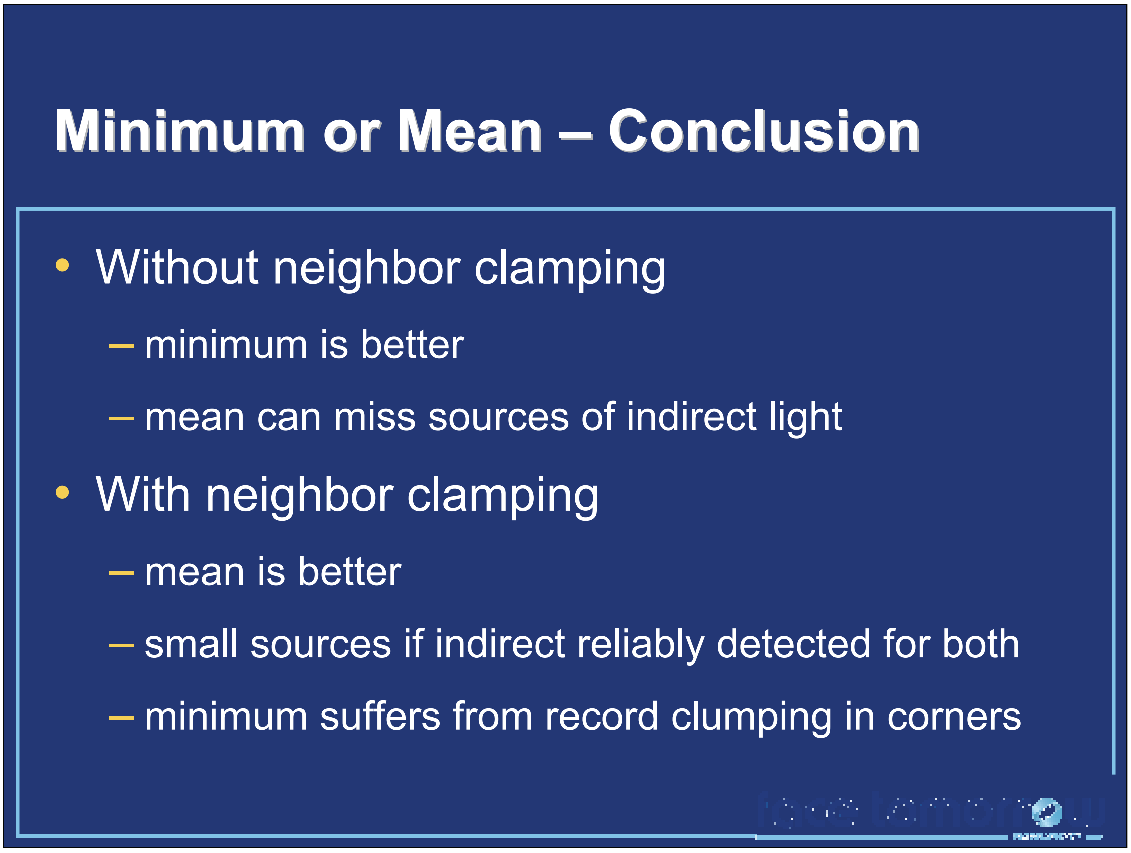
Minimum or Mean?

gradient limit, neighbor clamping

MIN a=0.8, #recs 10.2k
clumped in corners

MEAN a=0.35, #recs 11.2k

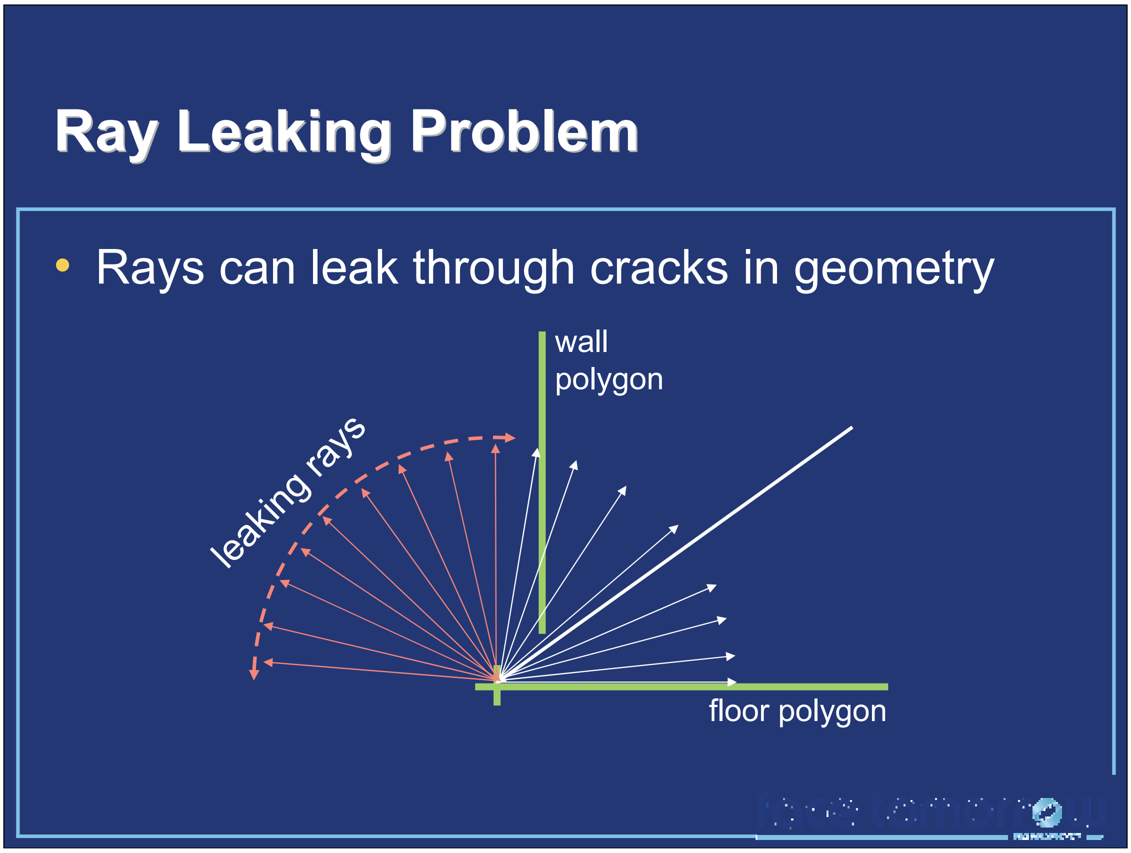
Looking at the record distribution, we see that with the mean distance, record spacing falls of gradually as we move away from the geometry, whereas with the minimum distance, records tend to concentrate in the corners.



Minimum or Mean – Conclusion

- Without neighbor clamping
  - minimum is better
  - mean can miss sources of indirect light
- With neighbor clamping
  - mean is better
  - small sources if indirect reliably detected for both
  - minimum suffers from record clumping in corners

In our experience, the gradual falloff of the spacing is desirable. When gradient limit on record spacing and neighbor clamping is used, then the mean distance produces better images with the same number of records than the minimum distance.



Ray Leaking Problem

- Rays can leak through cracks in geometry

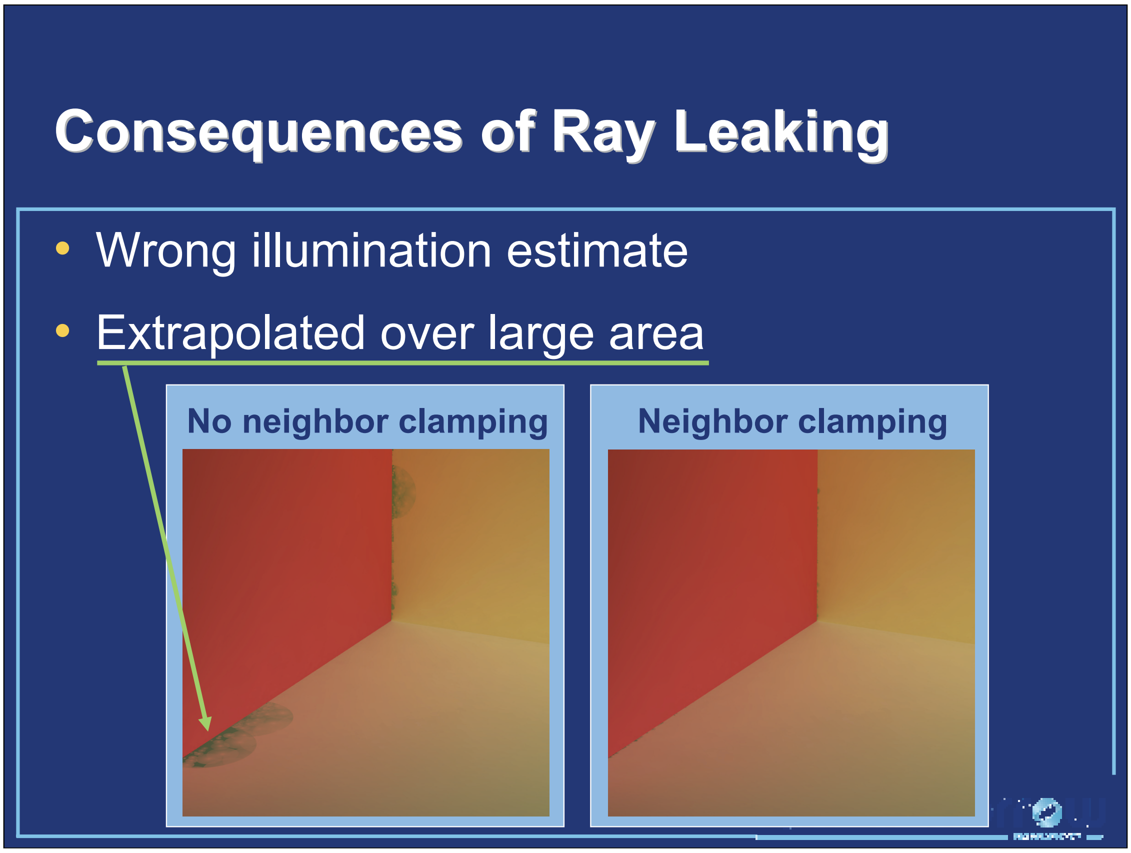
wall polygon

leaking rays

floor polygon

Another serious problem of irradiance caching occurs when handling scenes with small cracks between polygons. Such scenes are quite common in practice – either because the scene is not well modeled or because of a limited numerical precision in the exported scene.

If a primary ray happens to hit such a crack, then most of the secondary rays used in hemisphere sampling “leak” through the crack. As a result, the irradiance estimate is completely wrong and, more seriously, the mean distance is greatly overestimated. (If there were no ray leaking, all those leaking rays would actually be very short.)



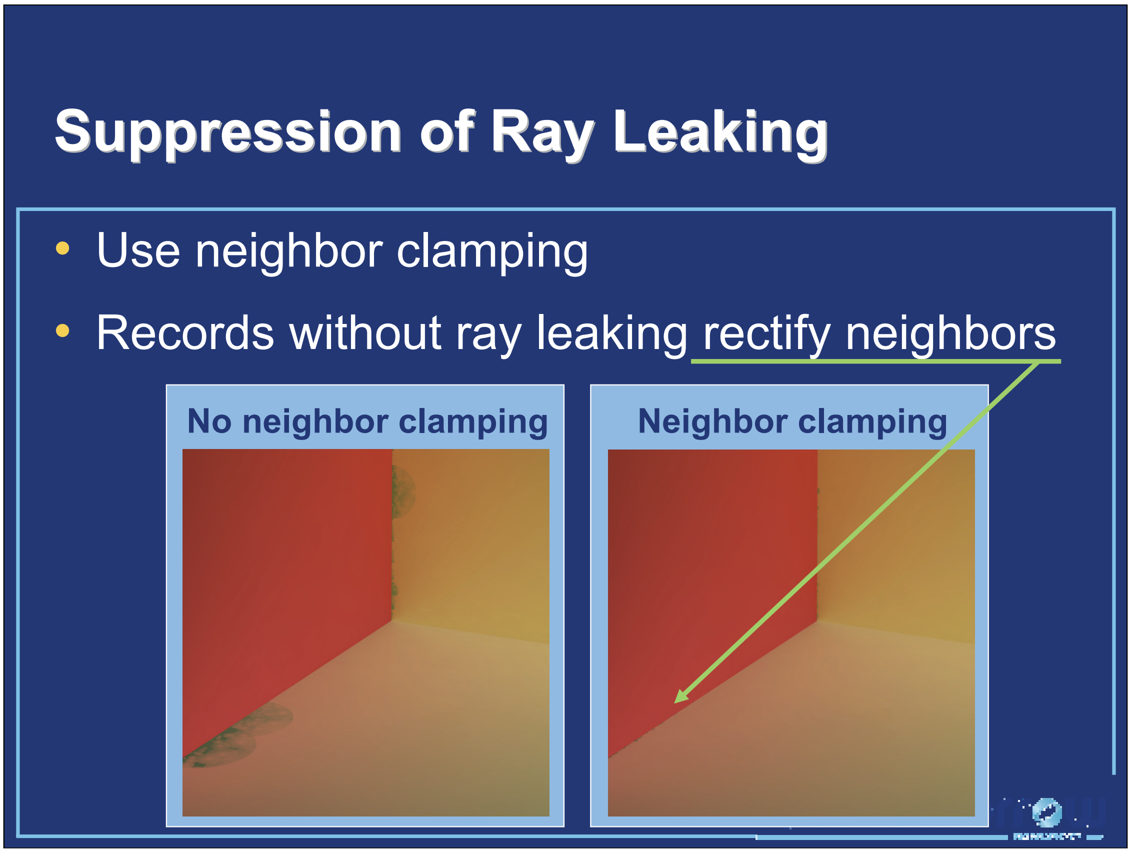
Consequences of Ray Leaking

- Wrong illumination estimate
- Extrapolated over large area

No neighbor clamping

Neighbor clamping

The incorrect irradiance estimate is then extrapolated over a large area.



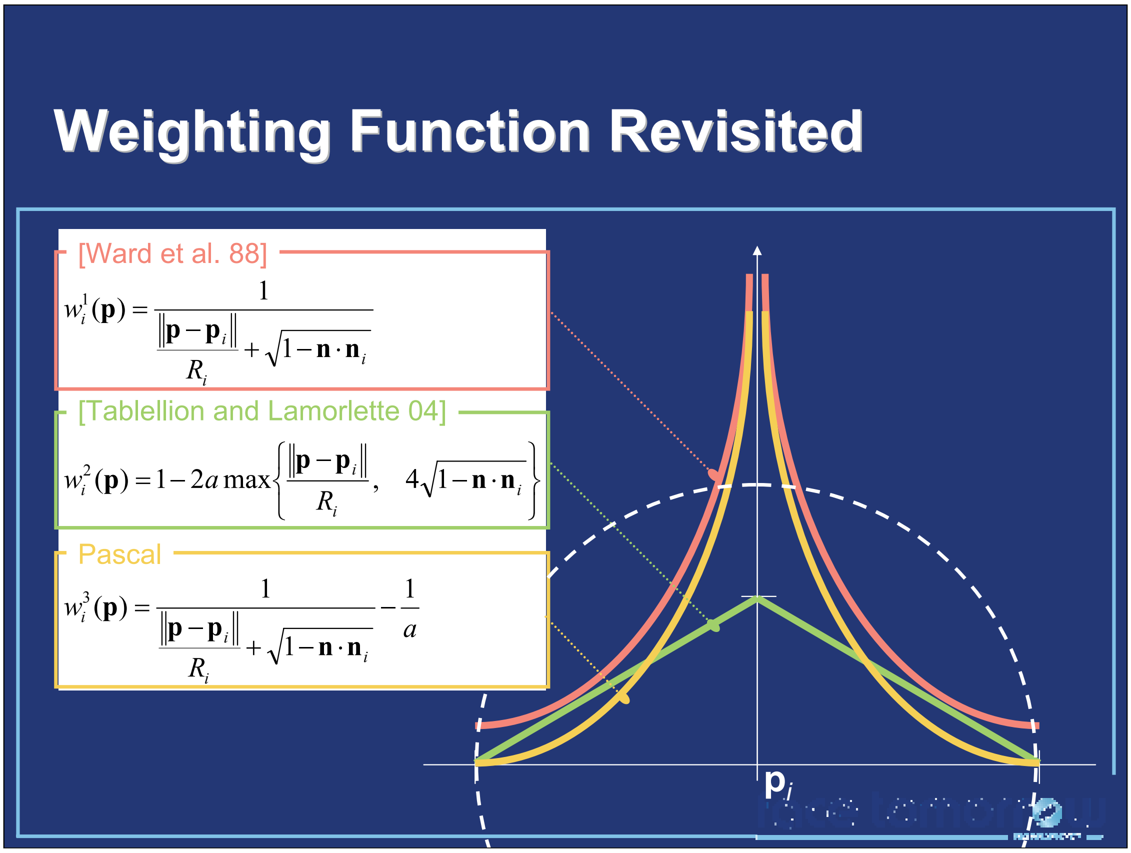
Suppression of Ray Leaking

- Use neighbor clamping
- Records without ray leaking rectify neighbors

No neighbor clamping

Neighbor clamping

A partial remedy is quite simple – just turning on neighbor clamping. Neighbor clamping detects and rectifies overestimated mean distance, so the wrong irradiance estimate is not extrapolated over such a large area. However, neighbor clamping cannot do anything about the wrong irradiance estimate.



Weighting Function Revisited

[Ward et al. 88]

$$w_i^1(\mathbf{p}) = \frac{1}{\frac{\|\mathbf{p}-\mathbf{p}_i\|}{R_i} + \sqrt{1-\mathbf{n}\cdot\mathbf{n}_i}}$$

[Tablellion and Lamorlette 04]

$$w_i^2(\mathbf{p}) = 1-2a\max\left\{\frac{\|\mathbf{p}-\mathbf{p}_i\|}{R_i},\quad 4\sqrt{1-\mathbf{n}\cdot\mathbf{n}_i}\right\}$$

Pascal

$$w_i^3(\mathbf{p}) = \frac{1}{\frac{\|\mathbf{p}-\mathbf{p}_i\|}{R_i} + \sqrt{1-\mathbf{n}\cdot\mathbf{n}_i}} - \frac{1}{a}$$

Moving on to another topic---the weighting function used in the weighted average in irradiance interpolation.

The original weighting function proposed by Ward et al. [1988] has two undesirable properties. First, it goes to infinity when the distance between the point of interpolation **p** and the location of a record, $\mathbf{p}_i$, goes to zero. Second, there is a discontinuity at the border of the influence area of a record. This tends to produce some visible seams in the images. One solution, used in *Radiance*, is to randomize the acceptance of a record for interpolation. In our experience, a better way is to make the weight function zero at the border of the influence area. Two possibilities for this are shown on the slide. The image quality does not depend much on which of them is used.

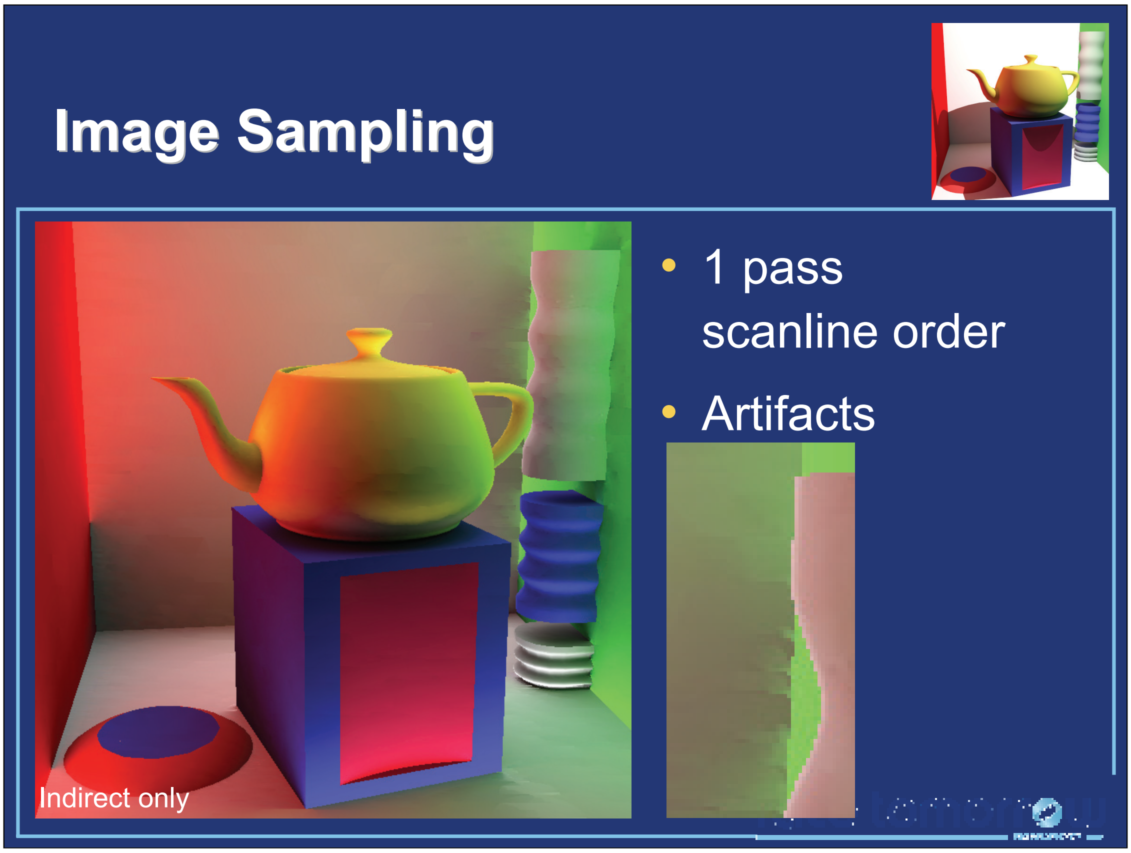


Image Sampling

- 1 pass scanline order
- Artifacts

Lazy evaluation of irradiance values is a great feature of irradiance caching that makes the algorithm very flexible. However, if not used carefully, it has a negative impact on the image quality. This slide shows the kind of artifacts you may expect when generating image pixels in the scanline order, adding new irradiance values lazily as needed.

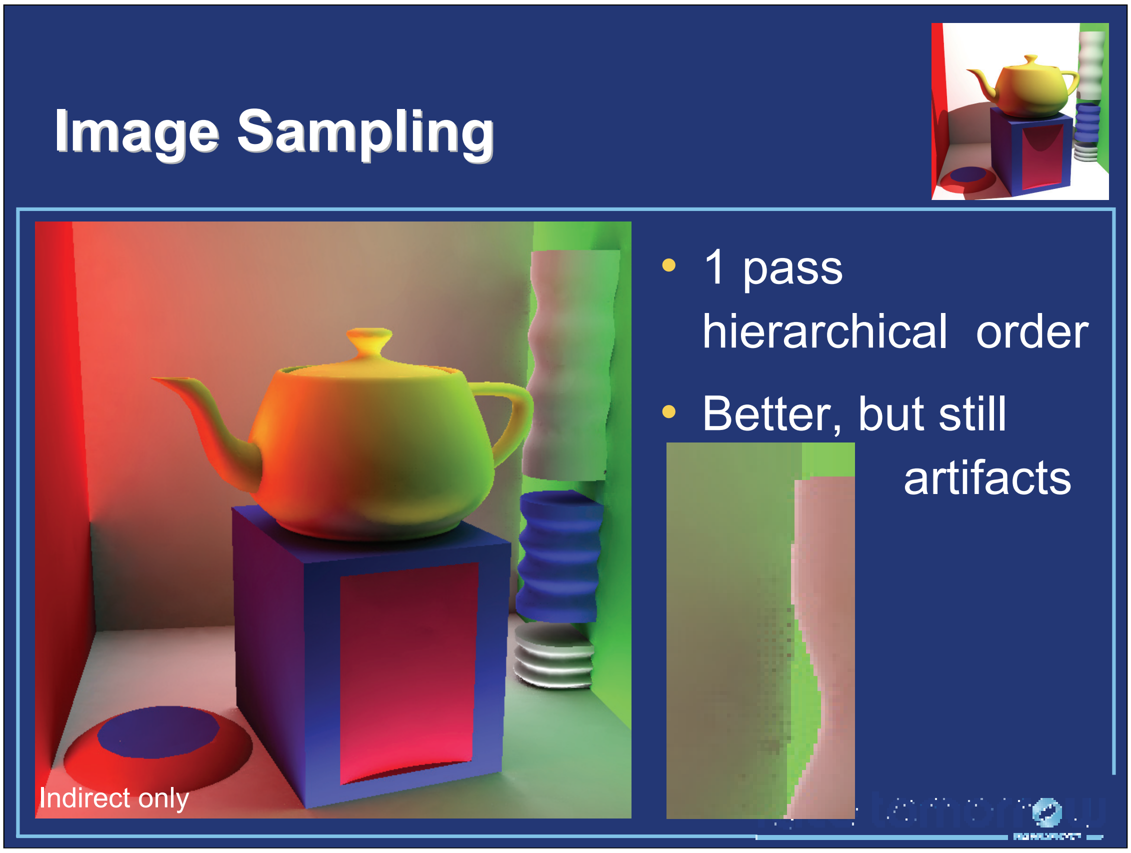


Image Sampling

- 1 pass hierarchical order
- Better, but still artifacts

Using hierarchical image traversal instead of the scanline order improves the image quality (and, actually, decreases the number of records needed to cover the whole image), but image artifacts still remain.

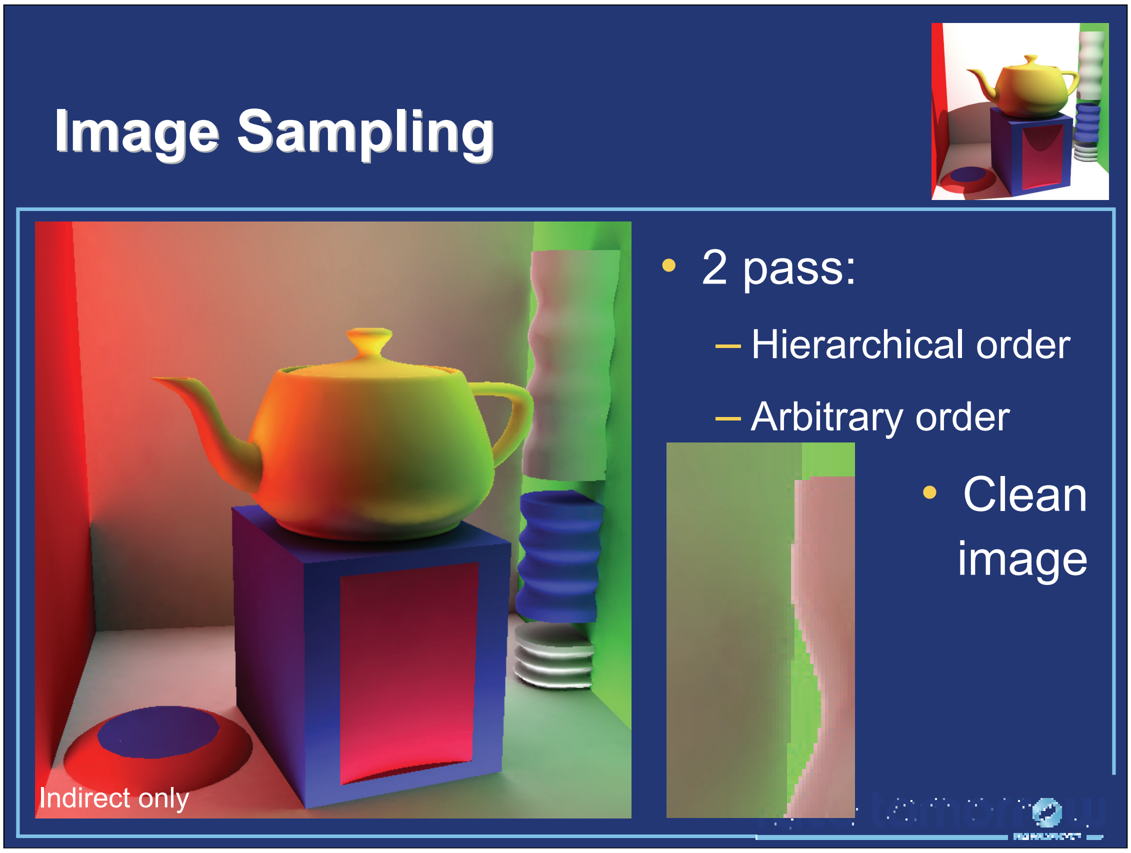
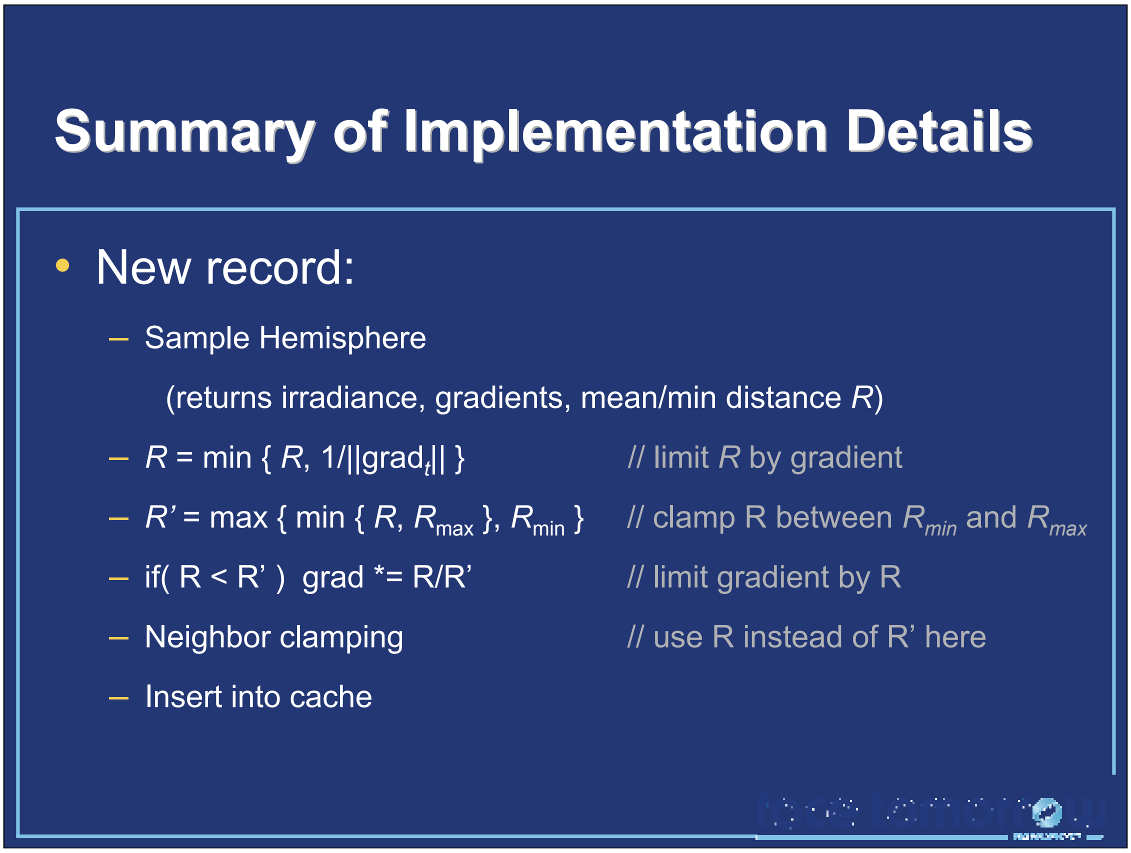


Image Sampling

- 2 pass:
  - Hierarchical order
  - Arbitrary order
- Clean image

Indirect only

In our experience, the best solution is a two pass traversal of the image. In the first pass, the irradiance cache is filled so that all pixels are covered, but no image is generated. In the second pass, arbitrary pixel traversal can then be used to generate the image.



Summary of Implementation Details

- New record:
  - Sample Hemisphere

    (returns irradiance, gradients, mean/min distance R)
  - $R = \min\{R, 1/\|grad_t\|\}$ // limit R by gradient
  - $R' = \max\{\min\{R, R_{\max}\}, R_{\min}\}$ // clamp R between R_{min} and R_{max}
  - if(R < R') grad *= R/R' // limit gradient by R
  - Neighbor clamping // use R instead of R' here
  - Insert into cache

This slide summarizes the actions taken to add a new irradiance value into the cache. First, we sample the hemisphere by casting a number of secondary rays. This gives the irradiance estimate, the translational and the rotational gradients, and the estimate of the mean (or minimum) distance to the neighboring geometry, *R*. We then limit the value of *R* by the translational gradient. After that we clamp *R* by the minimum and maximum threshold (determined from the projected pixel size). This produces the clamped value *R*'. If *R* was increased by this clamping, we then decrease the gradient magnitude accordingly, in order to avoid negative values in extrapolation. The next step is neighbor clamping. For its correct functionality, it is essential to use the original, unclamped value of *R* (not *R*'). Finally, we insert the new record into the cache.

SIGGRAPH2007

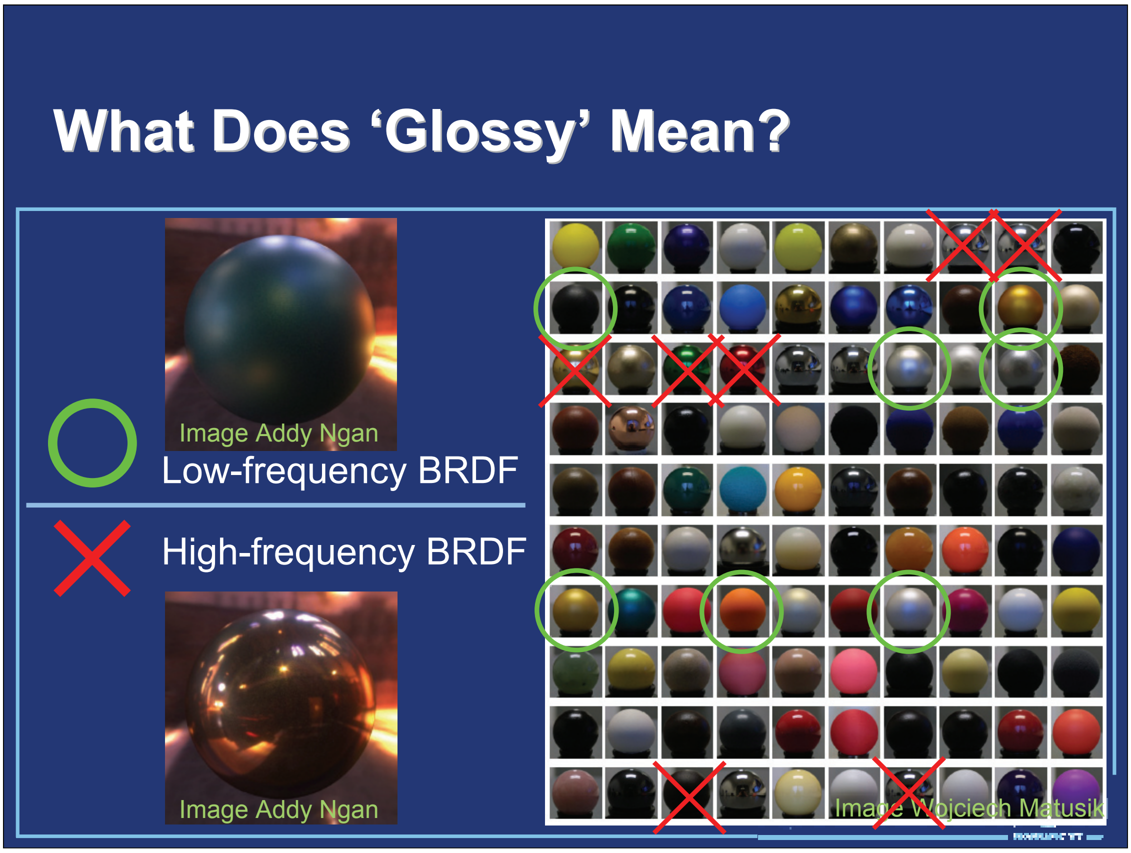
Extension to Glossy Surfaces: Radiance Caching

SIGGRAPH2007

Jaroslav Křivánek

ČVUT v Praze – CTU Prague

The interpolation scheme used in irradiance caching on diffuse surfaces can be also used on glossy surfaces. However, some modifications are necessary due to the view-dependence of glossy surfaces. These modifications will be discussed in this part of the course.



What Does ‘Glossy’ Mean?

Image Addy Ngan

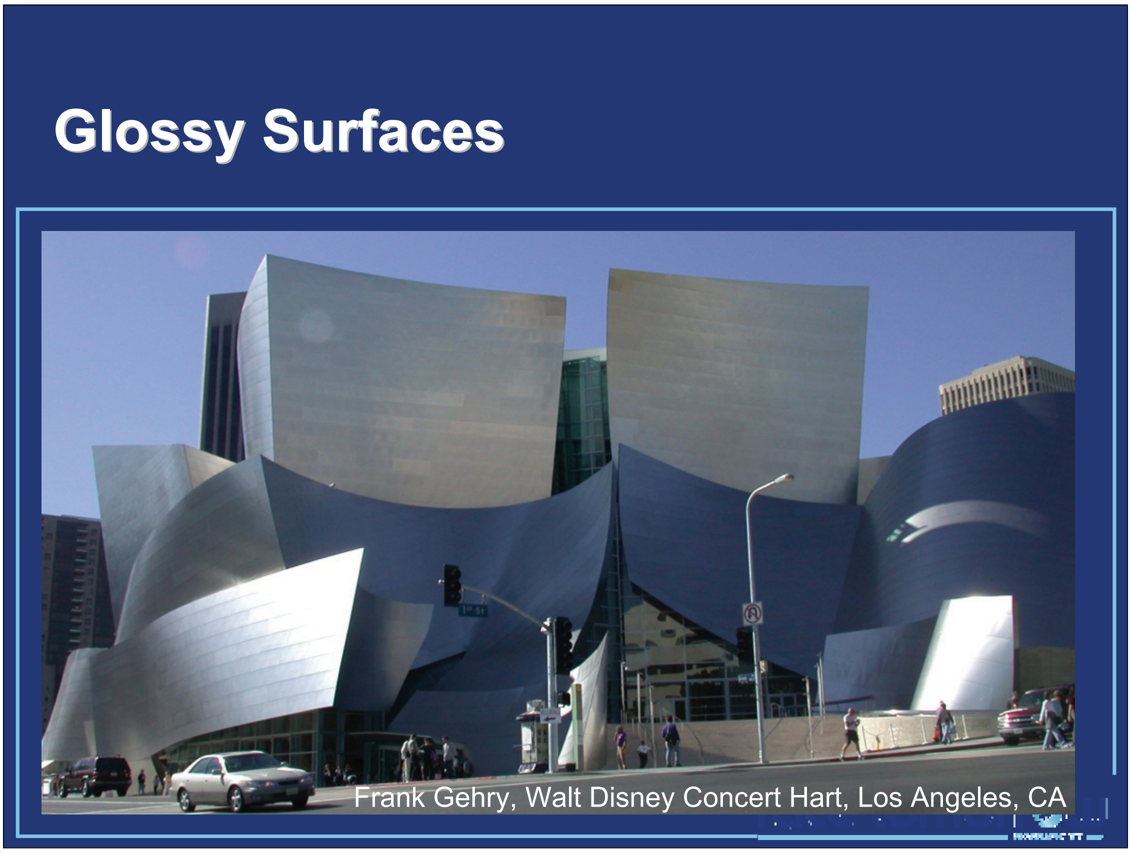
○ Low-frequency BRDF

✕ High-frequency BRDF

Image Addy Ngan

Image Wojciech Matusik

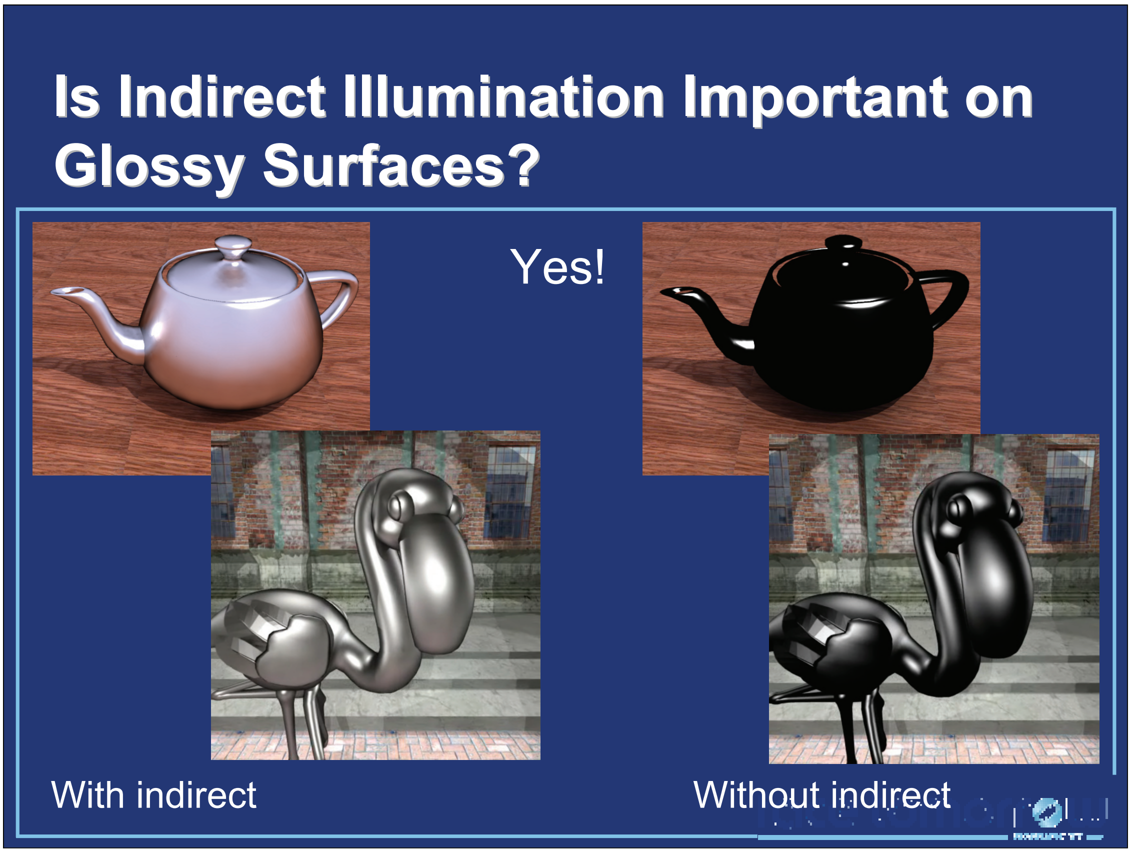
What do we exactly mean by ‘glossy’? We are referring to rough surfaces that reflect their environment, but the reflections are blurry. This is in contrast to ‘specular’ surfaces that show sharp reflections of their environment. Glossy surfaces have ‘low-frequency’ BRDFs while specular surfaces have ‘high-frequency’ BRDFs. Indirect illumination on specular surfaces can be quite efficiently computed by stochastic ray tracing with importance sampling (proportional to the BRDF) using only a couple of secondary rays. In what follows, we will focus on computing indirect illumination on glossy surfaces (with low-frequency BRDFs), for which importance sampling is not very effective.



Glossy Surfaces

Frank Gehry, Walt Disney Concert Hart, Los Angeles, CA

A nice real-world example of glossy surfaces are the walls of Frank Gehry’s Walt Disney Concert Hall in Los Angeles (as well as the walls of Gehry’s Guggenheim Museum in Bilbao, Spain). Other examples of glossy surfaces include rough plastic surfaces or rough pottery.



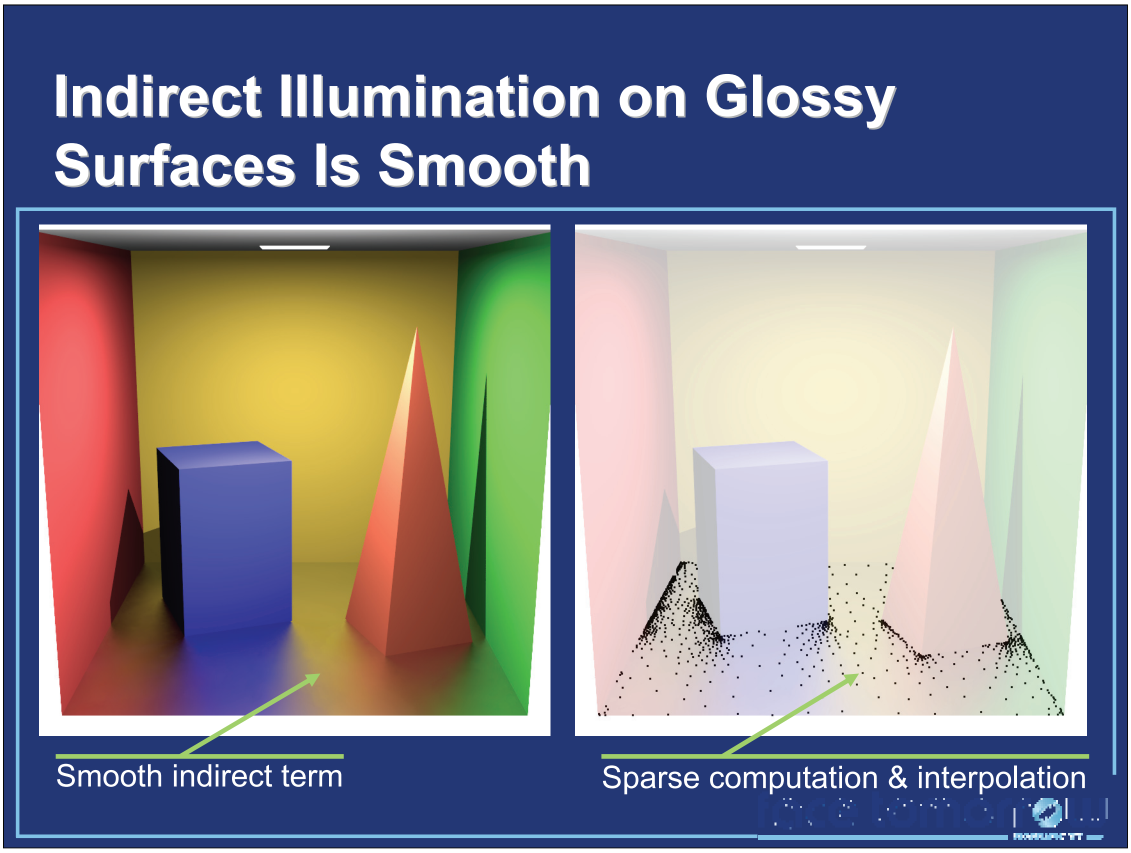
Is Indirect Illumination Important on Glossy Surfaces?

Yes!

With indirect

Without indirect

Why do we bother computing indirect illumination on glossy surfaces? Is it worth the effort? Yes, it certainly is. Actually, Fleming et al. [2003] have shown that correct perception of material properties depend very much on using correct natural illumination. Under a point light, we don't know what an object is made from.

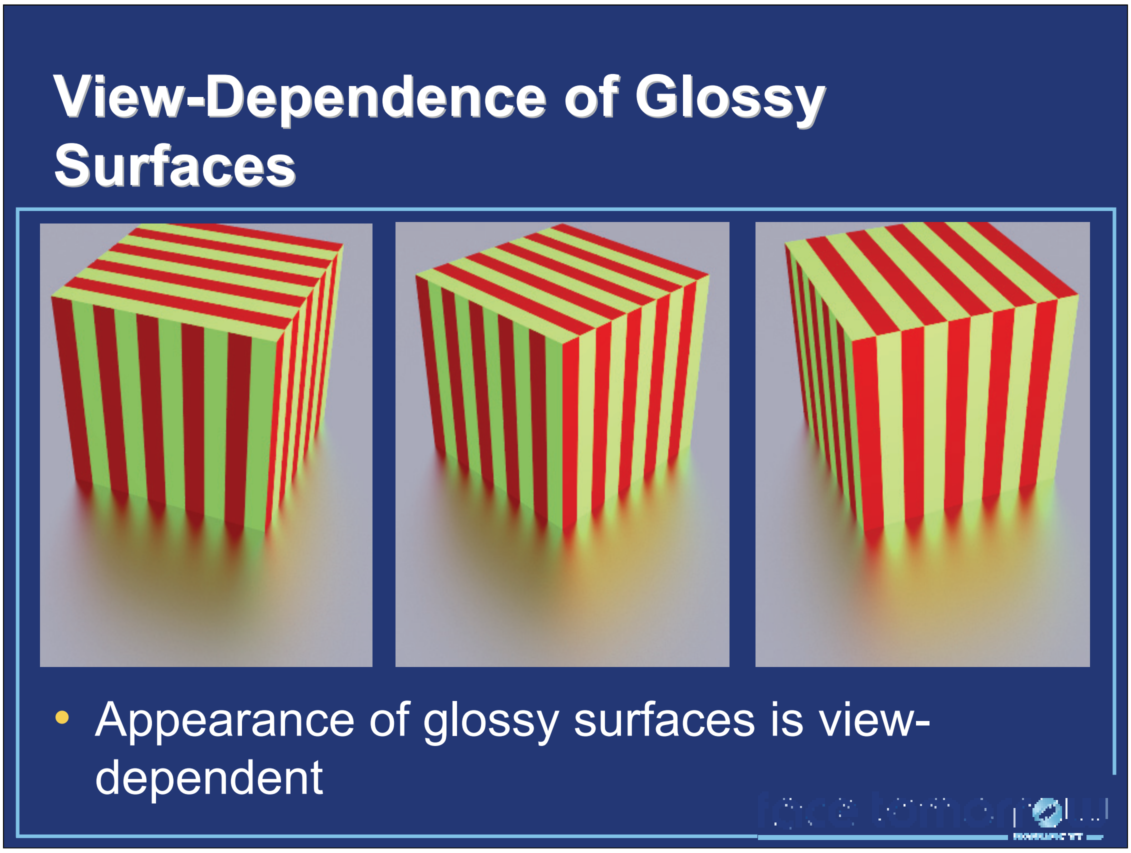


Indirect Illumination on Glossy Surfaces Is Smooth

Smooth indirect term

Sparse computation & interpolation

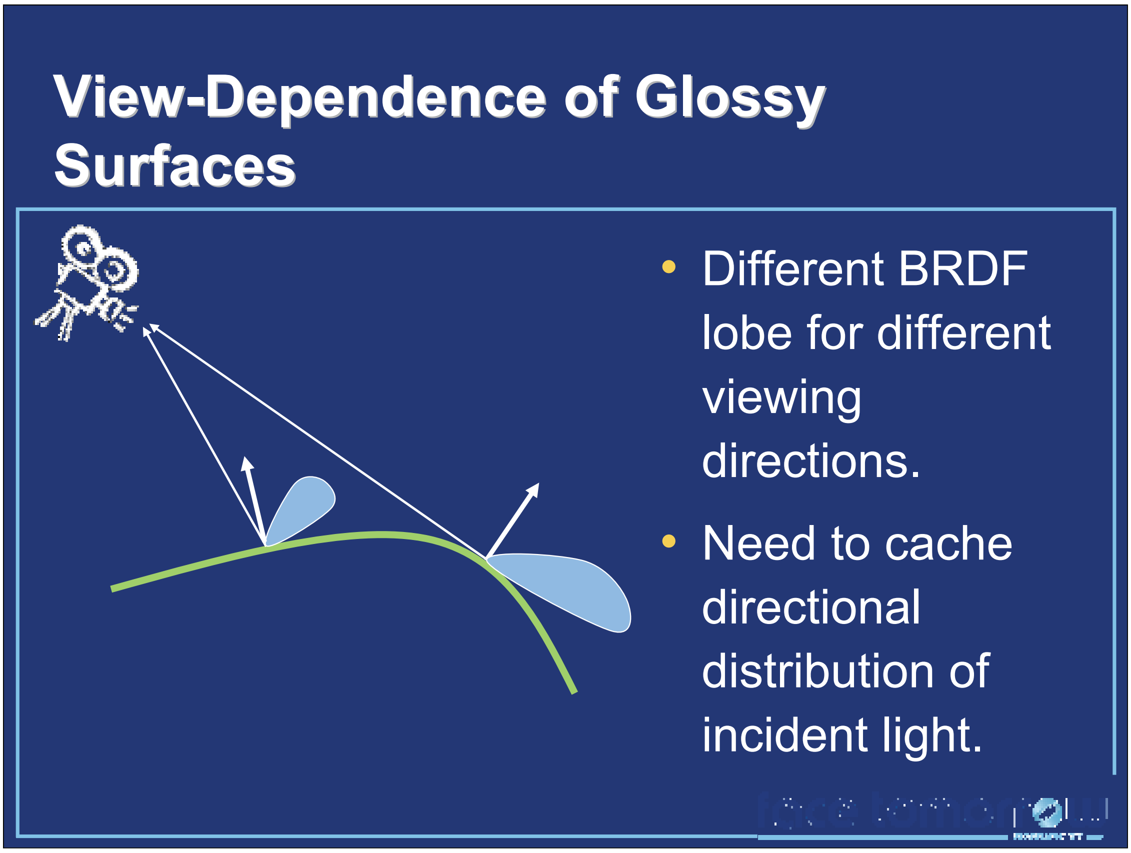
Fortunately, glossy surfaces show a very blurry reflections of their environment. In other words, indirect illumination on glossy surfaces changes slowly over surfaces. This allows to use the same trick as in irradiance caching – compute illumination lazily at sparse locations in the scene and interpolate elsewhere.



View-Dependence of Glossy Surfaces

- Appearance of glossy surfaces is view-dependent

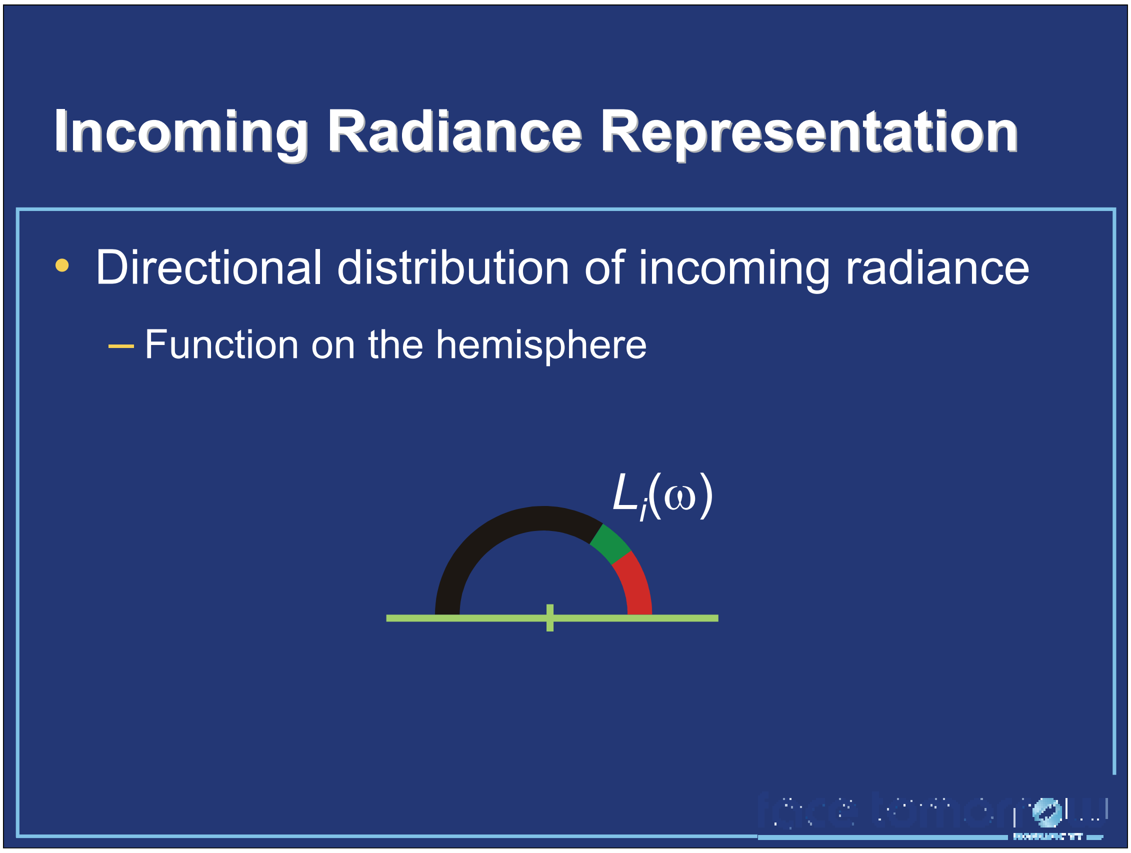
Unlike for diffuse surfaces, appearance of glossy surfaces is view-dependent. In other words, the surface looks different when observed from different viewing angles. This makes interpolation of illumination more complicated than on purely diffuse surfaces.



View-Dependence of Glossy Surfaces

- Different BRDF lobe for different viewing directions.
- Need to cache directional distribution of incident light.

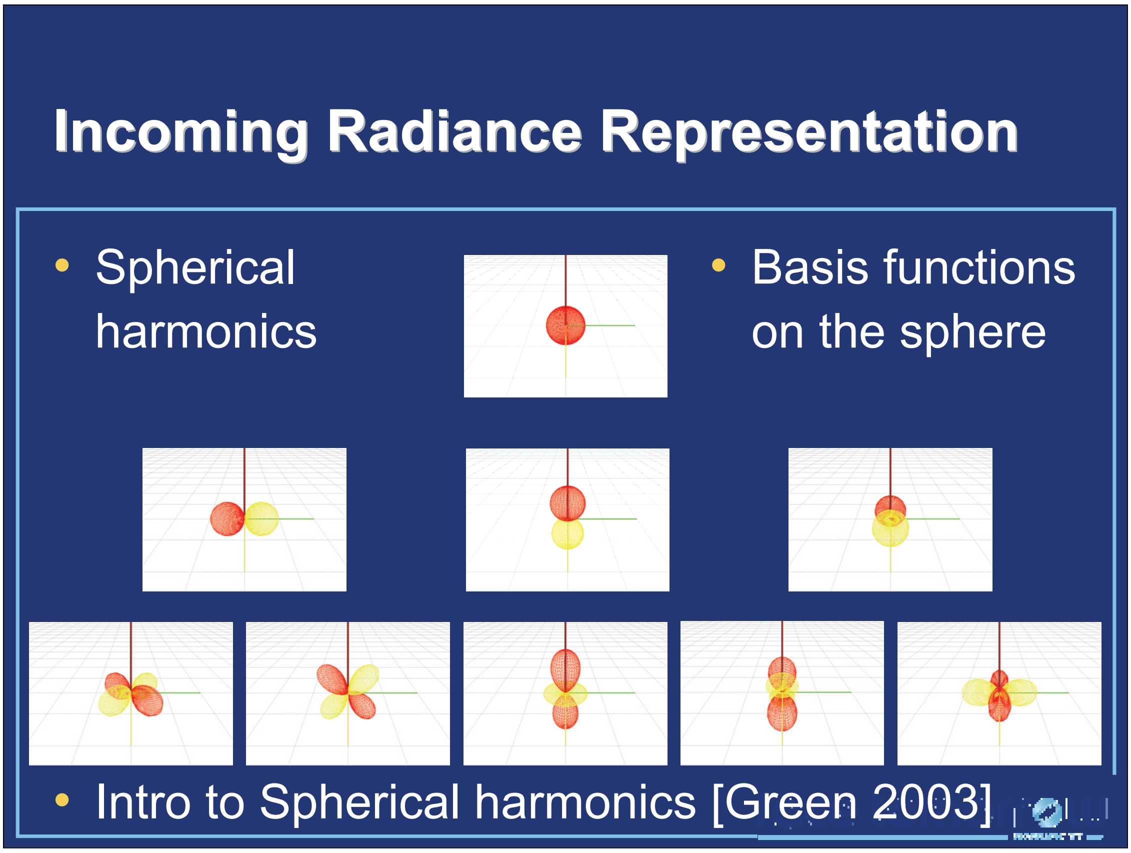
The view-dependence of glossy surfaces is due to the fact that different part of incident light is reflected towards the camera for different viewing directions. So if we want to interpolate illumination on glossy surfaces, we need to cache the full directional distribution of incoming radiance – i.e. we cache a representation of incoming radiance at a point for all possible directions on the hemisphere. Then, at each interpolation point, we ‘extract’ the part of the incoming light which is reflected towards the camera. This ‘extraction’ is done by evaluating the illumination integral, i.e. by integrating incoming light multiplied by the BRDF lobe over the hemisphere. This will be discussed in more detail later.



Incoming Radiance Representation

- Directional distribution of incoming radiance
  - Function on the hemisphere

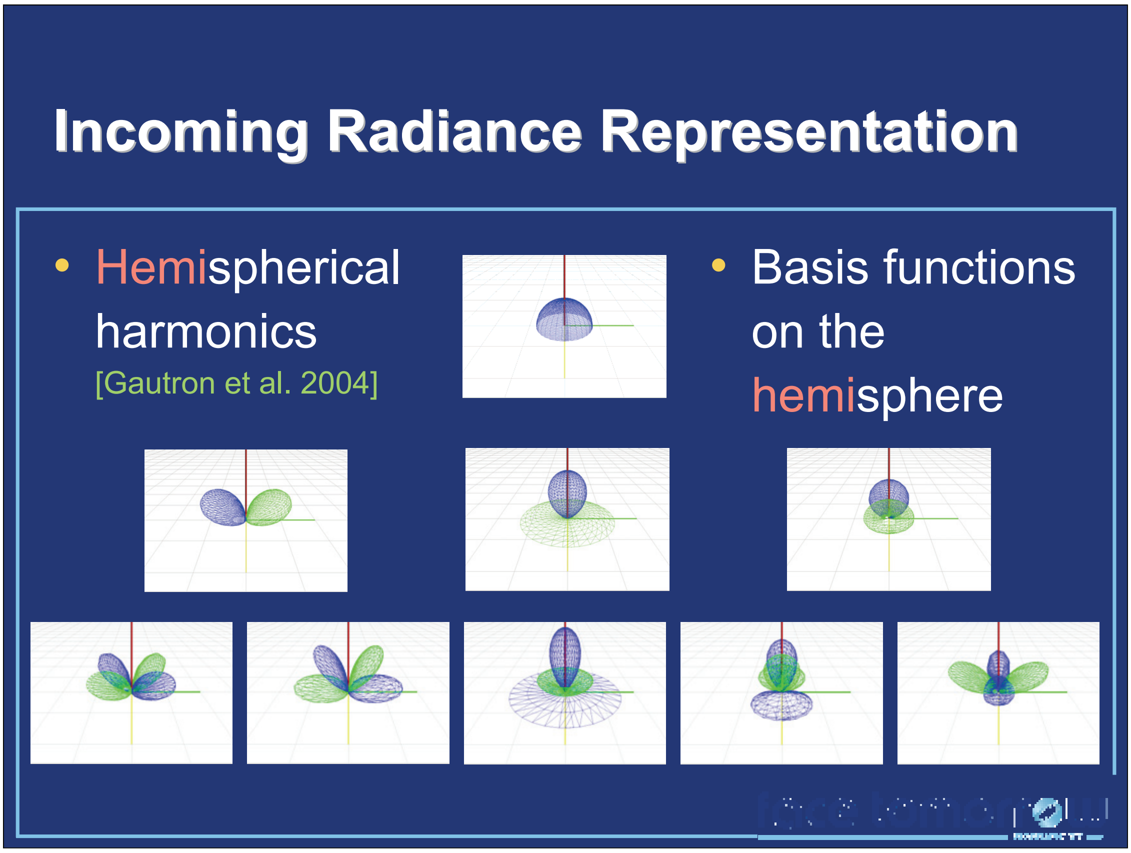
The cached quantity – directional distribution of incoming radiance - is a function defined on the hemisphere. We need to find a suitable representation of this function to make caching possible.



Incoming Radiance Representation

- Spherical harmonics
- Basis functions on the sphere
- Intro to Spherical harmonics [Green 2003]

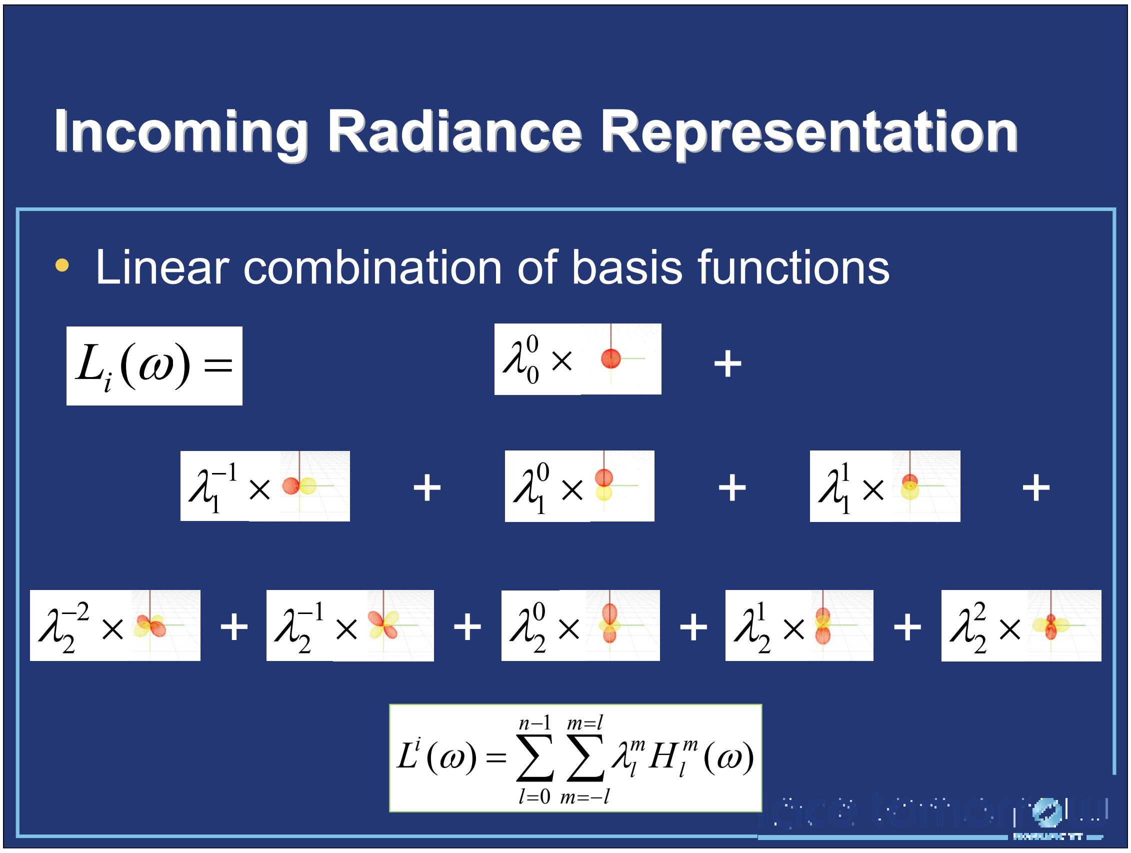
For various reasons, which will become clearer later, we use spherical harmonics to represent the incoming radiance at a point. Spherical harmonics is a set of basis functions defined on the unit sphere.



Incoming Radiance Representation

- Hemispherical harmonics
  [Gautron et al. 2004]
- Basis functions on the hemisphere

Incident radiance at a point on a surface does not span the whole sphere of directions, but only a hemisphere. Therefore, we can use hemispherical harmonics proposed by Gautron et al. [2004] as an alternative to spherical harmonics. This gives us better accuracy with the same number of coefficients.



Incoming Radiance Representation

- Linear combination of basis functions

$$L_i(\omega) = \lambda_0^0 \times \quad +$$

$$\lambda_1^{-1} \times \quad + \quad \lambda_1^0 \times \quad + \quad \lambda_1^1 \times \quad +$$

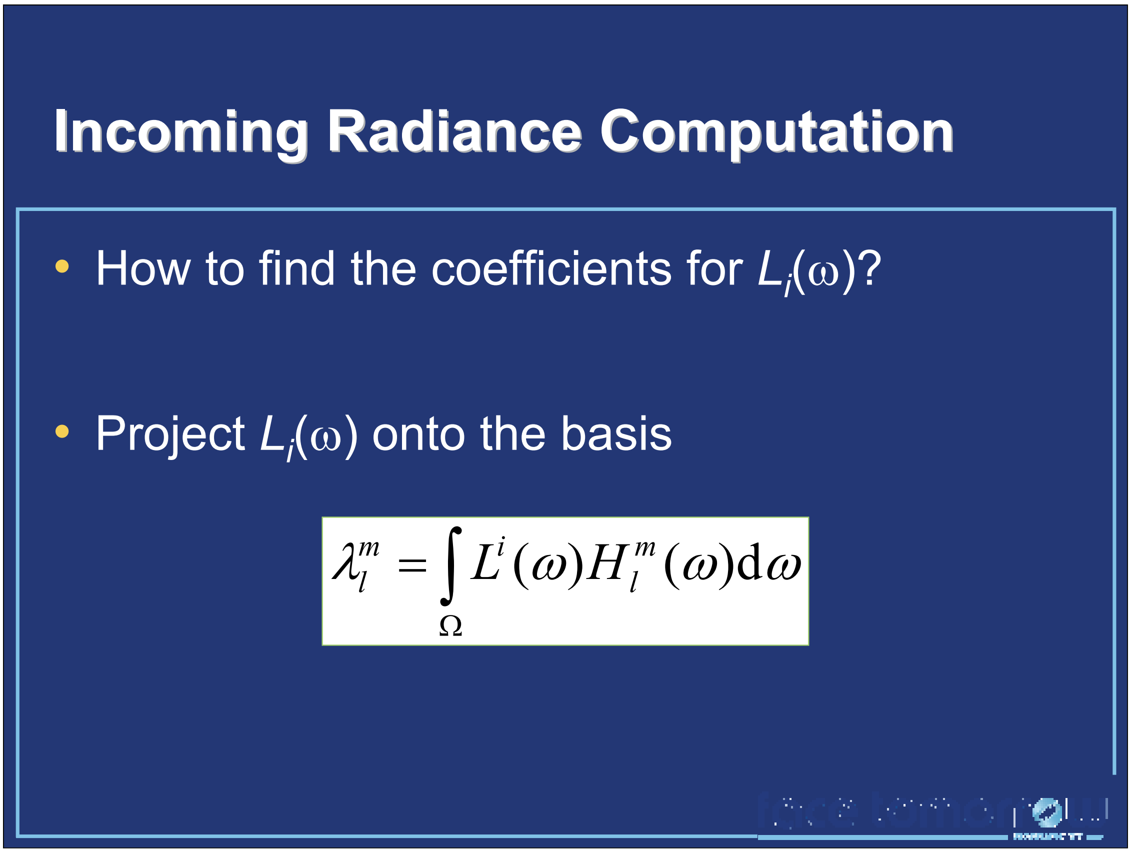
$$\lambda_2^{-2} \times \quad + \quad \lambda_2^{-1} \times \quad + \quad \lambda_2^0 \times \quad + \quad \lambda_2^1 \times \quad + \quad \lambda_2^2 \times$$

$$L^i(\omega) = \sum_{l=0}^{n-1} \sum_{m=-l}^{m=l} \lambda_l^m H_l^m(\omega)$$

Incoming radiance, as well as any other function defined on the hemisphere, can be represented as a linear combination of the basis functions (no matter whether we use spherical or hemispherical harmonics). The coefficients in this linear combination make up the representation of our function.

For spherical harmonics, the coefficients are indexed by two indices, *l* (lower index) and *m* (upper index). The *l*-index runs from 0 to *n*-1, where *n* is the order of the spherical harmonics representation. The higher the order, the better (more accurate) representation. Spherical harmonics with the same *l*-index form a band (a row on the slide above). Within one band, the *m* index goes from –*l* to +*l*. So there is one harmonic in the first band, three harmonics in the second band, five in the third band etc. There are n^2 coefficients in total for an order-*n* representation. The double sum in the formula on the slide simply stands for summing over all harmonics up to order *n*. It could also be expressed as a single sum for *i* from 0 to n^2-1, with *i* given by : *i* = *l*(*l*+1) + *m*.

To summarize, an order-*n* representation of a function consists of n^2 coefficients, which is what we to store in the cache. (Actually, we store one coefficient vector for each color component). Typically, we use order up to 10 in radiance caching, corresponding to 100 coefficients.

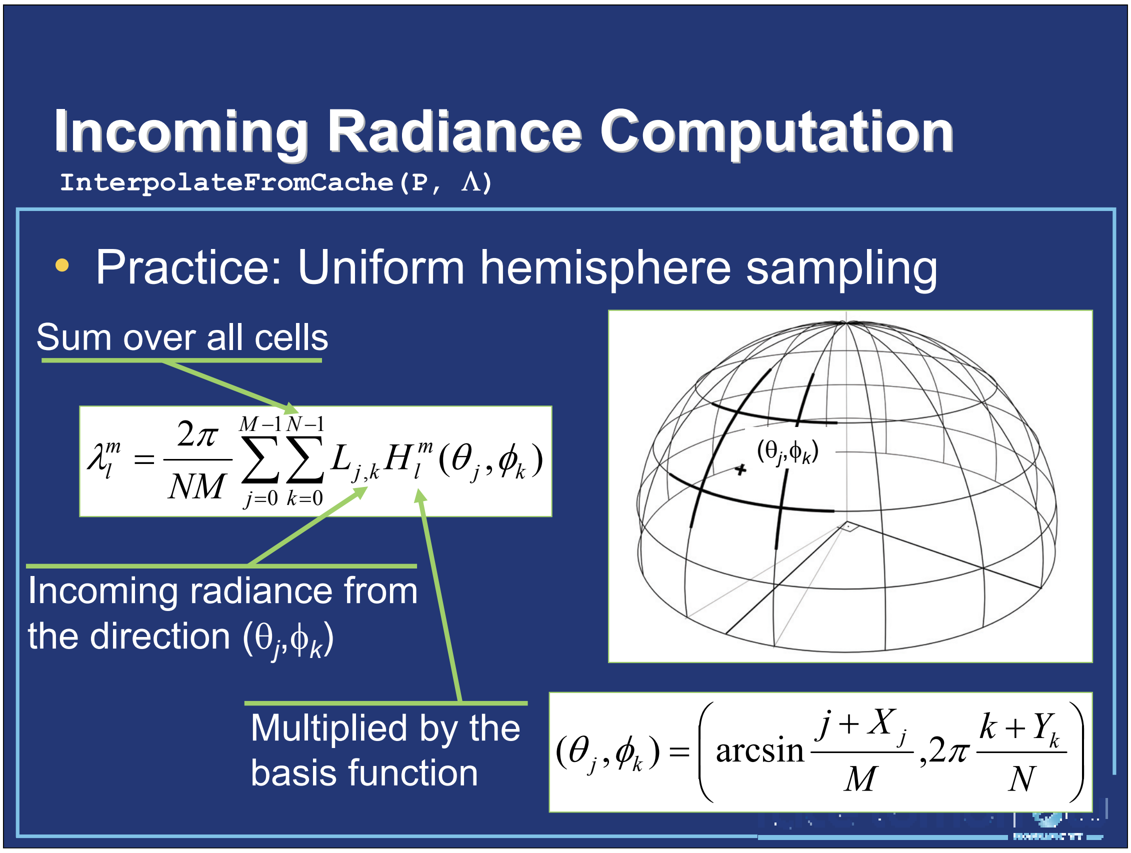


Incoming Radiance Computation

- How to find the coefficients for $L_i(\omega)$?
- Project $L_i(\omega)$ onto the basis

$$\lambda_l^m = \int_\Omega L^i(\omega) H_l^m(\omega) \mathrm{d}\omega$$

The coefficients representing a function are found by *projection*. For our purposes, it is sufficient to say that the projection is computed by integrating the product of the function being projected with the basis function, as shown on the slide.



Incoming Radiance Computation

`InterpolateFromCache(P, Λ)`

- Practice: Uniform hemisphere sampling

Sum over all cells

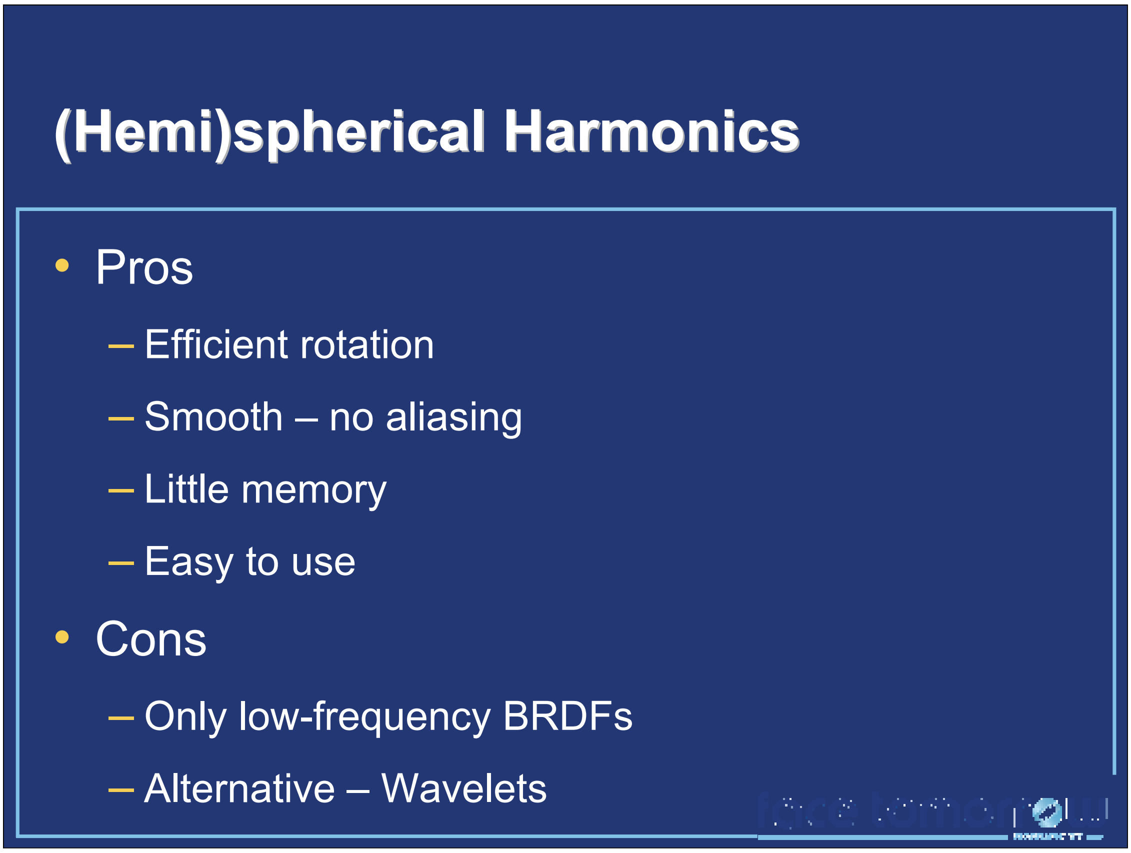
$$\lambda_l^m = \frac{2\pi}{NM}\sum_{j=0}^{M-1}\sum_{k=0}^{N-1} L_{j,k} H_l^m(\theta_j,\phi_k)$$

Incoming radiance from the direction (θ_j,ϕ_k)

Multiplied by the basis function

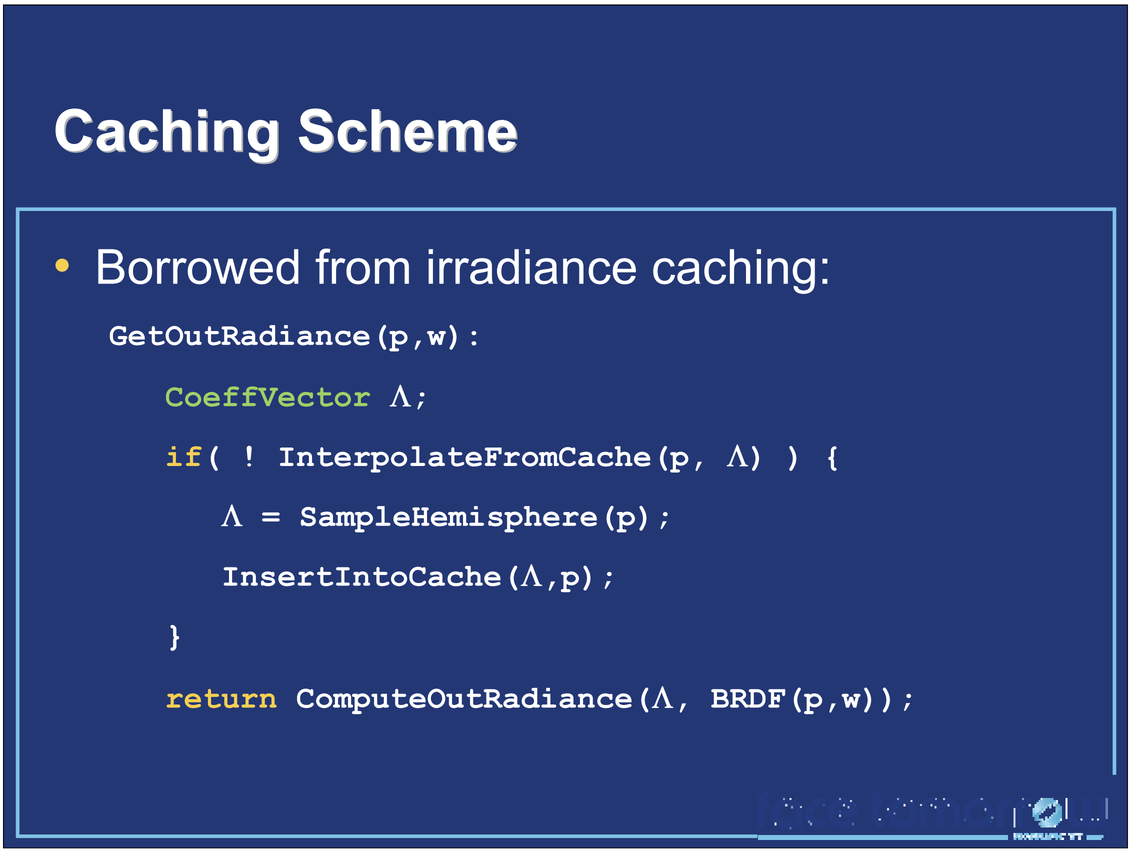
$$(\theta_j,\phi_k) = \left(\arcsin\frac{j+X_j}{M}, 2\pi\frac{k+Y_k}{N}\right)$$

To find the coefficients for the incoming radiance at a point, we need to numerically estimate the projection integral. Since our only means to evaluate the incoming radiance is point sampling – i.e. casting secondary rays – the obvious choice is Monte Carlo quadrature. We divide the hemisphere *uniformly* into *M* x *N* cells, where *M.N* is the desired number of rays (we use 500 – 8000 our renderings), and *N* ≈ 4*M*. For each cell [j,k] we pick a random direction using the formula above. Note that this is slightly different from irradiance caching, since we are using uniform sampling (as opposed to cosine-proportional sampling in irradiance caching). For each sampling direction, we evaluate the basis functions, multiply them by the incoming radiance from that direction and accumulate to the coefficient vector.



(Hemi)spherical Harmonics

- Pros
  - Efficient rotation
  - Smooth – no aliasing
  - Little memory
  - Easy to use
- Cons
  - Only low-frequency BRDFs
  - Alternative – Wavelets

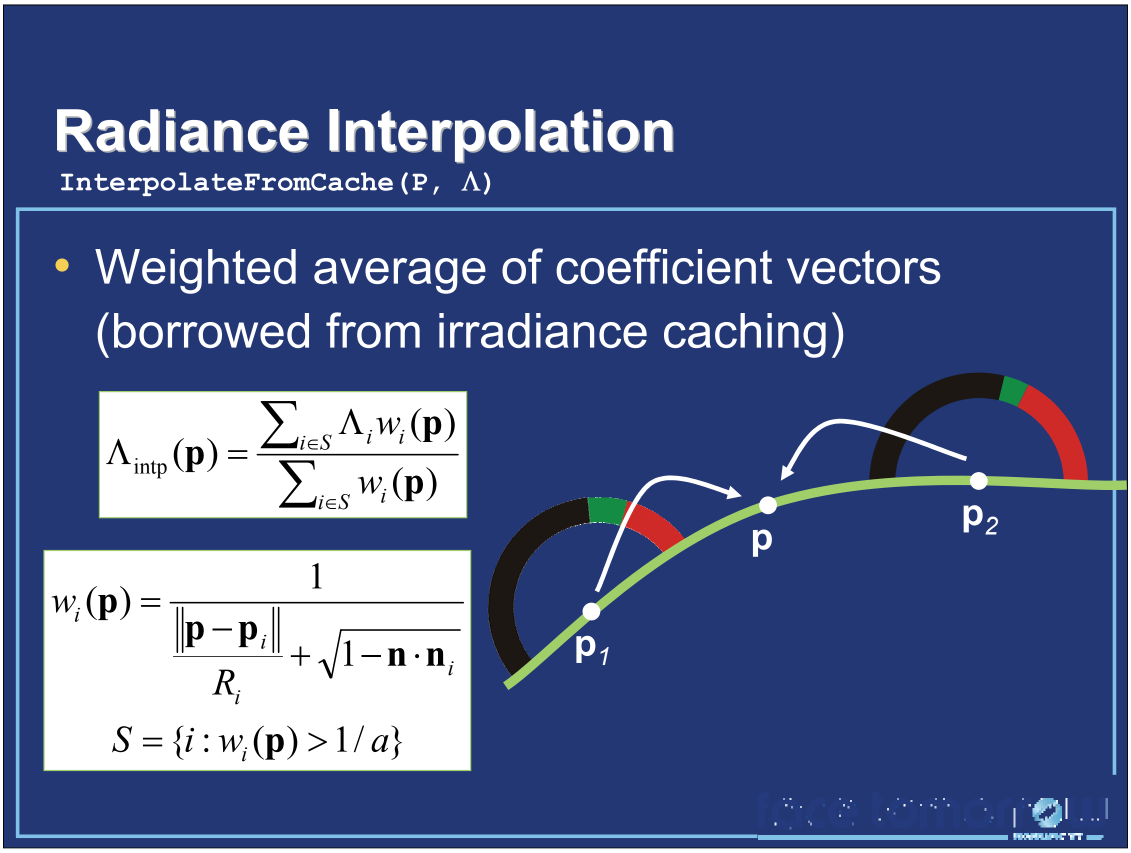


Caching Scheme

- Borrowed from irradiance caching:

```
GetOutRadiance(p,w):
    CoeffVector Λ;
    if( ! InterpolateFromCache(p, Λ) ) {
        Λ = SampleHemisphere(p);
        InsertIntoCache(Λ,p);
    }
    return ComputeOutRadiance(Λ, BRDF(p,w));
```

The overall caching scheme on glossy surfaces is the same as on diffuse ones. There are some important differences, though, that have to be taken care of. The procedure to compute outgoing radiance at a point due to glossy indirect illumination in a given (viewing) direction proceeds as shown on the slide. First, we query the radiance cache for previously computed records available for interpolation. If some are available, the coefficients for incoming radiance are interpolated from the cache. Otherwise, they are computed using hemisphere sampling as described on previous slides and stored in the cache. Finally, the incoming radiance is integrated against the BRDF to get the outgoing radiance. Various sub-routines of this procedure will be discussed on the following slides.



Radiance Interpolation

`InterpolateFromCache(P, Λ)`

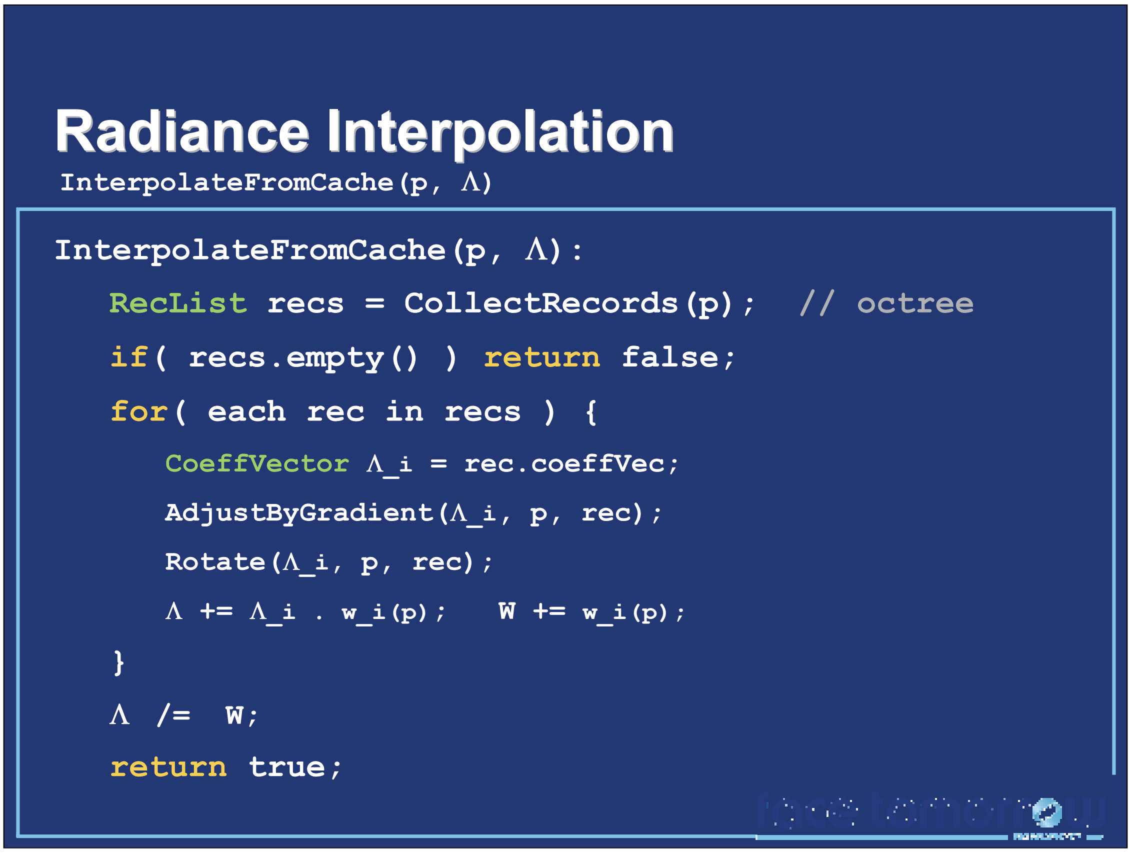
- Weighted average of coefficient vectors (borrowed from irradiance caching)

$$\Lambda_{\text{intp}}(\mathbf{p}) = \frac{\sum_{i \in S} \Lambda_i w_i(\mathbf{p})}{\sum_{i \in S} w_i(\mathbf{p})}$$

$$w_i(\mathbf{p}) = \frac{1}{\frac{\|\mathbf{p}-\mathbf{p}_i\|}{R_i} + \sqrt{1-\mathbf{n}\cdot\mathbf{n}_i}}$$

$$S = \{i : w_i(\mathbf{p}) > 1/a\}$$

To get the coefficient of interpolated incoming radiance at a point, we use the same weighted sum as used in irradiance caching. Instead of interpolating cached irradiance values, we interpolate the coefficient vectors. However the coefficient vectors have to be adjusted by translational gradients and by rotation as described later.



Radiance Interpolation

InterpolateFromCache(p, Λ)

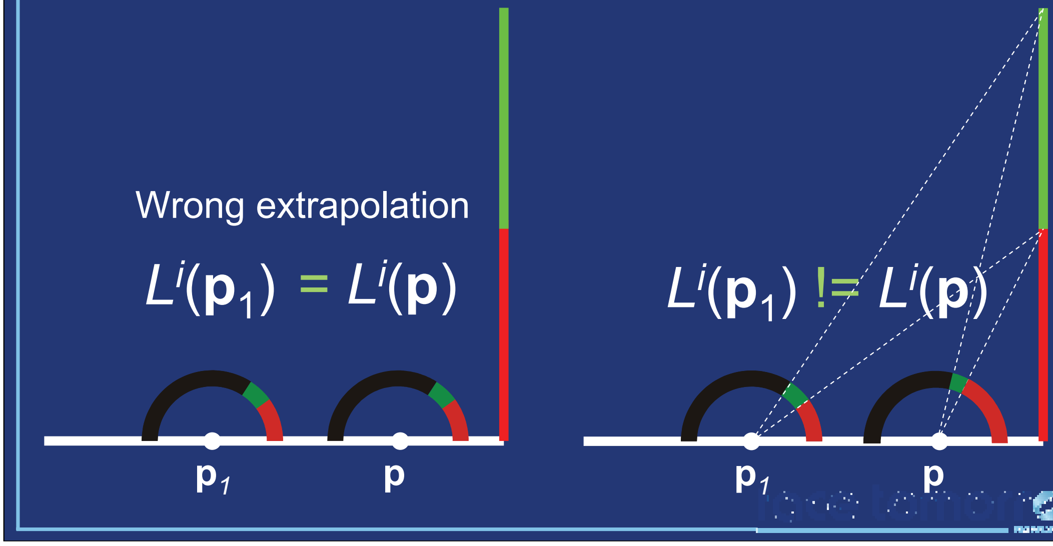
```
InterpolateFromCache(p, Λ):
   RecList recs = CollectRecords(p);   // octree
   if( recs.empty() ) return false;
   for( each rec in recs ) {
      CoeffVector Λ_i = rec.coeffVec;
      AdjustByGradient(Λ_i, p, rec);
      Rotate(Λ_i, p, rec);
      Λ += Λ_i . w_i(p);    W += w_i(p);
   }
   Λ /=  W;
   return true;
```

The radiance interpolation procedure starts by locating the existing records available for interpolation at **p**. This is uses the same octree structure as the irradiance cache. Then we loop over the records available for interpolation. For each record, we first adjust its coefficient vector by the translational gradient. The gradients are computed during hemisphere sampling and stored with the records in the cache. After the gradient adjustment, we apply a rotation to align coordinate frames at the point of interpolation and the location of the record.

Translational Gradients

With radiance caching

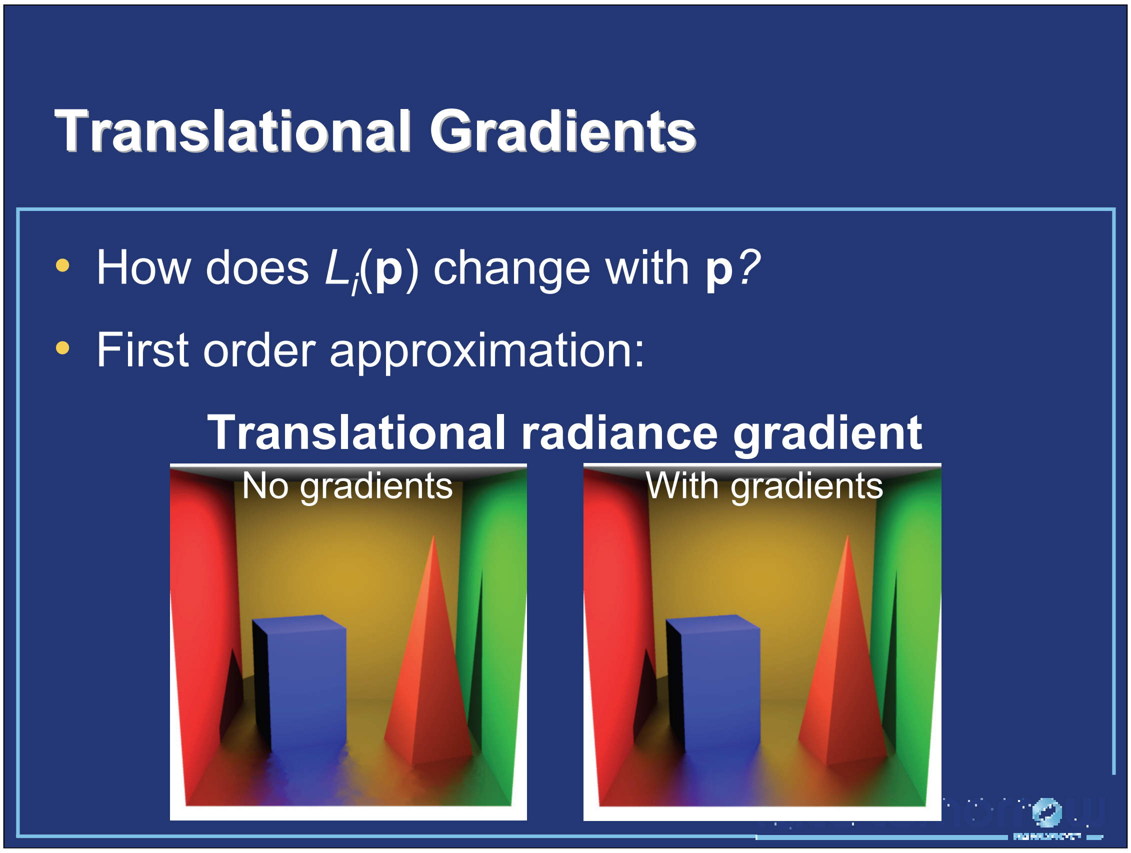
Reality



Wrong extrapolation

$L^i(\mathbf{p}_1) = L^i(\mathbf{p})$

$L^i(\mathbf{p}_1) \mathrel{!=} L^i(\mathbf{p})$



Translational Gradients

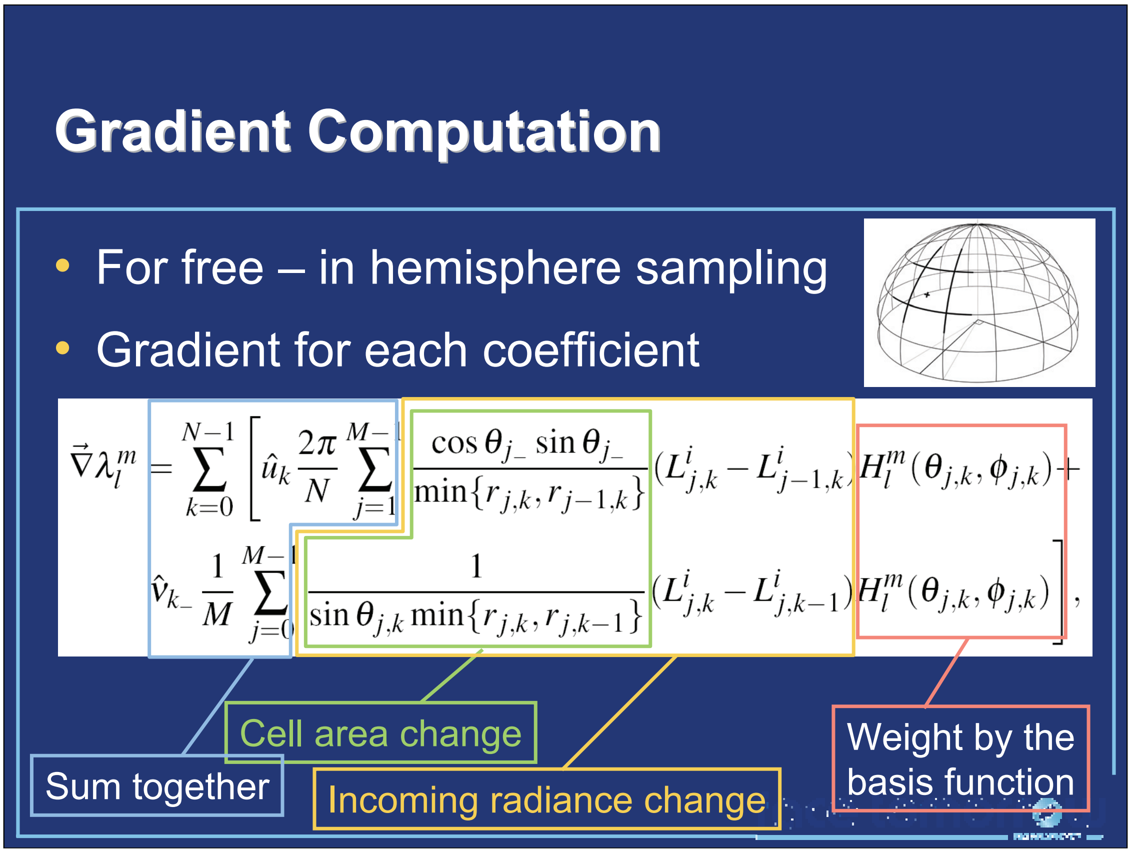
- How does $L_i(\mathbf{p})$ change with $\mathbf{p}$?
- First order approximation:

Translational radiance gradient

No gradients

With gradients

Translational gradients are used in interpolation to compensate for the change of incoming radiance with translation. If radiance is interpolated without the use of gradients, we see undesirable discontinuities in the images. The discontinuities are suppressed by the use of gradients.



Gradient Computation

- For free – in hemisphere sampling
- Gradient for each coefficient

$$\vec{\nabla}\lambda_l^m = \sum_{k=0}^{N-1}\left[\hat{u}_k \frac{2\pi}{N}\sum_{j=1}^{M-1}\frac{\cos\theta_{j\_}\sin\theta_{j\_}}{\min\{r_{j,k}, r_{j-1,k}\}}\left(L_{j,k}^i - L_{j-1,k}^i\right)H_l^m(\theta_{j,k},\phi_{j,k}) + \hat{v}_{k\_}\frac{1}{M}\sum_{j=0}^{M-1}\frac{1}{\sin\theta_{j,k}\min\{r_{j,k}, r_{j,k-1}\}}\left(L_{j,k}^i - L_{j,k-1}^i\right)H_l^m(\theta_{j,k},\phi_{j,k})\right],$$

Sum together

Cell area change

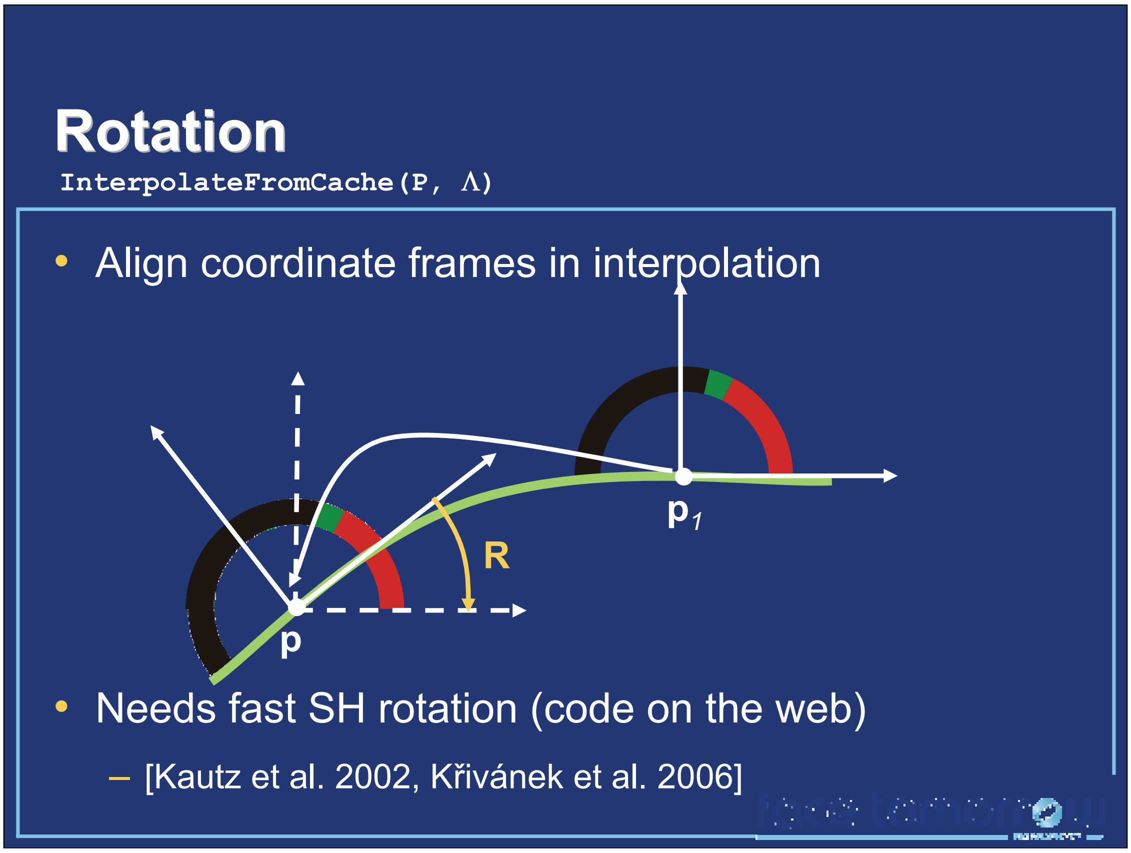
Incoming radiance change

Weight by the basis function

The gradients represent a first-order approximation of the change of incoming radiance with translation. For each coefficient of incoming radiance, there is one translational gradient, which describes how this coefficient changes with translation. Gradients are computed in hemisphere sampling along with the coefficient vectors using a formula shown above, and stored in the record.

To save space, we store the gradients in form of two coefficient derivative vectors – with respect to *x* and *y* axes of the local coordinate frame at the record. We disregard the gradient with respect to the local *z* axis, since the displacements along *z* are small in the interpolation. The *xy* plane is the local tangent plane and *z* is the normal vector.

The gradients are then used in interpolation to adjust the coefficient vector stored in the record.



Rotation

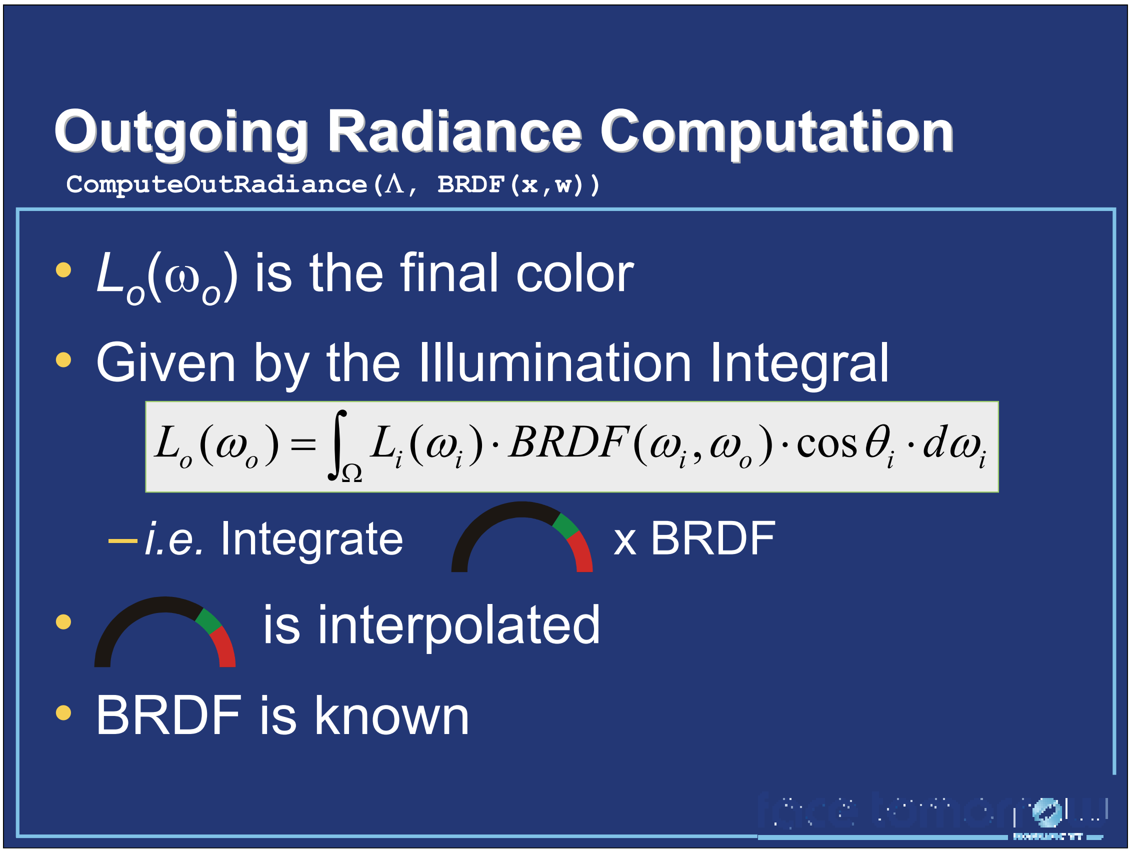
`InterpolateFromCache(P, Λ)`

- Align coordinate frames in interpolation

- Needs fast SH rotation (code on the web)
  - [Kautz et al. 2002, Křivánek et al. 2006]

After the gradient adjustment, the interpolation procedure proceeds by applying a rotation. This is required to align the coordinate frame at the point of interpolation with the coordinate frame with respect to which the coefficients of the incoming radiance were computed. The rotation procedure takes a vector of spherical harmonic coefficients for the incoming radiance and a 3x3 rotation matrix, and outputs a coefficient vector representing the rotated incoming radiance. This rotation procedure is the bottleneck of in the radiance interpolation and hence it is essential to have an optimized procedure for this. We won't go into detail here and rather refer to the radiance caching web page where the rotation code can be downloaded.

This completes the interpolation procedure.



Outgoing Radiance Computation

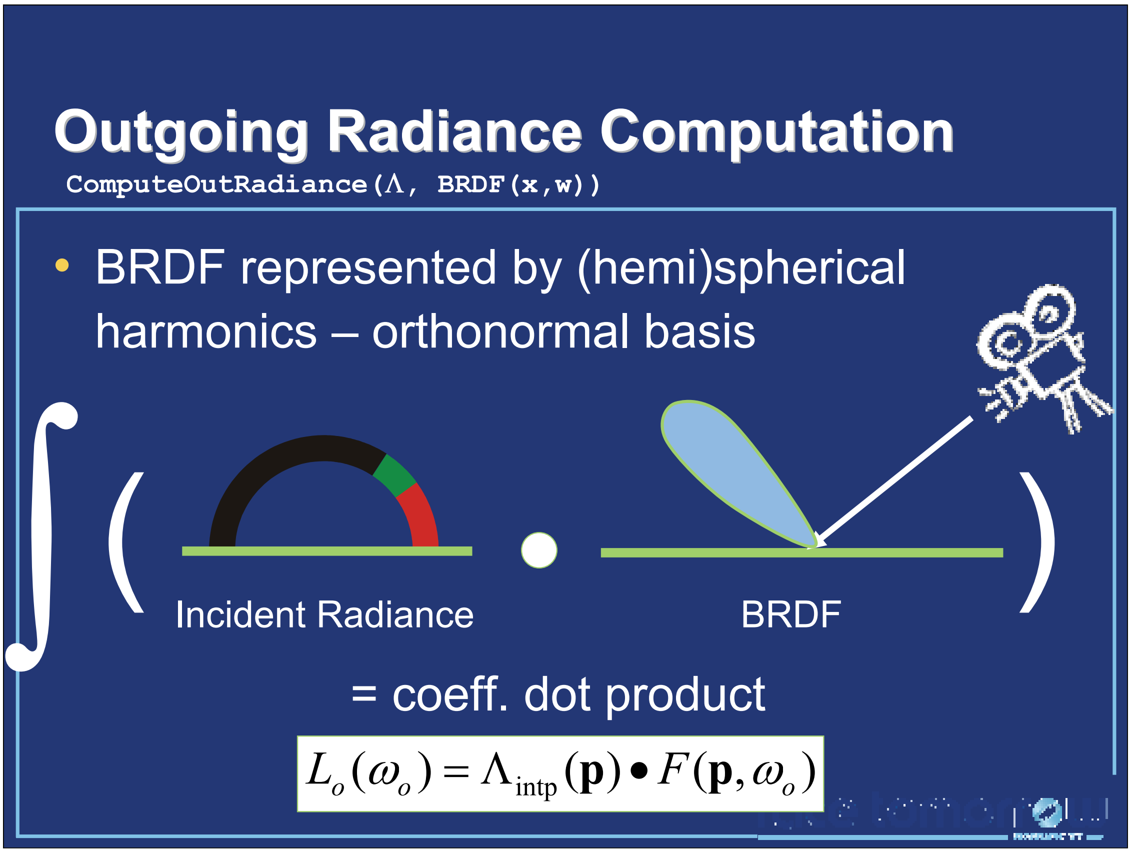
`ComputeOutRadiance(Λ, BRDF(x,w))`

- $L_o(\omega_o)$ is the final color
- Given by the Illumination Integral

$$L_o(\omega_o) = \int_\Omega L_i(\omega_i) \cdot BRDF(\omega_i, \omega_o) \cdot \cos\theta_i \cdot d\omega_i$$

  - *i.e.* Integrate x BRDF
- is interpolated
- BRDF is known

The final result we want to compute is the outgoing radiance at a point for a given outgoing direction – this is the color that corresponds to glossy indirect illumination. So far, we know how to compute or possibly interpolate the coefficient vector representing the directional distribution of incoming radiance at a point. Now we turn this into the outgoing radiance by evaluating the illumination integral, shown on the slide. Since the incoming radiance is interpolated and the BRDF can be looked up from the scene database, all the information required to evaluate the illumination integral is readily available.



Outgoing Radiance Computation

`ComputeOutRadiance(Λ, BRDF(x,w))`

- BRDF represented by (hemi)spherical harmonics – orthonormal basis

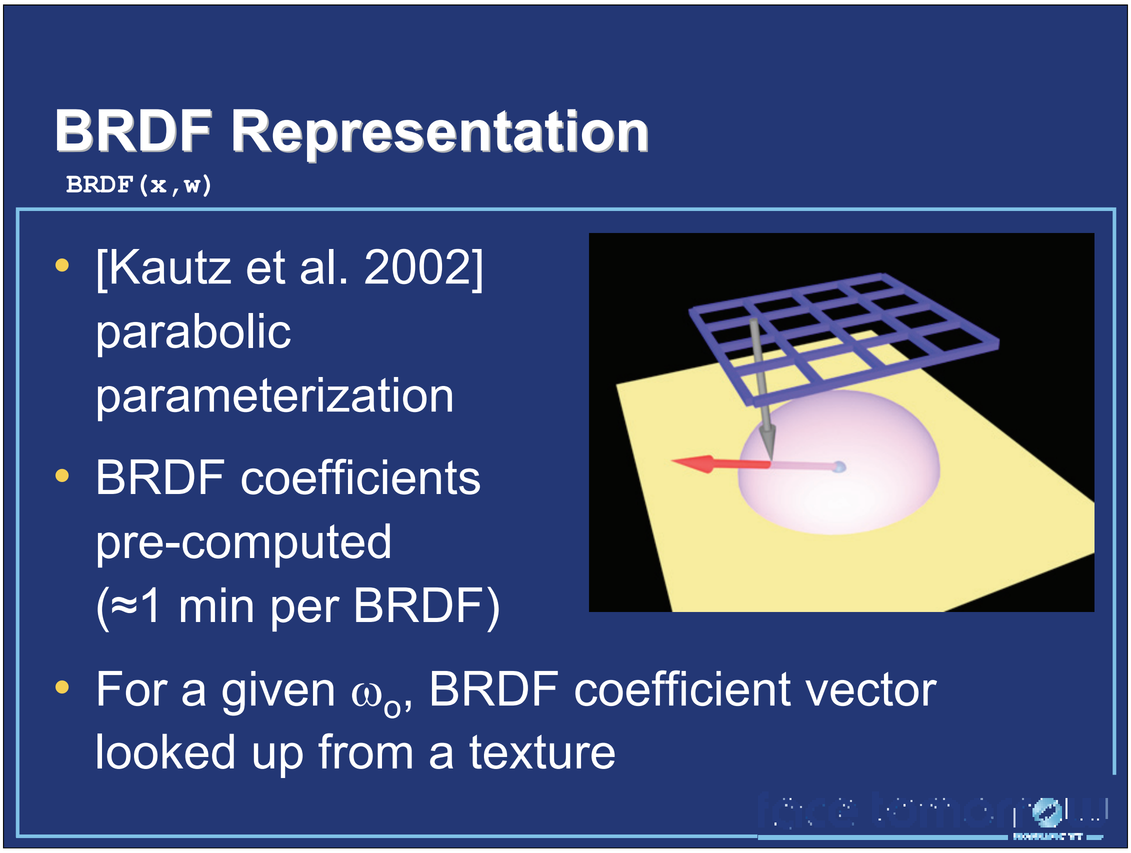
Incident Radiance

BRDF

= coeff. dot product

$$L_o(\omega_o) = \Lambda_{\text{intp}}(\mathbf{p}) \bullet F(\mathbf{p}, \omega_o)$$

To make the evaluation of the illumination integral fast, we take advantage of the dot-product property of orthonormal bases (which spherical and hemispherical harmonics are): Integral of the product of two functions can be computed as the dot product of the coefficient vectors of these function with respect to the basis. So if we represent the BRDF times the cosine term using (hemi)spherical harmonics, all we need to do in order to compute the outgoing radiance is to dot the BRDF coefficients with the incoming radiance coefficients.

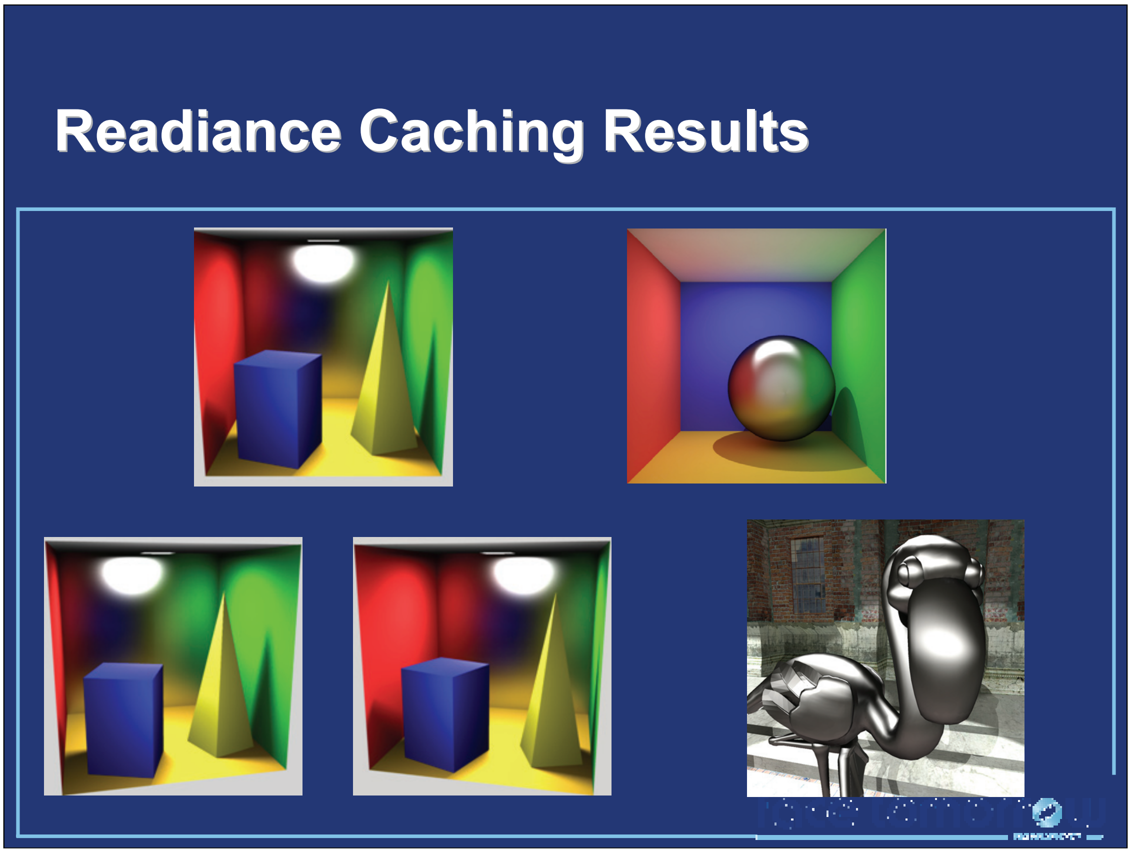


BRDF Representation

```
BRDF(x,w)
```

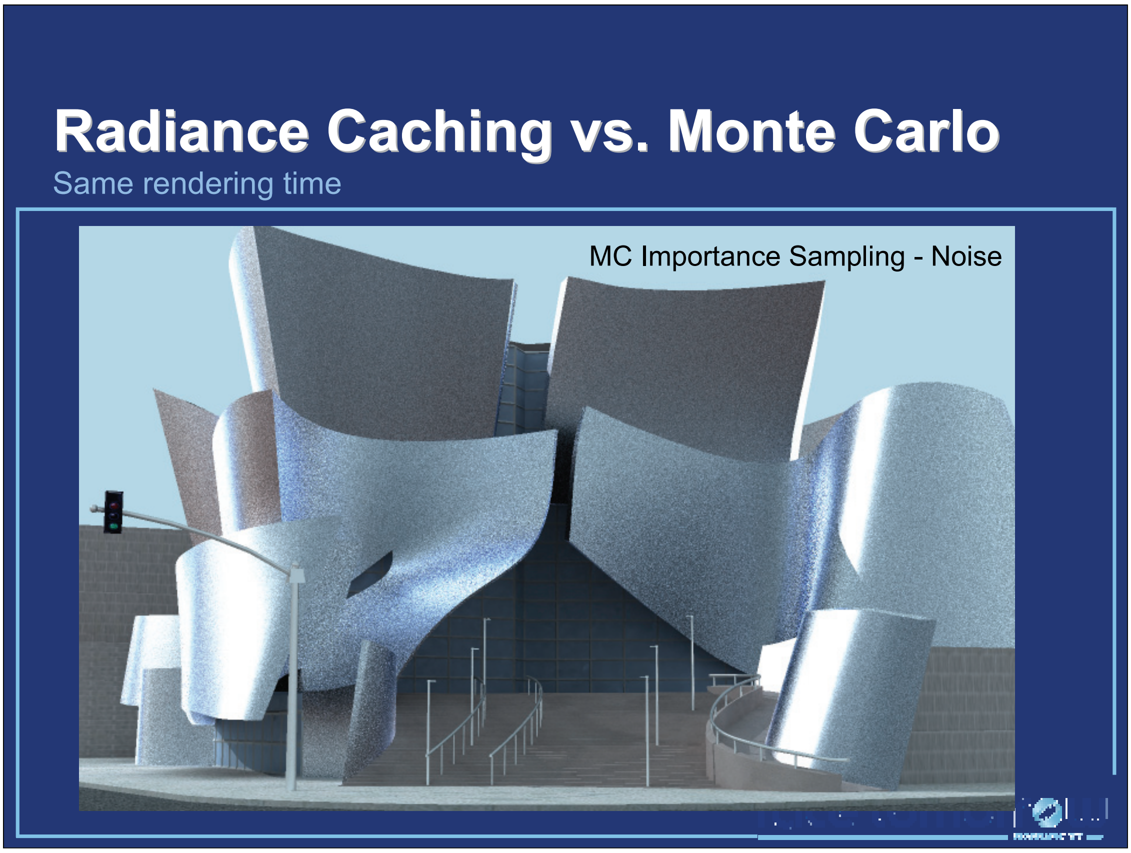
- [Kautz et al. 2002] parabolic parameterization
- BRDF coefficients pre-computed (≈1 min per BRDF)
- For a given ω_o, BRDF coefficient vector looked up from a texture

For a given, fixed, outgoing direction, the BRDF lobe is a hemispherical function. We represent this function by (hemi)spherical harmonics. We discretize the outgoing directions and for each discrete outgoing direction, we compute the coefficient vector representing the BRDF lobe. This is done in pre-process.



Readiance Caching Results

This slide shows some images rendered with radiance caching. The images show that radiance caching can handle fairly sharp glossy reflections (boxes on the left), anisotropy (sphere) and curved geometry (flamingo). For sharper or more anisotropic reflections and for more complex geometry, radiance caching is usually outperformed by Monte Carlo importance sampling. As such, radiance caching complements importance sampling.

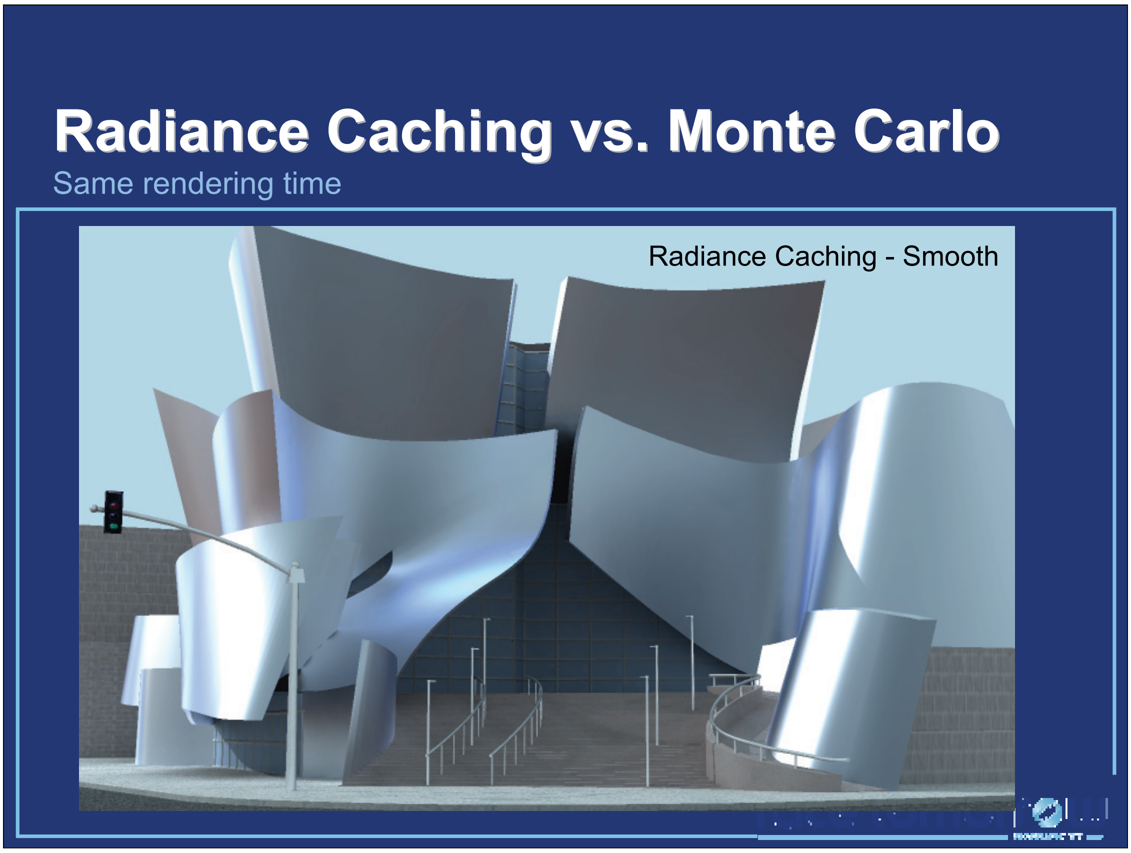


Radiance Caching vs. Monte Carlo

Same rendering time

MC Importance Sampling - Noise

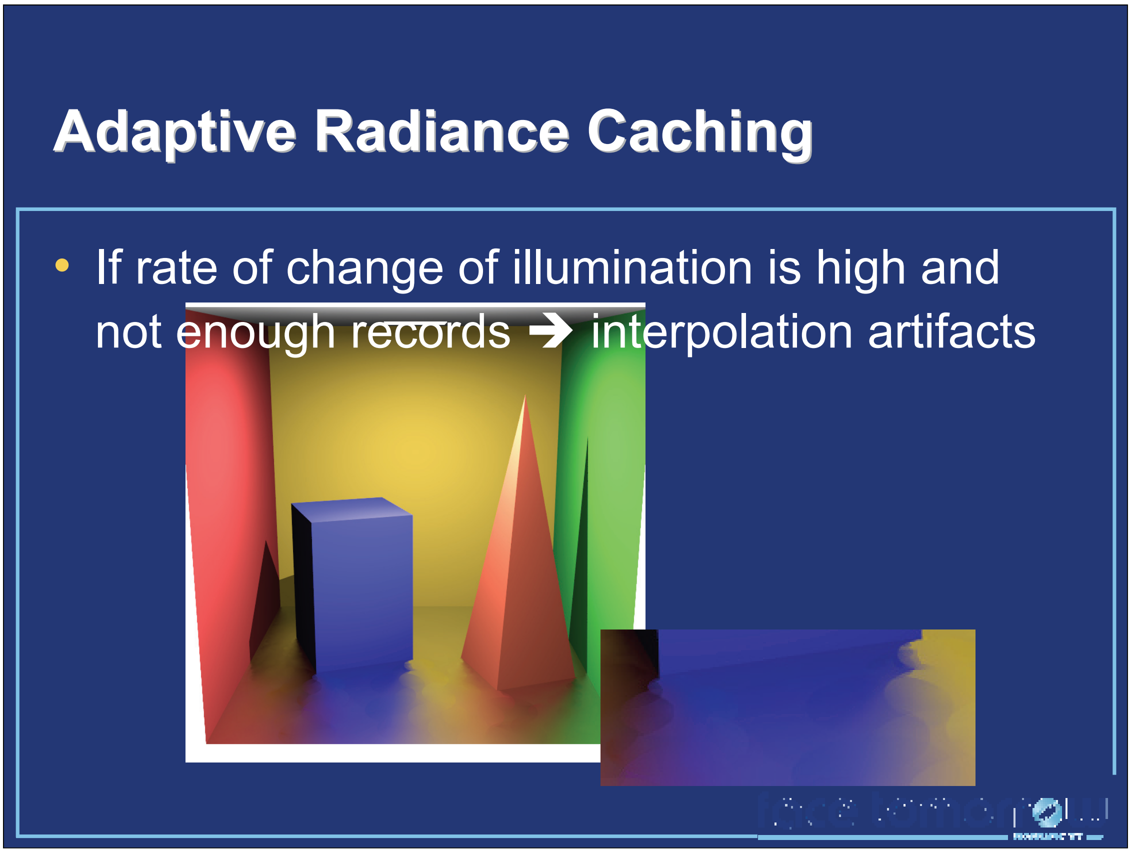
This and the following slide compare image quality delivered by MC importance sampling and radiance caching given the same rendering time. Here we see that Monte Carlo produces noisy image.



Radiance Caching vs. Monte Carlo

Same rendering time

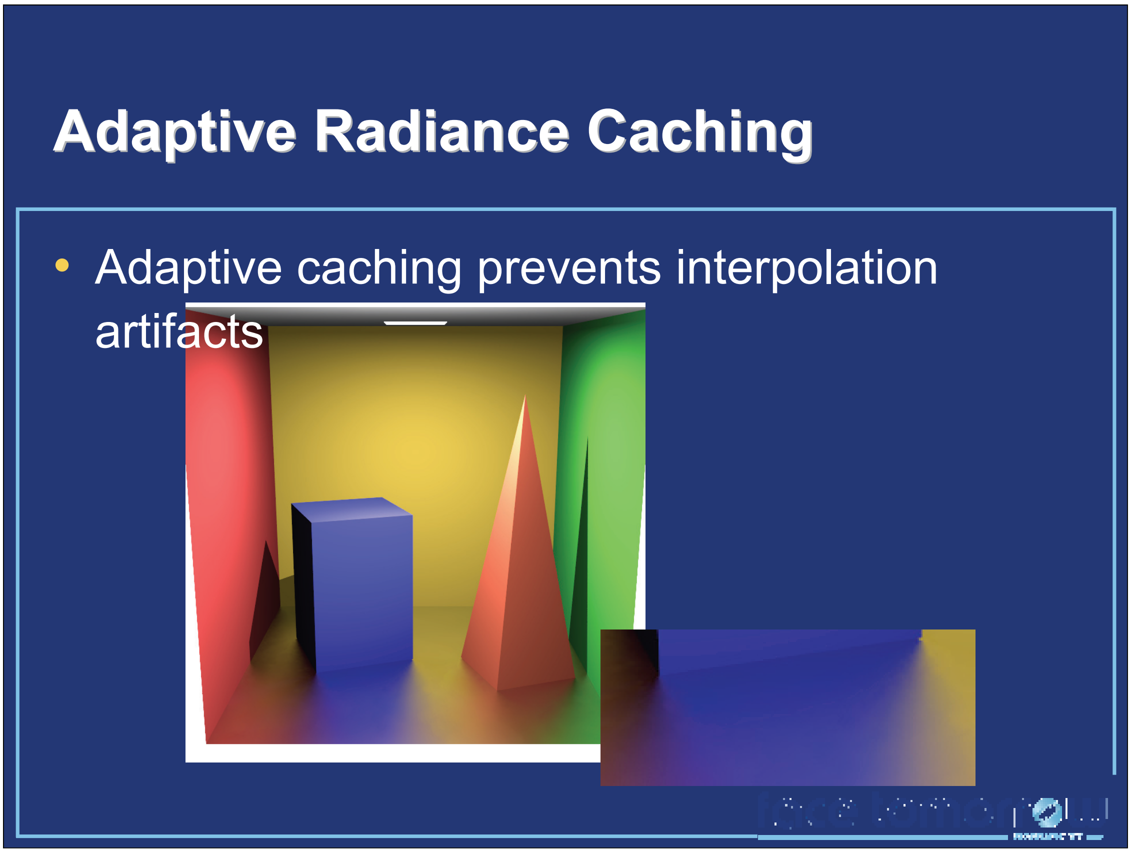
Given the same rendering time, radiance caching, produces smooth rendering.



Adaptive Radiance Caching

- If rate of change of illumination is high and not enough records ➔ interpolation artifacts

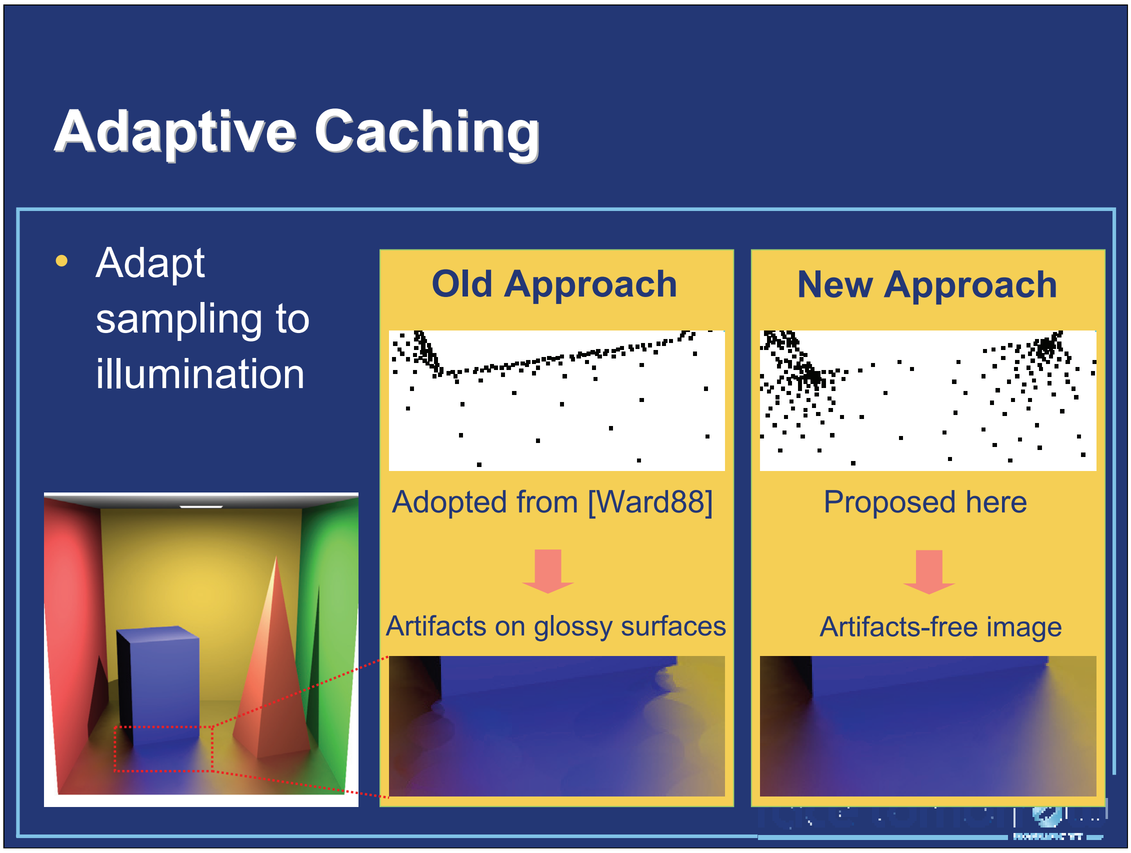
The basic radiance caching algorithm as described so far works fine in many cases, but it fails more often than irradiance caching.



Adaptive Radiance Caching

- Adaptive caching prevents interpolation artifacts

We use adaptive radiance caching to handle the failure cases.



Adaptive Caching

- Adapt sampling to illumination

Old Approach

Adopted from [Ward88]

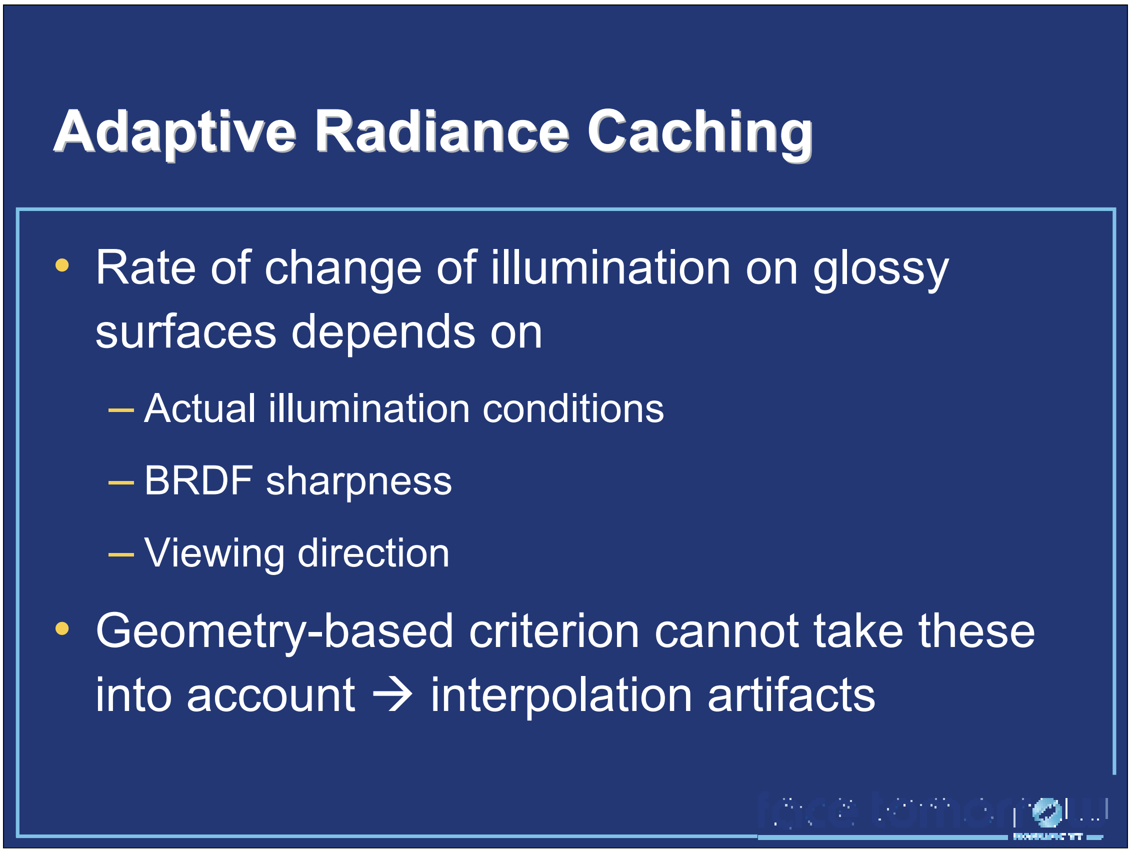
Artifacts on glossy surfaces

New Approach

Proposed here

Artifacts-free image

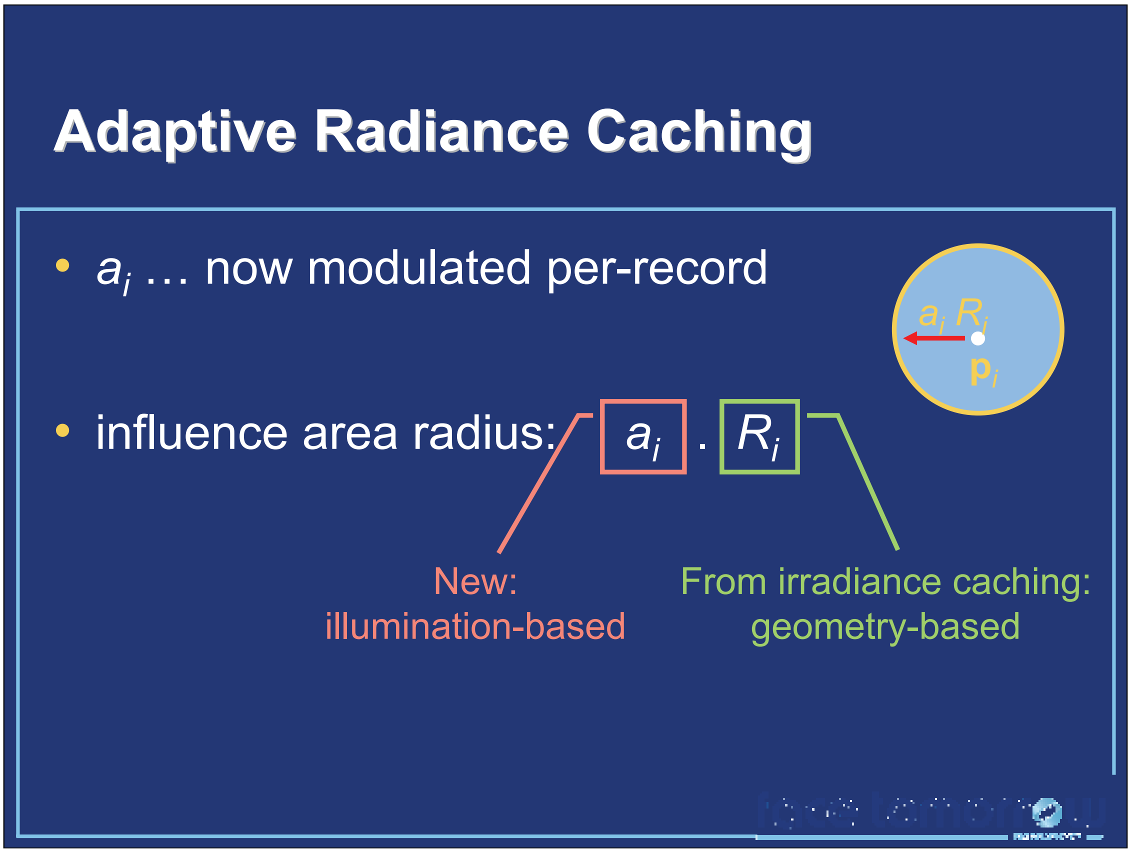
In particular, we adapt the density of spatial sampling of the indirect illumination to the actual local illumination conditions. This is in contrast to irradiance caching, where the sampling density is adapted based solely on the scene geometry.



Adaptive Radiance Caching

- Rate of change of illumination on glossy surfaces depends on
  - Actual illumination conditions
  - BRDF sharpness
  - Viewing direction
- Geometry-based criterion cannot take these into account → interpolation artifacts

When we take the geometry-based criterion from irradiance caching and use it on glossy surfaces, we may fail to predict important illumination changes. This is because indirect glossy illumination (as opposed to indirect diffuse) very much depends on the actual illumination conditions in the scene. In addition, it depends on the BRDF sharpness and the viewing direction.



Adaptive Radiance Caching

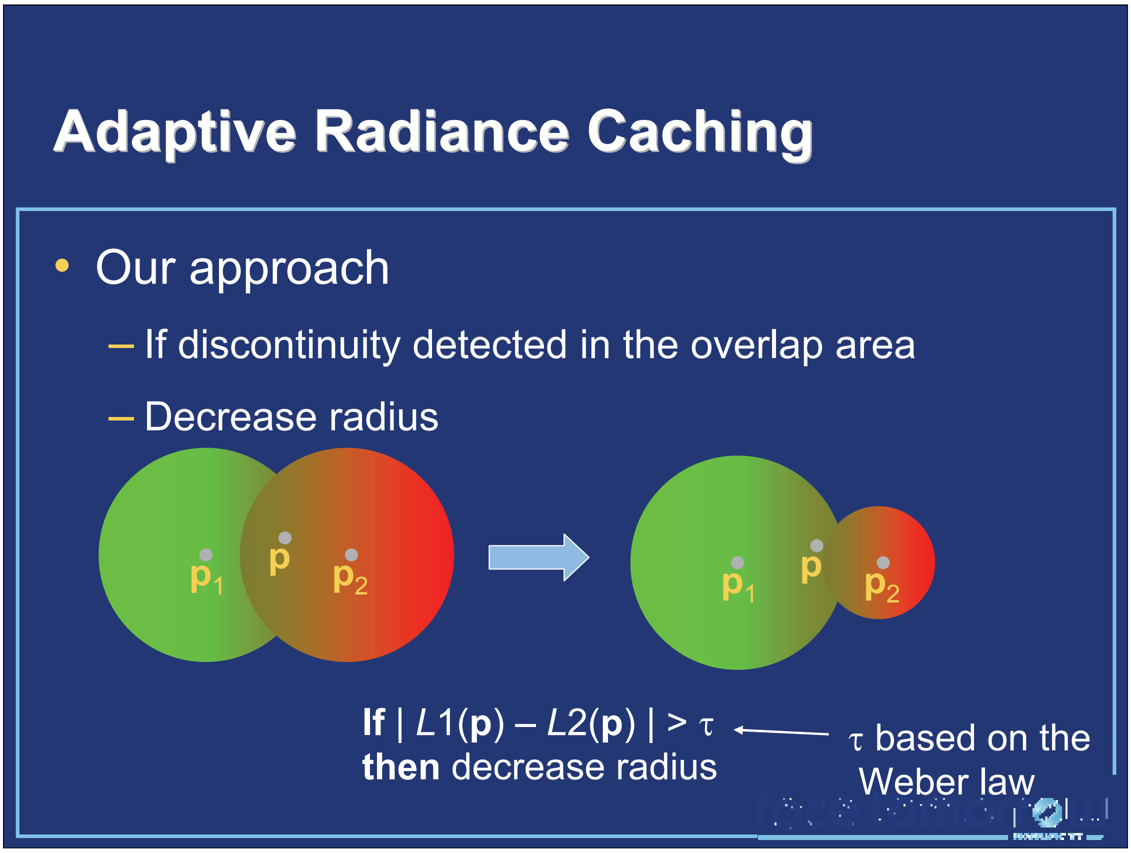
- a_i … now modulated per-record
- influence area radius: $\boxed{a_i} \,.\, \boxed{R_i}$

New:
illumination-based

From irradiance caching:
geometry-based

$a_i\,R_i$
$\mathbf{p}_i$

Remember that in irradiance caching and in the basic radiance caching, the record spacing is determined by the mean distance to the surrounding geometry, denoted R_i, multiplied by a user supplied constant *a*. In adaptive radiance caching, we extend this by also modulating the allowed error *a* automatically on a per-record basis in order to produce higher sampling in areas of high lighting complexity.



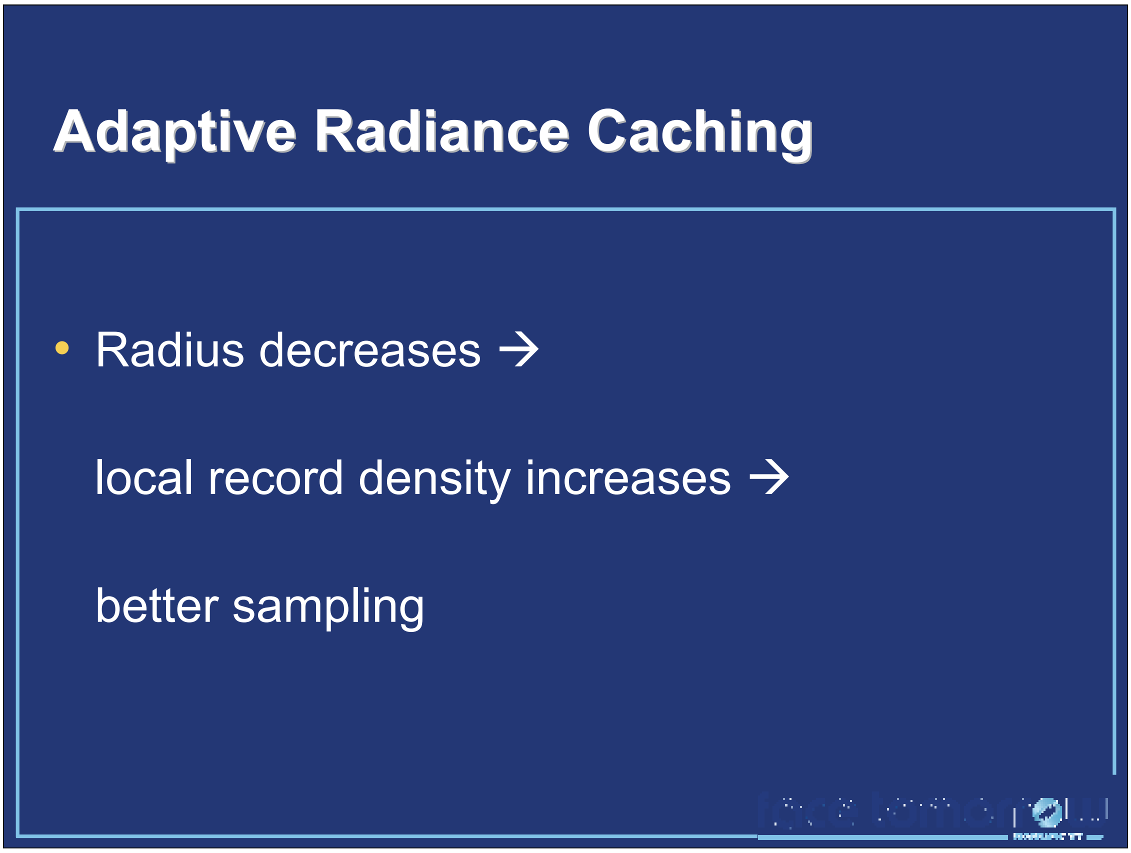
Adaptive Radiance Caching

- Our approach
  - If discontinuity detected in the overlap area
  - Decrease radius

If | $L1(\mathbf{p}) - L2(\mathbf{p})$ | > τ
then decrease radius

τ based on the Weber law

To adapt the a_i-value to the local illumination conditions, we proceed as follows. During the radiance interpolation, we perform an additional check on the difference of radiance values of the contributing records. If this difference is perceptually important, we conclude that a visible discontinuity would be produced by interpolation. To determine the perceptual importance, we use the simplest possible metric – the Weber law. It states that the minimum perceivable difference of luminance is given by a fixed fraction of the luminance value. We use the threshold of 2 to 3 %. If a potentially visible discontinuity is detected, we decrease the a_i value associated with one of the records in order to exclude it from interpolation. This ‘shrinks’ the area over which that record can be reused for interpolation.



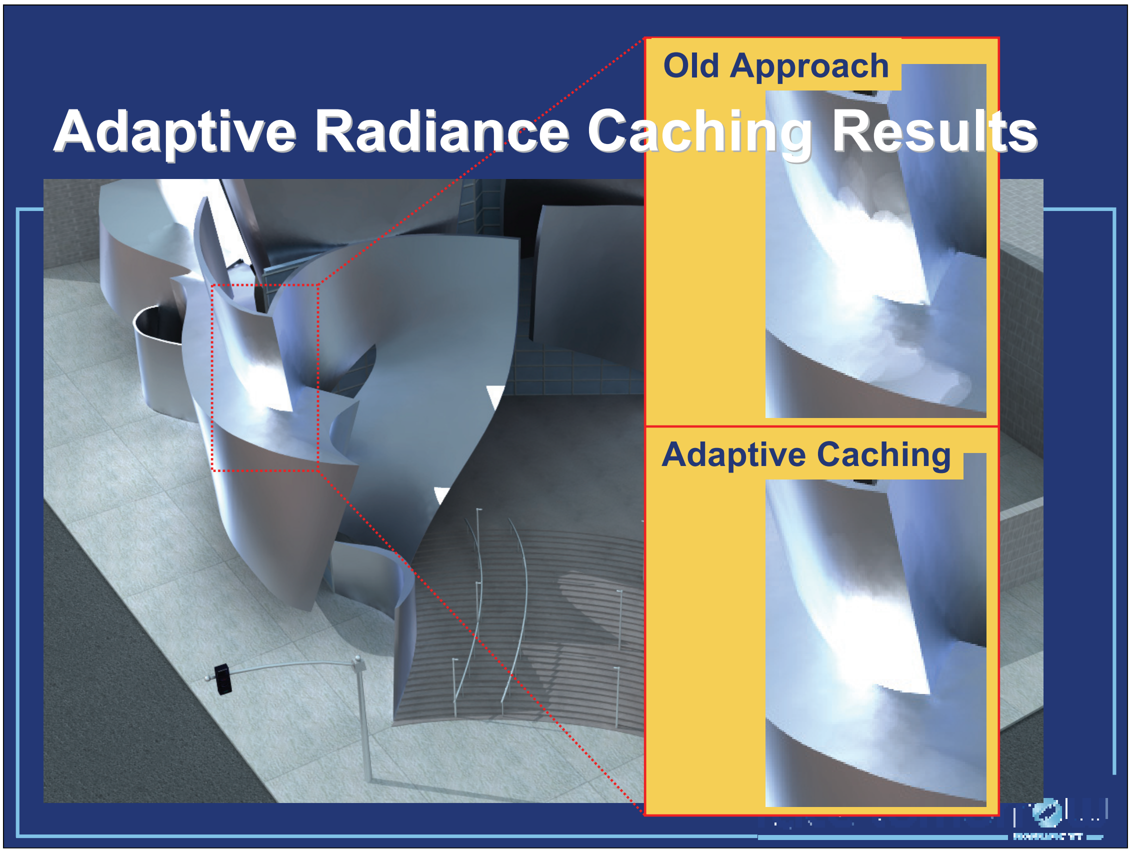
Adaptive Radiance Caching

- Radius decreases →

  local record density increases →

  better sampling

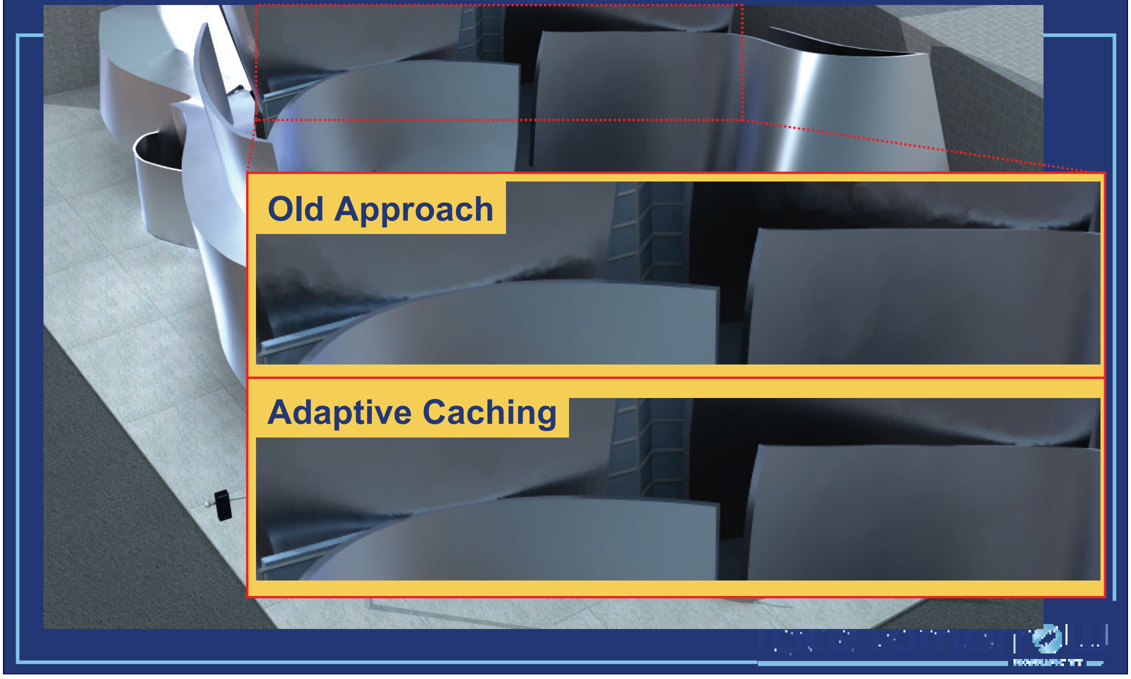
Shrinking the influence area of a records has for consequence increased local density of radiance records and thus better local sampling of indirect illumination.

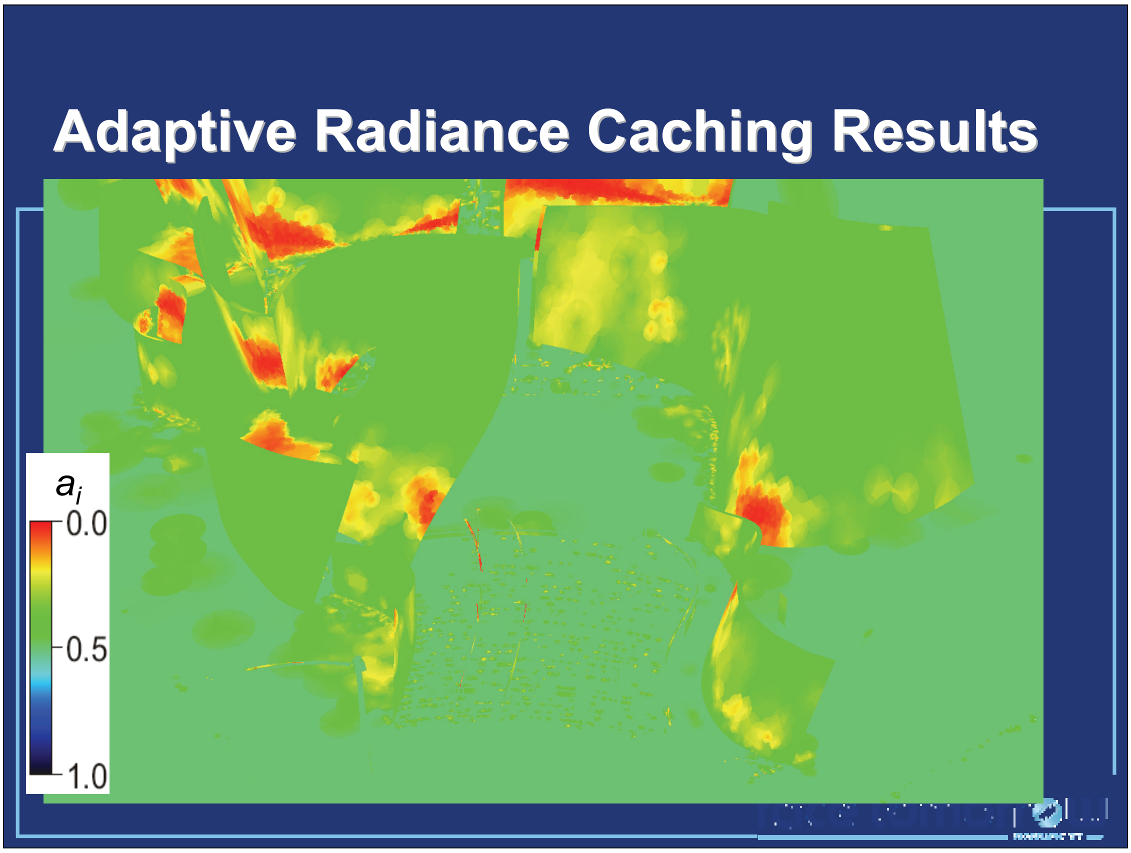


Adaptive Radiance Caching Results

This and the following slide show the effect of adaptive radiance caching on the rendered images. The two rendering were created in the same rendering time. Without adaptive radiance caching, we see some discontinuities because of the interpolation, which are successfully suppressed by adaptive caching.

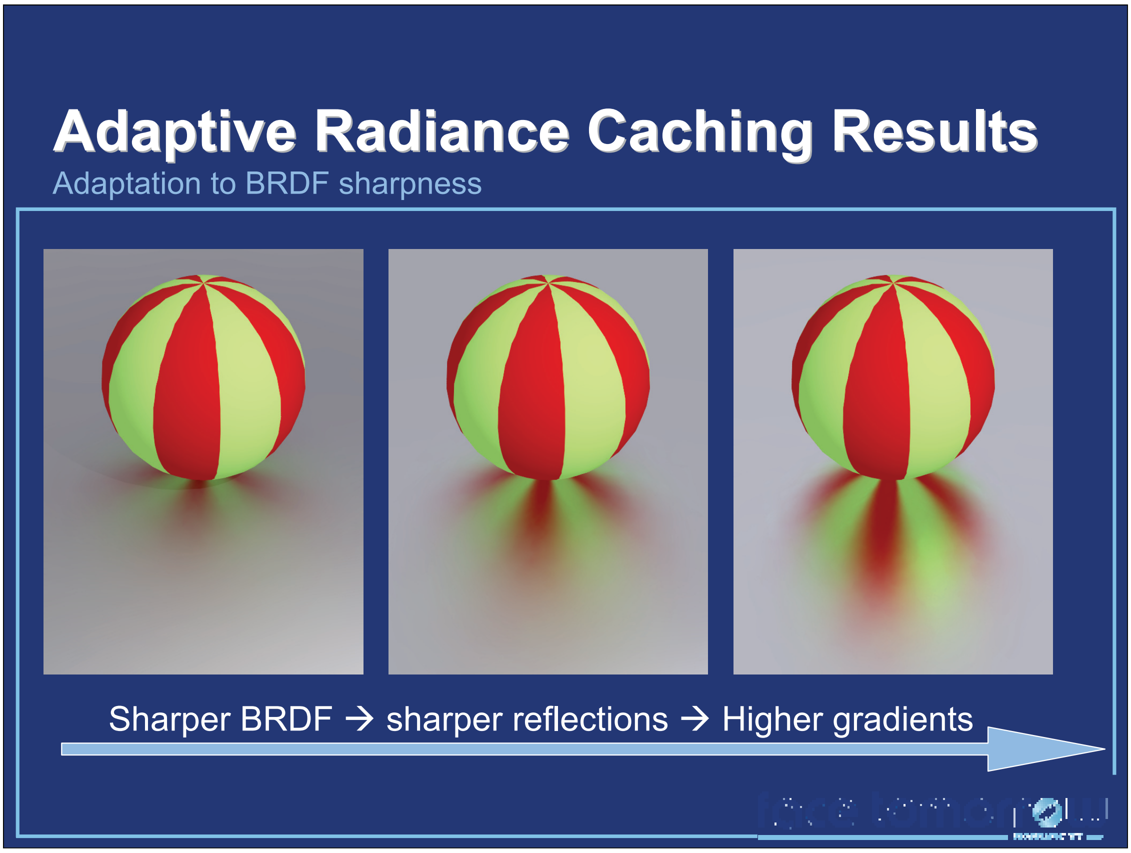
Adaptive Radiance Caching Results





Adaptive Radiance Caching Results

This image shows the locally adapted *ai* value for the rendering of the Walt Disney Hall.

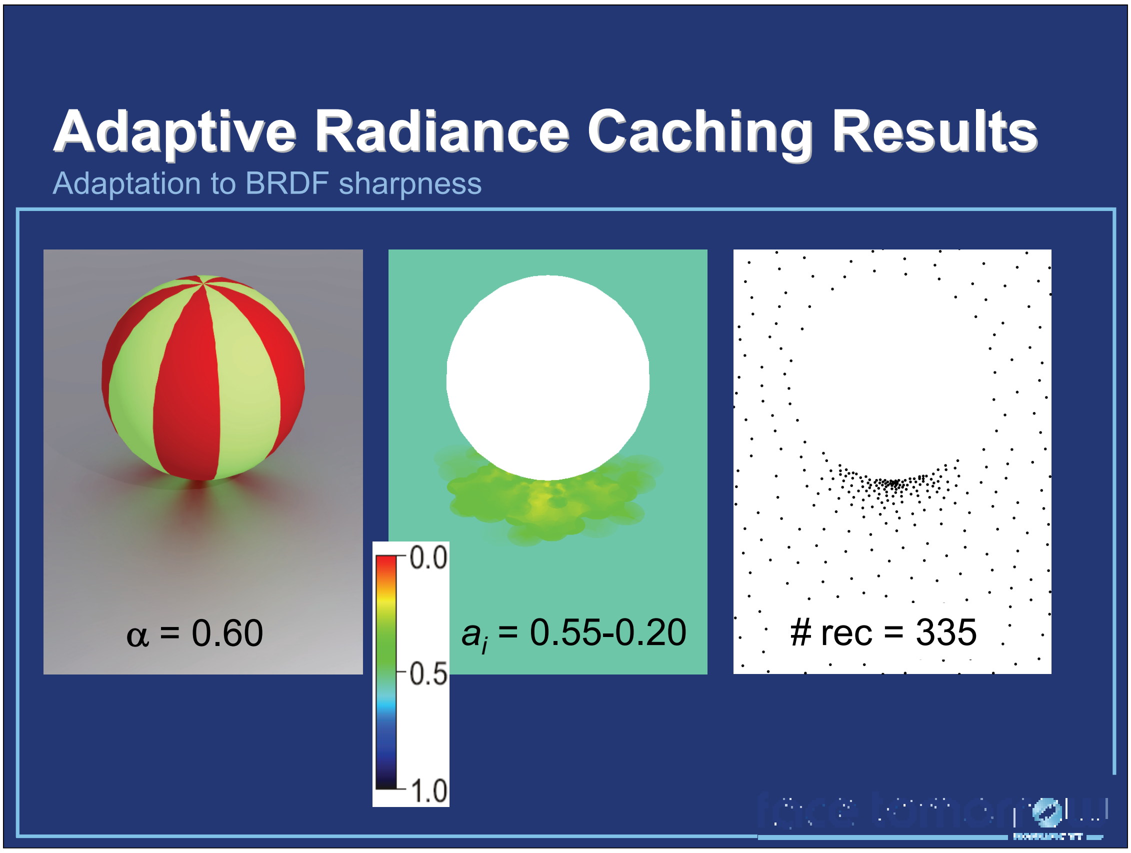


Adaptive Radiance Caching Results

Adaptation to BRDF sharpness

Sharper BRDF → sharper reflections → Higher gradients

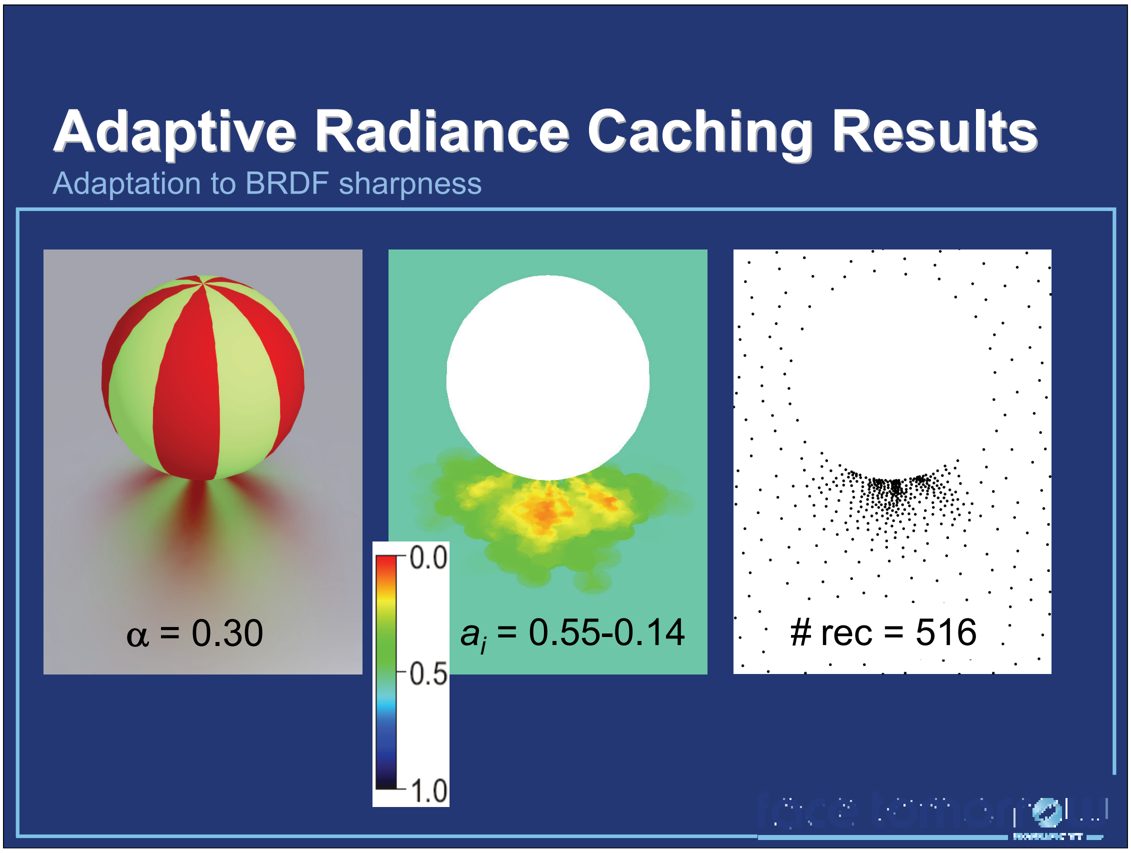
Adaptive caching also automatically adapts the record density to the BRDF sharpness. On these three images, the BRDF sharpness of the glossy floor gradually increases from left to right, giving sharper and sharper reflections. Sharper reflections result in higher illumination gradients.



Adaptive Radiance Caching Results

Adaptation to BRDF sharpness

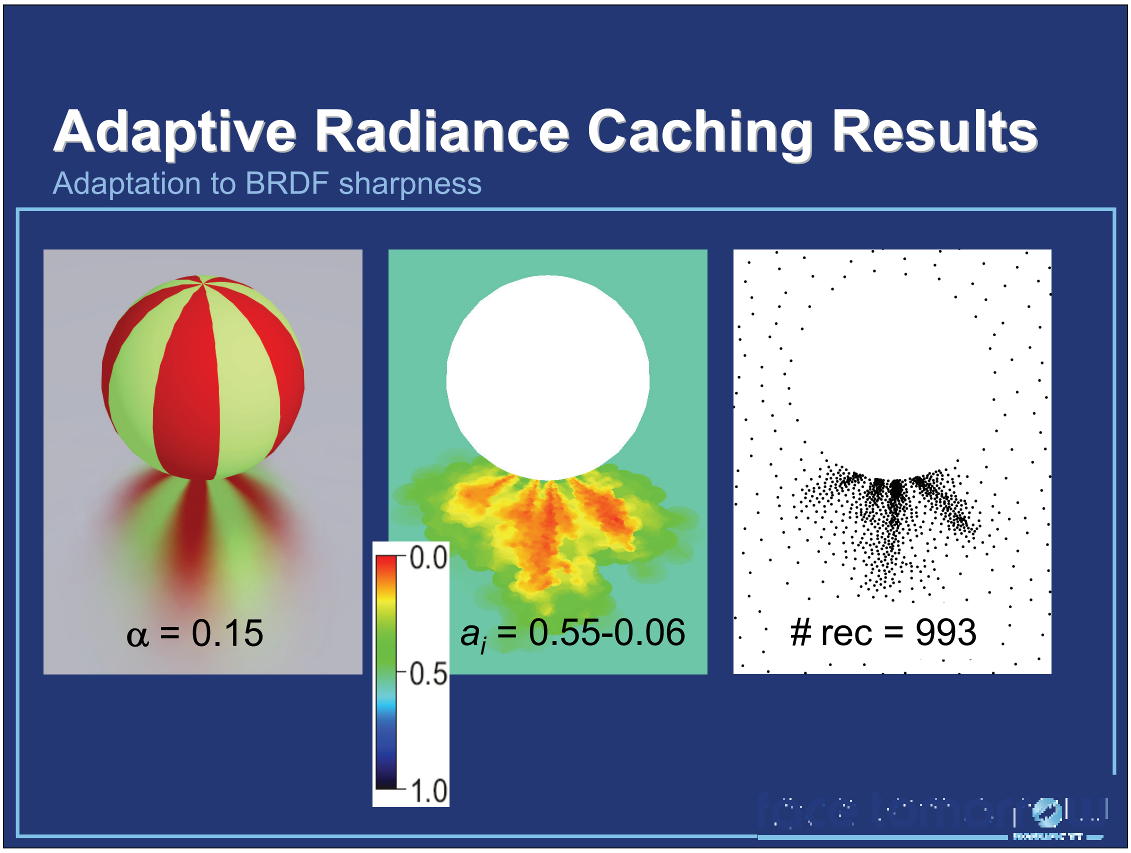
When the BRDF sharpness is small, we can get away with very sparse sampling.



Adaptive Radiance Caching Results

Adaptation to BRDF sharpness

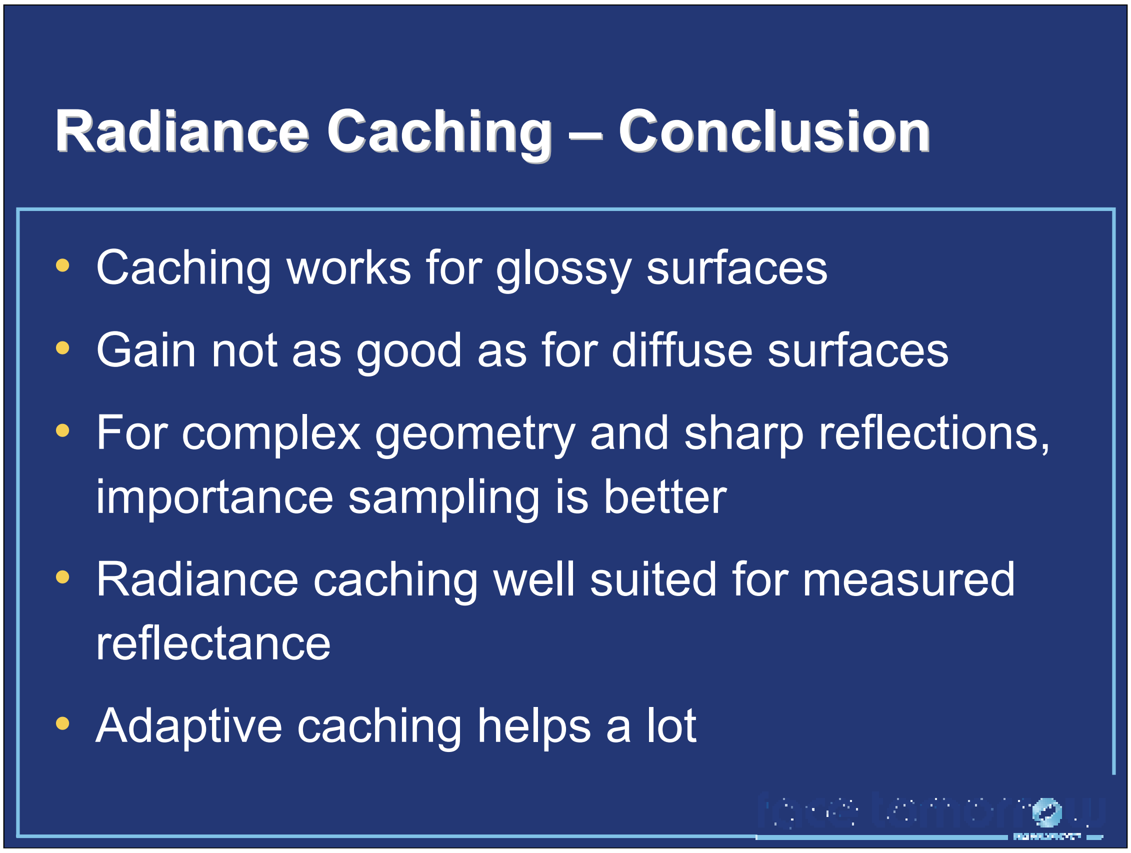
As the BRDF sharpness increases, the adaptive caching increases the record density accordingly, in order to avoid interpolation artifacts.



Adaptive Radiance Caching Results

Adaptation to BRDF sharpness

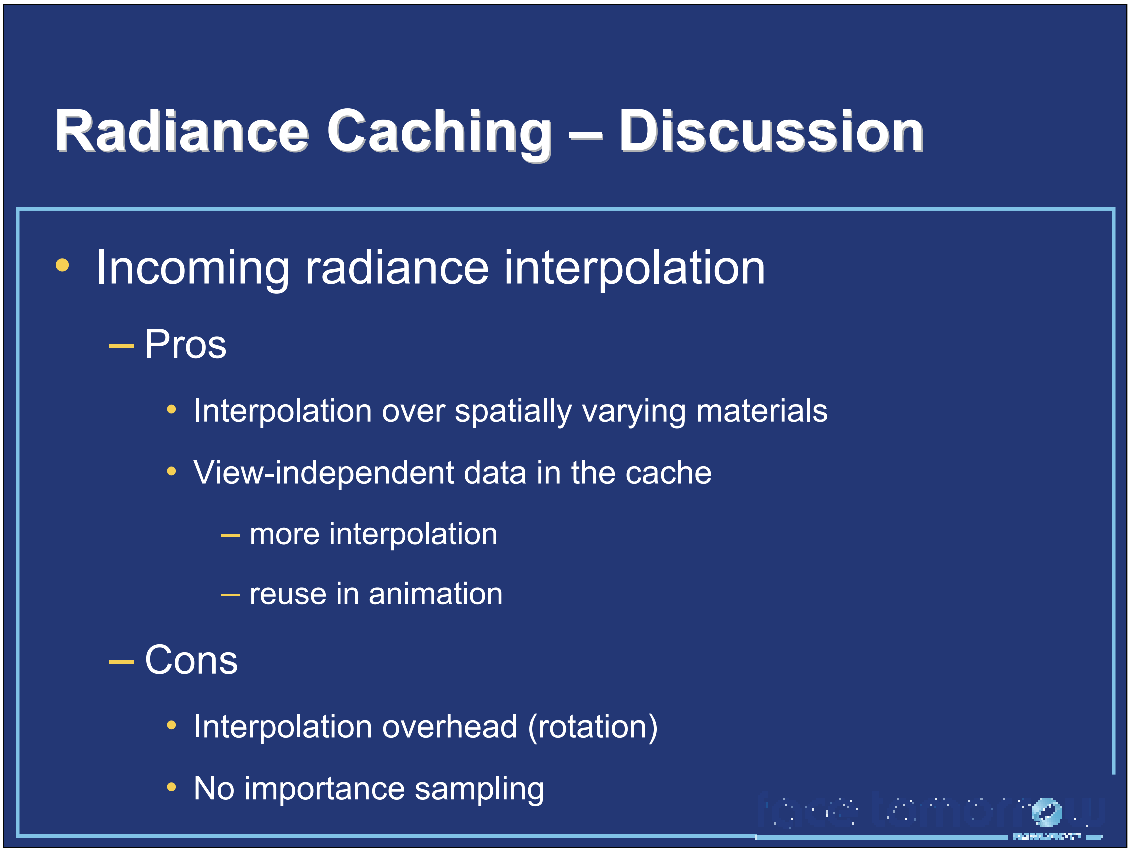
Increasing the BRDF sharpness further, still more records are automatically added by adaptive caching to preserve smoothness of indirect illumination. Notice that if we based the interpolation criterion a priori on the BRDF sharpness, we would have to increase the sampling density all over the glossy floor, which would result in a overly dense sampling and a significant performance drop.



Radiance Caching – Conclusion

- Caching works for glossy surfaces
- Gain not as good as for diffuse surfaces
- For complex geometry and sharp reflections, importance sampling is better
- Radiance caching well suited for measured reflectance
- Adaptive caching helps a lot

We have shown that lazy illumination caching can be used not only for diffuse surfaces, but even for glossy ones. The performance gain is not as substantial as for irradiance caching, though. The main overhead in radiance caching is due to uniform hemisphere sampling in radiance caching – this requires many sampling rays and, for a certain threshold of BRDF sharpness, importance sampling out-performs radiance caching. In addition, the interpolation performance is crippled by the spherical harmonic rotation. Nevertheless, radiance caching brings significant performance gains over Monte Carlo importance sampling when rendering fairly smooth geometry with low-frequency BRDF. In particular, radiance caching is very effective for rendering with measured reflectance data, for which importance sampling is difficult.

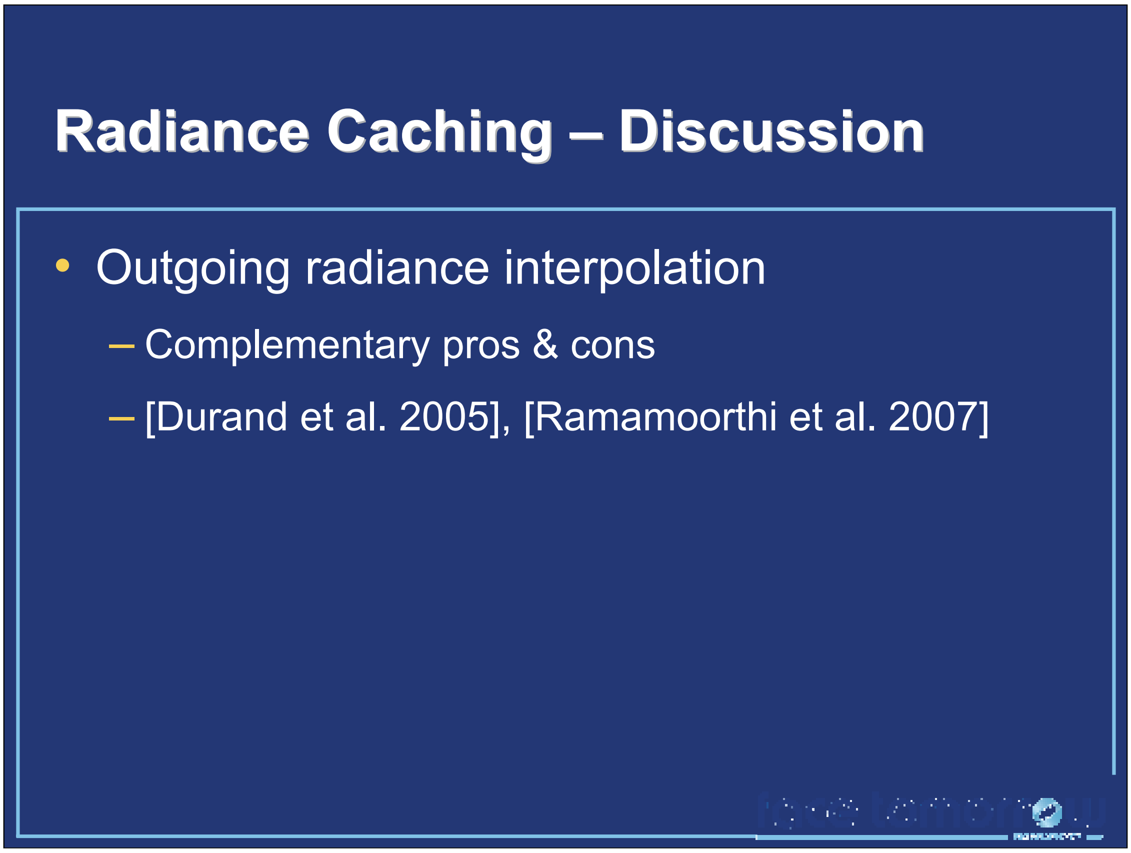


Radiance Caching – Discussion

- Incoming radiance interpolation
  - Pros
    - Interpolation over spatially varying materials
    - View-independent data in the cache
      - more interpolation
      - reuse in animation
  - Cons
    - Interpolation overhead (rotation)
    - No importance sampling

The essential design choice in radiance caching is to interpolate the directional distribution of incoming radiance (as opposed to interpolation outgoing radiance for the viewing direction). Since the cached quantity is view-independent, we can reuse cached values over larger areas and we can re-use the them in animations. In addition, we can also interpolate over spatially varying materials.

On the other hand, the overhead related to caching and interpolating a directional function may be quite large as discussed on the previous slide.



Radiance Caching – Discussion

- Outgoing radiance interpolation
  - Complementary pros & cons
  - [Durand et al. 2005], [Ramamoorthi et al. 2007]

An alternative to caching incoming radiance is to interpolate outgoing radiance for the viewing direction.

Radiance Caching References

- P. Gautron, J. Křivánek, S. Pattanaik, and K. Bouatouch, A Novel Hemispherical Basis for Accurate and Efficient Rendering, Eurographics Symposium on Rendering, 2004.
- J. Křivánek, P. Gautron , S. Pattanaik, and K. Bouatouch, Radiance Caching for Efficient Global Illumination Computation , IEEE TVCG, Vol. 11, No. 5, Sep./Oct. 2005
- J. Křivánek, K. Bouatouch, S. Pattanaik, and J. Žára, Making Radiance and Irradiance Caching Practical: Adaptive Caching and Neighbor Clamping, Eurographics Symposium on Rendering, 2006.

Source code:

http://www.cgg.cvut.cz/show_research.php?page=01

Hardware Implementation

SIGGRAPH2007

Pascal Gautron

Post-Doctoral Researcher

France Telecom R&D Rennes

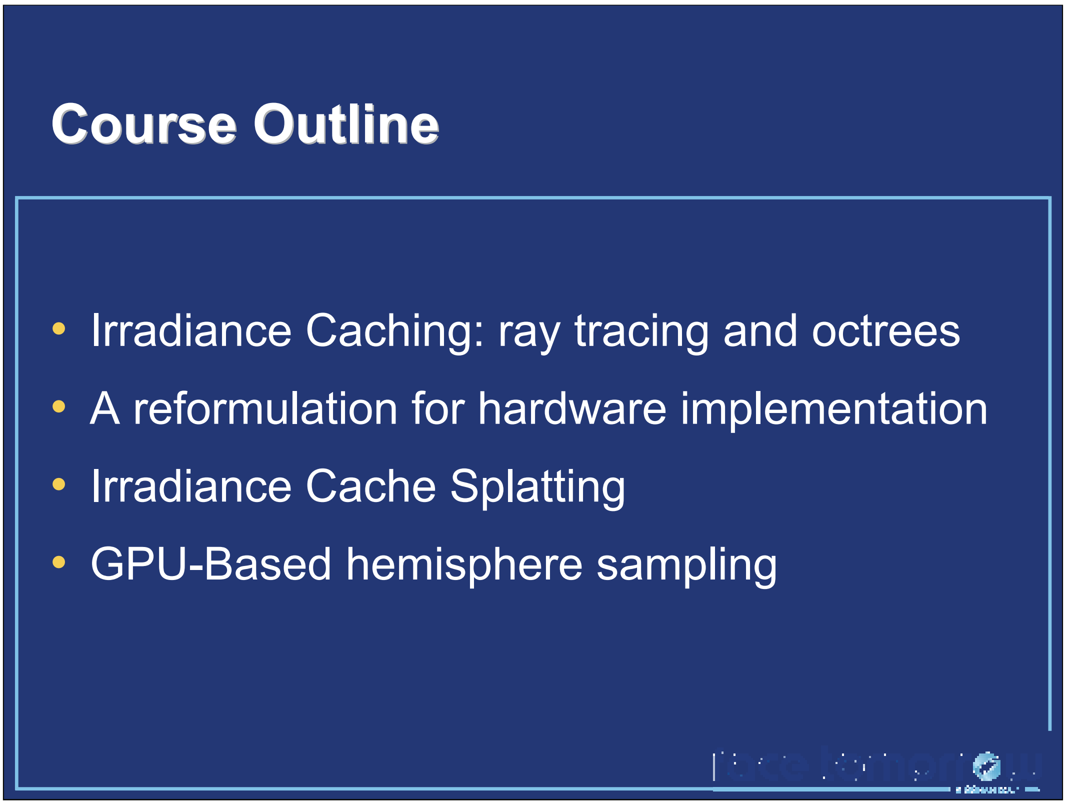
France

SIGGRAPH2007

Radiance Cache Splatting

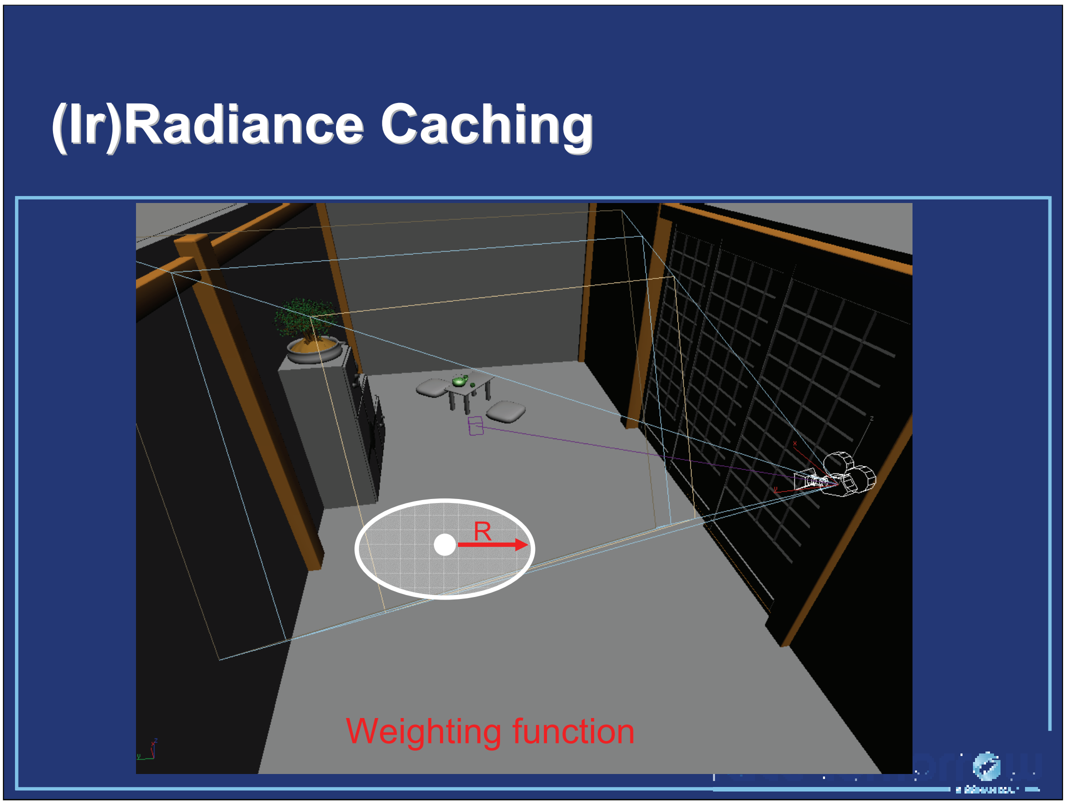
Pascal Gautron, Jaroslav Krivanek, Kadi Bouatouch, Sumanta Pattanaik

In Proceeding of Eurographics Symposium on Rendering 2005



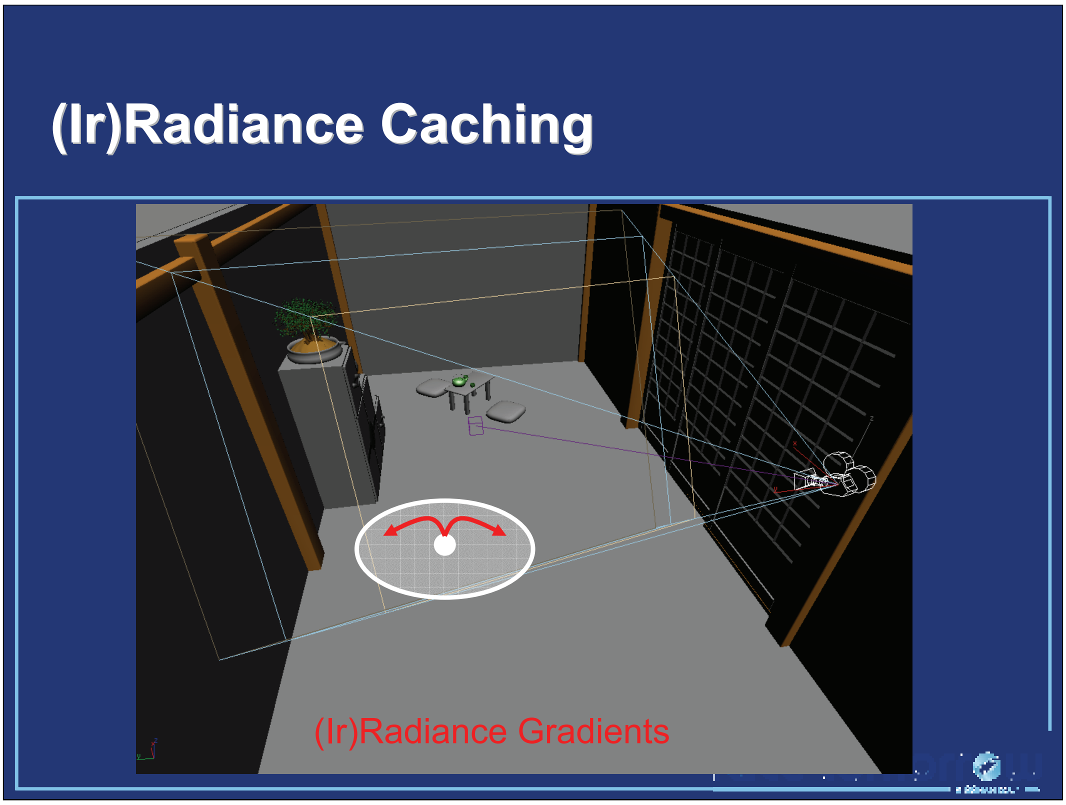
Course Outline

- Irradiance Caching: ray tracing and octrees
- A reformulation for hardware implementation
- Irradiance Cache Splatting
- GPU-Based hemisphere sampling



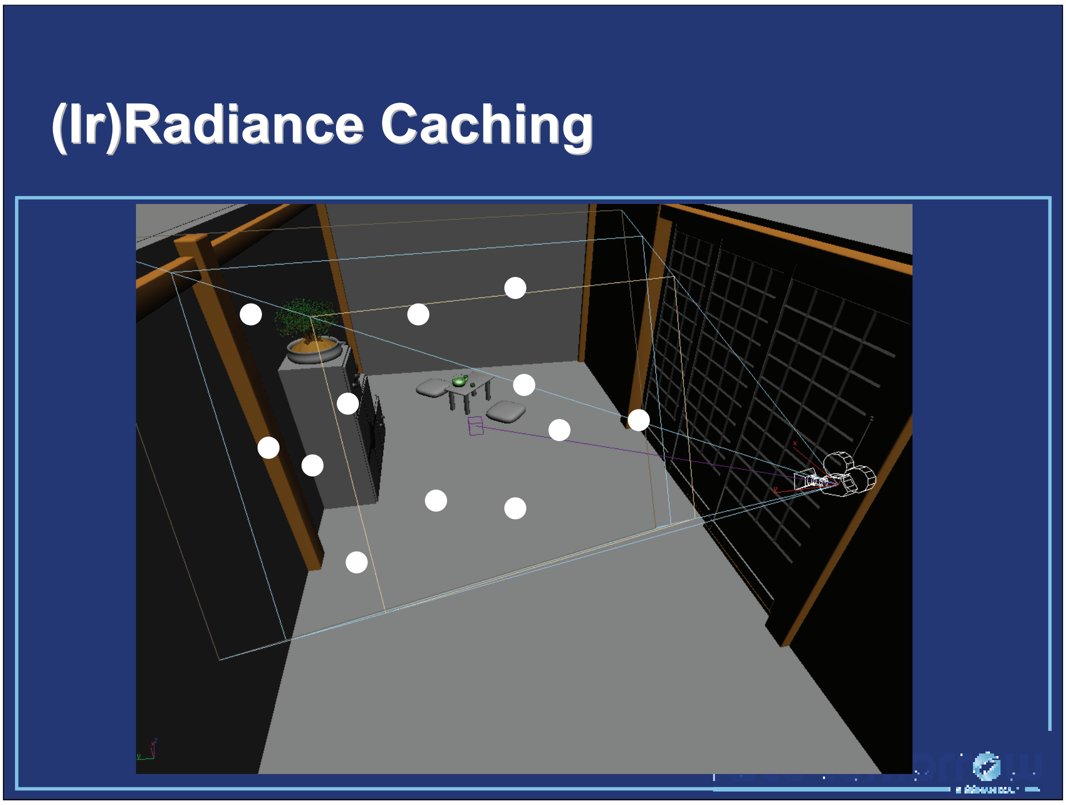
(Ir)Radiance Caching

The irradiance caching algorithm is based on sparse sampling and interpolation of indirect diffuse lighting at visible points. Each irradiance record contributes to the indirect lighting of points within its zone of influence. The size of this zone is adapted according to the mean distance R to the surrounding objects using the irradiance weighting function [Ward88].



(Ir)Radiance Caching

The irradiance value at points within the zone of influence of a record can be extrapolated using irradiance gradients [Ward92, Krivanek05a, Krivanek05b].



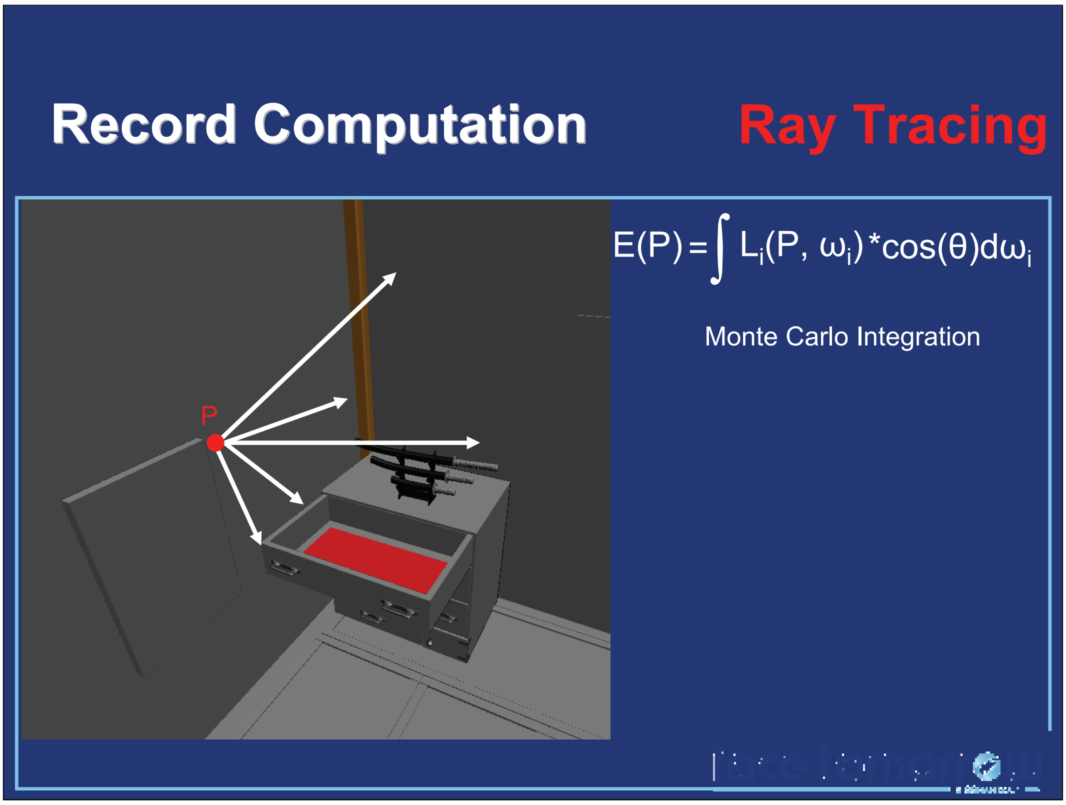
(Ir)Radiance Caching

When the zones of influence of the records cover the entire zone visible from the viewpoint, the image representing the indirect lighting can be rendered. The irradiance caching algorithm is then divided into three steps:

-The computation of the records

-The records storage

-The estimation of the indirect lighting using nearby records



Record Computation

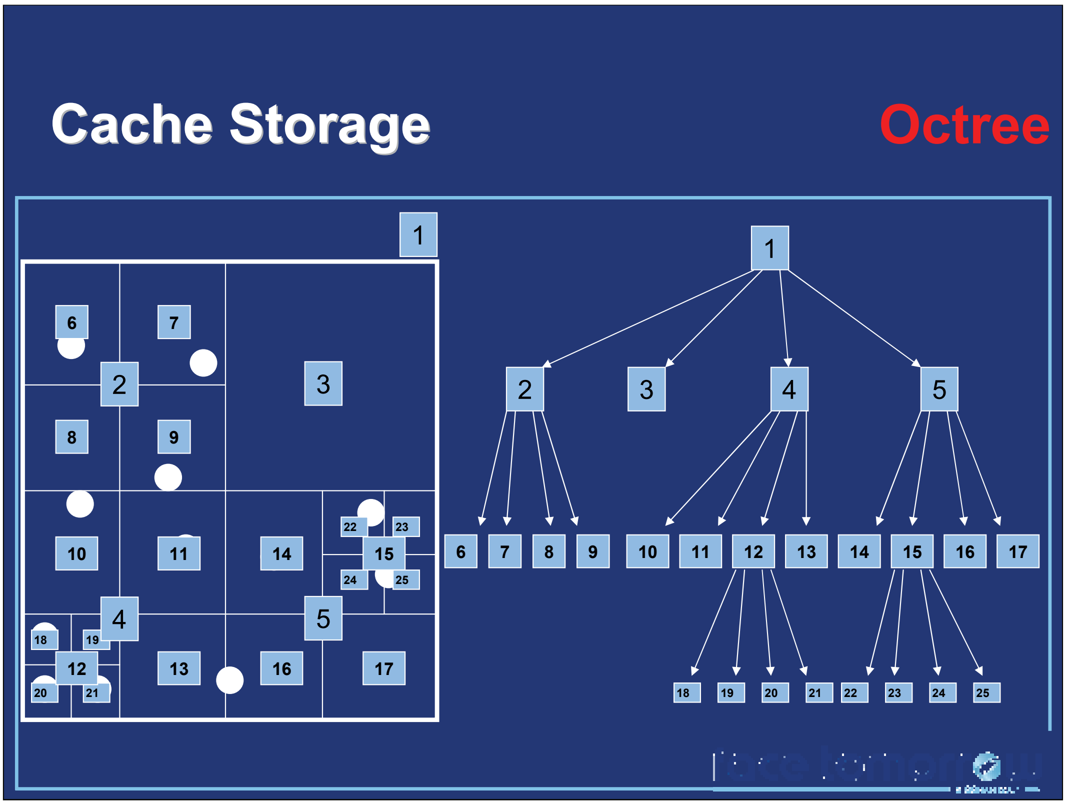
Ray Tracing

$$E(P) = \int L_i(P, \omega_i) * \cos(\theta) d\omega_i$$

Monte Carlo Integration

The computation of the irradiance value at a given point P requires the evaluation of the integral of the lighting over the surrounding hemisphere. This integral is typically estimated by Monte Carlo integration. Since the irradiance caching algorithm reuses the value of irradiance records over many pixels, the irradiance value of the record is computed with high precision, typically by tracing several hundreds to thousands rays.

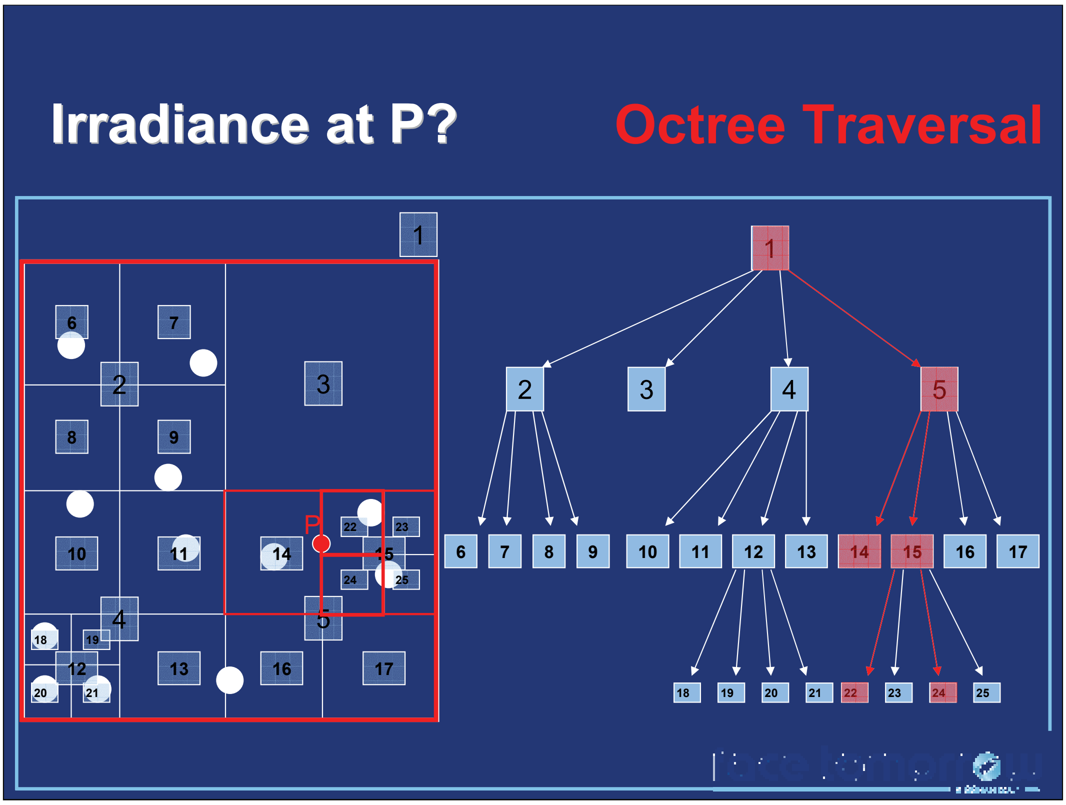
Once the irradiance value is computed, a record is created. This record contains the corresponding position, normal, irradiance, gradients, and mean distance to the surrouding objects.



Cache Storage

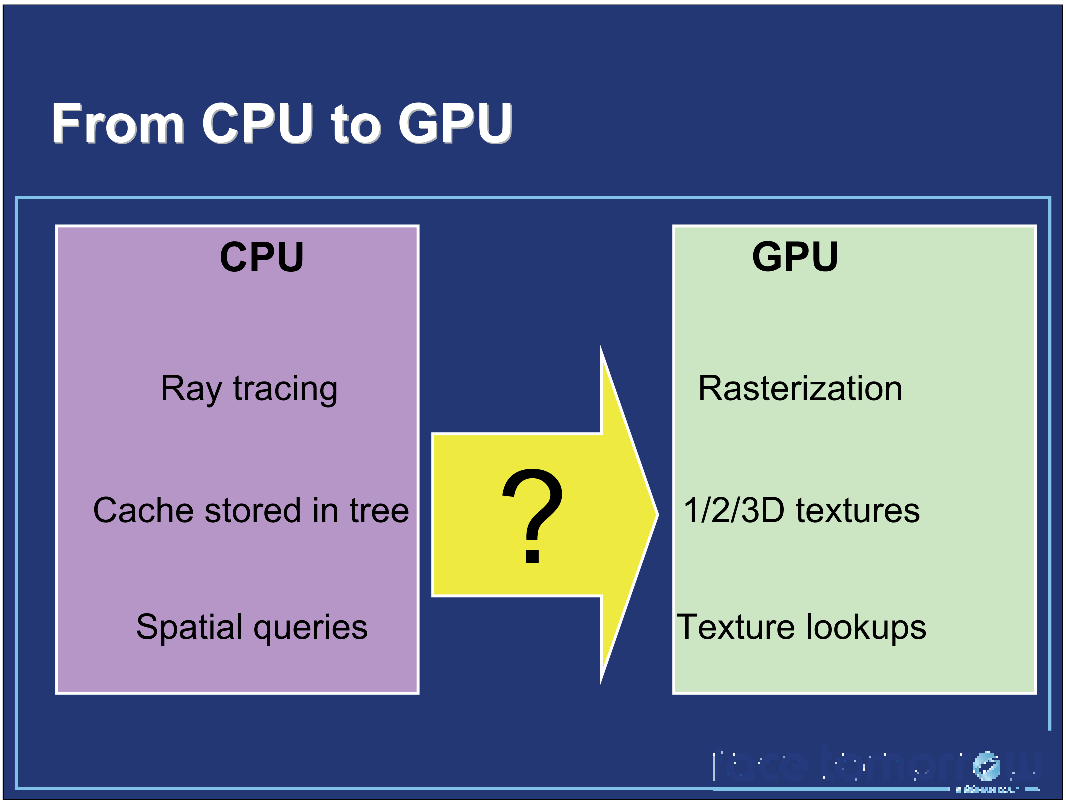
Octree

The records are then stored in an octree, which allows for fast spatial queries. Note that the octree is a recursive data structure, which construction involves many conditional statements.



Irradiance at P? Octree Traversal

The irradiance at a point P can then be estimated efficiently by querying the octree for nearby records. However, this involves a traversal of the structure, which also involves many conditional statements.



From CPU to GPU

CPU

Ray tracing

Cache stored in tree

Spatial queries

?

GPU

Rasterization

1/2/3D textures

Texture lookups

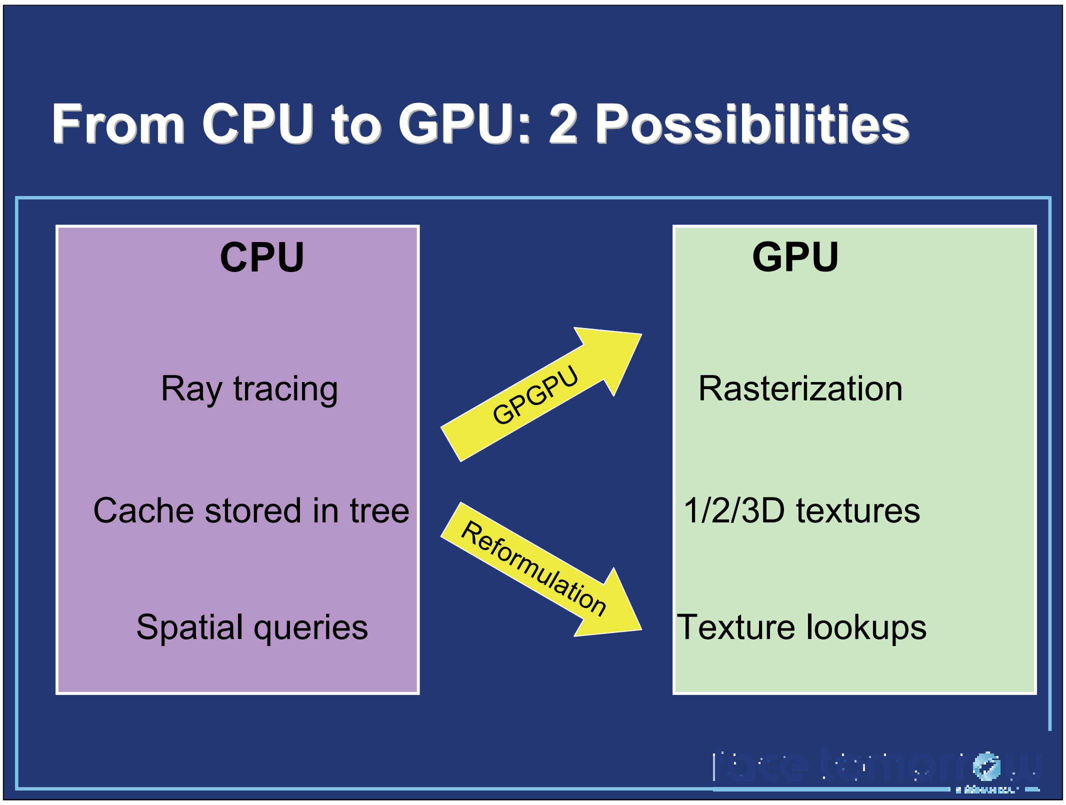
To summarize, the classical irradiance caching algorithm is implemented on the CPU, and is based on:

-Ray tracing

-Octrees

-Spatial queries in the octree

However, the GPU does not natively implement those operations: the visibility tests are performed using rasterization and Z-Buffering, and the only data structures available are 1, 2, and 3D textures. Particularly, the GPUs do not support pointers, which are the most common way of implementing recursive data structures.



From CPU to GPU: 2 Possibilities

CPU

Ray tracing

Cache stored in tree

Spatial queries

GPGPU

Reformulation

GPU

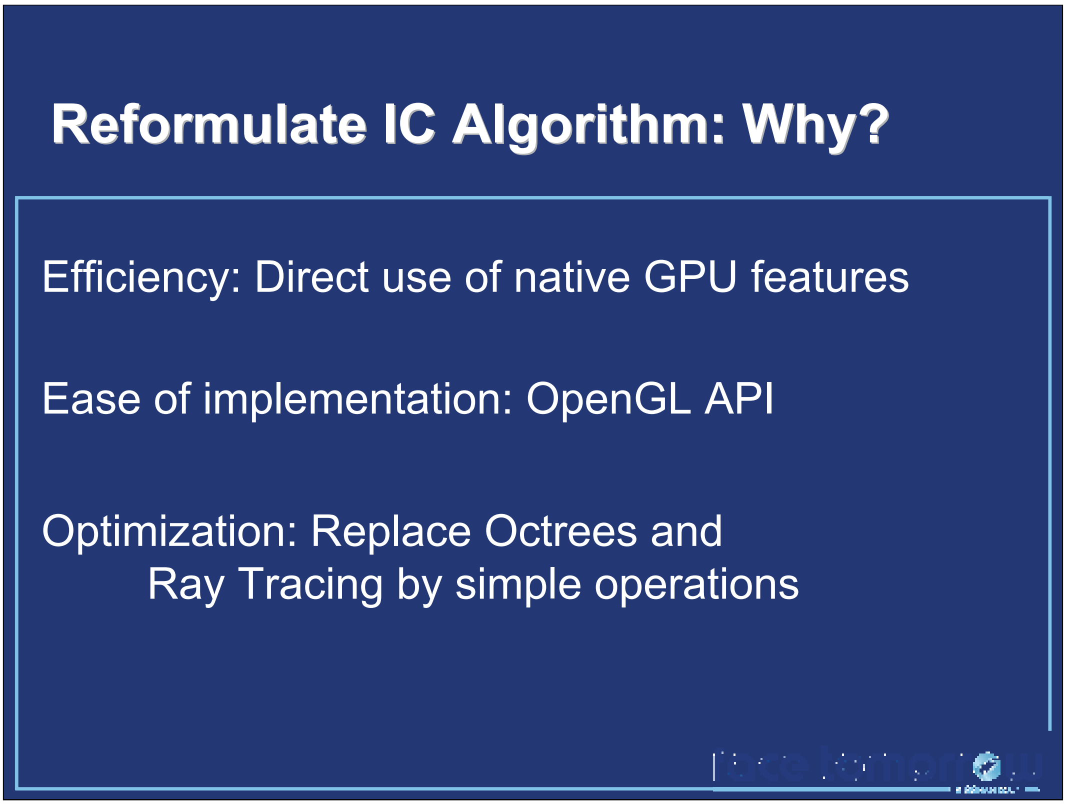
Rasterization

1/2/3D textures

Texture lookups

The irradiance caching algorithm cannot be directly mapped to the GPU architecture. Two methods can be considered: first, the use of libraries for general purpose computations on the GPU (such as Brook for GPU, http://graphics.stanford.edu/projects/brookgpu/). An important amount of research work has been performed to achieve interactive ray tracing on GPUs (such as [Purcell02, Purcell03]). A kD-Tree implementation for GPUs has also been proposed [Foley05].

An other way of performing irradiance caching on graphics hardware is to reformulate the algorithm so that only the native features of the GPU are used. This would allow us to get the best performance out of the graphics processors.



Reformulate IC Algorithm: Why?

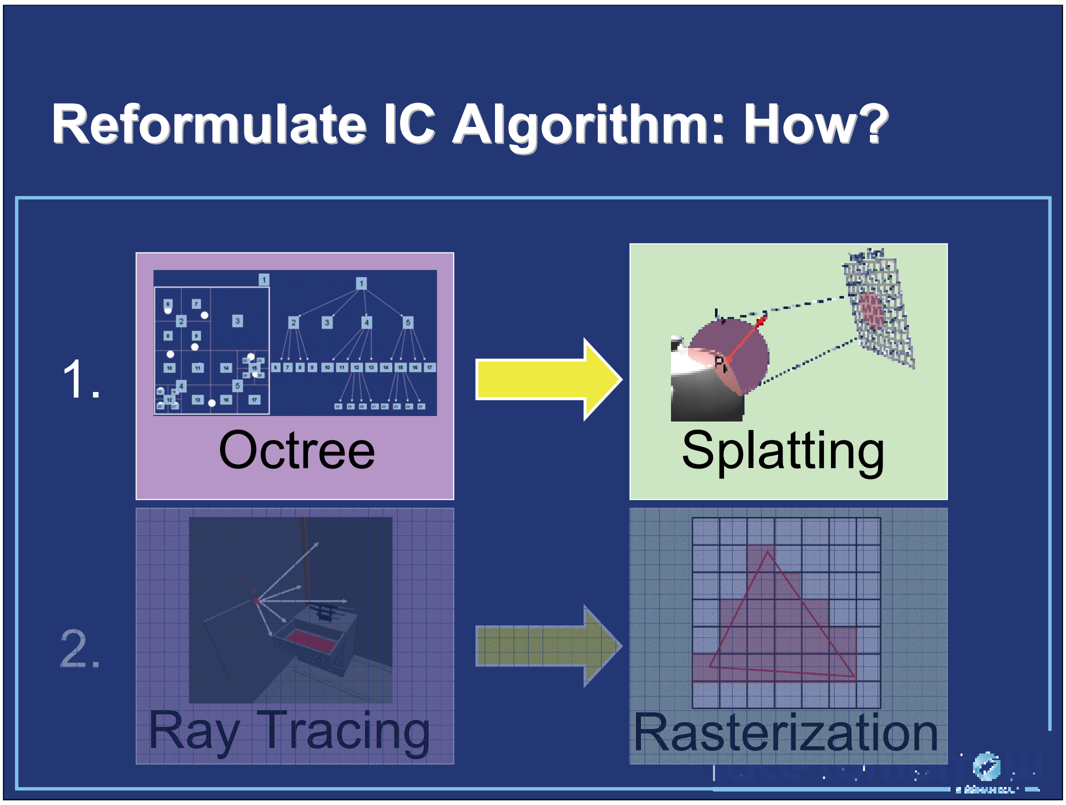
Efficiency: Direct use of native GPU features

Ease of implementation: OpenGL API

Optimization: Replace Octrees and
Ray Tracing by simple operations

More precisely, we chose to reformulate the irradiance caching algorithm for three reasons. First, the direct use of native GPU features allow us to use each part of the GPU at its best, improving the performance. Second, the reformulated algorithm can be implemented directly using at 3D graphics API such as OpenGL or DirectX.

Third, this reformulation gives us the occasion of attacking two costly aspects of the irradiance caching algorithm: the hierarchical data structure, and the irradiance computation using ray tracing. Replacing those by more simple operations on the GPU increases the performance, yielding interactive frame rates in simple scenes.

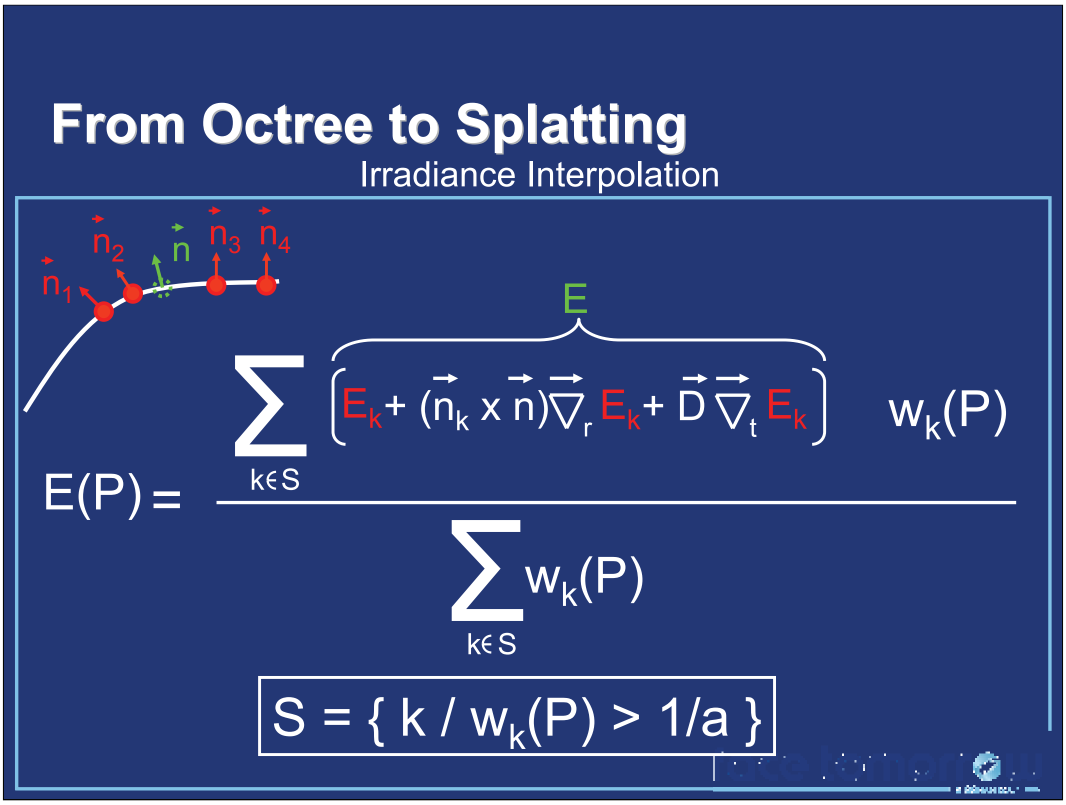


Reformulate IC Algorithm: How?

1. Octree → Splatting

2. Ray Tracing → Rasterization

The reformulation is divided into two tasks: the replacement of the octree by a splatting operation, and the use of rasterization in place of ray tracing.



From Octree to Splatting

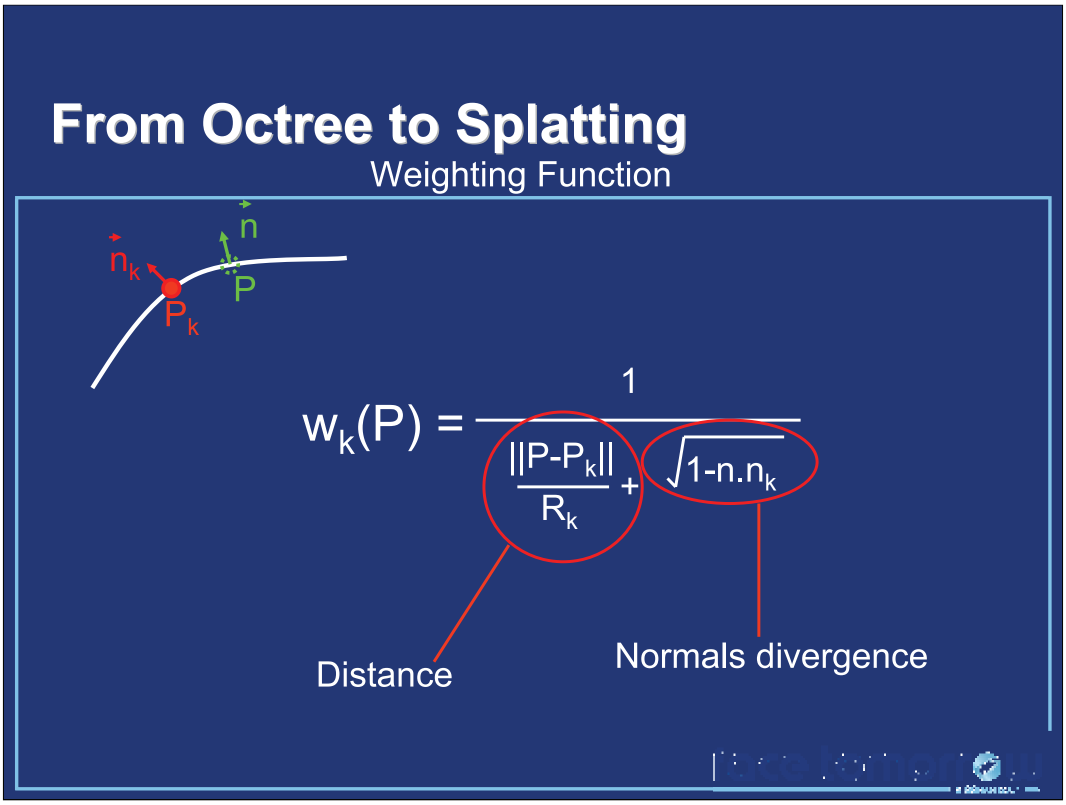
Irradiance Interpolation

$\vec{n}_1$
$\vec{n}_2$
$\vec{n}$
$\vec{n}_3$
$\vec{n}_4$

$$E(P) = \frac{\displaystyle\sum_{k\in S} \overbrace{\left[E_k + (\vec{n}_k \times \vec{n})\vec{\nabla}_r E_k + \vec{D}\,\vec{\nabla}_t E_k\right]}^{E} \; w_k(P)}{\displaystyle\sum_{k\in S} w_k(P)}$$

$$\boxed{S = \{\, k \,/\, w_k(P) > 1/a \,\}}$$

The reformulation is based on the irradiance interpolation equation [Ward88, Ward92] presented before: the irradiance estimate at a point P is the weighted average of the contributions of the surrounding records. The contribution of each record is computed using irradiance gradients for translation and rotation. The set S of contributing records is defined as the set of records for which the weigthing function evaluated at P is above a user-defined threshold a.



From Octree to Splatting

Weighting Function

$$w_k(P) = \frac{1}{\frac{\|P-P_k\|}{R_k} + \sqrt{1-\vec{n}.\vec{n}_k}}$$

Distance

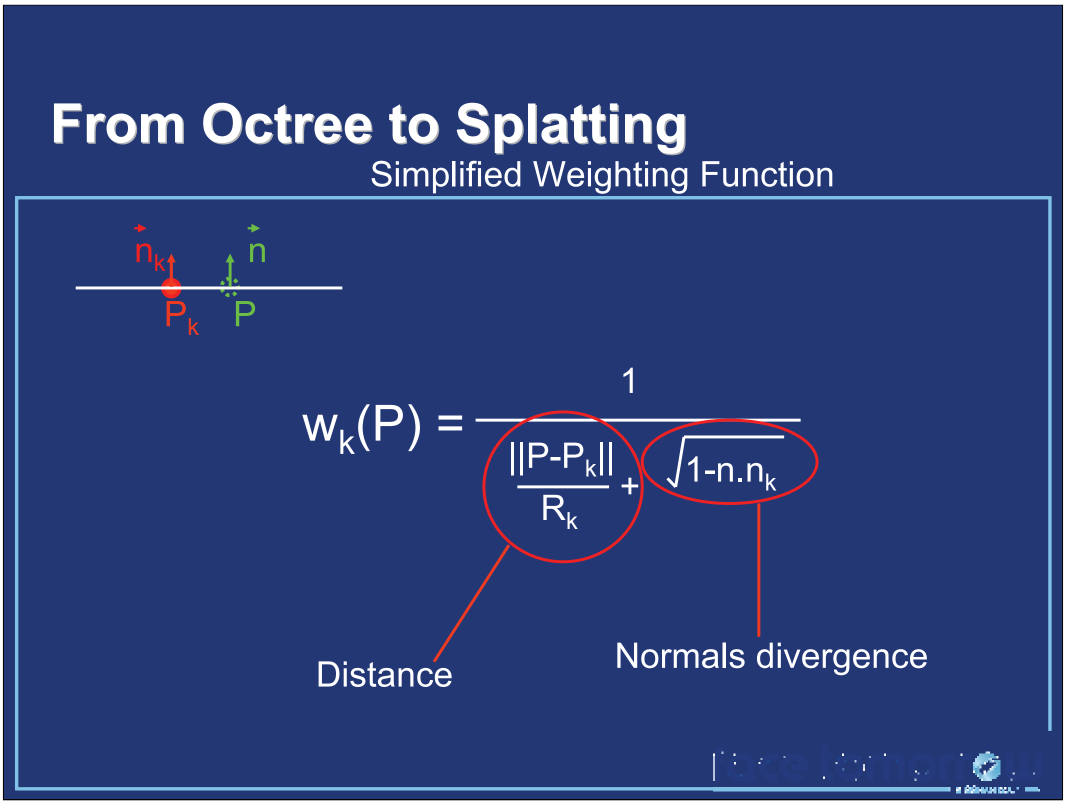
Normals divergence

The weighting function is very simple, and depends on:

-The distance between the record location and the point P

-The mean distance R to the surrounding objects

-The divergence of the surface normals between the record location and the point P

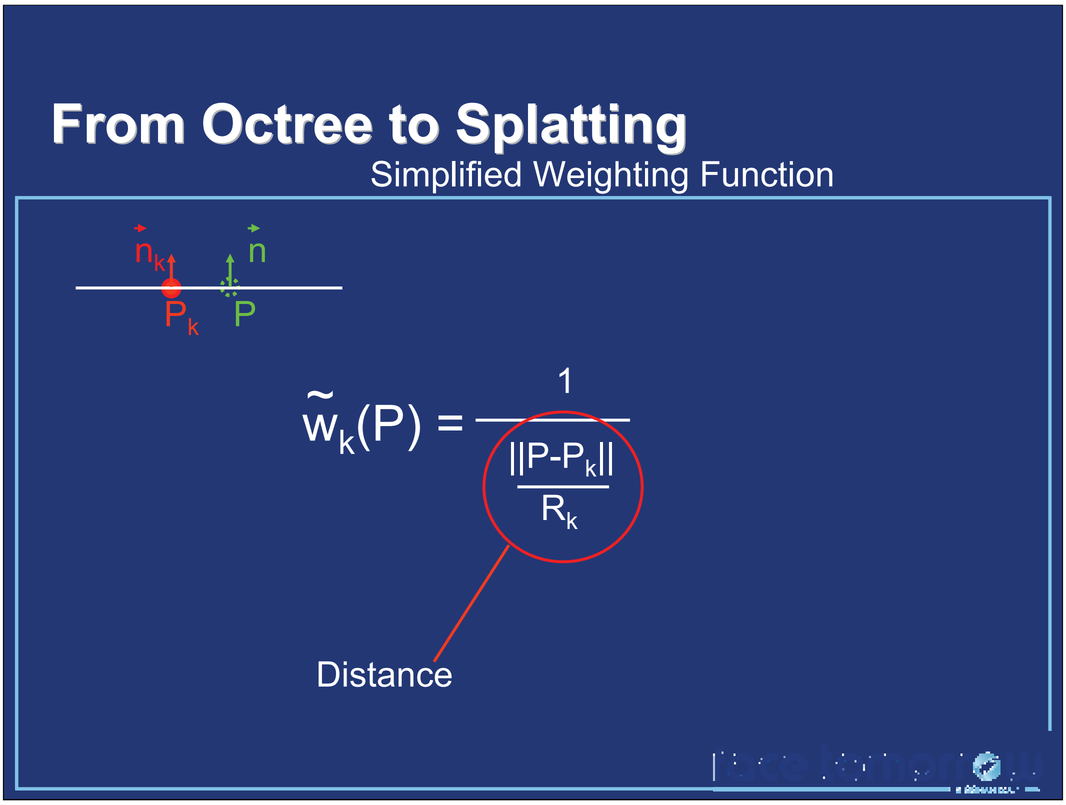


From Octree to Splatting

Simplified Weighting Function

$$w_k(P) = \frac{1}{\underbrace{\frac{\|P-P_k\|}{R_k}}_{\text{Distance}} + \underbrace{\sqrt{1-\vec{n}.\vec{n}_k}}_{\text{Normals divergence}}}$$

If we assume that the record location and the point P **always have the same normal**, the weighting function can be simplified by removing the dependence to the surface normals.



From Octree to Splatting

Simplified Weighting Function

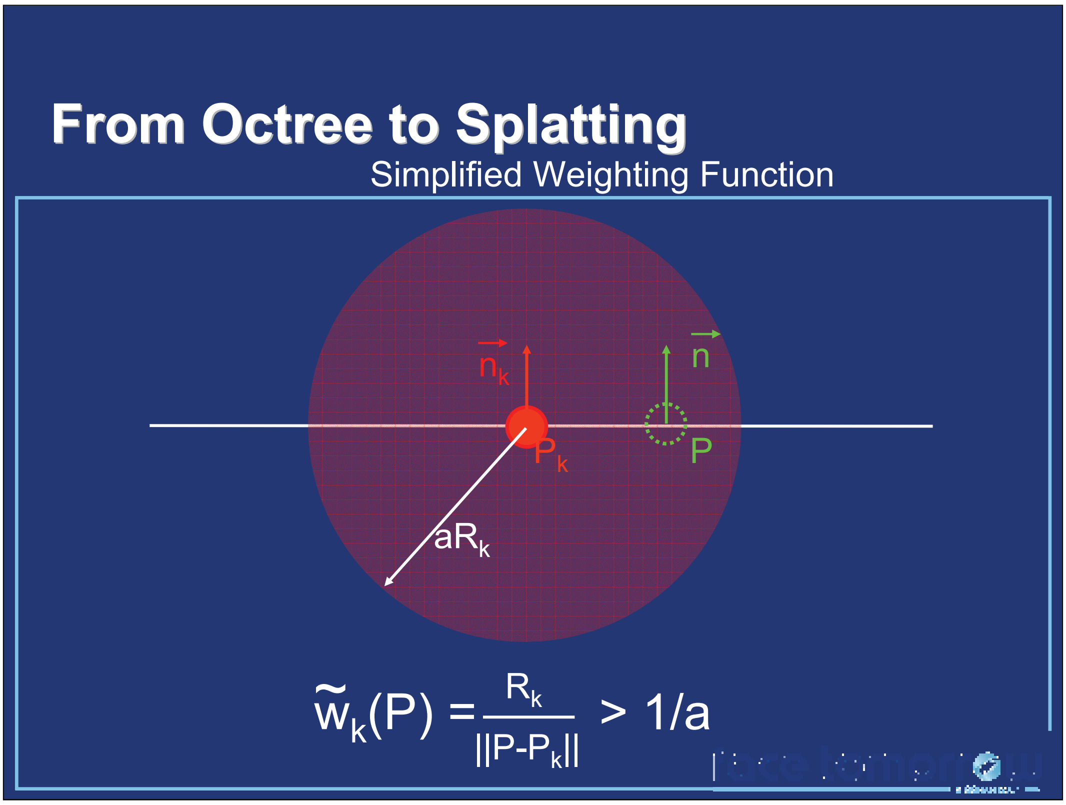
$$\tilde{w}_k(P) = \frac{1}{\frac{\|P-P_k\|}{R_k}}$$

Distance

This yields a simplified weighting function, which depends only on two factors:

-The distance between the record location and the point P

-The mean distance to the surrounding objects



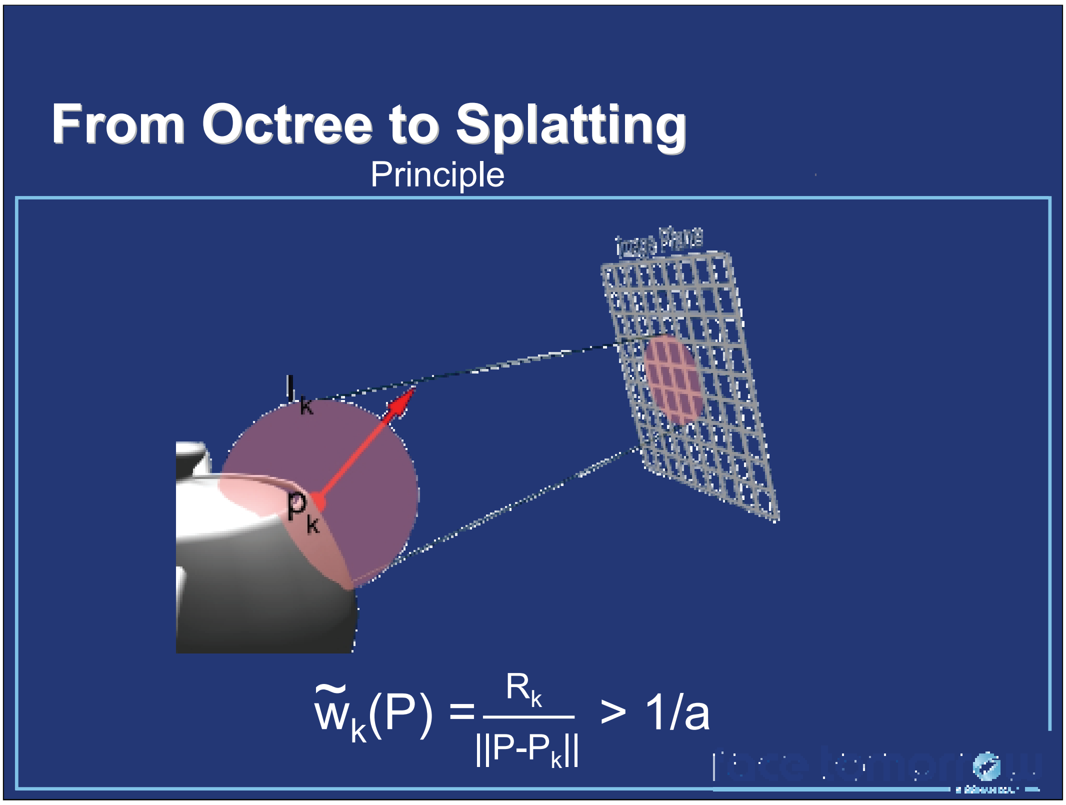
From Octree to Splatting

Simplified Weighting Function

$$\tilde{w}_k(P) = \frac{R_k}{||P-P_k||} > 1/a$$

Using this simplified weighting function, a record k contributes to **all** points located within a sphere centered at the record location, with radius aRk. Hence this radius depends not only on the user-defined parameter a, but also on the mean distance to the surrounding objects.

Note that the simplified weighting function removes the constraint on the surface normals. Therefore, the set of points at which the record actually contributes (with respect to the full weighting function) is a subset of the points located within the sphere.

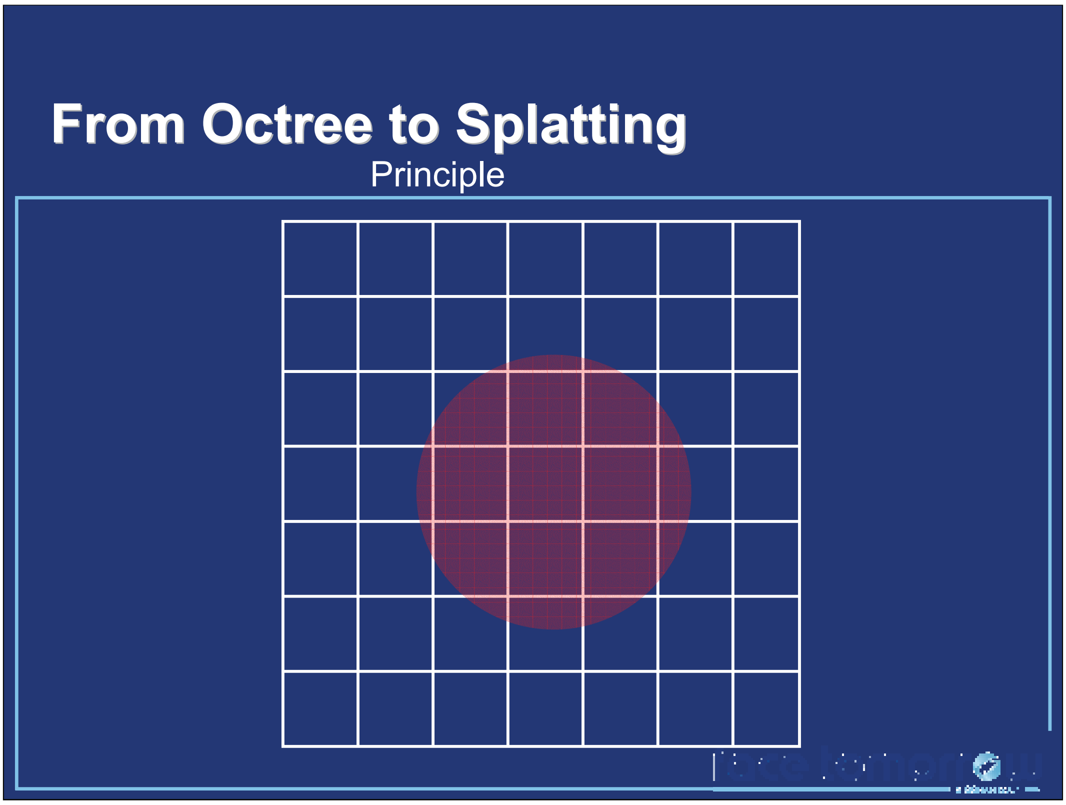


From Octree to Splatting

Principle

$$\widetilde{w}_k(P) = \frac{R_k}{||P-P_k||} > 1/a$$

The sphere can be splatted onto the image plane. Hence the covered pixels correspond to the visible points at which the record may contribute.

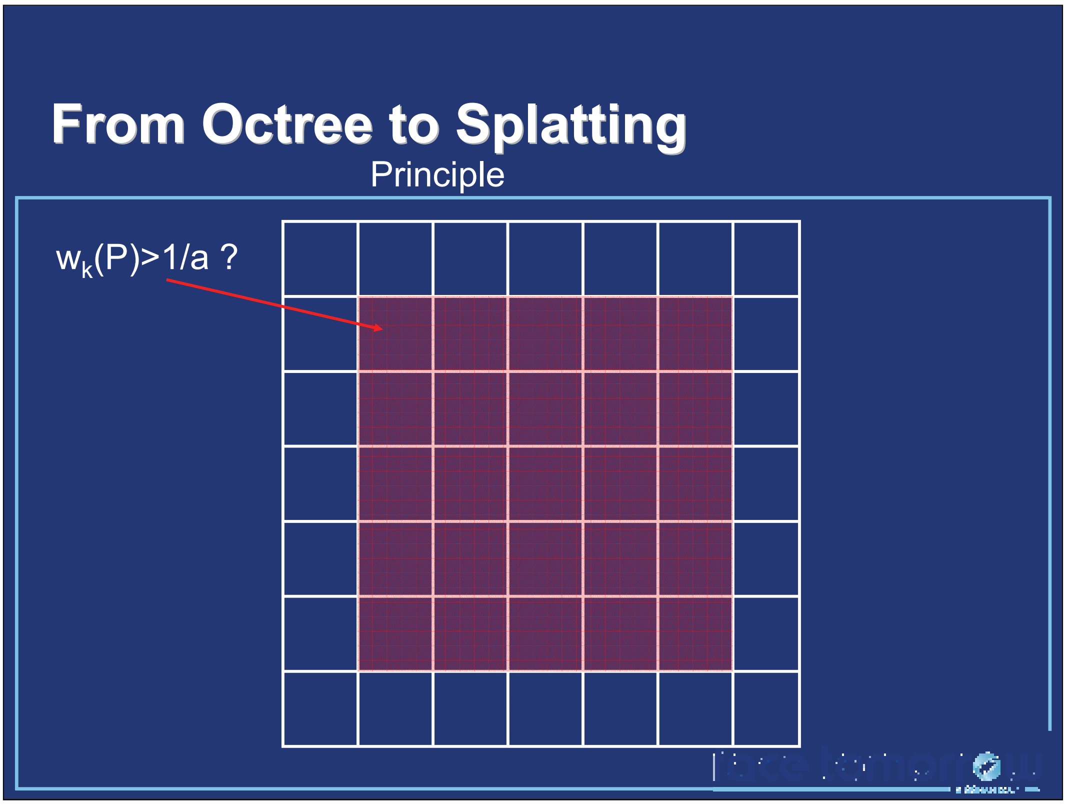


From Octree to Splatting

Principle

Let us consider the image plane, on which the sphere has been splatted. The splatted sphere encloses **all** the visible points at which the considered record may contribute (with respect to the full weighting function). Our goal is now to select the points for which the condition on the **full** weighting function is satisfied, that is

wk(P) > 1/a .

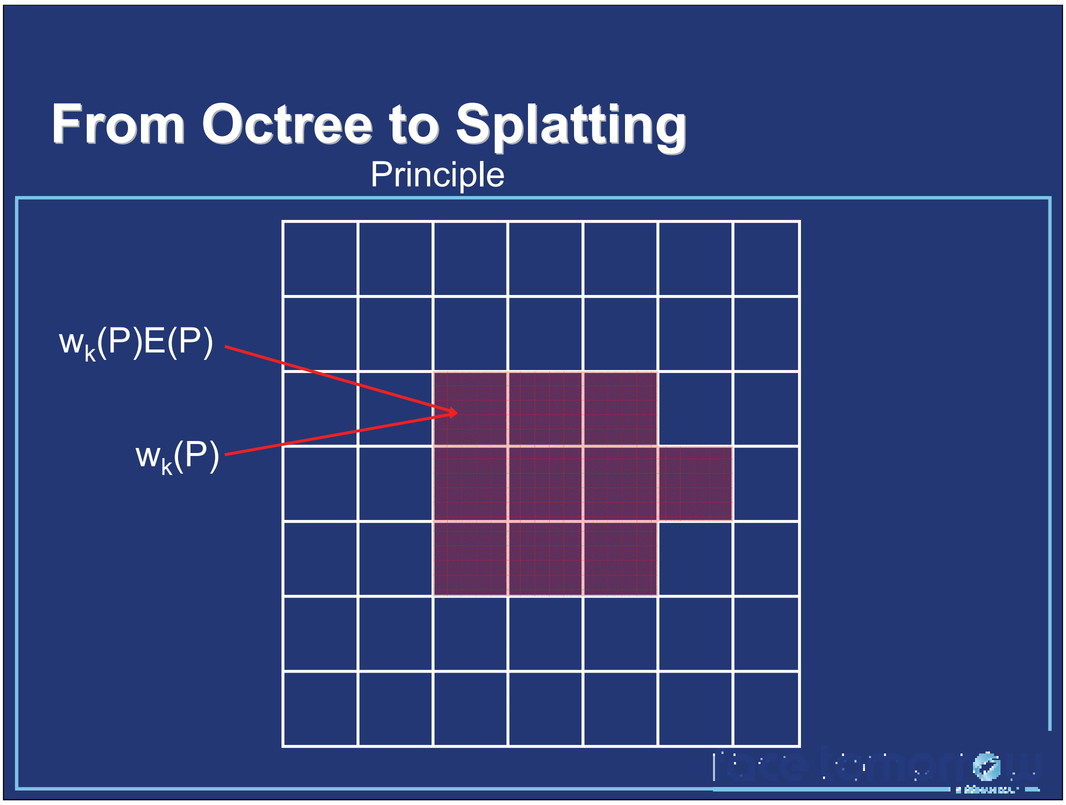


From Octree to Splatting

Principle

$w_k(P)>1/a$?

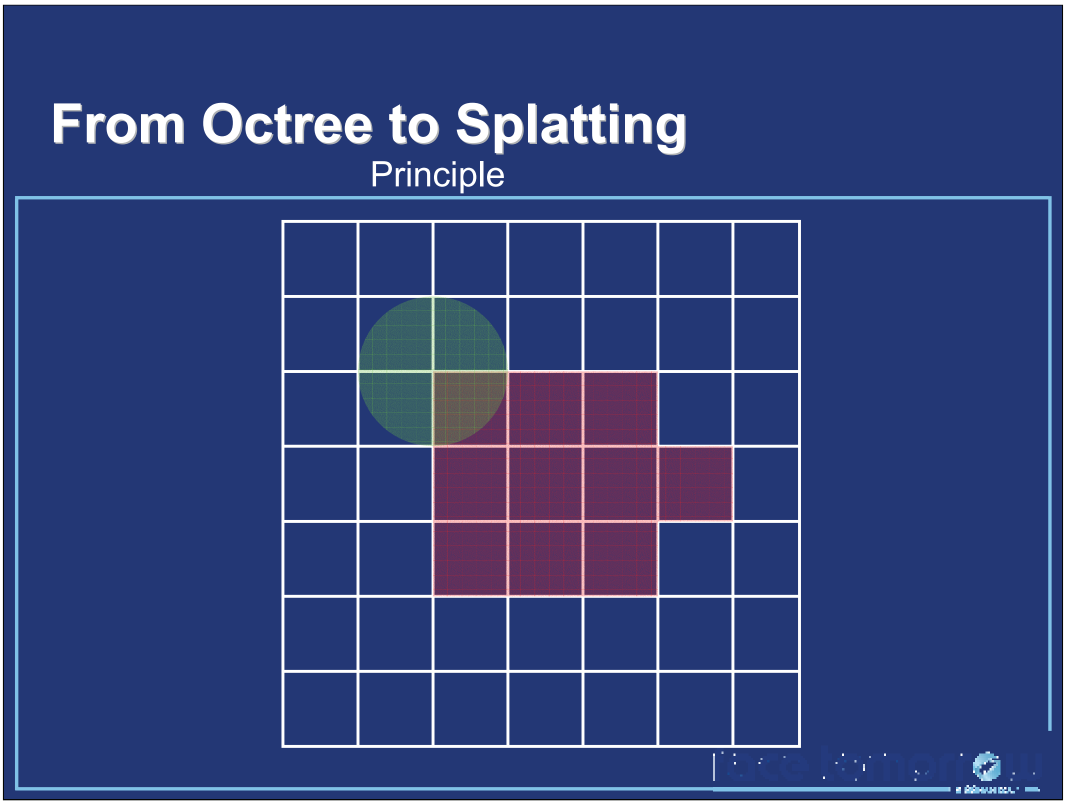
For the convenience of implementation, we use a quadrilateral tightly enclosing the splatted sphere. For each point visible through the pixels of the quadrilateral, the full weighting function is evaluated, and tested against the user-defined threshold. If the condition is not satisfied, the pixel is discarded.



From Octree to Splatting

Principle

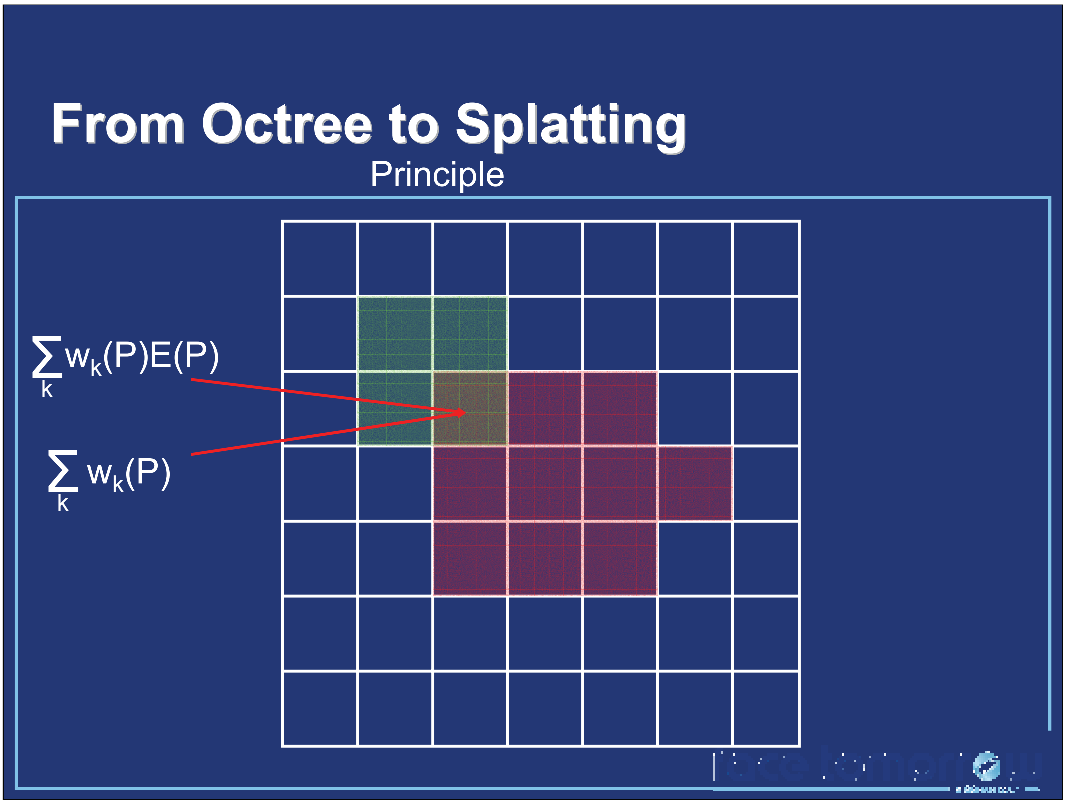
This yields a set of pixels, corresponding to the set of visible points at which the record actually contributes. At those points, we compute separately the weighted contribution of the record (with respect to the full weighting function and to the irradiance gradients) and the weight of the contribution. This information can be easily stored within floating point RGBA pixels, using RGB for the weighted contribution, and the alpha channel for the weight value.



From Octree to Splatting

Principle

When splatting an other record, the same operations are to be done. However, the zones of influence of the two records overlap.



From Octree to Splatting

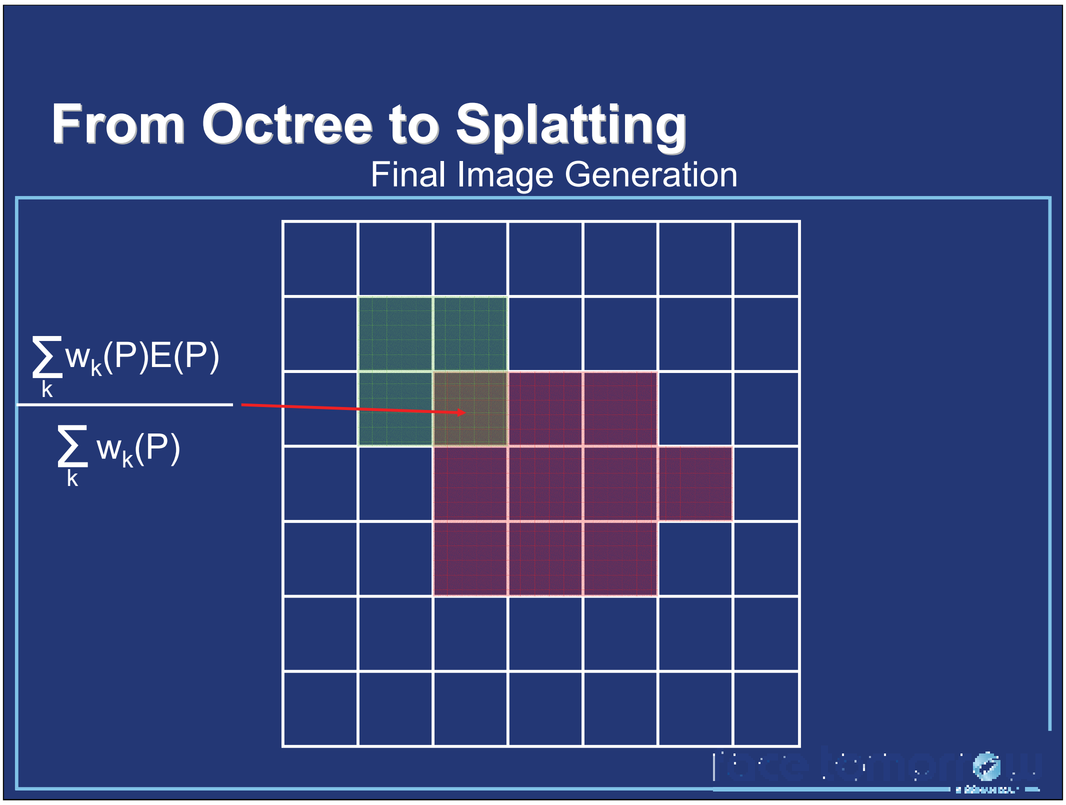
Principle

$\sum_k w_k(P)E(P)$

$\sum_k w_k(P)$

In this case, we first compute the weighted contribution and weight of the second record as described before. Then, the built-in alpha blending of graphics processors adds the contributions and weights together in the overlapping area.

Then, each pixel contains both the weighted sum of the contributions, and the sum of the contribution weights.

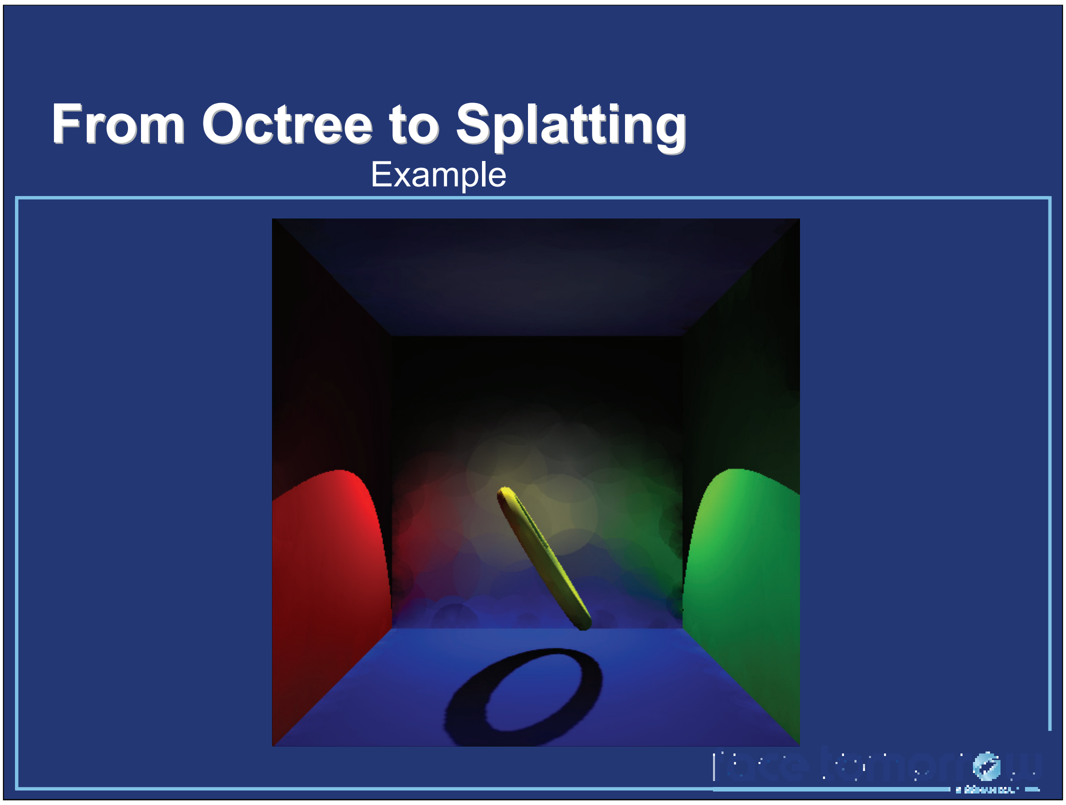


From Octree to Splatting

Final Image Generation

$$\frac{\sum_k w_k(P)E(P)}{\sum_k w_k(P)}$$

Once all the necessary records have been splatted, the result of the irradiance interpolation equation can then be obtained by dividing the weighted sum of contributions by the sum of the weights. This operation can be easily performed within a fragment shader, by dividing the RGB components (that is, the weighted sum of contributions) by the alpha channel (the sum of weights).

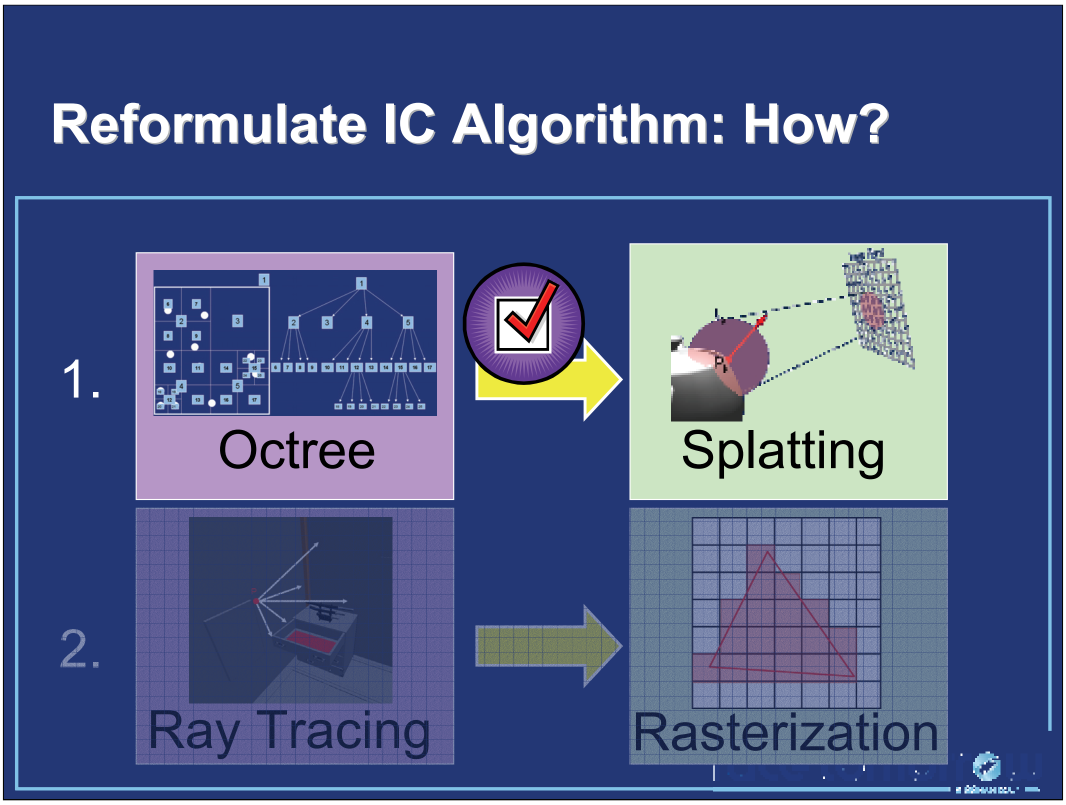


From Octree to Splatting

Example

To summarize, let us consider an example. At the beginning of the algorithm, the cache contains no records. Hence the generated image only features direct illumination. Then, we add a record on the back wall. This record contributes to the points within its neighborhood. When adding a second record, the zones of influence overlap. For the pixels within the overlapping area, the estimated irradiance is calculated using the weighted average of the contributions. Other records are successively added to the cache, until the entire visible area of the scene is covered by the zones of influence of the records. The resulting image features both direct and indirect lighting for every visible point.

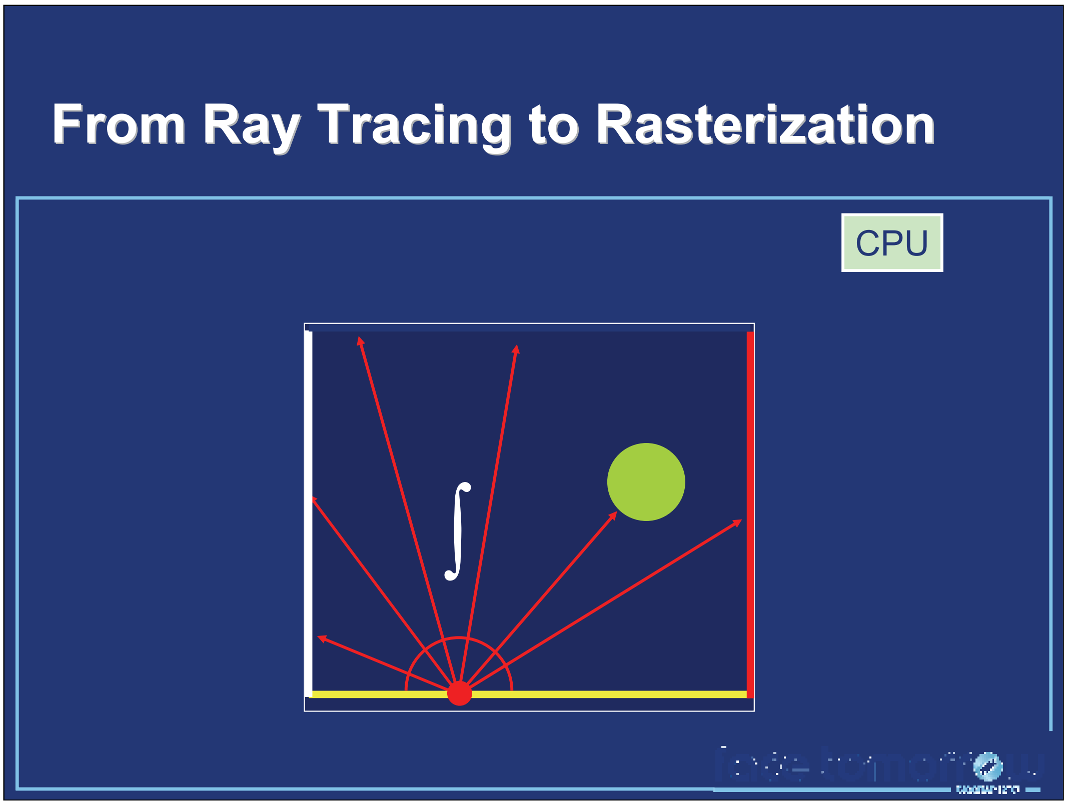
Note that for explanation purposes, the reconstruction of the indirect lighting is very coarse to highlight the zones of influence of each record. Also, the gradients are not used. As with the classical irradiance caching algorithm, high quality can be obtained by using irradiance gradients and setting an appropriate value for the user-defined parameter a.



Reformulate IC Algorithm: How?

1. Octree → Splatting

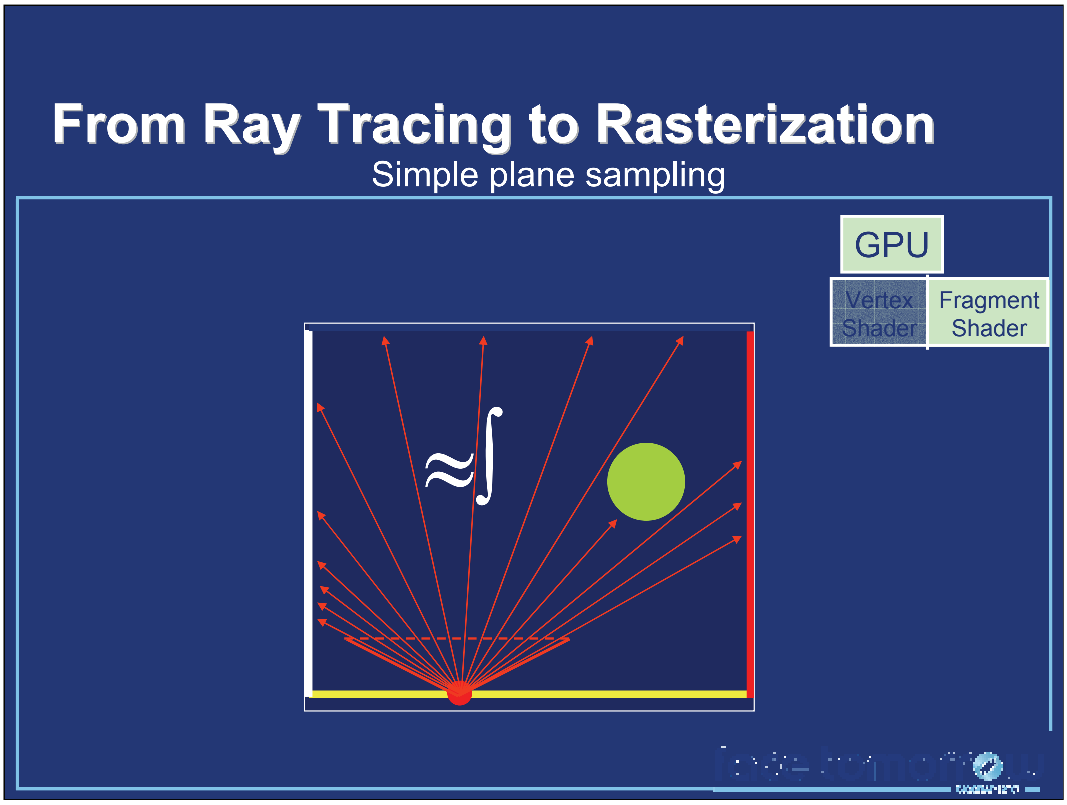
2. Ray Tracing → Rasterization



From Ray Tracing to Rasterization

CPU

In classical irradiance caching, the irradiance at a point is estimated by Monte Carlo ray tracing: random rays are traced through the scene, gathering the lighting incoming from the surrounding environment. This estimate is usually computed on the CPU.



From Ray Tracing to Rasterization

Simple plane sampling

GPU

Vertex Shader

Fragment Shader

A well-known approximate method for hemisphere sampling on the GPU is the simple plane sampling: a virtual camera with a large aperture is placed at the point of interest. The scene is then rendered on graphics hardware, yielding an image of the surrounding objects. On recent graphics hardware, this data can be computed in high dynamic range (HDR) using floating-point render target and programmable shaders. The shadowing effects can be efficiently accounted for using fast shadowing techniques such as shadow mapping [Wil78].

The solid angle subtended by a pixel p is defined as $\Omega_p=A*\cos(\theta)/(d*d)$ where:

•A is the surface of a pixel

•θ is the angle between the direction passing through the pixel and the surface normal

•d is the distance between the point of interest and the image plane

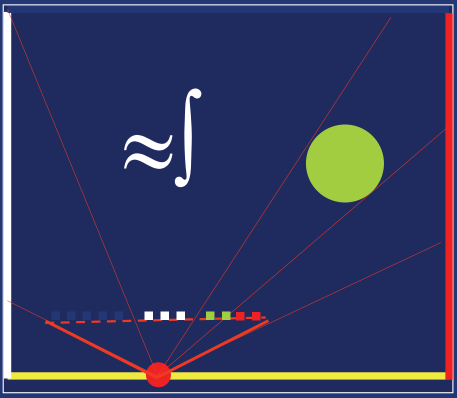
From Ray Tracing to Rasterization

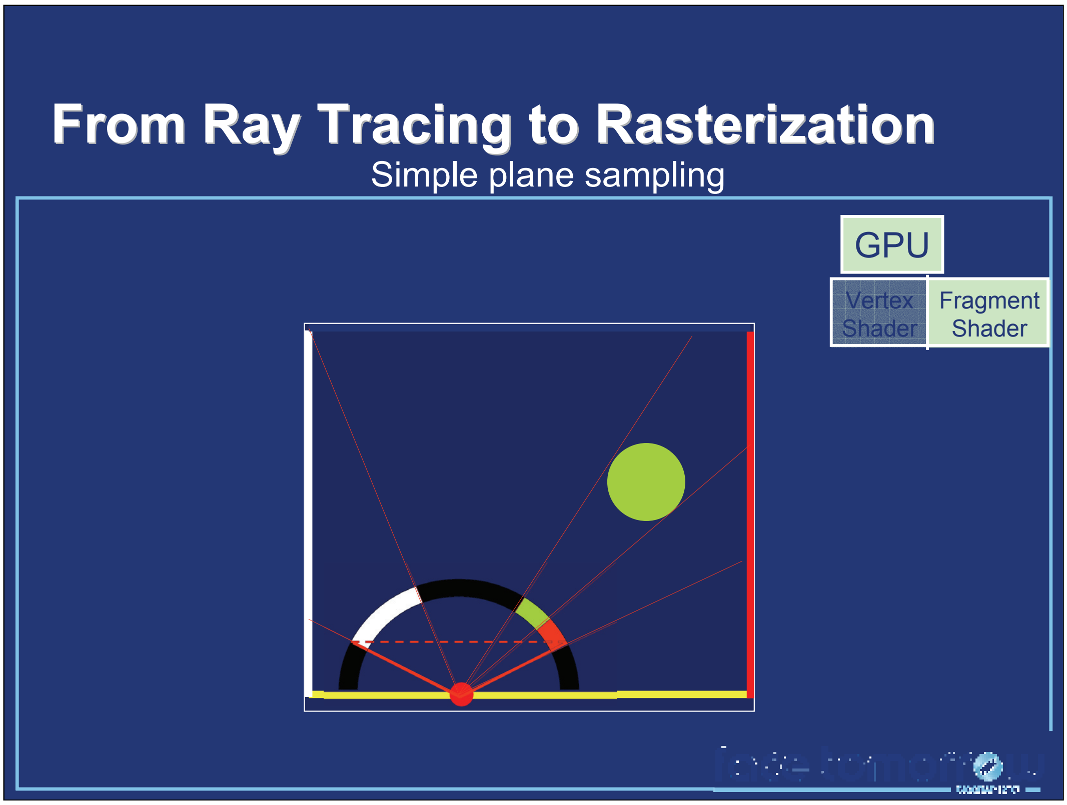
Simple plane sampling

GPU

Vertex Shader

Fragment Shader





From Ray Tracing to Rasterization

Simple plane sampling

However, the aperture of the camera cannot allow us to sample the entire hemisphere: a perspective camera is represented by a perspective projection matrix which is applied to the visible contents of the scene. In OpenGL, such a camera is modeled using the gluPerspective function, whose values are calculated using the field of view (FOV) of the camera. More precisely, a key value in this matrix is f = cot(FOV/2), which is undefined for FOV=180°. Furthermore, using a very large aperture such as 179.999° leads to important perspective deformation and sampling problems.

In [LC04], Larsen et al. show that an aperture of 126.87° is sufficient for capturing 86% of the incoming directions. However, the remaining 14% are unknown and must be compensated to avoid a systematic underestimation of the incoming radiance. Furthermore, this compensation must respect the directional information of the incoming radiance to allow for a later implementation of radiance caching for global illumination computation on glossy surfaces.

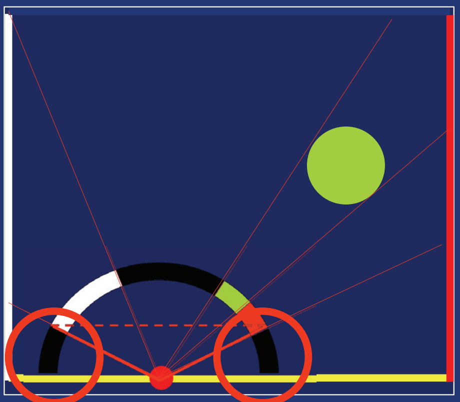
From Ray Tracing to Rasterization

Simple plane sampling

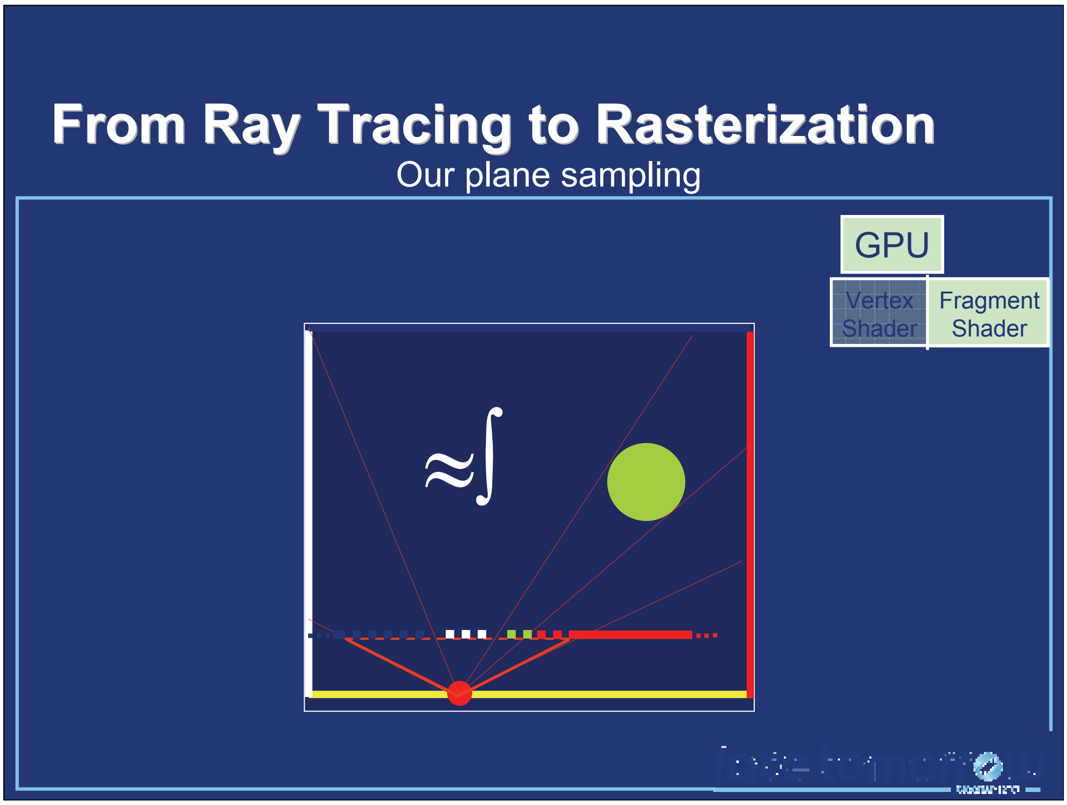
GPU

Vertex Shader

Fragment Shader



Incoming radiance loss



From Ray Tracing to Rasterization

Our plane sampling

GPU

Vertex Shader

Fragment Shader

We propose a very simple compensation method, in which the border pixels are virtually « extended » to fill the parts of the hemisphere which have not been actually sampled. To this end, border pixels are considered as covering a solid angle of:

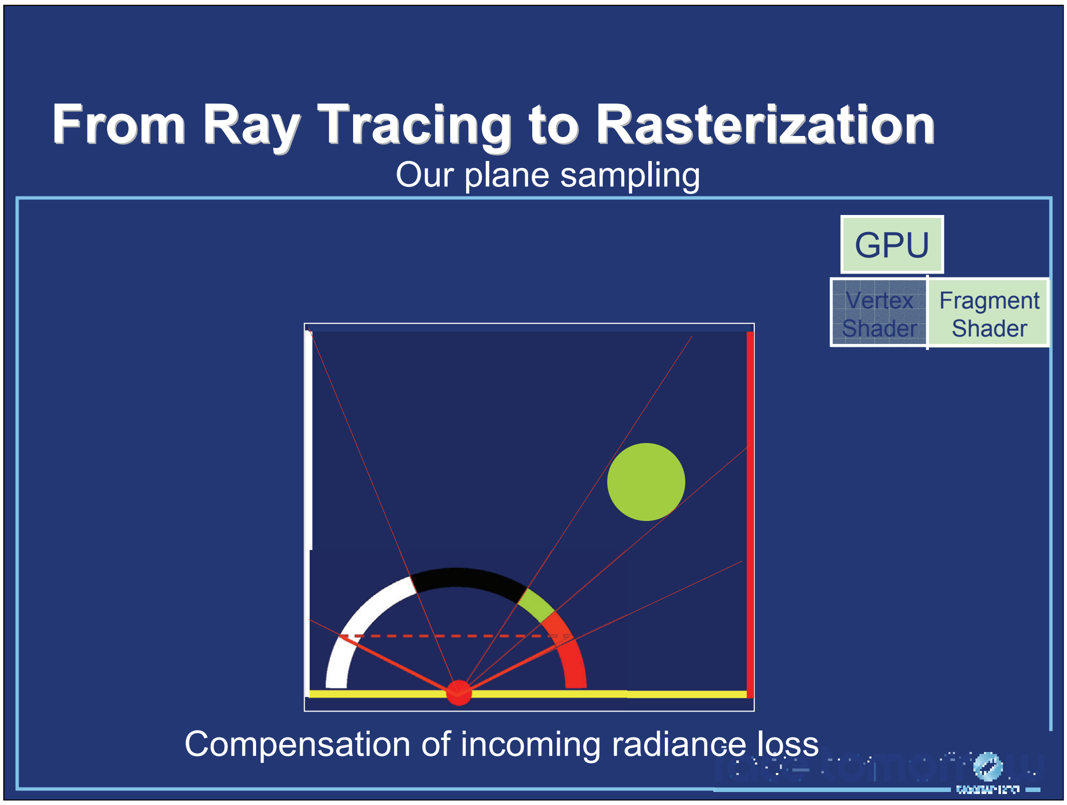
$\Omega_{border} = \Omega_p + \cos(\theta_{border}) * \delta_\Phi$

Where:

•Ω_p is the solid angle subtended by the pixel

•θ_{border} is the largest θ covered by the aperture of the camera

•δ_Φ is the interval of Φ spanned by the pixel



From Ray Tracing to Rasterization

Our plane sampling

GPU

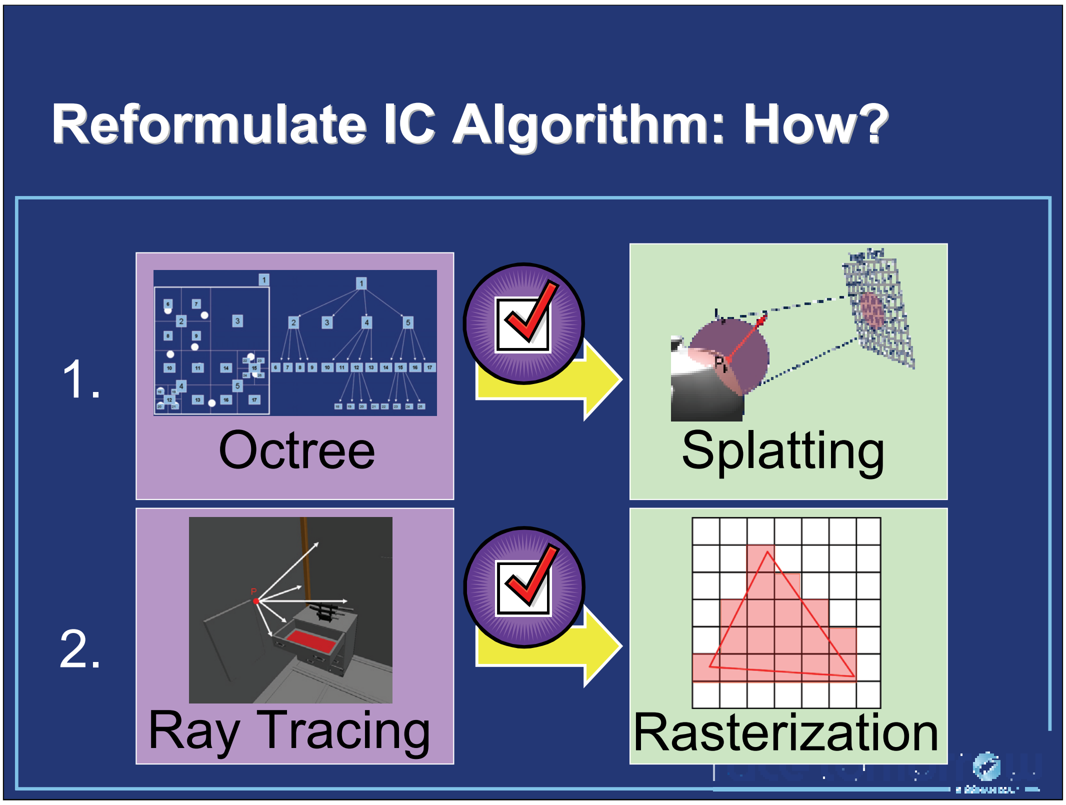
Vertex Shader

Fragment Shader

Compensation of incoming radiance loss

This allows us to compensate the missing information by extrapolating the radiance values at the extremities of the sampling plane. This extrapolation provides a plausible estimate of the missing radiances, hence making it suitable for radiance caching also.

It must be noted that other techniques can be used, such as the full hemicube sampling (which requires several passes), or a hemispherical parametrization of the hemisphere in vertex shaders.

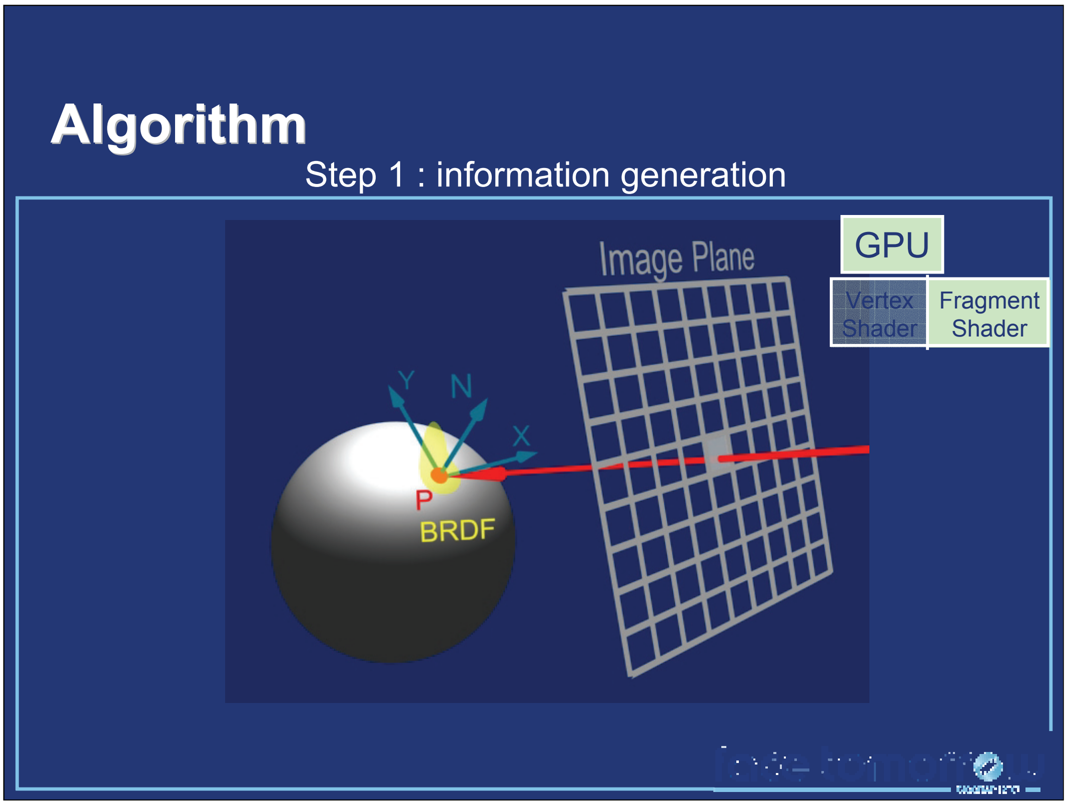


Reformulate IC Algorithm: How?

1. Octree → Splatting

2. Ray Tracing → Rasterization

Now the algorithm has been reformulated for implementation on graphics hardware. Next section will present the entire algorithm for global illumination computation using irradiance splatting.



Algorithm

Step 1 : information generation

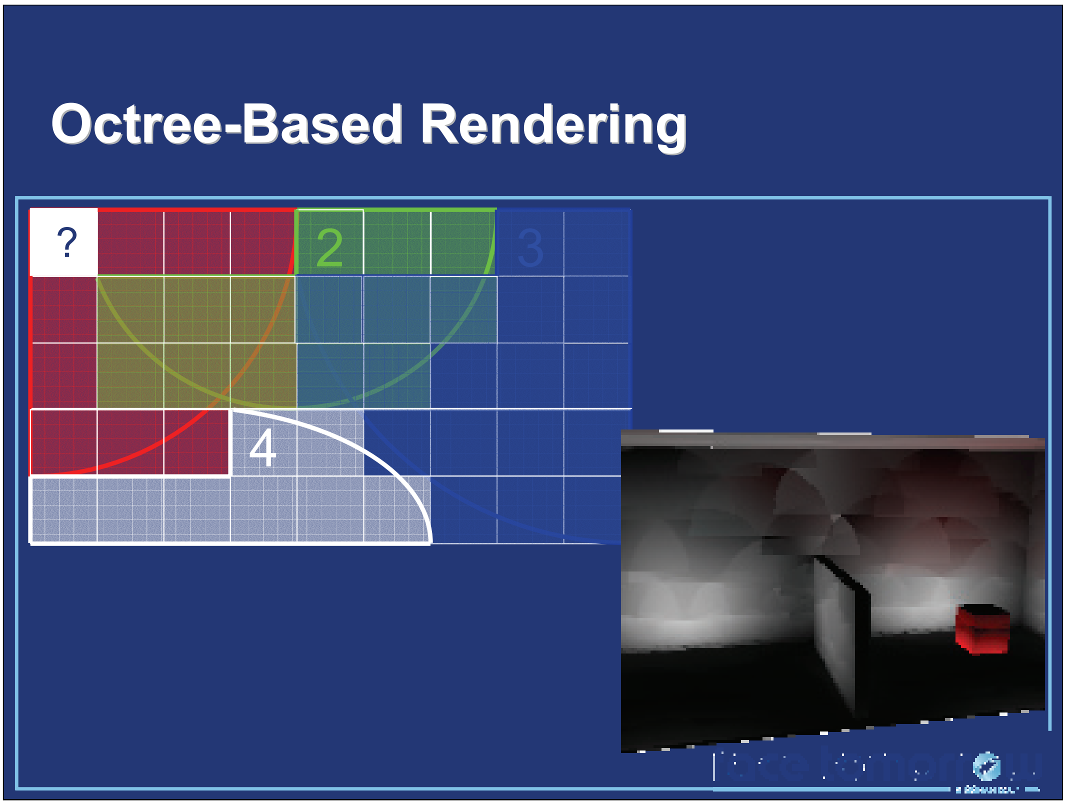
GPU

Vertex Shader

Fragment Shader

The first step of the algorithm consists in obtaining basic information about the points visible from the user point of view. In particular, this information includes the position and normal of the visible points. This data can be easily generated in one pass on the GPU using floating-point multiple render targets and programmable shaders.

In the next step, the algorithm transfers this data into the main memory. The CPU is then used to traverse the image and detect where new records are required to render the image.

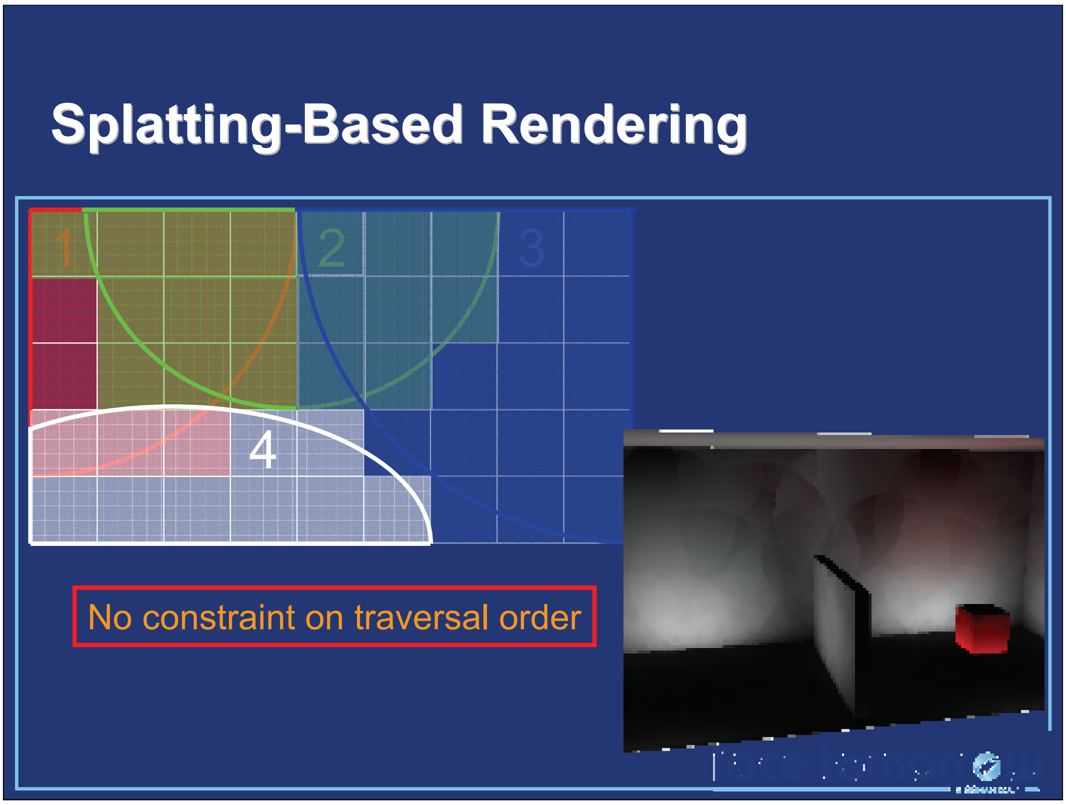


Octree-Based Rendering

Usually, the detection is performed by querying the irradiance cache (hence the octree) for each pixel as follows:

```
•For each visible point P with normal N corresponding to pixel p
    •W = GetSumOfContributions(P,N)
    •If (W < a)
        •R = CreateNewRecord(P,N)
        •IrradianceCache.StoreRecord(R)
        •p.radiance = R.irradiance*SurfaceReflectance(P)
    •Else
        •E = EstimateIrradiance(P,N)
        •P.radiance = E*SurfaceReflectance(P)
    •EndIf
•EndFor
```

While this algorithm ensures the presence of a sufficient number of records, such records are not used in an optimal way: when record 2 is created, the algorithm only propagates the contribution of 2 to the pixels which have not been checked yet. The previous pixels remain unchanged, even though record 2 may contribute to their radiance. If the image is traversed linearly, disturbing artifacts may appear: in this example the image is traversed from bottom to top. Therefore, the records contribute only to the points located above them in the image. This problem is usually compensated by using other traversal algorithms, typically based on a hierarchical subdivision of the image.



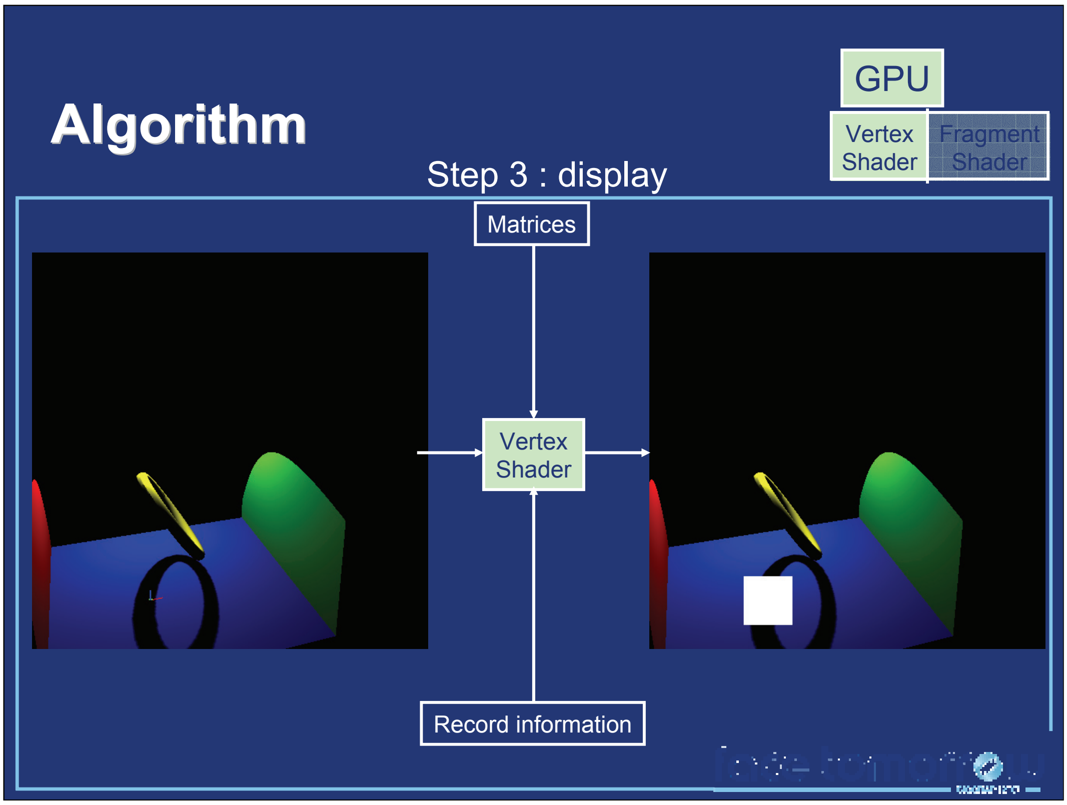
Splatting-Based Rendering

No constraint on traversal order

When using irradiance splatting, the detection code becomes:

```
•For each visible point P with normal N corresponding to pixel p
    •W = GetSumOfContributions(P,N)
    •If (W < a)
        •R = CreateNewRecord(P,N)
        •IrradianceCache.StoreRecord(R)
        •SplatRecord(R)
        •p.radiance = R.irradiance*SurfaceReflectance(P)
    •Else
        •// Nothing to do: the radiance value depends of surrounding records and may
        be updated
    •EndIf
•EndFor
```

In our method, the records are splatted onto their entire zone of influence: even pixels which have been previously checked can be updated by the addition of a novel record. Hence this method removes the constraint on the image traversal order. In the example image, we used a simple linear traversal. The zone of influence of each record is completely accounted for by our splatting method.



Algorithm

GPU

Vertex Shader | Fragment Shader

Step 3 : display

Matrices

Vertex Shader

Record information

Once all records have been computed, each record must be rendered on the GPU. Let us consider a record located at point P with normal N, and with an harmonic mean distance to surrounding objects H. The zone of influence of the record is contained within a screen-aligned square. Let us consider the vertices of a square such that:

v0 = (1.0, 0.0, 0.0)

v1 = (1.0, 1.0, 0.0)

v2 = (0.0, 1.0, 0.0)

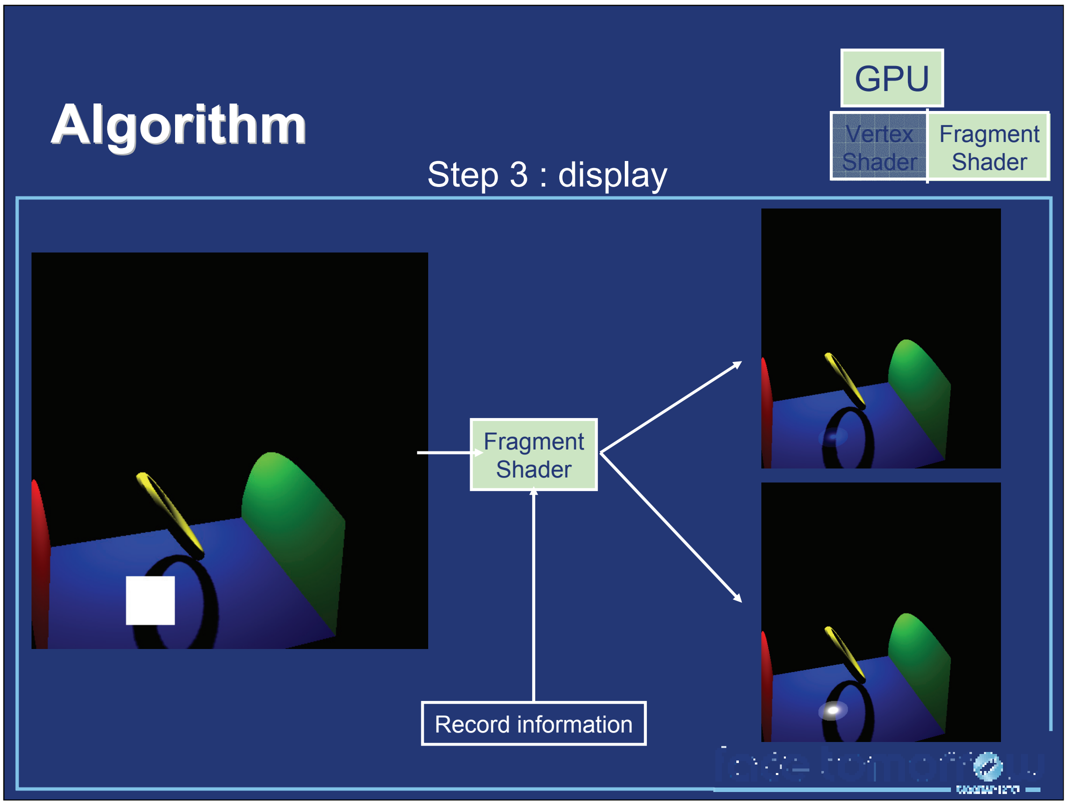
v3 = (0.0, 0.0, 0.0)

Each vertex can be transformed into screen space using the following function:

```
float4 TransformVertex(Vertex v)
{

                float4 projPos = ModelViewMatrix*P; // Transform the point into
camera space
                float scale = projPos.w;
                float4 projPos /= projPos.w;

                float radius = a*H; // Radius of the zone of influence in camera
space
```



Algorithm

Step 3 : display

GPU

Vertex Shader

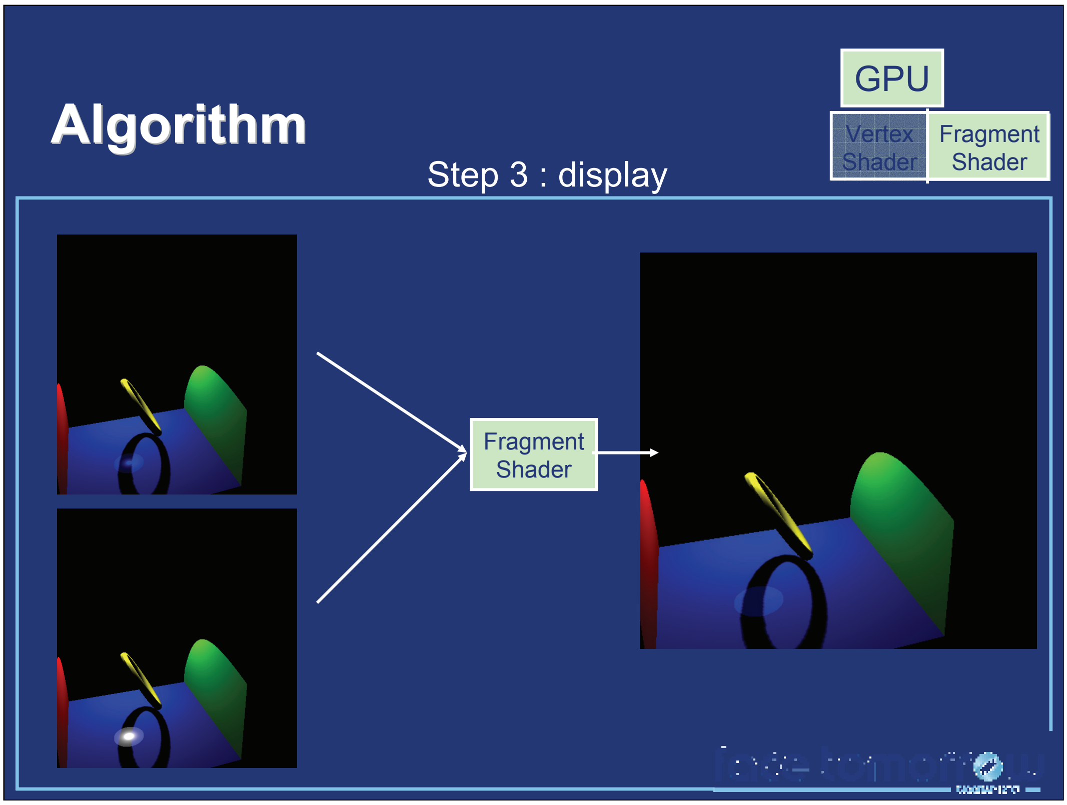
Fragment Shader

Fragment Shader

Record information

The fragment shader then computes the actual value of the weighting function for each point within the quadrilateral, using the formula defined in [Ward88]. If the record is allowed to contribute to the lighting of a visible point, its contribution is computed using irradiance gradients such as in [Ward92]. The output of the shader for the corresponding fragment is a HDR RGBA value, where RGB represents the weighted contribution of the record, and A represents the weight of the record's contribution.

When several records are rendered, their RGBA values are simply added together using the floating-point alpha blending of graphics hardware (glBlendFunc(GL_ONE, GL_ONE)). This yields a sum of weighted contributions in the RGB components, and a sum of weights in the A component.



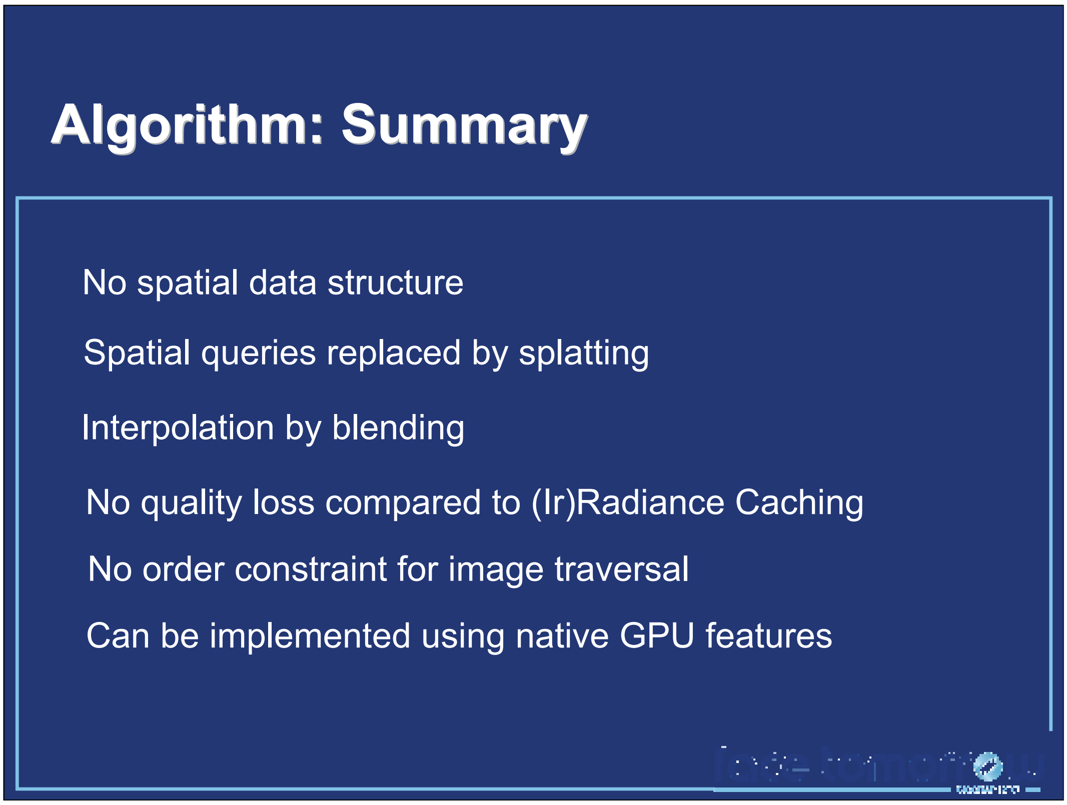
Algorithm

GPU

Vertex Shader | Fragment Shader

Step 3 : display

In a final step, a texture containing the RGBA values discussed above is used in a final render pass. This pass renders a single screen-sized quadrilateral. For a given pixel the fragment shader fetches the corresponding RGBA value, and outputs RGB/A to perform the irradiance estimation described in [Ward88].



Algorithm: Summary

No spatial data structure

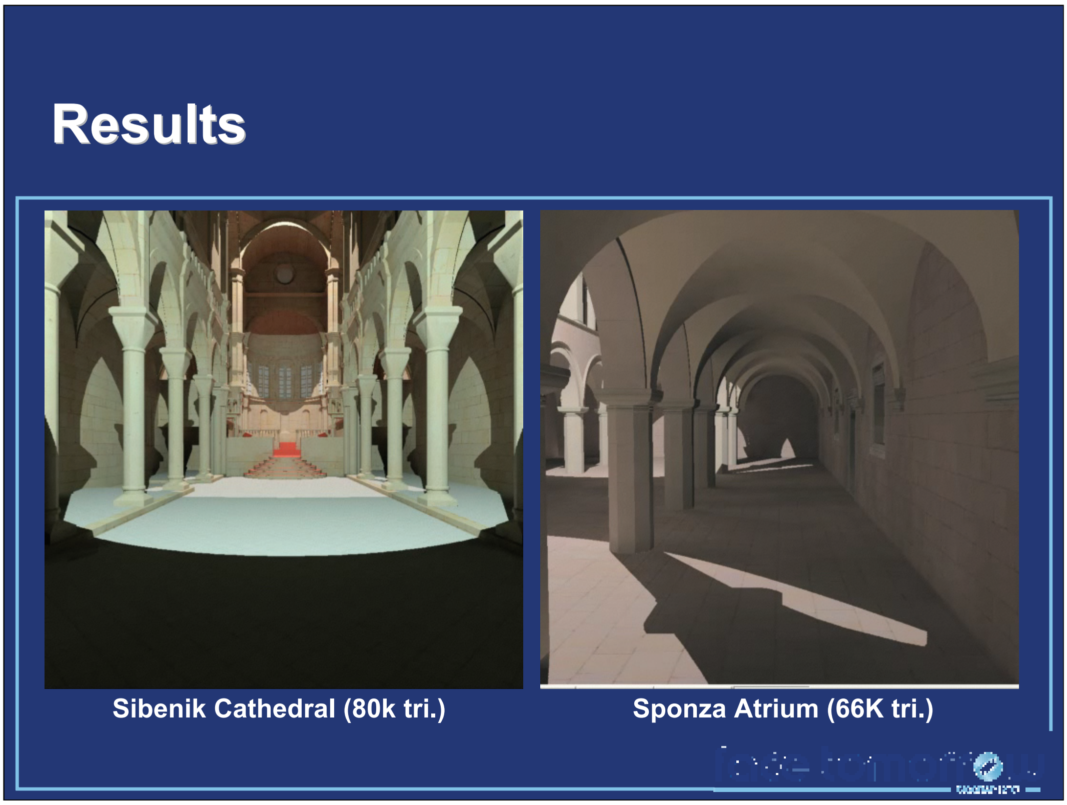
Spatial queries replaced by splatting

Interpolation by blending

No quality loss compared to (Ir)Radiance Caching

No order constraint for image traversal

Can be implemented using native GPU features

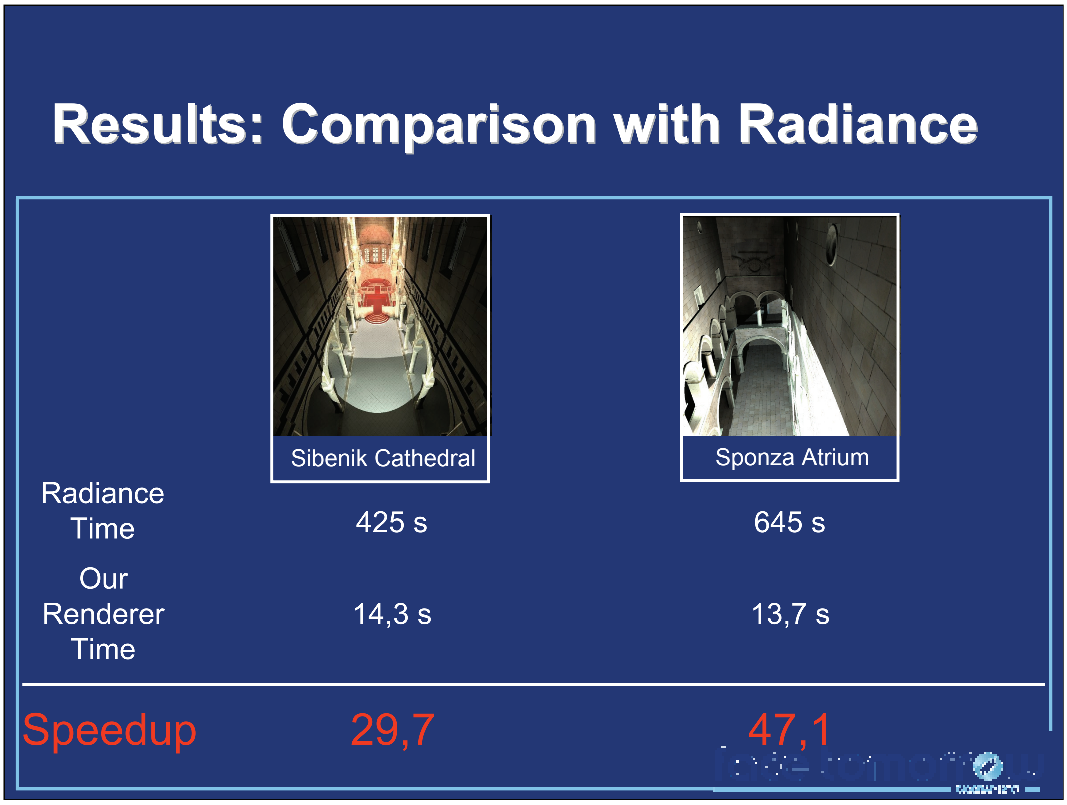


Results

Sibenik Cathedral (80k tri.)

Sponza Atrium (66K tri.)

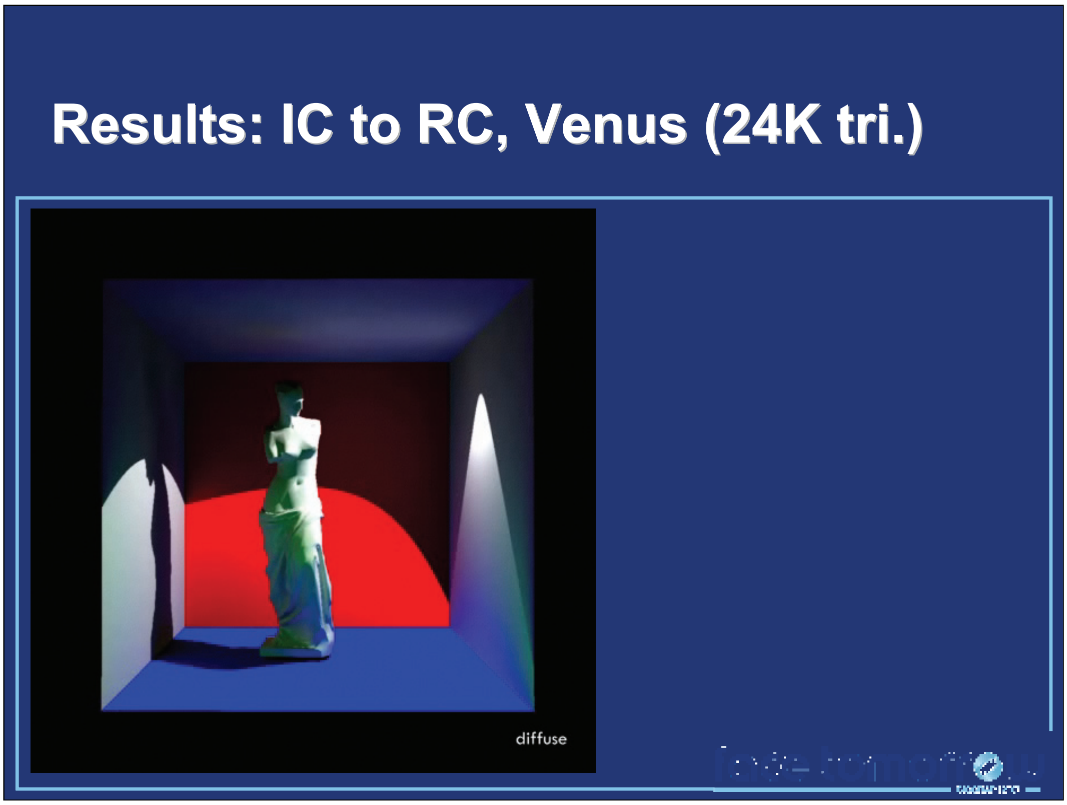
Those videos were rendered on a GeForce 6800. The scene being static, the irradiance cache is reused across frames. Hence the first frame has been computed in approximately 20s, while the other frames took approximately 1s to render.



Results: Comparison with Radiance

	Sibenik Cathedral	Sponza Atrium
Radiance Time	425 s	645 s
Our Renderer Time	14,3 s	13,7 s
Speedup	29,7	47,1

A comparison between our GPU-based method and the Radiance Software.



Results: IC to RC, Venus (24K tri.)

This method can be straightforwardly extended to glossy global illumination using radiance caching.

Temporal Coherence

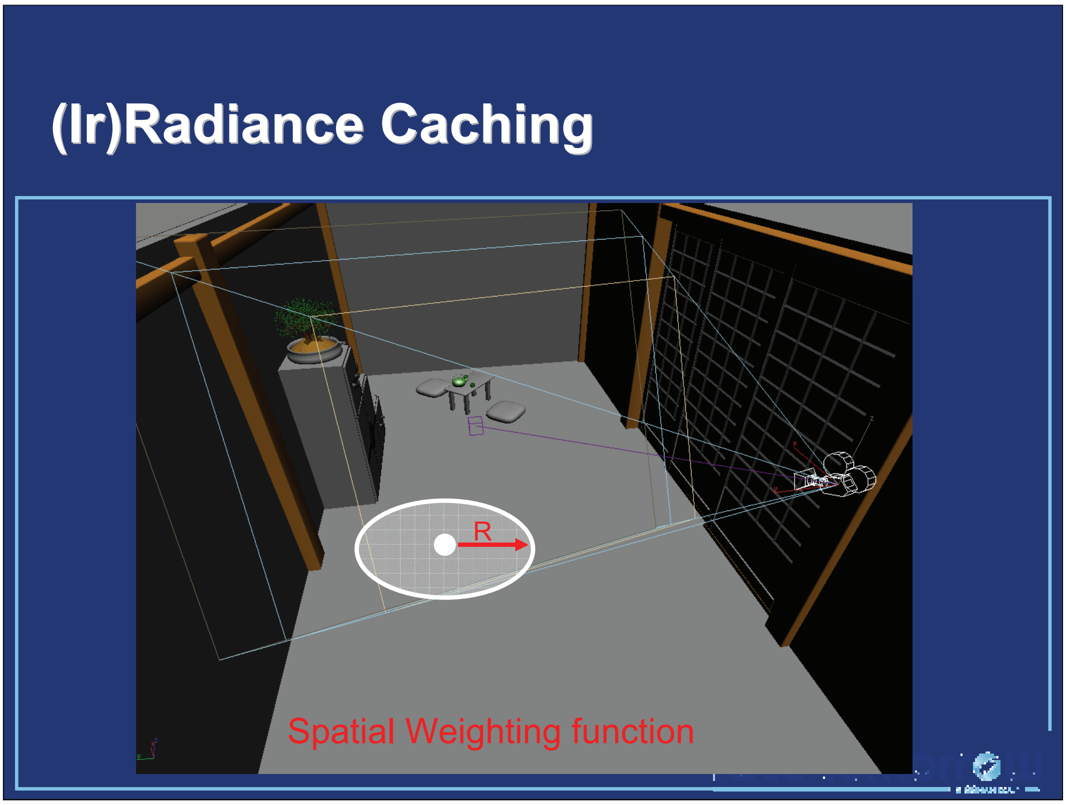
SIGGRAPH2007

Pascal Gautron

Post-Doctoral Researcher

France Telecom R&D Rennes

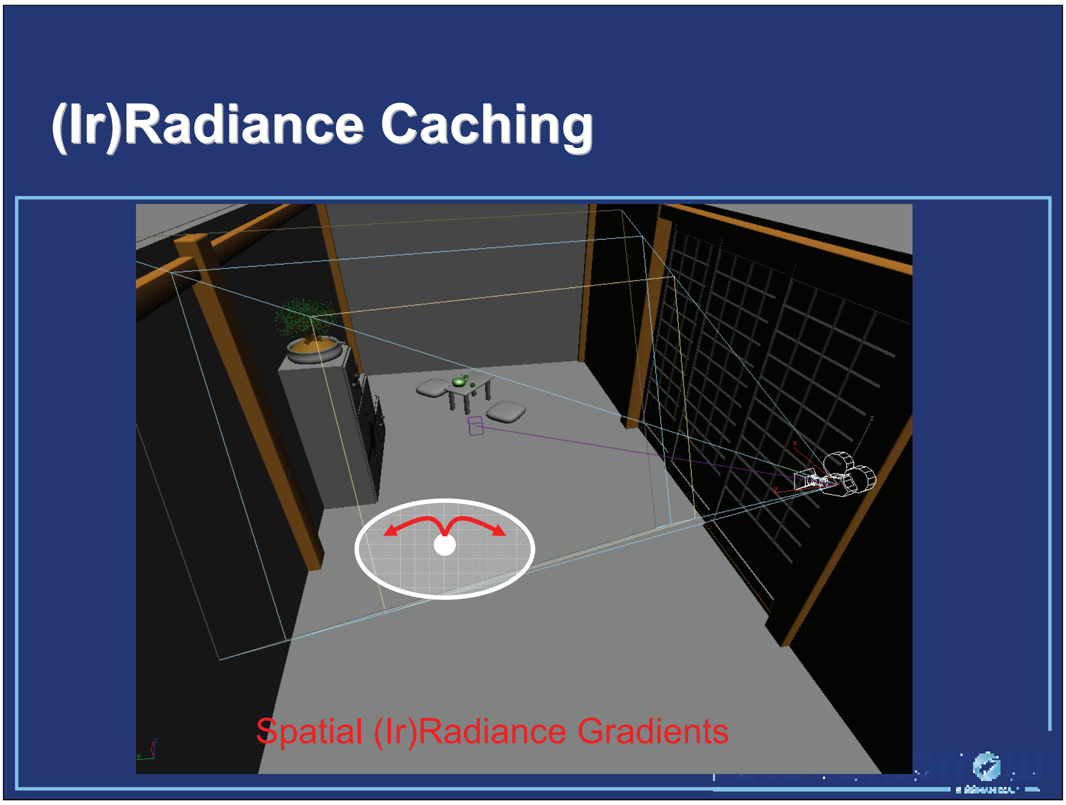
France



(Ir)Radiance Caching

Spatial Weighting function

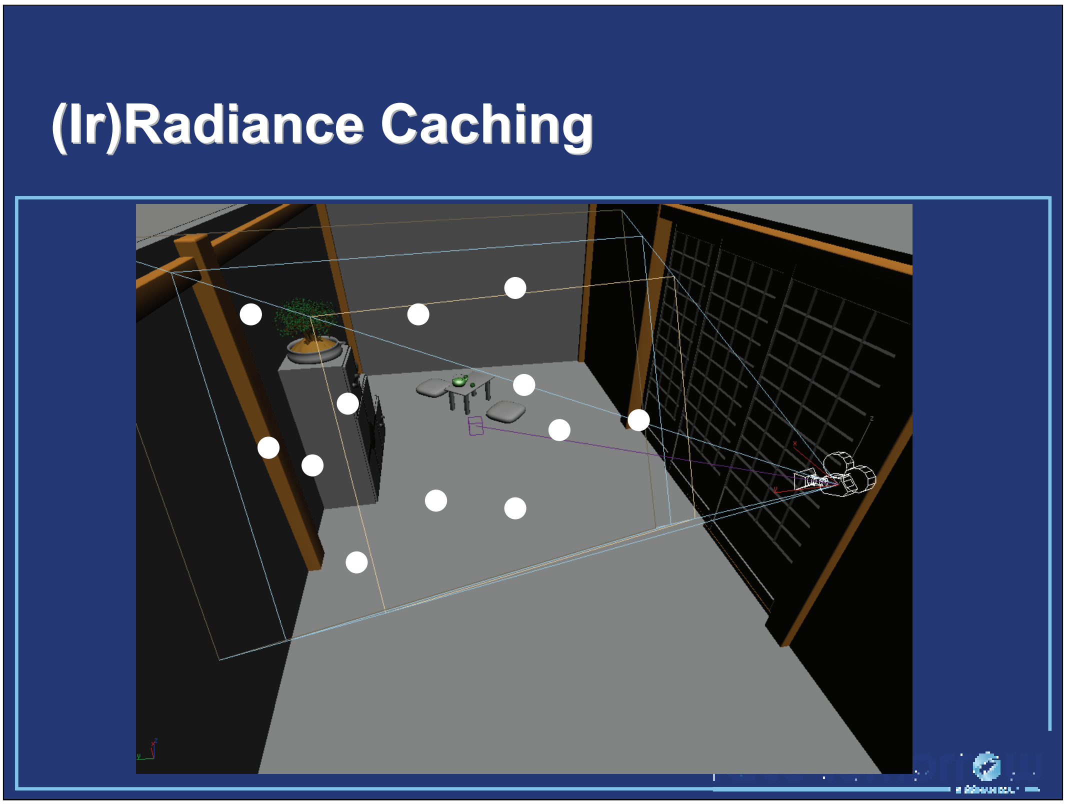
As explained in the previous parts of this course, the zone of influence of a given irradiance record is defined by a spatial weighting function.



(Ir)Radiance Caching

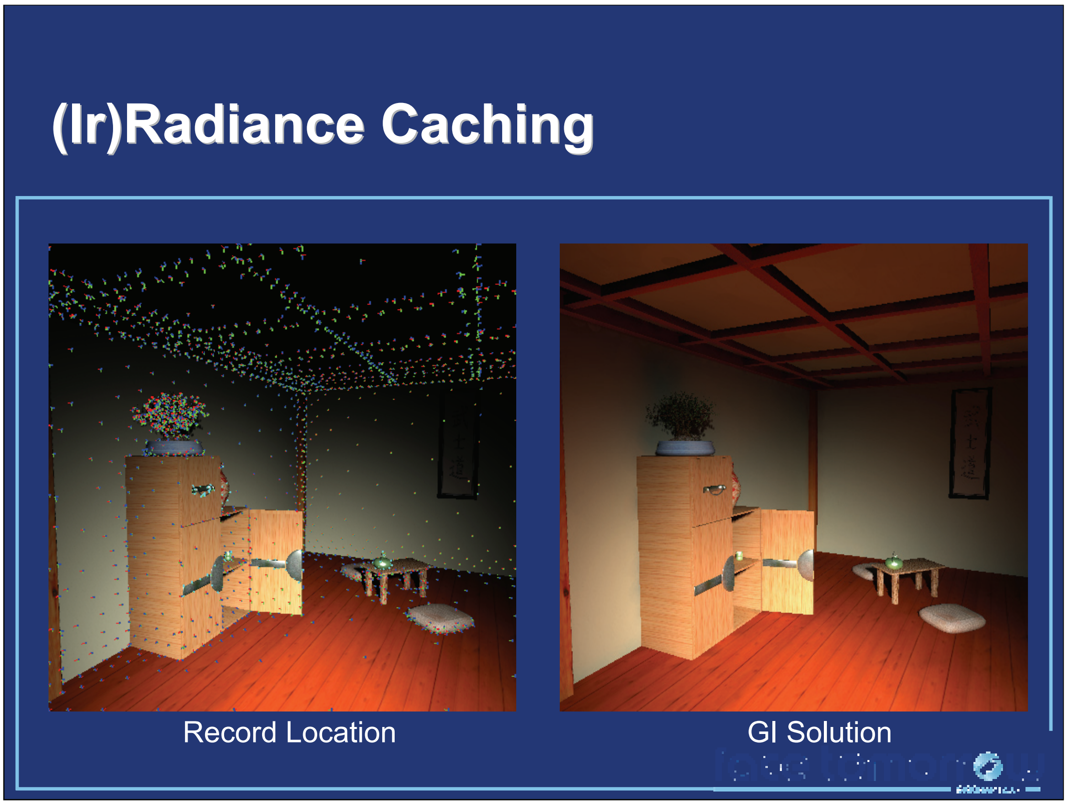
Spatial (Ir)Radiance Gradients

The contribution of a record within its zone of influence is estimated using irradiance gradients [Ward92].



(Ir)Radiance Caching

The lighting of the visible points is then computed explicitly at the location of the records, and **extrapolated** for all the other visible points.

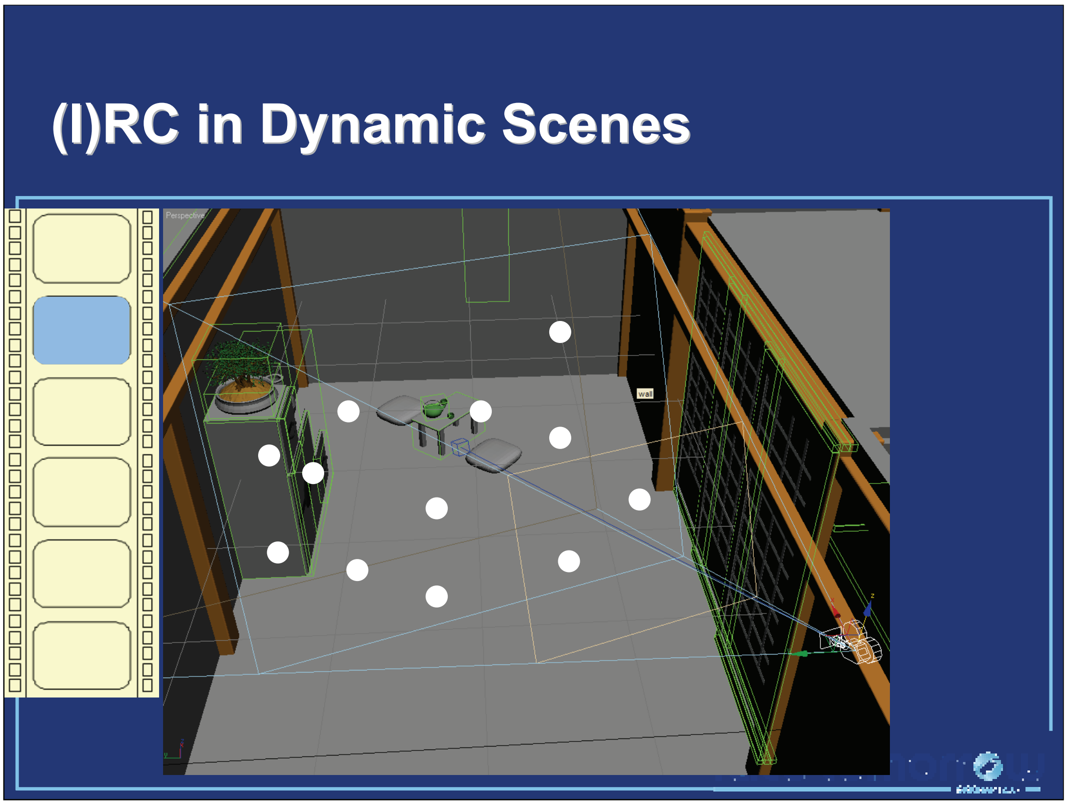


(Ir)Radiance Caching

Record Location

GI Solution

In a single image, this method provides high quality results even when using a very sparse sampling. However, since the estimated lighting is generally obtained through extrapolation from the location of the records, the result depends on the **distribution** of the records.

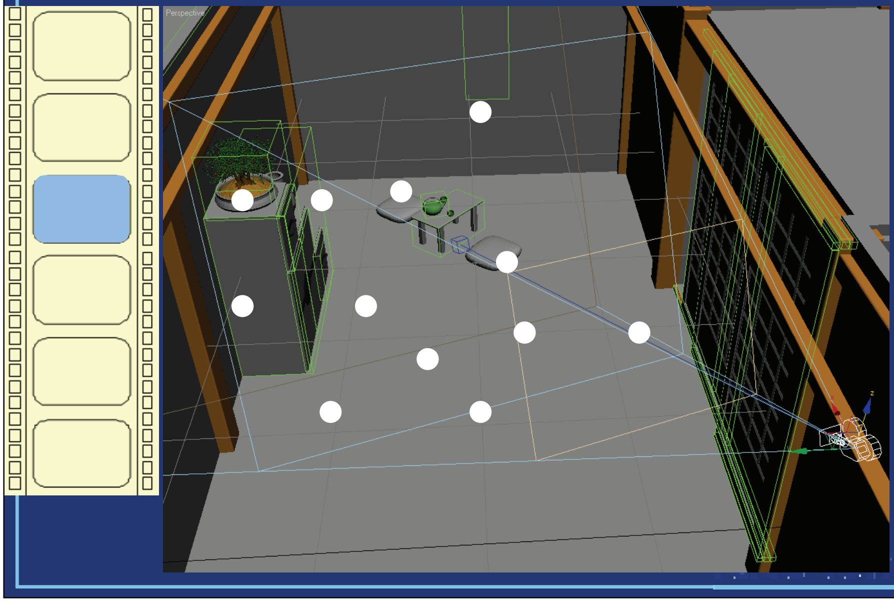


(I)RC in Dynamic Scenes

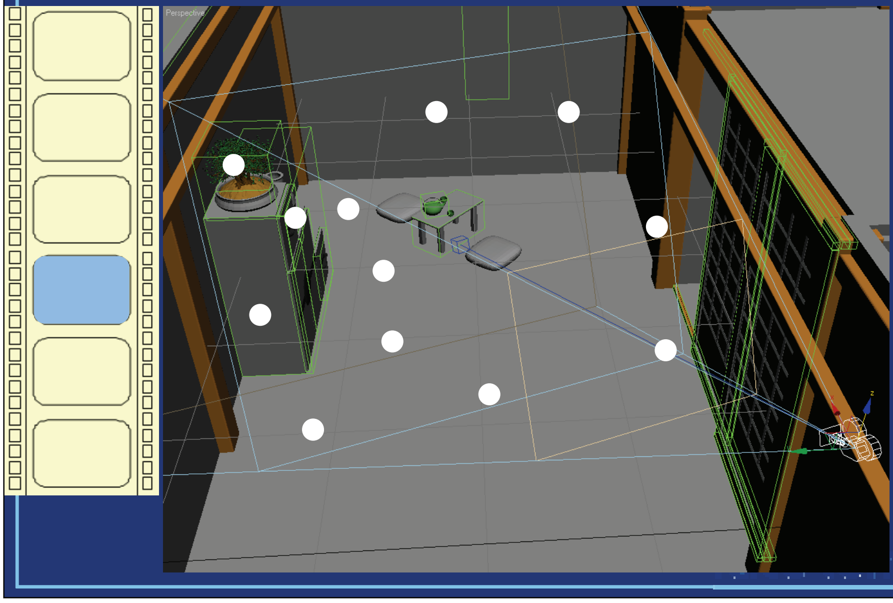
In dynamic scenes, the indirect lighting must be changes in each frame. A simple method for animation rendering using irradiance caching is the entire recomputation of the global illumination solution for each frame. The camera and objects being dynamic, the distribution of records is likely to change across frames, yielding flickering artifacts. A video illustrating this method can be found on [MyWebSite].

Furthermore, the indirect lighting tends to change slowly across frames. The indirect lighting computed at a given frame may thus be reused in several subsequent frames without degrading the quality. However, the lighting may change very quickly in some areas of the scene, and be nearly-static elsewhere. The method described in this course leverages these observations by assigning a distinct lifespan to each record in the cache.

(I)RC in Dynamic Scenes



(I)RC in Dynamic Scenes

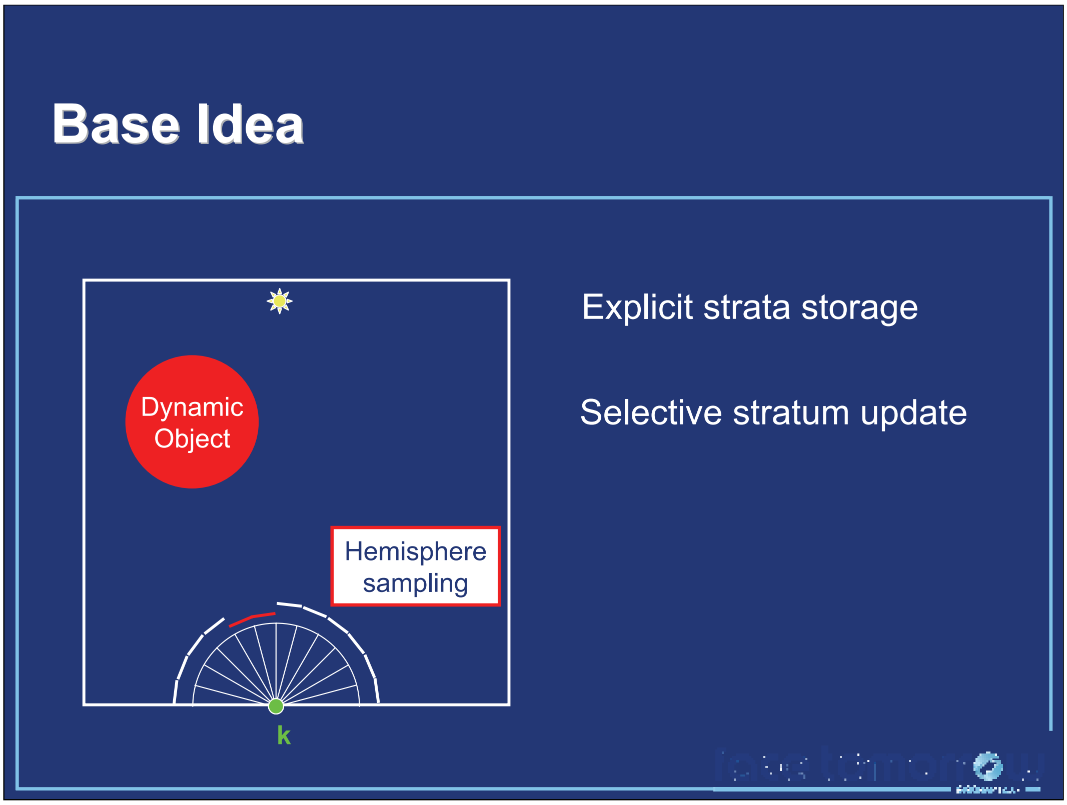


SIGGRAPH2007

Temporally Coherent Irradiance Caching for High Quality Animation Rendering

Miloslaw Smyk, Shin-ichi Kinuwaki, Roman Durikovic, Karol Myszkowski

In Proceeding of Eurographics 2005

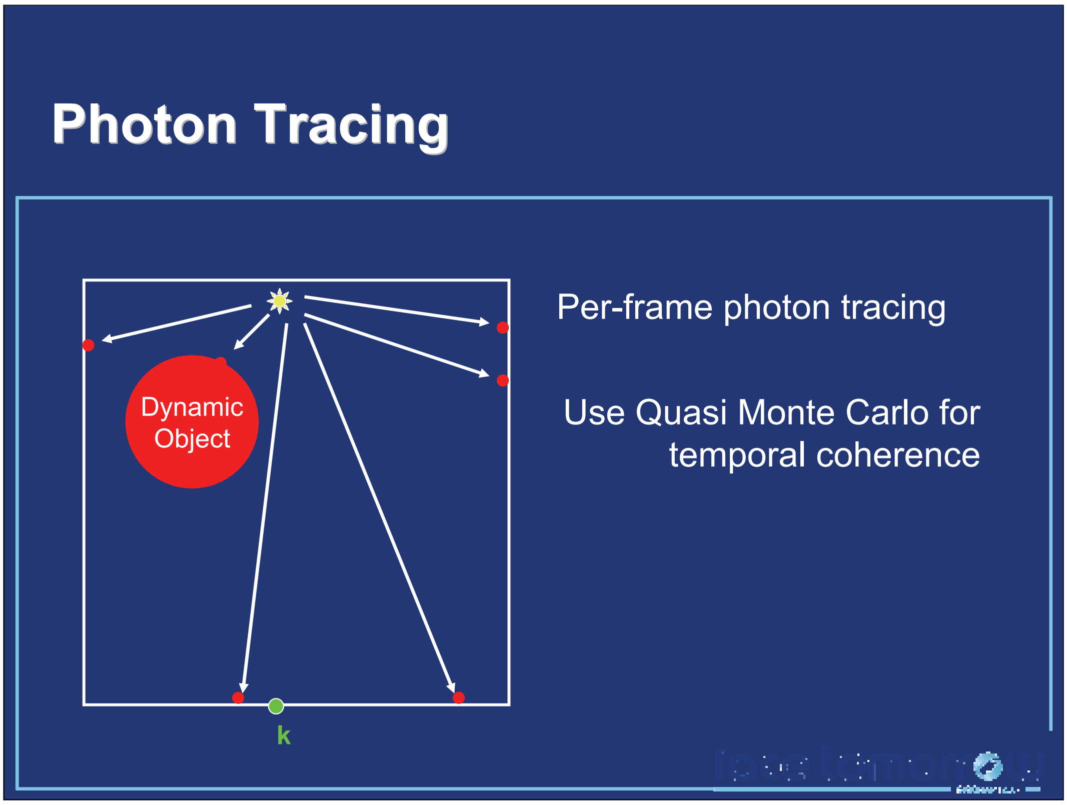


Base Idea

Explicit strata storage

Selective stratum update

The base idea of this paper is the explicit storage and update of all the incoming radiance samples used to estimate the irradiance value of a record. In a dynamic scene, only the stratum corresponding to dynamic objects have to be updated in the course of time, hence reducing the computational cost compared to the classical approach, in which all records are entirely recomputed for each frame of an animation.

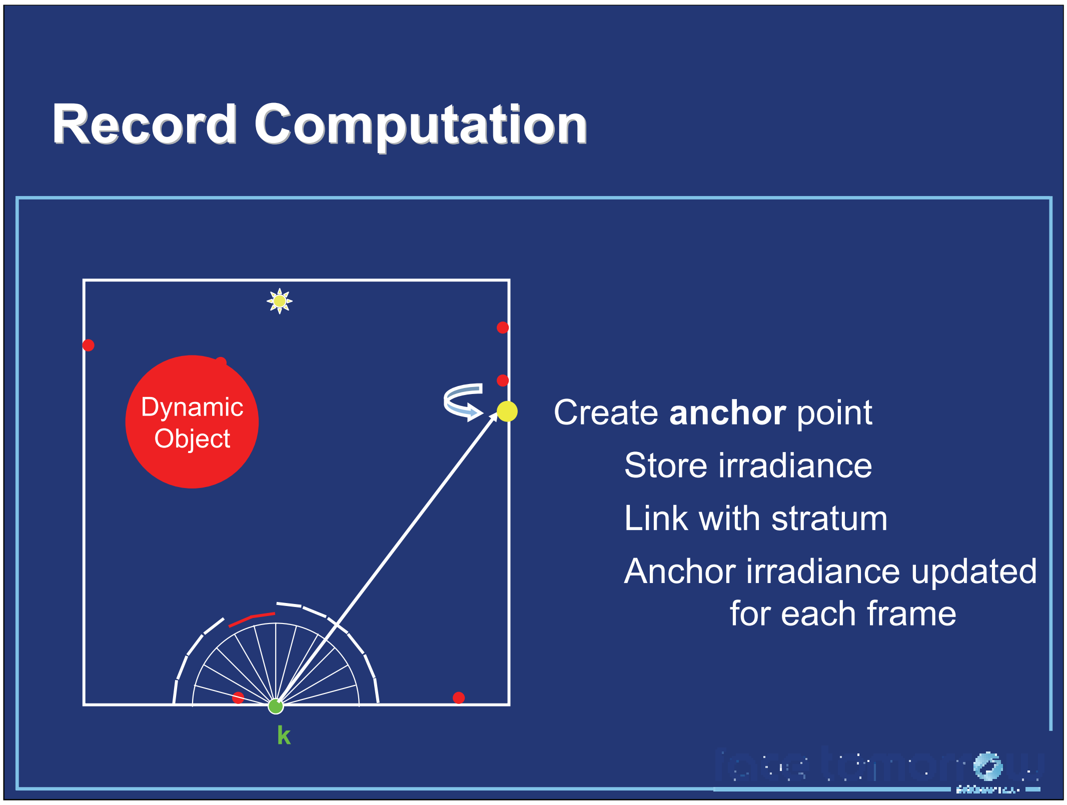


Photon Tracing

Per-frame photon tracing

Use Quasi Monte Carlo for temporal coherence

For each frame, a photon map is computed by tracing random photons using Quasi Monte Carlo method. Using this method, the paths of the photons are likely to be similar across frames, hence increasing the temporal coherence of the photon map.



Record Computation

Create **anchor** point

Store irradiance

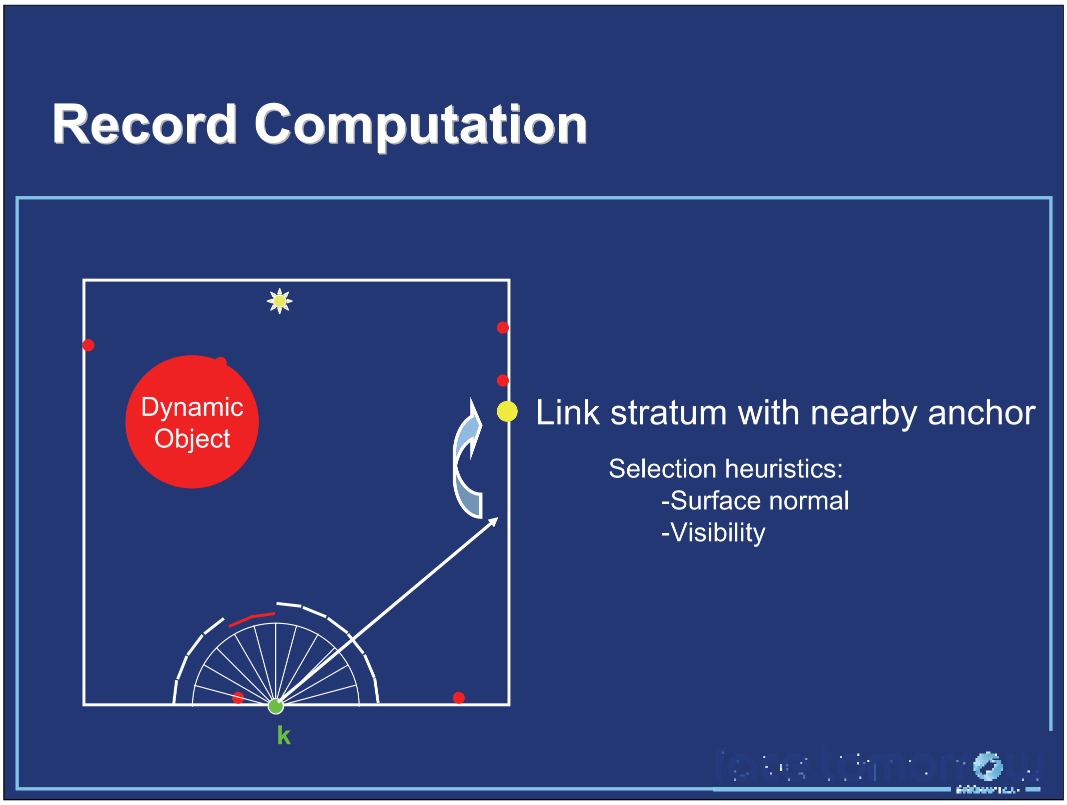
Link with stratum

Anchor irradiance updated for each frame

When creating an irradiance record, a ray is traced for each stratum of the hemisphere. At the intersection point of this ray, an **anchor** point is created. The anchor structure stores the location, the normal, and the irradiance at the intersection point. This anchor irradiance is simply estimated by density estimation on the photon map.

Then, the stratum of the record is explicitly linked to the anchor point.

Note that the location of anchor points is persistent over time. However, their irradiance value is recomputed at each frame using the photon map.



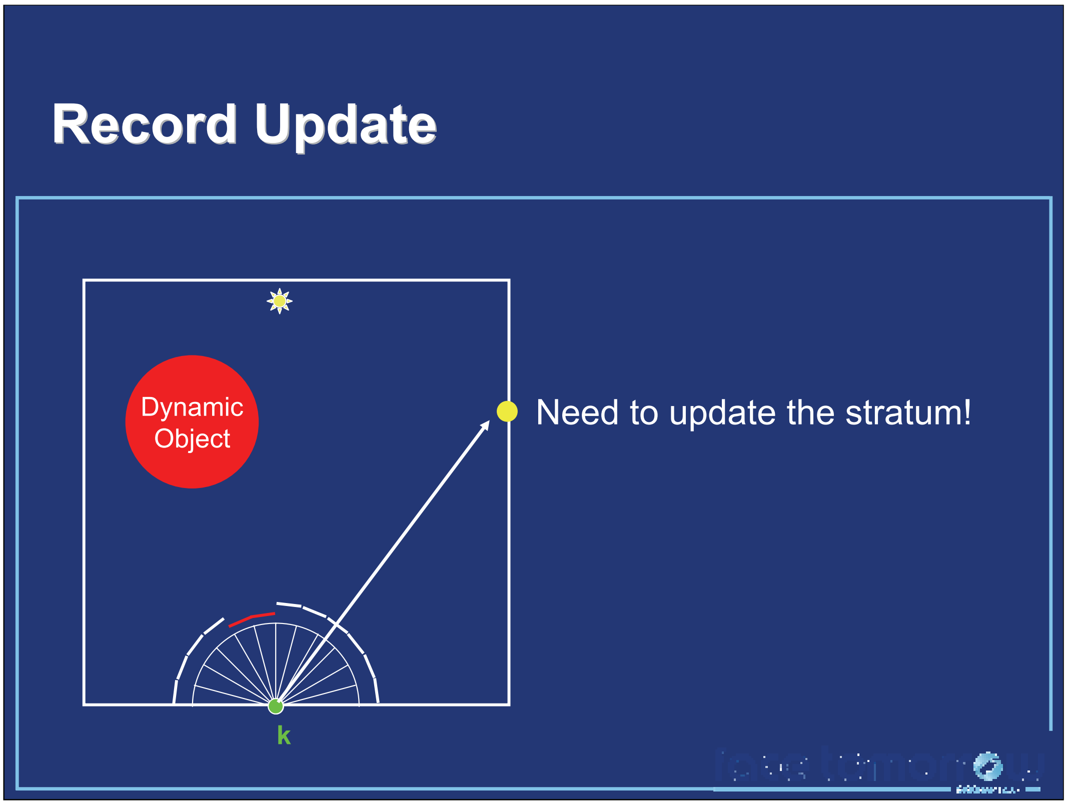
Record Computation

Link stratum with nearby anchor

Selection heuristics:

- -Surface normal
- -Visibility

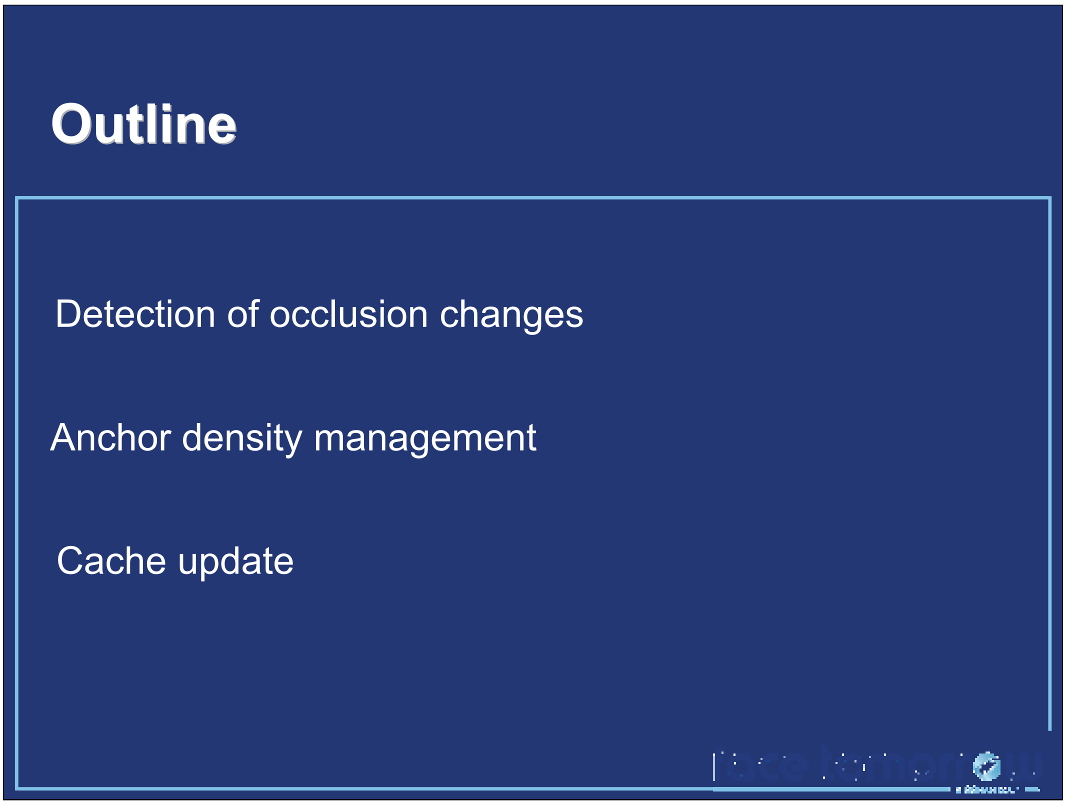
If the intersection point corresponding to the stratum is close to an existing anchor points, the stratum can be directly linked to the anchor point. Several heuristics have been proposed to estimate the suitability of the link to an existing anchor. Among them, the surface at the intersection point and at the anchor point must have similar normals. Also, the visibility between the record location and the anchor point is explicitly tested by tracing a shadow ray.



Record Update

Need to update the stratum!

The most important problem to solve in this method is the detection of occlusion changes: when the red sphere crosses the ray between the record and the anchor point, the radiance value of the stratum must be updated accordingly.



Outline

Detection of occlusion changes

Anchor density management

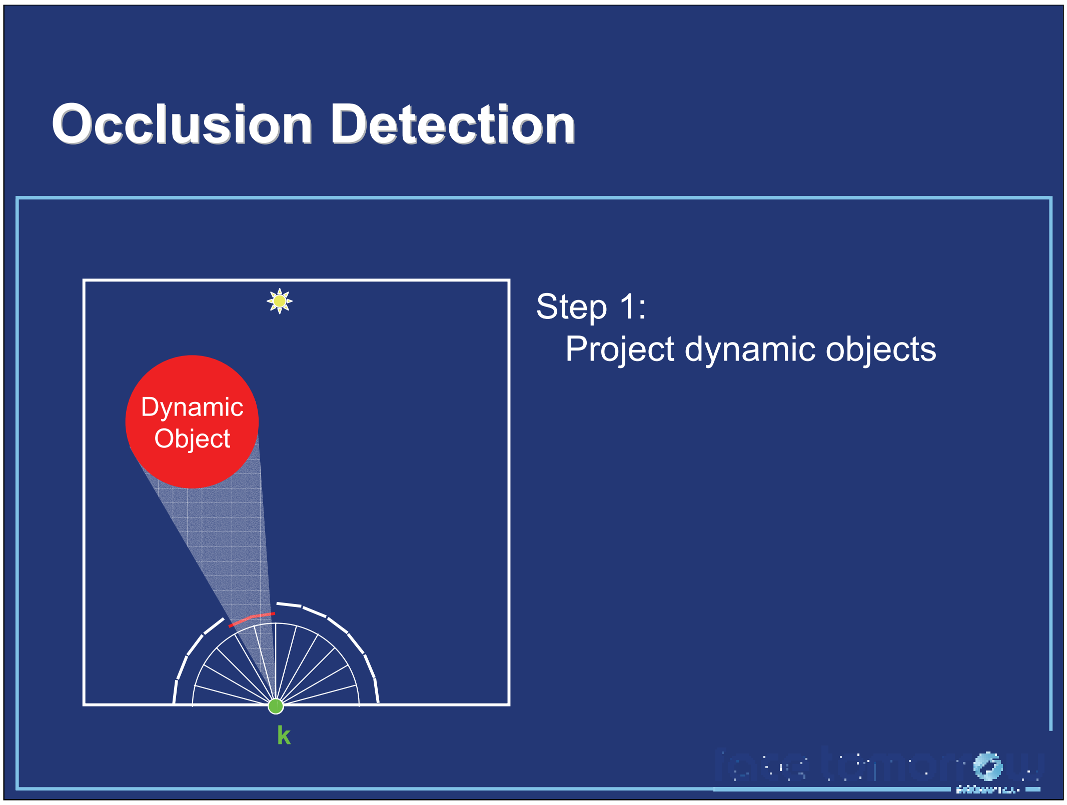
Cache update

Outline

Detection of occlusion changes

Anchor density management

Cache update

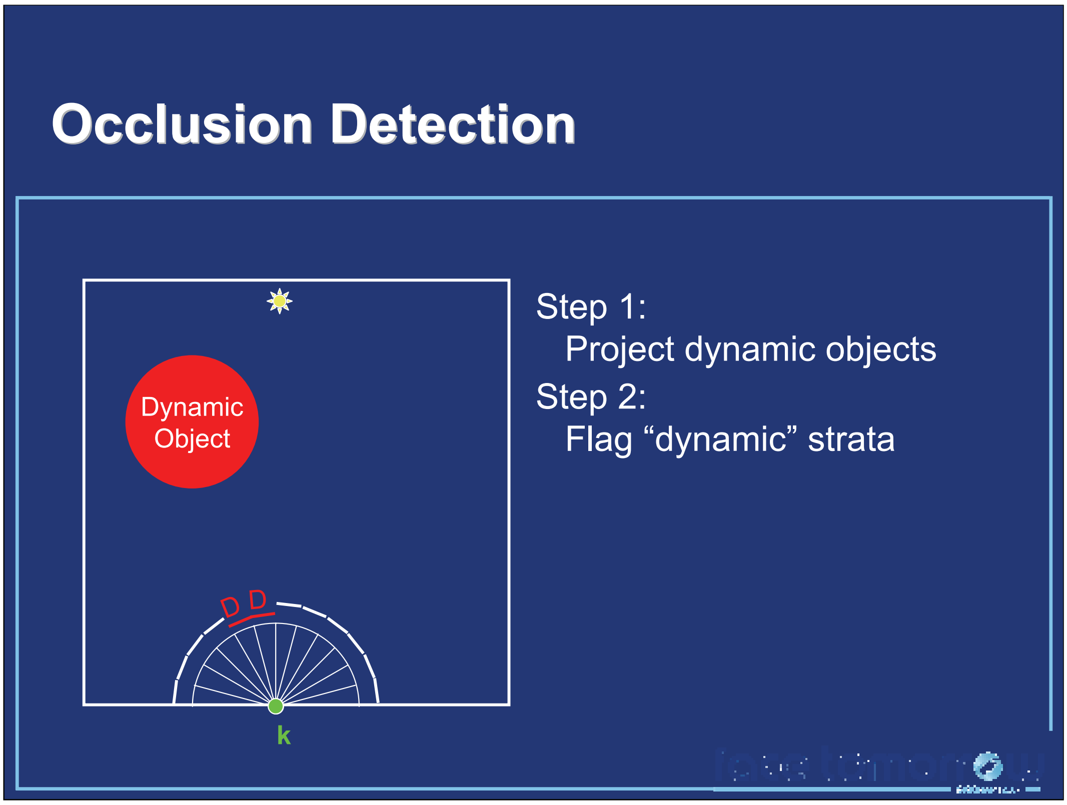


Occlusion Detection

Step 1:
Project dynamic objects

The occlusion detection is performed in 4 steps, and may be performed using graphics hardware.

In the first step, the dynamic objects **only** are projected onto the hemisphere above the record.

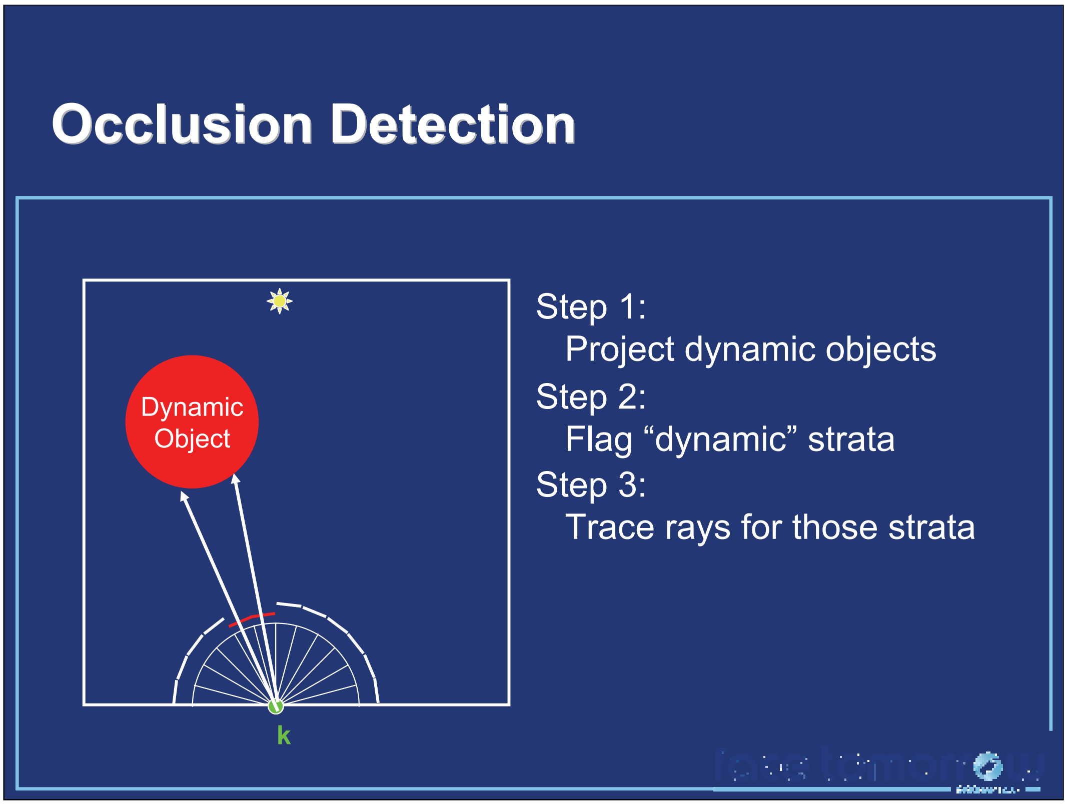


Occlusion Detection

Step 1:
Project dynamic objects

Step 2:
Flag “dynamic” strata

This projection of the dynamic objects allows us to flag the corresponding strata as « dynamic strata », e.g. strata for which the incoming radiance is due to a dynamic object.



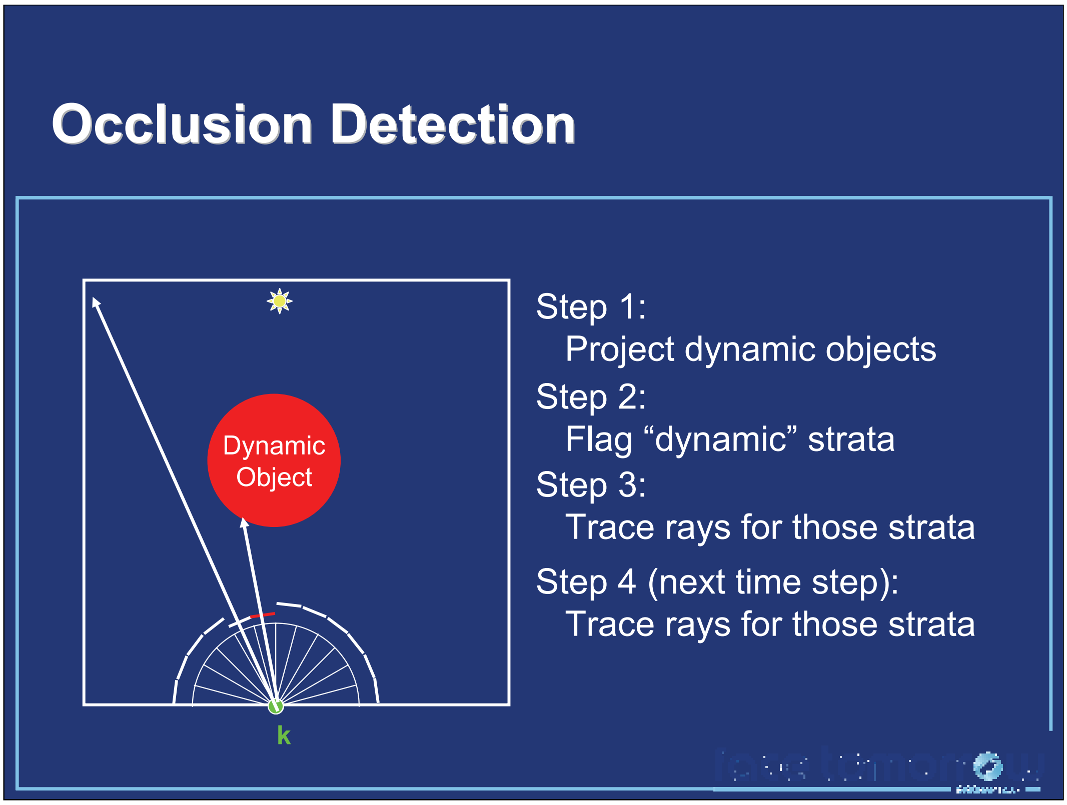
Occlusion Detection

Step 1:
Project dynamic objects

Step 2:
Flag “dynamic” strata

Step 3:
Trace rays for those strata

For those strata, rays are traced to estimate the incoming radiance. Note that anchor points are not created on dynamic objects. Instead, the irradiance at each intersection point is computed by density estimation in the photon map. Combined with the existing information contained in « static » strata, the irradiance of the record can be easily deduced.



Occlusion Detection

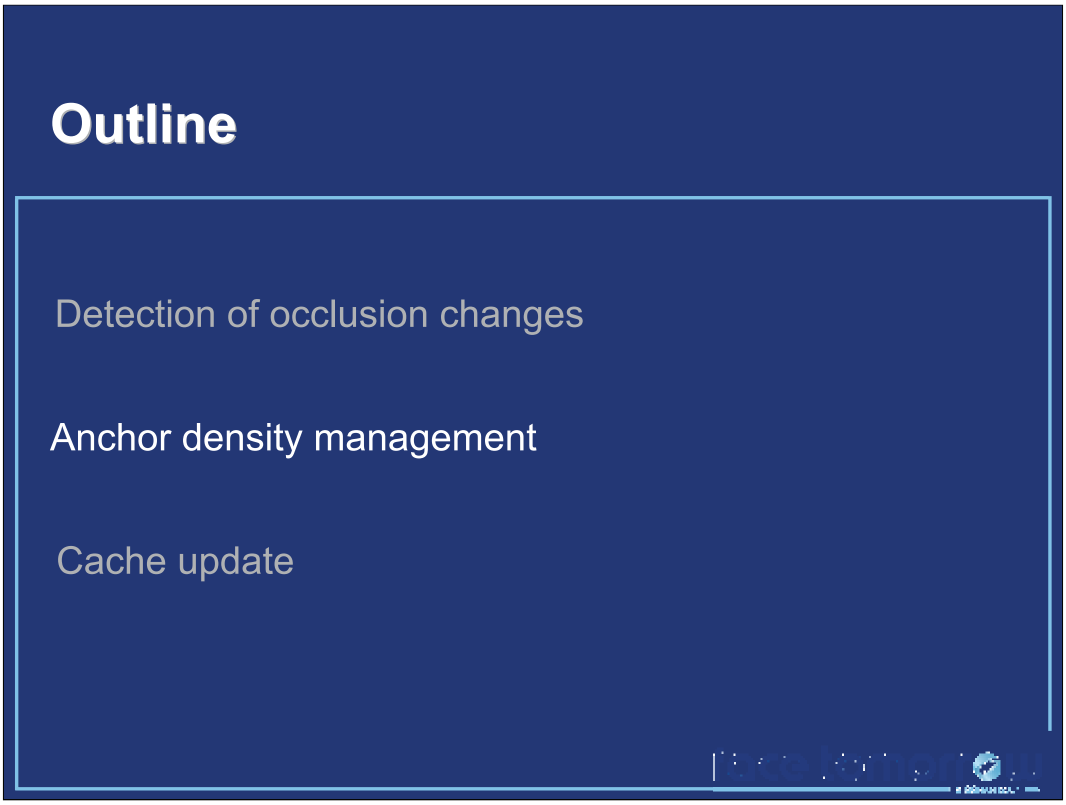
Step 1:
Project dynamic objects

Step 2:
Flag “dynamic” strata

Step 3:
Trace rays for those strata

Step 4 (next time step):
Trace rays for those strata

At the next time step, the « dynamic strata » will also be explicitly sampled by ray tracing to update their values.



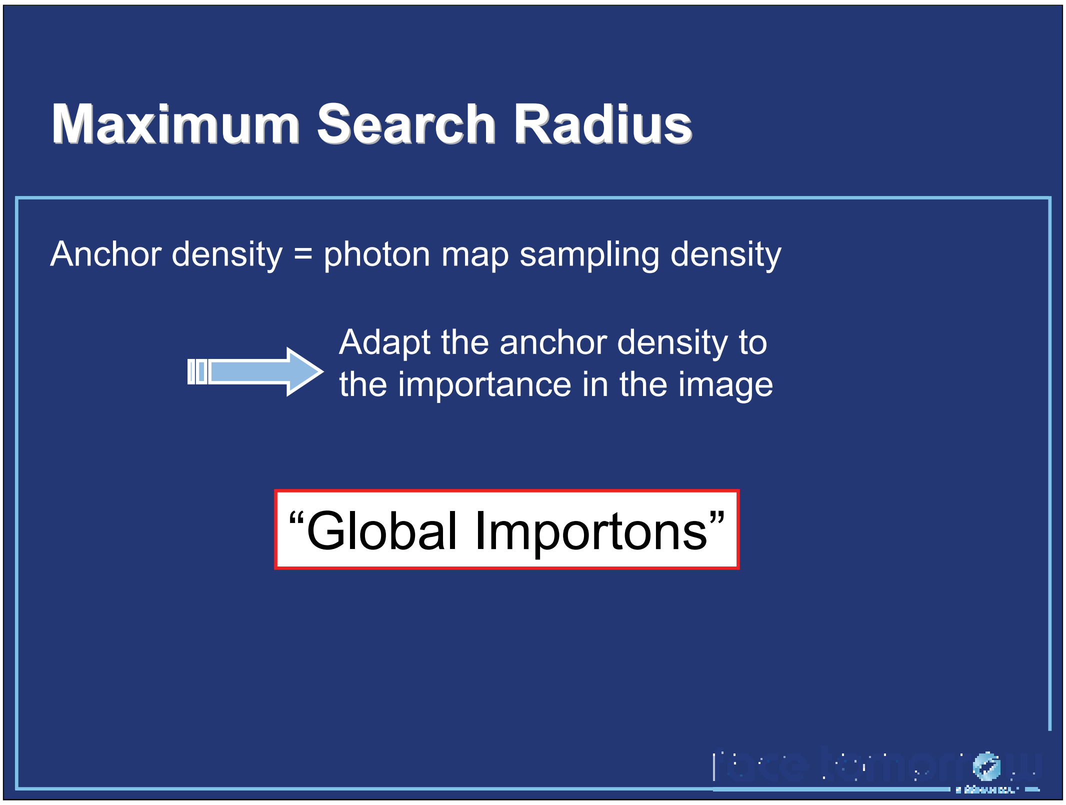
Outline

Detection of occlusion changes

Anchor density management

Cache update

The anchor data structure allows us to reduce the number of density estimations to render an animation. However, the efficiency of the method lies in an adaptive distribution of anchors.



Maximum Search Radius

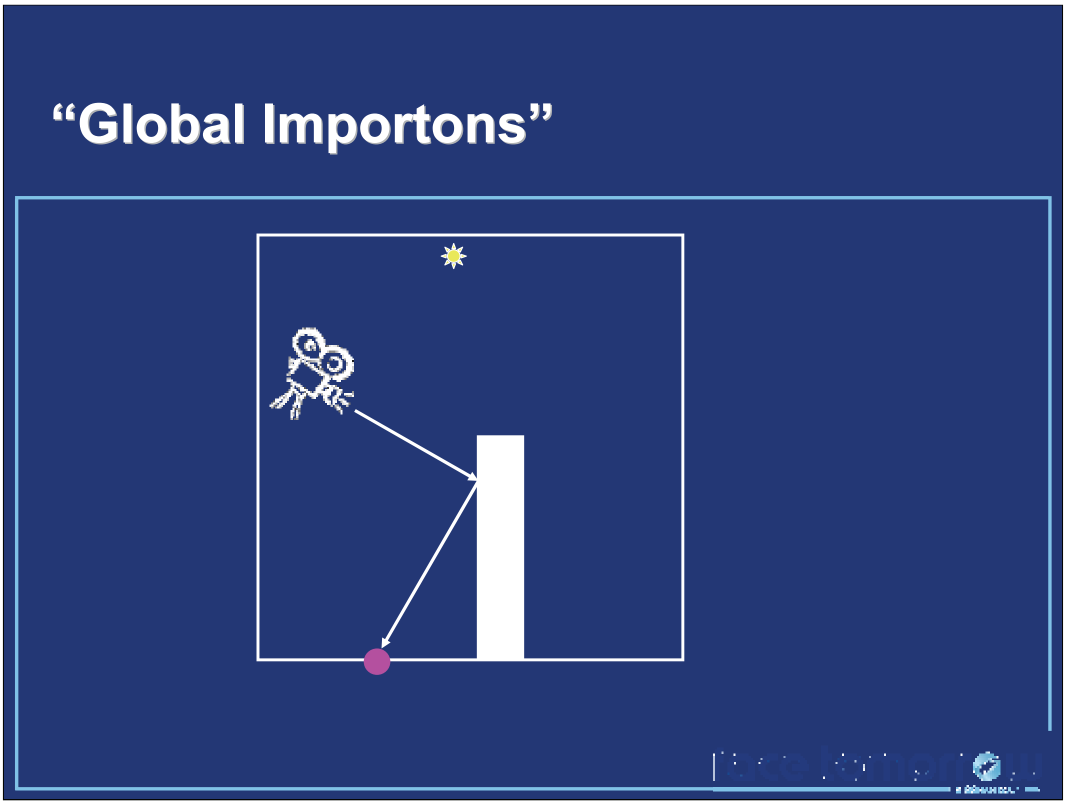
Anchor density = photon map sampling density

⇒ Adapt the anchor density to the importance in the image

“Global Importons”

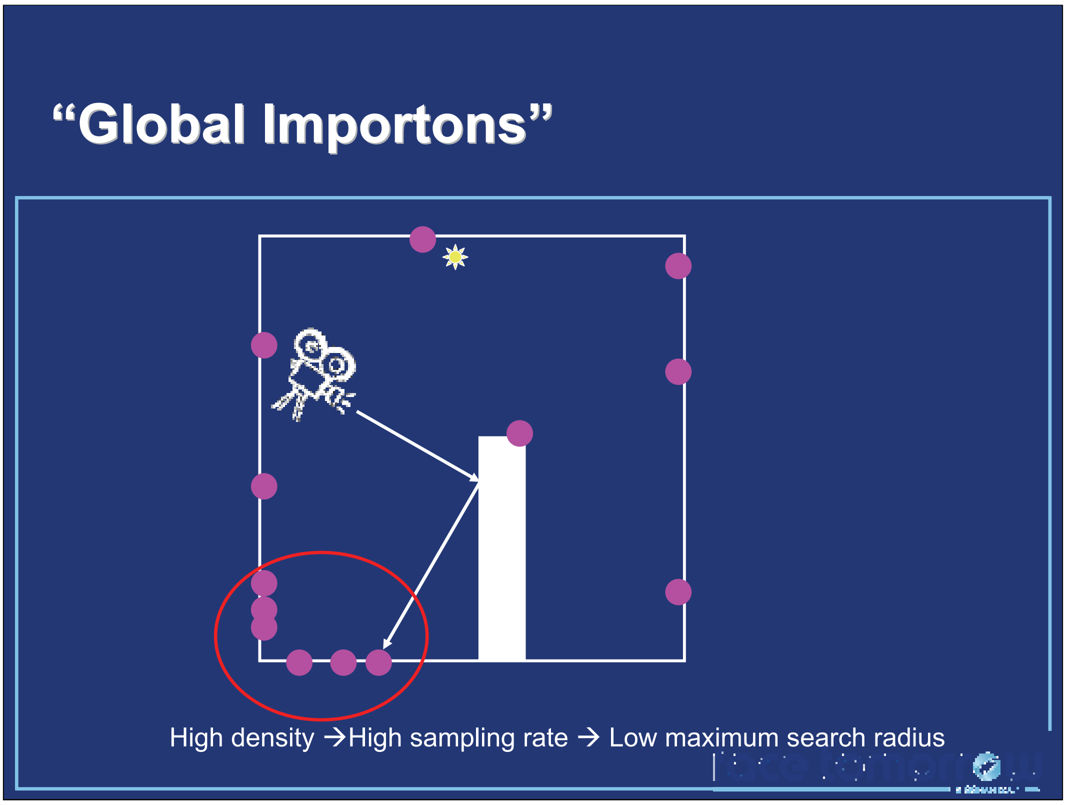
The density of anchor points directly impacts the quality of the lighting estimate. In particular, the distribution of anchor points should be very dense at points which highly contribute to the lighting of visible points. This density is controlled by the maximum search radius used when looking for an anchor nearby an intersection point. If this radius is small, the queries in the anchor structure may not find any suitable nearby anchor. Therefore, a small radius increases the anchor density. Converserly, a large radius decreases the anchor density.

The authors use “global importons” to adapt the density of anchors.



“Global Importons”

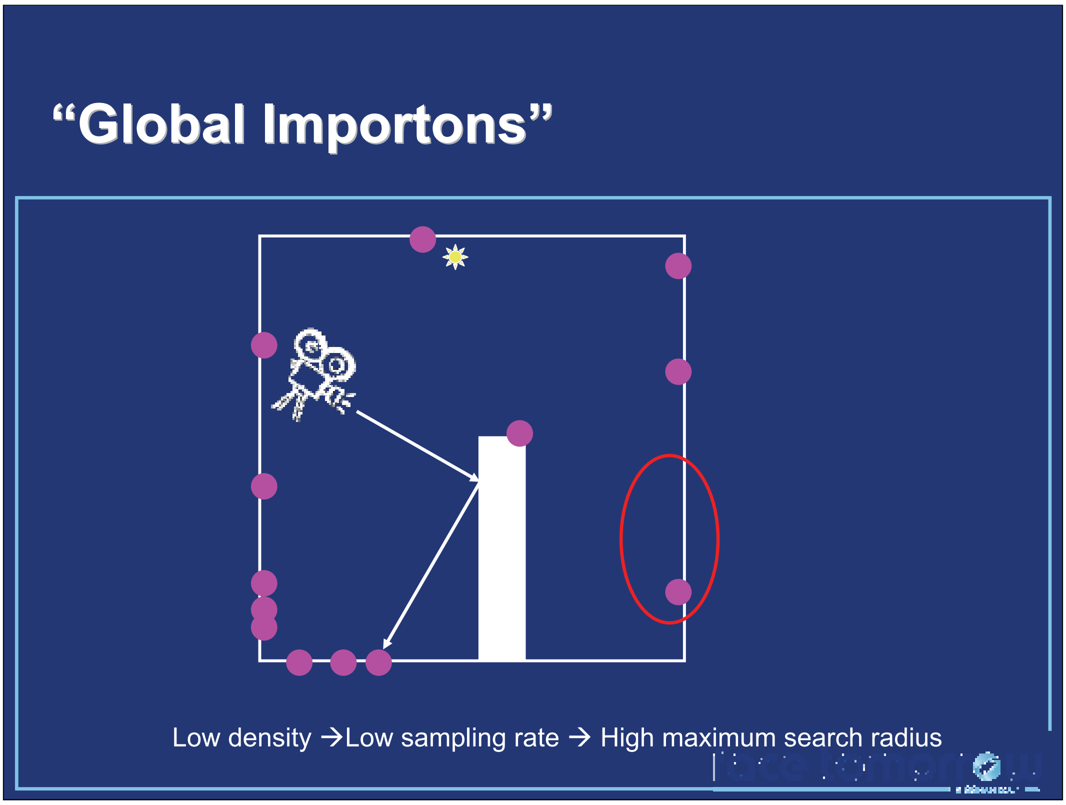
The “global importons” are obtained by tracing rays from the camera, and storing the second bounce of the ray in the scene.



“Global Importons”

High density →High sampling rate → Low maximum search radius

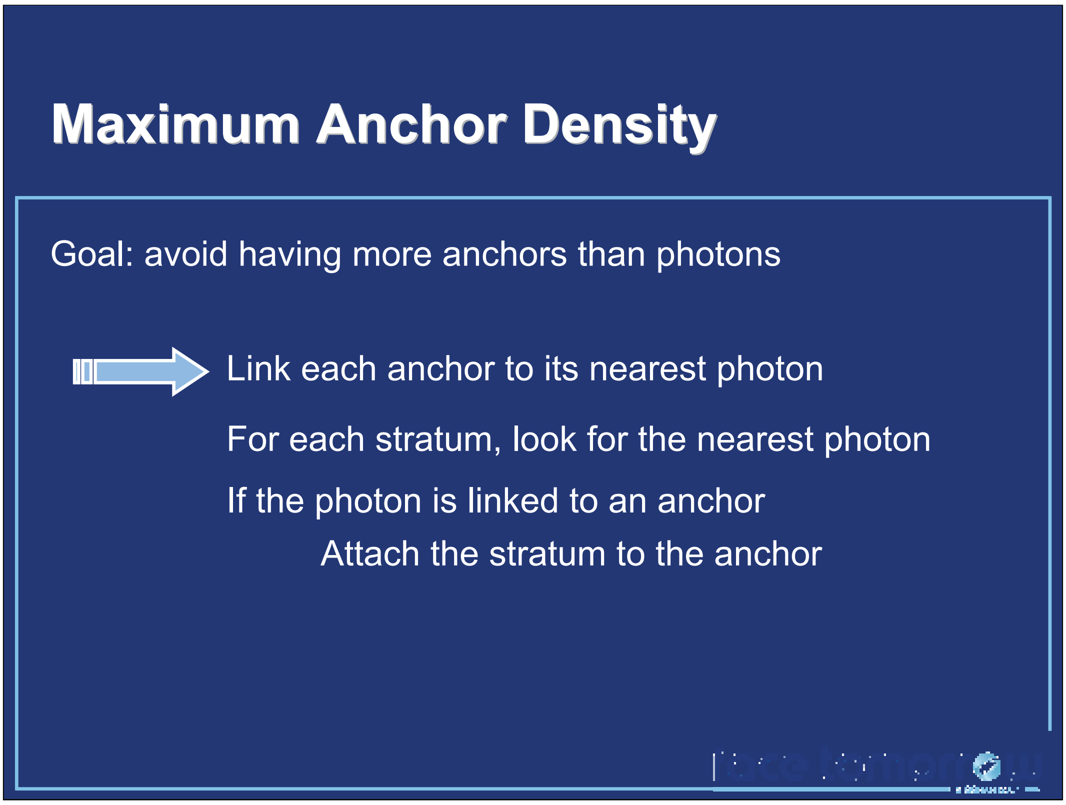
The zones containing many “global importons” are zones which highly contribute to the indirect lighting of visible points. Therefore, the anchor density should be raised to ensure the image quality. Therefore, the maximum search radius is lowered.



“Global Importons”

Low density →Low sampling rate → High maximum search radius

Conversely, zones with very few importons do not contribute much to the indirect lighting of visible points. Therefore, the sampling rate is kept low to increase the rendering speed. The maximum search distance is then set to a high value.



Maximum Anchor Density

Goal: avoid having more anchors than photons

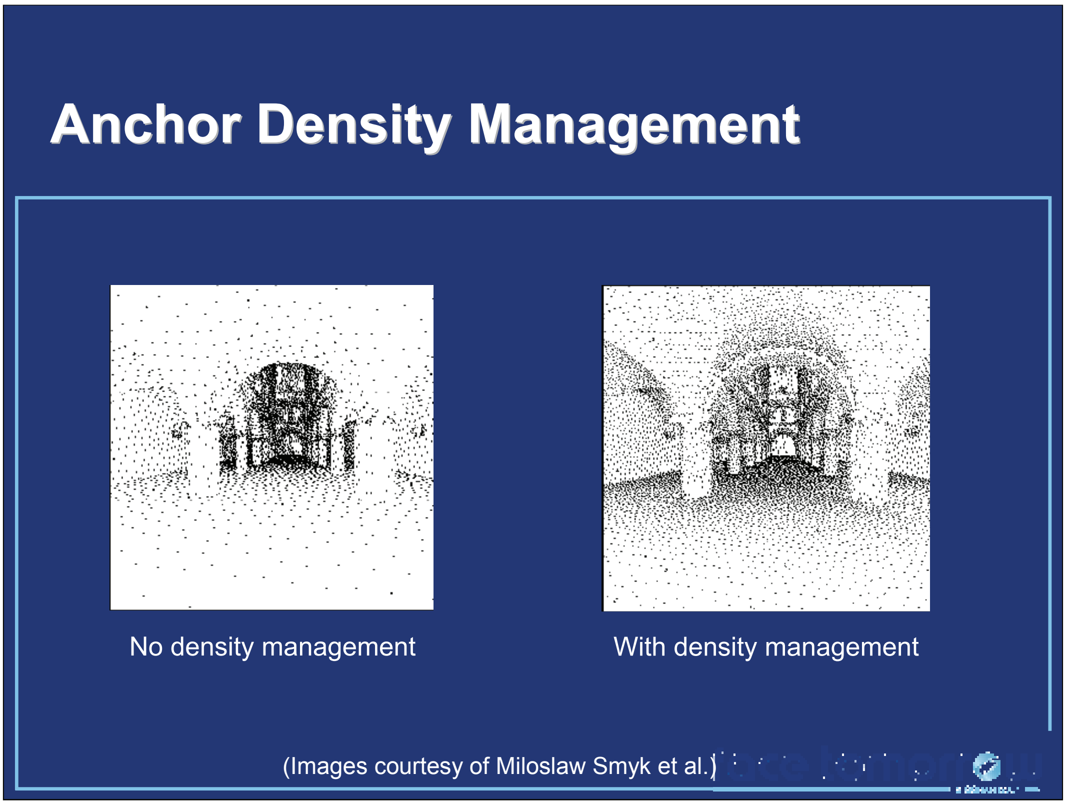
➡ Link each anchor to its nearest photon

For each stratum, look for the nearest photon

If the photon is linked to an anchor

    Attach the stratum to the anchor

One of the goals of this work is the reduction of the photon searches during global illumination computation. Therefore, if the structure contains more anchors than photons, the algorithm would become very inefficient. The method described in this slide ensures that there cannot be more anchors than photons

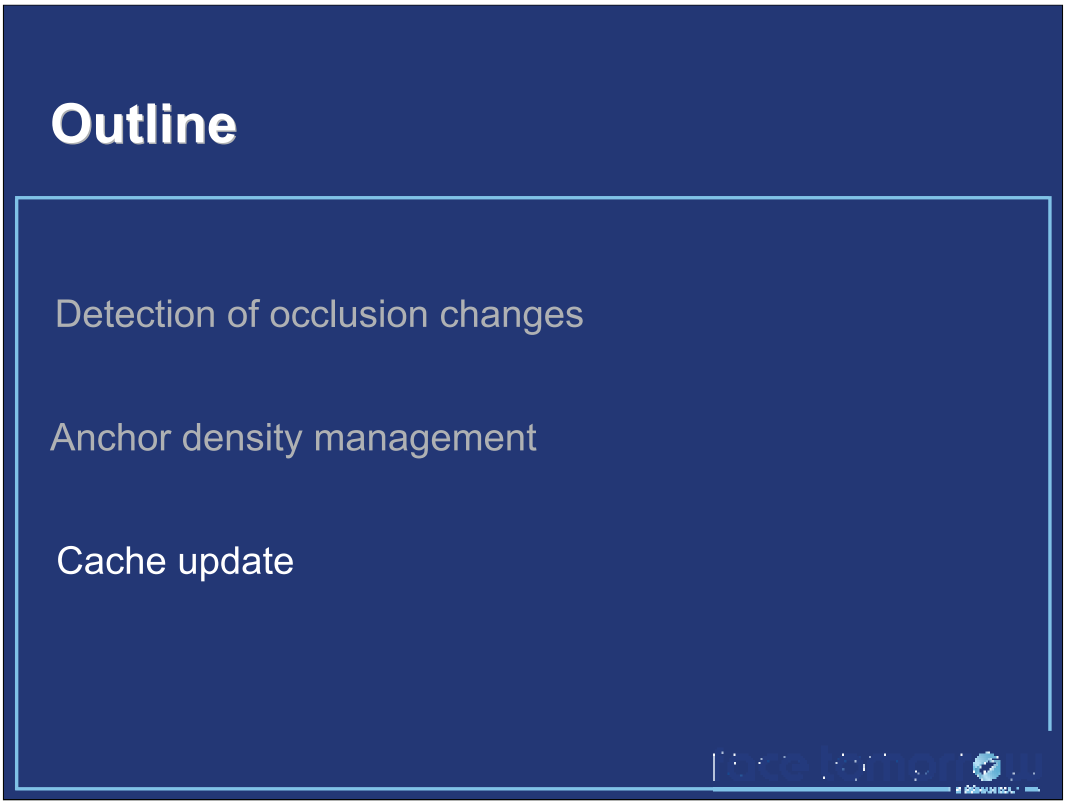


Anchor Density Management

No density management

With density management

(Images courtesy of Miloslaw Smyk et al.)

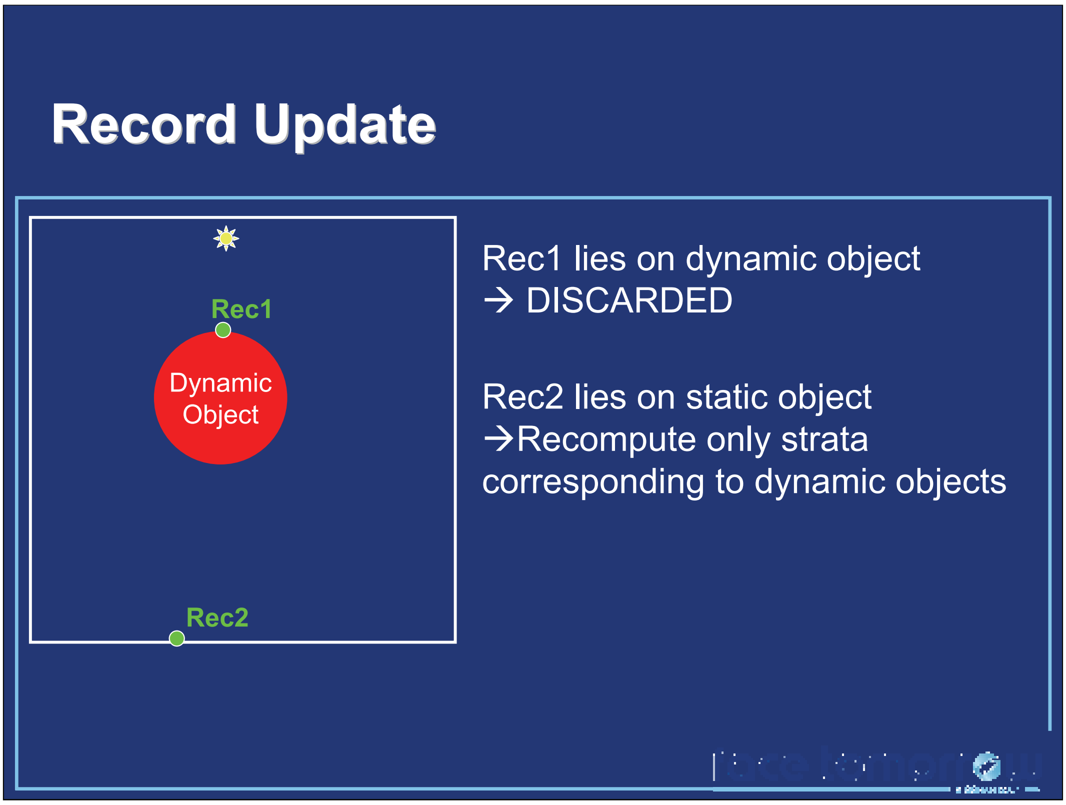


Outline

Detection of occlusion changes

Anchor density management

Cache update

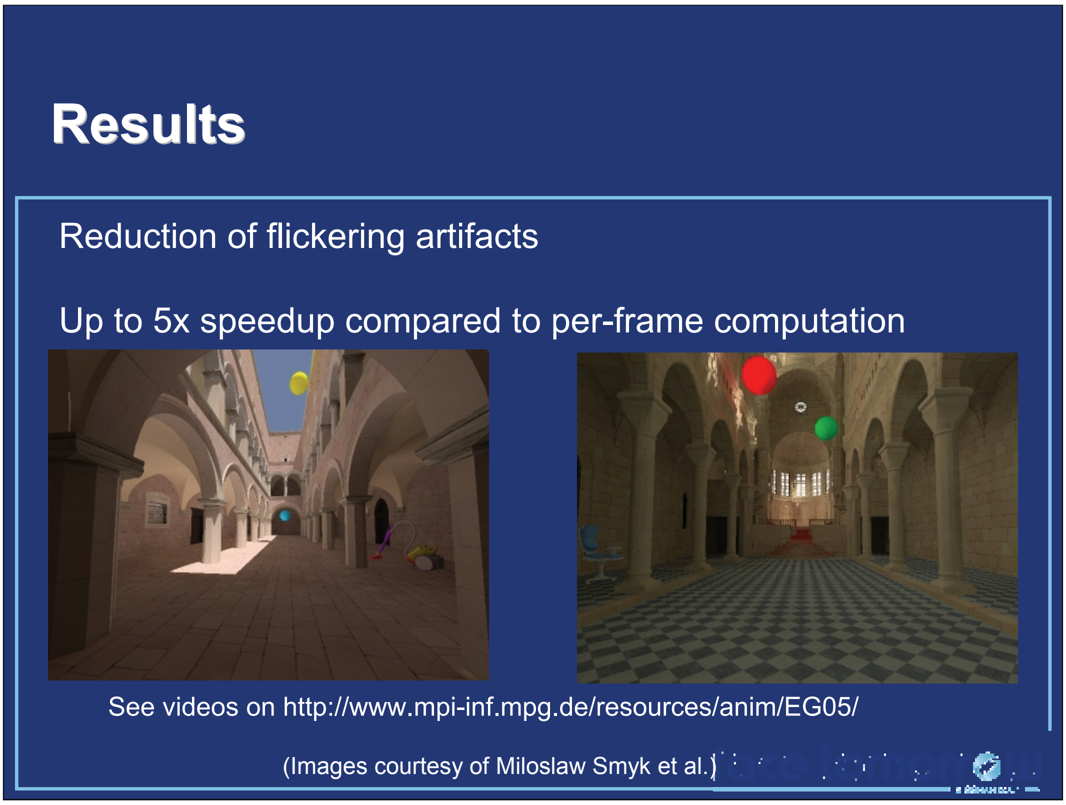


Record Update

Rec1 lies on dynamic object
→ DISCARDED

Rec2 lies on static object
→Recompute only strata corresponding to dynamic objects

The irradiance value of the records must be updated across frames. Several cases can happen: records located on dynamic objects are discarded for each frame. Records on static objects undergo adaptive strata replacement as explained previously.



Results

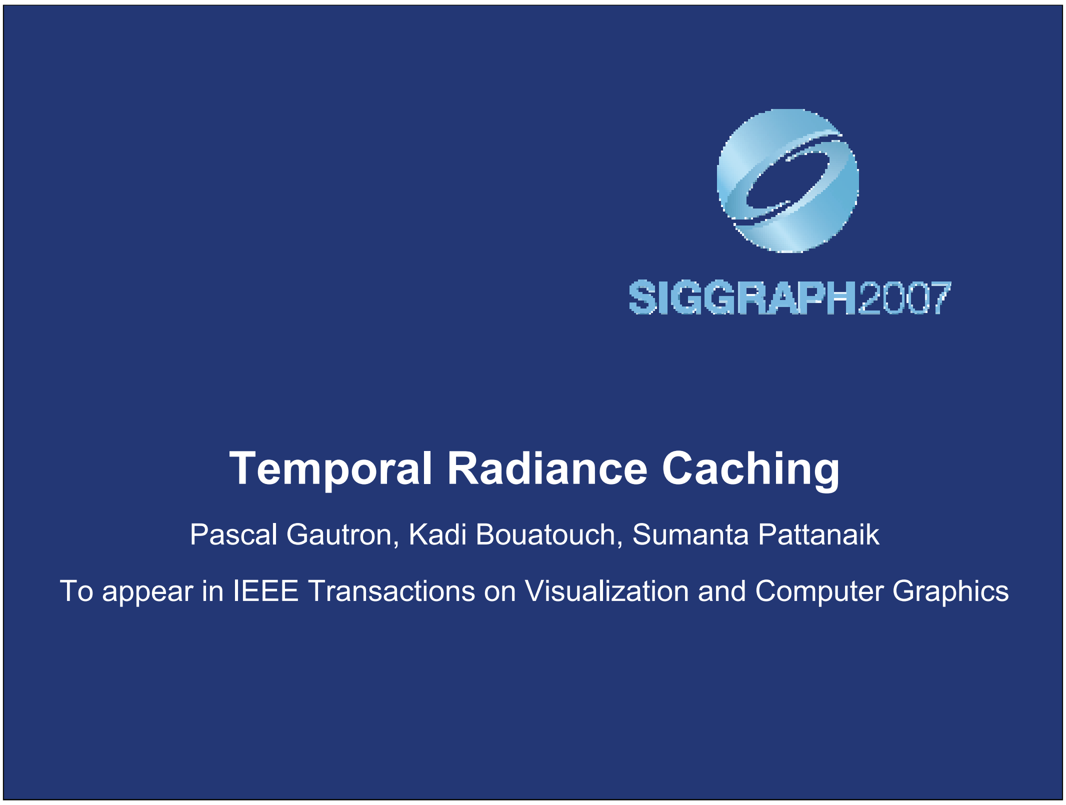
Reduction of flickering artifacts

Up to 5x speedup compared to per-frame computation

See videos on http://www.mpi-inf.mpg.de/resources/anim/EG05/

(Images courtesy of Miloslaw Smyk et al.)

The resulting animations exhibit a significant reduction of flickering, while reducing the computational cost by a factor up to 5.

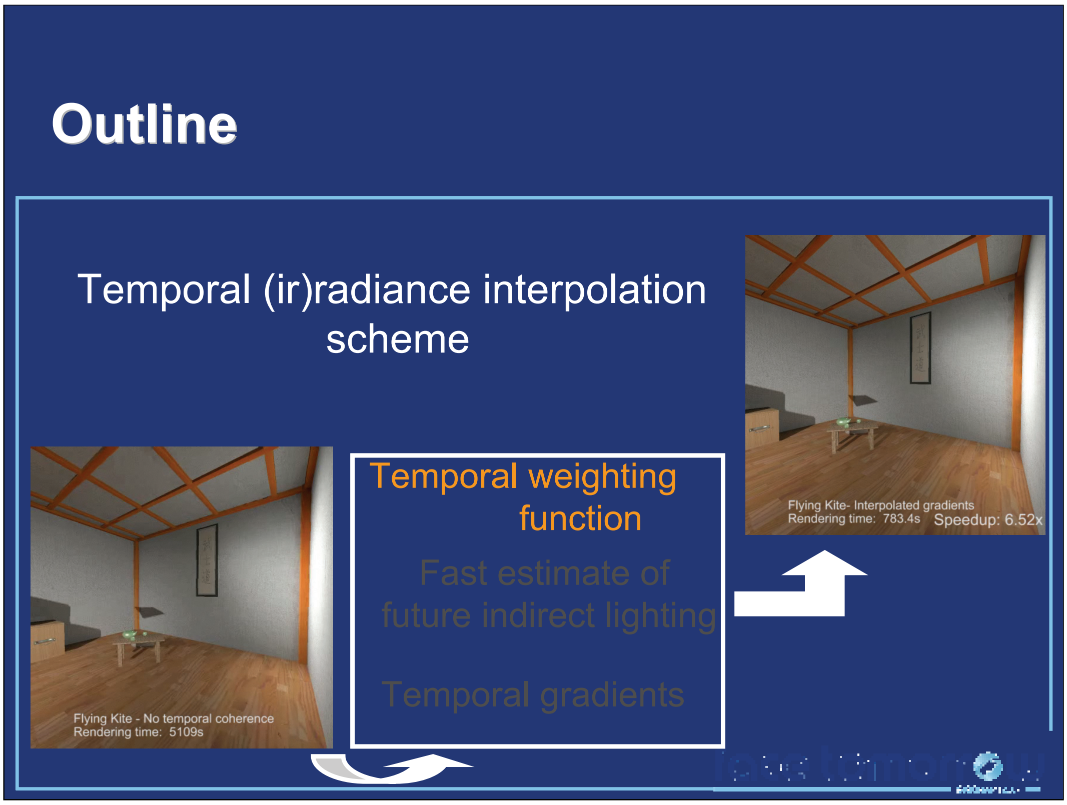


SIGGRAPH2007

Temporal Radiance Caching

Pascal Gautron, Kadi Bouatouch, Sumanta Pattanaik

To appear in IEEE Transactions on Visualization and Computer Graphics



Outline

Temporal (ir)radiance interpolation scheme

Temporal weighting function

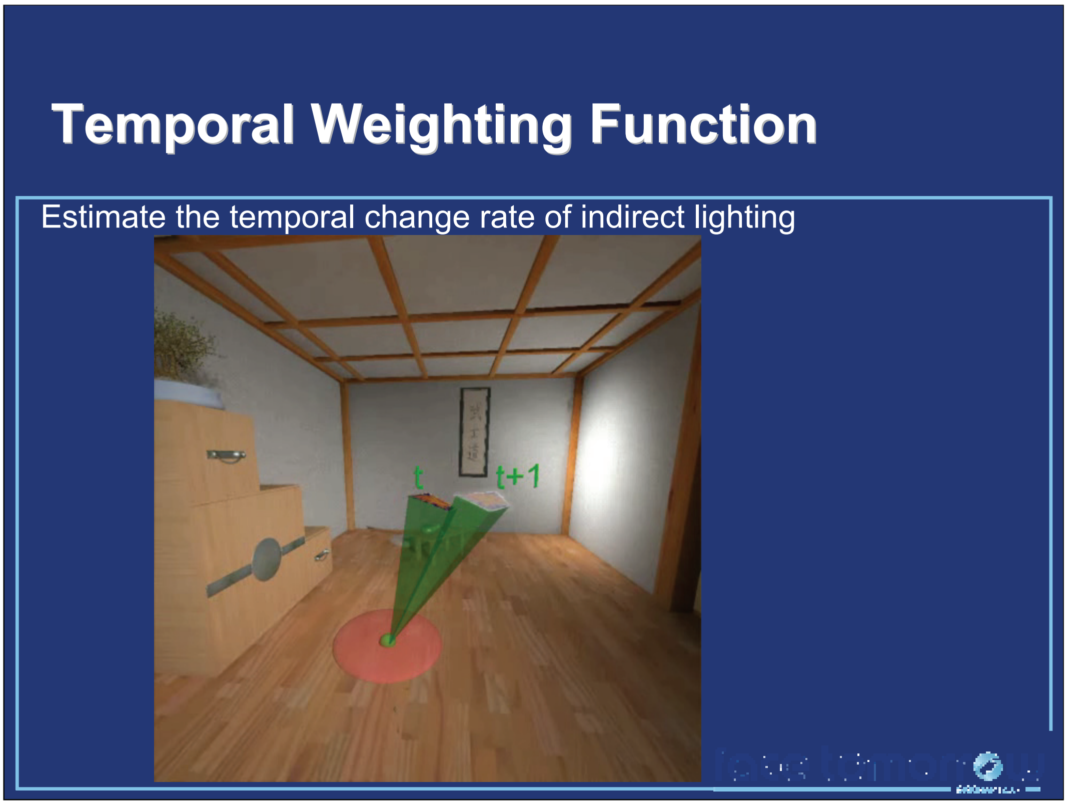
Fast estimate of future indirect lighting

Temporal gradients

Flying Kite - No temporal coherence
Rendering time: 5109s

Flying Kite- Interpolated gradients
Rendering time: 783.4s Speedup: 6.52x

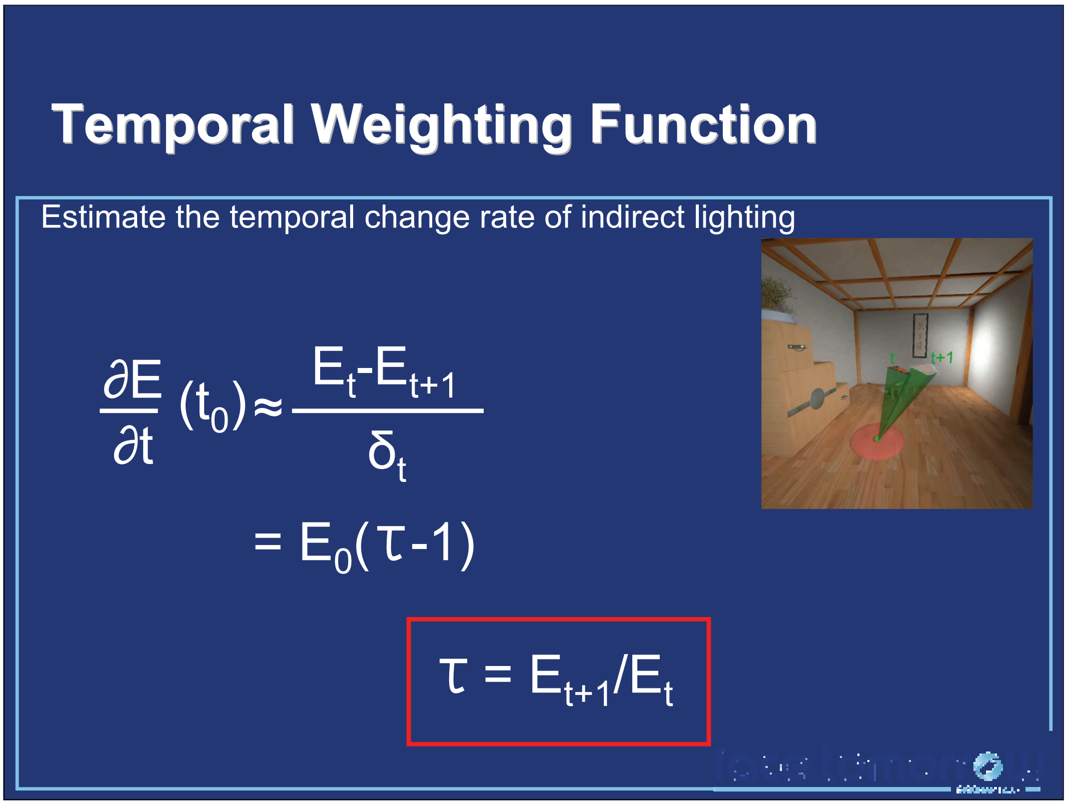
More precisely, the method described in this course is an extension of the irradiance caching interpolation scheme to the temporal domain. To this end, we devise a temporal weighting function to determine the lifespan of a record, e.g. the number of frames in which a record can be reused without degrading the rendering quality. The contribution of a record within its lifespan is estimated using temporal irradiance gradients.



Temporal Weighting Function

Estimate the temporal change rate of indirect lighting

The spatial weighting function described in [Ward88] is based on an estimate of the change of indirect lighting with respect to displacement and rotation. The temporal weighting function is thus based on an estimate of the **temporal change** of indirect lighting across frames.



Temporal Weighting Function

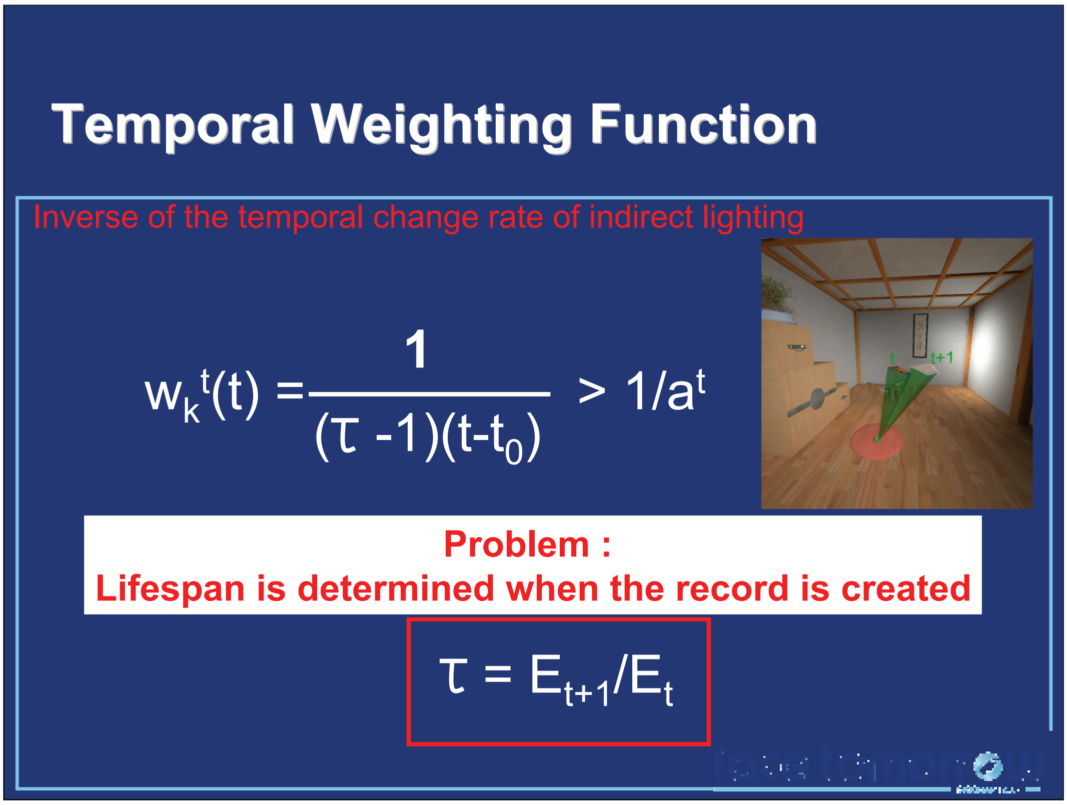
Estimate the temporal change rate of indirect lighting

$$\frac{\partial E}{\partial t}(t_0) \approx \frac{E_t - E_{t+1}}{\delta_t}$$

$$= E_0(\tau - 1)$$

$$\boxed{\tau = E_{t+1}/E_t}$$

Let us consider the irradiance E_t at current frame, and the irradiance E_{t+1} at next frame. The temporal change of indirect lighting can be estimated by a numerical estimation of the temporal derivative of the lighting. We define the value τ as the ratio of the future and current lighting.



Temporal Weighting Function

Inverse of the temporal change rate of indirect lighting

$$w_k^t(t) = \frac{1}{(\tau - 1)(t - t_0)} > 1/a^t$$

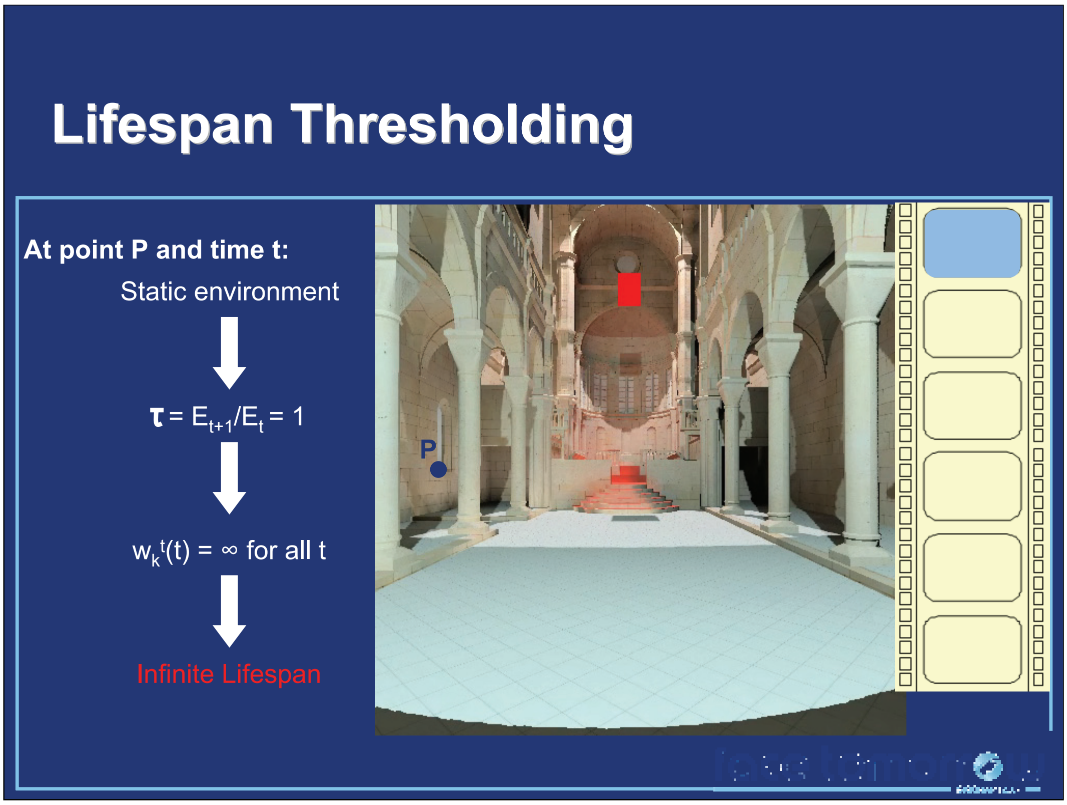
Problem :
Lifespan is determined when the record is created

$$\tau = E_{t+1}/E_t$$

Using a derivation similar to [Ward88], we obtain a temporal weighting function defined as the inverse of the change of indirect lighting over time. The lifespan of a record is thus adapted to the **local change** of indirect lighting: fast changes yield a short lifespan, while records can be reused across many frames if the temporal changes are slow. The user-defined threshold value a^t conditions the temporal accuracy of the computation: a high value leads to long lifespans, hence reducing the rendering time at the detriment of the quality. Conversely, a small value ensures frequent updates at the expense of rendering time.

At the end of its lifespan, a record is discarded and replaced by a new record located at the same point. This method strenghtens the temporal coherence of the distribution of records, hence avoiding flickering artifacts due to changes of record distribution.

However, the lifespan is determined using the indirect lighting at current and next frames only. Therefore, later changes do not affect this lifespan, hence harming the reactivity of the algorithm. An overestimation of the lifespan yields residual global illumination effects known as « ghosting artifacts ».



Lifespan Thresholding

At point P and time t:

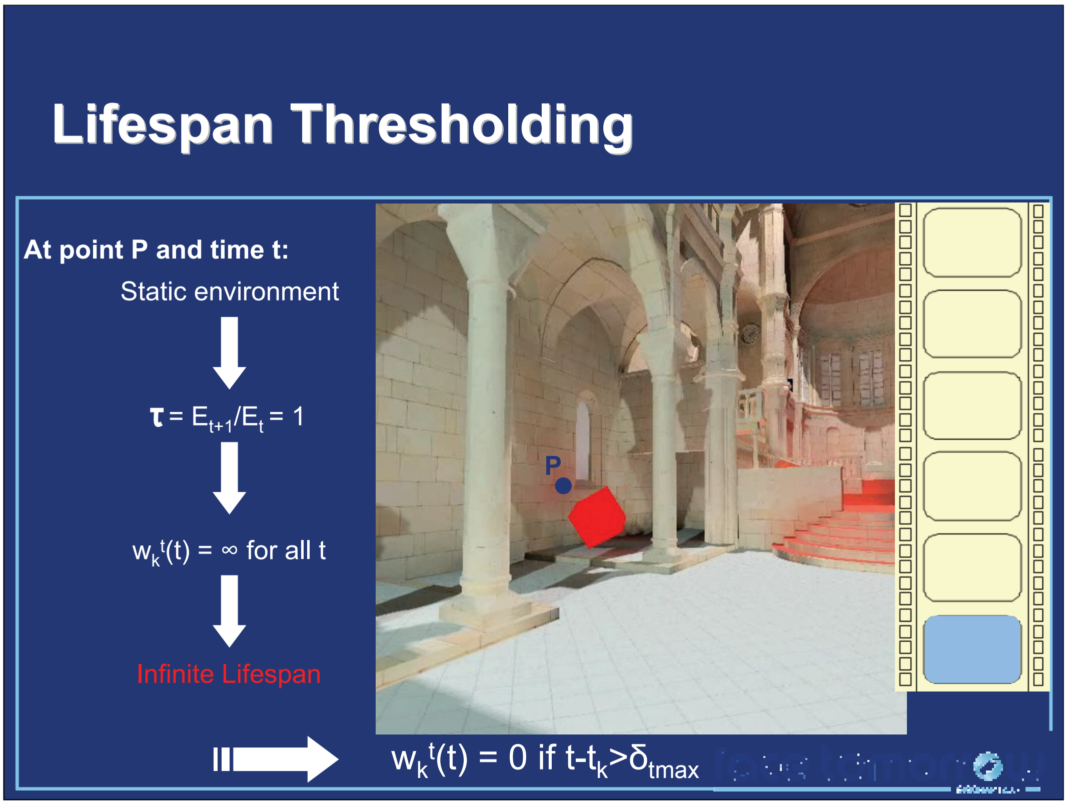
Static environment

$\tau = E_{t+1}/E_t = 1$

$w_k^t(t) = \infty$ for all t

Infinite Lifespan

Let us consider an example: at a point P and time t and t+1, the environment is static. Therefore, the indirect lighting is considered constant. In this case, the value of the temporal weighting function is infinite for any frame. The lifespan of the record is then considered infinite.



Lifespan Thresholding

At point P and time t:

Static environment

↓

$\tau = E_{t+1}/E_t = 1$

↓

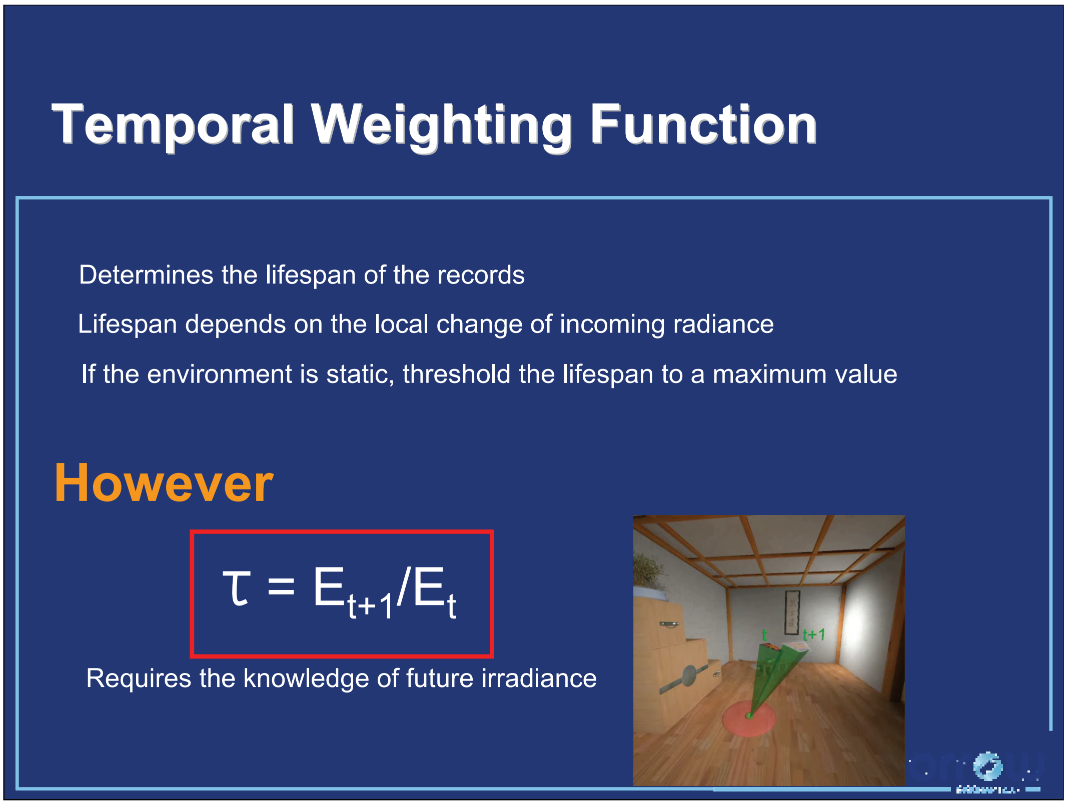
$w_k^t(t) = \infty$ for all t

↓

Infinite Lifespan

⇒ $w_k^t(t) = 0$ if $t-t_k > \delta_{tmax}$

However, it is easy to show that the lighting at P may change afterwards. Since the weighting function returns an infinite value regardless of the frame, the value of the lighting at P will never be updated, yielding ghosting artifacts. We do not solve this problem, but we simply ask the user to define a maximum lifespan for all records. This maximum lifespan ensures the update of the ligthing at least after



Temporal Weighting Function

Determines the lifespan of the records

Lifespan depends on the local change of incoming radiance

If the environment is static, threshold the lifespan to a maximum value

However

$$\boxed{\tau = E_{t+1}/E_t}$$

Requires the knowledge of future irradiance

The temporal weighting function is used to determine the lifespan of each record based on the local change of incoming radiance over time. The length of the lifespan is shortened or lengthened with respect to the magnitude of the change of indirect lighting. However, the estimate of this change is based on the knowledge of the indirect lighting at next frame, E_{t+1}. Since this value is generally unknown, next section will devise a method for fast estimation of the future incoming without actual sampling.

Contributions

Temporal (ir)radiance interpolation scheme

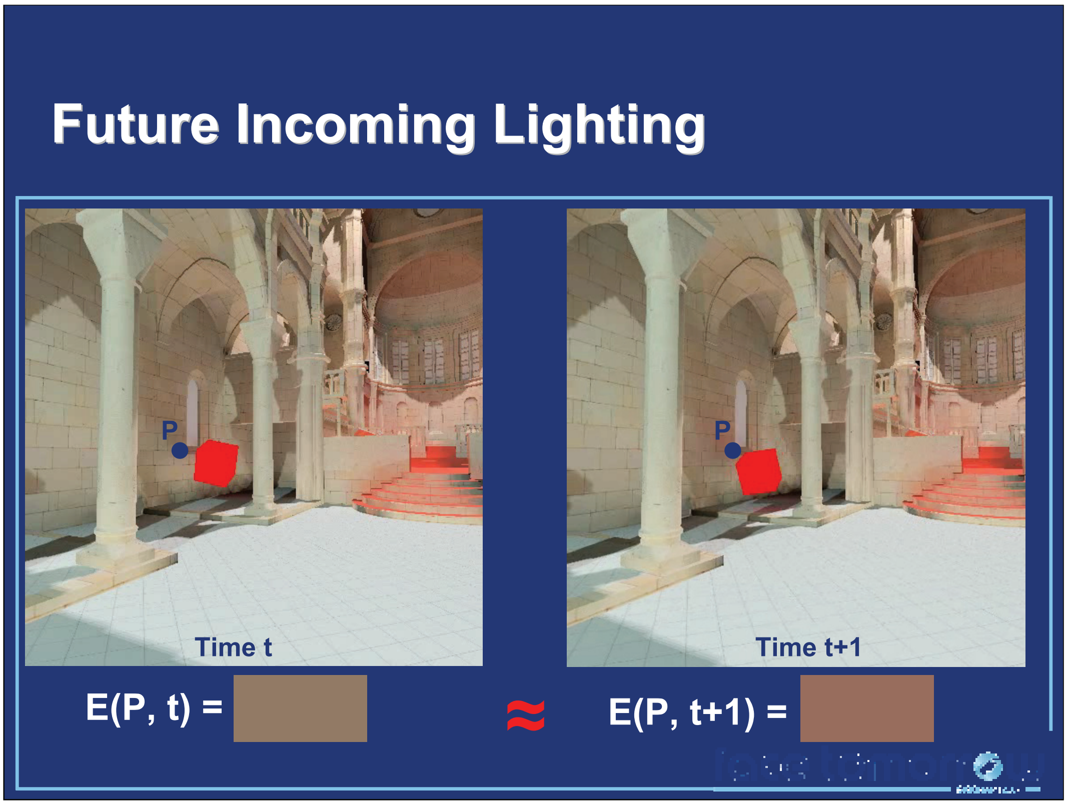
Flying Kite- Interpolated gradients
Rendering time: 783.4s
Speedup: 6.52x

Flying Kite - No temporal coherence
Rendering time: 5109s

Temporal weighting function

Fast estimate of future indirect lighting

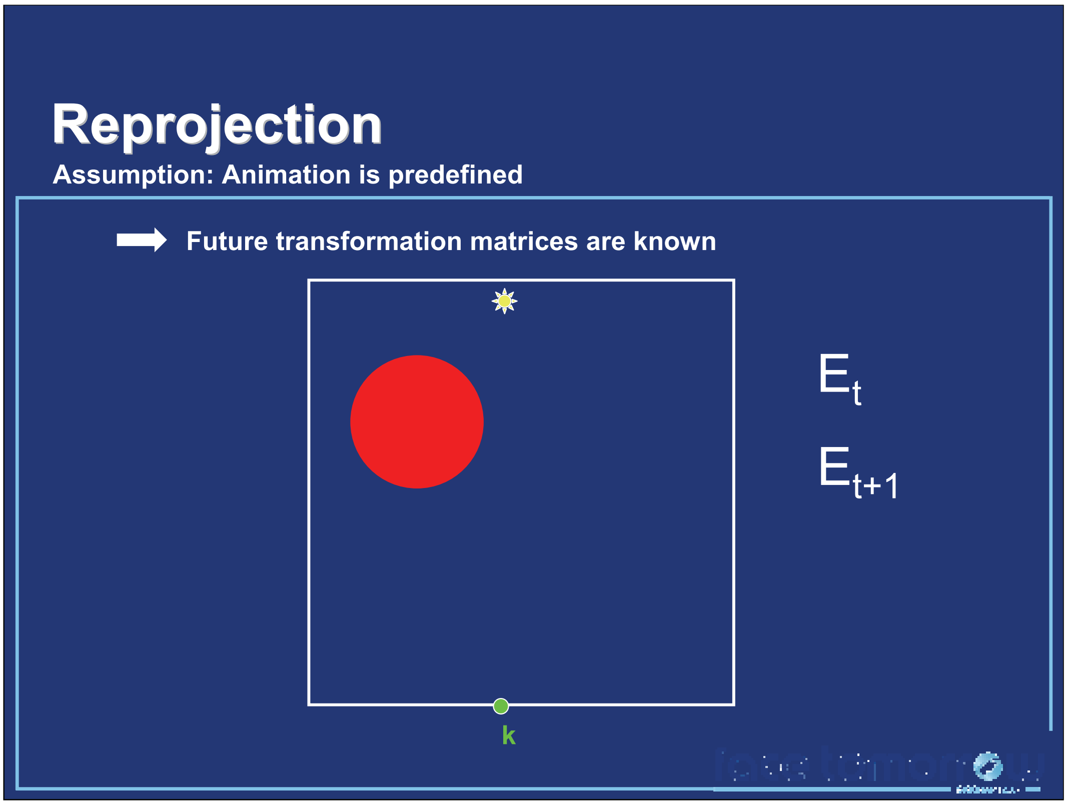
Temporal gradients



Future Incoming Lighting

$E(P, t) =$ $\approx$ $E(P, t+1) =$

This approach is based on the following observation: between time t and t+1, the change of lighting is low. Based on the knowledge of the dynamic properties of surrounding objects, we propose a method based on reprojection to estimate the future incoming lighting using only the data available at current frame.



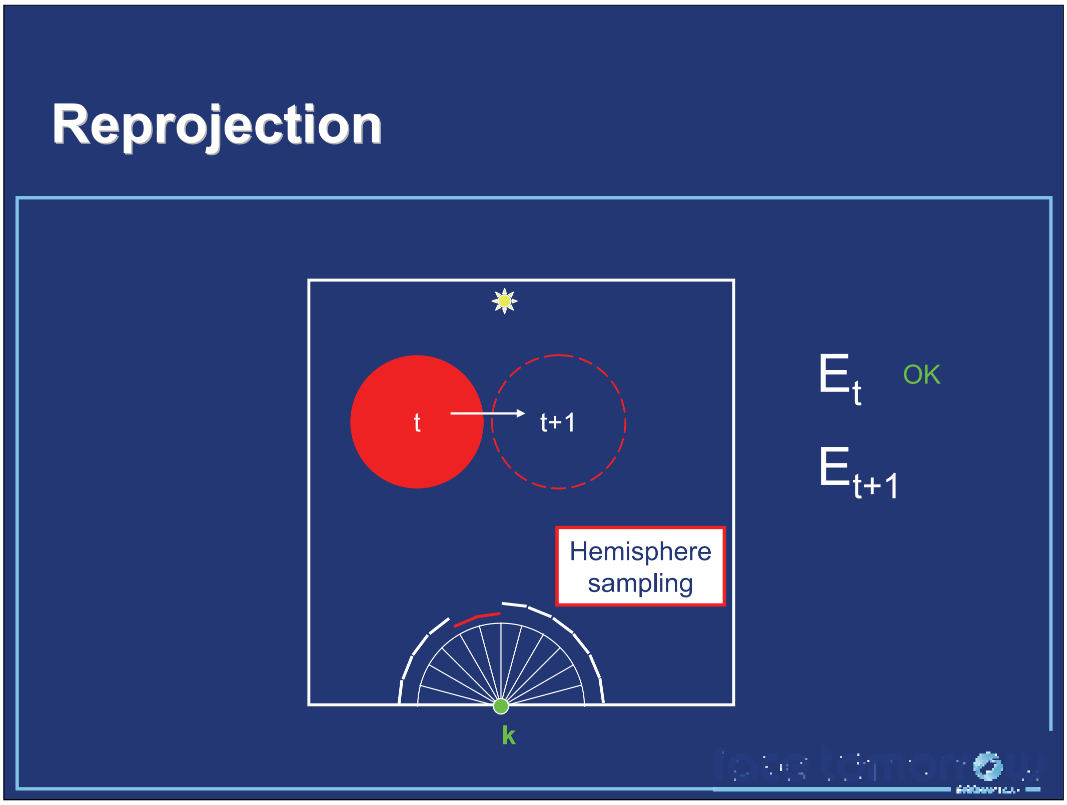
Reprojection

Assumption: Animation is predefined

➡ Future transformation matrices are known

Our reprojection method is similar to the one used in the Render Cache [Walter99]. However, in the Render Cache, the reprojection is used to avoid tracing primary rays from the camera. In our case, the reprojection is only used to estimate the future incoming radiance at the location of a record.

Let us consider a point k at which we want to create a record.



Reprojection

t

t+1

Hemisphere sampling

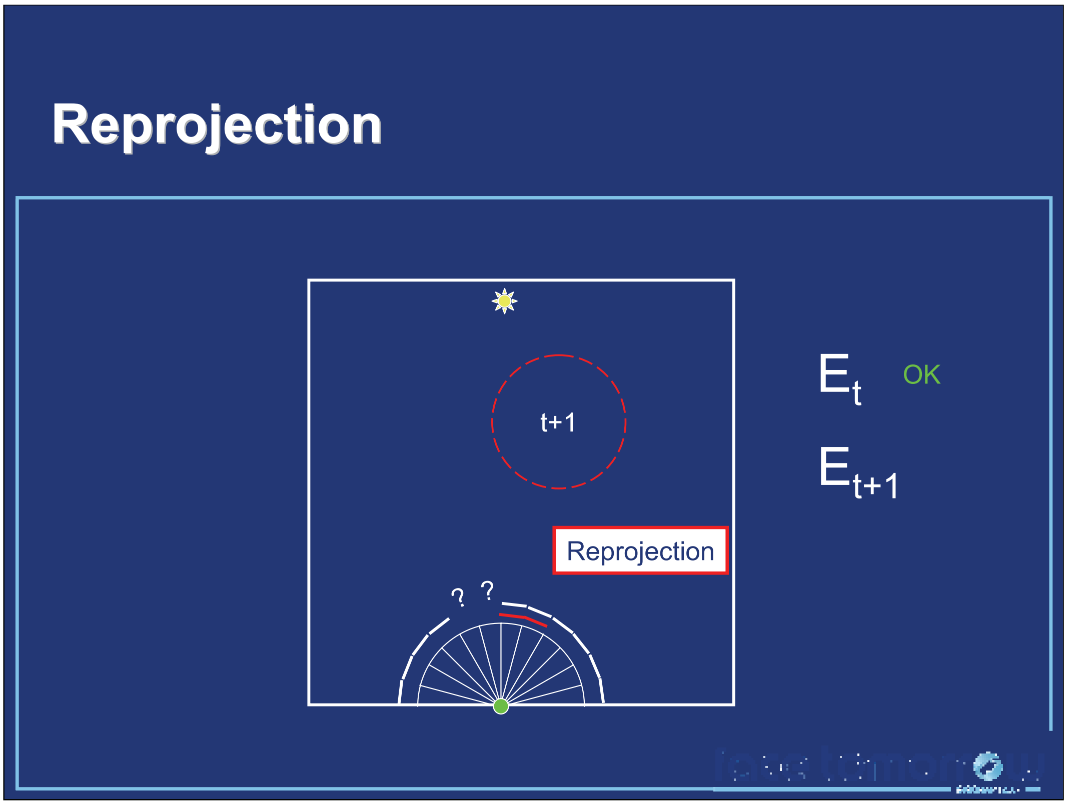
k

E_t OK

E_{t+1}

The first step is the computation of the incoming radiance in k at time t. This is performed by sampling the hemisphere above k either using ray tracing or GPU rasterization as explained in the previous chapter.

Assuming the animation is known in advance, the position of the red sphere at time t+1 is known.



Reprojection

t+1

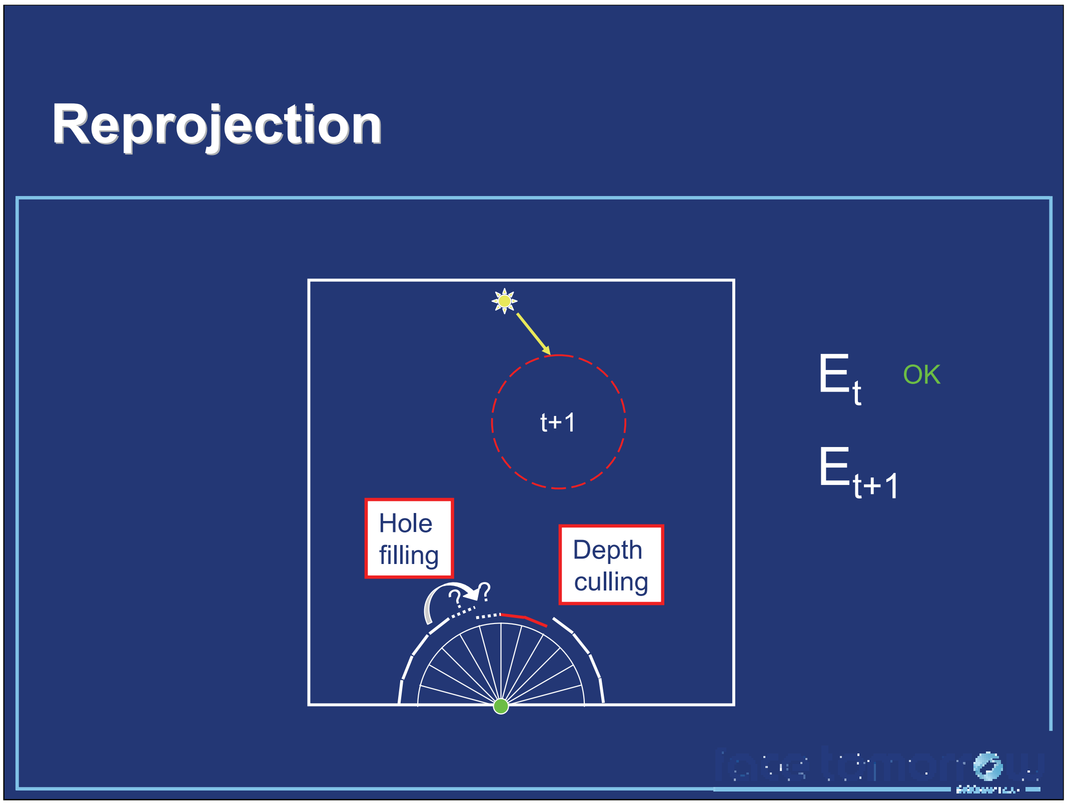
Reprojection

? ?

E_t OK

E_{t+1}

This future position of the red sphere is used to reproject the radiance samples corresponding to the sphere to the new position. This reprojection yields missing information at the former location of the sphere, and overlapping values at the novel position.



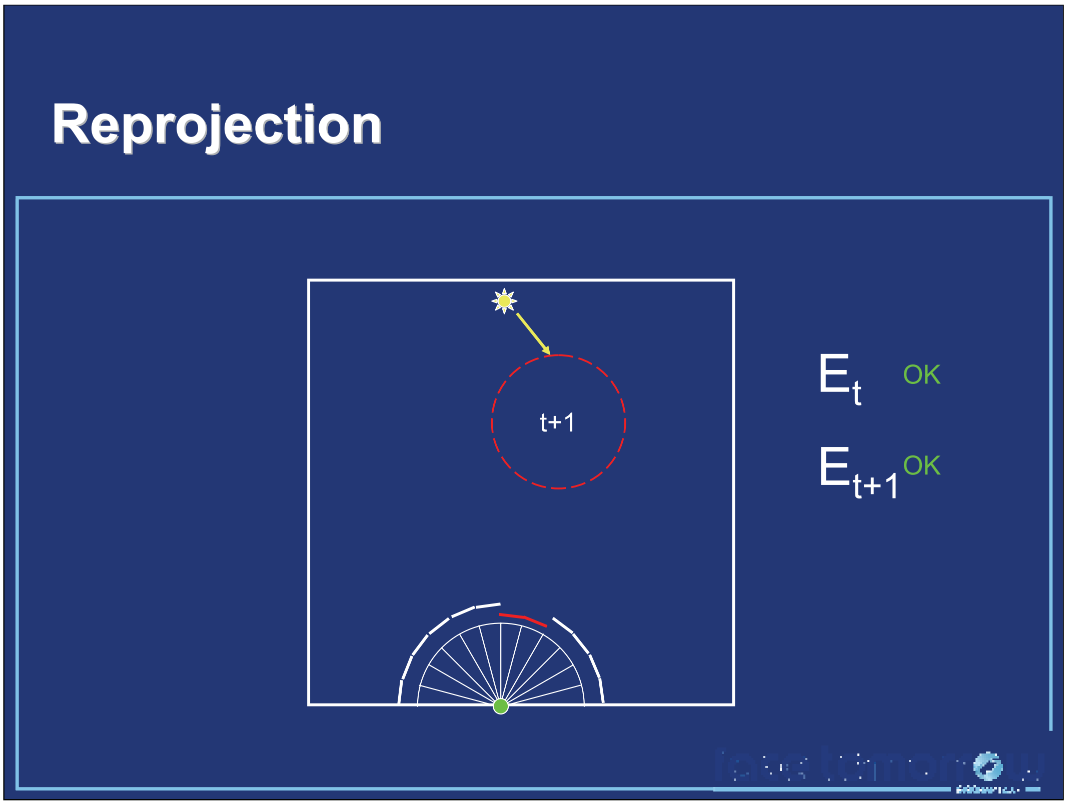
Reprojection

E_t OK

E_{t+1}

A depth culling step chooses the closest value in the case of overlapping radiance values. In this case, the background values are culled.

A simple hole-filling is applied in strata where information is missing. Since we only deal with small movements (1 frame), we simply fill the holes using the closest neighboring value.

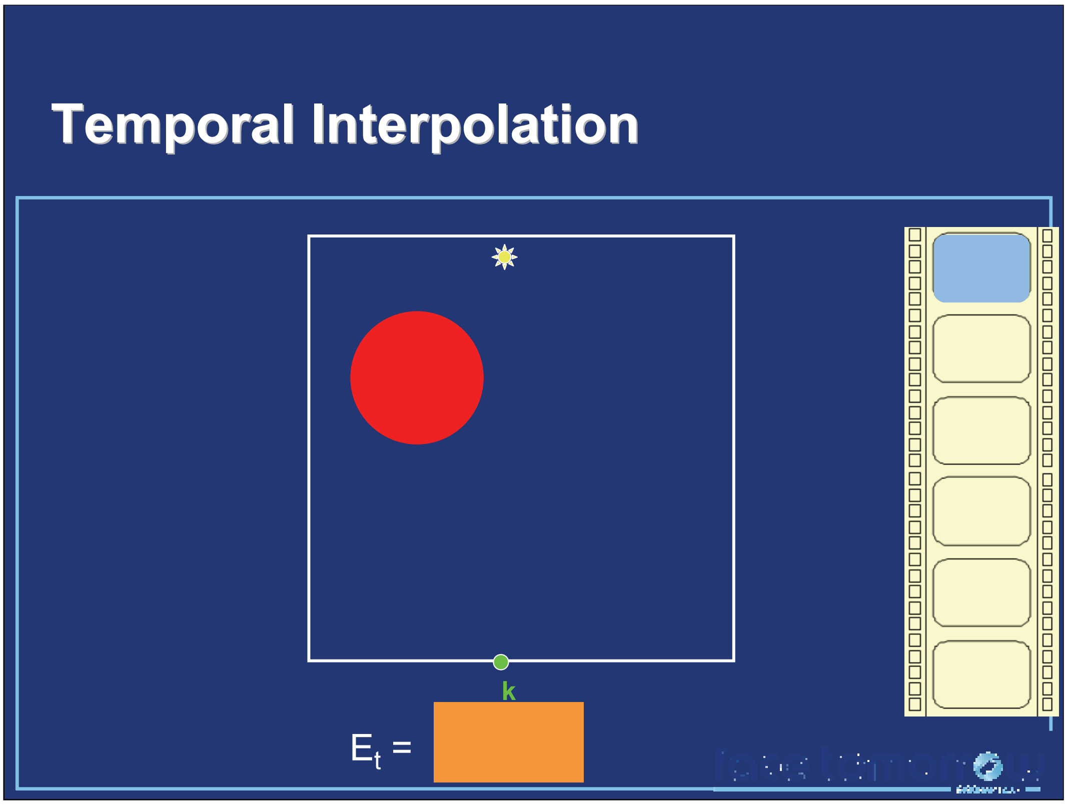


Reprojection

E_t OK

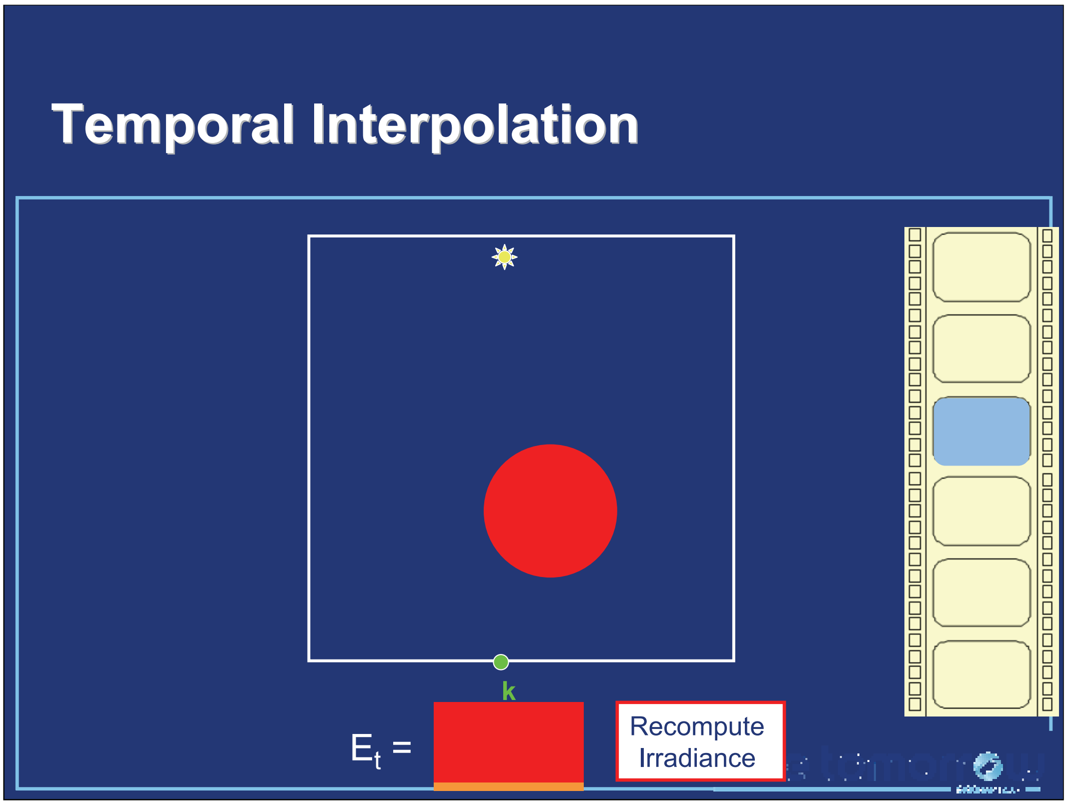
E_{t+1} OK

After depth culling and hole filling, our method provides a reliable estimate of the future incoming lighting. The lifespan of the record created in k is thus completely defined.



Temporal Interpolation

At this point, we know that the value of record k can be reused across n frames.

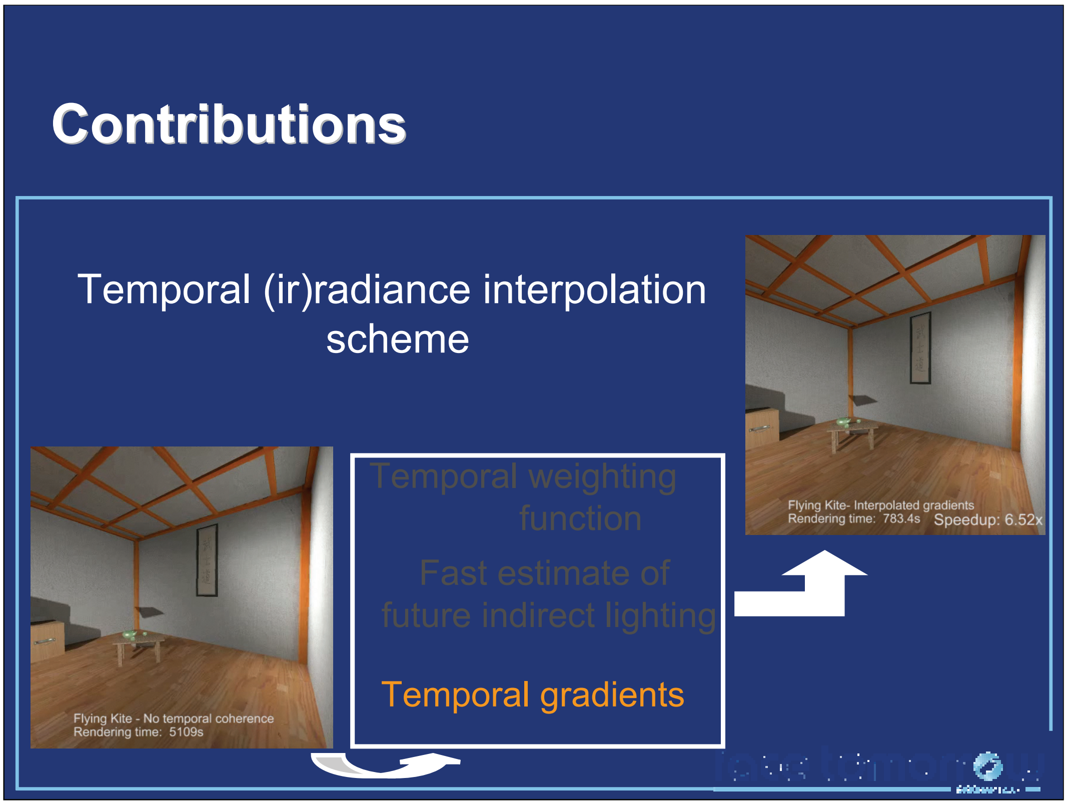


Temporal Interpolation

$E_t =$

Recompute Irradiance

At the end of the record lifespan, a new record is computed, containing an up-to-date irradiance value. In this case the red sphere got closer to k, hence the irradiance at time t and t+n are noticeably different. A consequence of this brutal replacement is sudden changes in the color of image portions, known as « popping » artifacts.



Contributions

Temporal (ir)radiance interpolation scheme

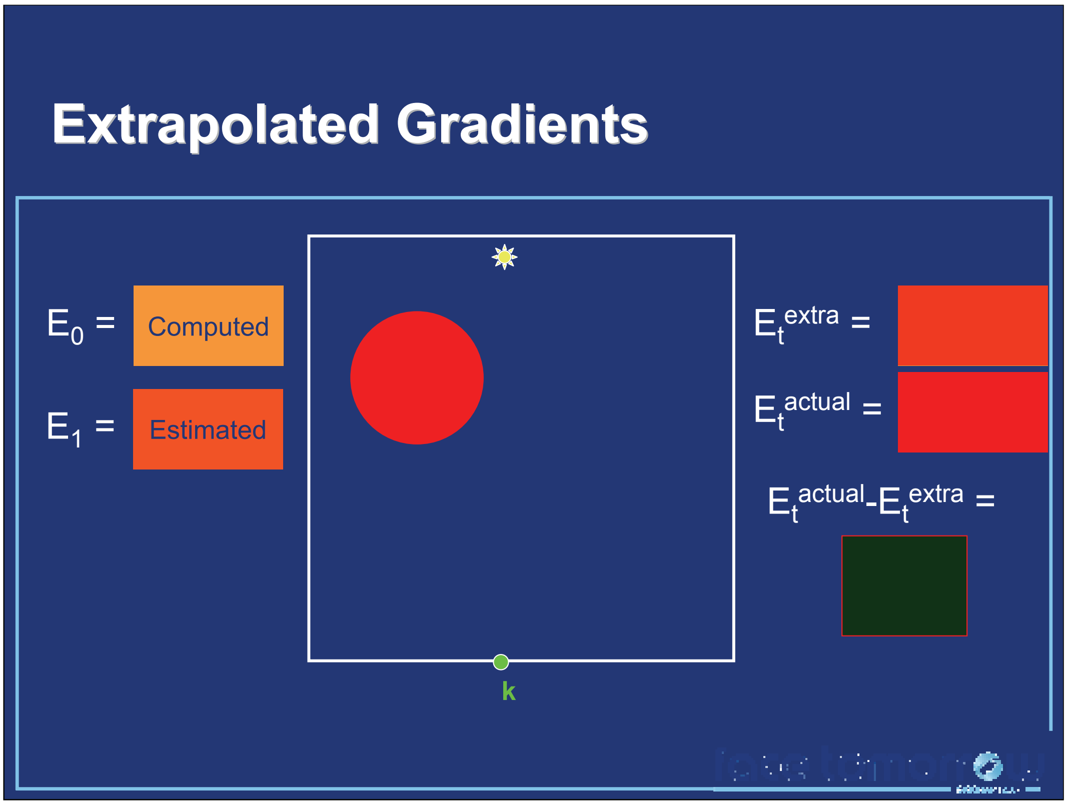
Flying Kite - No temporal coherence
Rendering time: 5109s

Temporal weighting function

Fast estimate of future indirect lighting

Temporal gradients

Flying Kite- Interpolated gradients
Rendering time: 783.4s Speedup: 6.52x



Extrapolated Gradients

E_0 = Computed

E_1 = Estimated

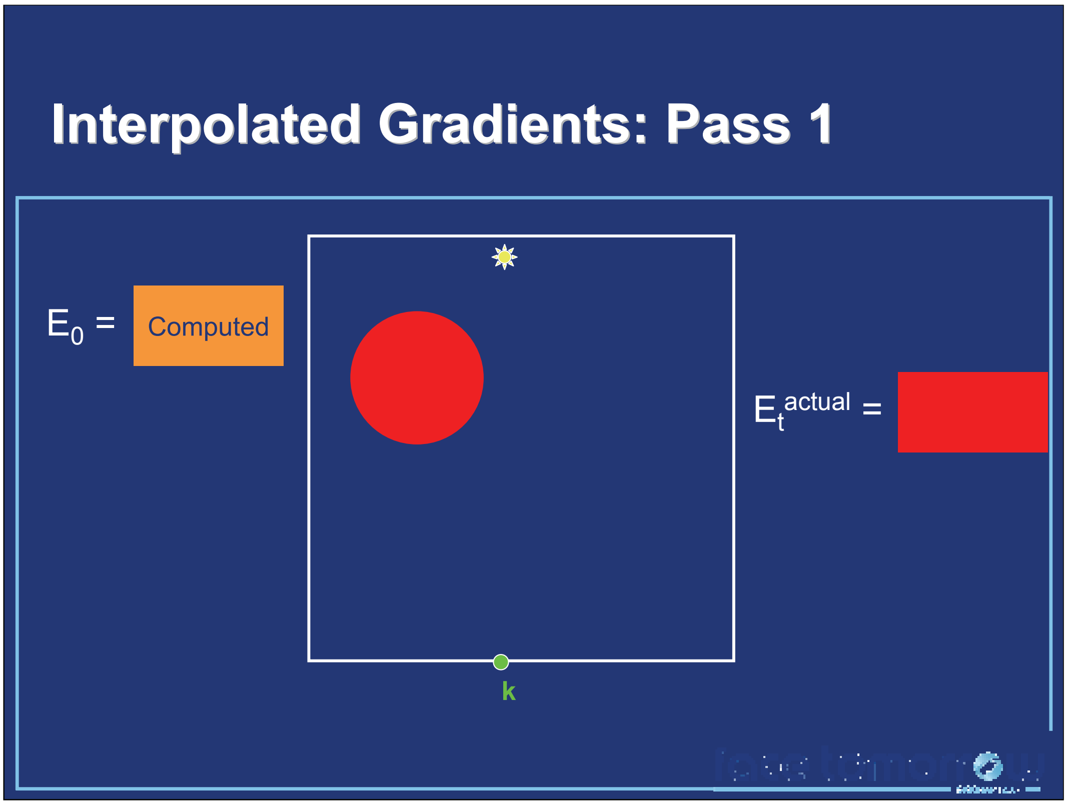
E_t^{extra} =

E_t^{actual} =

E_t^{actual}-E_t^{extra} =

First, we propose to use the estimate of the future incoming lighting to extrapolate the lighting over the entire lifespan of the record. While this method reduces the gap between the extrapolated and the actual irradiance values, the difference is still not negligible. As a consequence, some popping artifacts will remain visible.

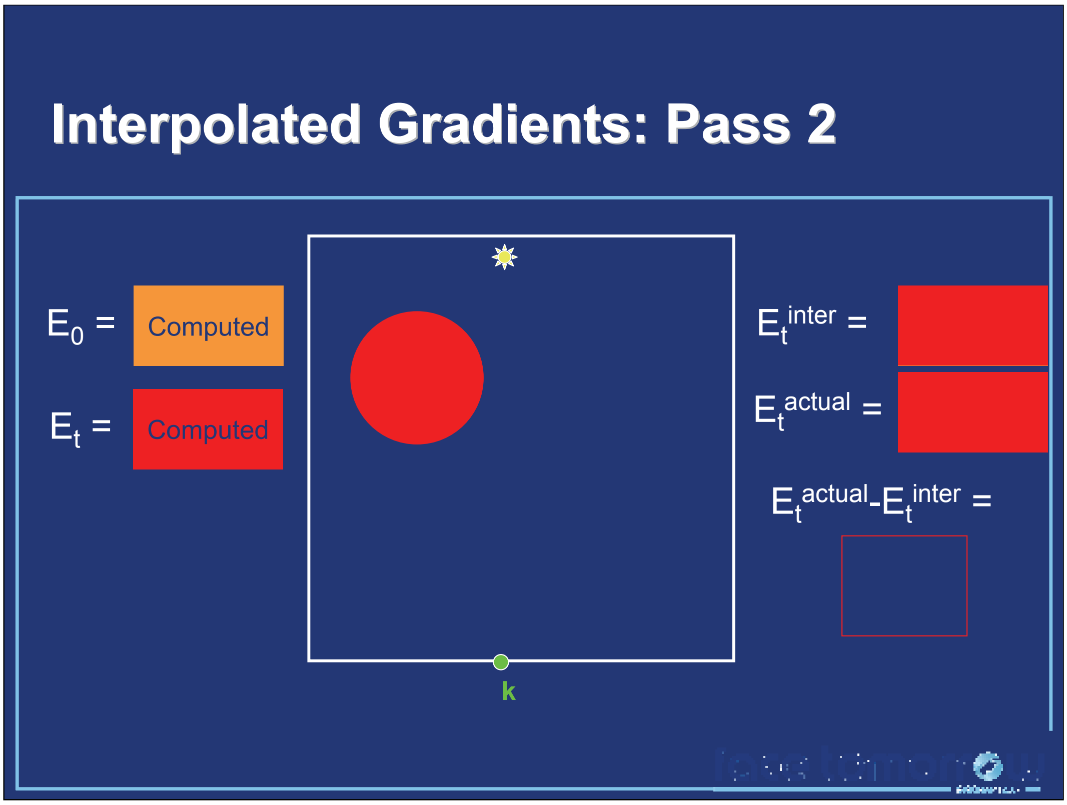
This method has one major advantage: the indirect lighting can be computed and displayed on the fly, as the animation is played. Particularly, such gradients could be used in the context of interactive global illumination computation.



Interpolated Gradients: Pass 1

Interpolated gradients completely avoid the popping artifacts by temporally interpolating the irradiance. In a first pass, records are generated as explained previously: each record is used within its lifespan, and new records are computed as the lifespans expire. Let us consider a record R_0 computed at time t_0 and located at point k. When this record expires after n frames, it is replaced by a new record R_1 also located at point k. The temporal gradient of R_0 is then deduced from $(E_{R1}-E_{R0})/n$.

This gradient approximates the change of lighting within the lifespan of a record by linearly interpolating the irradiance at the beginning and at the end of the lifespan. While completely removing popping artifacts, this linear approximation may smooth out high frequency changes that may happen during the lifespan of the record. Accounting for such changes either require a reduction of a^t and δ_{tmax}, or the definition of a higher order interpolation scheme.



Interpolated Gradients: Pass 2

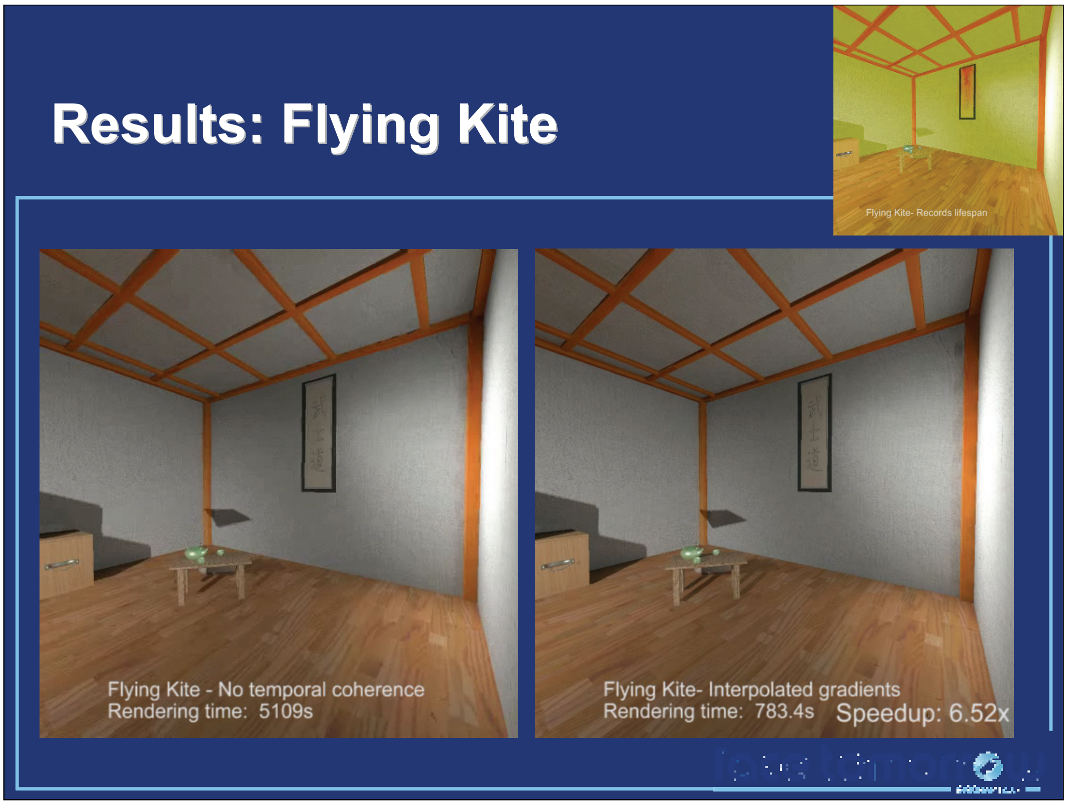
$E_0 =$ Computed

$E_t =$ Computed

$E_t^{inter} =$

$E_t^{actual} =$

$E_t^{actual} - E_t^{inter} =$



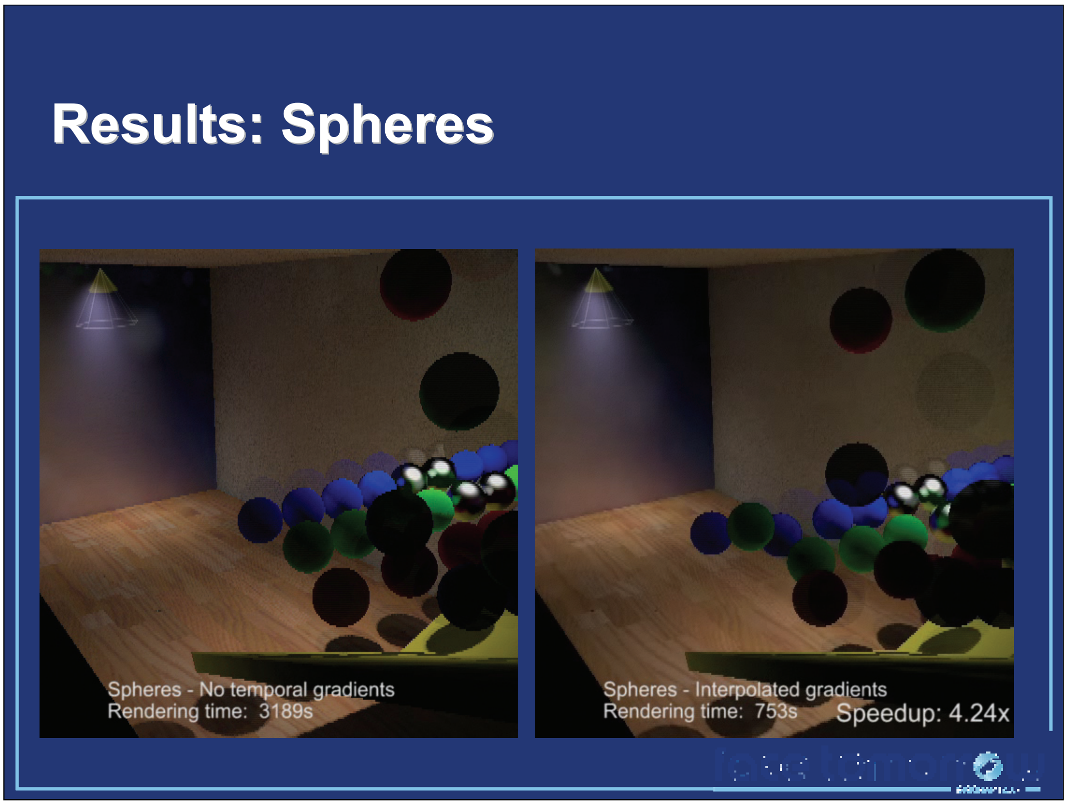
Results: Flying Kite

Flying Kite- Records lifespan

Flying Kite - No temporal coherence
Rendering time: 5109s

Flying Kite- Interpolated gradients
Rendering time: 783.4s Speedup: 6.52x

Videos illustrating the interpolated gradients and the adaptive record lifespan in different scenes can be found in [MyWebSite].



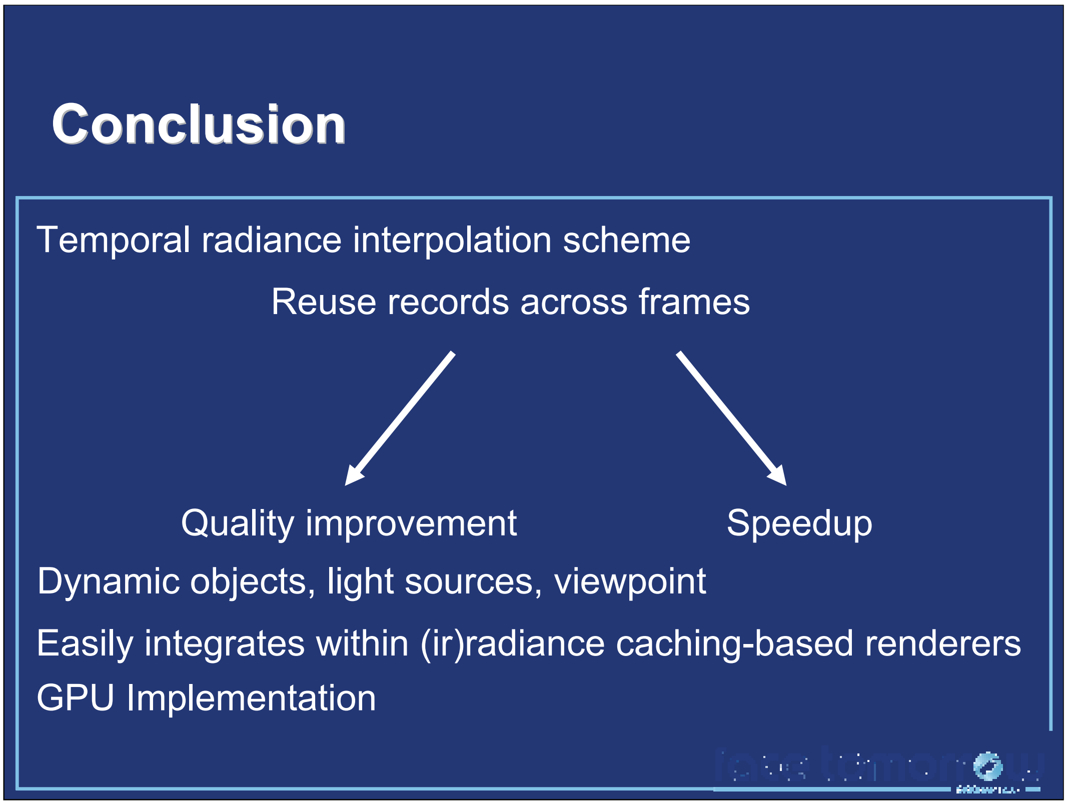
Results: Spheres

Spheres - No temporal gradients
Rendering time: 3189s

Spheres - Interpolated gradients
Rendering time: 753s

Speedup: 4.24x

This method can be straightforwardly extended to nondiffuse interreflexions using radiance caching.



Conclusion

Temporal radiance interpolation scheme

Reuse records across frames

Quality improvement

Speedup

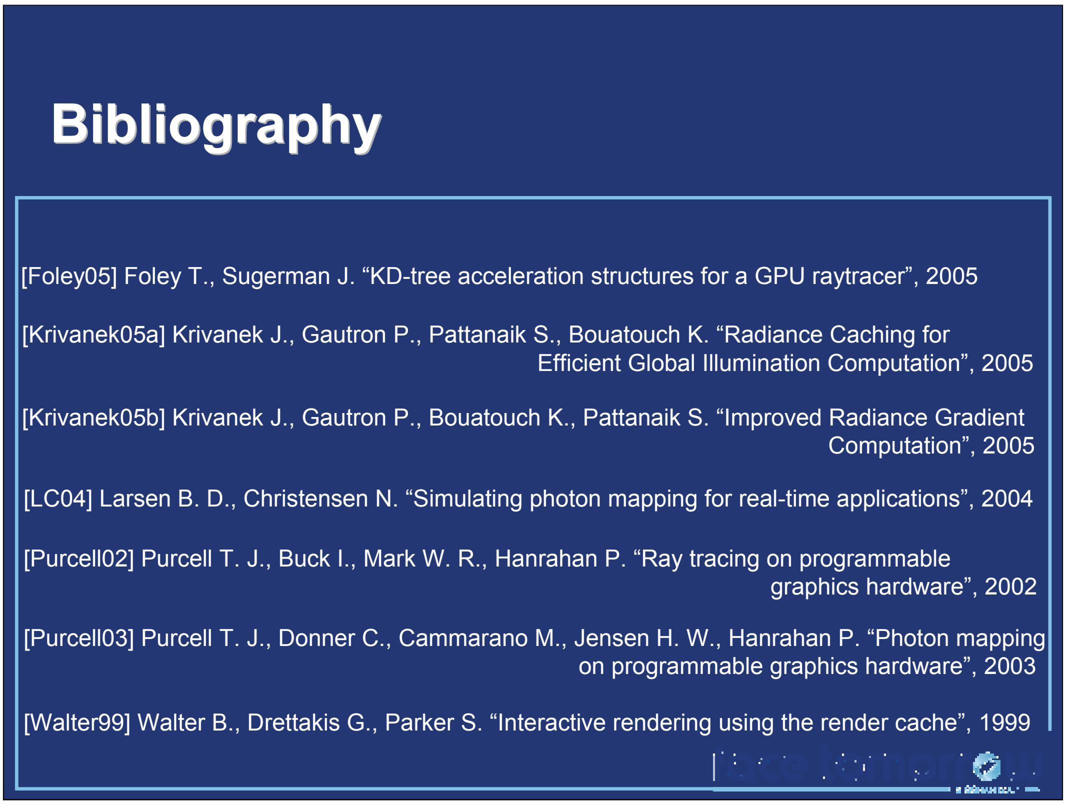
Dynamic objects, light sources, viewpoint

Easily integrates within (ir)radiance caching-based renderers

GPU Implementation

The method described in this part is based on the reuse of irradiance records across frames. While reducing the computational cost of animation rendering, the flickering artifacts are drastically reduced. This method supports any type of dynamic scene components, and can be easily integrated in existing renderers.

Future work will consider the elimination of the maximum lifespan δ_{tmax}. Also, we would like to devise methods for higher order temporal interpolation to account for sharp changes of indirect lighting.



Bibliography

[Foley05] Foley T., Sugerman J. “KD-tree acceleration structures for a GPU raytracer”, 2005

[Krivanek05a] Krivanek J., Gautron P., Pattanaik S., Bouatouch K. “Radiance Caching for Efficient Global Illumination Computation”, 2005

[Krivanek05b] Krivanek J., Gautron P., Bouatouch K., Pattanaik S. “Improved Radiance Gradient Computation”, 2005

[LC04] Larsen B. D., Christensen N. “Simulating photon mapping for real-time applications”, 2004

[Purcell02] Purcell T. J., Buck I., Mark W. R., Hanrahan P. “Ray tracing on programmable graphics hardware”, 2002

[Purcell03] Purcell T. J., Donner C., Cammarano M., Jensen H. W., Hanrahan P. “Photon mapping on programmable graphics hardware”, 2003

[Walter99] Walter B., Drettakis G., Parker S. “Interactive rendering using the render cache”, 1999

Bibliography

[Ward88] Ward G. J., Rubinstein F. M., Clear R. D. “A ray tracing solution for diffuse interreflection”, 1988

[Ward92] Ward G. J., Heckbert P. S. “Irradiance gradients”, 1992

[Wil78] Williams L. “Casting curved shadows on curved surfaces”, 1978

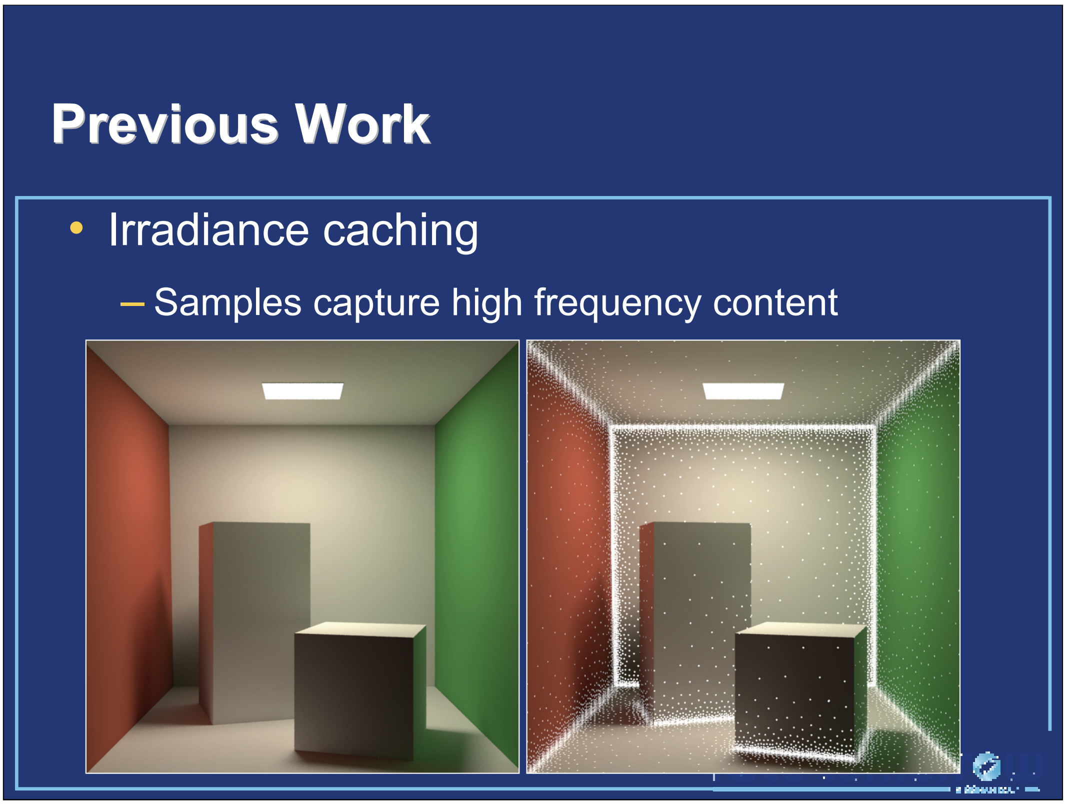
Additional materials available on [MyWebSite]:
http://www.irisa.fr/siames/Pascal.Gautron/

SIGGRAPH2007

Irradiance Decomposition

SIGGRAPH2007

Okan Arikan



Previous Work

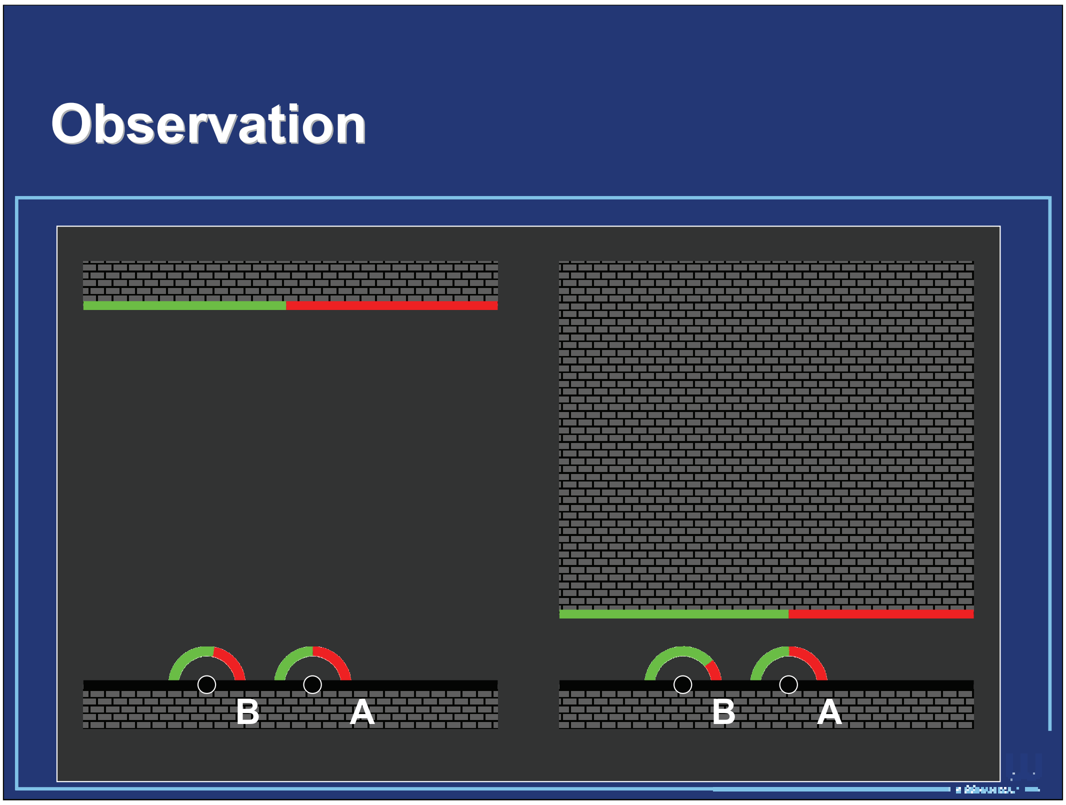
- Irradiance caching
  - Samples capture high frequency content

-Here’s an example of irradiance caching

-The picture on the right shows the sample locations

-Notice that samples are concentrated around the corners because the indirect illumination can change quickly around these regions

-The reason we sample densely around corners, or high geometric detail is to capture rapid changes in global illumination



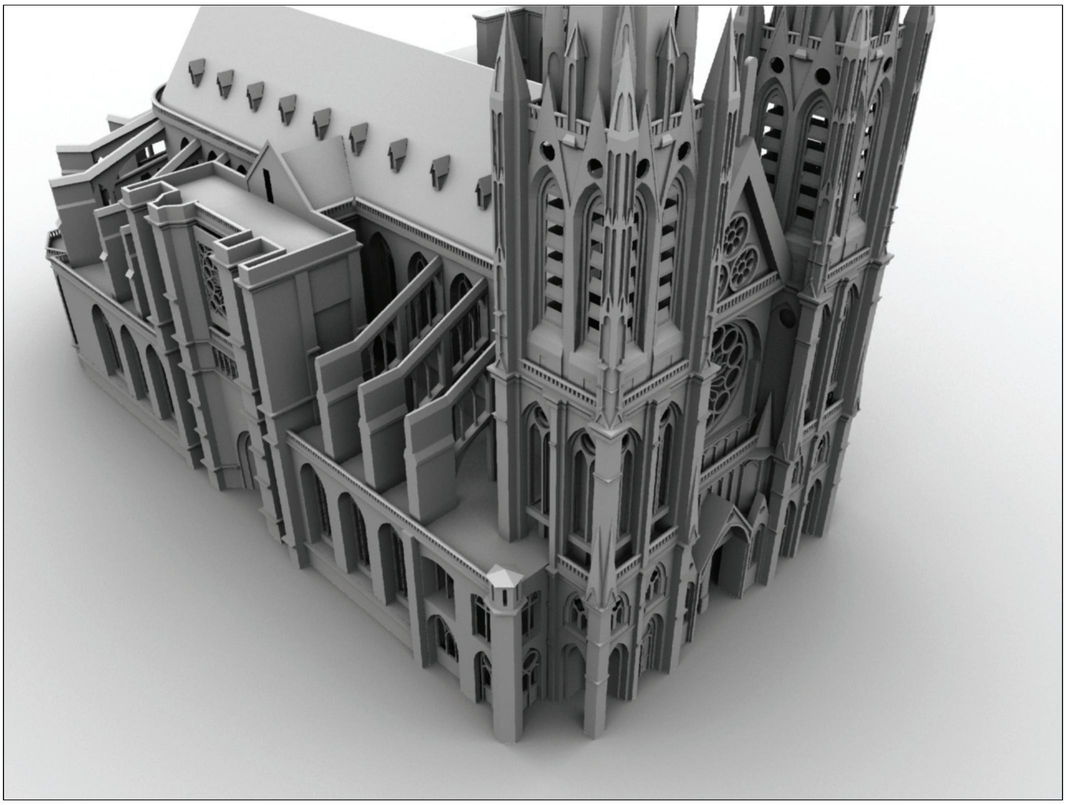
Observation

•You can see why we need denser sampling around corners and high geometric detail by visualizing this scene

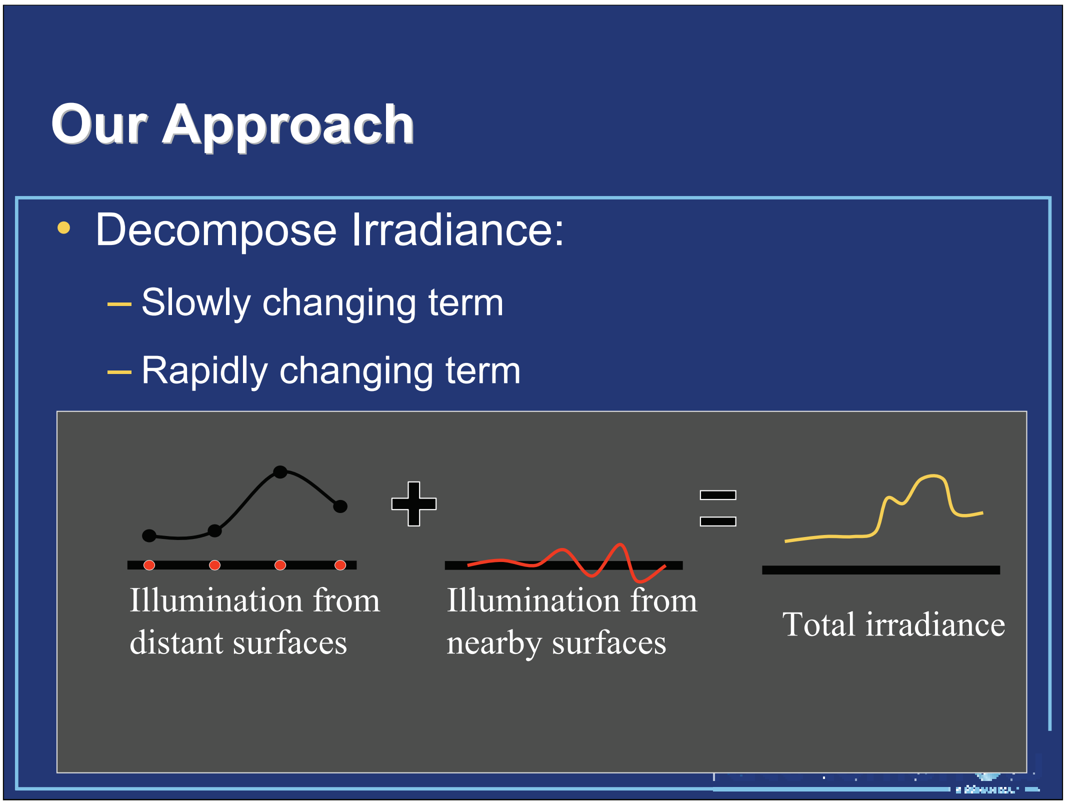
•As the ceiling is far away, when we go from point B to point A, their incident hemispheres do not change much

•If the ceiling was lower, then what B sees can be very different from what A sees

•This is why irradiance caches must sample densely around high geometric detail



-Because of the dense sampling around high geometric detail, irradiance caching can become inefficient for geometries like this cathedral.



Our Approach

- Decompose Irradiance:
  - Slowly changing term
  - Rapidly changing term

Illumination from distant surfaces + Illumination from nearby surfaces = Total irradiance

-In our method, we propose decomposing irradiance into two terms

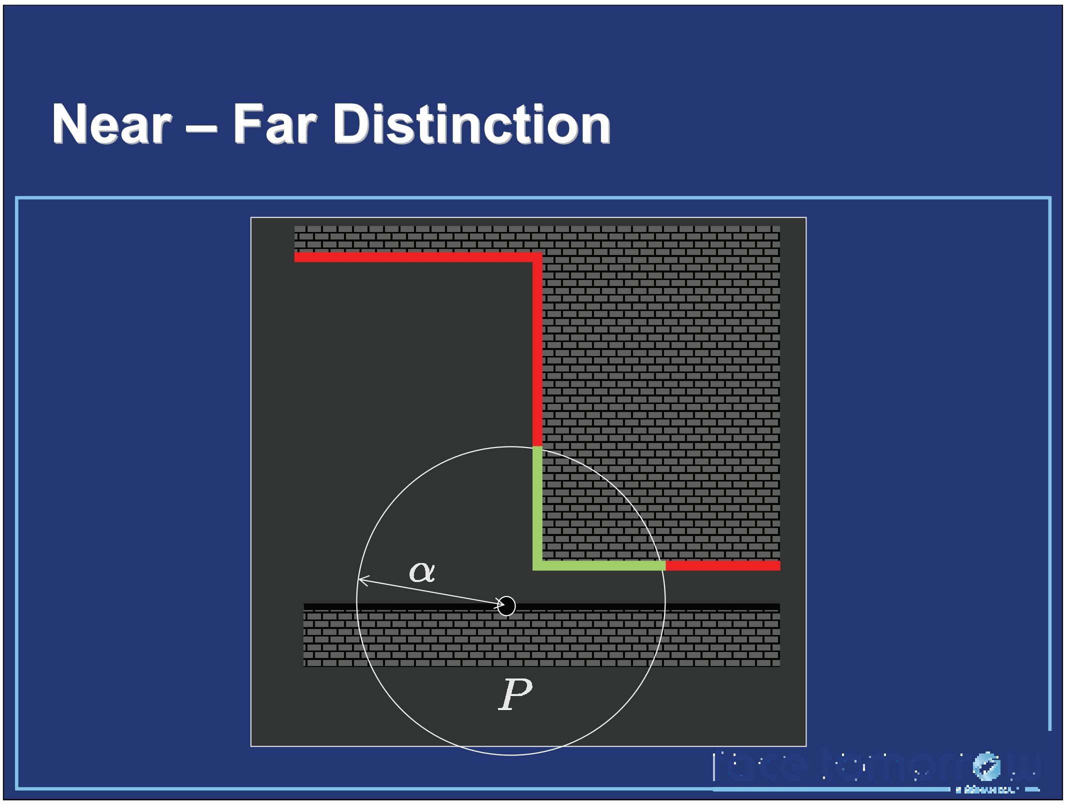
-Irradiance coming from distant surfaces

-And irradiance coming from nearby surfaces

-Because of our observation, irradiance coming from distant surfaces is smooth and can be interpolated using scattered data interpolation

-The irradiance coming from nearby surfaces is local and can be computed efficiently

-We will focus on diffuse surfaces, although I imagine this method can be incorporated into Jarda's radiance caching mechanism for more complicated BRDFs

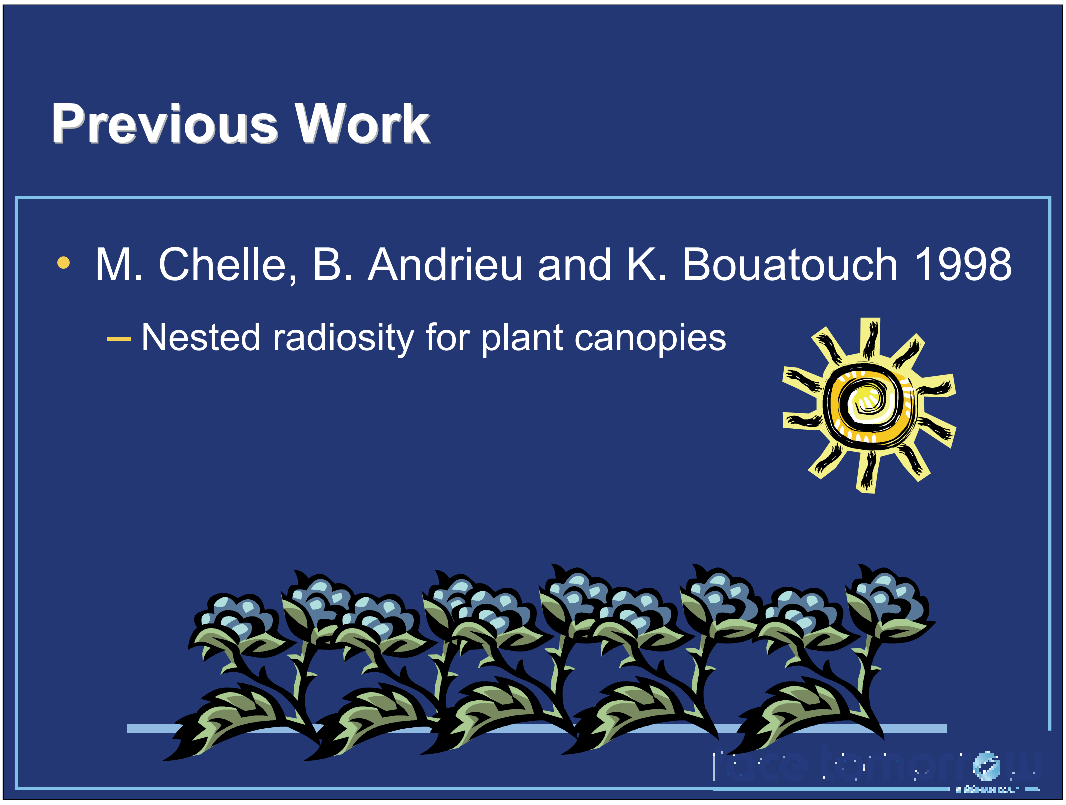


Near – Far Distinction

•Before we go further, we need a definition for what's near and what's far

•Given a point P, we assume all the surfaces within the sphere centered at P with radius alpha are near

•In this figure, green surfaces are near and red ones are far



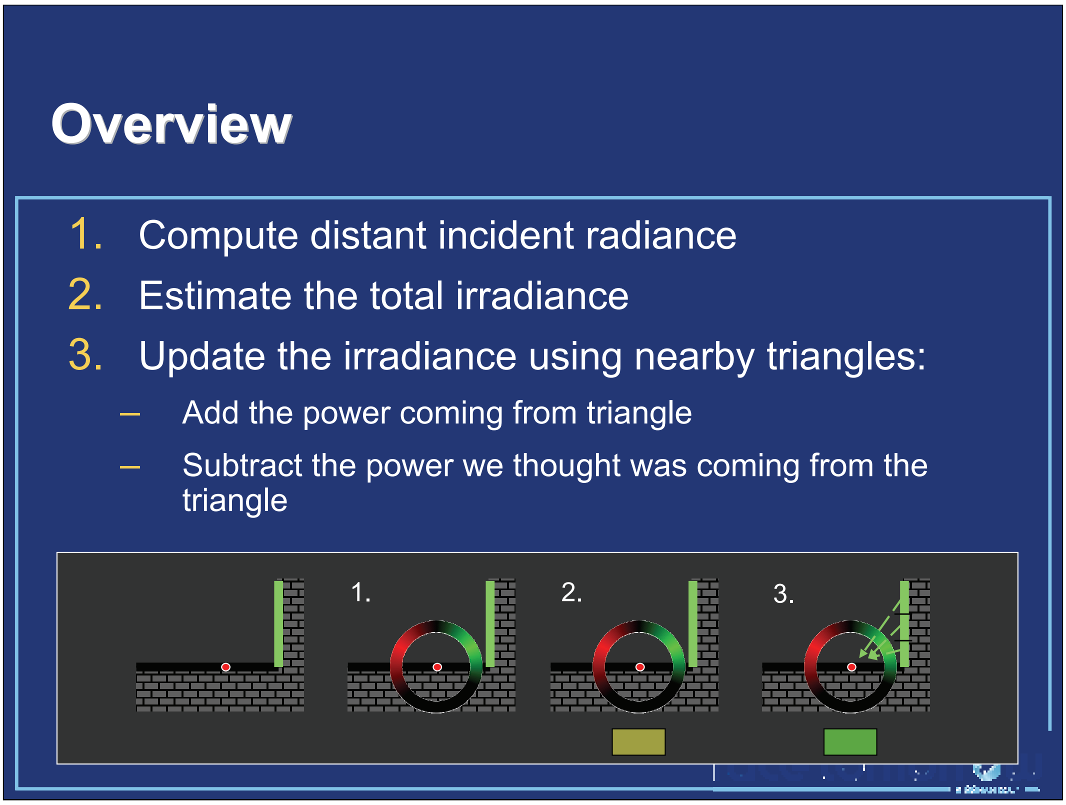
Previous Work

- M. Chelle, B. Andrieu and K. Bouatouch 1998
  - Nested radiosity for plant canopies

•Similar decomposition of irradiance has been proposed by [Chelle et al. 1998]

•They introduce a full radiosity framework for capturing power transfer between nearby leaves and a volumetric model for power transfer from distant leaves

•But their method is geared towards full global illumination computation and plant canopies



Overview

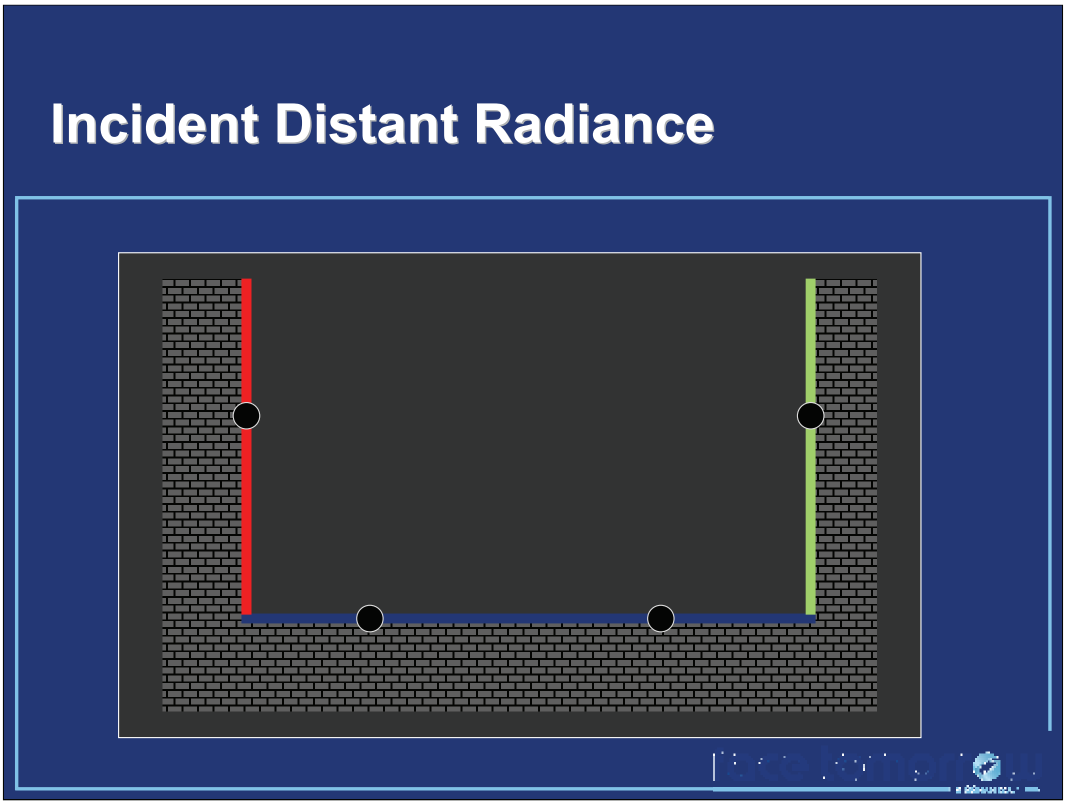
1. Compute distant incident radiance
2. Estimate the total irradiance
3. Update the irradiance using nearby triangles:
   - Add the power coming from triangle
   - Subtract the power we thought was coming from the triangle

•To compute the irradiance integral at a point, our algorithm follows three steps

•1 – We compute a representation of the incident radiance from only distant surfaces, so nearby surfaces are not accounted for

•2- We compute the irradiance integral over this representation

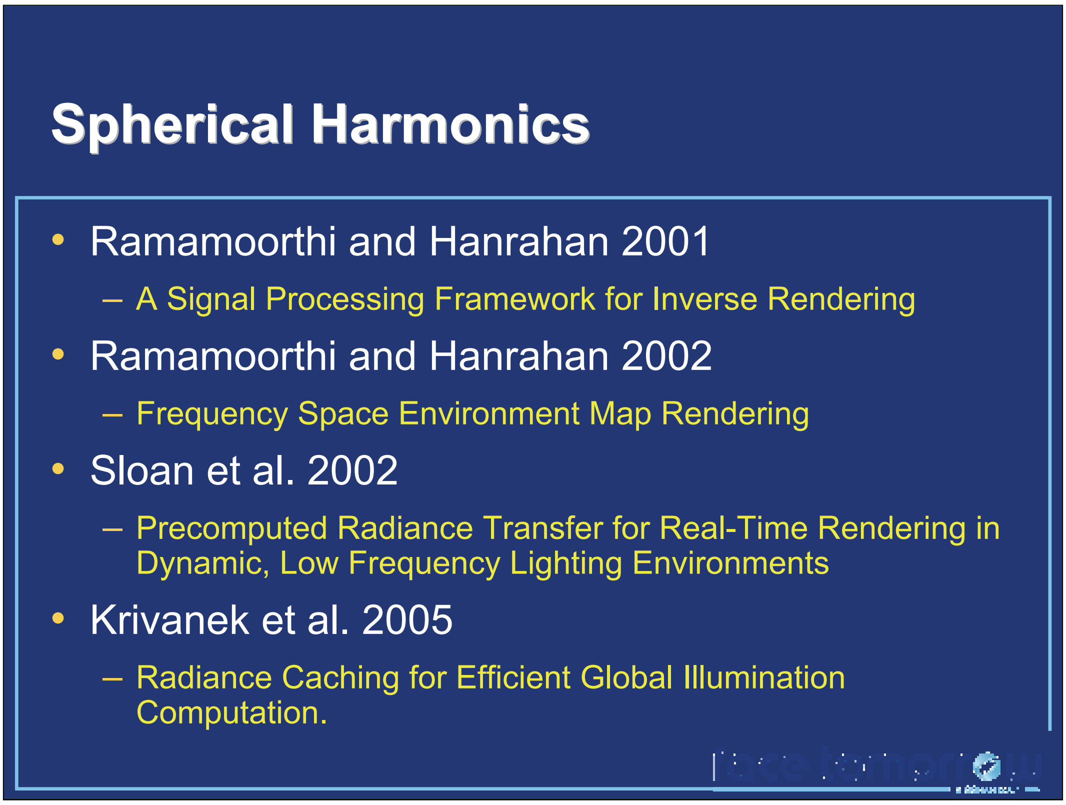
•3- We go over nearby scene triangles and update the irradiance integral by taking the power that they reflect or occlude into account



Incident Distant Radiance

-To compute the incident radiance from distant surfaces we use scattered data interpolation

-We compute the incident radiance at some points, similar to irradiance caching



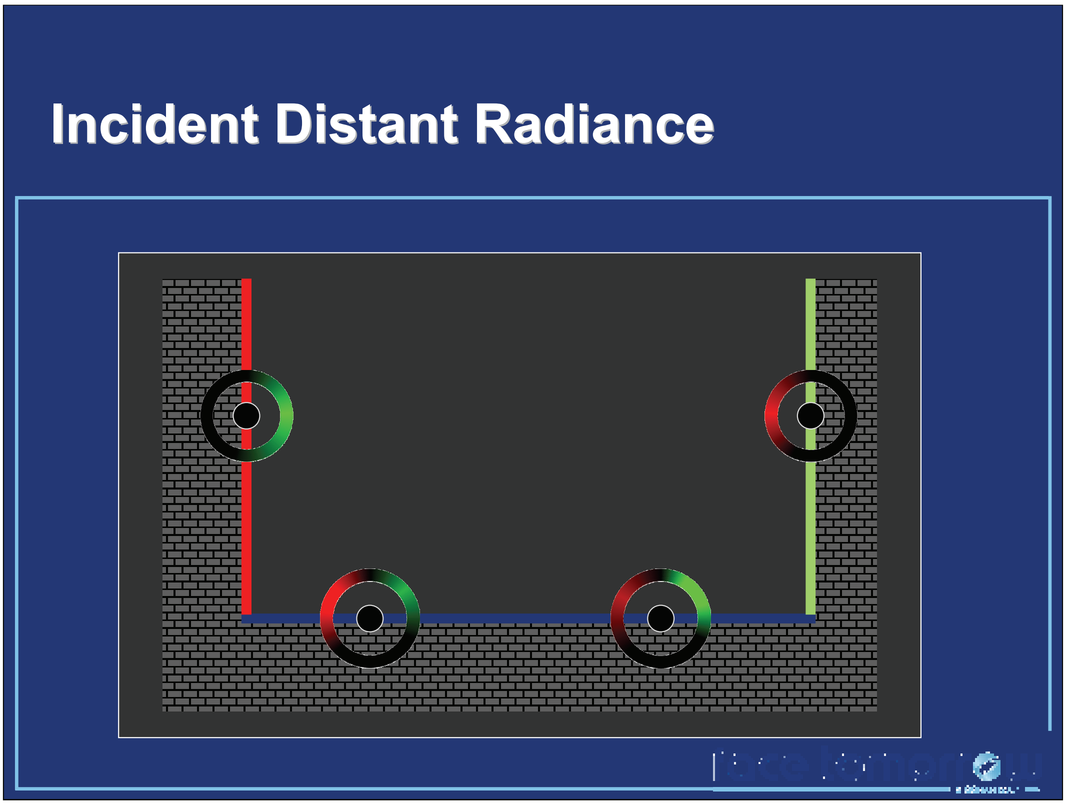
Spherical Harmonics

- Ramamoorthi and Hanrahan 2001
  - A Signal Processing Framework for Inverse Rendering
- Ramamoorthi and Hanrahan 2002
  - Frequency Space Environment Map Rendering
- Sloan et al. 2002
  - Precomputed Radiance Transfer for Real-Time Rendering in Dynamic, Low Frequency Lighting Environments
- Krivanek et al. 2005
  - Radiance Caching for Efficient Global Illumination Computation.

-To represent incident radiance, we use spherical harmonics (SH)

-The advantages of using this representation have been shown previously in these excellent papers

-Notice that this is very similar to the representation used in the previous talk



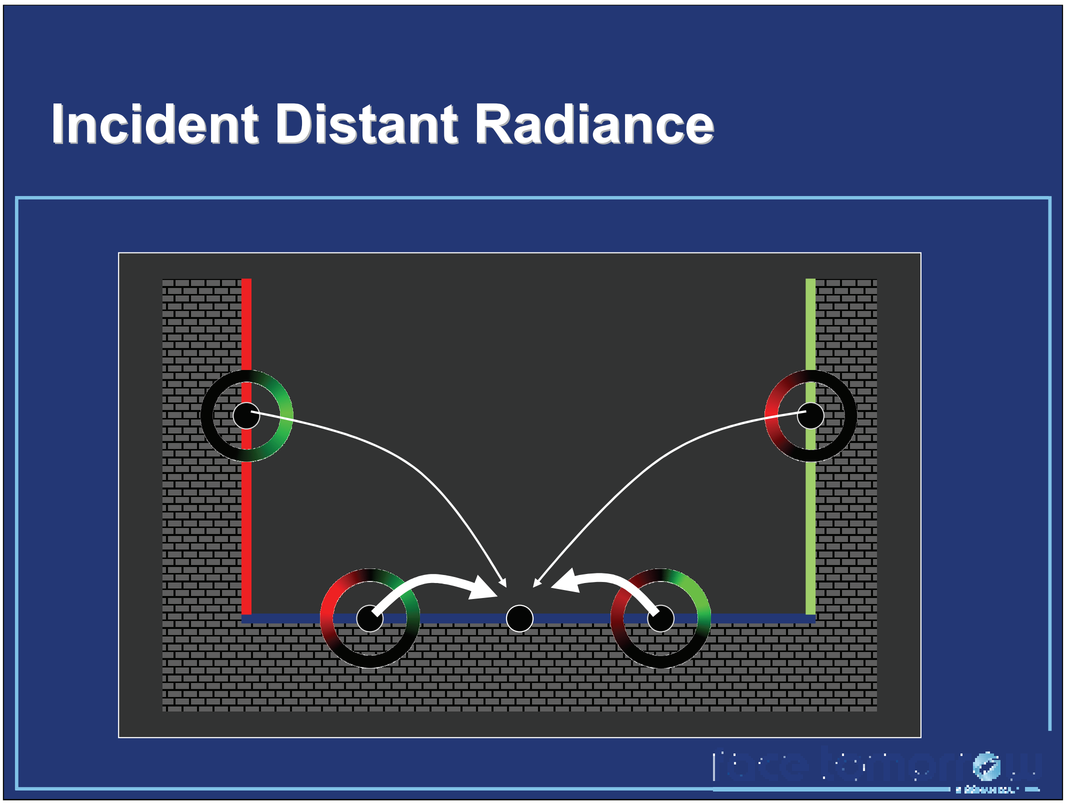
Incident Distant Radiance

-SH samples representing the incident radiance are computed at sparse set of points

-To compute these samples, we sample the incident radiance by shooting rays

-Wherever these rays hit, we lookup the proxy global illumination (such as a photon map)

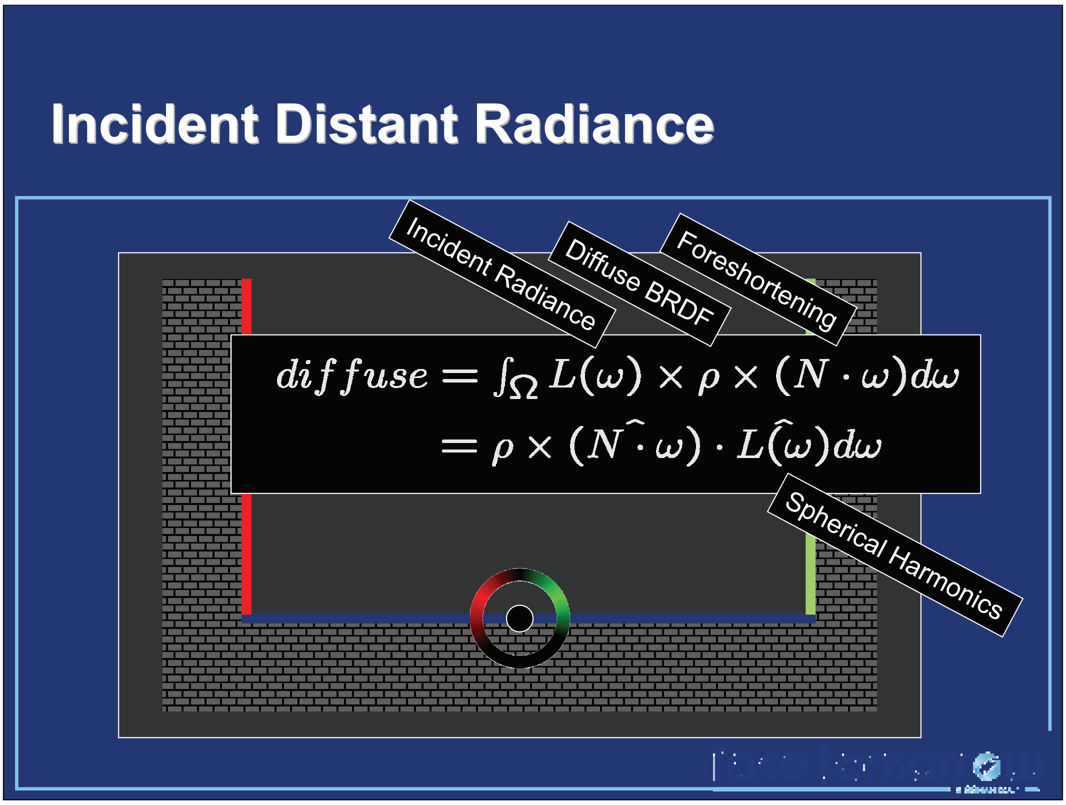
-We then find the least squares spherical harmonic representation for these rays (the incident radiance)



Incident Distant Radiance

•Given a query location, we interpolate these samples (similar to irradiance caching) to obtain an approximation to the incident radiance

•This interpolation is very similar to the one presented in the irradiance caching section



Incident Distant Radiance

$$diffuse = \int_\Omega L(\omega) \times \rho \times (N \cdot \omega) d\omega$$
$$= \rho \times (N \hat{\cdot} \omega) \cdot L(\hat{\omega}) d\omega$$

-Because of orthanormality of SH, the distant component becomes a dot product (easy)

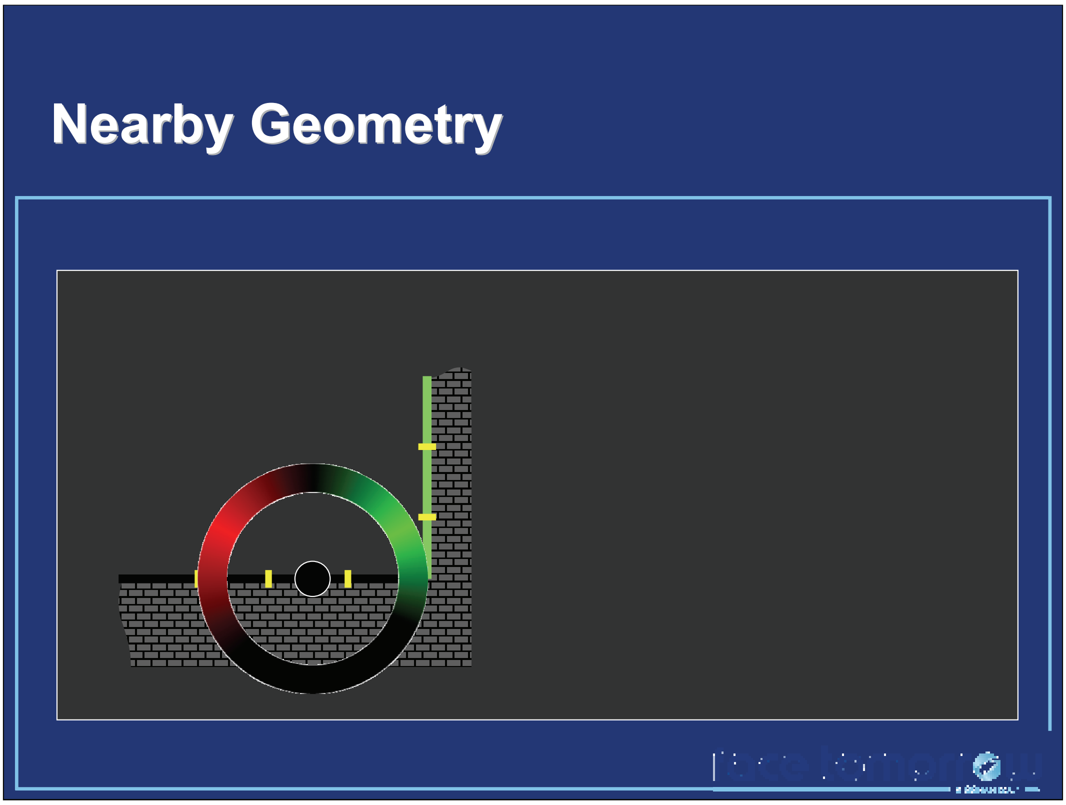
-In particular, the diffuse illumination at this point is the integral of the incident radiance times the diffuse BRDF times a foreshortening term

-Spherical harmonics is a frequency space representation that shares the same features as Fourier representation for functions

-Therefore we can write this convolution as a dot product in the spherical harmonic represention

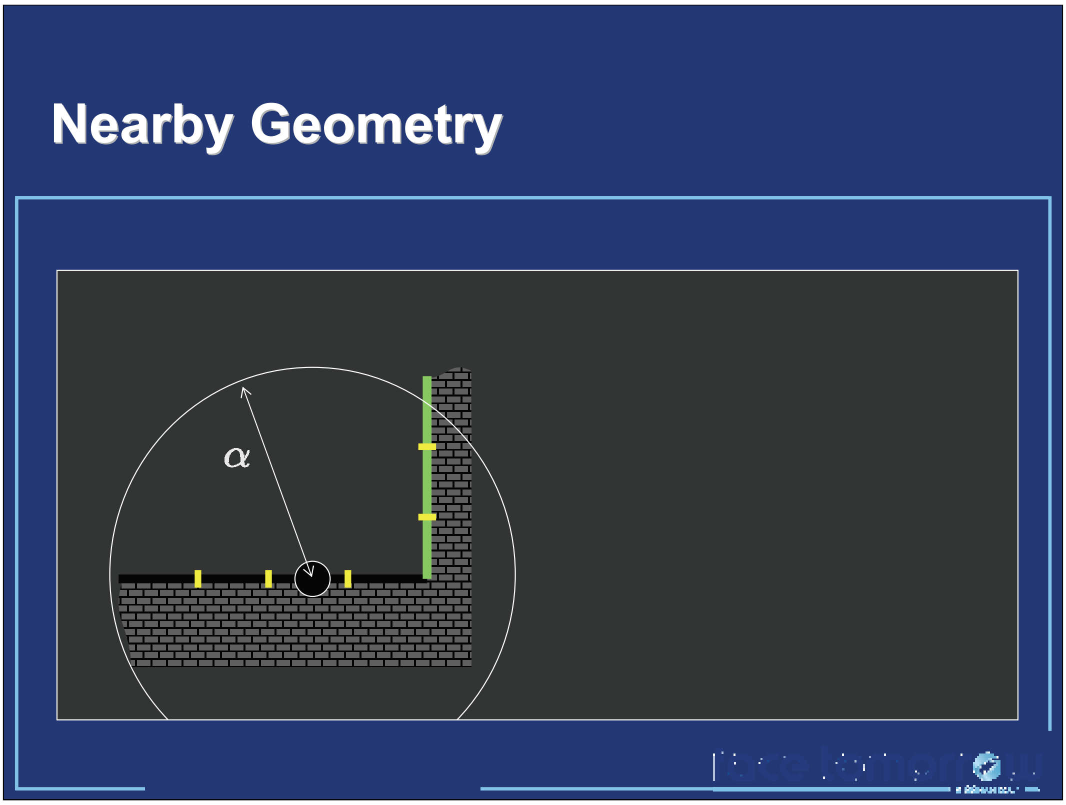
-We already have the SH representation for the incident radiance

-There are analytical forms for the SH representation of the foreshortening term (see [Ramamoorthi and Hanrahan 2001])



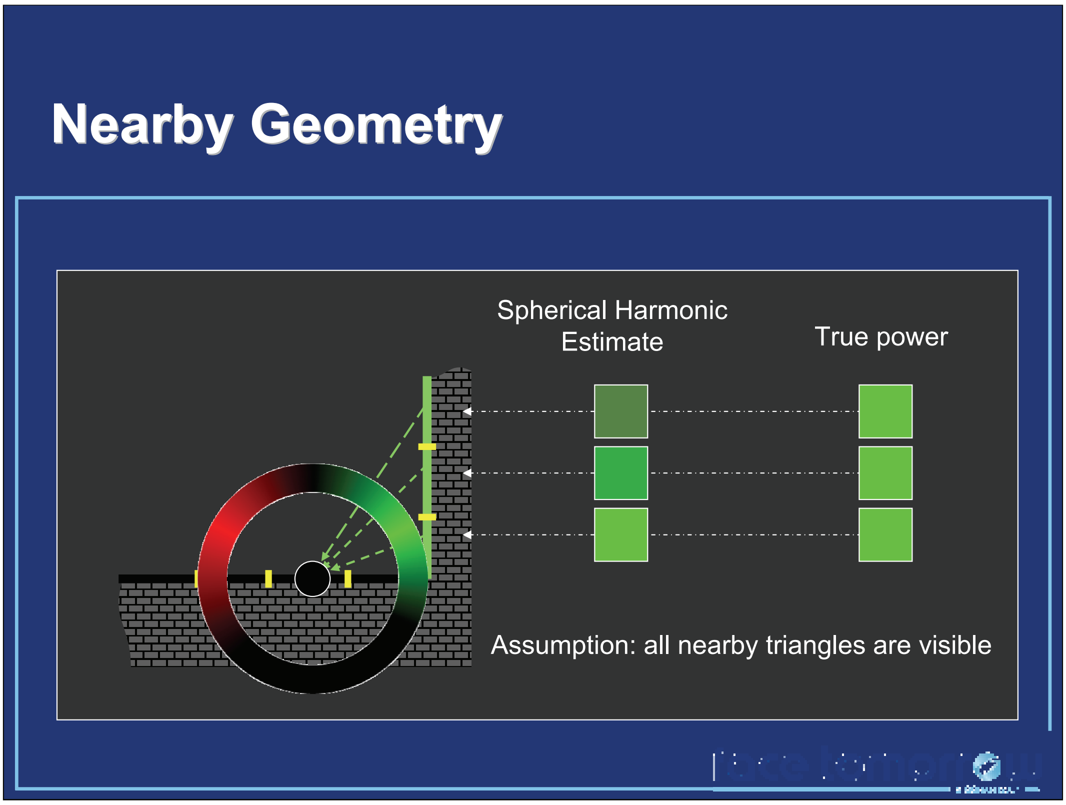
Nearby Geometry

- We should now account for the nearby geometry



Nearby Geometry

- To do this, for each query point, we iterate over the nearby scene triangles



Nearby Geometry

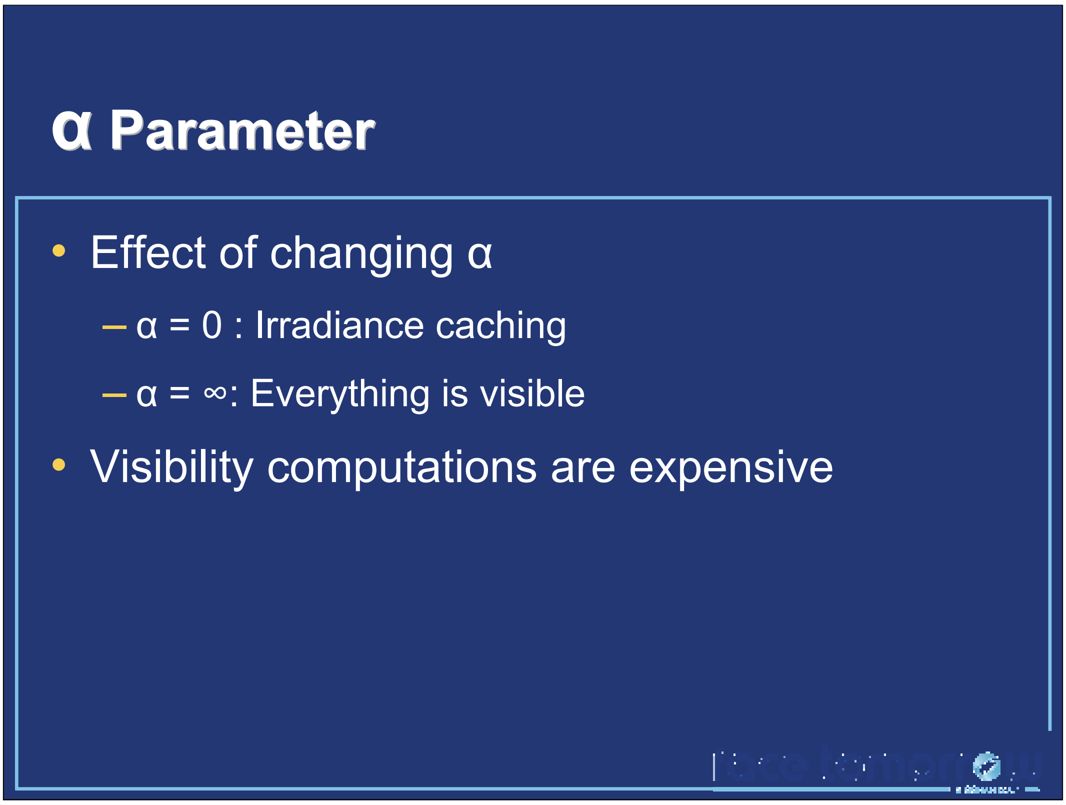
Assumption: all nearby triangles are visible

-For each nearby triangle,

- -Subtract the energy from the SH representation that the triangle covers
- -We do this by Monte Carlo Integration. In particular since we only use low order harmonics, the SH function we're sampling is smooth and MC integration is efficient
- -Add the energy from the triangle to the point
- -We do this by computing the form factor of the triangle and looking up the radiosity of the triangle by looking into our proxy global illumination representation (photon map)

-If we assume all nearby scene triangles are visible (barring those that are backfacing to the query point), this can be computed efficiently using analytical triangle to point form factors.

-This step restores the high frequency content of the global illumination due to geometric detail.



α Parameter

- Effect of changing α
  - α = 0 : Irradiance caching
  - α = ∞: Everything is visible
- Visibility computations are expensive

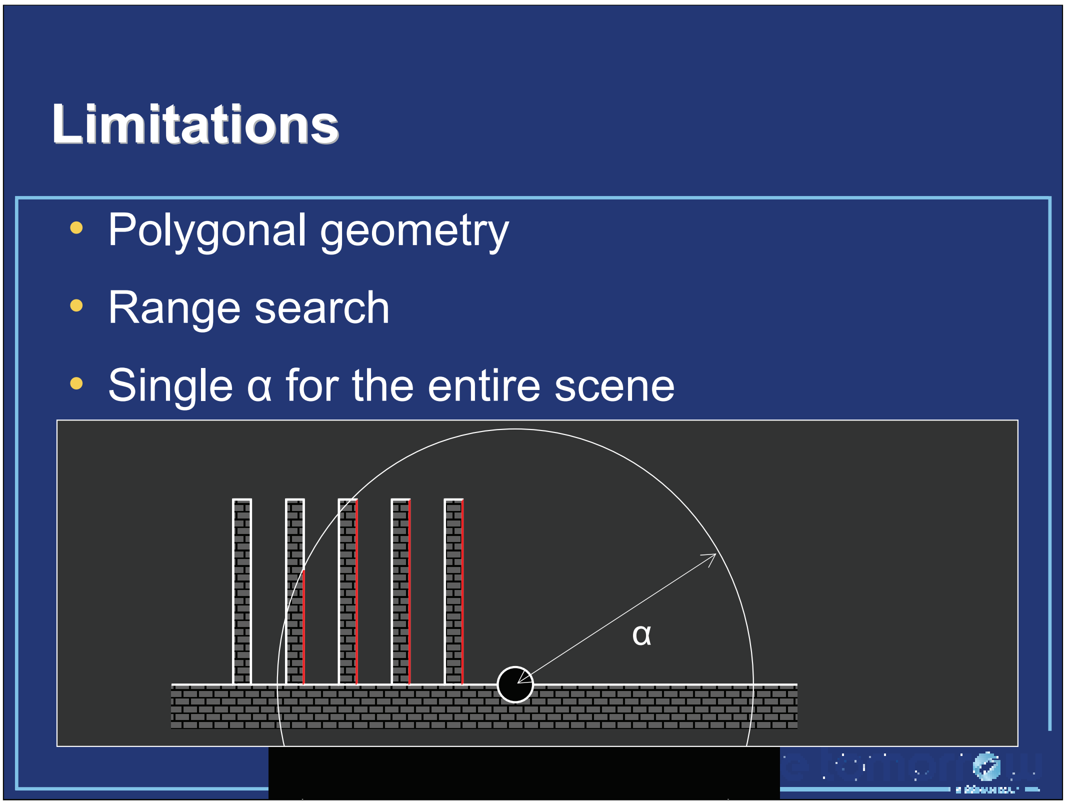
-Crucial parameter is alpha (the nearby-distant threshold)

-When alpha is zero, the method is similar to irradiance caching. We will collect denser samples of incident radiance around corners or high geometric detail.

-When alpha is large, the method will assume everything is visible and the results will be inaccurate.

-Alpha allows us to ignore expensive visibility computations between nearby surfaces.

-This is where we make our money, nearby surfaces are usually visible to each other. By ignoring these visibility computations, we obtain good speedup without creating too much error.

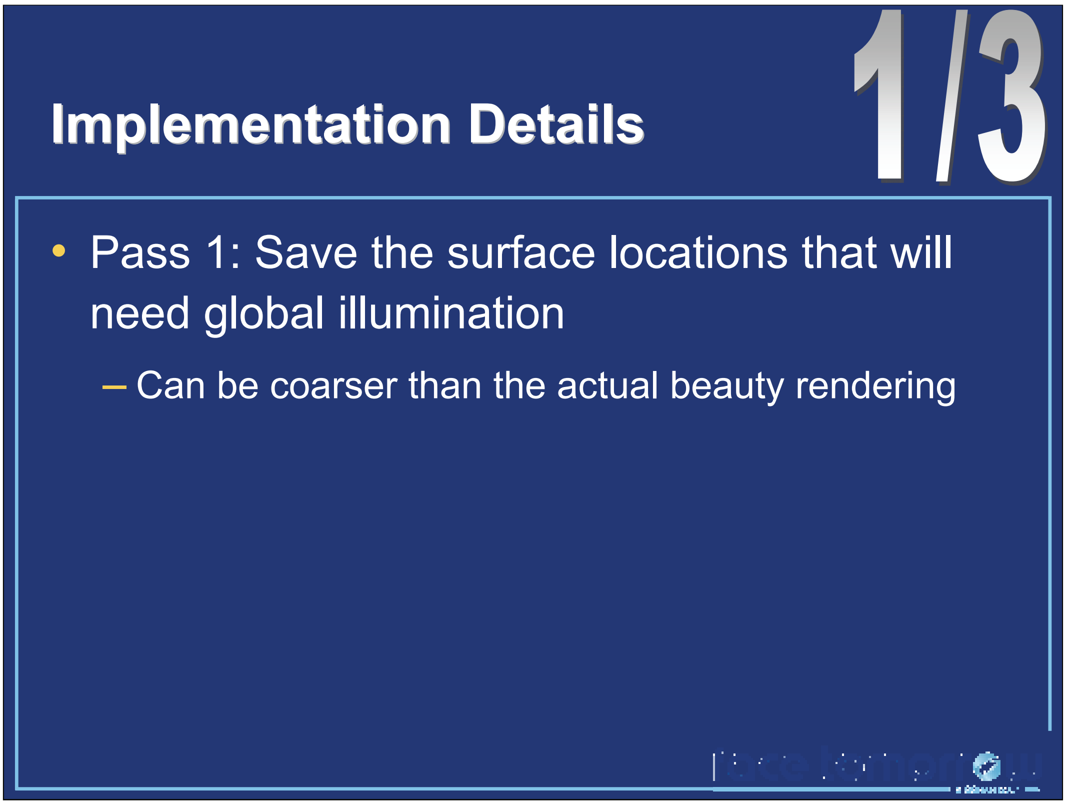


Limitations

- Polygonal geometry
- Range search
- Single α for the entire scene

•This method has several limitations:

- •We assume the scene is polygonal so that we can compute analytical form factors for nearby geometry
- •For each query point, we need to locate nearby triangles which involves a range search similar to the one in photon mapping
- •Having a single alpha value for the entire scene may not be too optimal
- •Also, for scenes such as this one, our assumption about the nearby surfaces being visible may fail.



Implementation Details

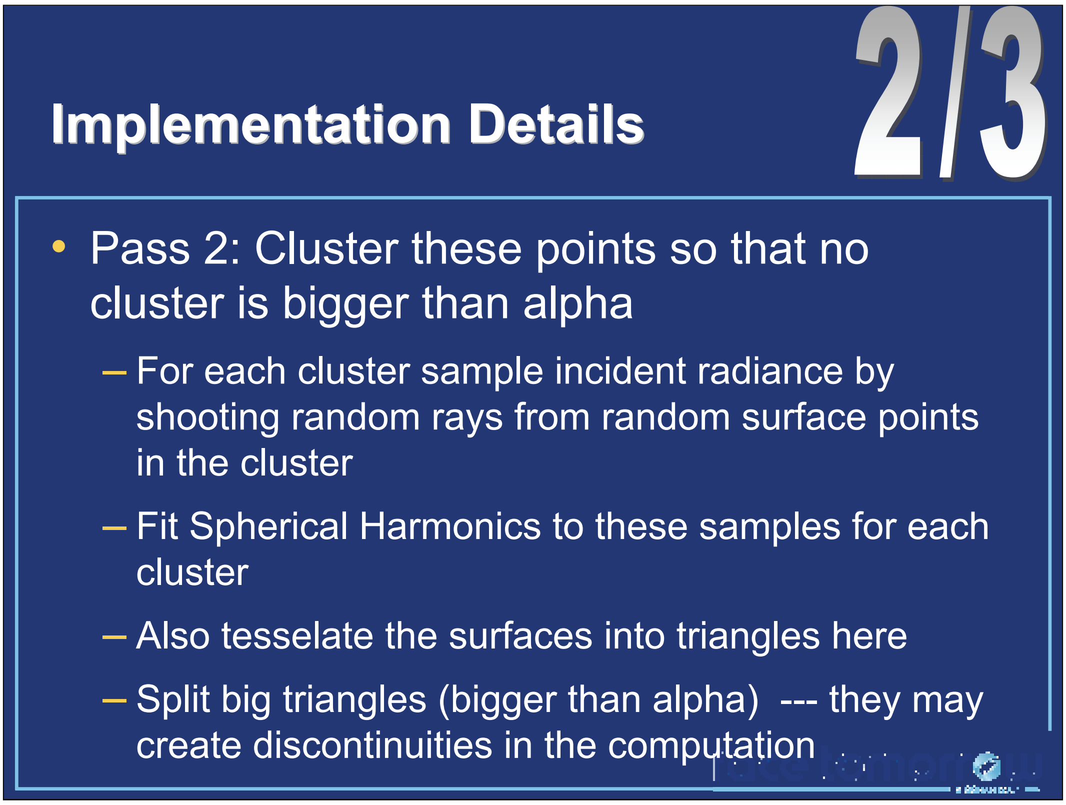
1/3

- Pass 1: Save the surface locations that will need global illumination
  - Can be coarser than the actual beauty rendering

•This is a 3 pass algorithm

•In the first pass, we record the surface locations that will need the indirect illumination computation

•Therefore this is a lazy algorithm, we do not need to sample all surfaces, only the visible ones



Implementation Details

2/3

- Pass 2: Cluster these points so that no cluster is bigger than alpha
  - For each cluster sample incident radiance by shooting random rays from random surface points in the cluster
  - Fit Spherical Harmonics to these samples for each cluster
  - Also tesselate the surfaces into triangles here
  - Split big triangles (bigger than alpha) --- they may create discontinuities in the computation

•The second pass involves hierarchical vector quantization to cluster these surface locations so that each cluster is no bigger than alpha

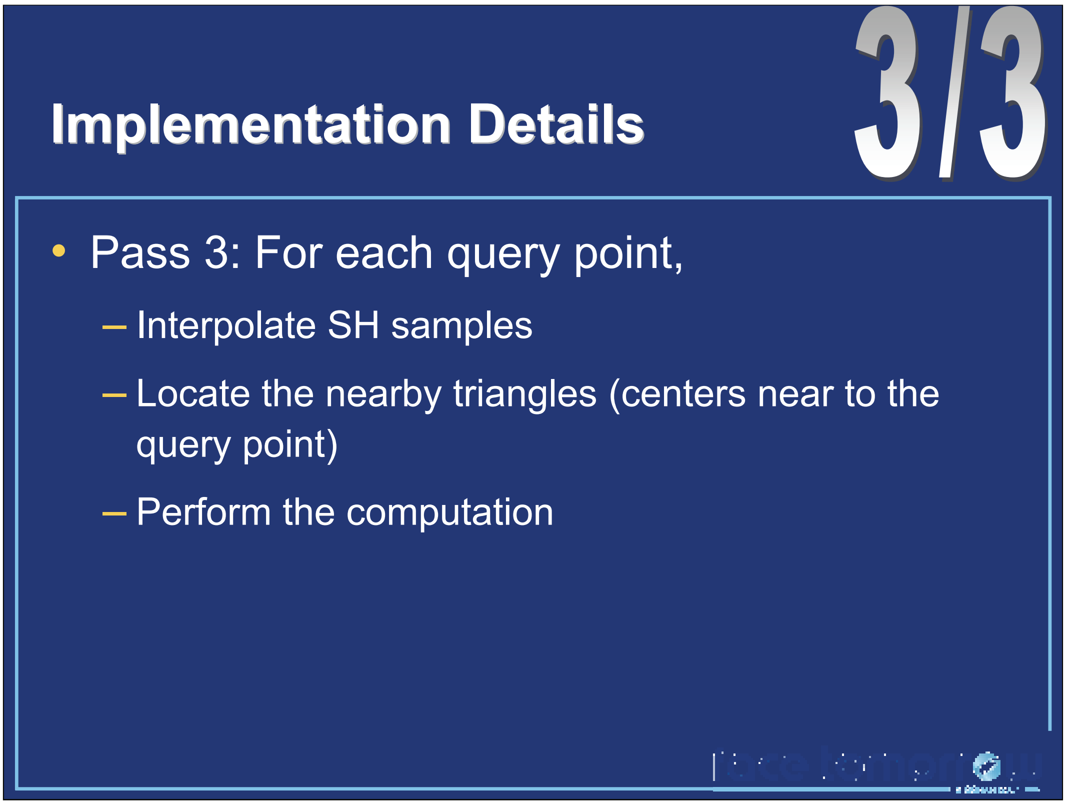
•For each cluster we can now go ahead and sample the spherical harmonics

•We do that by shooting random rays from random surface locations within each cluster

•In this step, we also convert the scene geometry into triangles

•We must avoid big triangles because they may create discontinuities (imagine a big triangle which was not near suddenly becoming near)

•We split these triangles so that they are no bigger than alpha in scale



Implementation Details

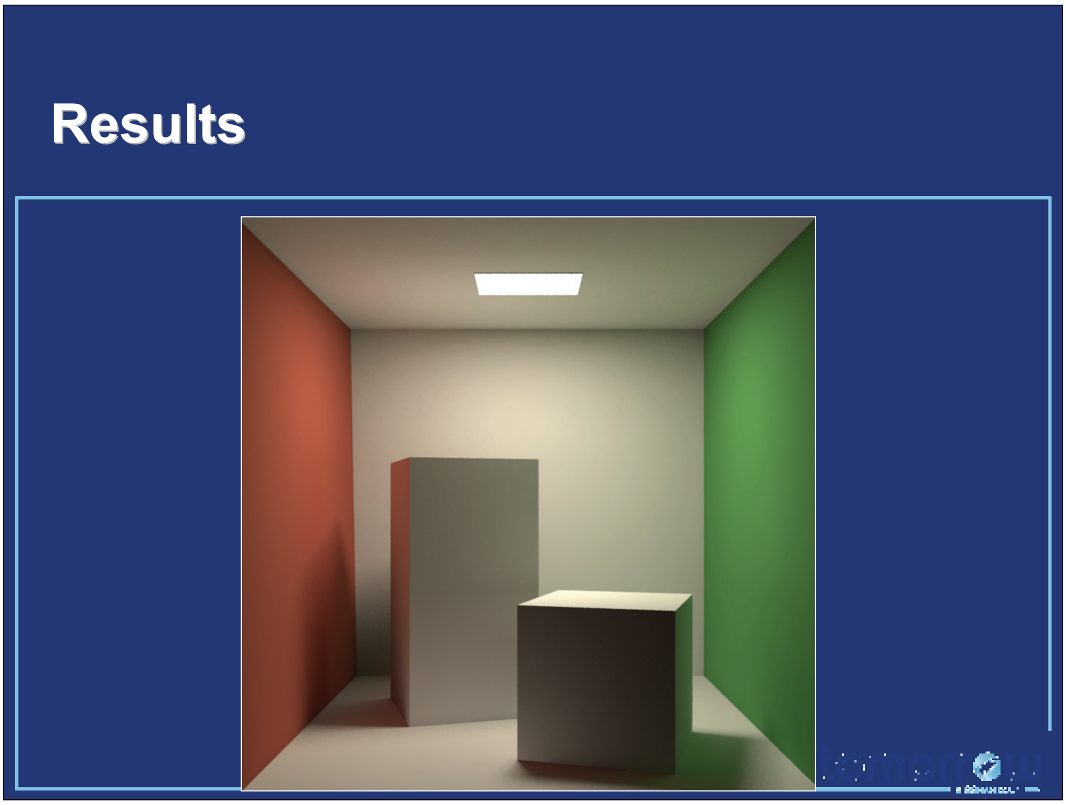
3/3

- Pass 3: For each query point,
  - Interpolate SH samples
  - Locate the nearby triangles (centers near to the query point)
  - Perform the computation

•Third pass is the beauty pass

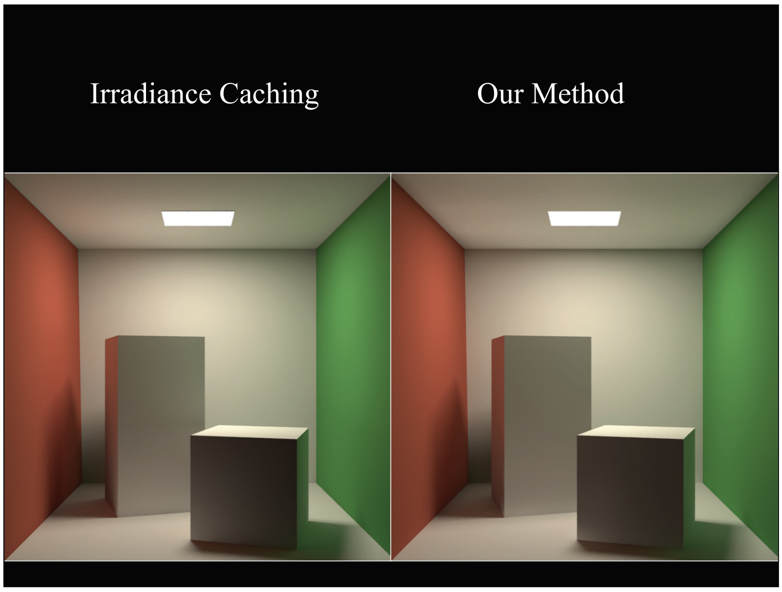
•Here for every query point, we interpolate the SH samples and locate nearby triangles (whose centers are near to the query point)

•We then compute the indirect illumination



Results

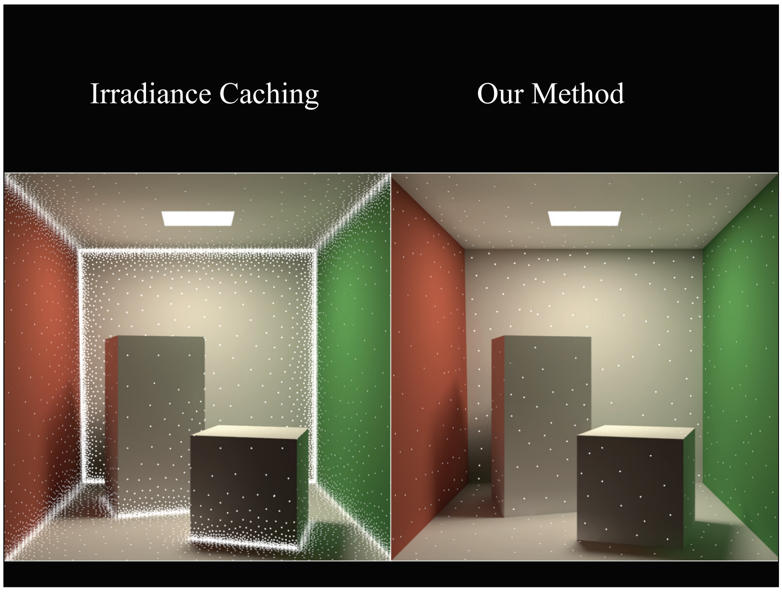
•Let’s move onto the results and talk about what we can do with this method.



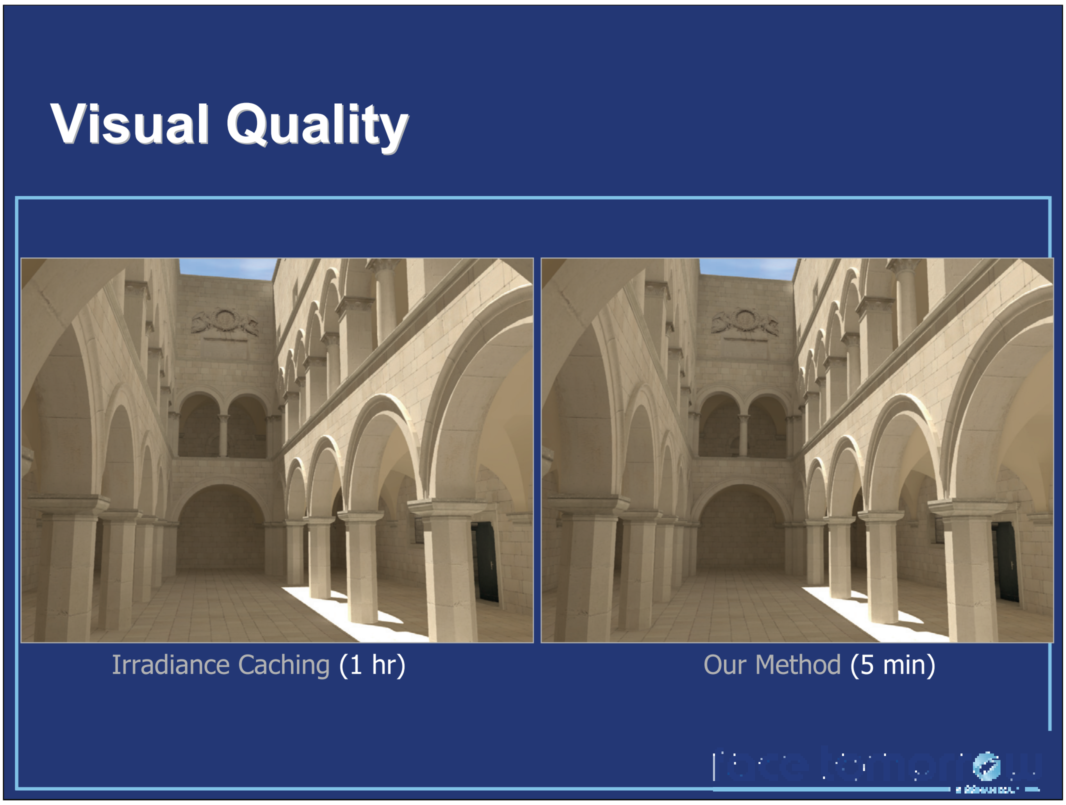
Irradiance Caching

Our Method

•Here’s a comparison with irradiance caching.



- Here’re the corresponding sample locations.
- The cost of computing one of these samples for irradiance caching and our method is the same.
- (the cost of the dots is the same)

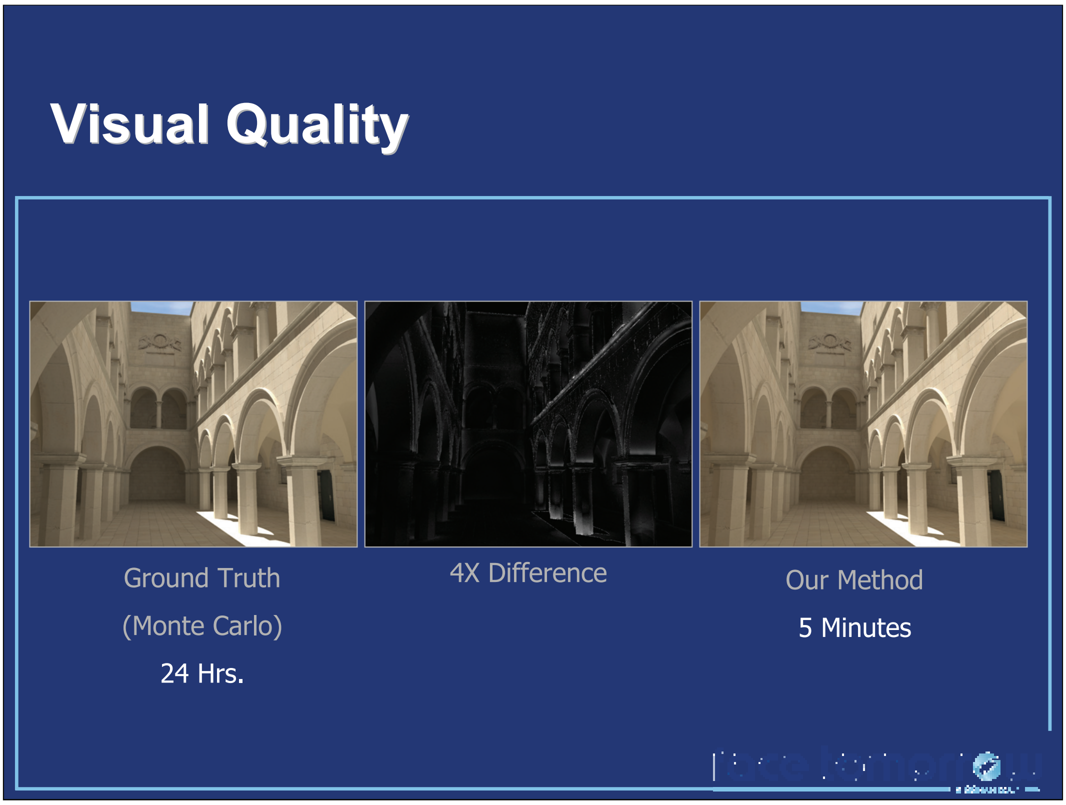


Visual Quality

Irradiance Caching (1 hr)

Our Method (5 min)

•Here’s a comparison on a more complicated scene.



Visual Quality

Ground Truth

(Monte Carlo)

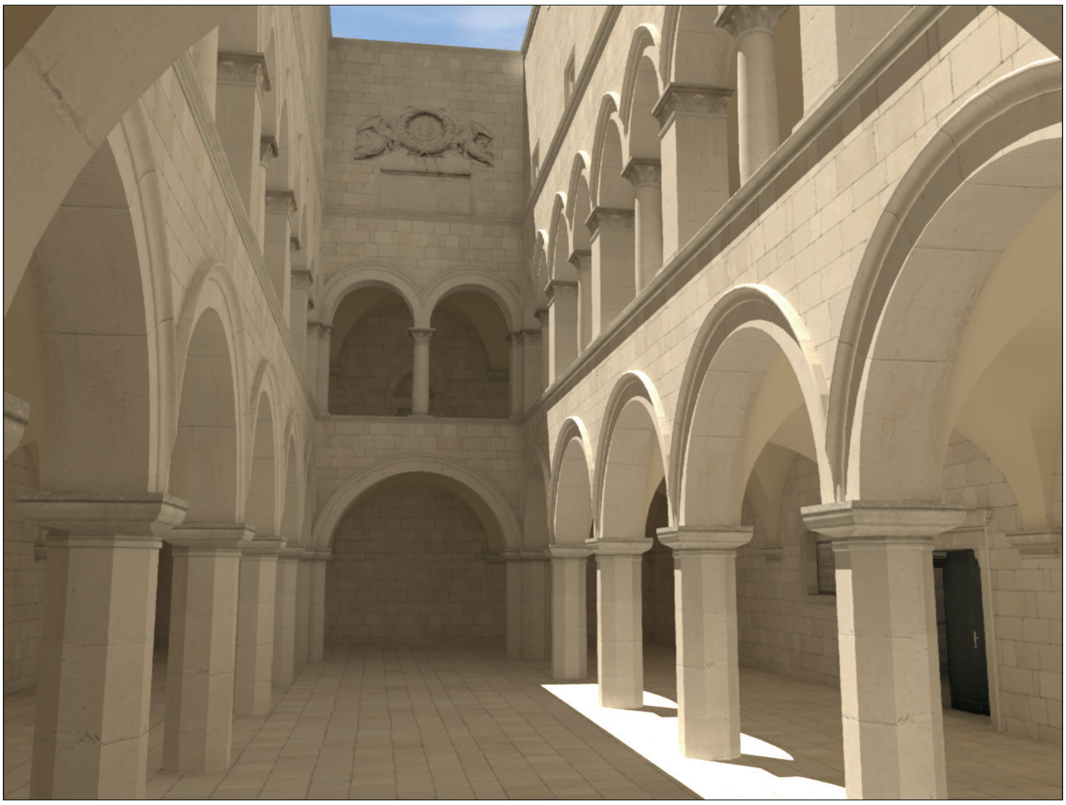
24 Hrs.

4X Difference

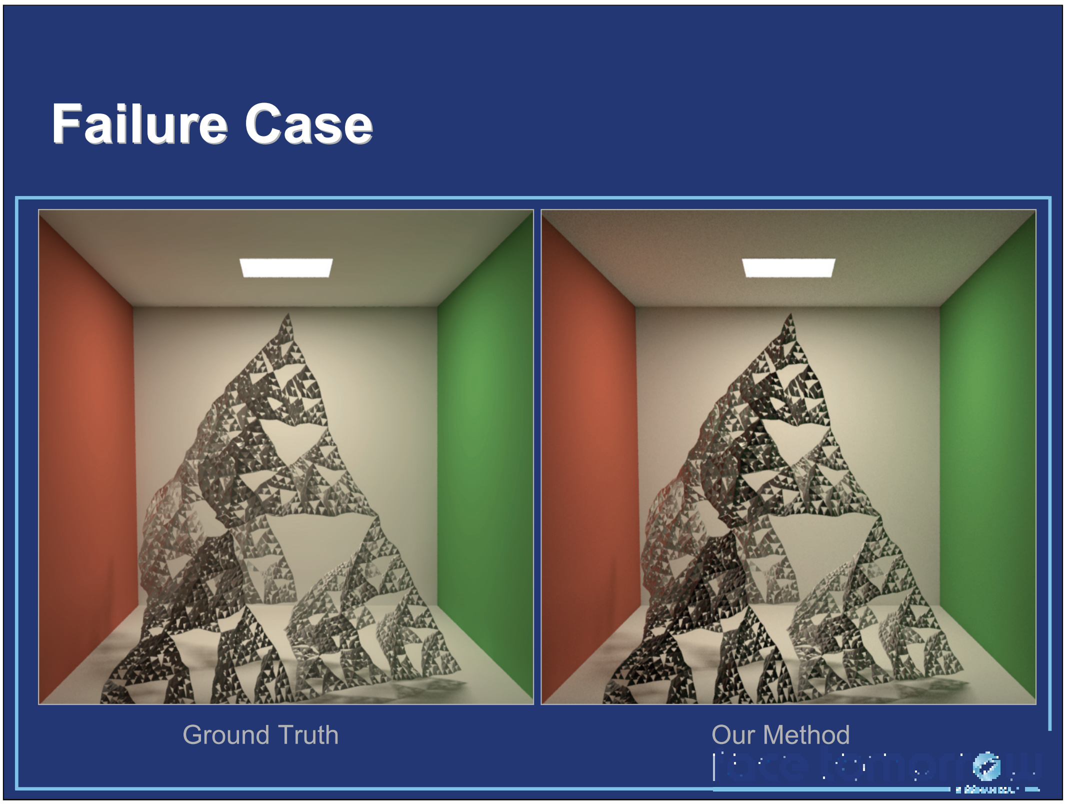
Our Method

5 Minutes

- The middle image shows the 4 times magnified difference between the ground truth and this method.



- Here’s a bigger version of the atrium scene.



Failure Case

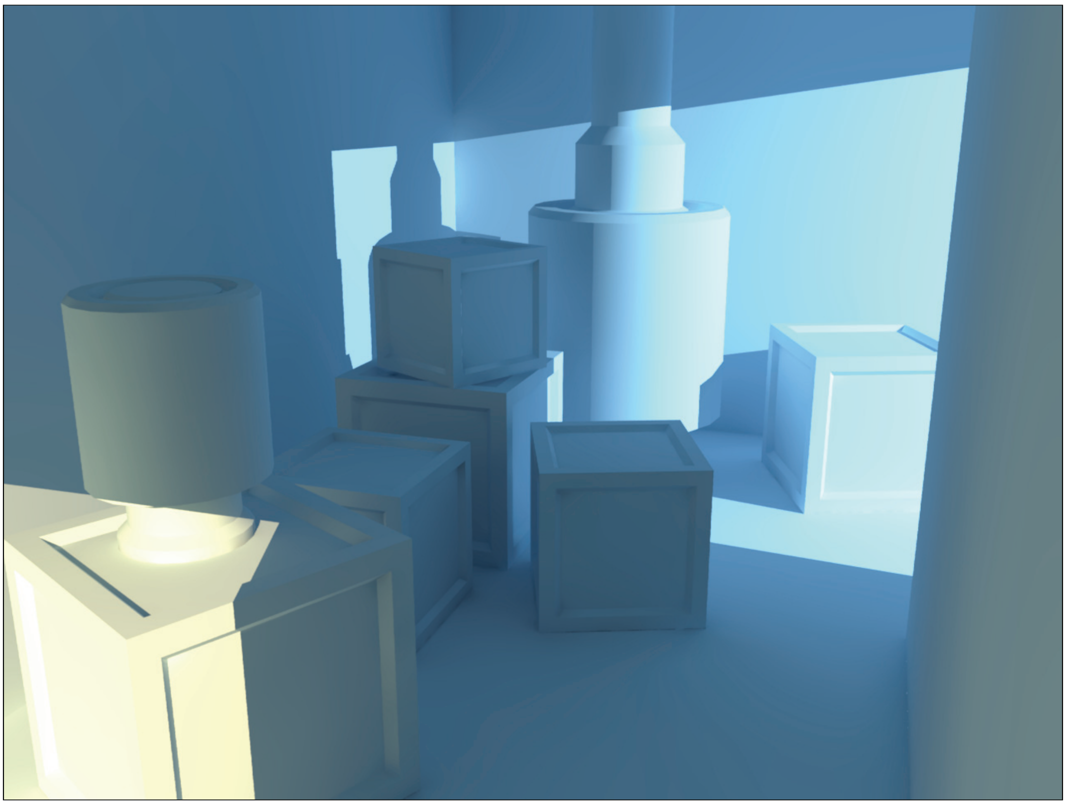
Ground Truth

Our Method

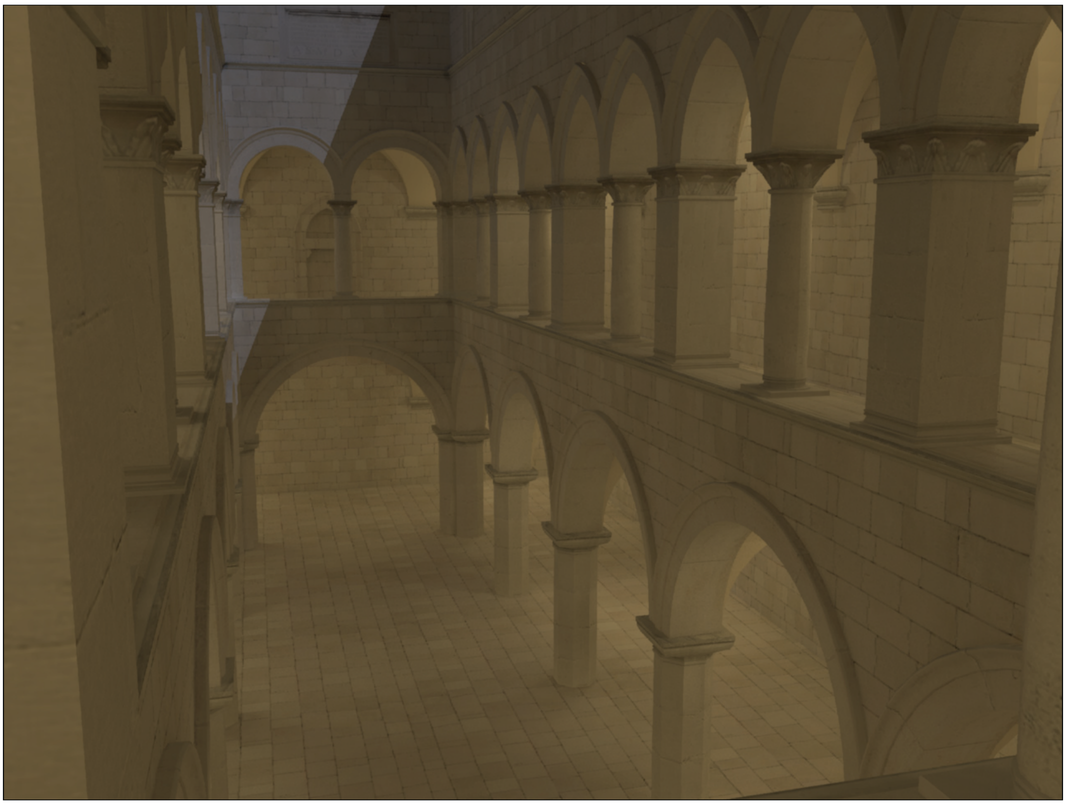
•Here’s a failure case.

•Notice that our method is generating a darker appearance because the visibility assumption is being violated in this colvoluted example.

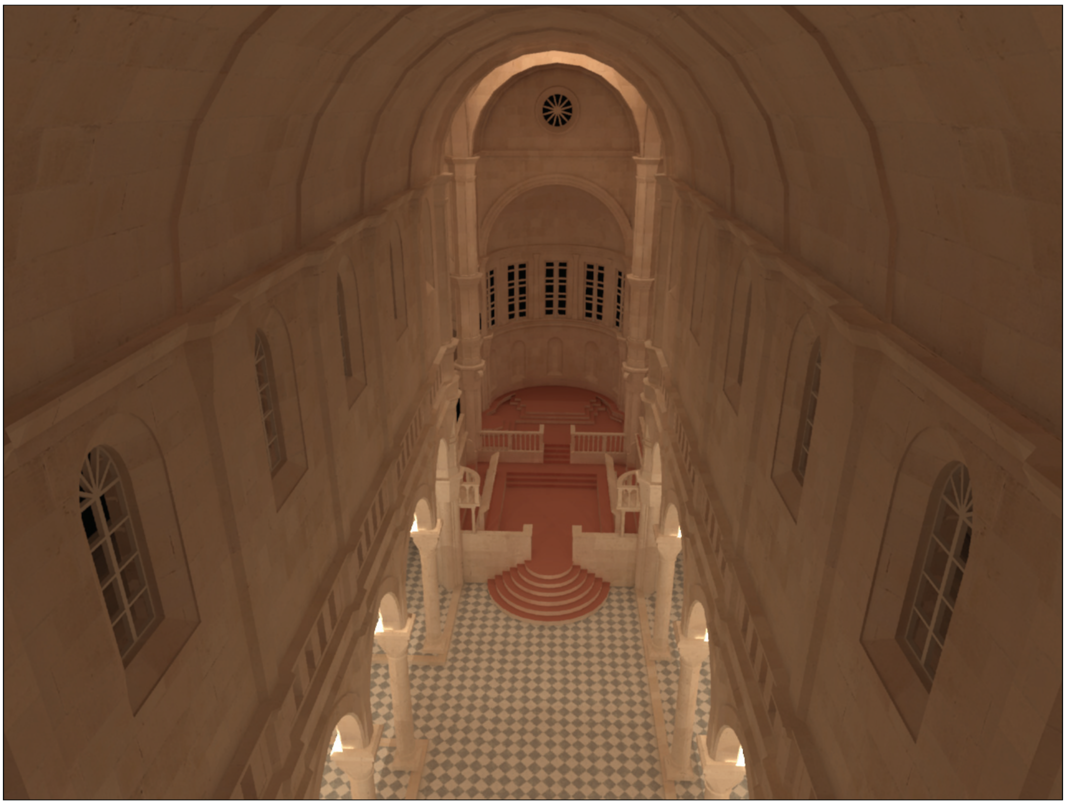
•Fortunately, such examples are rare.



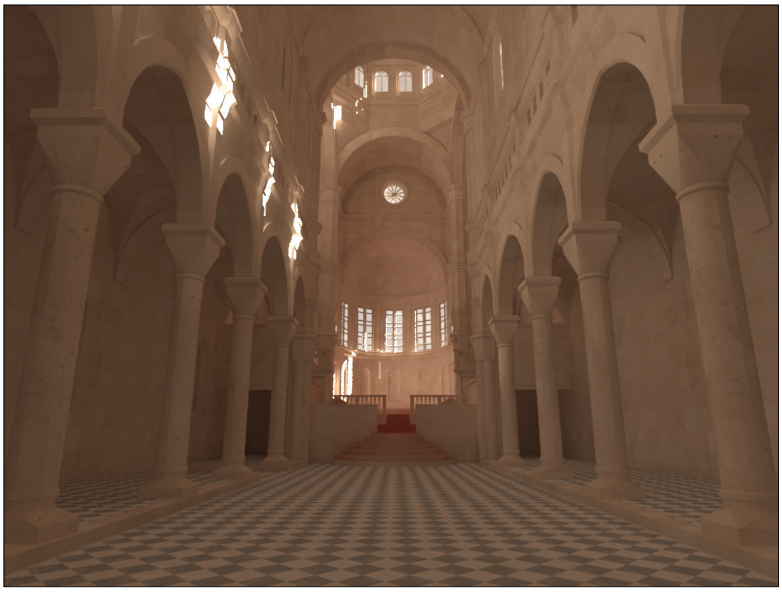
•Let’s look at some more examples.



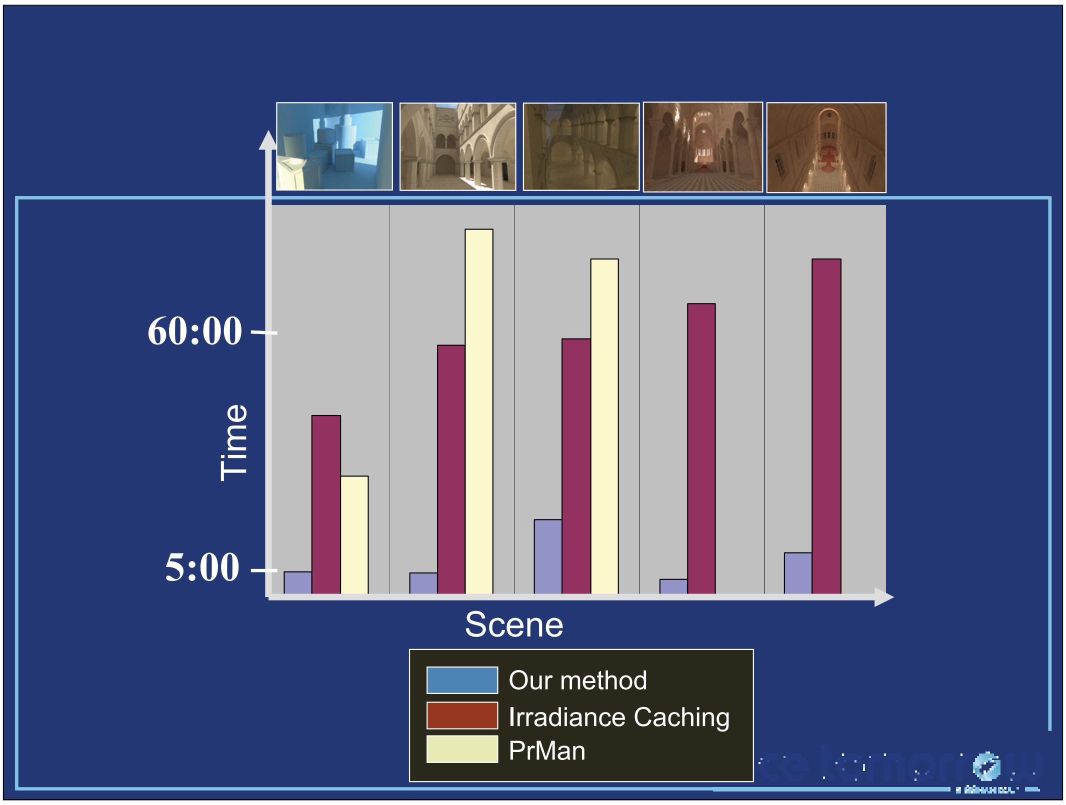
•A night time scene.



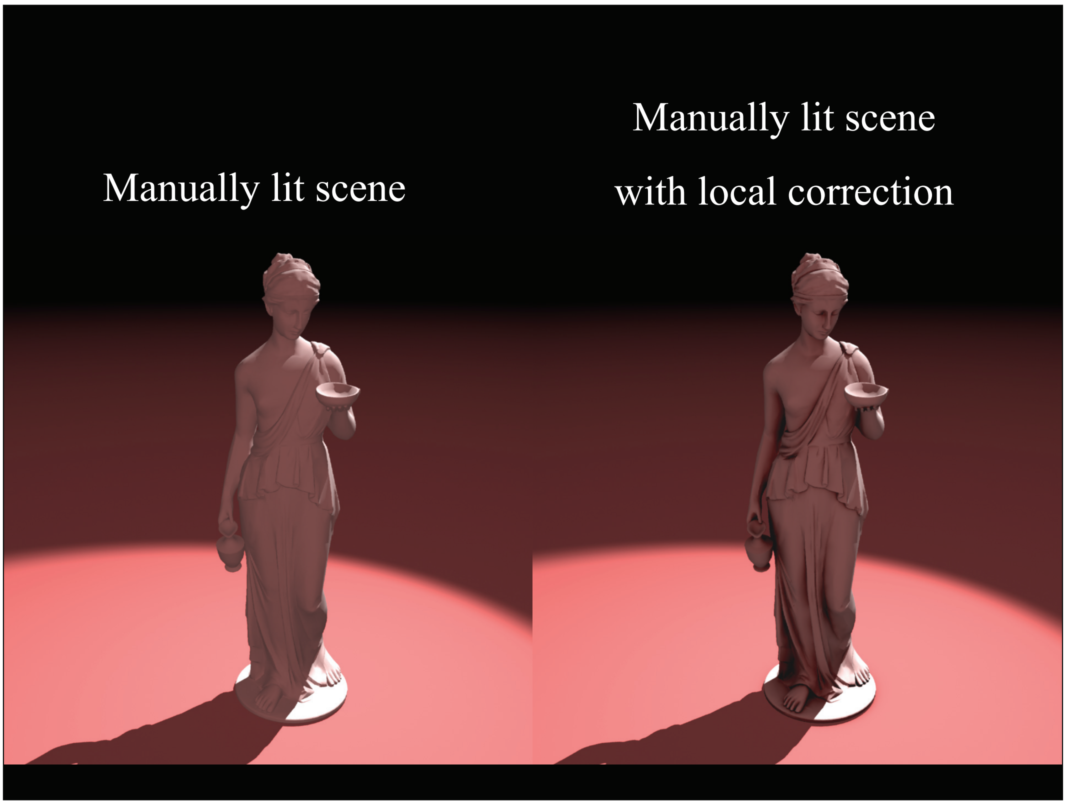
- A cathedral during night.



- The cathedral during the day.



- Here’s a timing comparison



Manually lit scene

Manually lit scene with local correction

- By using only the nearby scene triangles and subtracting the energy that they cover, we can simulate ambient occlusion as well.
- The image on the right is generated without firing a single ray.

PIXIE

Source Code:

http://pixie.sourceforge.net

Thanks to:

- PIXAR
- Wayne Wooten

A Ray Tracing Solution for Diffuse Interreflection

Gregory J. Ward
Francis M. Rubinstein
Robert D. Clear

Lighting Systems Research
Lawrence Berkeley Laboratory
1 Cyclotron Rd., 90-3111
Berkeley, CA 94720
(415) 486-4757

Abstract

An efficient ray tracing method is presented for calculating interreflections between surfaces with both diffuse and specular components. A Monte Carlo technique computes the indirect contributions to illuminance at locations chosen by the rendering process. The indirect illuminance values are averaged over surfaces and used in place of a constant "ambient" term. Illuminance calculations are made only for those areas participating in the selected view, and the results are stored so that subsequent views can reuse common values. The density of the calculation is adjusted to maintain a constant accuracy, permitting less populated portions of the scene to be computed quickly. Successive reflections use proportionally fewer samples, which speeds the process and provides a natural limit to recursion. The technique can also model diffuse transmission and illumination from large area sources, such as the sky.

General Terms: Algorithm, complexity.

Additional Keywords and Phrases: Caching, diffuse, illuminance, interreflection, luminance, Monte Carlo technique, radiosity, ray tracing, rendering, specular.

1. Introduction

The realistic computer rendering of a geometric model requires the faithful simulation of light exchange between surfaces. Ray tracing is a simple and elegant approach that has produced some of the most realistic images to date. The standard ray tracing method follows light backwards from the viewpoint to model reflection and refraction from specular surfaces, as well as direct diffuse illumination and shadows [15]. Accuracy has been improved with better reflection models [4] and stochastic sampling techniques [6]. Unfortunately, the treatment of diffuse interreflection in conventional ray tracers has been limited to a constant "ambient" term. This approximation fails to produce detail in shadows, and precludes the use of ray tracing where indirect lighting is important.

We present a method for modeling indirect contributions to illumination using ray tracing. A diffuse interreflection calculation replaces the ambient term directly, without affecting the formulas or algorithms used for direct and specular components. Efficiency is obtained with an appropriate mix of view-dependent and view-independent techniques.

©1988 ACM-0-89791-275-6/88/008/0085 $00.75

2. Interreflection in Ray Tracing

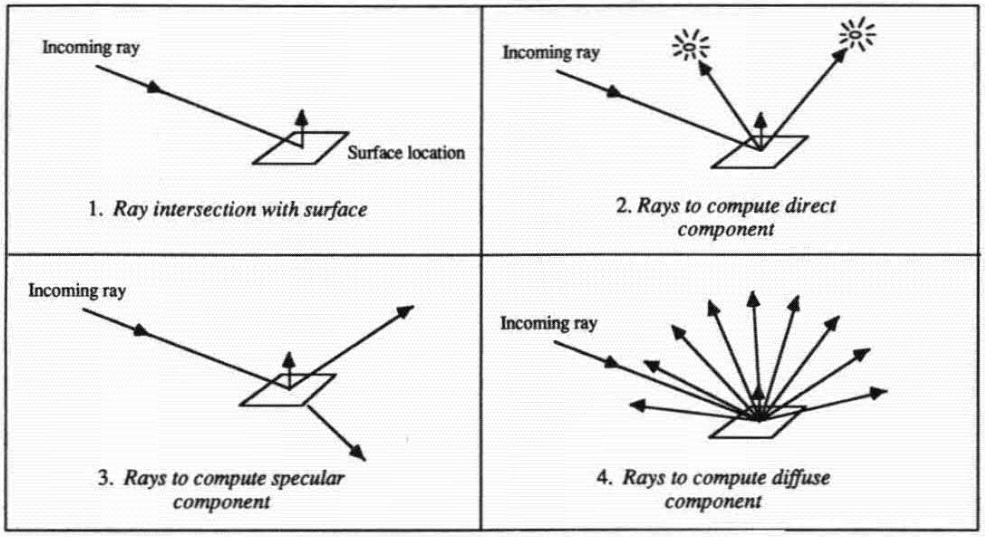
Ray tracing computes multiple reflections by recursion (Figure 1). At each level, the calculation proceeds as follows:

1. Intersect the ray with scene geometry.
2. Compute direct contributions from light sources.
3. Compute specular contributions from reflecting surfaces.
4. Compute diffuse contributions from reflecting surfaces.

The complexity of the calculation is closely related to the difficulty of step 1, and the number of times it is executed as determined by the propagation (recursion) of steps 2 through 4. Step 2 requires as many new rays as there are light sources, but the rays do not propagate so there is no growth in the calculation. Step 3 can result in a few propagating rays that lead to geometric growth if unchecked. Methods for efficient specular component computation have been described by [8], [5] and [14]. The diffuse contributions in step 4, however, require many (>100) propagating rays that quickly overwhelm a conventional calculation. Most methods simply avoid this step by substituting a constant ambient term. Our goal is to find an efficient method for computing diffuse interreflection and thereby complete the ray tracing solution. We start with a summary of previous work in this area.

An advanced ray tracing method developed by Kajiya follows a fixed number of paths to approximate global illumination at each pixel [8]. Using hierarchical "importance" sampling to reduce variance, the illumination integral is computed with fewer rays than a naive calculation would require. This brings ray tracing closer to a full solution without compromising its basic properties: separate geometric and lighting models, view-dependence for efficient rendering of specular effects, and pixel-independence for parallel implementations. Unfortunately, the method is not well suited to calculating diffuse interreflection, which still requires hundreds of samples. A high-resolution image simply has too many pixels to compute global illumination separately at each one.

The radiosity method, based on radiative heat transfer, is well suited to calculating diffuse interreflection [12][10][2]. Surfaces are discretized into patches of roughly uniform size, and the energy exchange between patches is computed in a completely view-independent manner. The method makes efficient use of visibility information to compute multiple reflections, and sample points are spaced so that there is sufficient resolution without making the calculation intractable. In areas where illumination changes rapidly, the patches can be adaptively subdivided to maintain accuracy [3]. However, the standard radiosity method models only diffuse surfaces, which limits the realism of its renderings. Immel extended the approach to include non-diffuse environments, adding bidirectional reflectance to the energy equations [7]. Unfortunately, the view-independent solution of specular interreflection between surfaces requires sampling radiated directions over very small (approaching pixel-sized) surface patches. The resulting computation is intractable for all but the simplest scenes.



1. *Ray intersection with surface*

2. *Rays to compute direct component*

3. *Rays to compute specular component*

4. *Rays to compute diffuse component*

Figure 1: The four steps of ray tracing.

A combined ray tracing and radiosity approach was designed by Wallace to take advantage of the complementary properties of the two techniques [13]. Wallace divides energy transport into four "mechanisms:" diffuse-diffuse, specular-diffuse, diffuse-specular, and specular-specular. He then proceeds to account for most of these interactions with clever combinations of ray tracing and radiosity techniques. Unfortunately, there are really an infinite number of transport mechanisms, such as specular-specular-diffuse, which are neglected by his calculation. The generalization Wallace suggests for his approach is equivalent to *view-independent* ray tracing, which is even more expensive than general radiosity [7].

3. Diffuse Indirect Illumination

Our development of an efficient ray tracing solution to diffuse interreflection is based on the following observations:

- Because reflecting surfaces are widely distributed, the computation of diffuse indirect illumination requires many sample rays.
- The resulting "indirect illuminance" value† is view-independent by the Lambertian assumption [9].
- The indirect illuminance tends to change slowly over a surface because the direct component and its associated shadows have already been accounted for by step 2 of the ray tracing calculation.

For the sake of efficiency, indirect illuminance should not be recalculated at each pixel, but should instead be averaged over surfaces from a small set of computed values. Computing each value might require many samples, but the number of values would not depend on the number of pixels, so high resolution images could be produced efficiently. Also, since illuminance does not depend on view, the values could be reused for many images.

How can we benefit from a view-independent calculation in the inherently view-dependent world of ray tracing? We do not wish to limit or burden the geometric model with illuminance information, as required by the surface discretization of the radiosity method. By the same token, we do not wish to take view-independence too far, calculating illuminance on surfaces that play no part in the desired view. Instead we would like to take our large sample of rays only when and where it is necessary for the accurate computation of an image, storing the result in a separate data structure that puts no constraints on the surface geometry.

In our enhancement of the basic ray tracing technique, indirect illuminance values are cached in the following manner:

```
If one or more values is stored near this point
    Use stored value(s)
Else
    Compute and store new value at this point
```

The computation of a new value uses the "primary method." The technique for finding and using stored values is called the "secondary method." The primary method is invoked to calculate a new value the first time it is needed, which is when the secondary method fails to produce a usable estimate from previous calculations (Figure 2). Determining the appropriate range and presenting a surface-independent storage technique are the two main points of this paper. Before we explore these issues, we present a basic computation of indirect illuminance.

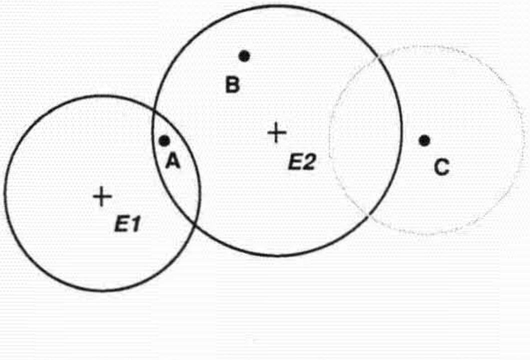


Figure 2: Illuminances *E1* and *E2* were calculated previously using the primary method. Test point **A** uses an average of *E1* and *E2*. Point **B** uses *E2*. Point **C** results in a new indirect illuminance value at that location.

†We define indirect illuminance as the light flux per unit area arriving at a surface location via non-self-luminous surfaces.

3.1. The Illuminance Integral

Illuminance is defined on a surface as the integral of luminance over the projected hemisphere [9]:

$$E = \int_0^{2\pi}\int_0^{\frac{\pi}{2}} L(\theta, \phi) \cos\theta \sin\theta \, d\theta \, d\phi \tag{1}$$

where θ = polar angle

ϕ = azimuthal angle

$L(\theta, \phi)$ = luminance from direction (θ, ϕ)

In our primary method for calculating indirect illuminance, the integral is approximated with a discrete set of sample rays that do not intersect light sources. A uniform segmented Monte Carlo distribution is derived by standard transformation methods [11]:

$$E \approx \frac{\pi}{2n^2} \sum_{j=1}^{n} \sum_{k=1}^{2n} L(\theta_j, \phi_k) \tag{2}$$

where: $\theta_j = \sin^{-1}\left(\sqrt{\frac{j - X_j}{n}}\right)$

$\phi_k = \pi \frac{(k - Y_k)}{n}$

X_j, Y_k = uniform random numbers between 0 and 1

$2n^2$ = total number of samples

In general, a better approximation to (1) may be obtained with fewer rays using hierarchical sampling techniques [8]. The particular method chosen does not affect the remainder of this discussion.

3.2. Illuminance Averaging

The secondary method performs two functions alternatively. It either approximates illuminance by averaging between primary values, or determines that a new primary value is needed. To maintain a constant accuracy with a minimum of primary evaluations, it is necessary to estimate the illuminance gradient on each surface. Where the illuminance changes slowly, as in flat open areas, fewer values are required. Where there is a large gradient, from high surface curvature or nearby objects, more frequent primary evaluations are necessary. Our method uses an estimate of the change in illuminance over a surface based on scene geometry. The inverse of this change serves as the weight for each primary value during averaging. If none of the values has a weight above a specified minimum, the primary method is invoked at that location.

We introduce a simple model to relate the illuminance gradient to scene geometry based on the assumption that narrow concentrations of luminance can be neglected. (Such localized sources should be included in the direct component calculation, since Monte Carlo sampling is a bad way to find them.) A surface element is located at the center of a sphere (Figure 3). Half of the sphere is bright, the other half is dark. The surface element faces the dividing line between the two halves. The "split sphere" has the largest gradient possible for an environment without concentrated sources.

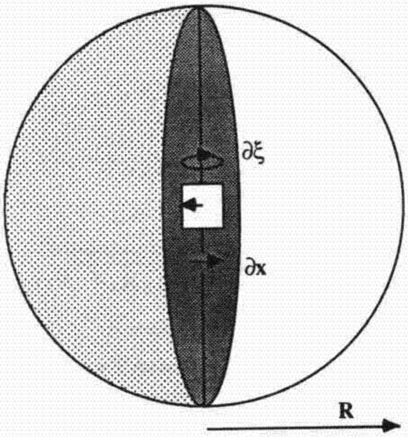


Figure 3: The split sphere model. A surface element is located at the center of a half-dark sphere.

An approximate bound to the change in illuminance in the split sphere, ϵ, is given by the first order Taylor expansion for a function of two variables:

$$\varepsilon \leq \left| \frac{\partial E}{\partial x}(x - x_o) + \frac{\partial E}{\partial \xi}(\xi - \xi_o) \right| \tag{3a}$$

Because the illuminance at the center is proportional to the projected area of the bright half of the hemisphere, the partial differentials with respect to x and ξ are proportional to the partial changes in this projection. In terms of x, the differential change over the projected area is $\frac{2R\,\partial x}{½\,\pi R^2}$, which is $\frac{4\partial x}{\pi R}$. In terms of ξ, the ratio is $\frac{½\,\pi R^2 \partial\xi}{½\,\pi R^2}$, or simply $\partial\xi$. Combining these results with the triangle inequality, we get:

$$\varepsilon \leq \frac{4}{\pi}\frac{E_o}{R}\left| x - x_o \right| + E_o \left| \xi - \xi_o \right| \tag{3b}$$

Note that the change in illuminance with respect to location is inversely proportional to the radius, R, while the change with respect to orientation does not depend on the sphere geometry. We can extend our approximation to more complicated geometries by replacing x and ξ with vector-derived values:

$$\varepsilon(\vec{P}) \leq E_o \left[\frac{4}{\pi} \frac{\| \vec{P} - \vec{P}_o \|}{R_o} + \sqrt{2 - 2\vec{N}(\vec{P})\cdot\vec{N}(\vec{P}_o)} \right] \tag{4}$$

where: $\vec{N}(\vec{P})$ = surface normal at position $\vec{P}$

$\vec{P}_o$ = surface element location

E_o = illuminance at $\vec{P}_o$

R_0 = "average" distance to surfaces at $\vec{P}_0$

The change in x becomes the distance between two points, and the change in ξ becomes the angle between two surface normals. This equation is used to estimate the relative change in illuminance for any geometry. Both the points and the surface normals are determined by the ray intersection calculation. R_0 is the harmonic mean (reciprocal mean reciprocal) of distances to visible surfaces, which can be computed from ray lengths during primary evaluation.

The inverse of the estimated error is used in a weighted average approximation of illuminance:

$$E(\vec{P}) = \frac{\sum_{i \in s} w_i(\vec{P})\, E_i}{\sum_{i \in s} w_i(\vec{P})} \qquad (5)$$

where: $w_i(\vec{P}) = \dfrac{1}{\dfrac{\| \vec{P} - \vec{P}_i \|}{R_i} + \sqrt{1 - \vec{N}(\vec{P}) \cdot \vec{N}(\vec{P}_i)}}$

E_i = computed illuminance at $\vec{P}_i$

R_i = harmonic mean distance to objects visible from $\vec{P}_i$

S = $\{ i : w_i(\vec{P}) > 1/a \}$

a = user selected constant

The approximate illuminance, $E(\vec{P})$, is given by the weighted mean of all "adjacent" illuminance values. The weight of a value is equal to the inverse of its estimated error, without the constant terms that are valid only for the split sphere ($4/\pi$ and $\sqrt{2}$). An illuminance value with an error of zero ($\vec{P}_i$ equal to $\vec{P}$) will have infinite weight. All values with an estimated error less than a will be included in the set of adjacent illuminances, S. If S is empty, a new primary illuminance value must be calculated at $\vec{P}$. (An efficient method for determining the members of S is given in the next section.)

The constant a is directly related to the maximum approximation error. When the approximation is applied to the split sphere, the error is less than $1.4a\overline{E}$, where $\overline{E}$ is a straight average of E_i over S. In general, the illuminance gradient may be larger or smaller than the split sphere, but it will always be roughly proportional to a. It is interesting to note that for a less than or equal to 1, S will not contain any value farther than the average spacing or with a surface normal more than 90 degrees from the test location. Intuitively, such a value would be expected to have 100% error.

In practice, additional tests are required to restrict the values included in S. The ray recursion depth must be considered so that values computed after one or more bounces are not substituted for final illuminances. This is easily prevented by keeping separate value lists at each recursion level. A different problem arises from our generalization of the split sphere model. Equation (4) assumes that motion in any direction is equivalent to motion in x. As a result, the set S can include illuminance values that lie on objects shadowing the test point, $\vec{P}$ (Figure 4). We therefore introduce a test to reject illuminance values that are "in front" of $\vec{P}$:

$$d_i(\vec{P}) = \left(\vec{P} - \vec{P}_i\right) \cdot \left[\vec{N}(\vec{P}) + \vec{N}(\vec{P}_i)\right] / 2 \qquad (6)$$

If $d_i(\vec{P})$ is less than zero, then $\vec{P}_i$ is in front of $\vec{P}$ so the value is excluded.

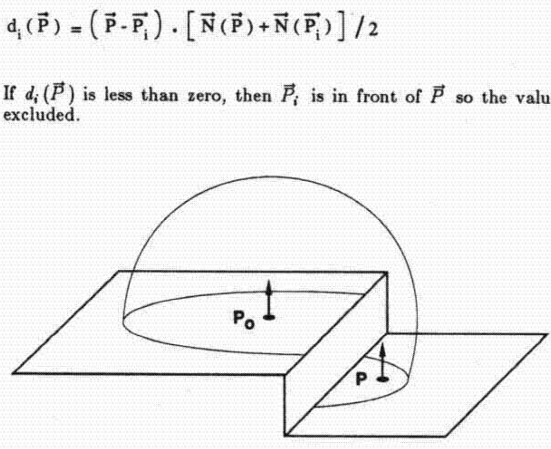


Figure 4: $\vec{P}_0$ sees few close-by surfaces, so its estimated error at $\vec{P}$ is small. But $\vec{P}$ is shadowed by the surface under $\vec{P}_0$, and the true illuminance is different.

Caching indirect illuminance is simple and efficient. The error estimate results in a minimum of primary evaluations and a nearly constant accuracy. Sections of the scene that do not contribute to the image, directly or indirectly, are not examined since no rays reach them. Areas where the indirect illuminance varies rapidly, from changing surface orientation or the influence of nearby objects, will have a higher concentration of values. Flat areas without nearby influences will have only a few values. Dynamic evaluation obviates surface discretization and presampling, so scene representation is not restricted.

Figure 5a shows three colored, textured blocks on a table illuminated by a low-angle light source. Figure 5b shows the placement of indirect illuminance values. Note that the values crowd around inside corners, where surfaces are in close visual contact, and outside corners, where the surface curvature is large. Also, the space between and immediately surrounding the blocks is more densely populated than the background, where only a few values are spread over a wide area. This distribution is different from the standard radiosity technique, which computes values at grid points on each surface. By selecting value locations based on the estimated illuminance gradient, a more accurate calculation is obtained with fewer samples.

Averaging illuminance values over surfaces results in lower pixel variance than produced by standard ray tracing techniques. Figure 5c was produced by a pure Monte Carlo computation that used as many rays as the calculation of Figure 5a. The speckling results from inadequate integration of the indirect contributions at each pixel. Since every pixel requires a separate calculation, only a few diffuse samples are possible over the hemisphere. If a sample happens to catch a bright reflection, the illuminance computed at that point will be disproportionately large. Caching permits a better integration to be performed less frequently, thereby obtaining a more realistic rendering than is feasible with pixel-independent ray tracing.

3.3. Illuminance Storage

For the secondary method to be significantly faster than the primary method, we need an efficient technique for finding the members of S (Equation 5). Without placing any restrictions on scene geometry, an octree permits efficient range searching in three dimensions [1]. A global cube is identified that encompasses all finite surfaces in the scene. When the primary method calculates a new indirect illuminance at a scene location, the global cube is subdivided as necessary to contain the value. Each illuminance, E_i, is stored in the octree node containing its position, $\vec{P}_i$, and having a size (side length) greater than twice but not more than four times the appropriate "valid domain," aR_i. This guarantees that the stored illuminance value will satisfy the condition for S in no more than eight cubes on its own octree level, and a value with a small valid domain will only be examined in close-range searches. Each node in the octree will contain a (possibly empty) list of illuminance values, and a (possibly nil) pointer to eight children. (A two-dimensional analogy is given in Figure 6.) To search the tree for values whose valid domain may contain the point $\vec{P}$, the following recursive procedure is used:

```
For each illuminance value at this node
    If $w_i(\vec{P}) > 1/a$ and $d_i(\vec{P}) \geq 0$
        Include value
For each child
    If $\vec{P}$ is within half the child's size of its cube boundary
        Search child node
```

This algorithm will not only pick up the nodes containing $\vec{P}$, but will also search nodes having boundaries within half the cube size of $\vec{P}$. In this way, all lists that might have an illuminance value whose valid domain contains $\vec{P}$ will be examined. The worst case performance of this algorithm is $O(N)$, where N is the number of values. Performance for a uniform distribution is $O(\log(N))$.

The scale of the sorting algorithm can be changed so that the octree cubes are either larger or smaller than the domains of the values they contain. If the cubes are smaller, each examined list will be more likely to contain usable values. However, many of the cubes will be empty. If the cubes are larger, more of the values will have to be examined, but less searching through the tree will be necessary. In any case, changing the scale does not affect the functioning of the algorithm, only its performance in a given situation.

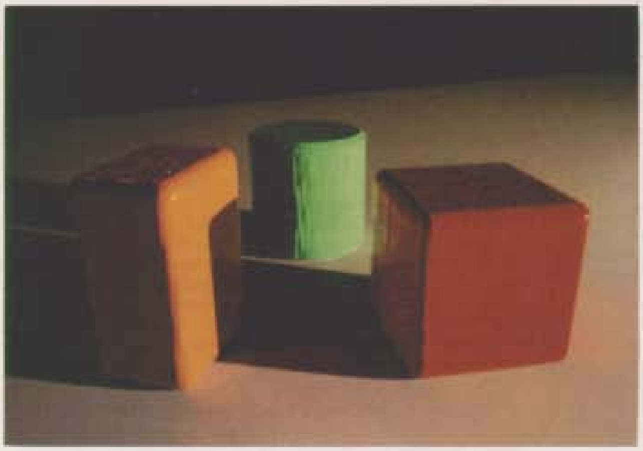


Figure 5a: Colored blocks with diffuse indirect calculation.

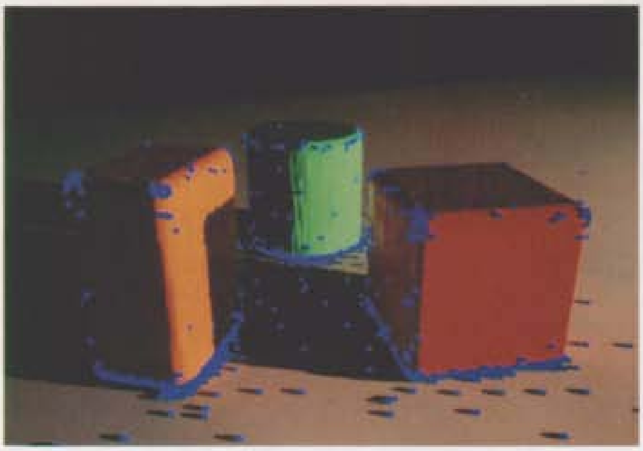


Figure 5b: Blocks with illuminance value locations in blue.

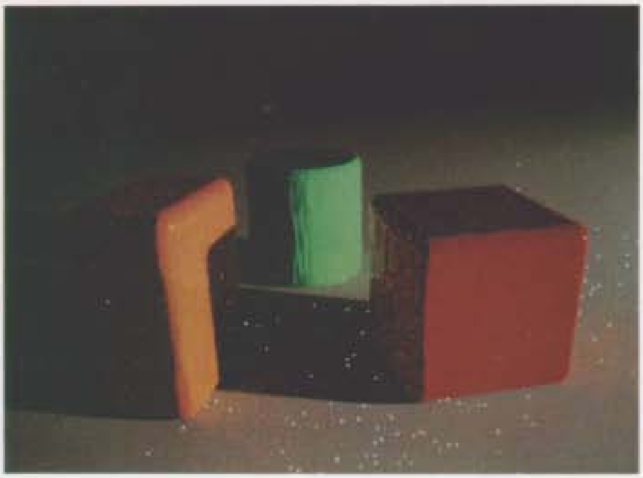


Figure 5c: Blocks using conventional ray tracing techniques.

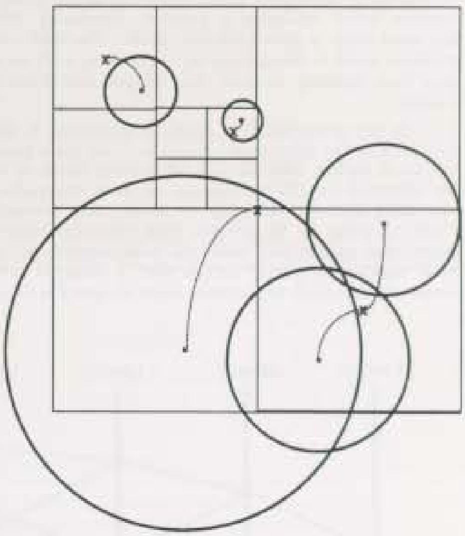


Figure 6: Five indirect illuminance values are shown with their respective domains (circles) linked by dotted lines to the appropriate nodes (squares).

Writing illuminance values to a file permits their reuse in subsequent renderings. By reusing old values, the indirect calculation will not only proceed more quickly, it will be more accurate since the illuminance is already calculated in some areas of the scene. Normally, the secondary method takes the estimated error right to its tolerance level before calculating a new value. Where the necessary values are precalculated, the tolerance is never reached because all points are within one or more valid domains.

3.4. Multiple Diffuse Reflections

It is often desirable to limit the calculation of diffuse reflections separately from the direct and specular components. A record is kept of how many diffuse bounces have occurred, and this is checked against a user-specified limit. When the limit is reached, a constant ambient value is substituted for the calculation. (This value can be zero.)

The first ray traced in a computation with multiple diffuse reflections begins a cascade of illuminance values (Figure 7). The initial primary evaluation uses many ray samples, and these rays in turn produce many more samples, with the last recursion level exhibiting the densest sampling. As the higher recursion levels become filled with primary illuminance values, fewer rays propagate in the calculation. The computation of multiple diffuse reflections therefore begins slowly, and speeds up as fewer recursion levels require primary evaluation. This process is similar to the "solution" stage of a radiosity technique, which calculates energy transfer between all surfaces before rendering is possible. Producing different views of the same scene is then relatively quick. The thrifty computation of multiple views is also present in our method, with an additional savings from ignoring surfaces that do not contribute to the desired images.

In the computation of multiple reflections, a simple optimization reduces the number of samples required for a given accuracy. If the mean surface reflectance is 50 percent, twice as much error can be tolerated in the calculation of each successive bounce. By increasing the value of a by 40 percent and decreasing the Monte Carlo sampling by 50 percent, each reflection uses one quarter as many rays as the last, with the same contribution to error. The total number of sample rays is then a bounded series, which can serve as a soft limit to recursion when accuracy is critical.

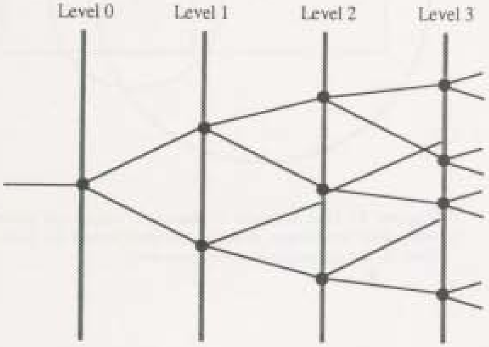


Figure 7: The lines represent rays, and the points represent primary evaluations. The rays that reuse computed values do not propagate.

4. Results

The accuracy of the secondary method was tested with an analytical solution of a sphere resting on an infinite plane with seventy percent reflectance, and a parallel light source overhead (Figure 8). The illumination on the sphere due to direct light plus first bounce was determined. Since the scene is radially symmetric, the sphere illuminance is completely described by a one dimensional function of the angle below the horizontal (γ), as measured from the center of the sphere. A closed form for illuminance was found for the upper half of the sphere, and an analytical function was integrated numerically for the lower half. The illuminance caching calculation was then applied to the problem, and the mean and maximum errors of the secondary method were found for different values of a. In each case, the distribution of error was relatively uniform over the sphere, though the density of primary evaluations varied by several orders of magnitude. The mean error was about one fourth, and the maximum error was about twice the estimated error for the split sphere. The relationship between error and a had the expected linear correlation.

Figures 9a, 9b and 9c show a daylit office space with direct only, first bounce, and seven bounces, respectively. The blind-covered window was modeled as six area sources with precalculated distributions accounting for solar and sky components. The images took 25, 40, and 70 hours in separate calculations on a VAX 11/780.

Figure 10 shows an ice cream store illuminated by indirect cove lighting. The total computation took about 30 hours on a Sun 3/60. We estimate the image would have taken more than 500 hours using pixel-independent ray tracing, or 100,000 hours for an accurate radiosity solution. Although a combined radiosity and ray tracing approach would be comparable to our method in computation time, it would not model many of the interactions shown in this image, such as the illumination under the parfait glass.

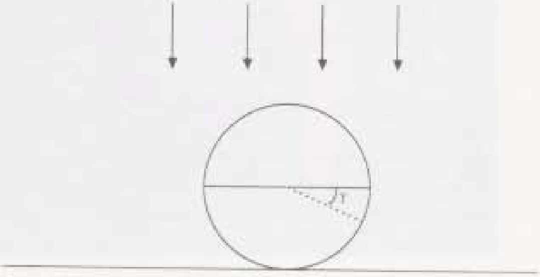


Figure 8: A sphere on an infinite plane, used to validate the secondary method.

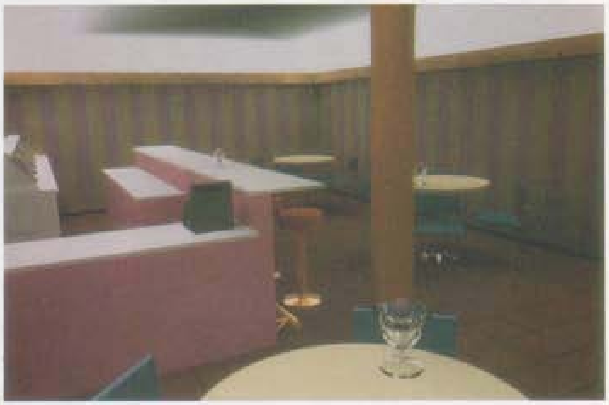


Figure 10: Ice cream store with indirect cove lighting.

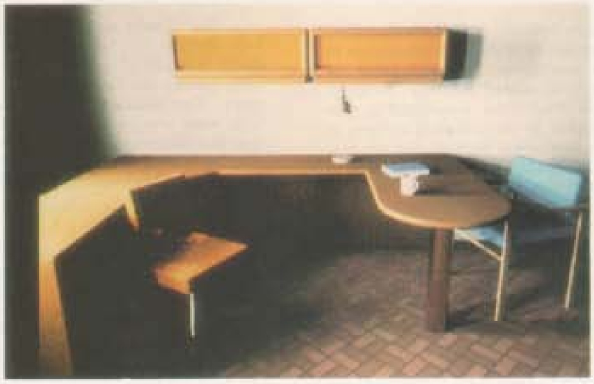


Figure 9a: Daylit office, direct only calculation.

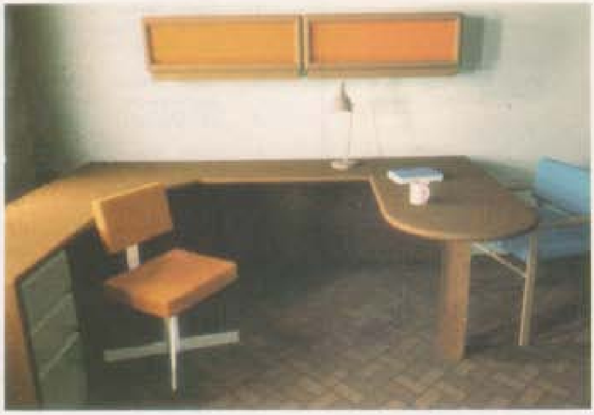


Figure 9b: Daylit office, first bounce calculation.

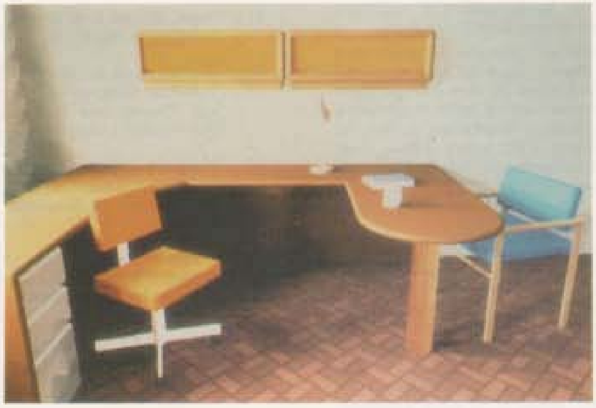


Figure 9c: Daylit office, seven bounce calculation.

5. Discussion

Because our averaging technique was derived from a simplified model (the split sphere), it is important to study its performance in situations where the model is not predictive. Two such cases are illustrated in Figure 11. They are both related to bright, localized reflections, such as those that might result from a spotlight or mirror. If a bright spot is partially hidden by an occluding surface, or on the horizon, then small changes in element location and orientation can result in large changes in illuminance. The averaging technique we have developed will not respond appropriately, and the error related to a will be much larger than the original split sphere model. However, bright spots also make trouble for the Monte Carlo calculation, which requires a higher sample density to find and integrate such luminance spikes. There is no known lighting calculation that can track these small "secondary sources" efficiently. Our technique uses a smaller value for a together with a higher Monte Carlo sample density to model these effects, with a corresponding increase in complexity.

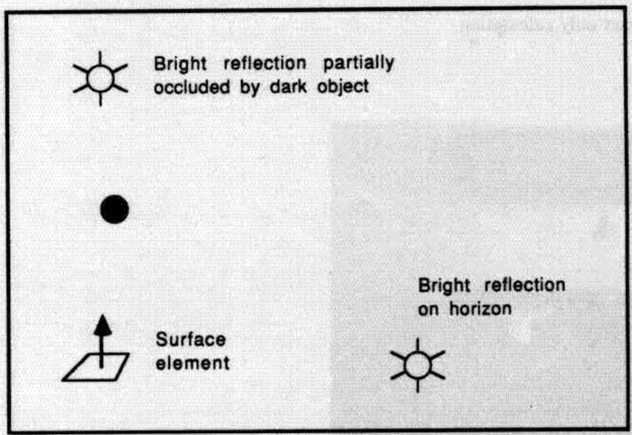

Bright reflection partially occluded by dark object
Bright reflection on horizon
Surface element

Figure 11: Two cases of indirect illumination that are difficult to model.

Besides diffuse interreflection, the caching technique can also be applied to illumination from large sources, such as a window or the sky. Wide area sources present a problem for conventional ray tracing calculations because they are difficult to sample adequately. Normally, when a ray participating in the diffuse interreflection calculation hits a light source, it is ignored. This prevents counting light sources twice, since they participate in a separate direct component calculation. By moving a large source from the direct to the indirect step of the computation, it is possible to obtain a more accurate sampling of its area. We have found this approach effective for sources with a solid angle greater than 1 steradian.

The calculation of diffuse transmission can also be accelerated by caching. Translucent surfaces become more difficult to model with conventional point sampling techniques as they become more nearly diffuse. The indirect calculation can be used to obtain a more accurate integral of light striking a translucent surface on either side. If the transmission function is not purely diffuse, scattered specular rays can be used to supplement the Monte Carlo calculation, just as they are for reflection.

6. Conclusion

We have developed an efficient ray tracing method for calculating diffuse interreflection, which when combined with standard computations of direct and specular contributions results in a complete simulation of global illumination. Only those illuminance computations required for accurate rendering are performed, and the values can be reused in other images. Thus the method provides an good mix of view-dependent and view-independent qualities. The criterion for evaluation of diffuse interreflection is an estimate of the illuminance gradient from convenient measures of scene geometry. The separation of lighting and geometric models is a basic strength of ray tracing, and it is preserved in this technique. The method can also model diffuse transmission and illumination from large area sources.

7. Acknowledgements

Our thanks go to the reviewers for their comments, and to Sam Berman for his continuing support. We would also like to thank the LBL Computer Science Research Department and the UCB Center for Environmental Design for the use of their equipment. Our special thanks go to Bill Johnston and Paul Heckbert. This work was supported by the Assistant Secretary for Conservation and Renewable Energy, Office of Building Energy Research and Development, Buildings Equipment Division of the U.S. Department of Energy under Contract No. DE-AC03-76SF00098.

8. References

1. Bentley, Jon Louis and Jerome Friedman, "Data Structures for Range Searching," *ACM Computing Surveys*, Vol. 11, No. 4, 1979, pp. 397-409.

2. Cohen, Michael and Donald Greenberg, "A Radiosity Solution for Complex Environments," *Computer Graphics*, Vol. 19, No. 3, July 1985, pp. 31-40.

3. Cohen, Michael, Donald Greenberg, David Immel, Phillip Brock, "An Efficient Radiosity Approach for Realistic Image Synthesis," *IEEE Computer Graphics and Applications*, Vol. 6, No. 2, March 1986, pp. 26-35.

4. Cook, Robert L. and Kenneth E. Torrance, "A Reflection Model for Computer Graphics," *ACM Transactions on Graphics*, Vol. 1, No. 1, January 1982, pp. 7-24.

5. Cook, Robert, Thomas Porter, Loren Carpenter, "Distributed Ray Tracing," *Computer Graphics*, Vol. 18, No. 3, July 1984, pp. 137-147.

6. Cook, Robert L., "Stochastic Sampling in Computer Graphics," *ACM Transactions on Graphics*, Vol. 5, No. 1, January 1986, pp. 51-72.

7. Immel, David S., Donald P. Greenburg, Michael F. Cohen, "A Radiosity Method for Non-Diffuse Environments," *Computer Graphics*, Vol. 20, No. 4, August 1986, pp. 133-142.

8. Kajiya, James T., "The Rendering Equation," *Computer Graphics*, Vol. 20, No. 4, August 1986, pp. 143-150.

9. Kaufman, John, *IES Lighting Handbook*, Reference Volume, IESNA, New York, NY, 1981.

10. Nishita, Tomoyuki and Eihachiro Nakamae, "Continuous Tone Representation of Three-Dimensional Objects Taking Account of Shadows and Interreflection," *Computer Graphics*, Vol. 19, No. 3, July 1985, pp. 23-30.

11. Rubenstein, R.Y., *Simulation and the Monte Carlo Method*, J. Wiley, New York, 1981.

12. Siegel, R. and J. R. Howell, *Thermal Radiation Heat Transfer*, Hemisphere Publishing Corp., Washington DC., 1981.

13. Wallace, John R., Michael F. Cohen, Donald P. Greenburg, "A Two-Pass Solution to the Rendering Equation: A Synthesis of Ray Tracing and Radiosity Methods," *Computer Graphics*, Vol. 21, No. 4, July 1987, pp. 311-320.

14. Weghorst, Hank, Gary Hooper, Donald P. Greenburg. "Improved computational methods for ray tracing" *ACM Transactions on Graphics*, Vol. 3, No. 1, January 1984, pp. 52-69.

15. Whitted, Turner, "An Improved Illumination Model for Shaded Display," *Communications of the ACM*, Vol. 23, No. 6, June 1980, pp. 343-349.

Reprinted with permission of Eurographics

Irradiance Gradients

Gregory J. Ward*
LESO-PB
Ecole Polytechnique Federale de Lausanne
CH-1015 Lausanne

Paul S. Heckbert**
Department of Technical Mathematics & Informatics
Delft University of Technology
Julianalaan 132
2628 BL Delft
Netherlands

ABSTRACT

A new method for improving the accuracy of a diffuse interreflection calculation is introduced in a ray tracing context. The information from a hemispherical sampling of the luminous environment is interpreted in a new way to predict the change in irradiance as a function of position and surface orientation. The additional computation involved is modest and the benefit is substantial. An improved interpolation of irradiance resulting from the gradient calculation produces smoother, more accurate renderings. This result is achieved through better utilization of ray samples rather than additional samples or alternate sampling strategies. Thus, the technique is applicable to a variety of global illumination algorithms that use hemicubes or Monte Carlo sampling techniques.

1. Introduction

Global illumination can be simulated using both ray tracing and radiosity algorithms. Both approaches typically rely on calculations of patch irradiances which are used to revise other patch irradiances iteratively or to render a final image†. In most radiosity algorithms, patch radiosities are considered

*First author's current address:
Lawrence Berkeley Laboratory
1 Cyclotron Rd., 90-3111
Berkeley, CA 94720
USA
+1-510-486-4757
+1-510-486-4089 fax
e-mail: gjward@lbl.gov
**Second author's current address:
School of Computer Science
Carnegie Mellon University
5000 Forbes Ave
Pittsburgh PA 15213-3890
USA
+1-412-268-7897
e-mail: ph@cs.cmu.edu

†Irradiance is the energy flux per unit area arriving on a surface. Radiosity is the emissive flux per unit area, plus the irradiance times the diffuse surface reflectance.

constant during the solution stage, and bilinear interpolation (Gouraud shading) is used to compute pixel values during rendering. It has been shown that these piecewise-constant approximations are quite inaccurate, and that much more accurate approximations can be simulated using linear, quadratic, and higher order approximation [Heckbert91b]. Linear approximations have recently been implemented for 3-D radiosity [Max92] and 2-D radiosity [Heckbert91a] [Lischinski91].

Higher order approximations require more information about the irradiance function in order to be worthwhile, however; one must increase either the number of samples or the information content of each sample. If higher order interpolation were used without additional information, the resulting shading would look smoother, but it would be objectively no more accurate than standard bilinear interpolation.

Rather than increase the number of samples, as more brute-force algorithms have done, our method increases the information content of each sample to include estimates of the first derivative, or gradient, of the irradiance.

In this paper, we will show how the irradiance gradient at a point can be computed during a standard Monte Carlo evaluation of irradiance at almost no additional cost. The key to this innovation is the wealth of information contained in a sampling of the hemisphere. During the sampling process, the distances, brightnesses, and directions of the visible surfaces are all known, and from this knowledge it is possible to deduce with reasonable accuracy how the irradiance will change with respect to position and orientation of the test surface element. The gradient approximation given here is based on minimal, intuitive assumptions of geometric continuity.

Knowing the irradiance gradient along with the irradiance value at a point allows us to justify a bicubic or like-order interpolation method, and produces not only smoother but significantly more accurate results. The irradiance gradient method will be demonstrated in the context of a meshless irradiance caching scheme [Ward88b], though the technique may also be applied in mesh-dependent radiosity algorithms.

2. The RADIANCE Simulation

Radiance is a physically-based lighting simulation system developed over the past seven years at the Lawrence Berkeley Laboratory in California and the Ecole Polytechnique Federale de Lausanne in Switzerland. The software is free and publicly available from anonymous ftp sites at both locations. Since the algorithm described in this paper has been implemented in the context of the *Radiance* program, it is necessary to briefly explain the workings of this simulation before delving into the irradiance gradient calculation itself.

Radiance is basically a light-backwards ray tracing program [Whitted80] that uses irradiance caching to efficiently account for diffuse interreflection between surfaces. The basic algorithms used in *Radiance* are described in [Ward88a] and a general overview of the software is provided in [Ward90]. Basically, *Radiance* uses ray tracing in a recursive evaluation of the radiance equation†:

$$L_r(\theta_r,\phi_r) = \int_0^{2\pi}\int_0^{\pi/2} L_i(\theta_i,\phi_i)\, f(\theta_i,\phi_i;\theta_r,\phi_r)\, \cos\theta_i\, \sin\theta_i\, d\theta_i\, d\phi_i \qquad (1)$$

where:

θ is the polar angle measured from the surface normal

ϕ is the azimuthal angle measured about the surface normal

$L_r(\theta_r,\phi_r)$ is the reflected radiance (watts/steradian/meter^2 in SI units)

$L_i(\theta_i,\phi_i)$ is the incident radiance

$f(\theta_i,\phi_i;\theta_r,\phi_r)$ is the bidirectional reflectance-transmittance distribution function (steradian^{-1})

†The radiance equation is essentially Kajiya's rendering equation [Kajiya86] with the notion of energy transfer between

To reduce the variance between samples and speed convergence of Monte Carlo integration, light sources are accounted for separately using an adaptive sampling scheme [Ward91]. As in most ray tracing algorithms, specular contributions are computed with separate rays in the appropriate directions. Once the direct and specular contributions have been removed from the integral, the indirect diffuse contribution is computed using a Monte Carlo sampling of the hemisphere. Since this "indirect irradiance" value is view-independent, and it changes slowly over surfaces in most scenes, it is more efficient to perform the calculation only occasionally, caching the computed values for local interpolation. This caching of irradiance samples is a significant optimization of more brute-force Monte Carlo ray tracing algorithms such as Kajiya's [Kajiya86].

In the meshless caching scheme described in [Ward88b], the location of the computed indirect irradiance values is determined by the proximity and curvature of the surfaces, and does not fall on a regular grid, so a weighted sum is used in place of a more standard bilinear interpolation. Furthermore, since values are only computed as needed by the algorithm, extrapolation may occur in a region where no previous values existed, until the need for a new value is strong enough to trigger another Monte Carlo calculation. This process can result in some rather disturbing artifacts in a single pass scanline rendering, as shown in Figure 1a. The commonly applied solution to this problem involves a low-resolution overture calculation to fill the desired view with indirect irradiance values prior to the final high-resolution pass. Although this prepass requires only a modest additional expense, it is rather annoying that it should be required by an otherwise elegant rendering algorithm.

The benefit of calculating the irradiance gradient is two-fold for the caching scheme used by *Radiance*. First, we are able to produce more accurate interpolated values because we can use the gradient information effectively in a higher order interpolating function. Second, we are able to produce more accurate *extrapolated* values and thus greatly reduce caching artifacts. Figure 1b shows the same single-pass calculation, this time using estimates of the irradiance gradient to more accurately extrapolate values in unsampled regions of the image. We emphasize that the second image took approximately the same time to produce as the first, and used the same hemisphere samples to compute the irradiances. The only difference is that the second image extracted additional information from the hemisphere samples to deduce the irradiance gradient at each sample point, and these gradients were used to better interpolate and extrapolate irradiance values for the image. (Since the test environment contains no specular surfaces and no direct illumination sources, changes in the diffuse interreflection calculation are more evident here than in most scenes.)

Figure 1c plots interpolated irradiance values for a vertical line passing just under the right side of the sphere. The difference between the actual irradiance and the cubic interpolation is too small to see in this plot, but the poor match of the linearly interpolated irradiance is clearly evident. The relative errors shown in Figure 1d amply demonstrate that a cubic interpolation of irradiance based on gradient information is much more accurate than a standard linear interpolation.

2.1. Indirect Irradiance Calculation

The indirect irradiance is calculated in *Radiance* using a fixed number of samples in a uniformly weighted, stratified Monte Carlo sampling:

$$E = \frac{\pi}{M \cdot N} \sum_{j=0}^{M-1} \sum_{k=0}^{N-1} L_{j,k} \tag{2}$$

where:

$L_{j,k}$ is the indirect radiance in the direction $(\theta_j, \phi_k) = \left(\sin^{-1} \sqrt{\frac{j+X_j}{M}},\ 2\pi \frac{k+Y_k}{N} \right)$

X_j, Y_k are uniformly distributed random variables in the range [0,1)

two points replaced by energy passing through a point in a specific direction.

$M \cdot N$ is the total number of samples and $N \approx \pi M$

Note that sample rays that intersect light sources must be excluded from the above summation because direct illumination is accounted for in a separate step. The resulting sum is the indirect contribution to irradiance at a specific point on a surface.

Irradiance samples, consisting of a point, a normal vector, and an irradiance value, are stored in an octree for later interpolation. This irradiance octree is separate from the octree used to optimize ray-surface intersection. In the original implementation, interpolation was a simple weighted sum over usable irradiance values. The weight of a sample decreased as the interpolation point or normal vector deviated away from the sample's point and normal. A sample was considered usable if the estimated error of its contribution to the approximation was less than a user-specified accuracy tolerance. The error was estimated from the local geometry using the "split sphere model" to compute an upper bound on the magnitude of the irradiance gradient.

The split sphere model is only a crude estimate of the gradient magnitude. It may be sufficient to decide the spacing of irradiance calculations, but to improve our interpolation we need to know the actual irradiance gradient, not just a directionless upper bound. Fortunately, the information we need is already contained in the hemisphere sampling.

3. The Gradient Calculation

Since the irradiance in a scene is a function of five variables, three for the position and two for the direction, the irradiance gradient should be a five-dimensional vector. For computational convenience, we will compute instead two separate three-dimensional vectors. One will correspond to the expected direction and magnitude of the **rotational gradient** and the other to the direction and magnitude of the **translational gradient**. Both gradient vectors will lie in the base plane of the sample hemisphere, which is the tangent plane of the sample. Thus, each vector will in fact represent only two degrees of freedom. This representation of the gradient is used because we only interpolate across a surface. Furthermore, the irradiance above and below most surfaces is discontinuous, and the gradient with respect to displacement in the polar direction is therefore undefined.

Our calculations of the rotational and translational irradiance gradients are based on very simple observations about the sampled environment. The sampling of rays over a hemisphere tells us much more than the total light falling on the surface. It tells us the distance, direction, and brightness of each contribution.

The directions and brightnesses tell us how irradiance changes as the sample hemisphere is rotated because they indicate how the cosine projection of those contributions affects the overall sum. To take a simple example, Figure 2a shows a single contributing surface. The background is assumed to be darker than the surface. If we rotate our sample hemisphere to face this surface, its contribution becomes proportionally larger than other contributions. If we rotate away from the surface, its overall contribution is diminished. By summing over all such potential changes, we can compute the total rotational gradient for the hemisphere.

For the translational gradient, the distances to the contributing surfaces must be considered because occlusion plays an important role. In Figure 2b, a darker surface occludes a brighter surface in the background. As the sample hemisphere is moved to the right in the diagram, the influence of the brighter background surface becomes stronger, and therefore the translational gradient is positive in this direction. By summing over at all such changes, we can compute the overall gradient with respect to translation.

As an example of the kind of information available during the hemisphere sampling, see Figure 3. Figure 3a shows a projected hemisphere as seen from a point on the floor of a conference room. Figure 3b is a false color image showing the distances to the surfaces as determined by a ray tracing calculation. If we were to move towards the chair in the upper left of the image, we would note a decrease in the overall irradiance as the chair's dark underside covered more of our view of the ceiling. Figure 3c shows a uniformly weighted stratification of about 2000 samples as computed by the *Radiance* interreflection calculation. Notice that the light sources appear dark, as they must be excluded from the indirect contributions. Notice also that this image appears very crude as a rendering, yet it contains many more samples than are typically used to calculate the indirect irradiance.

3.1. The Rotational Gradient

The rotational gradient formula simply sums the differential of the cosine for each contribution sample. For the hemisphere sampling given in Equation (2), the formula is:

$$\vec{\nabla}_r E = \frac{\pi}{M \cdot N} \sum_{k=0}^{N-1} \left\{ \hat{v}_k \sum_{j=0}^{M-1} -\tan\theta_j \cdot L_{j,k} \right\} \tag{3}$$

where:

$\hat{v}_k$ is the base plane unit vector in the $\phi_k + \frac{\pi}{2}$ direction

The tangent function appears in the summation because the differential of the cosine is negative sine and our sampling contains the cosine weighting implicitly, thus it is necessary to multiply the sample values by the tangent (sine over cosine) to get back a sine weighting.

3.2. The Translational Gradient

To compute the translational irradiance gradient, we consider how the projected solid angle of each of the *MN* hemisphere cells is affected by a translation of the hemisphere center. The change in projected solid angle, when multiplied by the radiance of a cell, will give the change in that cell's contribution. This rate of change is determined in part by the distance to the contributing surface. The closer the surface, the higher the rate of change with respect to a displacement perpendicular to the boundary between neighboring cells. In fact, it is always the distance to the *closer* surface that determines the rate of change in occlusion, since the relative motion of a foreground surface is greater than that of a background surface.

Figure 4 shows the projection of a stratified hemisphere sampling onto the tangent plane of the surface. Each cell has an equal projected solid angle (ie. $\frac{\pi}{MN}$ steradians), thus each cell should cover the same area in this diagram. To determine how the irradiance changes with translation in this tangent plane, we sum the marginal changes for each cell. For cell (j,k) shown in the diagram, we consider two approximately perpendicular directions. (Note that we have shown the sample direction at the center of the cell, but in fact it lies at some random location in the cell.) One direction is polar, the other is azimuthal. Computing the marginal change in irradiance for this cell reduces to computing the marginal change in the two highlighted cell walls with respect to translation.

The change in irradiance with respect to translation for each cell wall is simply the length of the cell wall multiplied by the rate of motion of the wall with respect to motion in a specific direction. For the wall separating the two adjacent cells with the same θ, the length of the cell wall is given by the integral of cos(θ) from θ_{j_-} to θ_{j_+}. This is simply $(\sin\theta_{j_+} - \sin\theta_{j_-})$. For motion perpendicular to this wall (ie. in the direction $\hat{v}_{k_-}$ defined below), it is simple to show that motion of the cell wall is proportional to $1/Min(r_{j,k}, r_{j,k-1})$, where $r_{j,k}$ is the intersection distance in cell (j,k). Thus, the change in irradiance with respect to motion along $\hat{v}_{k_-}$ for cell (i,j) is:

$$\frac{\sin\theta_{j_+} - \sin\theta_{j_-}}{Min(r_{j,k}, r_{j,k-1})} \cdot (L_{j,k} - L_{j,k-1})$$

Since we have computed the change in the location of the cell wall, we must use the difference in the adjacent radiance samples to determine how this will affect the overall irradiance sum.

For the cell wall separating the two adjacent cells in the polar direction, the length of the cell wall is $\frac{2\pi}{N} \cdot \sin\theta_{j-}$. The motion of the wall with respect to the vector perpendicular to it (ie. $\hat{u}_k$ defined below)

Figure 4. Cells of an example hemisphere sampling.

is equal to $\dfrac{\cos^2\theta_{j_-}}{Min(r_{j,k}, r_{j-1,k})}$. One cosine comes from the projection of the hemisphere's tangent into the plane, and the other cosine comes from the reduced change in θ as a function of angle. (A rigorous proof of this formula is left as an exercise for the reader.) Combining these terms, we arrive at the following formula for the change in irradiance with respect to motion along $\hat{u}_k$ for the cell (i,j):

$$\frac{2\pi}{N}\cdot\frac{\sin\theta_{j_-}\cdot\cos^2\theta_{j_-}}{Min(r_{j,k}, r_{j-1,k})}\cdot(L_{j,k} - L_{j-1,k})$$

Combining these terms into a sum over all cells, we arrive at the following formula for the translational irradiance gradient:

$$\vec{\nabla}_t E = \sum_{k=0}^{N-1}\left\{\hat{u}_k \frac{2\pi}{N}\sum_{j=1}^{M-1}\frac{\sin\theta_{j_-}\cdot\cos^2\theta_{j_-}}{Min(r_{j,k}, r_{j-1,k})}\cdot(L_{j,k} - L_{j-1,k}) \;+\; \hat{v}_{k_-}\sum_{j=0}^{M-1}\frac{\sin\theta_{j_+} - \sin\theta_{j_-}}{Min(r_{j,k}, r_{j,k-1})}\cdot(L_{j,k} - L_{j,k-1})\right\} \qquad (4)$$

where:

$\hat{u}_k$ is the unit vector in the ϕ_k direction

$\hat{v}_{k_-}$ is the unit vector in the $\phi_{k_-} + \dfrac{\pi}{2}$ direction

θ_{j_-} is the polar angle at the previous boundary, $\sin^{-1}\sqrt{\dfrac{j}{M}}$

θ_{j_+} is the polar angle at the next boundary, $\sin^{-1}\sqrt{\dfrac{j+1}{M}}$

$\phi_{k\_}$ is the azimuthal angle at the previous boundary, $2\pi \frac{k}{N}$

$r_{j,k}$ is the intersection distance for cell (j,k)

This summation has been regrouped to use the *differences* in radiance between neighboring samples, summing over the boundaries rather than the cells themselves. This yields a much simpler formula.

3.3. Applying Gradients to Interpolation

If the irradiance were being interpolated on a mesh, it would be straightforward to apply the rotational and translational gradients in a bicubic interpolation. However, the meshless interpolation used by *Radiance* demands a slightly different approach. The method we have chosen uses the same weighted average and even the same weights as the original method, but with the irradiance values adjusted by their corresponding rotational and translational gradients. This in effect increases the order of the interpolation. The interpolation does not have a polynomial basis, however, so it is difficult to compare to more conventional forms.

The irradiance interpolation using the gradient vectors calculated by Equations (3) and (4) is as follows:

$$E(\vec{P}) = \frac{\sum_{\mathbf{S}} w_i(\vec{P}) \left[E_i \; + \; (\hat{n}_i \times \hat{n}) \cdot \vec{\nabla}_r E_i \; + \; (\vec{P} - \vec{P}_i) \cdot \vec{\nabla}_t E_i \right]}{\sum_{\mathbf{S}} w_i(\vec{P})} \qquad (5)$$

where:

$$w_i(\vec{P}) = \frac{1}{\frac{||\vec{P} - \vec{P}_i||}{R_i} + \sqrt{1 - \hat{n} \cdot \hat{n}_i}} \; = \; \text{weight of sample } i$$

$\vec{P}$ is the test point position

$\hat{n}$ is the surface normal at the test point

E_i is the irradiance value at sample i

$\vec{P}_i$ is the position of sample i

$\hat{n}_i$ is the surface normal at sample i

R_i is the harmonic mean distance to objects visible from $\vec{P}_i$

S is the set of valid irradiance values, $\{ i : w_i(\vec{P}) > 1/a \}$

a is a user-selected accuracy goal

Note that our calculation of the irradiance gradient is not used to determine the spacing of samples. Such an approach would tend to bias the calculation, since areas that just happened to have small irradiance gradients would be sampled at low density, even though there could still be sudden changes in the irradiance value due to nearby surfaces. It is better to use the split sphere model to respond to the largest expected gradient in determining the sample density, so that we do not miss anything important. However, we may still want to use the more accurate formula (4) to increase the sample density where the split sphere model underestimates the gradient.

The interpretation of the irradiance gradient allows one to perform the calculation separately for each sample wavelength, or to combine into a spectral average. The latter was chosen for our implementation to save on storage costs, although this may be an unwarranted compromise in some contexts.

4. Conclusion

The irradiance gradient can be calculated from the same hemisphere samples used to compute the point irradiance. The extra computational cost is negligible compared to the cost of computing the samples, and the benefit is greatly improved interpolation accuracy. This may sound like we are getting something for nothing, but in fact we are only better utilizing the information already available from a hemisphere sampling.

Figure 5a shows a room with daylight entering through a doorway. This image was computed without irradiance gradients in a two-pass calculation using interpolated values. Figure 5b shows the same scene rendered using irradiance gradients to effectively increase the order of interpolation. Notice that the second image is smoother around the doorway, where the indirect component is most influential. Both images took roughly the same time to compute.

The gradient calculation as described in this paper is appropriate to any global illumination method where surface brightnesses are being sampled over a hemisphere, such as a gathering radiosity or Monte Carlo algorithm. Z-buffer methods such as the hemicube also yield the information necessary to compute gradients, so the approach is not limited to ray tracing algorithms.

It may also be possible to use hemisphere sampling information to speed convergence of a progressive radiosity or shooting Monte Carlo algorithm by noting the arrangement of visible surfaces and subdividing shooting patches where a shadow boundary is indicated. Since the final radiance of samples is not known in such a technique, subdivision would have to be based mainly on geometric considerations. Nevertheless, the information contained in a hemisphere sampling is considerable, and it seems wasteful to ignore it.

5. Acknowledgements

The first author was supported by the Swiss LUMEN Project, and the Assistant Secretary for Conservation and Renewable Energy, Office of Building Technologies, Buildings Equipment Division of the U.S. Department of Energy under Contract No. DE-AC03-76SF00098. The second author was supported by a fellowship from the Delft University of Technology. Jean-Louis Scartezzini of the University of Geneva and Raphael Compagnon of EPFL in Switzerland provided invaluable support and assistance. Raphael and his students created the model shown in Figure 5. Anat Grynberg created the conference room model shown in Figure 3.

6. Software Availability

The *Radiance* software discussed in this paper is available from anonymous ftp at the following sites:

hobbes.lbl.gov	128.3.12.38	Berkeley, California
dasun2.epfl.ch	128.178.62.2	Lausanne, Switzerland

7. References

[Heckbert91a]
Paul Heckbert, *Simulating Global Illumination Using Adaptive Meshing*, PhD Thesis, Tech. Report UCB/CSD 91/636, Computer Science Division, University of California at Berkeley, June 1991.

[Heckbert91b]
Paul Heckbert and Jim Winget, ‘‘Finite Element Methods for Global Illumination,’’ Tech. Report UCB/CSD 91/643, Computer Science Division, University of California at Berkeley, July 1991.

[Kajiya86]
James T. Kajiya, ‘‘The Rendering Equation,’’ *Computer Graphics,* Vol. 20, No. 4, August 1986.

[Lischinski91]
Dani Lischinski, Filippo Tampieri, and Donald P. Greenberg, *Improving Sampling and*

Reconstruction Techniques for Radiosity, Computer Science Dept., Cornell University, Tech. Report 91-1202, Aug. 1991.

[Max92]
Nelson Max and Michael Allison, ‘‘Linear Radiosity Approximation using Vertex-to-Vertex Form Factors,’’ *Graphics Gems III*, edited by David Kirk, Academic Press, 1992 (to appear).

[Ward88a]
Gregory Ward and Francis Rubinstein, ‘‘A New Technique for Computer Simulation of Illuminated Spaces,’’ *Journal of the Illuminating Engineering Society*, Vol. 17, No. 1, Winter 1988.

[Ward88b]
Gregory Ward, Francis Rubinstein, and Robert Clear, ‘‘A Ray Tracing Solution for Diffuse Interreflection,’’ *Computer Graphics*, Vol. 22, No. 4, August 1988.

[Ward90]
Gregory Ward, ‘‘Visualization,’’ *Lighting Design and Application*, Vol. 20, No. 6, June 1990.

[Ward91]
Gregory Ward, ‘‘Adaptive Shadow Testing for Ray Tracing,’’ *Second Eurographics Workshop on Rendering,* Barcelona, Spain, April 1991.

[Whitted80]
Turner Whitted, ‘‘An Improved Illumination Model for Shaded Display,’’ *Communications of the ACM,* Vol. 23, No. 6, June 1980, pp. 343-349.

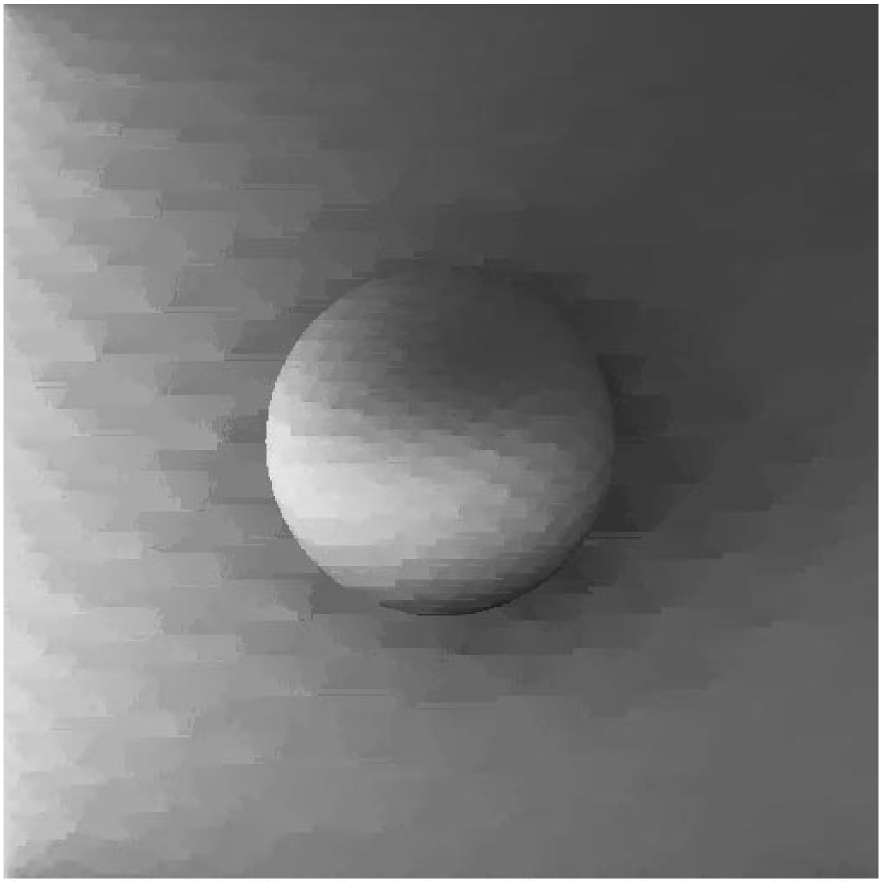


Figure 1a. Irradiance extrapolation without gradients.

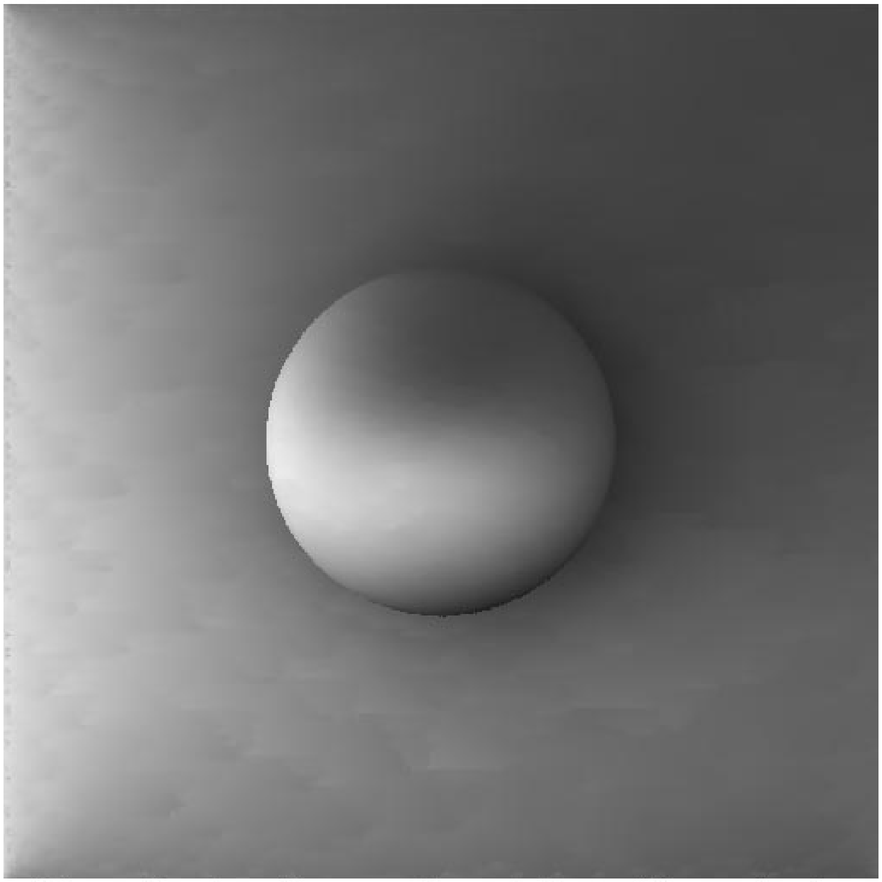
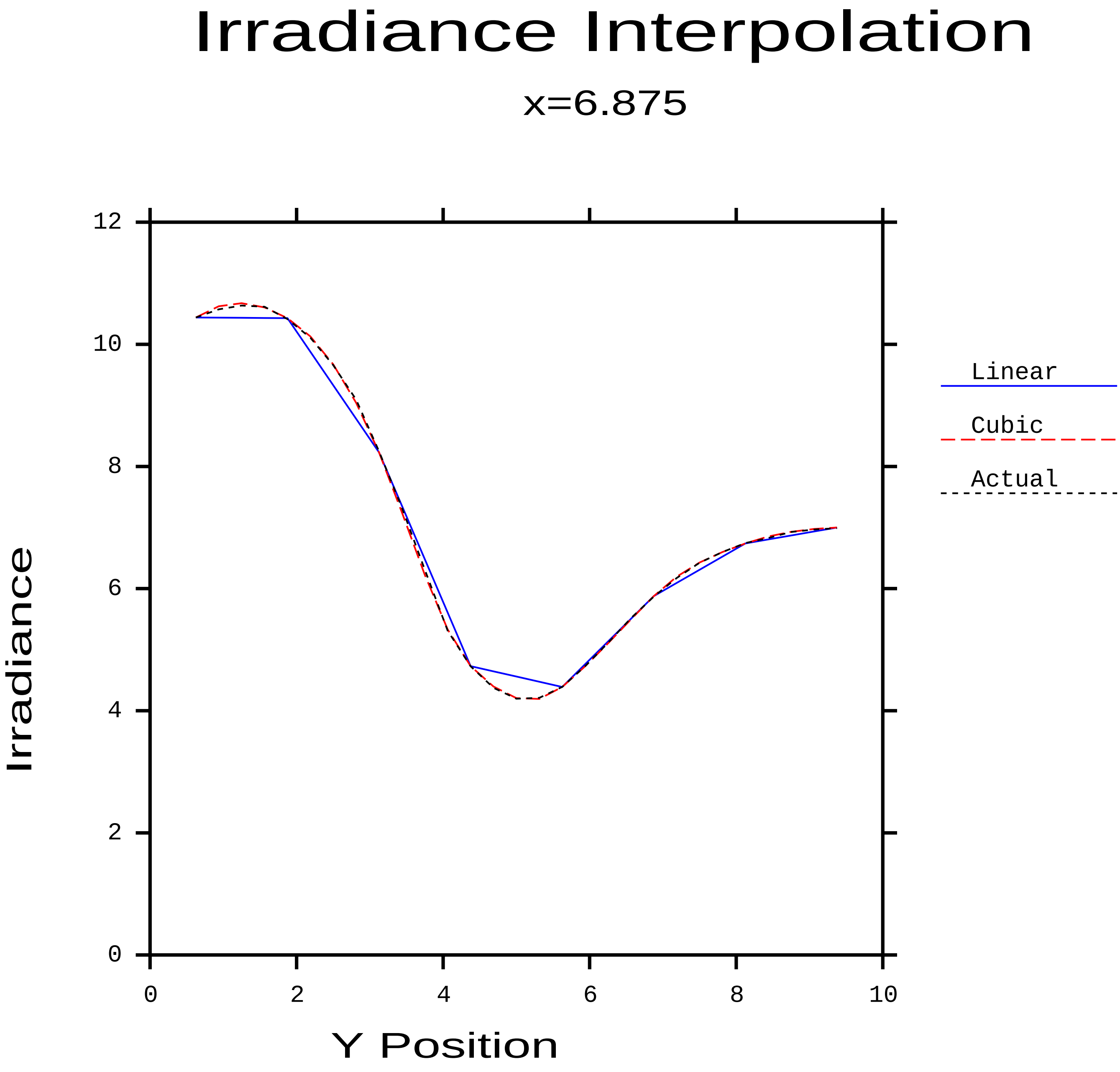


Figure 1b. Irradiance extrapolation with gradients.

Reprinted with permission of Eurographics



Irradiance Interpolation

x=6.875

Reprinted with permission of Eurographics

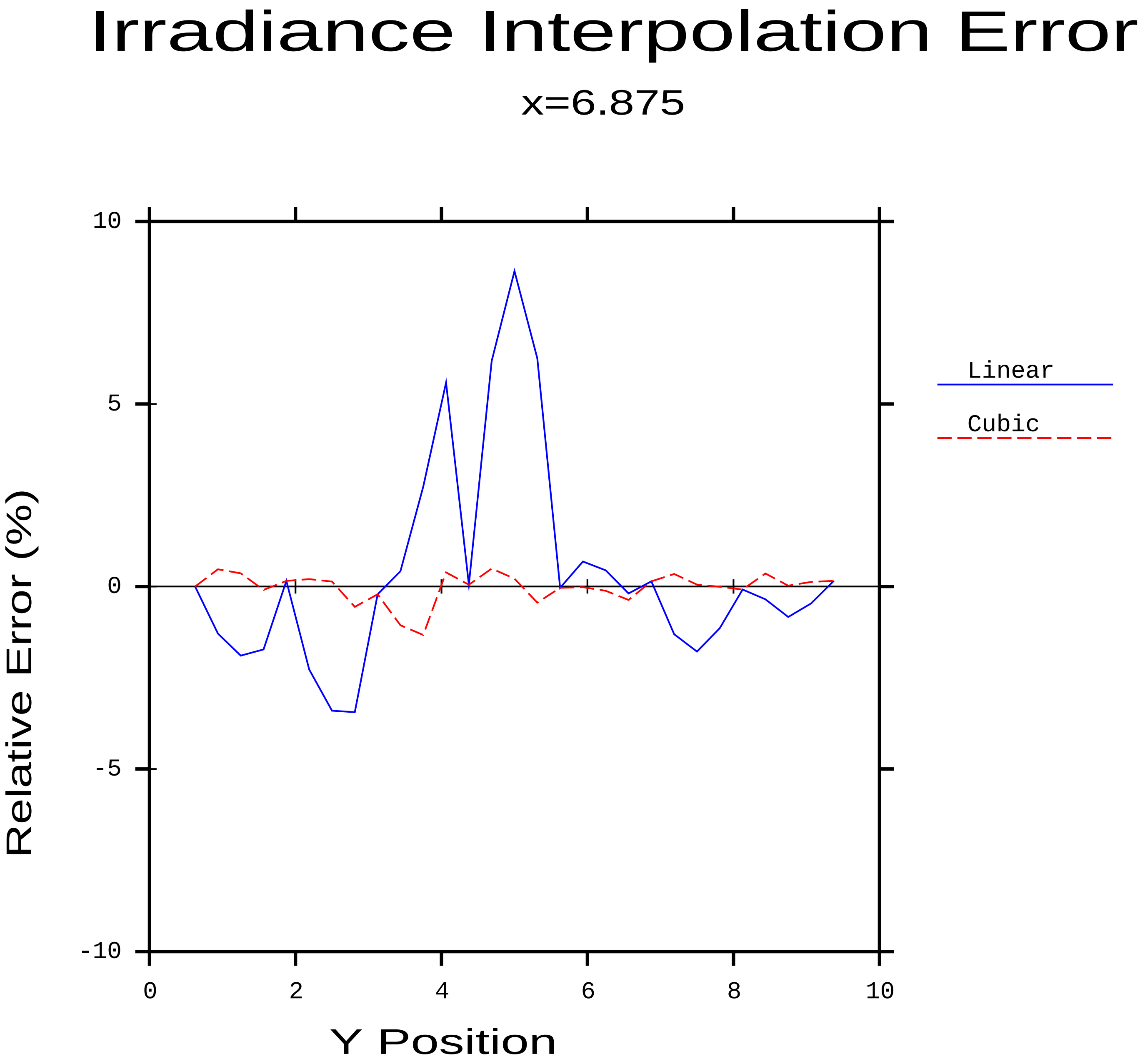


Figure 1c. Comparison between actual irradiance and interpolated values under diffuse sphere.

Figure 1d. Relative error due to interpolation under diffuse sphere.

Reprinted with permission of Eurographics

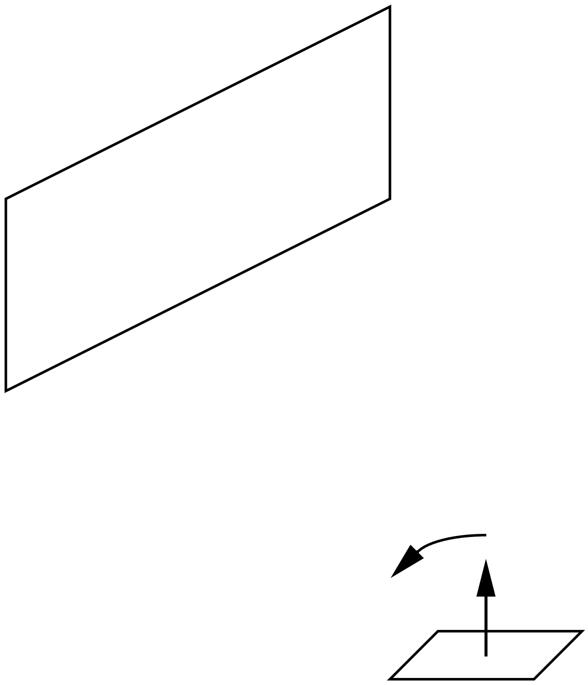


Figure 2a. As our point is rotated counter-clockwise, the surface's contribution increases.

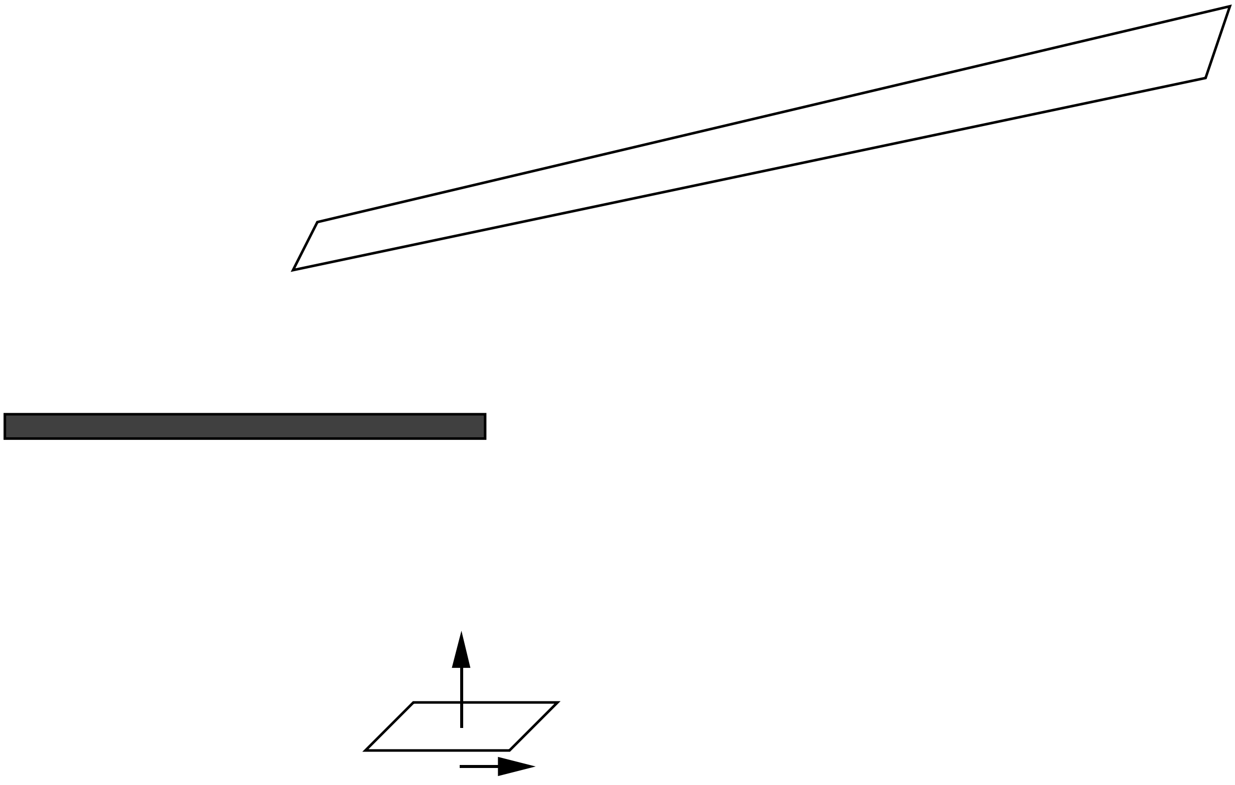


Figure 2b. Translational Gradient. As our point moves to the right, irradiance increases.

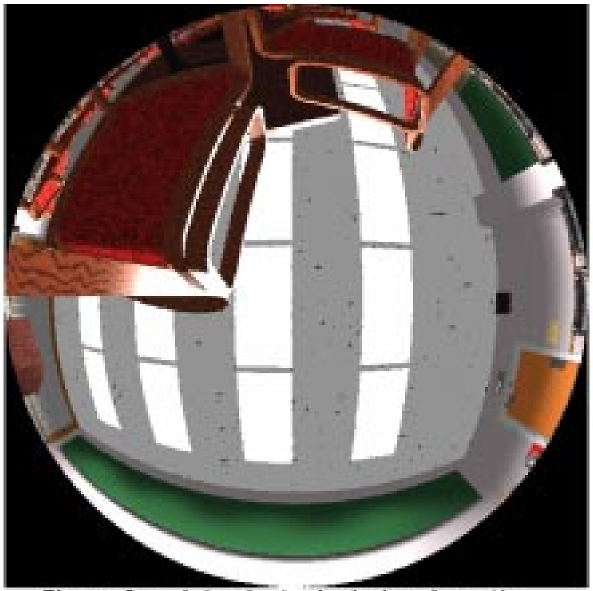


Figure 3a. A hemispherical view from the the floor of a conference room.

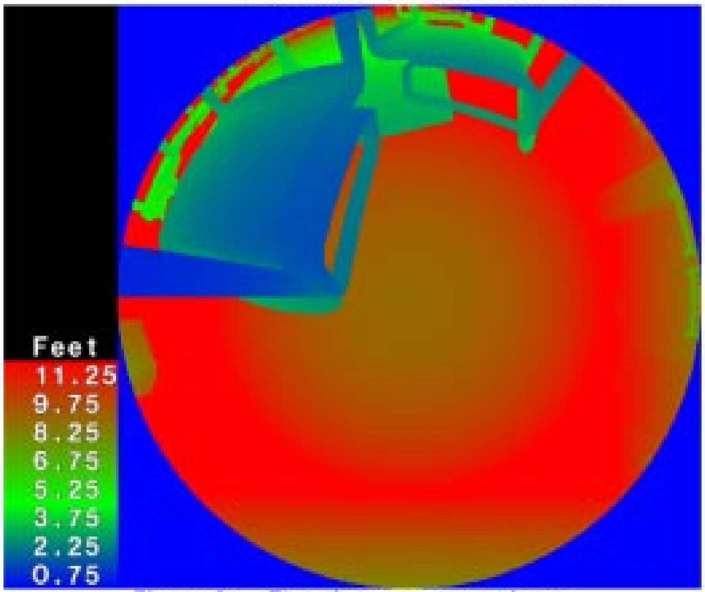


Figure 3b. The distances to surfaces visible in Figure 3a.

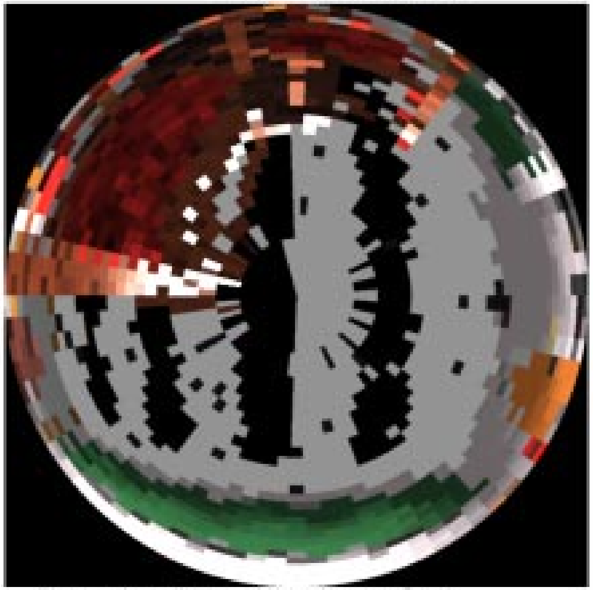


Figure 3c. A stratified Monte Carlo sampling of the same hemisphere.

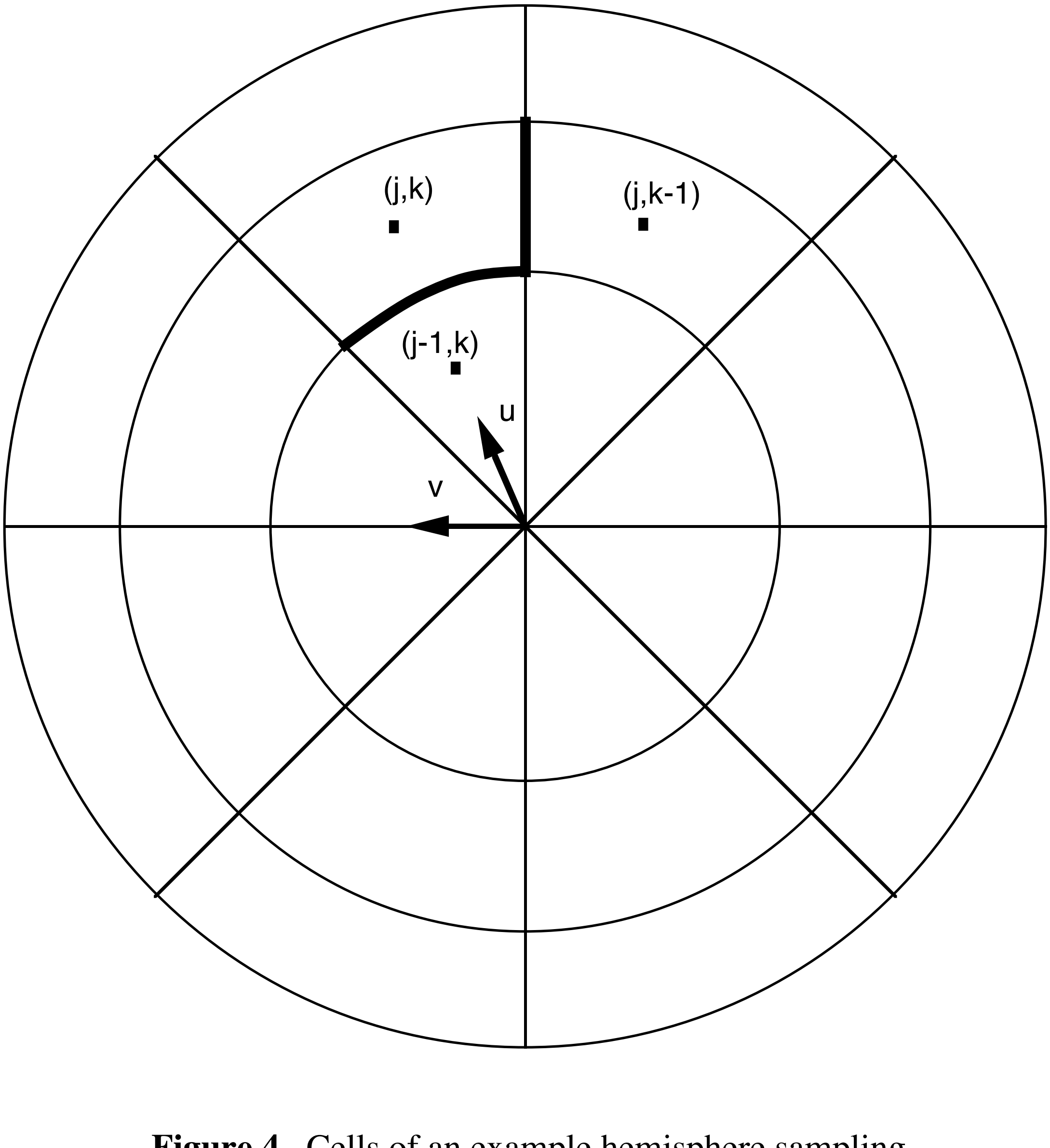


Figure 4. Cells of an example hemisphere sampling.

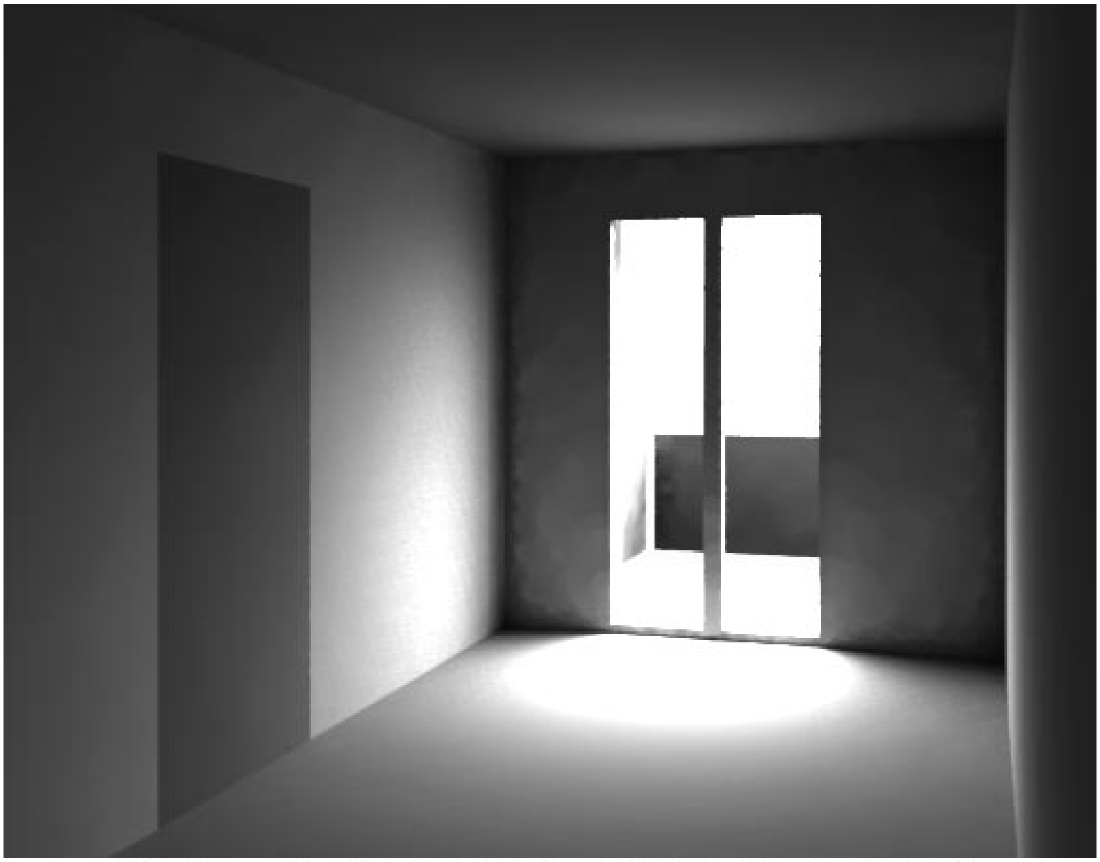


Figure 5a. Interpolation without gradients.

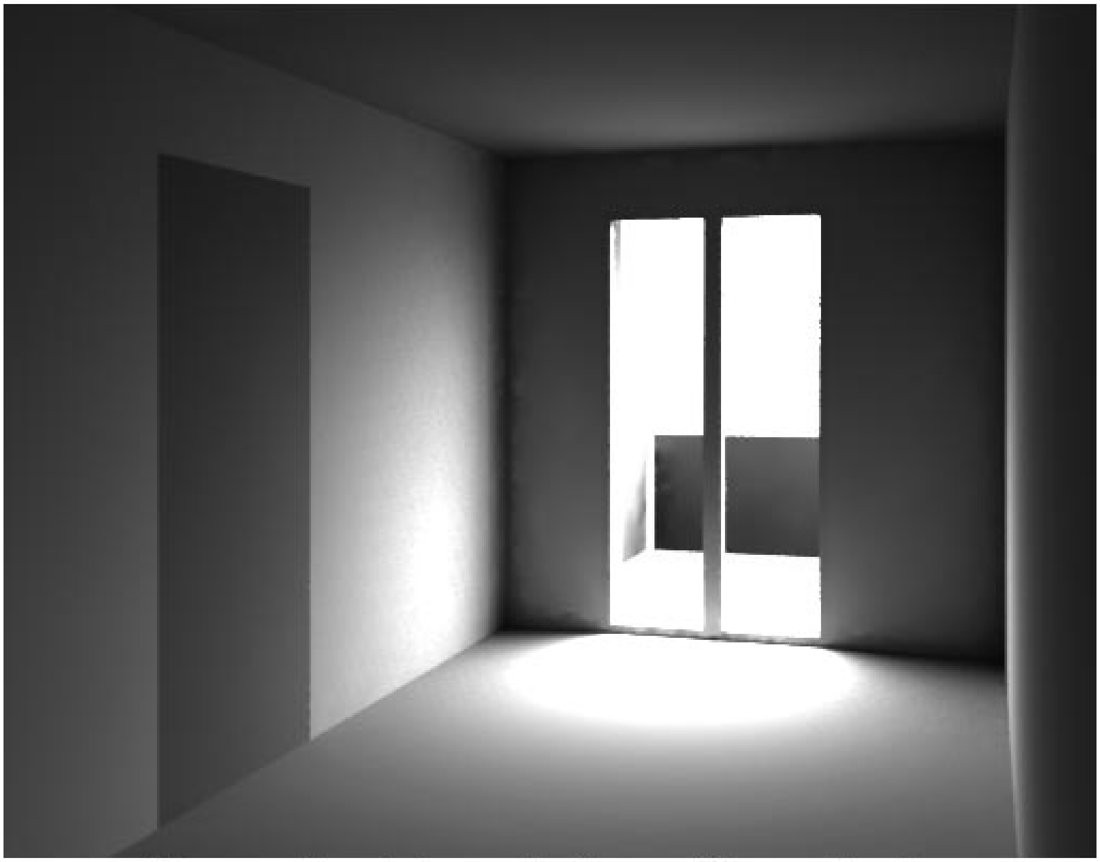


Figure 5b. Interpolation with gradients.

Eurographics Symposium on Rendering (2004)
H. W. Jensen, A. Keller (Editors)

Reprinted with permission of Eurographics

A Novel Hemispherical Basis for Accurate and Efficient Rendering

Pascal Gautron[†] Jaroslav Krivanek [‡] Sumanta Pattanaik [§] Kadi Bouatouch [¶]

Abstract

This paper presents a new set of hemispherical basis functions dedicated to hemispherical data representation. These functions are derived from associated Legendre polynomials. We demonstrate the usefulness of this basis for representation of surface reflectance functions, rendering using environment maps and for efficient global illumination computation using radiance caching. We show that our basis is more appropriate for hemispherical functions than spherical harmonics. This basis can be efficiently combined with spherical harmonics in applications involving both hemispherical and spherical data.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism - Rendering, Global Illumination

1. Introduction

High quality rendering and global illumination techniques require an accurate computation of the interaction between light and surfaces. Light/surface interaction models usually rely on hemispherical functions. For example, incident and outgoing radiances at any surface point are defined on a hemisphere, and a BRDF (bi-directional reflectance distribution function) is defined on the cartesian product of two hemispheres. For real time realistic rendering the efficiency and accuracy of the representation of these hemispherical functions is crucial. Researchers have adapted basis functions defined over spheres to represent hemispherical functions such as BRDFs and incident radiance functions (spherical harmonics [RH02, SKS02, WAT92, SAWG91, CMS87], or spherical wavelets [SS95]). However, hemispherical functions introduce discontinuities in the spherical domain at the boundary of the hemisphere and hence their accurate representation using spherical basis requires a large number of coefficients.

We introduce a *hemispherical* basis that ensures a more accurate representation of hemispherical functions compared to spherical harmonics basis. With minimal effort we combine functions represented by our hemispherical basis with those represented by spherical harmonics for various rendering purposes, such as environment map rendering. This allows our work to be easily integrated into existing algorithms that are designed for using spherical harmonic basis functions.

Our contributions in this paper are:

- The definition of a set of orthogonal basis functions to serve as a hemispherical basis.
- Methods for rotating functions represented using our new basis.
- Application of these basis functions to the representation of BRDFs and hemispherical radiance functions.
- Application of these basis functions to extend the irradiance caching scheme [War94] for non-Lambertian surfaces.

The organization of the paper is as follows. After a description of the state-of-the-art related to functional representations of hemispherical functions (Section 2), we show how the shifting of associated Legendre polynomials can lead to a basis of orthogonal polynomials defined on the

[†] University of Central Florida, Orlando, FL, USA - IRISA (Université de Rennes 1), Rennes, France

[‡] University of Central Florida, Orlando, FL, USA - Czech Technical University , Prague, Czech Republic - IRISA (Université de Rennes 1), Rennes, France

[§] University of Central Florida, Orlando, FL, USA

[¶] IRISA (Université de Rennes 1), Rennes, France

hemisphere (Section 3). Section 4 defines methods for efficiently rotating functions represented using our basis functions. In Sections 5 and 6, we present applications of our basis for BRDF representation and radiance caching in the context of global illumination.

2. Previous Work

Three general approaches for functional representation of hemispherical functions have been adopted in graphics. They are spherical harmonics, spherical wavelets and Zernike polynomial representations. A fourth approach, based on Jacobi polynomials, has been defined in the context of radiative transfer.

Spherical harmonics. Spherical harmonics (referred to as SH in the rest of the paper) represent functions defined on the sphere. They are widely used for representation of BRDFs and environment maps [RH02, SKS02, WAT92, SAWG91, CMS87]. Using spherical harmonics for both BRDFs and environment maps, the lighting integral can be computed as a vector dot product. Moreover SH are used to store precomputed shadowing and inter-reflection data [SKS02]. Note that BRDFs, shadowing and inter-reflections at a point are all functions defined on a hemisphere, whereas the SH basis functions are defined on the whole sphere. Hence a larger number of SH coefficients is required for accurate representation.

Sloan et al. [SHHS03] proposed a solution to the domain mismatch by padding a hemispherical function with a mirrored copy of the function itself ("even reflection"). This process significantly improves the accuracy of the representation of a hemispherical function. However, with this approach, the SH coefficients no longer represent the original function. Problems appear when combining with a spherical environment map, since the nonzero values in the lower hemisphere yield erroneous dot product result. To distinguish this process of hemispherical function representation from the standard SH representation we will refer to it as ESH (for Even reflection Spherical Harmonics) representation.

In [SHHS03] Sloan et al. also propose a least squares optimal spherical harmonics (LSOSH) projection for hemispherical functions. Although this method improves the representation accuracy in the upper hemisphere, any hemispherical function projected using this method would contain nonzero values in the lower hemisphere. Therefore this method is only suitable for evaluation purposes, since the added nonzero values yield erroneous results when computing dot product with other LSOSH-projected functions. Consequently this method will not be further detailed.

Spherical wavelets. Wavelets are well known for efficient representation of functions defined on a plane. Schröder and Sweldens [SS95] extended wavelets to the spherical domain for efficient representation of spherical functions. They demonstrated the usefulness of spherical wavelets to accurately represent hemispherical functions, particularly BRDFs. Wavelets are advantageous especially for functions of complex shape. But using a small number of wavelet coefficients to approximate a smooth function might lead to aliasing (see Figure 13 in [SS95]). Aliasing affects the visual quality of the generated images more than an incorrect, although smooth, BRDF shape obtained by spherical harmonics or our new basis. Moreover the hierarchical data structure proposed in [SS95] is not well suited for current graphics hardware.

Zernike polynomials. Zernike polynomials [WC92] are basis functions defined on a disc. Those functions have been adapted by Koenderink et al. [KvDS96] to build a hemispherical basis. The CUReT BRDF database [DvGNK99] uses these basis functions for representing measured BRDFs. However, Zernike polynomials have a high evaluation cost (up to 10 times more than associated Legendre polynomials used for spherical harmonics), and rotation matrices are not available for them.

Makhotkin hemispherical harmonics These basis functions [Mak96, Sme98] are derived from shifted adjoint Jacobi polynomials. The basis functions can be evaluated quickly using a simple recurrence relation, but this representation lacks rotation matrices.

In this paper we propose a new basis defined directly on the hemisphere. Representation of hemispherical functions using this basis is natural and we demonstrate that such a representation is efficient as well.

3. A Novel Hemispherical Basis

In this section, we define a hemispherical basis as an adapted version of SH that relies on the associated Legendre polynomials. In particular, we propose to shift the associated Legendre polynomials to derive our new basis.

Orthogonal Polynomials. A set of polynomials $\{p_l(x)\}$ are said to be orthogonal over an interval $[a,b]$ if $p_l(x)$ is a polynomial of degree l, and for $n,m \geq 0$

$$\int_a^b w(x)p_n(x)p_m(x)\mathrm{d}x = \delta_{nm}c_n \qquad (1)$$

where δ_{nm} is 0 or 1 according as $n \neq m$ or $n = m$, and $w(x)$ is a non negative weighting function. If $c_n = 1$ then the polynomials are orthonormal.

Shifted Polynomials. We define shifting as a linear transformation of x to $k_1x + k_2$, where $k_1 \neq 0$. If the polynomials $\{p_l(x)\}$ are orthogonal over an interval $[a,b]$ with weight function $w(x)$, then the polynomials $\{p_l(k_1x+k_2)\}$ are orthogonal over the interval $[\frac{a-k_2}{k_1}, \frac{b-k_2}{k_1}]$ with the weighting function $w(k_1x+k_2)$ [Sze75]. If $\{p_l(x)\}$ are orthonormal then $\{(\text{sign } k_1)^l\sqrt{|k_1|}p_l(k_1x+k_2)\}$ are orthonormal. The polynomials $\{p_l(k_1x+k_2)\}$ are said to be the shifted versions of $\{p_l(x)\}$.

Associated Legendre Polynomials. The associated Legendre polynomials (or ALP from now on), $\{P_l^m(x)\}$ where $m \in \{0,..,l\}$, are a set of polynomials orthogonal over $[-1,+1]$ with respect to l with the weighting function $w(x) = 1$ [Wei04]:

$$\int_{-1}^{1} P_l^m(x)P_{l'}^m(x)\mathrm{d}x = \frac{2(m+l)!}{(2l+1)(l-m)!}\delta_{ll'}. \qquad (2)$$

Replacing the argument x with $\cos\theta$, we get a set of functions, $\{P_l^m(\cos\theta)\}$ defined over the angular interval $[0,\pi]$. Real valued SH functions, orthogonal over $[0,\pi]\times[0,2\pi)$, are constructed from $P_l^m(\cos\theta)$ as:

$$Y_l^m(\theta,\phi) = \begin{cases} \sqrt{2}K_l^m \cos(m\phi)P_l^m(\cos\theta) & \text{if } m>0 \\ \sqrt{2}K_l^m \sin(-m\phi)P_l^{-m}(\cos\theta) & \text{if } m<0 \\ K_l^0 P_l^0(\cos\theta) & \text{if } m=0 \end{cases} \qquad (3)$$

where K_l^m is the SH normalization value:

$$K_l^m = \sqrt{\frac{(2l+1)(l-|m|)!}{4\pi(l+|m|)!}} \qquad (4)$$

Shifted ALPs. Using the linear transformation of x to $2x-1$ we get shifted ALPs [Wei04] over the interval $x \in [0,1]$:

$$\widetilde{P}_l^m(x) \;=\; P_l^m(2x-1) \qquad (5)$$

The rationale behind this shifting is that by replacing the argument x with $\cos\theta$, we get functions $\{\widetilde{P}_l^m(\cos\theta)\}$ that are defined in the interval $\theta \in [0,\frac{\pi}{2}]$, the polar angle range of the hemisphere:

$$\widetilde{P}_l^m(\cos\theta) \;=\; P_l^m(2\cos\theta-1) \text{ with } \theta\in[0,\tfrac{\pi}{2}]. \qquad (6)$$

The shifted ALPs remain orthogonal, but the normalization changes according to the definition given above. The following equation shows the orthogonal relation with respect to l with 1 as the weighting function.

$$\begin{aligned}\int_0^1 \widetilde{P}_l^m(x)\widetilde{P}_{l'}^m(x)dx &= \int_0^1 P_l^m(2x-1)P_{l'}^m(2x-1)\mathrm{d}x \\ &= \frac{(m+l)!}{(2l+1)(l-m)!}\delta_{ll'}\end{aligned} \qquad (7)$$

From Shifted ALPs to Hemispherical Basis. In the same way that SH functions are constructed from ALPs, we construct our hemispherical basis functions $\{H_l^m(\theta,\phi)\}$ from shifted ALPs.

$$H_l^m(\theta,\phi) = \begin{cases} \sqrt{2}\widetilde{K}_l^m \cos(m\phi)\widetilde{P}_l^m(\cos\theta) & \text{if } m>0 \\ \sqrt{2}\widetilde{K}_l^m \sin(-m\phi)\widetilde{P}_l^{-m}(\cos\theta) & \text{if } m<0 \\ \widetilde{K}_l^0 \widetilde{P}_l^0(\cos\theta) & \text{if } m=0 \end{cases} \qquad (8)$$

with the following normalization value:

$$\widetilde{K}_l^m = \sqrt{\frac{(2l+1)(l-|m|)!}{2\pi(l+|m|)!}} \qquad (9)$$

By analogy with spherical harmonics, we name the $H_l^m(\theta,\phi)$ basis functions “hemispherical harmonics” (abbreviated by HSH in the remainder of this document). HSH are orthogonal over $[0,\frac{\pi}{2}]\times[0,2\pi)$ with respect to both l and m. 3D plots of the first few HSH functions are given in Appendix A. The following section presents rotation methods for HSH-projected functions.

4. HSH Rotation

In the context of realistic rendering using HSH, one might have to rotate HSH-projected functions efficiently (Section 6.2). This section contains three methods dealing with such rotations: by converting to spherical harmonics, using spherical harmonics rotation matrices, and precomputing rotation matrices.

The first method works as follows : in section 6.1 we define a matrix **C** used to convert coefficients from SH to HSH and vice-versa. The HSH rotation can be carried out by converting the HSH coefficients to SH using **C**, then applying a SH rotation matrix, and finally convert the SH coefficients back to HSH using $\mathbf{C}^{-1}$. Since the SH representation is less accurate than HSH, we propose to use a non-square conversion matrix: the intermediate SH representation uses more coefficients than HSH. It must be noted that this method automatically removes the data that disappears underneath the horizon during rotation (Figure 2(c)).

The two next methods rely on Euler's rotation theorem: any rotation may be described using three angles. We represent our rotations by the ZYZ convention, where Z is the vertical axis of a right handed orthogonal coordinate system.

The rotation of HSH functions around the Z axis is the same as that of SH functions [SAWG91]. The rotation around the Y axis is lossy for all hemispherical functions: after such a rotation, a part of the function must be set to zero (Figure 2). Hence we carry out the Y axis rotation in two steps.

In the first step we delete the hemisphere digon I (Figure 1) that disappears after the rotation around axis Y by angle β (Figure 2(b)). We carry out this deletion as a coefficient vector transformation with a matrix $\mathbf{M}(\beta)$. The elements of this matrix are defined as:

$$\mathbf{M}_{l,l'}^{m,m'}(\beta) = \int_{H-I} H_l^m(\theta,\phi) H_{l'}^{m'}(\theta,\phi) \sin\theta \mathrm{d}\theta \mathrm{d}\phi \qquad (10)$$

where $H - I$ is the hemisphere after the removal of the digon. As the digon deletion adds high frequencies in the signal, this matrix is dense. For practical purposes, we precompute matrices for a set of rotation angles, then linearly interpolate them for any other angle at runtime.

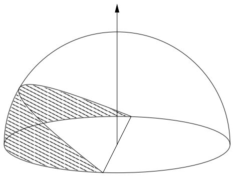


Figure 1: *Hemisphere and a digon [Dutre 2003]. The digon is the shaded part of the hemisphere.*

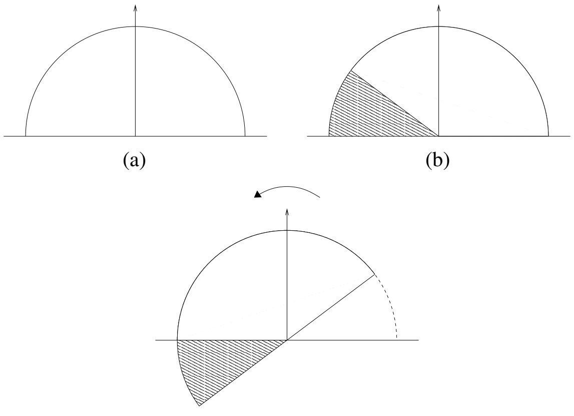


Figure 2: *HSH rotation around Y axis. Hemisphere before rotation (a), shaded digon that is set to zero (b), Hemisphere after rotation (c).*

Once this "deletion" transformation is applied, we have to compute and apply a rotation matrix for rotation around Y. Since HSH definition is close to SH, we chose to use SH rotation matrices [IR96] to derive HSH rotations.

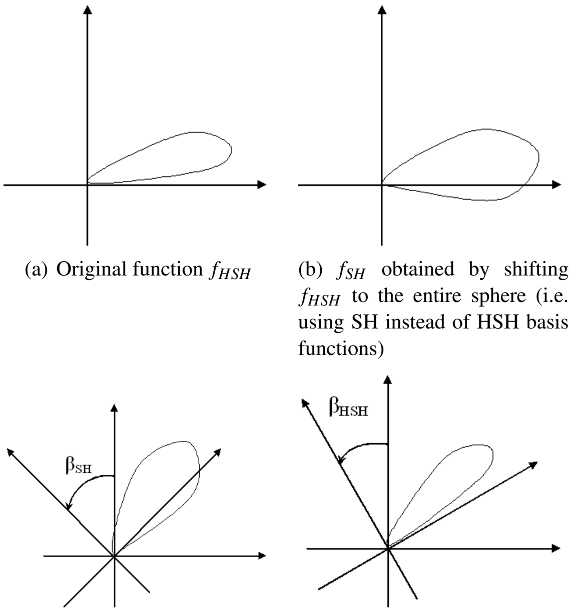
Applying a SH rotation matrix of angle β_{SH} around Y axis to a HSH vector of coefficients $\{f_i\}$ is equivalent to:

- shifting the HSH-projected function f_{HSH} to the whole sphere (Eqs. 11, 12), yielding f_{SH} (Figure 3(b)).

$$f_{HSH}(\theta,\phi) = \sum_i f_i H_i(\theta,\phi) \quad [0,\frac{\pi}{2}] \times [0,2\pi] \to \mathbf{R} \quad (11)$$

$$f_{SH}(\theta,\phi) = \sum_i f_i Y_i(\theta,\phi) \quad [0,\pi] \times [0,2\pi] \to \mathbf{R} \quad (12)$$

- applying a SH rotation matrix $R_{\beta_{SH}}^{SH}$ to f_{SH} (Figure 3(c))
- shifting $R_{\beta_{SH}}^{SH}(f_{SH})$ to the upper hemisphere to obtain $R_{\beta_{HSH}}^{HSH}(f_{HSH})$ (Figure 3(d))



(a) Original function f_{HSH}

(b) f_{SH} obtained by shifting f_{HSH} to the entire sphere (i.e. using SH instead of HSH basis functions)

(c) Spherical harmonics rotation applied to f_{SH}

(d) The rotated hemispherical function is obtained by shifting $R_{\beta_{SH}}^{SH}(f_{SH})$ to the upper hemisphere

Figure 3: *HSH rotation process : the original hemispherical function is first shifted to the whole sphere, then rotated using SH rotation matrices. The last step shifts the rotated function back to the hemisphere.*

Our aim is to determine the relation between β_{SH} and β_{HSH} : since SH and HSH domains are different, $\beta_{SH} \neq \beta_{HSH}$. Using Equation 6, a rotation of angle β_{HSH} on the hemisphere is equivalent to a rotation of angle β_{HSH} (Eq. 13).

$$\beta_{SH} = \arccos(2\cos\beta_{HSH} - 1) \qquad (13)$$

This equation shows that the SH rotation matrices can be applied to HSH for a rotation around Y axis provided that no part of the original function moves into the lower hemisphere after rotation (this condition is fulfilled by removing the considered digon (Eq. 10)).

The last rotation method consists of the inclusion of the rotation in the precomputed matrices in Equation 10. The matrix element values are defined by

$$\mathbf{M}_{\mathbf{R}_{l,l'}}^{m,m'}(\beta) = \int_{H-I} H_l^m(\theta,\phi) H_{l'}^{m'}(R_\beta(\theta,\phi)) \sin\theta \mathrm{d}\theta \mathrm{d}\phi \qquad (14)$$

where $H - I$ is the hemisphere after removing the digon,

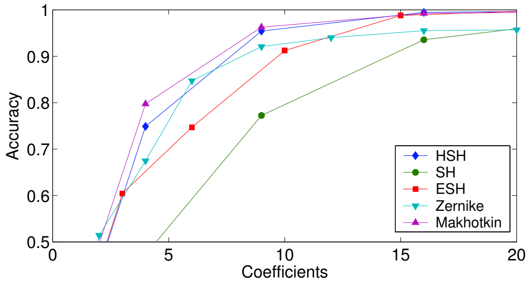


Figure 4: *Accuracy versus number of coefficients for approximation of a Phong lobe, $(\cos\alpha)^5$, using HSH, SH, ESH, Zernike and Makhotkin approaches.*

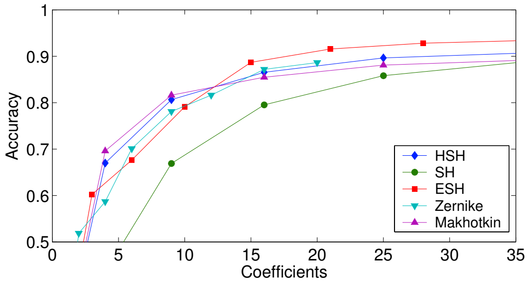


Figure 5: *Accuracy versus number of coefficients for HSH, SH, ESH, Zernike and Makhotkin representations. These values are obtained for an anisotropic Ward BRDF with $k_d = 0$, $k_s = 1$, $\alpha_x = 0.2$, $\alpha_y = 0.5$.*

and $R_\beta(\theta,\phi)$ represents the direction (θ,ϕ) rotated around Y axis by angle β.

The accuracy of this method comes at the expense of memory: an accurate rotation requires a large number of precomputed matrices. The two next sections present applications of HSH in the context of image synthesis.

5. HSH for Hemispherical Function Approximation

The $[0,\frac{\pi}{2}]\times[0,2\pi)$ domain of HSH basis functions makes them ideal for representation of hemispherical functions such as BRDFs and radiance functions around surface points. In this section we show a accuracy comparison between HSH representation and previous methods.

We first show the approximation of a Phong lobe [Pho75] $(\cos\alpha)^5$, where α is the angle between the specular reflection direction and the viewing direction. Figure 4 plots the approximation accuracy as function of the number of coefficients in HSH, SH, ESH, Zernike and Makhotkin's representation. We estimate this accuracy by computing the fraction of the total energy captured for a given number of coefficients [RH02]:

$$\text{Accuracy} = \frac{\sum_{l=0}^{n}\sum_{m=-l}^{l}|f_l^m|^2}{\int_0^{2\pi}\int_0^{\frac{\pi}{2}} f(\theta,\phi)^2 \sin\theta \mathrm{d}\theta\mathrm{d}\phi} \qquad (15)$$

The observation in this figure and the results in later sections show that the HSH representation indeed provides a better representation than SH with fewer coefficients. Moreover, our basis performs comparably to Makhotkin's method, and exhibits a higher accuracy than both ESH and Zernike approaches in the Phong case.

HSH Representation of BRDFs. BRDFs are functions defined over the cartesian product of two hemispheres, corresponding to incoming and outgoing directions. One can use either of the following two approaches to represent them using HSH basis. One approach is to use two successive HSH transformations, one for the incoming hemisphere and the other for the outgoing hemisphere (similar to the approach used by Westin et al. [WAT92] using SH basis). The other approach is to discretize the outgoing hemisphere and use HSH to represent the incoming hemisphere corresponding to each outgoing direction (similar to the approach used by Kautz et al. [KSS02]).

We use the latter approach, in which the n-th order representation of a BRDF $f_{(\theta_o,\phi_o)}$ for any given outgoing sample direction (θ_o,ϕ_o) is

$$f_{(\theta_o,\phi_o)}(\theta_i,\phi_i) \approx \sum_{l=0}^{n}\sum_{m=-l}^{l} c_l^m(\theta_o,\phi_o)H_l^m(\theta_i,\phi_i)\ , \qquad (16)$$

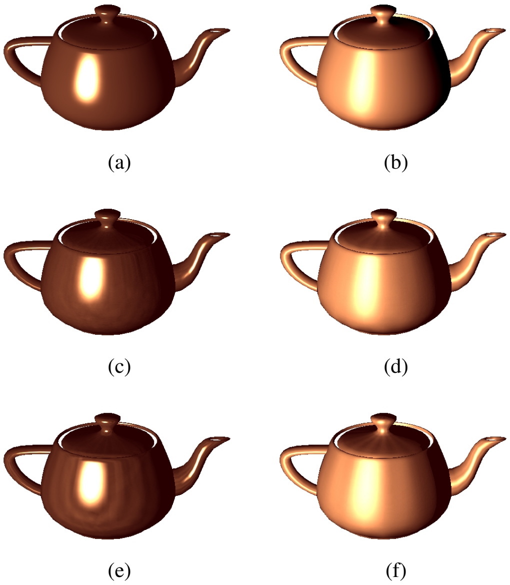
where

$$c_l^m(\theta_o,\phi_o) = \int_0^{2\pi}\int_0^{\pi/2} f(\theta_o,\phi_o,\theta_i,\phi_i)H_l^m(\theta_i,\phi_i)\sin\theta_i \mathrm{d}\theta_i\mathrm{d}\phi_i\ . \qquad (17)$$

We sample the outgoing hemisphere for (θ_o,ϕ_o) using parabolic parametrization proposed by Heidrich and Seidel [HS99].

Figure 5 plots the approximation accuracy of the HSH, SH, ESH, Zernike and Makhotkin's representation of an anisotropic Ward BRDF [War92] as a function of the number of coefficients. The plot shows that although the ESH provides better results in this case, the HSH representation is still more accurate than the SH representation when the same number of coefficients is used. As in the previous example, HSH, Zernike and Makhotkin's method perform comparably.

Figure 6 shows image frames from our GPU based realtime renderer. The teapots shown in the images are lit by a single point light source. Those in the left column are assigned a Phong BRDF, and in the right column an anisotropic Ward BRDF. The renderings in the three rows from top to bottom correspond to analytic BRDF functions, and their HSH and SH representations respectively. 32 × 32 outgoing sample directions are used for the HSH and SH representations. The coefficients and the basis function values are stored as textures. For any incoming direction (θ_i, ϕ_i) and outgoing direction (θ_o, ϕ_o), $c_l^m(\theta_o, \phi_o)$ and $H_l^m(\theta_i, \phi_i)$ values are computed by bilinear texture interpolation. Equation 16 for $f_{(\theta_o,\phi_o)}(\theta_i, \phi_i)$ is evaluated using a multi-pass pixel shader program. The images using HSH representation (Figures 6(c) and (d)) are visually closer to the images obtained using analytical BRDFs. Note the ringing in the images obtained using SH representation of the BRDF (Figures 6(e) and (f)).


(a) (b)

(c) (d)

(e) (f)

Figure 6: *Images rendered using Phong BRDF (left column) and anisotropic Ward BRDFs (right column). Figures (a) and (b) use analytic BRDFs; Figures (c) and (d) use order 7 HSH representation and Figures (e) and (f) use order 7 SH representation.*

Figures 5 and 6 demonstrate that HSH representation of BRDF using a given number of coefficients outperforms SH representation both in terms of computed and visual accuracy.

6. HSH in Realistic Rendering

6.1. Environment Map Rendering

Hardware rendering of surfaces with arbitrary BRDFs illuminated by environment maps has recently received wide attention [KSS02, RH02, SKS02]. Environment maps are discrete representations of incoming radiance from directions defined on a whole sphere. Consequently they can be approximated using a spherical basis such as SH. In this section, we show that a SH representation of environment maps is straightforward to combine with a HSH representation of BRDFs. We demonstrate interactive hardware rendering using HSH representation for BRDFs and SH representation for environment maps.

Since SH and HSH bases are not mutually orthogonal, we cannot directly compute an integral containing a SH representation of an environment map and a HSH representation of a BRDF. However, incoming light from the environment at any surface point is only defined on a hemisphere. Consequently, we propose to convert the environment map coefficients from SH to HSH after rotating them from the global to the local frame. This conversion is achieved by transforming the rotated SH coefficients with a basis change matrix **C**. Each element of this matrix is defined by:

$$\mathbf{C}_{l,l'}^{m,m'} = \int_0^{2\pi} \int_0^{\pi/2} H_l^m(\theta,\phi) Y_{l'}^{m'}(\theta,\phi) \sin\theta \mathrm{d}\theta \mathrm{d}\phi \qquad (18)$$

This matrix is very sparse, nonzero values occur only for $m = m'$, which significantly reduces the cost of the transformation from SH to HSH representation. Note that this transformation is not invertible since the integration in the interval $\theta \in [0, \pi/2]$ only accounts for the information belonging to one half of the sphere, and completely discards the information in the other half. For $l < 3$ and $l' < 3$ the conversion matrix $\mathbf{C}_{3,3}$ is:

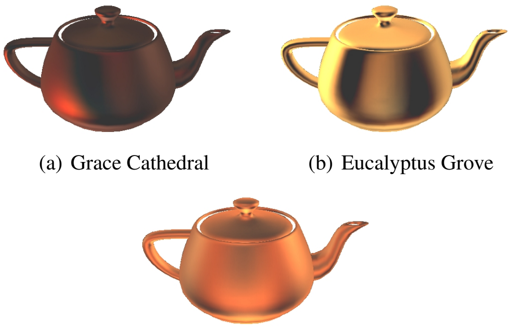
$$\begin{pmatrix}
0.71 & 0 & 0.41 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0.60 & 0 & 0 & 0 & 0.19 & 0 & 0 & 0 \\
0.82 & 0 & 0.71 & 0 & 0 & 0 & 0.18 & 0 & 0 \\
0 & 0 & 0 & 0.60 & 0 & 0 & 0 & 0.19 & 0 \\
0 & 0 & 0 & 0 & 0.53 & 0 & 0 & 0 & 0 \\
0 & 0.77 & 0 & 0 & 0 & 0.51 & 0 & 0 & 0 \\
0.40 & 0 & 0.78 & 0 & 0 & 0 & 0.53 & 0 & 0 \\
0 & 0 & 0 & 0.77 & 0 & 0 & 0 & 0.51 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.53
\end{pmatrix}$$

Rendering with environment maps proceeds as follows:

- Compute SH representation of the environment map.
- Compute HSH representation of the surface BRDF for every surface in the scene
- For every rendered surface point:
  - Rotate the coefficients of the environment map to fit the local frame associated with the surface point.
  - Transform the coefficients from SH to HSH using the precomputed basis change matrix **C**.
  - Compute the dot product between the HSH coefficients of the environment map and HSH coefficients of the surface BRDF for the required outgoing direction.

Figure 7 shows renderings of a teapot using three different environment maps. Table 1 compares frame rates achieved by the HSH approximated BRDF against the SH approximated BRDF. Our method achieves similar frame rates to the SH approach for the same number of coefficients since the conversion step does not require much additional computation.

It should be noted that the ESH representation can not be used in this context because of the extra non-zero BRDF



(a) Grace Cathedral

(b) Eucalyptus Grove

(c) St Peter's Basilica

Figure 7: *Environment map rendering using HSH representation of the BRDF (order 10). The glossy teapot is lit by three different environment maps.*

information present in the lower hemisphere: the dot product between the environment map and the BRDF would take into account both upper and lower hemispheres, leading to erroneous results.

Order of representation (n)	2	3	5	8	10
HSH (fps)	220	138	26.2	3.1	1.6
SH (fps)	223	160	29	3.28	1.73

Table 1: *Timing results (in frames per second) for environment maps rendering using HSH and SH representation of BRDFs. The timing was obtained on a 2.4GHz Intel Xeon with ATI Radeon 9800 Pro card for an image size 700x600. The teapot model used has 1873 vertices.*

6.2. Radiance Caching for Efficient Global Illumination Computation

Monte Carlo (MC) ray tracing remains the method of choice for accurately computing global illumination. It is, however, very expensive when it comes to computing illumination on surfaces with low frequency BRDFs. Many sample rays have to be traced to get a reasonable estimate of the following illumination integral at a point.

$$L(\theta_o,\phi_o) = \int_0^{2\pi}\int_0^{\pi/2} L_i(\theta_i,\phi_i) f(\theta_i,\phi_i,\theta_o,\phi_o) \cos\theta_i \sin\theta_i \mathrm{d}\theta_i \mathrm{d}\phi_i \tag{19}$$

where L is the outgoing radiance, L_i is the incoming radiance and f is the BRDF. Various algorithms have been proposed to reduce the cost of this computation. Ward et al.'s [WRC88] irradiance caching is one of such algorithms. It significantly reduces the amortized cost of the illumination integral estimation at Lambertian surfaces by caching and reusing irradiance.

```
for (every BRDF in the scene) do
    - Compute the HSH representation of the BRDF.
end for
for (every surface point $\vec{P}$ visible through the image pixels) do
    if ( radiance cache samples exist near $\vec{P}$ ) then
        - Compute HSH coefficients of the incoming radiance at $\vec{P}$ by gradient based interpolation.
    else
        - Monte Carlo sample the incoming hemisphere around $\vec{P}$ and get the incoming radiance along a number of incoming directions
        - Compute HSH coefficients of the incoming radiance function at $\vec{P}$ by Monte Carlo quadrature.
        - Compute the gradient of the coefficients at $\vec{P}$.
        - Store the coefficients and the gradients in the radiance cache.
    end if
    - Compute the outgoing radiance at $\vec{P}$ as the dot product of the HSH coefficients of incoming radiance and surface BRDF.
end for
```

Figure 8: *Outline of our radiance caching algorithm.*

We extend this algorithm to support non-Lambertian surfaces by caching and reusing the incoming radiance function instead of irradiance. The incoming radiance is represented by hemispherical harmonics.

Figure 8 briefly outlines the steps of our radiance caching algorithm. They are similar to those used by Ward et al. [WRC88] in their irradiance caching. We make use of the same basic observation that the indirect illumination changes slowly over each surface with low-frequency BRDFs. The main difference is that we cache the directional radiance function instead of scalar irradiance. This allows us to compute outgoing radiance from surfaces with non-Lambertian BRDFs.

Though this method can support all types of BRDFs, we would like to point out that the accurate representation of sharp BRDFs requires a large number of HSH coefficients. Hence, for high frequency BRDFs, direct computation of the outgoing radiance by MC importance sampling is likely to be more efficient than interpolation based computation. The various steps of the algorithm shown in Figure 8 are described in the following paragraphs. For a more detailed description of radiance caching please refer to [KGPB04].

BRDF Representation. We precompute the HSH representation of surface BRDFs using the method described in Section 5. Note that we *premultiply* BRDFs by the cosine term,

$\cos\theta_i$, of the illumination integral (Eq. 19) before computing their HSH representation.

Incoming Radiance Computation. We represent the incoming radiance function L_i at a point $\vec{P}$ by a vector of HSH coefficients λ_l^m:

$$L_i(\theta,\phi) \approx \sum_{l=0}^{n} \sum_{m=-l}^{l} \lambda_l^m H_l^m(\theta,\phi), \qquad (20)$$

where n is the order of HSH representation and

$$\lambda_l^m = \int_0^{2\pi} \int_0^{\pi/2} L_i(\theta,\phi) H_l^m(\theta,\phi) \sin\theta \mathrm{d}\theta \mathrm{d}\phi. \qquad (21)$$

We compute λ_l^m by MC quadrature with uniform sampling of the hemisphere of incoming directions. The equation for this quadrature is

$$\lambda_l^m = \frac{2\pi}{N} \sum_{k=1}^{N} L_i(\theta_k,\phi_k) H_l^m(\theta_k,\phi_k), \qquad (22)$$

where $L_i(\theta_k,\phi_k)$ is the incoming radiance coming from the sampled direction (θ_k,ϕ_k) and N is the number of sample directions. We use a fixed number N of sample directions.

Radiance Gradient computation. There are two major differences between the gradient based interpolation used here and by Ward and Heckbert [WH92].

- We rotate the radiance functions before interpolation, hence we do not use any rotational gradient.
- We use uniform sampling for computing λ_l^m, whereas Ward and Heckbert [WH92] use cosine sampling for computing irradiance. Hence our translational gradient computation is different.

We compute the translational gradient $\nabla\lambda_l^m = \left[\frac{\partial\lambda_l^m}{\partial x}, \frac{\partial\lambda_l^m}{\partial y}, 0\right]$ for each λ_l^m. The translational gradient is defined in the local coordinate frame at a sample point $\vec{P}$. The derivative $\partial\lambda_l^m/\partial x$ at $\vec{P}$ is computed by displacing the sample point $\vec{P}$ along the local X axis by Δx to $\vec{P}'$. The coefficient $\lambda_l^{m\prime}$ at $\vec{P}'$ is computed as

$$\lambda_l^{m\prime} = \frac{2\pi}{N} \sum_{k=1}^{N} \frac{\|\vec{W}_k - \vec{P}\|^2}{\|\vec{W}_k - \vec{P}'\|^2} \frac{\cos\xi'}{\cos\xi} L_i(\theta_k,\phi_k)\, H_l^m\left(\theta_k',\phi_k'\right),$$

and $\partial\lambda_l^m/\partial x$ is computed as

$$\frac{\partial\lambda_l^m}{\partial x} = \frac{\lambda_l^{m\prime} - \lambda_l^m}{\Delta x},$$

where the denotation is:

(θ_k,ϕ_k)	Direction of the k-th ray used to sample the hemisphere at $\vec{P}$.
$L_i(\theta_k,\phi_k)$	Incoming radiance from the direction (θ_k,ϕ_k).
$\vec{W}_k$	Ray - surface intersection point for direction (θ_k,ϕ_k).
ξ	Angle between the vector $\vec{W}_k - \vec{P}$ and the surface normal at $\vec{W}_k$.
(θ_k',ϕ_k')	Direction from $\vec{P}'$ to $\vec{W}_k$.
ξ'	Angle between the vector $\vec{W}_k - \vec{P}'$ and the surface normal at $\vec{W}_k$.

The computation of $\partial\lambda_l^m/\partial y$ proceeds in a similar way. The derivation of those formulas is given in [KGPB04].

Radiance Interpolation. We use a weighted interpolation scheme similar to the one proposed in [WH92] for interpolating coefficients λ_l^m at any required surface point $\vec{P}$. The only difference in the schemes is that we replace the use of the rotational gradient by applying a rotation to the vector of coefficients $\Lambda_i = \{\lambda_{l,i}^m\}$. This aligns the coordinate frame at the sample point $\vec{P}_i$ with the frame at $\vec{P}$. The weight $w_i(\vec{P})$ of sample $\vec{P}_i$ is

$$w_i(\vec{P}) = \left(\frac{\|\vec{P} - \vec{P}_i\|}{R_i} + \sqrt{1 - \vec{N}\cdot\vec{N}_i} \right)^{-1}, \qquad (23)$$

where $\vec{N}$ is the surface normal at $\vec{P}$, $\vec{N}_i$ is the surface normal at $\vec{P}_i$, and R_i is the harmonic mean distance to objects visible from $\vec{P}_i$. The coefficient vector $\Lambda = \{\lambda_l^m\}$ of the interpolated radiance is:

$$\Lambda(\vec{P}) = \frac{\sum_{\mathrm{S}} \mathbf{R}_i \left(\Lambda_i + d_x \frac{\partial\Lambda_i}{\partial x} + d_y \frac{\partial\Lambda_i}{\partial y} \right) w_i(\vec{P})}{\sum_{\mathrm{S}} w_i(\vec{P})}, \qquad (24)$$

where $S = \{i | w_i(\vec{P}) > 1/a\}$ and a is a user defined desired accuracy. The definition of the set S is the criterion used to decide which radiance cache samples can be used for interpolation. Derivatives $\partial\Lambda_i/\partial x$ and $\partial\Lambda_i/\partial y$ are the translational gradient components and (d_x, d_y) are the displacements of $\vec{P} - \vec{P}_i$ along the X and Y axes of the sample i's local coordinate frame. $\mathbf{R}_i$ is the HSH rotation (Section 4) that aligns the coordinate frame at $\vec{P}_i$ with the frame at $\vec{P}$. It transforms the whole coefficient vector $\Lambda_i = \{\lambda_{l,i}^m\}$ after the translational gradient is applied.

Outgoing Radiance Computation. As both the incoming radiance L_i and the BRDF f in the illumination integral (Eq. 19) are now represented as HSH coefficient vectors, the integral computation reduces to the dot product

$$L(\theta_o,\phi_o) = \sum_{l=0}^{n-1} \sum_{m=-l}^{l} \lambda_l^m f_l^m(\theta_o,\phi_o),$$

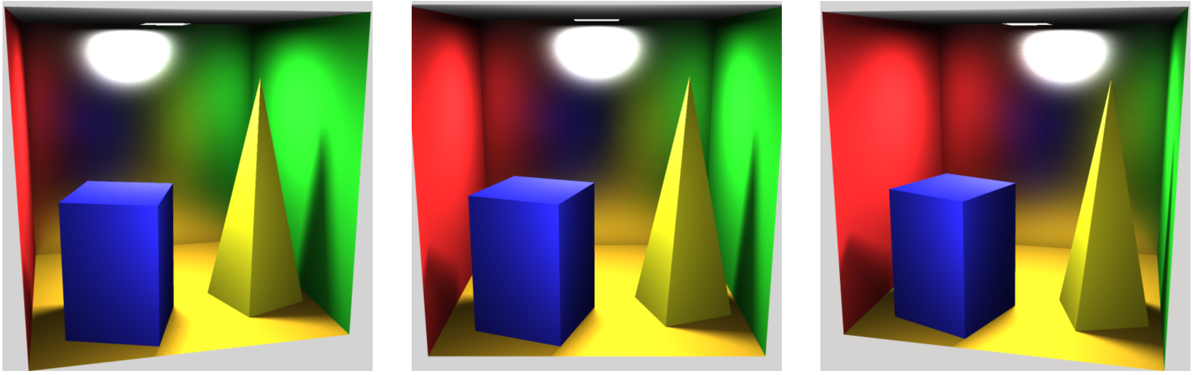


Figure 9: *Cornell box with glossy back wall rendered using radiance caching. Radiance function at sample points are computed as its HSH coefficients. 1121 radiance samples are used to render the center view (b). The number of additional samples needed when the viewpoint changes to left (a) and right (c) is 250 and 198. The size of each of the images is* 850×850.

where λ_l^m is an interpolated incoming radiance coefficient and $f_l^m(\theta_o, \phi_o)$ is a BRDF coefficient at $\vec{P}$ for the outgoing direction (θ_o, ϕ_o).

Results. Figure 9 shows three renderings of a Cornell box with glossy back wall (Phong, exponent 22), taken from three different viewpoints at resolution 850×850. The number of rays cast to sample each hemisphere is $N = 4000$. The order of HSH used is $n = 10$. Except for the back wall, all objects are Lambertian. Direct lighting and first bounce reflection for the back wall were used to generate the images. The timings given here do not include direct illumination computation. Image 9(b) took 25.8 sec. to render[†] with 1121 radiance cache samples. Images 9(a) and (c) took 11.2 and 10.3 seconds to render since only 250 and 198 additional radiance cache samples were needed. Memory consumption of the radiance cache was 1.5 MB. The timings compare well with 260 seconds rendering time for the images of approximately the same visual quality generated with MC importance sampling.

7. Conclusion

This paper described a new basis for hemispherical data representation. This basis is ideal for the representation of low-frequency hemispherical functions. These functions are often encountered in many applications where the interaction between a point on a surface and its environment needs to be computed. We demonstrated its use for the following representative sets of applications: efficient representation of BRDFs, real-time environment map rendering of non-diffuse surfaces and efficient global-illumination computation. As shown, this basis provides a more compact and accurate way of representing hemispherical functions than spherical harmonics.

Acknowledgments

We would like to thank Jaakko Konttinen for developing our hardware-accelerated application and making the videos. Thanks also to Daniel Meneveaux for proofreading early versions of this paper and providing helpful comments and suggestions. This work was partially supported by ATI Research, I-4 Matching fund and Office of Naval Research.

References

[CMS87] Cabral B., Max N., Springmeyer R.: Bidirectional reflection functions from surface bump maps. In *Proceedings of SIGGRAPH* (1987), ACM Press, pp. 273–281.

[Dut03] Dutre P.: Global illumination compendium. http://www.cs.kuleuven.ac.be/ phil/GI/.

[DvGNK99] Dana K. J., van Ginneken B., Nayar S. K., Koenderink J. J.: Reflectance and texture of real-world surfaces. *ACM Transactions on Graphics 18*, 1 (1999), 1–34.

[HS99] Heidrich W., Seidel H.-P.: Realistic, hardware-accelerated shading and lighting. In *Proceedings of SIGGRAPH* (1999), ACM Press/Addison-Wesley Publishing Co., pp. 171–178.

[IR96] Ivanic J., Ruedenberg K.: Rotation matrices for real spherical harmonics. direct determination by recursion. *J. Phys. Chem. 100*, 15 (1996), 6342–6347.

[KGPB04] Křivánek J., Gautron P., Pattanaik S., Bouatouch K.: *Radiance Caching for Efficient Global Illumination Computation*. Tech. Rep. 1623, IRISA, Rennes, France, June 2004.

[KSS02] Kautz J., Sloan P.-P., Snyder J.: Fast, arbitrary BRDF shading for low-frequency lighting using spherical harmonics. In *Proceedings of the 13th*

[†] The timings in this section were measured on a 2.26GHz Pentium 4 with 512MB RAM running Windows XP.

Eurographics workshop on Rendering (2002), Eurographics Association, pp. 291–296.

[KvDS96] KOENDERINK J., VAN DOORN A., STAVRIDI M.: Bidirectional reflection distribution function expressed in terms of surface scattering modes. *ECCV B* (1996), 28–39.

[Mak96] MAKHOTKIN O. A.: Analysis of radiative transfer between surfaces by hemispherical harmonics. *Journal of Quantitative Spectroscopy and Radiative Transfer 56*, 6 (1996), 869–879.

[Pho75] PHONG B. T.: Illumination for computer generated pictures. *Commun. ACM 18*, 6 (1975), 311–317.

[RH02] RAMAMOORTHI R., HANRAHAN P.: Frequency space environment map rendering. In *Proceedings of SIGGRAPH* (2002), ACM Press, pp. 517–526.

[SAWG91] SILLION F. X., ARVO J. R., WESTIN S. H., GREENBERG D. P.: A global illumination solution for general reflectance distributions. In *Proceedings of SIGGRAPH* (1991), ACM Press, pp. 187–196.

[SHHS03] SLOAN P.-P., HALL J., HART J., SNYDER J.: Clustered principal components for precomputed radiance transfer. *ACM Trans. Graph. 22*, 3 (2003), 382–391.

[SKS02] SLOAN P.-P., KAUTZ J., SNYDER J.: Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. *ACM Trans. Graph. 21*, 3 (2002), 527–536.

[Sme98] SMELOV V. V.: On completeness of semispherical harmonics system. *Siberian Journal of Mathematics 1*, 4 (1998), 391–395.

[SS95] SCHRÖDER P., SWELDENS W.: Spherical wavelets: efficiently representing functions on the sphere. In *Proceedings of SIGGRAPH* (1995), ACM Press, pp. 161–172.

[Sze75] SZEGÖ G.: *Orthogonal polynomials*, 4 ed. American Mathematical Society, Providence, Rhode Island, 1975, p. page 29.

[War92] WARD G. J.: Measuring and modeling anisotropic reflection. In *Proceedings of SIGGRAPH* (1992), ACM Press, pp. 265–272.

[War94] WARD G. J.: The radiance lighting simulation and rendering system. In *Proceedings of SIGGRAPH* (1994), ACM Press, pp. 459–472.

[WAT92] WESTIN S. H., ARVO J. R., TORRANCE K. E.: Predicting reflectance functions from complex surfaces. In *Proceedings of SIGGRAPH* (1992), ACM Press, pp. 255–264.

[WC92] WYANT J. C., CREATH K.: Basic wavefront aberration theory for optical metrology. In *Applied optics and Optical Engineering, Vol XI* (1992), Academic Press, Inc., pp. 27–39.

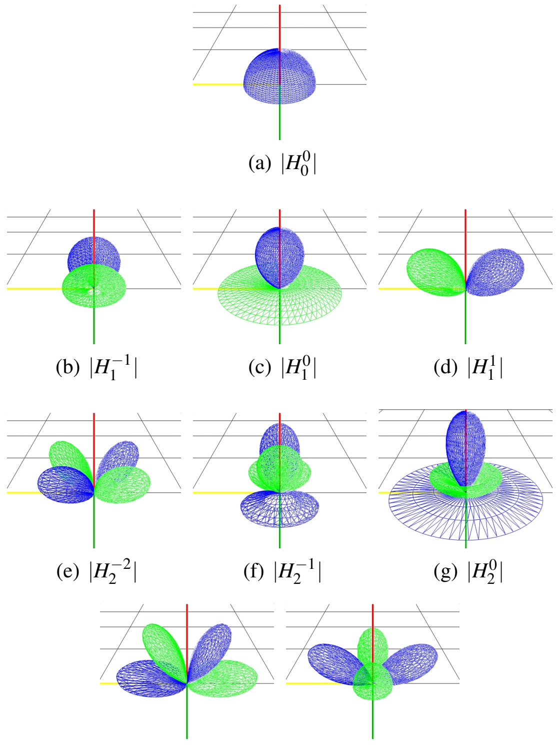
[Wei04] WEISSTEIN E.: *World of Mathematics: A Wolfram Web Resource*. http://mathworld.wolfram.com/LegendrePolynomial.html, 2004.

[WH92] WARD G. J., HECKBERT P. S.: Irradiance gradients. In *Proceedings of 2nd Eurographics Workshop on Rendering* (Bristol, 1992).

[WRC88] WARD G. J., RUBINSTEIN F. M., CLEAR R. D.: A ray tracing solution for diffuse interreflection. In *Proceedings of SIGGRAPH* (1988), ACM Press, pp. 85–92.

Appendix A: HSH Function Plots

In Figure 10 we show the shape of the first few HSH functions.



(a) $|H_0^0|$

(b) $|H_1^{-1}|$ (c) $|H_1^0|$ (d) $|H_1^1|$

(e) $|H_2^{-2}|$ (f) $|H_2^{-1}|$ (g) $|H_2^0|$

(h) $|H_2^1|$ (i) $|H_2^2|$

Figure 10: *Function plots of H_l^m for l from 0 to 2.*

Appendix B: Associated Legendre Polynomials

The ALPs are defined as

$$\begin{aligned} P_l^m(x) &= (-1)^m(1-x^2)^{m/2}\frac{\mathrm{d}^m}{\mathrm{d}x^m}P_l(x) && (25)\\ &= \frac{(-1)^m}{2^l l!}(1-x^2)^{m/2}\frac{\mathrm{d}^{m+l}}{\mathrm{d}x^{m+l}}(x^2-1)^l && (26)\end{aligned}$$

where the $P_l(x)$ are the unassociated Legendre Polynomials. These polynomials can be computed efficiently using the following recurrence relations :

$$\begin{aligned} (l-m)P_l^m(x) &= x(2l-1)P_{l-1}^m(x) && (27)\\ &\quad -(l+m-1)P_{l-2}^m(x) \\ P_m^m(x) &= (-1)^m(2m-1)!!(1-x^2)^{m/2} && (28)\\ P_{m+1}^m(x) &= x(2m+1)P_m^m(x) && (29)\end{aligned}$$

where $n!!$ is the double factorial of n, i.e. the product of all positive integers less than or equal to n with same parity as n.

Radiance Caching for Efficient Global Illumination Computation

Jaroslav Křivánek, Pascal Gautron,
Sumanta Pattanaik, *Member*, *IEEE*, and Kadi Bouatouch, *Member*, *IEEE*

Abstract—In this paper, we present a ray tracing-based method for accelerated global illumination computation in scenes with low-frequency glossy BRDFs. The method is based on sparse sampling, caching, and interpolating radiance on glossy surfaces. In particular, we extend the irradiance caching scheme proposed by Ward et al. [1] to cache and interpolate directional incoming radiance instead of irradiance. The incoming radiance at a point is represented by a vector of coefficients with respect to a hemispherical or spherical basis. The surfaces suitable for interpolation are selected automatically according to the roughness of their BRDF. We also propose a novel method for computing translational radiance gradient at a point.

Index Terms—Global illumination, ray tracing, hemispherical harmonics, spherical harmonics, directional distribution.

1 Introduction

Monte Carlo ray tracing is the method of choice for computing images of complex environments with global illumination [2]. Even for the radiosity method, high quality images are created by final gathering, often using Monte Carlo ray tracing [3].

Monte Carlo ray tracing is, however, expensive when it comes to computing indirect illumination on surfaces with low-frequency BRDFs (bidirectional reflectance distribution functions). Too many rays have to be traced to get a reasonably precise estimate of the outgoing radiance at a point. Fortunately, a high degree of coherence in the outgoing radiance field on those surfaces [1], [4], [5], [6] can be exploited by interpolation [1], [7] to obtain a significant performance gain.

Our goal is to accelerate Monte Carlo ray tracing-based global illumination computation in the presence of surfaces with low-frequency glossy BRDFs. We achieve it through sparse sampling, caching, and interpolating radiance on those surfaces. In particular, we extend Ward et al.'s irradiance caching [1], [8] to glossy surfaces. Irradiance caching is based on the observation that reflected radiance on *diffuse* surfaces due to indirect illumination changes very slowly with position. However, this applies to surfaces with arbitrary low-frequency BRDFs. Motivated by this observation, we extend Ward et al.'s work to cache and interpolate the *directional* incident radiance instead of the irradiance. This allows us to accelerate indirect lighting computation on surfaces with glossy BRDFs. We call the new method *radiance caching*.

The incoming radiance at a point is represented by a vector of coefficients with respect to spherical or hemispherical harmonics [9]. Due to the basis orthogonality, the illumination integral evaluation (1) reduces to a dot product of the interpolated incoming radiance coefficients and the BRDF coefficients. Radiance interpolation is carried out by interpolating the coefficients. We enhance the interpolation quality by the use of translational gradients. We propose novel methods for computing gradients that are more general than the method of Ward and Heckbert [8].

Radiance caching shares all the advantages of Ward et al.'s work. Computation is concentrated on visible parts of the scene; no restrictions are imposed on the scene geometry; implementation and integration with a ray tracer is easy. Our approach can be directly used with any BRDF represented by (hemi)spherical harmonics, including measured BRDFs.

This paper extends the initial description of radiance caching given in [9]. The main contributions of this paper are the extension of irradiance caching to glossy surfaces, an automatic method for selecting BRDFs suitable for radiance caching, new methods for computing translational radiance gradient, and integration of radiance caching in a ray tracer.

The rest of the paper is organized as follows: Section 2 summarizes the related work. Section 3 gives an overview of how radiance caching works and how it is integrated in a ray tracer. Section 4 details different aspects of radiance caching. Section 5 presents the results. Section 6 discusses various topics not covered in the algorithm description. Section 7 concludes the paper and summarizes our ideas for future work.

2 Related Work

2.1 Interpolation

Interpolation can be used in global illumination whenever there is a certain level of smoothness in the quantity being computed. The radiosity method uses interpolation in the form of surface discretization, e.g., [10], [11], [12]. In the context of Monte Carlo ray tracing, approaches have been

- *J. Křivánek, P. Gautron, and K. Bouatouch are with IRISA, Campus de Beaulieu, 35042 Rennes, France. E-mail: {jkrivane, pgautron, kadi}@irisa.fr.*
- *S. Pattanaik is with the School of Computer Science, Computer Science Building, University of Central Florida, Orlando, FL 32816. E-mail: sumant@cs.ucf.edu.*

Manuscript received 26 July 2004; revised 22 Dec. 2004; accepted 17 Jan. 2005; published online 12 July 2005.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org, and reference IEEECS Log Number TVCG-0080-0704.

proposed for *screen space interpolation* [5], [13], [14], [15]. The goal of these methods is to display an approximate solution quickly. However, they do not accelerate the computation of the final high quality solution, which is the objective of our work. *Object space interpolation* has also been used for the purpose of fast previewing [16], [17]. Sparse sampling and interpolation for high quality rendering was used in [7], [1]. The approach of Bala et al. [7] is suitable only for deterministic ray tracing. Ward et al. [1] use interpolation only for diffuse surfaces. Our approach extends this work to support caching and interpolation of the directional incoming radiance on glossy surfaces.

2.2 Caching Directional Distributions

Caching directional distributions has been used to extend the radiosity method to support glossy surfaces, e.g., [18], [19], [20], [21], [22], [23], [24], [25]. It has also been used in Monte Carlo ray tracing on diffuse surfaces [6], [26], [27]. Slusallek et al. [6] and Kato [26] use reprojection of radiance samples. Tawara et al. [27] selectively update a radiance sample list in time to exploit temporal coherence. Storing light particles in the scene can also be thought of as caching a directional distribution [28], [29].

2.3 Spherical Function Representation

A representation of functions on a (hemi)sphere is necessary for incoming radiance caching. Piecewise constant representation [6], [24], [26], [27] is simple but prone to aliasing and memory demanding. Unless higher order wavelets are used, even wavelet representation [21], [22], [23], [25], [30] does not remove the aliasing problems. Nevertheless, wavelets are a viable alternative to the use of spherical or hemispherical harmonics, especially for higher frequency BRDFs.

Spherical harmonics [18], [20], [31], [32], [33], [34], [35], [36], [37] remove the aliasing problem and are efficient for representing low-frequency functions. However, representation of sharp functions requires many coefficients and ringing might appear. Hemispherical harmonics [9] are better suited for representing functions on a hemisphere. Basis functions similar to spherical harmonics are Zernike polynomials [38], [39] and the hemispherical harmonics of Makhotkin [40]. Unlike for spherical harmonics, the rotation procedure is not available for these basis functions. We choose hemispherical and spherical harmonics because they are the only basis for which an efficient rotation procedure is available. They also have good antialiasing properties, low storage cost, and are easy to use.

2.4 Illumination Gradient Computation

Arvo [41] computes the irradiance Jacobian due to partially occluded polygonal emitters of constant radiosity. Holzschuch and Sillion [42] handle polygonal emitters with arbitrary radiosity. Ward and Heckbert [8] compute irradiance gradient using the information from hemisphere sampling. Our gradient computation is also based on hemisphere sampling. We use gradients to improve the smoothness of the radiance interpolation. One of the algorithms we use for gradient computation was independently developed by Annen et al. [43].

2.5 Irradiance Caching

Ward et al. [1] propose irradiance caching as a means of computing indirect diffuse interreflections in a ray tracer [2]. They sample the irradiance sparsely over surfaces, cache the results, and interpolate them. For each ray hitting a surface, the irradiance cache is queried. If irradiance records are available, the irradiance is interpolated. Otherwise, a new irradiance record is computed by sampling the hemisphere and added to the cache. In [8], the interpolation quality is improved by the use of irradiance gradients.

We retain the basic structure of the original algorithm, but each record stores the incoming radiance function over the hemisphere. This allows us to apply the interpolation to glossy surfaces.

3 Algorithm Overview

Radiance caching is a part of a ray tracing approach to global illumination. At every ray-surface intersection, the outgoing radiance is evaluated with the *illumination integral*:

$$L(\theta_o, \phi_o) = \int_0^{2\pi}\int_0^{\pi/2} L_i(\theta_i, \phi_i) f(\theta_i, \phi_i, \theta_o, \phi_o)\cos\theta_i \sin\theta_i \mathrm{d}\theta_i \mathrm{d}\phi_i, \tag{1}$$

where L is the outgoing radiance, L_i is the incoming radiance, and f is the BRDF. The integral is split into parts and each of them is solved by a different technique:

- Direct illumination uses a deterministic method for point light sources and area sampling for area light sources [44].
- Perfect specular reflections/refractions are solved by tracing a single deterministic secondary ray.
- Ward's irradiance caching computes the indirect diffuse term for *purely* diffuse surfaces.
- Two different techniques may be used for glossy surfaces.
  - *Low-frequency BRDF.* Our radiance caching computes the indirect glossy *and diffuse* terms.
  - *High-frequency BRDF.* Monte Carlo importance sampling computes the indirect glossy term and irradiance caching computes the indirect diffuse term.

Radiance caching is not used for high-frequency BRDFs since many coefficients would be needed. Moreover, high-frequency BRDFs are well-localized and importance sampling provides good accuracy with a few secondary rays. The distinction between low and high-frequency BRDFs is done automatically, as described in Section 4.1. The steps of the rendering algorithm related to radiance caching are shown in Fig. 1.

The ith *radiance cache record* contains:

- position $\mathbf{p}_i$,
- local coordinate frame $(\mathbf{u}_i, \mathbf{v}_i, \mathbf{n}_i)$,
- hemispherical harmonics coefficient vector Λ_i representing the incoming radiance,
- two derivative vectors $\frac{\partial \Lambda_i}{\partial x}$ and $\frac{\partial \Lambda_i}{\partial y}$ representing the translational gradient, and
- harmonic mean distance R_i of objects visible from $\mathbf{p}_i$.

Λ denotes a coefficient vector and λ_l^m denotes a coefficient, that is, $\Lambda = \{\lambda_l^m\}$. Each record stores the incident radiance function in a view-independent manner so that it can be

```
// preprocessing - BRDF conversion
for (every surface in the scene) do
  if (surface suitable for radiance caching) then
    Compute and store the HSH representation of the surface's BRDF.
  end if
end for
// rendering with radiance caching
for (every ray-surface intersection $\mathbf{p}$) do
  Retrieve the HSH representation of the BRDF at $\mathbf{p}$.
  if ( HSH representation of the BRDF not available) then
    // high-frequency BRDF
    Importance sampling computes the indirect glossy term.
    Irradiance caching computes the indirect diffuse term.
  else
    // low-frequency BRDF; use radiance caching
    if ( radiance cache records exist near $\mathbf{p}$ ) then
      Compute the HSH coefficients of the incident radiance at $\mathbf{p}$ by gradient-based interpolation.
    else
      Compute incident radiance at $\mathbf{p}$ by sampling the hemisphere above $\mathbf{p}$.
      Compute HSH coefficients of the incoming radiance.
      Compute the translational gradient of the coefficients.
      Store the new radiance record in the radiance cache.
    end if
    Compute the outgoing radiance at $\mathbf{p}$ as the dot product of the coefficient vector of the incoming radiance and that of the BRDF.
  end if
end for
```

Fig. 1. Outline of the radiance caching algorithm. (HSH stands for hemispherical harmonics.)

reused for different viewpoints. The records are stored in an octree, as described by Ward et al. [1].

4 Radiance Caching Details

4.1 BRDF Representation

We represent the BRDFs using the method of Kautz et al. [35] which we briefly describe here. We discretize the hemisphere of outgoing directions. For each discrete outgoing direction (θ_o, ϕ_o), we use hemispherical harmonics to represent the BRDF over the hemisphere of incoming directions. The nth order representation of a cosine weighted[1] BRDF $f_{(\theta_o,\phi_0)}$ for an outgoing direction (θ_o, ϕ_o) is

$$f_{(\theta_o,\phi_o)}(\theta_i, \phi_i) \approx \sum_{l=0}^{n-1} \sum_{m=-l}^{l} c_l^m(\theta_o, \phi_o) H_l^m(\theta_i, \phi_i), \qquad (2)$$

where

$$c_l^m(\theta_o, \phi_o) = \int_0^{2\pi} \int_0^{\pi/2} f(\theta_o, \phi_o, \theta_i, \phi_i) H_l^m(\theta_i, \phi_i) \sin\theta_i \mathrm{d}\theta_i \mathrm{d}\phi_i. \qquad (3)$$

H_l^m are the hemispherical harmonics basis functions [9]. We sample the outgoing hemisphere for (θ_o, ϕ_o) using the parabolic parametrization [45].

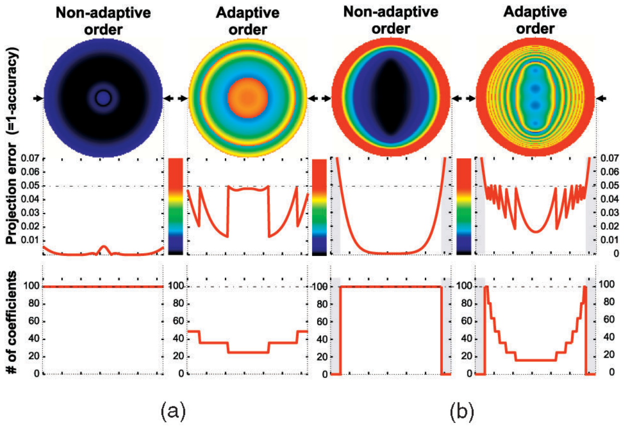


Fig. 2. Adaptive BRDF representation for (a) Phong BRDF with exponent $h = 15$ and (b) anisotropic Ward BRDF [46] with $k_d = 0$, $k_s = 1$, $\alpha_x = 0.6$, $\alpha_y = 0.25$. The order of the hemispherical harmonics representation adapts to the BRDF without ever exceeding the specified maximum representation error (here, 5 percent). The color disks represent BRDF representation error for different outgoing directions (θ_o, ϕ_o). Directions are mapped on the disk with the parabolic parametrization. (One can imagine the disks as looking at the hemisphere from the top.) The graphs represent one scanline from the disk images (i.e., fixed y and varying x component of the outgoing direction). In the case of the Ward BRDF, radiance caching is not used for some directions since the representation error would be too high.

Adaptive BRDF Representation. Hemispherical harmonics representation for all scene BRDFs is computed before the rendering starts. For each outgoing direction, the adaptive representation of $f_{(\theta_o,\phi_o)}$ uses the minimum order n sufficient to avoid exceeding the user specified maximum error, measured as described in [33] (see Fig. 2). If no order $n < n_{\max}$ is sufficient for the specified error, the hemispherical harmonics representation is discarded: Radiance caching will not be used for that BRDF and that outgoing direction. After applying this procedure, only low-frequency BRDFs are represented using the harmonics and radiance caching is used for them. This constitutes an automatic criterion for discerning low and high-frequency BRDFs in our rendering framework. $n_{\max}$ is user specified; we use $n_{\max} = 10$ for our examples. The aim is to have $n_{\max}$ such that a BRDF is classified as low-frequency if and only if radiance caching is more efficient than Monte Carlo importance sampling. While $n_{\max} = 10$ was a good value for our scenes, it would not have to be so in other ones. Higher $n_{\max}$ allows using radiance caching for higher frequency BRDFs. The higher the frequency of the BRDFs, the more rays will be needed to sample the hemisphere for a new record and the less reuse will be possible for that record.

4.2 Incoming Radiance Computation

Whenever interpolation is not possible at a point $\mathbf{p}$, a new radiance record is computed and stored in the cache. We represent the incoming radiance L_i by a vector of hemispherical harmonics coefficients $\Lambda = \{\lambda_l^m\}$ as $L_i(\theta, \phi) \approx \sum_{l=0}^{n-1} \sum_{m=-l}^{l} \lambda_l^m H_l^m(\theta, \phi)$, where n is the representation order. The coefficients λ_l^m for an analytical L_i would be computed with the integral

$$\lambda_l^m = \int_0^{2\pi} \int_0^{\pi/2} L_i(\theta, \phi) H_l^m(\theta, \phi) \sin\theta \mathrm{d}\theta \mathrm{d}\phi.$$

[1] All BRDFs are multiplied by the cosine term $\cos\theta_i$ before computing the harmonics representation.

Our knowledge of L_i is based only on sampling (ray casting). Hence, we compute λ_l^m by a Monte Carlo quadrature with uniform sampling:

$$\lambda_l^m = \frac{2\pi}{N} \sum_{k=1}^{N} L_i(\theta_k, \phi_k) H_l(\theta_k, \phi_k), \tag{4}$$

where $L_i(\theta_k, \phi_k)$ is the incoming radiance coming from the sampled direction (θ_k, ϕ_k) and N is the number of sampled directions. We use a fixed N, but adaptive hemisphere sampling [47], [48] is desirable.

The order n for the incoming radiance representation is equal to the order of the BRDF representation at $\mathbf{p}$. This cuts off high frequencies from the incoming radiance. The approach is justified by a low-frequency BRDF acting as a low-pass filter on the incoming radiance [34]. If incoming radiance vectors with different number of coefficients are interpolated, the shorter vectors are padded with zeros.

4.3 Translational Gradient Computation

We want to compute the translational gradient $\nabla\lambda_l^m$ for each λ_l^m computed with (4). The gradient is used to improve the interpolation smoothness. We did not succeed in extending the gradient computation of Ward and Heckbert [8] to handle our case since their method tightly couples the cosine probability density and the cosine weighting used for irradiance computation.

Instead, we have developed two new methods for computing translational gradient $\nabla\lambda_l^m$. The first, numerical, displaces the center of the hemisphere. The second, analytical, is based on differentiating the terms of (4). In both cases, we compute the gradient $\nabla\lambda_l^m = \left[\frac{\partial\lambda_l^m}{\partial x}, \frac{\partial\lambda_l^m}{\partial y}, 0\right]$ by computing the partial derivatives $\partial\lambda_l^m/\partial x$ and $\partial\lambda_l^m/\partial y$. The gradient is defined in the local coordinate frame at the point $\mathbf{p}$. The derivative with respect to z is not computed since typical displacements along z are very small and using it does not visually improve the interpolation quality. We compute the gradients simultaneously with the computation of the coefficients λ_l^m during the hemisphere sampling.

4.3.1 Numerical Gradient Computation

To compute the derivative $\partial\lambda_l^m/\partial x$ numerically, we displace the point $\mathbf{p}$, along the local x-axis, by Δx to $\mathbf{p}'$ (Fig. 3). For each Monte Carlo sample $L_i(\theta_k, \phi_k)$, we:

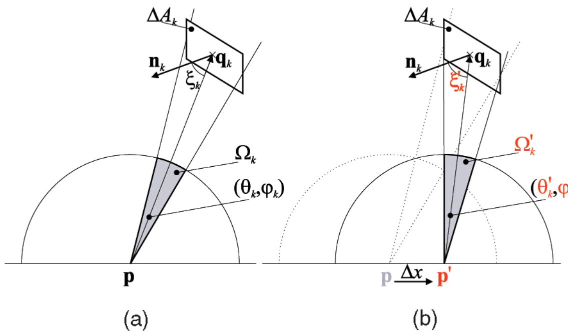


Fig. 3. Gradient computation by displacing the hemisphere center from $\mathbf{p}$ to $\mathbf{p}'$ ((a) before and (b) after the displacement). The quantities changing with the displacement are (shown in red): sample ray direction (θ_k, ϕ_k), the solid angle Ω_k associated with this sample, and the angle ξ_k between the sample direction and the surface normal at the hit point $\mathbf{q}_k$. Neither the hit point $\mathbf{q}_k$ nor the area ΔA_k visible through Ω_k change with the displacement.

1. Compute the new direction (θ_k', ϕ_k') at $\mathbf{p}'$ as $(\theta_k', \phi_k') = \frac{\mathbf{q}_k - \mathbf{p}'}{r_k'}$. Here, $\mathbf{q}_k$ is the point hit by the ray from $\mathbf{p}$ in direction (θ_k, ϕ_k) and $r_k' = \|\mathbf{q}_k - \mathbf{p}'\|$. We will also denote $r_k = \|\mathbf{q}_k - \mathbf{p}\|$. See Fig. 3 for the various terms used here.
2. Compute the solid angle Ω_k' associated with the new direction (θ_k', ϕ_k'). The solid angle Ω_k associated with each direction in (4) is uniform and equal to $2\pi/N$. With the displacement of the point $\mathbf{p}$, the solid angles no longer remain uniform. The change in solid angle is due to the change in distance $r_k = \|\mathbf{q}_k - \mathbf{p}\|$ and orientation of the surface at $\mathbf{q}_k$, as seen from the hemisphere center $\mathbf{p}$ or $\mathbf{p}'$. The solid angle before the displacement is $\Omega_k = \Delta A_k \frac{\cos\xi_k}{r_k^2} = \frac{2\pi}{N}$, where ξ_k is the angle between the surface normal at $\mathbf{q}_k$ and the vector from $\mathbf{q}_k$ to $\mathbf{p}$. The area $\Delta A_k = \frac{2\pi}{N}\frac{r_k^2}{\cos\xi_k}$ is the part of the environment visible through Ω_k. It does not change with the displacement because we assume that the environment visible from $\mathbf{p}$ and $\mathbf{p}'$ is the same. After the displacement, the solid angle subtended by ΔA_k becomes

$$\Omega_k' = \Delta A_k \frac{\cos\xi_k'}{r_k'^2} = \frac{2\pi}{N}\frac{r_k^2}{r_k'^2}\frac{\cos\xi_k'}{\cos\xi_k}.$$

We now estimate the coefficient $\lambda_l^{m\prime}$ at $\mathbf{p}'$ as

$$\lambda_l^{m\prime} = \frac{2\pi}{N}\sum_{k=1}^{N}\frac{r_k^2}{r_k'^2}\frac{\cos\xi_k'}{\cos\xi_k} L_i(\theta_k, \phi_k) H_l^m(\theta_k', \phi_k'),$$

and, finally, we compute $\partial\lambda_l^m/\partial x = (\lambda_l^{m\prime} - \lambda_l^m)/\Delta x$.

The computation of $\partial\lambda_l^m/\partial y$ proceeds in a similar way. This completes the numerical estimation of the translational gradient $\nabla\lambda_l^m$ at the point of interest.

4.3.2 Analytical Gradient Computation

We rewrite (4) as

$$\lambda_l^m = \sum_{k=1}^{N} \Omega_k L_i(\theta_k, \phi_k) H_l(\theta_k, \phi_k), \tag{5}$$

with $\Omega_k = \frac{2\pi}{N}$ for uniform hemisphere sampling. We have seen that Ω_k does not remain constant with displacement of $\mathbf{p}$ and, therefore, it has to be included in the sum and differentiated.

The partial derivative $\partial\lambda_l^m/\partial x$ is computed by differentiating the terms of the sum in (5):

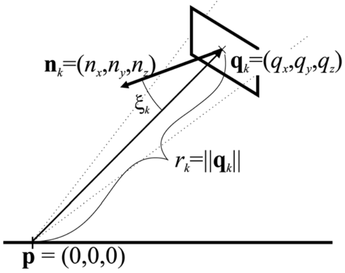


Fig. 4. Quantities in the computation of $\frac{\partial}{\partial q_x}\frac{\cos\xi_k}{r_k^2}$.

$$\frac{\partial\lambda_l^m}{\partial x}=\sum_{k=1}^{N}\frac{\partial}{\partial x}\Big(\Omega_k L_i(\theta_k,\phi_k)H_l^m(\theta_k,\phi_k)\Big)= \sum_{k=1}^{N}L_i(\theta_k,\phi_k)\left(\frac{\partial\Omega_k}{\partial x}H_l^m(\theta_k,\phi_k)+\Omega_k\frac{\partial H_l^m(\theta_k,\phi_k)}{\partial x}\right). \tag{6}$$

The derivative of the basis function is

$$\frac{\partial H_l^m(\theta_k,\phi_k)}{\partial x}=\frac{\partial\theta_k}{\partial x}\frac{\partial H_l(\theta_k,\phi_k)}{\partial\theta_k}+\frac{\partial\phi_k}{\partial x}\frac{\partial H_l^m(\theta_k,\phi_k)}{\partial\phi_k}, \tag{7}$$

with [49]

$$\begin{aligned}\partial\theta_k/\partial x&=-\cos\theta_k\cos\phi_k/r_k,\\ \partial\phi_k/\partial x&=\sin\phi_k/(r_k\sin\theta_k).\end{aligned} \tag{8}$$

Those derivatives with respect to y would be

$$\begin{aligned}\partial\theta_k/\partial y&=-\cos\theta_k\sin\phi_k/r_k,\\ \partial\phi_k/\partial y&=-\cos\phi_k/(r_k\sin\theta_k).\end{aligned} \tag{9}$$

Derivatives $\partial H_l^m/\partial\theta_k$ and $\partial H_l^m/\partial\phi_k$ are given in Appendix A.

The derivative of the solid angle Ω_k is

$$\frac{\partial\Omega_k}{\partial x}=\frac{\partial}{\partial x}\Delta A_k\frac{\cos\xi_k}{r_k^2}=\Delta A_k\frac{\partial}{\partial x}\frac{\cos\xi_k}{r_k^2}.$$

The area $\Delta A_k=\frac{2\pi}{N}\frac{r_k^2}{\cos\xi_k}$ is the part of the environment visible through Ω_k. It does not change with the displacement. The change of $\cos\xi_k/r_k^2$ with the displacement of $\mathbf{p}$ is opposite to its change with the displacement of $\mathbf{q}_k=(q_x,q_y,q_z)$, i.e.,

$$\frac{\partial}{\partial x}\frac{\cos\xi_k}{r_k^2}=-\frac{\partial}{\partial q_x}\frac{\cos\xi_k}{r_k^2}.$$

The derivative $\frac{\partial}{\partial q_x}\frac{\cos\xi_k}{r_k^2}$ can be computed with the assumption that $\mathbf{p}$ lies at the origin because only the relative position of $\mathbf{p}$ and $\mathbf{q}_k$ matters (see Fig. 4). Since $\cos\xi_k=-\frac{\mathbf{n}_k\cdot\mathbf{q}_k}{r_k}$ and $r_k=\|\mathbf{q}_k\|=\sqrt{q_k^2+q_y^2+q_z^2}$, we have

$$\begin{aligned}\frac{\partial}{\partial q_x}\frac{\cos\xi_k}{r_k^2}&=-\frac{\partial}{\partial q_x}\frac{n_xq_x+n_yq_y+n_zq_z}{(q_x^2+q_y^2+q_z^2)^{3/2}}\\ &=-\frac{r_kn_x+3q_x\cos\xi_k}{r_k^4}.\end{aligned} \tag{10}$$

Here, $\mathbf{n}_k=(n_x,n_y,n_z)$ is the surface normal at $\mathbf{q}_k$. Combining this result with $\Delta A_k=\frac{2\pi}{N}\frac{r_k^2}{\cos\xi_k}$, we get

$$\frac{\partial\Omega_k}{\partial x}=\frac{2\pi}{N}\frac{r_kn_x+3q_x\cos\xi_k}{r_k^2\cos\xi_k}. \tag{11}$$

Plugging (11) and (7) into (6), we get the complete formula for $\partial\lambda_l^m/\partial x$. The formulas for $\partial\lambda_l^m/\partial y$ are similar; only (8) must be replaced by (9).

A similar gradient calculation was also proposed in [29]. This method disregards the change of Ω_k and, hence, does not provide good results. The analytical method was also independently developed by Annen et al. [43]. A code fragment evaluating one term of the sum in (6) is given in the accompanying material [50].

4.3.3 Discussion

For the derivation of both numerical and analytical methods, we assumed:

- The radiance $L_i(\theta_k,\phi_k)$ from the point $\mathbf{q}_k$ incident at $\mathbf{p}$ does not change with the displacement of $\mathbf{p}$.
- The visibility of ΔA_k, the small area around $\mathbf{q}_k$, does not change with the displacement of $\mathbf{p}$.

Though none of these assumptions is necessarily valid in all scenes, they are reasonable for small displacements.

The numerical and analytical methods are equivalent, their results are indistinguishable. The numerical method is easier to implement since we do not need to evaluate the basis function derivatives. The analytical method is numerically more stable near edges and corners and also slightly faster to evaluate.

4.3.4 Irradiance Gradient Computation

Note that both methods we propose can still be used if H_l^m is replaced by any other hemispherical function. We also do not rely on uniform hemisphere sampling. Any probability density $p(\theta,\phi)$ can be used for sampling. The only change is that the solid angle becomes $\Omega_k=\frac{1}{Np(\theta_k,\phi_k)}$ instead of $\Omega_k=\frac{2\pi}{N}$, used for the uniform sampling.

As an example, we compute the irradiance gradient ∇E with a cosine-weighted hemisphere sampling. $H_l^m(\theta,\phi)$ is replaced by $\cos\theta$, the probability density of sampling in direction (θ,ϕ) is $p(\theta,\phi)=\frac{\cos\theta}{\pi}$ and, therefore, $\Omega_k=\frac{\pi}{N\cos\theta_k}$. The resulting formula for the analytical method is

$$\frac{\partial E}{\partial x}=\sum_{k=1}^{N}L_i(\theta_k,\phi_k)\left(\frac{\partial\Omega_k}{\partial x}\cos\theta_k-\frac{\pi}{N}\frac{\sin\theta_k}{\cos\theta_k}\frac{\partial\theta_k}{\partial x}\right)$$

with

$$\frac{\partial\Omega_k}{\partial x}=\frac{\pi}{N\cos\theta_k}\frac{r_kn_x+3q_x\cos\xi_k}{r_k^2\cos\xi_k}.$$

We implemented this irradiance gradient computation method and that of Ward and Heckbert [8] and we compared them on a sample scene (Fig. 5). The results were similar for both methods. Ward and Heckbert's method gives better results when some surfaces are seen at very sharp grazing angles from the sampling point $\mathbf{p}$. Otherwise, our method gives slightly smoother results.

Even though the quality of Ward and Heckbert's method is subtly superior to ours, our method provides many advantages. It works with any sampling distribution and with any function used to weight the radiance samples. The contribution of radiance samples to the gradient is independent of each other and, therefore, our method is more easily amenable for parallelization or hardware implementation. We do not assume any stratification of

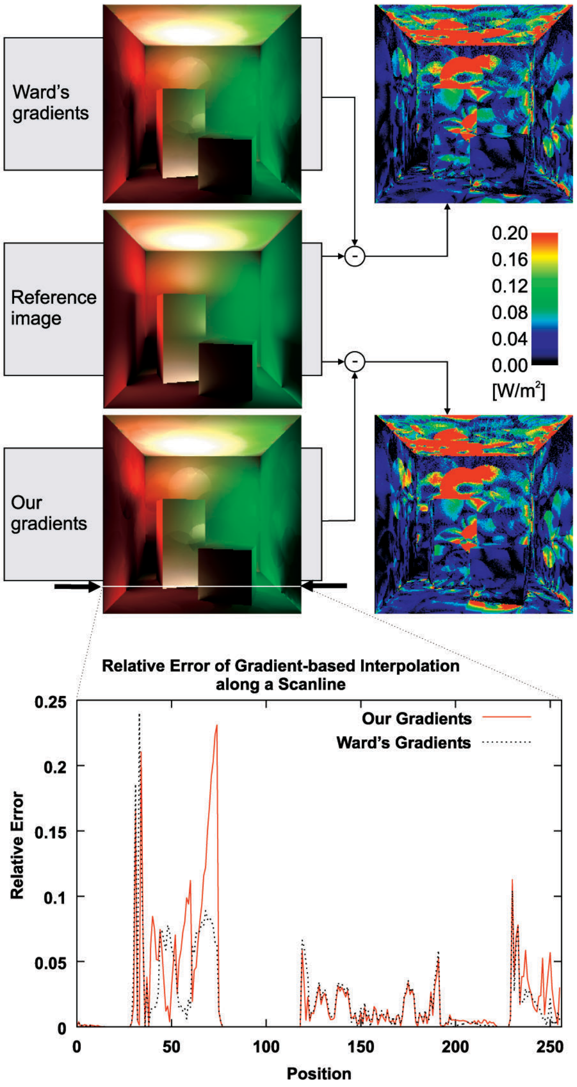


Fig. 5. Comparison of irradiance gradient computation. The scene is a diffuse Cornell box; only first bounce indirect illumination is computed. The color-coded images show the difference between the gradient-based interpolation and the reference solution (10,000 samples per hemisphere at each pixel). RMS error of the images is 0.125 for Ward's method and 0.131 for our method. The graph shows the relative error of the interpolation along a single scanline as compared to the reference solution. Ward's method gives better results when there are surfaces seen at very sharp grazing angles from the sampling point. Otherwise, our method gives a slightly lower error.

the hemisphere. This allows us to use our gradient computation with different sampling strategies, e.g., quasi Monte Carlo sampling.

4.4 Radiance Interpolation

If a query to the radiance cache succeeds, the incoming radiance is interpolated as described in this section. We use a weighted interpolation scheme similar to the one proposed in [8] for interpolating the coefficient vectors Λ_i at any required surface point $\mathbf{p}$. The difference is that we replace the use of the rotational gradient by truly rotating the incident radiance function. This aligns the coordinate frame at the position $\mathbf{p}_i$ of the cache record and the frame at $\mathbf{p}$ (see Fig. 6). The weight $w_i(\mathbf{p})$ of record i with respect to $\mathbf{p}$ is

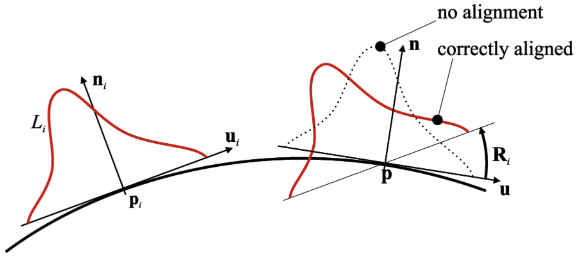


Fig. 6. Rotation $\mathbf{R}_i$ aligns the coordinate frame at $\mathbf{p}_i$ with that at $\mathbf{p}$ to make interpolation possible.

$$w_i(\mathbf{p}) = \left(\frac{\|\mathbf{p}-\mathbf{p}_i\|}{R_i} + \sqrt{1-\mathbf{n}\cdot\mathbf{n}_i}\right)^{-1},$$

where $\mathbf{n}$ is the surface normal at $\mathbf{p}$, $\mathbf{n}_i$ is the surface normal at $\mathbf{p}_i$, and R_i is the harmonic mean distance to objects visible from $\mathbf{p}_i$. The coefficient vector of the interpolated radiance is computed as a weighted average:

$$\Lambda(\mathbf{p}) = \frac{\sum_S \mathbf{R}_i \left(\Lambda_i + d_x \frac{\partial \Lambda_i}{\partial x} + d_y \frac{\partial \Lambda_i}{\partial y}\right) w_i(\mathbf{p})}{\sum_S w_i(\mathbf{p})}, \tag{12}$$

where the set S of radiance records used for interpolation at $\mathbf{p}$ is defined as $S = \{i | w_i(\mathbf{p}) > 1/a\}$ and a is a user-defined desired accuracy. The definition of the set S effectively represents the criterion used to decide which radiance cache records can be used for interpolation. If S is nonempty, interpolation (or extrapolation) is possible. d_x and d_y are the displacements of $\mathbf{p} - \mathbf{p}_i$ along the x and y axes of record i's local coordinate frame. Displacement along z is not taken into account for the gradient-enhanced interpolation since it is typically very small. $\mathbf{R}_i$ is the hemispherical harmonics rotation matrix [9] that aligns the coordinate frame at $\mathbf{p}_i$ with the frame at $\mathbf{p}$.

The interpolation scheme is borrowed from Ward et al. [1]. They derived it from the "split sphere" model that estimates an upper bound on the magnitude of the change of irradiance. Although this model does not apply for radiance caching, the results are satisfying. We observe that a lower a has to be used for higher frequency BRDFs and interpolation errors are more apparent when surfaces are viewed from grazing angles. In future work, we want to devise an interpolation scheme suitable for radiance caching based on these observations.

4.5 Outgoing Radiance Computation

The incoming radiance obtained by interpolation or hemisphere sampling is integrated against the BRDF to compute the outgoing radiance. With an orthonormal basis, the integral evaluation reduces to the dot product [51]:

$$L(\theta_o, \phi_o) = \sum_{l=0}^{n-1} \sum_{m=-l}^{l} \lambda_l^m c_l^m(\theta_o, \phi_o). \tag{13}$$

λ_l^m is an interpolated incoming radiance coefficient and $c_l^m(\theta_o, \phi_o)$ is a BRDF coefficient at $\mathbf{p}$ for the outgoing direction (θ_o, ϕ_o).

	Cornell Box 1280 × 1280		Disney Hall 1440 × 840		Flamingo 1280 × 1280		
	Frame I	Frame II	Frame I	Frame II	Frame I	Frame II	Frame III
RC filling	28.7	9.2	218.0	169.0	63.5	53.2	63.9
RC interpolation	4.6	4.6	17.3	17.0	4.42	11.4	8.7
Total	82.9	68.7	357.8	295.6	108.0	101.0	104.0

Fig. 7. Timing breakdown for the test scenes. "RC filling" is the time spent on computing and adding radiance cache records. "RC interpolation" is the time spent on looking up the existing radiance cache records and interpolating the radiance. "Total" is the total rendering time. The difference between the total time and the time spent by radiance caching consists of primary ray casting, direct lighting, and irradiance caching (in Disney Hall and Flamingo). All times given in seconds.

5 Results

Fig. 7 gives the breakdown of rendering times for the the three scenes we used to test radiance caching (Cornell Box, Walt Disney Hall, Flamingo). The timings were measured on a 2.2GHz Pentium 4 with 1 GB RAM running Windows XP. The resulting renderings are shown in Figs. 8, 9, and 10 and in the accompanying video [50]. The maximum hemispherical harmonics order for radiance caching was set to $n = 10$, which corresponds to approx. 3.6 kB sized radiance cache records.

We compared the solutions obtained by radiance caching with those obtained by Monte Carlo importance sampling. Importance sampling uses the surface BRDF as the importance function. The two rendering methods exhibit artifacts with very different characteristics: high-frequency noise for importance sampling ("specks" in images) and low-frequency error in radiance caching (uneven illumination gradients). It is therefore difficult to compare rendering times needed to attain the same visual quality. Instead, we have chosen to fix the rendering time and compare the image quality delivered by the two methods.

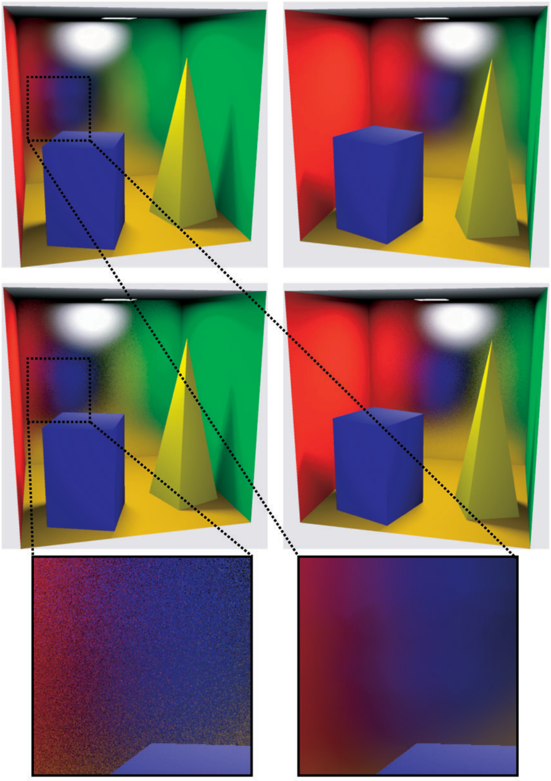
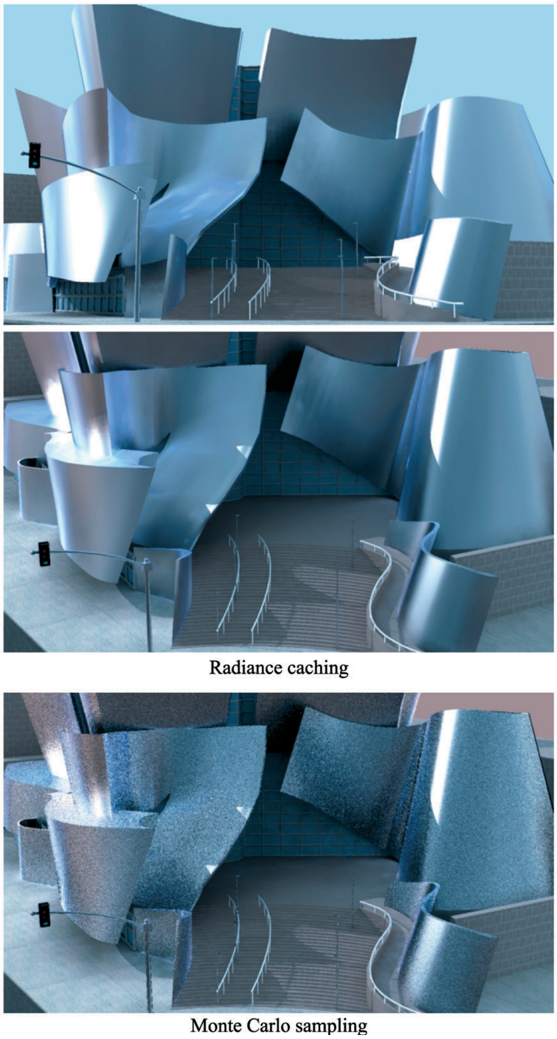


Fig. 8. Two views of a Cornell Box with glossy back wall rendered using radiance caching (top) and Monte Carlo importance sampling (bottom).

5.1 Cornell Box

Fig. 8 shows renderings of a Cornell Box with a glossy back wall (Phong BRDF, exponent 22), taken from two viewpoints at resolution $1,280 \times 1,280$. Except for the back wall, all objects are Lambertian. Only direct lighting and first bounce indirect glossy lighting for the back wall were computed.



Radiance caching

Monte Carlo sampling

Fig. 9. Rendering of a simple model of Walt Disney Hall in Los Angeles. The top and the middle images, compared with radiance caching, show the building from two different viewpoints. The bottom image was computed with Monte Carlo importance sampling.

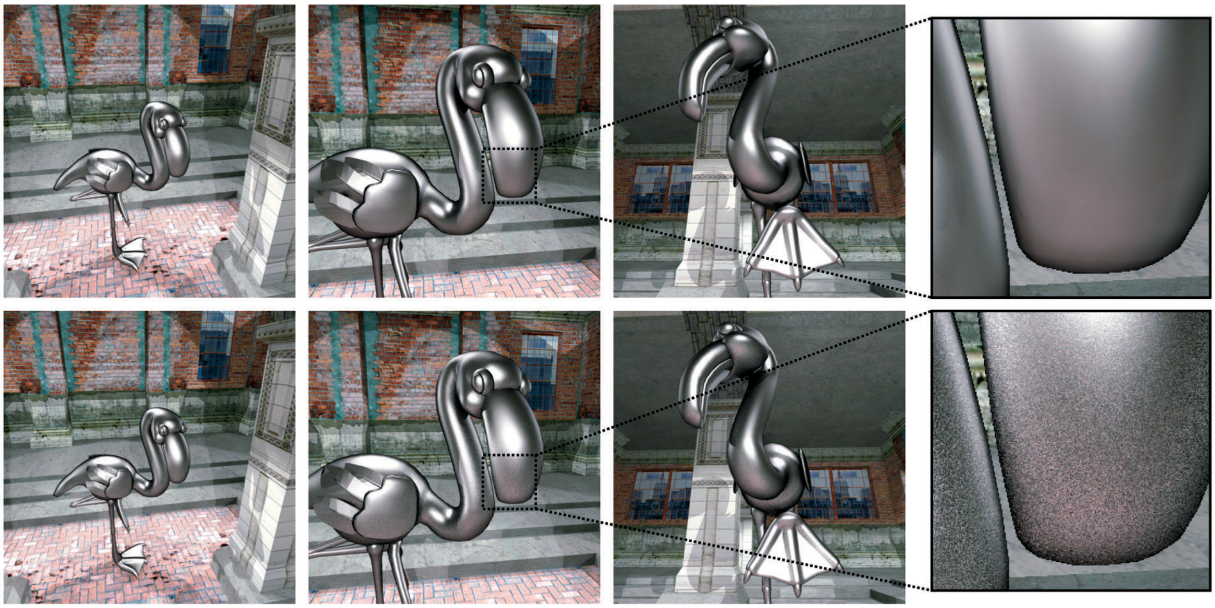


Fig. 10. Frames from the Flamingo animation rendered using radiance caching (top row) and Monte Carlo importance sampling (bottom row).

Images in the top row were computed using radiance caching with the caching accuracy set to $a = 0.15$ and the number of rays cast to sample each hemisphere set to $N = 6,000$. The indirect glossy term took 33.3 seconds to compute for the left image; total rendering time was 82.9 sec (direct illumination uses eight samples per pixel to sample the area source). The number of radiance cache records was 600. The time spent on the indirect glossy term computation in the right image was only 13.8 sec since the records from the left rendering were retained and only 164 additional records were required.

The indirect glossy term for each of the two bottom images was computed in 35 seconds using Monte Carlo importance sampling with 12 reflected rays per pixel on a glossy surface. Those rendering methods exhibit a high noise level whose perception is even amplified in the temporal domain, as shown in the video.

The average time spent on radiance caching for a 180 frames long animation with the camera moving between the position in the left and right images was just 4.9 sec per frame. Most of this time is spent on interpolation since only a few records are needed for additional frames. The average frame time with Monte Carlo importance sampling is 35 seconds since this method does not reuse any information from the previous frames. Even though this time is seven times longer than for radiance caching, the quality is much lower.

5.2 Walt Disney Hall

Fig. 9 shows renderings of a simple model of Walt Disney Hall in Los Angeles. The curved walls of the real building are covered with brushed metal tiles, whose BRDF we approximate by a three-lobe Lafortune model [52]. The illumination is due to the sun (modeled as a directional light), the sky (modeled as a constant blue light), and the surrounding urban environment (modeled as a constant brownish light). Similarly to the real building (see [50]), walls reflect the sky or the surroundings depending on their normals and the viewpoint (compare the top and the middle image).

A full global illumination solution with one ray per pixel was computed at resolution $1,440 \times 840$ and then scaled down. Indirect illumination and the illumination coming from the sky and from the surroundings was computed in the same way: with irradiance caching on diffuse surfaces, with radiance caching on glossy surfaces in the top and the middle image, and with Monte Carlo sampling on glossy surfaces in the bottom image.

To capture the high indirect illumination variations on the metal walls, the caching accuracy was set to $a = 0.1$, leading to as many as 38,000 radiance cache records. We used 800 rays for hemisphere sampling. The rendering time was 357.8 s and 295.6 s for the top and the middle images, respectively. Monte Carlo importance sampling in the bottom image used 15 reflected rays on each visible glossy pixel (first bounce) and one ray for other bounces. The rendering time was similar to that for radiance caching, but the quality of radiance caching results is higher.

The BRDF we used for the metallic walls [53] was relatively sharp; the Lafortune lobes had exponents of $16, 88,$ and 186. It is impossible to represent such a BRDF accurately using hemispherical harmonics of order 10—the average representation error was over 20 percent. Nonetheless, radiance caching renderings show no serious distortion of the material appearance compared to Monte Carlo sampling.

5.3 Glossy Flamingo

Fig. 10 shows three frames from the Flamingo animation. The bird was assigned the Phong BRDF (exponent 7) and all other surfaces are purely diffuse. The rendering resolution was $1,280 \times 1,280$ pixels. Full global illumination up to four bounces was computed. Irradiance caching was used to compute indirect lighting on diffuse surfaces.

First bounce indirect lighting on glossy surfaces for the images in the top row was computed using radiance caching; path tracing was used for deeper recursion levels. The caching accuracy was set to $a = 0.15$ and the number of rays for hemisphere sampling was $N = 1,000$. Fig. 7 gives the rendering times for the three images when rendered

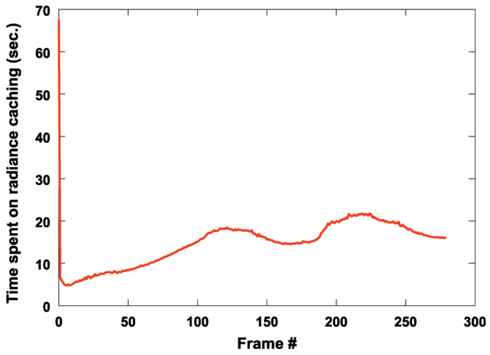


Fig. 11. Time spent on radiance caching for the Flamingo animation (see the accompanying video). Cached records are shared between the frames.

independently, without retaining radiance cache records from previous renderings. Fig. 11 shows the time spent on radiance caching in a 280 frames long animation with camera moving between the shown images (see the accompanying video).

The images in the bottom row use Monte Carlo importance sampling instead of radiance caching. In order to keep the rendering time the same as for radiance caching, the number of reflected rays per pixel on a glossy surface was set to 12, 4, 6, respectively (from left to right).

This scene is particularly challenging for radiance caching since the glossy surface is curved. On such a surface, radiance cache records cannot be used for interpolation at as many pixels as on a flat surface. Moreover, costly alignment is required before each interpolation. In the left image, the flamingo occupies only a small part of the screen and, therefore, one does not take advantage of radiance caching's independence on image resolution. The quality advantage of radiance caching over Monte Carlo importance sampling can be seen only by a very close inspection (see the images in the accompanying material [50]). However, in the other two images, the noise introduced by Monte Carlo sampling is more obvious. Notice also that, for the animation rendering, the average time for radiance caching is 15 seconds per frame. Rendering the same animation using Monte Carlo importance sampling with only two reflected rays on a glossy pixel leads to 27 seconds per frame spent on indirect glossy lighting computation. Notice, in the accompanying video, that the quality obtained by radiance caching is considerably better.

6 Discussion

6.1 Rotation Loss

There is a loss of information when radiances are interpolated on a curved surface (Fig. 12). A part of the radiance incident at $\mathbf{p}_i$ should disappear under the surface (marked "a" in Fig. 12) and should not contribute to the interpolated radiance at $\mathbf{p}$. A part of the radiance actually incident at $\mathbf{p}$ is not represented by the radiance record at $\mathbf{p}_i$ (marked "b" in Fig. 12) and is missing in the interpolated radiance.

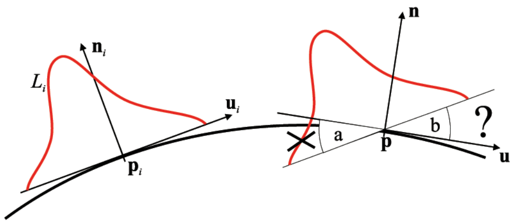


Fig. 12. Information loss when radiances are interpolated on a curved surface.

This problem is not due to using a hemispherical basis for representing the incoming radiance, but due to the fact that the incident radiance at a surface point *is* a hemispherical function. Using spherical, instead of hemispherical, harmonics would not solve this problem. In practice, the error introduced by this problem is very small because the difference between the normal at $\mathbf{p}$ and the normal at any record used for interpolation at $\mathbf{p}$ is small. Note also that Ward et al.'s irradiance caching suffers from the same problem.

6.2 Global versus Local Coordinates

Incoming radiance at a point can be represented either in the local frame at that point or in the global frame. This influences how the interpolation at $\mathbf{p}$ is performed:

- *Incoming radiance in the global frame*

```
**for** (each available record $i$) **do**
    Update the interpolation sums in (12).
**end for**
Align the result with the local frame at $\mathbf{p}$.
Compute the dot product.
```

- *Incoming radiance in the local frame*

```
**for** (each available record $i$) **do**
    Align the frame at $\mathbf{p}_i$ with the frame at $\mathbf{p}$.
    Update the interpolation sums in (12).
**end for**
Compute the dot product.
```

The final dot product (13) is always carried out in the local frame at $\mathbf{p}$. On curved surfaces, fewer rotations are needed if the incoming radiance is represented in the global frame. On the other hand, if the incoming radiance is represented in the local frame, no alignment (even with the BRDF) is needed on flat surfaces. The lowest number of rotations is obtained if the incoming radiance is represented in the local frame on flat surfaces and in the global frame on curved surfaces. Note that full spherical function representation (e.g., using spherical harmonics) is needed to represent the incoming radiance in the global frame.

6.3 Suitability of Hemispherical Harmonics

We use (hemi)spherical harmonics since they are simple, computationally efficient (manipulations of vectors of floats), avoid aliasing, and rotation is available for them. The essential disadvantage is the lack of directional localization. When we create a new radiance cache record, the full hemisphere must be sampled, whatever the incoming ray direction is. The more directional the BRDF is, the more this approach becomes wasteful. With a basis that localizes in directions, only the required part of the hemisphere would need to be sampled. For this purpose, one can use piecewise constant representation [6], [26], [27], but it would presumably introduce aliasing. The use of spherical wavelets [30] is probably a good choice, and is left for further investigation.

6.4 General Limitations

The BRDF frequencies used in the example scenes are near the limit of what our technique can currently handle. The restriction to low-frequency BRDFs is the main limitation of our technique because we use hemispherical harmonics. We believe that higher frequency BRDFs can be handled within the presented framework by using a localized basis such as wavelets. Additionally, the radiance caching, as a gathering technique, cannot solve certain types of light transfer phenomena, such as caustics. Those have to be solved by a more adapted technique.

7 Conclusion and Future Work

We have presented radiance caching, a method for accelerating the computation of the indirect illumination on surfaces with low-frequency glossy BRDFs. Radiance caching is based on sparse sampling, caching, and interpolating incoming radiance on those surfaces. Radiance is represented by hemispherical or spherical harmonics in our approach. The interpolation quality is enhanced by the use of translational gradients that can be computed with two novel methods we have presented in this paper. We have also proposed an automatic criterion to decide for which BRDFs radiance caching is suitable. We have shown in several examples that radiance caching is more efficient than pure Monte Carlo sampling at every surface point and delivers images of superior quality.

In future work, we would like to use adaptive hemisphere sampling to compute the incoming radiance coefficients [29], [48], [54], use a different representation for the incoming radiance that localizes better in directions, and devise interpolation criteria better suited for glossy surfaces. In the long term, we wish to investigate the relationship between the frequency content of a BRDF and the suitability of interpolating radiance on surfaces with that BRDF.

Appendix

Derivatives of Spherical and Hemispherical Harmonics

Partial derivatives for spherical harmonics are:

$$\frac{\partial Y_l^m}{\partial \theta}(\theta,\phi) =$$
$$\begin{cases} -\sqrt{2}K_l^m \cos(m\phi)\sin(\theta)\frac{\mathrm{d}P_l^m}{\mathrm{d}x}(\cos\theta) & \text{if } m>0 \\ -\sqrt{2}K_l^m \sin(-m\phi)\sin(\theta)\frac{\mathrm{d}P_l^{-m}}{\mathrm{d}x}(\cos\theta) & \text{if } m<0 \\ -K_l^0 \sin(\theta)\frac{\mathrm{d}P_l^0}{\mathrm{d}x}(\cos\theta) & \text{if } m=0, \end{cases}$$
$$\frac{\partial Y_l^m}{\partial \phi}(\theta,\phi) = \begin{cases} 0 & \text{if } m=0 \\ -mY_l^{-m}(\theta,\phi) & \text{otherwise,} \end{cases}$$

where $K_l^m = \sqrt{\frac{(2l+1)(l-|m|)!}{4\pi(l+|m|)!}}$.The derivative of the associated Legendre polynomials can be found from the recurrence formula:

$$\frac{\mathrm{d}P_l^m}{\mathrm{d}x}(x) = \begin{cases} \frac{1}{x^2-1}\Big(xlP_l^m(x) - (m+l)P_{l-1}^m(x)\Big) & \text{if } m<l \\ -(-1)^m x(2m-1)!!(1-x^2)^{\frac{m}{2}-1} & \text{if } m=l, \end{cases}$$

where $x!!$ is the double factorial (product of all odd integers less than or equal to x).

The partial derivatives for hemispherical harmonics are:

$$\frac{\partial H_l^m}{\partial \theta}(\theta,\phi) =$$
$$\begin{cases} -2\sqrt{2}\widetilde{K}_l^m \cos(m\phi)\sin(\theta)\frac{\mathrm{d}P_l^m}{\mathrm{d}x}(2\cos\theta-1) & \text{if } m>0 \\ -2\sqrt{2}\widetilde{K}_l^m \sin(-m\phi)\sin(\theta)\frac{\mathrm{d}P_l^{-m}}{\mathrm{d}x}(2\cos\theta-1) & \text{if } m<0 \\ -2\widetilde{K}_l^0 \sin(\theta)\frac{\mathrm{d}P_l^0}{\mathrm{d}x}(2\cos\theta-1) & \text{if } m=0, \end{cases}$$
$$\frac{\partial H_l^m}{\partial \phi}(\theta,\phi) = \begin{cases} 0 & \text{if } m=0 \\ -mH_l^{-m}(\theta,\phi) & \text{otherwise,} \end{cases}$$

where $\widetilde{K}_l^m = \sqrt{2}K_l^m$.

Acknowledgments

The authors thank Vlastimil Havran for letting them use his ray tracer and Marina Casiraghi and Eric Risser for the Walt Disney Hall model. This work was partially supported by ATI Research, the I-4 Matching fund, and the US Office of Naval Research.

References

[1] G.J. Ward, F.M. Rubinstein, and R.D. Clear, "A Ray Tracing Solution for Diffuse Interreflection," *Proc. SIGGRAPH '88,* pp. 85-92, 1988.

[2] G.J. Ward, "The Radiance Lighting Simulation and Rendering System," *Proc. SIGGRAPH '94,* pp. 459-472, 1994.

[3] X. Granier and G. Drettakis, "A Final Reconstruction Approach for a Unified Global Illumination Algorithm," *ACM Trans. Graphics,* vol. 23, no. 2, pp. 163-189, 2004.

[4] E. Groeller, "Coherence in Computer Graphics," PhD dissertation, Technische Universität Wien, 1992.

[5] B. Guo, "Progressive Radiance Evaluation Using Directional Coherence Maps," *Proc. SIGGRAPH '98,* 1998.

[6] P. Slusallek, W. Heidrich, and H.-P. Seidel, "Radiance Maps: An Image-Based Approach to Global Illumination," SIGGRAPH '98, Technical Sketch, 1998.

[7] K. Bala, J. Dorsey, and S. Teller, "Radiance Interpolants for Accelerated Bounded-Error Ray Tracing," *ACM Trans. Graphics,* vol. 18, no. 3, pp. 213-256, 1999.

[8] G.J. Ward and P.S. Heckbert, "Irradiance Gradients," *Proc. Eurographics Workshop Rendering,* 1992.

[9] P. Gautron, J. Křivánek, S.N. Pattanaik, and K. Bouatouch, "A Novel Hemispherical Basis for Accurate and Efficient Rendering," *Proc. Eurographics Symp. Rendering,* 2004.

[10] P. Hanrahan, D. Salzman, and L. Aupperle, "A Rapid Hierarchical Radiosity Algorithm," *Proc. SIGGRAPH '91,* 1991.

[11] P.S. Heckbert, "Simulating Global Illumination Using Adaptive Meshing," PhD dissertation, Univ. of California., 1991.

[12] D. Lischinski, F. Tampieri, and D.P. Greenberg, "Discontinuity Meshing for Accurate Radiosity," *IEEE Computer Graphics and Applications,* vol. 12, no. 6, pp. 25-39, Nov. 1992.

[13] B. Walter, G. Drettakis, and S. Parker, "Interactive Rendering Using Render Cache," *Proc. 13th Eurographics Workshop Rendering,* pp. 19-30, 1999.

[14] B. Walter, G. Drettakis, D.P. Greenberg, and O. Deussen, "Enhancing and Optimizing the Render Cache," *Proc. 10th Eurographics Workshop Rendering,* June 2002.

[15] K. Bala, B. Walter, and D. Greenberg, "Combining Edges and Points for Interactive High-Quality Rendering," *ACM Trans. Graphics (Proc. SIGGRAPH 2003),* vol. 22, no. 3, 2003.

[16] M. Simmons and C.H. Séquin, "Tapestry: A Dynamic Mesh-Based Display Representation for Interactive Rendering," *Proc. 11th Eurographics Workshop Rendering,* pp. 329-340, June 2000.

[17] P. Tole, F. Pellacini, B. Walter, and D.P. Greenberg, "Interactive Global Illumination in Dynamic Scenes," *ACM Trans. Graphics (Proc. SIGGRAPH 2002),* vol. 21, no. 3, pp. 537-546, July 2002.

[18] F.X. Sillion, J.R. Arvo, S.H. Westin, and D.P. Greenberg, "A Global Illumination Solution for General Reflectance Distributions," *Proc. SIGGRAPH '91*, pp. 187-196, 1991.

[19] L. Aupperle and P. Hanrahan, "A Hierarchical Illumination Algorithm for Surfaces with Glossy Reflection," *Proc. SIGGRAPH '93*, pp. 155-162, 1993.

[20] F. Sillion, G. Drettakis, and C. Soler, "A Clustering Algorithm for Radiance Calculation in General Environments," *Rendering Techniques*, June 1995.

[21] S.N. Pattanaik and K. Bouatouch, "Haar Wavelet: A Solution to Global Illumination with General Surface Properties," *Proc. Fifth Eurographics Workshop Rendering*, 1995.

[22] P. Schröder and P. Hanrahan, "Wavelet Methods for Radiance Computations," *Proc. Fifth Eurographics Workshop Rendering*, G. Sakas and P. Shirley, eds., pp. 310-326, 1995.

[23] R.R. Lewis and A. Fournier, "Light-Driven Global Illumination with a Wavelet Representation," *Proc. Seventh Eurographics Workshop Rendering*, pp. 11-20, 1996.

[24] P.H. Christensen, D. Lischinski, E.J. Stollnitz, and D.H. Salesin, "Clustering for Glossy Global Illumination," *ACM Trans. Graphics*, vol. 16, no. 1, pp. 3-33, 1997.

[25] M. Stamminger, A. Scheel, X. Granier, F. Perez-Cazorla, G. Drettakis, and F. Sillion, "Efficient Glossy Global Illumination with Interactive Viewing," *Computer Graphics Forum*, vol. 19, no. 1, 2000.

[26] T. Kato, "Photon Mapping in Kilauea," *Siggraph 2002, Course Notes No. 43*, pp. 122-191, 2002.

[27] T. Tawara, K. Myszkowski, and H.-P. Seidel, "Exploiting Temporal Coherence in Final Gathering for Dynamic Scenes," *Proc. Computer Graphics Int'l*, 2004.

[28] H.W. Jensen, *Realistic Image Synthesis Using Photon Mapping.* AK Peters, July 2001.

[29] J. Zaninetti, X. Serpaggi, and B. Péroche, "A Vector Approach for Global Illumination in Ray Tracing," *Proc. Eurographics*, 1998.

[30] P. Schröder and W. Sweldens, "Spherical Wavelets: Efficiently Representing Functions on the Sphere," *Proc. SIGGRAPH '95*, pp. 161-172, 1995.

[31] B. Cabral, N. Max, and R. Springmeyer, "Bidirectional Reflection Functions from Surface Bump Maps," *Proc. SIGGRAPH '87*, pp. 273-281, 1987.

[32] S.H. Westin, J.R. Arvo, and K.E. Torrance, "Predicting Reflectance Functions from Complex Surfaces," *Proc. SIGGRAPH 92*, pp. 255-264, 1992.

[33] R. Ramamoorthi and P. Hanrahan, "Frequency Space Environment Map Rendering," *Proc. SIGGRAPH*, 2002.

[34] R. Ramamoorthi, "A Signal-Processing Framework for Forward and Inverse Rendering," PhD dissertation, Stanford Univ., 2002.

[35] J. Kautz, P.-P. Sloan, and J. Snyder, "Fast, Arbitrary BRDF Shading for Low-Frequency Lighting Using Spherical Harmonics," *Proc. 13th Eurographics Workshop Rendering*, pp. 291-296, 2002.

[36] P.-P. Sloan, J. Kautz, and J. Snyder, "Precomputed Radiance Transfer for Real-Time Rendering in Dynamic, Low-Frequency Lighting Environments," *Proc. SIGGRAPH*, 2002.

[37] P.-P. Sloan, J. Hall, J. Hart, and J. Snyder, "Clustered Principal Components for Precomputed Radiance Transfer," *Proc. SIGGRAPH 2003*, pp. 382-391, 2003.

[38] J.C. Wyant and K. Creath, "Basic Wavefront Aberration Theory for Optical Metrology," *Applied Optics and Optical Eng.*, vol XI, pp. 27-39, 1992.

[39] J. Koenderink, A. van Doorn, and M. Stavridi, "Bidirectional Reflection Distribution Function Expressed in Terms of Surface Scattering Modes," *Proc. European Conf. Computer Vision*, vol. B, pp. 28-39, 1996.

[40] O.A. Makhotkin, "Analysis of Radiative Transfer between Surfaces by Hemispherical Harmonics," *J. Quantitative Spectroscopy and Radiative Transfer*, vol. 56, no. 6, pp. 869-879, 1996.

[41] J. Arvo, "The Irradiance Jacobian for Partially Occluded Polyhedral Sources," *Proc. SIGGRAPH '94*, 1994.

[42] N. Holzschuch and F. Sillion, "Accurate Computation of the Radiosity Gradient with Constant and Linear Emitters," *Proc. Sixth Eurographics Workshop Rendering*, June 1995.

[43] T. Annen, J. Kautz, F. Durand, and H.-P. Seidel, "Spherical Harmonic Gradients for Mid-Range Illumination," *Proc. Eurographics Symp. Rendering 2004*, 2004.

[44] P. Shirley and C. Wang, "Direct Lighting Calculation by Monte Carlo Integration," *Proc. Second Eurographics Workshop Rendering*, pp. 54-59, 1994.

[45] W. Heidrich and H.-P. Seidel, "Realistic, Hardware-Accelerated Shading and Lighting," *Proc. SIGGRAPH '99*, 1999.

[46] G.J. Ward, "Measuring and Modeling Anisotropic Reflection," *Proc. SIGGRAPH '92*, pp. 265-272, 1992.

[47] X. Serpaggi and B. Péroche, "An Adaptive Method for Indirect Illumination Using Light Vectors," *Computer Graphics Forum (EUROGRAPHICS 2001 Proc.)*, vol. 20, no. 3, 2001.

[48] P. Shirley and K. Chiu, "Notes on Adaptive Quadrature on the Hemisphere," Technical Report TR-411, Indiana Univ., July 1994.

[49] E.W. Weisstein, "Spherical Coordinates," *MathWorld*, http://mathworld.wolfram.com/Spherical-Coordinates.html, 2004.

[50] http://www.cs.ucf.edu/graphics/RCache/index.html, the Web page accompanying this paper, 2005.

[51] G. Szegö, *Orthogonal Polynomials*, fourth ed. Providence, R.I.: Am. Math. Soc., 1975.

[52] E.P.F. Lafortune, S.-C. Foo, K.E. Torrance, and D.P. Greenberg, "Non-Linear Approximation of Reflectance Functions," *Proc. SIGGRAPH '97*, 1997.

[53] S.H. Westin, "Lafortune BRDF for RenderMan," http://www.graphics.cornell.edu/westin/lafortune/lafortune.html, 2000.

[54] J. Rigau, M. Feixas, and M. Sbert, "Refinement Criteria Based on f-Divergences," *Proc. 14th Eurographics Workshop Rendering*, pp. 260-269, 2003.

Jaroslav Křivánek received the Master's degree in computer science from the Czech Technical University. He is a PhD student at IRISA/INRIA Rennes and the Czech Technical University in Prague and a visiting research associate at the University of Central Florida. His research focuses on real-time and offline lighting simulation and using visual perception for efficient lighting computation.

Pascal Gautron received the Master's degree in computer science from the University of Poitiers. He is a PhD student at IRISA/INRIA Rennes, France. He works in collaboration with the University of Central Florida and the University of Poitiers. His main research interest is high quality real-time rendering using graphics hardware.

Sumanta Pattanaik is an associate professor in the Computer Science Department at the University of Central Florida. From 1995 to 2001, he was a research associate in the Program of Computer Graphics at Cornell University. Before that, he was a postdoctoral researcher at IRISA/INRIA (1993-1995), France. His main fields of research are realistic image synthesis and digital imaging. His current research focuses on real-time realistic rendering and the application of visual perception for efficient lighting computation and accurate display. He is a member of ACM SIGGRAPH, Eurographics, and the IEEE.

Kadi Bouatouch received the PhD degree in 1977 and a higher doctorate in computer science in the field of computer graphics in 1989. He is an electronics and automatic systems engineer (ENSEM 1974). His research interests are global illumination, lighting simulation, and remote rendering for complex environments, real-time high fidelity rendering, parallel radiosity, virtual and augmented reality, and computer vision. He is currently a professor at the University of Rennes 1, France, and researcher at IRISA. He is a member of Eurographics, ACM, and the IEEE. He is and was a member of the program committees of several conferences and workshops and a referee for several computer graphics journals

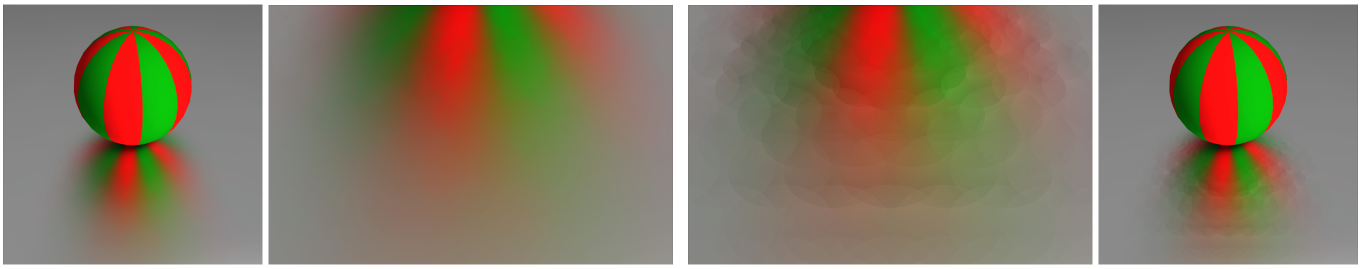
Improved Radiance Gradient Computation

Jaroslav Křivánek* IRISA/INRIA Rennes, Czech Technical University, Univ. of Central Florida

Pascal Gautron† IRISA/INRIA Rennes, Univ. of Central Florida

Kadi Bouatouch‡ IRISA/INRIA Rennes

Sumanta Pattanaik§ Univ. of Central Florida



New gradients

Gradients by [Křivánek et al. 2005]

Figure 1: **Right:** The gradient computation proposed by [Křivánek et al. 2005] does not properly handle significant change of occlusion in the sampled environment and leaves visible interpolation artifacts. **Left:** The radiance gradient computation proposed in this paper handles occlusion changes and leads to a smoother indirect illumination interpolation on the glossy floor. The two images in the middle are cut out from the two images on the very left and very right.

Abstract

We describe a new and accurate algorithm for computing translational gradients of incoming radiance in the context of a ray tracing-based global illumination solution. The gradient characterizes how the incoming directional radiance function changes with displacement on a surface. We use the gradient for a smoother radiance interpolation over glossy surfaces in the framework of the radiance caching algorithm. The proposed algorithm generalizes the irradiance gradient computation by [Ward and Heckbert 1992] to allow its use for non-diffuse, glossy, surfaces. Compared to previous method for radiance gradient computation, the new algorithm yields better gradient estimates in the presence of significant occlusion changes in the sampled environment, allowing a smoother indirect illumination interpolation.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Shading, Shadowing

Keywords: radiance gradients, radiance caching, irradiance gradients, global illumination, ray tracing

1 Introduction and Previous Work

Irradiance caching [Ward et al. 1988; Křivánek et al. 2005] is a ray tracing-based method for computing indirect diffuse illumination. It exploits the indirect illumination coherence by sparsely sampling, caching and interpolating the indirect diffuse illumination over surfaces, instead of computing it independently for each pixel. Indirect illumination at a caching point is computed by sampling the hemisphere of incoming directions through a number of secondary rays, and then stored as a new record in the cache. Later, when a ray hits a surface sufficiently close to the stored records, indirect illumination can be simply interpolated instead of computing it again by a costly hemisphere sampling.

Ward and Heckbert [1992] found that the interpolation quality on diffuse surfaces can be significantly improved by the use of translational and rotational irradiance gradients. The translational gradient characterizes the change of irradiance with a small displacement on a surface and the rotational gradient describes the change with surface rotation. The gradients are computed simultaneously with the hemisphere sampling and stored in the cache to be used later, during the interpolation, to effectively raise the interpolation order.

In our earlier work on *radiance caching* [Křivánek et al. 2005], we used the workings of irradiance caching to efficiently compute indirect illumination on glossy surfaces. We achieved it by caching the full directional incoming radiance function (represented by hemispherical or spherical harmonics [Gautron et al. 2004]) instead of just a scalar irradiance value. Similarly to Ward and Heckbert [1992], we also used translational gradients to improve the interpolation quality. However, some important differences between the ways hemisphere is sampled in irradiance and radiance caching prevented us from directly adapting Ward's and Heckbert's gradient computation algorithm. Instead, we proposed a more general translational gradient computation method that was applicable to the hemisphere sampling schemes used in both radiance and irradiance caching. This previous method works well for most situations, but it breaks down when there is a significant occlusion change with translation (Figure 1).

*jkrivane@irisa.fr
†pgautron@irisa.fr
‡kadi@irisa.fr
§sumant@cs.ucf.edu

In this paper we propose a novel algorithm for translational gradient computation which works well in the presence of occlusion changes and is general enough to be applicable to radiance caching. The algorithm is based on the gradient computation proposed in [Ward and Heckbert 1992], but we reformulate the problem without assuming uniform projected area hemisphere sampling used in irradiance caching. Our formulation allows an arbitrary weighting function to be used for incoming radiance samples, allowing projection of the incoming radiance function onto an arbitrary hemispherical basis.

Apart from the papers already mentioned, the following work on translational illumination gradient computation exists. Arvo [1994] computes the irradiance Jacobian due to partially occluded polygonal emitters of constant radiosity, whereas Holzschuch and Sillion [1995] handle polygonal emitters with arbitrary radiosity. The gradient computation presented by Annen et al. [2004] is equivalent to our previous gradient computation method described above.

The rest of the paper is organized as follows. Section 2 gives a short review of radiance caching, Section 3 presents the main contribution — a new gradient computation method. Results are showcased in Section 4 and Section 5 concludes the work.

2 Radiance Caching Overview

Radiance caching is based on sparse sampling, caching and interpolating incoming radiance function over visible glossy surfaces. Whenever a ray hits a surface, the cache is queried for nearby incoming radiance records and, if some are found, the indirect illumination is evaluated by a gradient-based interpolation as described in [Křivánek et al. 2005].

In this paper, we deal with the situation when a ray hits a surface and the interpolation is not possible due to the lack of nearby records. In this case, hemispherical incoming radiance function and its translational gradient are computed by Monte Carlo quadrature and stored as a new record in the radiance cache. For efficient representation, the incoming radiance function L^i is projected onto the basis of spherical or hemispherical harmonics and represented by a vector of projection coefficients $\Lambda = \{\lambda_l^m\}$ as $L^i(\theta,\phi) \approx \sum_{l=0}^{n-1}\sum_{m=-l}^{l}\lambda_l^m H_l^m(\theta,\phi)$. In this formula, H_l^m are the basis functions and n is the representation order. The coefficients λ_l^m are computed by a stratified Monte Carlo quadrature with uniform hemisphere sampling:

$$\lambda_l^m = \frac{2\pi}{N\cdot M}\sum_{j=0}^{M-1}\sum_{k=0}^{N-1} L^i_{j,k} H_l^m(\theta_{j,k},\phi_{j,k}), \qquad (1)$$

where

$L^i_{j,k}$ is the incoming radiance from the sampled direction $(\theta_{j,k},\phi_{j,k}) = \left(\arccos(1-\frac{j+\zeta_j}{M}), 2\pi\frac{k+\zeta_k}{N}\right)$. It is computed by tracing a secondary ray from the hemisphere center in the direction $(\theta_{j,k},\phi_{j,k})$.

ζ_j, ζ_k are uniformly distributed random variables in $[0,1)$,

$N\cdot M$ is the total number of sampled directions and $N \approx 4M$.

See [Křivánek et al. 2005] for further details on radiance caching.

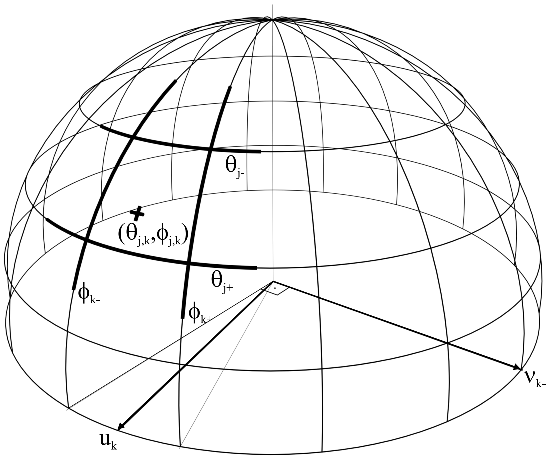


Figure 2: Uniform hemisphere subdivision for radiance sampling. The hemisphere is subdivided into rectangular cells of the same area. One random direction is selected within each cell for tracing secondary rays.

3 Novel Radiance Gradient Computation

This section presents the main contribution of this paper — a novel radiance gradient computation algorithm. The gradient computation problem consists in estimating the translational gradient $\vec{\nabla}\lambda_l^m$ for each coefficient λ_l^m from the hemisphere samples distributed according to Equation (1).

We keep the general idea of the irradiance gradient computation method of Ward and Heckbert [1992], which consists in dividing the hemisphere into cells and estimating each cell's contribution to the gradient separately. A cell's gradient is given by the marginal change of incoming radiance with the differential translation of the hemisphere center.

However, we cannot use the irradiance gradient calculation directly, for two reasons. First, we distribute the samples uniformly over the hemisphere, whereas irradiance gradients are based on uniform *projected* solid angle sampling. That is to say, the hemisphere is sampled more densely near the pole for irradiance gradients than for radiance gradients. Second, to compute the projection coefficients, we weight the incoming radiance samples by the basis functions H_l^m evaluated in the appropriate direction. No weighting is applied in the hemisphere sampling scheme used for irradiance gradients.

In the rest of this section we describe our approach to gradient computation, which relaxes the assumptions about hemisphere sampling made by Ward and Heckbert [1992] in the irradiance gradients algorithm. The stratified sampling employed in radiance caching divides the hemisphere into cells of equal solid angle, or equal area on the unit hemisphere, given by

$$A_{j,k} = (\cos\theta_{j_-} - \cos\theta_{j_+})(\phi_{k_+} - \phi_{k_-}) = \frac{1}{M}\frac{2\pi}{N}. \qquad (2)$$

The symbols relating to the geometry of the hemisphere division are illustrated in Figure 2 and summarized here:

(j,k) is the cell index.

$A_{j,k}$ is the cell area.

θ_{j_-} is the polar angle at the boundary between the current cell (j,k) and the previous cell $(j-1,k)$, $\quad \theta_{j_-} = \arccos(1-\frac{j}{M})$.

θ_{j_+} is the polar angle at the boundary between the current cell (j,k) and the next cell $(j+1,k)$, $\quad \theta_{j_+} = \arccos(1-\frac{j+1}{M})$.

ϕ_{k_-} is the azimuthal angle at the boundary between the current cell (j,k) and the previous cell $(j,k-1)$, $\quad \phi_{k_-} = 2\pi\frac{k}{N}$.

ϕ_k is the azimuthal angle at the center of the current cell (j,k), $\phi_k = 2\pi\frac{k+0.5}{N}$.

ϕ_{k_+} is the azimuthal angle at the boundary between the current cell (j,k) and the next cell $(j,k+1)$, $\quad \phi_{k_+} = 2\pi\frac{k+1}{N}$.

$\hat{u}_k$ is the unit vector in direction $(\pi/2,\ \phi_k)$.

$\hat{v}_{k_-}$ is the unit vector in direction $(\pi/2,\ \phi_{k_-}+\pi/2)$.

For each cell (j,k), we observe how its area changes with respect to the hemisphere center displacement along two near perpendicular vectors $\hat{u}_k$ and $\hat{v}_{k_-}$ defined above. The displacement along $\hat{u}_k$ causes a shift of the wall separating the considered cell (j,k) and its neighboring cell $(j-1,k)$. The induced change of the cell area is

$$\begin{aligned}\nabla_{\hat{u}_k} A_{j,k} &= \nabla_{\hat{u}_k}\theta_{j_-}\cdot\frac{\partial A_{j,k}}{\partial\theta_{j_-}}\\ &= \frac{-\cos\theta_{j_-}}{\min\{r_{j,k},r_{j-1,k}\}}\cdot\sin\theta_{j_-}(\phi_{k_+}-\phi_{k_-})\\ &= \frac{2\pi}{N}\frac{\cos\theta_{j_-}\sin\theta_{j_-}}{\min\{r_{j,k},r_{j-1,k}\}},\end{aligned}$$

where $\nabla_{\hat{u}_k}$ denotes the (scalar) directional derivative in the direction of $\hat{u}_k$, and $r_{j,k}$ is the distance from the hemisphere center to the closest surface in the sampled direction $(\theta_{j,k},\phi_{j,k})$. The derivative $\partial A_{j,k}/\partial\theta_{j_-} = \sin\theta_{j_-}(\phi_{k_+}-\phi_{k_-})$ follows directly from Equation (2).

Similarly, the displacement along $\hat{v}_{k_-}$ causes a shift of the wall separating the considered cell (j,k) and its neighboring cell $(j,k-1)$, and the induced change of the cell area is

$$\begin{aligned}\nabla_{\hat{v}_{k_-}} A_{j,k} &= \nabla_{\hat{v}_{k_-}}\phi_{k_-}\cdot\frac{\partial A_{j,k}}{\partial\phi_{k_-}}\\ &= \frac{-1}{\sin\theta_{j,k}\min\{r_{j,k},r_{j,k-1}\}}\cdot(\cos\theta_{j_+}-\cos\theta_{j_-}) \quad (3)\\ &= \frac{1}{M\sin\theta_{j,k}\min\{r_{j,k},r_{j,k-1}\}}.\end{aligned}$$

Here we used the equality $-(\cos\theta_{j_+}-\cos\theta_{j_-}) = 1/M$, which holds for uniform hemisphere sampling as follows from the way directions are generated in Equation (1).

The change of incoming radiance arriving at the hemisphere center through the cell is given by interpolating the radiance from two neighboring cells with the area used as the blending factor and is given by

$$\begin{aligned}\nabla_{\hat{u}_k} L^i_{j,k} &= \nabla_{\hat{u}_k} A_{j,k}(L^i_{j,k}-L^i_{j-1,k})\\ \nabla_{\hat{v}_{k_-}} L^i_{j,k} &= \nabla_{\hat{v}_{k_-}} A_{j,k}(L^i_{j,k}-L^i_{j,k-1}).\end{aligned}$$

The final gradient for a coefficient λ^m_l is given by summing the marginal radiance changes over all hemisphere cells weighted by the basis functions H^m_l:

$$\vec{\nabla}\lambda^m_l = \sum_{k=0}^{N-1}\Bigg[\hat{u}_k\frac{2\pi}{N}\sum_{j=1}^{M-1}\frac{\cos\theta_{j_-}\sin\theta_{j_-}}{\min\{r_{j,k},r_{j-1,k}\}}(L^i_{j,k}-L^i_{j-1,k})H^m_l(\theta_{j,k},\phi_{j,k})+ \hat{v}_{k_-}\frac{1}{M}\sum_{j=0}^{M-1}\frac{1}{\sin\theta_{j,k}\min\{r_{j,k},r_{j,k-1}\}}(L^i_{j,k}-L^i_{j,k-1})H^m_l(\theta_{j,k},\phi_{j,k})\Bigg],$$

The computed gradients lie in the tangent plane at the hemisphere center; in other words, we disregard the derivative with respect to the local Z axis. This is justified by the assumption of locally near-flat surfaces, in which case the displacements along Z are very small and therefore hardly influence the radiance change.

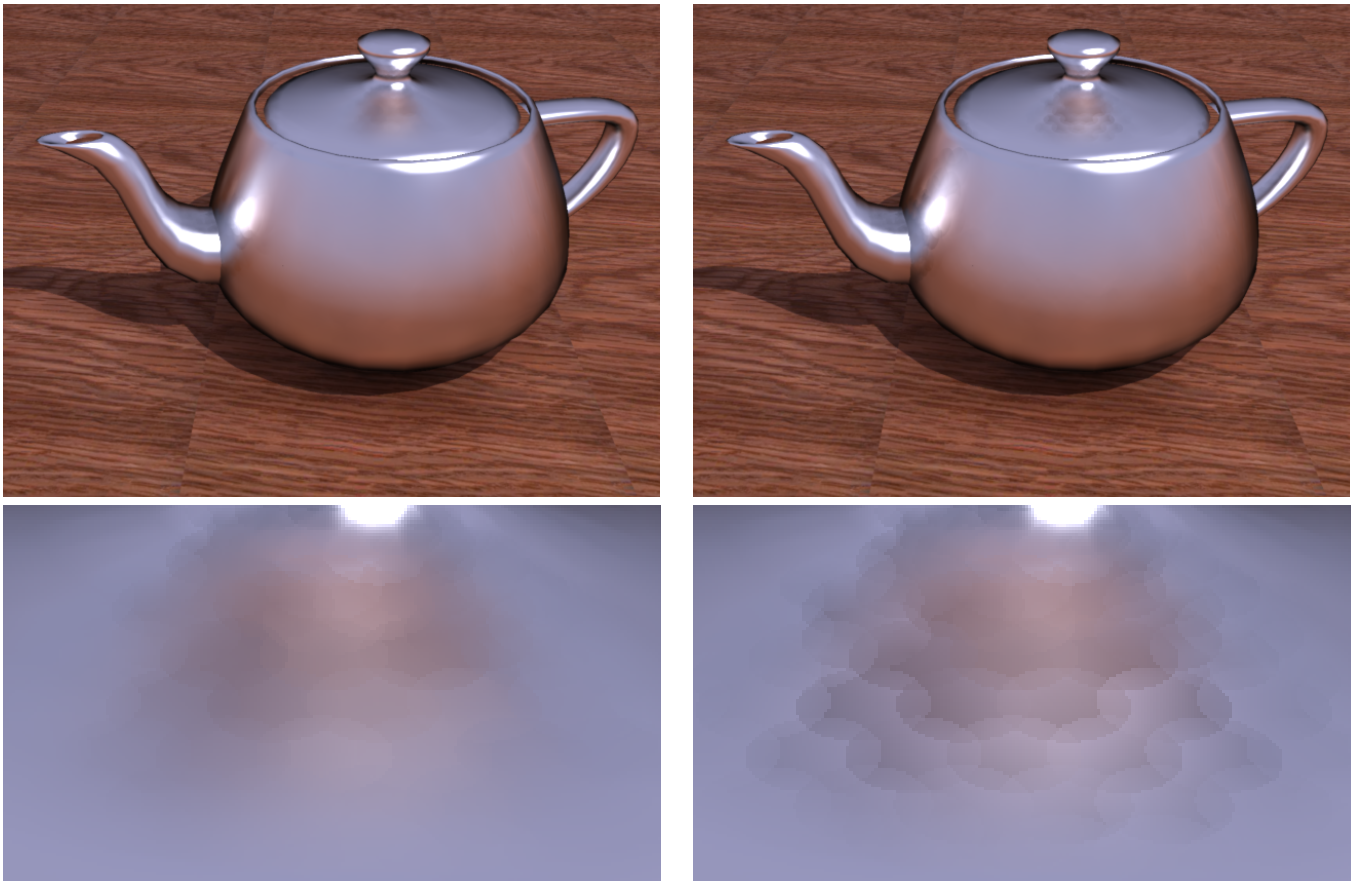
The change of occlusion is accounted for by differentiating the incoming radiance from the neighboring cells and using the $\min\{r_{j,k},r_{j,k-1}\}$ and $\min\{r_{j,k},r_{j-1,k}\}$ terms for estimating the relative wall movement between the current and the neighboring cells. The minimum of the two distances is important since "*it is always the distance to the* closer *surface that determines rate of change in occlusion*", as pointed out by Ward and Heckbert [1992]. On the other hand, the gradient computation by [Křivánek et al. 2005] treats each hemisphere cell completely independently from each other, based solely on the local geometrical information at the point where the cell's sampling ray hits another surface. Such an approach does not lend itself to estimating the occlusion changes. To sum up, the proposed formula is more accurate than the radiance gradient formula of [Křivánek et al. 2005] and more general than the irradiance gradient formula of [Ward and Heckbert 1992].

Unlike Ward and Heckbert, we do not compute rotational gradients. They are replaced by a full rotation of the incoming radiance function which is required anyway in radiance caching [Křivánek et al. 2005]. This rotation is applied *after* the application of translational gradients and therefore the translational gradients themselves do not have to be rotated — they are applied in the local coordinate frame of the radiance record before the rotation takes place.

Irradiance Gradient. The gradient formula above was derived for uniform hemisphere sampling with radiance samples weighted by the basis functions H^m_l. However, the same approach can be used to infer an *irradiance* gradient for the hemisphere sampling employed by Ward and Heckbert. Two slight changes have to be made. First, the weighting by $H^m_l(\theta_{j,k},\phi_{j,k})$ is replaced by $\cos\theta_{j,k}$. Second, as a consequence of the uniform *projected* solid angle sampling, the definition of $\theta_{j,k}$, θ_{j_-} and θ_{j_+} is different from ours and the equality $-(\cos\theta_{j_+}-\cos\theta_{j_-}) = 1/M$ does not hold anymore. Instead, the values $-(\cos\theta_{j_+}-\cos\theta_{j_-})$ have to be used directly in the final irradiance gradient formula:

$$\vec{\nabla}E = \sum_{k=0}^{N-1}\Bigg[\hat{u}_k\frac{2\pi}{N}\sum_{j=1}^{M-1}\frac{\cos\theta_{j,k}\cos\theta_{j_-}\sin\theta_{j_-}}{\min\{r_{j,k},r_{j-1,k}\}}(L^i_{j,k}-L^i_{j-1,k})+ \hat{v}_{k_-}\sum_{j=0}^{M-1}\frac{\cos\theta_{j,k}(\cos\theta_{j_-}-\cos\theta_{j_+})}{\sin\theta_{j,k}\min\{r_{j,k},r_{j,k-1}\}}(L^i_{j,k}-L^i_{j,k-1})\Bigg]$$

The θ related quantities are defined as $\theta_{j,k} = \arcsin\sqrt{\frac{j+\zeta_j}{M}}$, $\theta_{j_-} = \arcsin\sqrt{\frac{j}{M}}$ and $\theta_{j_+} = \arcsin\sqrt{\frac{j+1}{M}}$ [Ward and Heckbert 1992]. This formula yields numerically very similar results to that of Ward and Heckbert and the images generated with the two formulas are indistinguishable from each other. The computational performance of both methods is equal since all terms except from the hit distances $r_{j,k}$ and the sampled incoming radiances $L_{j,k}$ can be precomputed.



New gradients

Gradients by [Křivánek et al. 2005]

Figure 3: Indirect illumination on the teapot computed with radiance caching using gradient-based interpolation. The new gradients (on the left) provide smoother interpolation on the lid than the gradients of [Křivánek et al. 2005] (on the right).

4 Results

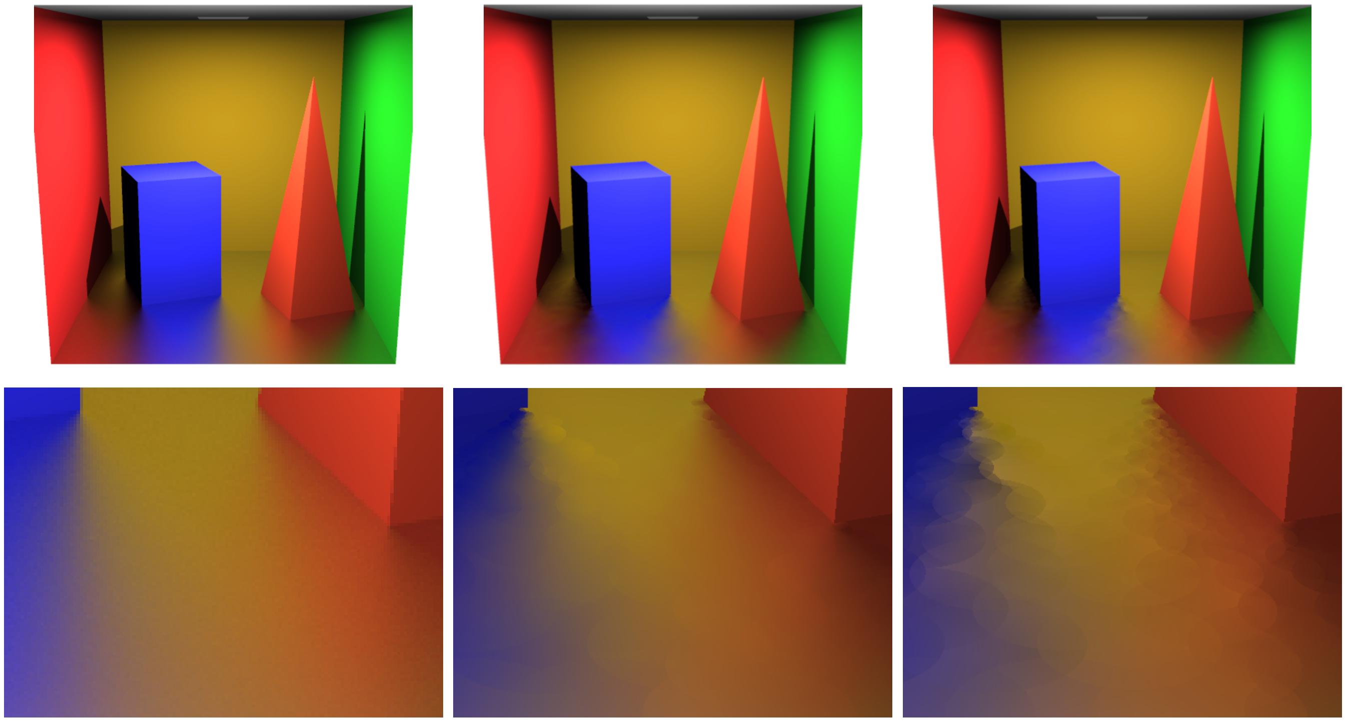
The scene in Figure 1 is a pathological case for the gradient computation of [Křivánek et al. 2005], but is handled correctly by the new gradient computation we have just presented. The scene consists of a glossy floor plane (isotropic Ward BRDF, $\rho_s = 0.9$, $\alpha = 0.15$, [Ward 1992]) with a diffuse sphere on it, which is the only source of indirect illumination for the glossy floor. Consequently, there is a large change of occlusion for the part of the floor that reflect the edge of the sphere. Since the gradient computation of [Křivánek et al. 2005] assumes continuity of the reflected environment, it does not handle such a situation correctly and the interpolation shows discontinuities (Figure 1 on the right). On the other hand, the new gradient computation does handle the occlusion change correctly and the sphere reflection on the floor is much smoother (Figure 1 on the right). Note that the interior of the sphere reflection is rendered near identically by both methods, since no occlusion changes prevent the method of [Křivánek et al. 2005] to perform well in that area. Both images took approximately same time to render.

Figures 3 and 4 illustrate again the same general observations. The method of [Křivánek et al. 2005] breaks down when a severe occlusion change is reflected on a glossy surface. In Figure 3 it is the part of the lid that reflects the lid handle (shown in the blow-up), in Figure 4 it is the floor reflecting the edges of the box and the pyramid (also shown in the blow-up). On the other hand, both methods perform similarly when no significant occlusion change is present. The teapot in Figure 3 is assigned the isotropic Ward BRDF [Ward 1992] with $\rho_s = 0.9$, $\alpha = 0.15$ and the Cornell box floor in Figure 4 uses measured metallic BRDF fit with three Lafortune BRDF lobes [Westin 2000; Lafortune et al. 1997].

5 Conclusion

We presented a new algorithm for computing translational gradient of incoming radiance function projected onto an arbitrary hemispherical basis. On the test rendering we illustrated that the radiance interpolation based on the new gradient estimate leads to smoother results without visible discontinuity artifacts. Compared to the previous method [Křivánek et al. 2005], the new gradient computation does not involve any additional computational cost and is even easier to implement, since derivatives of the basis functions do not have to be evaluated.

In future work, we would like to devise a gradient formula for localizing bases such as spherical wavelets [Schröder and Sweldens 1995], for which the gradient computation proposed here is not directly applicable because of their hierarchical, adaptive nature. This should allow us to use smooth gradient-based interpolation on surfaces with higher frequency BRDFs.



Reference solution

New gradients

Gradients by [Křivánek et al. 2005]

Figure 4: Indirect illumination on the glossy floor computed with path tracing (left) and with radiance caching using gradient-based interpolation (middle and right). Compared to the gradients of [Křivánek et al. 2005] (on the right), the new gradients (in the middle) provide smoother interpolation near the occlusion changes caused by the box and the pyramid, and provide an image visually much closer to the reference solution.

Acknowledgements

The authors thank Vlastimil Havran for letting them use his ray tracer. This work was partially supported by ATI Research and the US Office of Naval Research.

References

ANNEN, T., KAUTZ, J., DURAND, F., AND SEIDEL, H.-P. 2004. Spherical harmonic gradients for mid-range illumination. In *Proceedings of the Eurographics Symposium on Rendering 2004*.

ARVO, J. 1994. The irradiance jacobian for partially occluded polyhedral sources. In *Proceedings of SIGGRAPH '94*.

GAUTRON, P., KŘIVÁNEK, J., PATTANAIK, S. N., AND BOUATOUCH, K. 2004. A novel hemispherical basis for accurate and efficient rendering. In *Eurographics Symposium on Rendering*.

HOLZSCHUCH, N., AND SILLION, F. 1995. Accurate computation of the radiosity gradient with constant and linear emitters. In *Sixth Eurographics Workshop on Rendering*.

KŘIVÁNEK, J., GAUTRON, P., PATTANAIK, S., AND BOUATOUCH, K. 2005. Radiance caching for efficient global illumination computation. *Transactions on Visualization and Computer Graphics (accepted for publication)*. Also available as Technical Report #1623, IRISA, http://graphics.cs.ucf.edu/RCache/index.php.

LAFORTUNE, E. P. F., FOO, S.-C., TORRANCE, K. E., AND GREENBERG, D. P. 1997. Non-linear approximation of reflectance functions. In *Proceedings of SIGGRAPH '97*.

SCHRÖDER, P., AND SWELDENS, W. 1995. Spherical wavelets: efficiently representing functions on the sphere. In *Proceedings of SIGGRAPH*, ACM Press, 161–172.

WARD, G. J., AND HECKBERT, P. S. 1992. Irradiance gradients. In *Eurographics Workshop on Rendering*.

WARD, G. J., RUBINSTEIN, F. M., AND CLEAR, R. D. 1988. A ray tracing solution for diffuse interreflection. In *Proceedings of SIGGRAPH '88*, 85–92.

WARD, G. J. 1992. Measuring and modeling anisotropic reflection. In *Proceedings of SIGGRAPH '92*, ACM Press, 265–272.

WESTIN, S. H., 2000. Lafortune BRDF for RenderMan. http://www.graphics.cornell.edu/ westin/lafortune/lafortune.html.

Reprinted with permission of Eurographics

Eurographics Symposium on Rendering (2006)
Tomas Akenine-Möller and Wolfgang Heidrich (Editors)

Making Radiance and Irradiance Caching Practical: Adaptive Caching and Neighbor Clamping

Jaroslav Křivánek[1] Kadi Bouatouch[2] Sumanta Pattanaik[3] Jiří Žára[1]

[1]Czech Technical University in Prague, Czech Republic [2]IRISA – INRIA Rennes, France [3]University of Central Florida, USA

Abstract

Radiance and irradiance caching are efficient global illumination algorithms based on interpolating indirect illumination from a sparse set of cached values. In this paper we propose an adaptive algorithm for guiding spatial density of the cached values in radiance and irradiance caching. The density is adapted to the rate of change of indirect illumination in order to avoid visible interpolation artifacts and produce smooth interpolated illumination. In addition, we discuss some practical problems arising in the implementation of radiance and irradiance caching, and propose techniques for solving those problems. Namely, the neighbor clamping heuristic is proposed as a robust means for detecting small sources of indirect illumination and for dealing with problems caused by ray leaking through small gaps between adjacent polygons.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism - Rendering, Global Illumination

1. Introduction

One of the practical and widely used algorithms for computing diffuse indirect illumination is irradiance caching [WRC88, WH92, TL04]. The indirect illumination is computed only at a sparse set of points on surfaces, stored in a cache, and then interpolated elsewhere. Radiance caching [KGPB05, Kři05] generalizes irradiance caching to glossy surfaces with low-frequency BRDFs.

To faithfully represent indirect illumination with only a sparse set of values, their density must be proportional to the rate of change of indirect illumination; otherwise, interpolation artifacts may appear. In irradiance caching, the upper bound on the illumination change is estimated based on the scene geometry. Even though the actual illumination conditions are disregarded in such an interpolation error metric, it generates a record density that gracefully follows the indirect illumination changes, and provides good image quality with a relatively low number of cached values.

In radiance caching, however, estimating the rate of change of indirect illumination is more difficult. On glossy surfaces, not only the illumination characteristics, but also the sharpness of the surface BRDF and the viewing direction influence the actual rate of change of indirect illumination. It would be complicated to design an interpolation error criterion that takes all those factors into account, and the resulting formula is likely to be quite computationally involved. We believe that an adaptive algorithm that refines the density of cached values based on a simple perceptual metric is more appropriate.

In this paper we propose an adaptive algorithm for controlling density of cached values in radiance and irradiance caching that we refer to as *adaptive caching*. It starts with an initial set of cached indirect illumination values and then refines their density where necessary to eliminate interpolation artifacts. The main source of artifacts in radiance and irradiance caching are visible discontinuities at a boundary of influence areas of cached values. The proposed error criterion guiding the adaptive refinement is designed to detect those discontinuities. The criterion is, thus, perceptual in the sense that it does not primarily care about the physical correctness of the image, but about the visible artifact elimination.

With adaptive caching, the record density is adapted to the actual illumination conditions better than with the original criterion used in radiance caching [KGPB05]. If indirect illumination is simple, without sudden changes, adaptive caching generates fewer records and rendering is faster. Under complex indirect illumination, adaptive caching can generate more records than the original criterion would, but without the risk of compromising image quality by interpolation artifacts. Adaptive caching is the most effective in rendering of glossy surfaces with localized indirect illumination features, where it locally increases the record density. With adaptive caching, the record density also automatically adapts to the BRDF sharpness. Additionally, record density is relaxed under strong direct illumination, where the indirect illumination quality does not matter much.

Besides the adaptive caching algorithm, we discuss some issues one must address in a practical implementation of radiance and irradiance caching. We propose a heuristic called *neighbor clamping* and show how it is useful for solving two problems of radiance and irradiance caching: (1) missing small sources of indirect illumination and (2) artifacts caused by the ray leaking through gaps between polygons.

The rest of the paper is organized as follows: Section 2 presents the related work; Section 3 describes a common formulation of irradiance and radiance caching; Section 4 then describes the adaptive caching algorithm; Section 5 discusses some practical problems and describes the neighbor clamping heuristic; Section 6 presents the results and Section 7 concludes the work.

2. Related Work

The techniques proposed in this paper are tightly coupled with irradiance and radiance caching algorithms, both described in detail in Section 3. Ward *et al.* [WRC88] proposed the record density criterion in irradiance caching based on the estimate of the upper bound of the irradiance gradient. This has been adopted by Křivánek *et al.* [KGPB05, Kři05] for radiance caching. The adaptive record density control proposed in this paper is a simple yet practical extension of the Ward *et al.*'s criterion. Tabellion and Lamorlette [TL04] propose practical modifications of the original Ward *et al.*'s irradiance caching algorithm, discussed in detail in Section 5.2. Smyk *et al.* [SM02] increase the density of the cached values based on the gradient magnitude to better accommodate to abrupt changes in indirect illumination, similarly to the way done in the code of the Radiance lighting simulation system [War94] (described in Section 5.1 of this paper). However, this approach is not sufficient for radiance caching due to the view dependence of glossy BRDFs.

The adaptive algorithm we use for steering the density of cached values is closely related to adaptive mesh refinement in radiosity [SP94,CW93], with the difference that our representation of illumination is independent of scene geometry. In particular, our approach is similar to perceptually-driven radiosity [GH97, MPT97], where a perceptual criterion is used to control adaptive mesh refinement in hierarchical radiosity, based on the perceived smoothness of illumination.

In kernel density estimation [WHSG97], the kernel bandwidth is adjusted to reduce variance without introducing too much bias. However, samples of radiance (the photon hits) are given and fixed at the time of the density estimation. On the other hand, we take new samples as needed to perform smooth radiance reconstruction. In density control for photon maps [Suy02], the density of photons is limited by the local target density to avoid storing too many photons. The target density is proportional to pixel importance to equalize reconstruction error over the image.

3. Irradiance and Radiance Caching

In this section we give a common interpolation formula for both irradiance and radiance caching. Irradiance caching [WRC88,WH92] and radiance caching [KGPB05,Kři05] are based on the observation that indirect illumination changes slowly over a diffuse or a glossy surface. Both algorithms compute indirect illumination only at a sparse set of points on surfaces and store it in a cache. The indirect illumination computation is accelerated by reusing the cached indirect illumination through interpolation.

Cached Quantity. In irradiance caching, the cached quantity is irradiance, E, at a point, therefore the directional information about incoming radiance is discarded. In radiance caching, the directional information is retained, which allows to interpolate over surfaces with view-dependent BRDFs. The incoming radiance is represented by a coefficient vector, Λ, with respect to spherical or hemispherical harmonics [GKPB04]. The cached quantity is the only essential difference between the two algorithms; the caching scheme is the same for both.

Caching Scheme. Whenever indirect illumination needs to be computed at a point $\mathbf{p}$, the cache is queried for cached nearby indirect illumination values (called *records*). If no record is found near $\mathbf{p}$, interpolation cannot be used. In such a case the hemisphere above $\mathbf{p}$ is sampled, and irradiance E (in irradiance caching) or the coefficient vector Λ (in radiance caching) is computed and stored in the cache. If, on the other hand, cached records are found near $\mathbf{p}$, the cached indirect illumination is interpolated.

Indirect Illumination Interpolation. Here, we unify the interpolation formulas of irradiance and radiance caching by interpolating the outgoing radiance contributions:

$$L^o_{\mathscr{S}}(\mathbf{p}) = \frac{\sum_{i\in\mathscr{S}} L^o_i(\mathbf{p})w_i(\mathbf{p})}{\sum_{i\in\mathscr{S}} w_i(\mathbf{p})}, \qquad (1)$$

where $L^o_i(\mathbf{p})$ is the contribution of the i-th record to the outgoing radiance at the point $\mathbf{p}$. For irradiance caching at a point with diffuse reflectance $\rho(\mathbf{p})$, $L^o_i(\mathbf{p})$ is:

$$L^o_i(\mathbf{p}) = [E_i + (\mathbf{n}_i \times \mathbf{n})\cdot\nabla_r E_i + (\mathbf{p}-\mathbf{p}_i)\cdot\nabla_t E_i]\rho(\mathbf{p})/\pi.$$

For radiance caching, the outgoing radiance contribution is given by the dot product:

$$L_i^o(\mathbf{p}) = \left[\mathbf{R}_i\left(\Lambda_i + d_x\frac{\partial\Lambda_i}{\partial x} + d_y\frac{\partial\Lambda_i}{\partial y}\right)\right]\cdot C_{(\mathbf{p},\omega_o)}.$$

Various symbols used here are summarized below:

$\mathbf{p}$	point of interpolation.
$\mathbf{n}$	normal at $\mathbf{p}$.
$\mathbf{p}_i$	location of the i-th cache record.
$\mathbf{n}_i$	normal at $\mathbf{p}_i$.
d_x, d_y	displacements of $\mathbf{p}-\mathbf{p}_i$ along the x and y axes of the local coordinate frame at $\mathbf{p}_i$.
$w_i(\mathbf{p})$	weight of the i-th cache record with respect to point $\mathbf{p}$, defined below.
$\mathscr{S}$	set of records used for interpolation at $\mathbf{p}$, defined below.
E_i	cached irradiance value at $\mathbf{p}_i$.
$\nabla_r E_i$	cached rotational irradiance gradient at $\mathbf{p}_i$.
$\nabla_t E_i$	cached translational irradiance gradient at $\mathbf{p}_i$.
Λ_i	coefficient vector for the incoming radiance at $\mathbf{p}_i$.
$\frac{\partial\Lambda_i}{\partial x}$	cached derivative of Λ_i with respect to translation along the local x axis.
$\frac{\partial\Lambda_i}{\partial y}$	cached derivative of Λ_i with respect to translation along the local y axis.
$\mathbf{R}_i$	rotation matrix used to align the coordinate frames at $\mathbf{p}$ and $\mathbf{p}_i$.
$C_{(\mathbf{p},\omega_o)}$	coefficient vector representing the BRDF lobe at $\mathbf{p}$ for the outgoing direction ω_o.

Weighting Function. Weight of the i-th cache record with respect to point $\mathbf{p}$ is given by $w_i(\mathbf{p}) = \left(\frac{\|\mathbf{p}\quad\mathbf{p}_i\|}{R_i} + \sqrt{1-\mathbf{n}\cdot\mathbf{n}_i}\right)^{1}$. The harmonic mean distance, R_i, to the objects visible from $\mathbf{p}_i$, is computed from the ray lengths in hemisphere sampling and stored in the cache.

Interpolation Criterion. The definition of the set $\mathscr{S}$ of records used for interpolation is of particular interest for this paper. $\mathscr{S}$ is defined as $\mathscr{S} = \{i|w_i(\mathbf{p}) > 1/a\}$, where a is a user defined maximum allowed interpolation error. The higher the value of a, the bigger the allowance for interpolation and the smaller the accuracy of the final image. The definition of the set $\mathscr{S}$ means that the record i can be used for interpolation in the neighborhood of its location, where $w_i(\mathbf{p}) > 1/a$ holds. This neighborhood is larger in flat open areas, where the indirect illumination changes slowly, and smaller on curved objects and near scene geometry, where the indirect illumination changes more quickly. This adapts the density of the cached radiance values to the expected rate of change of the indirect illumination. The interpolation scheme was derived in [WRC88] for diffuse surfaces. Even though the derivation does not hold for glossy ones, the interpolation gives surprisingly good results even in radiance caching. However, in some situations it fails. This is why we have developed a perceptual error criterion to guide the record density, described in the following section.

4. Adaptive Caching

In this section we present the error criterion used to guide record density and an algorithm that uses this criterion to distribute records on object surfaces.

4.1. Error Criterion and Adaptive Refinement

Adaptive caching uses the original irradiance caching interpolation scheme described above, and an adaptive record density is achieved by *modulating the approximation error value, a, on a per-record basis*, based on our error criterion. So each record i now stores its own value of approximation error, a_i, instead of having one global value for the whole scene.

The goal of the adaptive record density control is to eliminate image artifacts caused by interpolation in places with high rate of change of indirect illumination. Those artifacts exhibit themselves as discontinuities at the boundary of influence areas of two or more records (see the cutout in Figure 7, top). The influence area of a record is defined as $\{\mathbf{p};\ w_i(\mathbf{p}) > 1/a\}$. Our error criterion reports a visible discontinuity if two records have a visibly different outgoing radiance contribution somewhere in the overlap of their influence areas. Whenever a visible discontinuity is detected, we decrease a record's value of a_i, effectively locally increasing the record density. In other words, we force the radiance contribution in the overlap areas to be indistinguishable from each other.

Consider a point $\mathbf{p}$ at which the outgoing radiance is computed with the interpolation formula (1) from a set $\mathscr{S}$ of at least two records. We say that the overlap of those records causes a discontinuity if a record i exists in $\mathscr{S}$, such that the outgoing radiance *without* this record's contribution, $L^o_{\mathscr{S}\setminus\{i\}}(\mathbf{p})$, is discernable from the outgoing radiance *with* this record's contribution, $L^o_{\mathscr{S}}(\mathbf{p})$.

If a discontinuity-causing record i is found, its contribution is excluded from the interpolated radiance at $\mathbf{p}$ by decreasing its approximation error to $a_i := \frac{\|\mathbf{p}\quad\mathbf{p}_i\|}{R_i} - \varepsilon$, where ε is a very small number (see Figure 1). This ensures that the condition $w_i(\mathbf{p}) > 1/a_i$ does not hold anymore and record i does not contribute to the interpolated radiance at $\mathbf{p}$. If there are more than one candidate for exclusion, we choose the one with the lowest weight $w_i(\mathbf{p})$, remove it from $\mathscr{S}$, and reiterate the process. Intuitively, the record with the lowest weight is relatively the farthest from point $\mathbf{p}$ and should be excluded. The metric used to assess the discernibility of two radiance values is discussed in Section 4.3.

4.2. Full Convergence

The criterion described above can be used to compute indirect illumination at a point $\mathbf{p}$ in a straightforward way:

```
$\mathscr{S} :=$ look_up_cache$(\mathbf{p},\mathbf{n})$
```

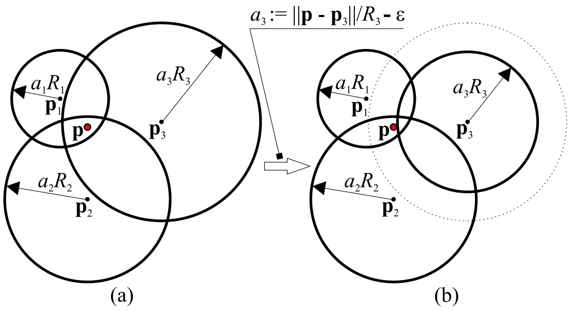


Figure 1: *(a) If the outgoing radiance $L^o_{\mathscr{S}}(\mathbf{p})$, $\mathscr{S} = \{1,2,3\}$ at the point $\mathbf{p}$ is visibly different from $L_{\mathscr{S}\setminus\{i\}}(\mathbf{p})$ for some $i \in \{1,2,3\}$, the overlap area is said to cause a discontinuity. (b) If a discontinuity is detected, the approximation error a_i of the record contributing with the minimum weight $w_i(\mathbf{p})$ is decreased (here record #3), so that point $\mathbf{p}$ be outside the record's influence area.*

If $\mathscr{S}$ is empty, add new record.
If $|\mathscr{S}| = 1$, interpolate from the single record in $\mathscr{S}$.
If $|\mathscr{S}| > 1$, run the adaptive refinement and interpolate from the records that remain in $\mathscr{S}$.

However, this approach does not work well, since a contribution from a record that will be shrunk later might have been mistakenly used earlier for interpolation. For a correct result, the record distribution must *fully converge* with respect to a set of shading points, before any record is used for interpolation at any of those points. We call *shading points* all the points with indirect illumination computed by means of radiance or irradiance caching (either by interpolation or by adding a new record). Typically those are the surface points visible through image pixels either directly or after an ideal specular reflection or refraction.

Definition. Given a set of shading points $\mathscr{P}$, we say that the *record density has fully converged* for points in $\mathscr{P}$ if:

1. each point in $\mathscr{P}$ has at least one contribution from existing records, and
2. for all points having contributions from two or more records, the error criterion does not report a visible discontinuity.

Algorithm 1 describes an efficient procedure for attaining full convergence for a given set of shading points. For the sake of efficiency, we store a list of record contributions with each shading point, so that the (ir)radiance cache does not have to be queried repetitively to find the contributing records for a shading point (line 7). Each contribution in the list stores a pointer to the contributing record, weight $w_i(\mathbf{p})$, outgoing radiance $L^o_i(\mathbf{p})$ (evaluated lazily as requested by the error criterion) and a change counter (described below).

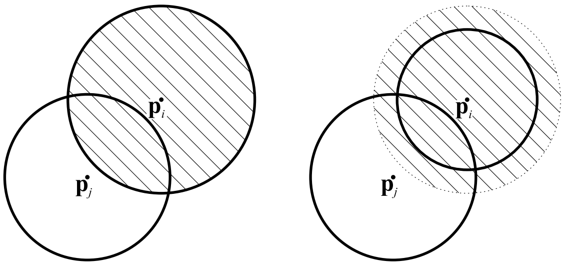
Algorithm 1 Full convergence for a set of shading points.

Input:
The set of shading points $\mathscr{P}$ for which indirect illumination is to be computed by radiance or irradiance caching. Each shading point contains position, local coordinate frame, BRDF pointer, and a *list of record contributions*, empty at start.

Ensures:
All the shading points in $\mathscr{P}$ have at least one contribution from a cache record and for all points having contributions from two or more records, the error criterion does not report a visible discontinuity.

```
 1: pointQueue.push($\mathscr{P}$)
 2: while pointQueue not empty do
 3:   // new pass starts here
 4:   while pointQueue not empty do
 5:     // for all points to be processed in this pass
 6:     $p_s$ := pointQueue.pop()
 7:     remove invalid contributions from the shading
        point $p_s$
 8:     if zero contributions in $p_s$ then
 9:       add a new record at the position of $p_s$
10:       add a contribution to each shading point in the
          new record's influence area
11:       add each point in the new record's influence area
          to pointQueue
12:     else if more than one contribution in $p_s$ then
13:       run adaptive refinement the contributing records
14:     end if
15:     // do nothing if $p_s$ has exactly one contribution
16:   end while
17:   for all records affected by refinement in this pass do
18:     add to pointQueue shading points in the record's
        influence area from the beginning of the pass
19:   end for
20:   // go to next pass
21: end while
```

The basic steps of Algorithm 1 are motivated by the two following situations, illustrated in Figure 2: (a) After a new record is added to the cache, all the shading points in its influence area should be checked again by the error criterion, hence are added to pointQueue (line 11). (b) If the approximation error a_i of the record i is decreased (line 13), some shading points might find themselves without any valid contribution. New records must then be added to the cache to cover those points. To efficiently detect the points with no valid contribution, the algorithm works in passes (the outermost while loop of Algorithm 1). At the end of each pass (lines 17-19), we loop over all records whose a_i was decreased in the current pass. For each of the affected records, all the shading points within that record's influence area *as it was at the beginning of the pass* are added to pointQueue. The possible invalid contributions due to the affected records are then removed from those points' contribution lists on



(a) - new record added at $\mathbf{p}_i$ (b) - a_i of record i reduced

Figure 2: *Shading points in the hatched areas are inserted into pointQueue in the following two situations:* ***(a)*** *After a new record is added to the cache.* ***(b)*** *At the end of each pass for each record affected by the adaptive refinement.*

line 7. New records are then possibly added to (ir)radiance cache on lines 9 to 11. The same effect could be achieved by inserting the shading points into pointQueue immediately after a record's a_i-value is decreased. However, this would not be efficient since the a single record's a_i-value is typically reduced more than once in a single pass.

Upon the reduction of a record's a_i value, some of the record's contributions may become invalid. It is thus necessary to check the validity of the entries in the contribution list (line 7) by re-evaluating the weight of a record that has changed since its contribution was last updated. To keep track of the contributions that require the validity check, we use *change counters*. Each record has a counter, incremented every time its a_i is reduced. The value of the change counter is copied to the contribution each time the validity of the contribution is verified on line 7. A contribution remains certainly valid as long as the record's counter and the contribution's counter are equal.

The whole algorithm finishes if no records have been changed in the last pass. Full convergence is ensured for all shading points at its end. In our test, the algorithm always finished in at most eleven passes. The overhead due to this algorithm and due to the evaluation of the perceptual criterion is very small (see Table 1).

There are a few important details to make the algorithm efficient:

- To avoid multiple insertions of one shading point to pointQueue, each shading point has a bit indicating its presence in the queue.
- The initial value of a for a new record is copied from the nearest existing record in the cache and multiplied by 1.4, a multiplier found empirically and tested on a wide range of scenes. Lower values lead to unnecessarily many records after the convergence; higher values do not decrease the number of records but increase the overhead of the convergence algorithm.
- Shading points are stored in a k-D tree [Jen01] to quickly locate the points in a record's influence area (lines 10 and 18).
- The contribution list for a shading point is a static array of length 6. We found that there are almost never more than six contributions for one shading point. Should this be the case, the excess contributions may safely be ignored (six is already enough). Using a dynamic linked list introduces a serious performance penalty because of the constant memory allocation and de-allocation.

This algorithm inverts the classical irradiance and radiance caching algorithm, where the basic operation is: 'given a shading point, find all contributing records and interpolate'. On the other hand, the basic operation in our convergence algorithm is: 'given a record, find all shading points it contributes to and add a contribution'. This is similar to radiance cache splatting [GKBP05] or reverse photon mapping [HHS05].

An alternative implementation of the convergence algorithm without the per-point contribution list uses a (ir)radiance cache look-up on line 7 to gather the contributions for a point p_s and leaves out line 10. Such an implementation consumes less memory, is simpler to implement, but it is significantly slower.

4.3. Discernability Metric

This section describes the metric used to assess discernability of two radiance values. After some experimentation, we have opted for the simplest metric, the Weber law. Weber law says that the minimum perceptible change in a visual signal is given by the fixed fraction of the signal. Although not conservative under all circumstances, the threshold of 2% is widely used in computer graphics (*e.g.* [WFA*05]) and we used it in all our renderings.

The input of the perceptual metric is two radiance values L_1 and L_2. L_1 is the interpolated outgoing radiance contribution from all overlapping records, plus direct illumination at $\mathbf{p}$,

$$L_1 = L^o_{\mathscr{S}}(\mathbf{p}) + L^{o_{direct}}(\mathbf{p}).$$

L_2 does not contain the contribution from the tested record, i,

$$L_2 = L^o_{\mathscr{S}\setminus\{i\}}(\mathbf{p}) + L^{o_{direct}}(\mathbf{p}).$$

The discernability metric based on the Weber law has the following form:

$$L_1 \text{ differs from } L_2 \quad \equiv \quad |Y(L_1) - Y(L_2)| > \Delta_{max},$$

where

$$\Delta_{max} = 0.02\,Y(L_1) + \max\{\sigma(L_1), \sigma(L_2)\}.$$

Y(.) denotes the luminance channel of a tri-stimulus value. Estimates of the standard deviation $\sigma(L_1)$ and $\sigma(L_2)$ are added to compensate for the randomness of L_1 and L_2, stemming from the fact that they are computed from quantities estimated by Monte Carlo hemisphere sampling. If the standard deviation was not added and only a few rays were used to sample a hemisphere when creating new cache records, it is likely that the irradiance or incoming radiance stored in neighboring records would be very different. In such a case, the criterion would constantly report visible discontinuities and lead to an excessive record density, which is undesirable.

To estimate the standard deviation of irradiance computed by Monte Carlo hemisphere sampling, an irradiance value E is computed from all the sample rays, and another value E' is computed from every second ray. Standard deviation of irradiance, estimated as $\sigma(E) = |E - E'|$ [DS04], is then used to compute the standard deviation of the outgoing radiance. We use the standard deviation of irradiance even in radiance caching with good results.

We experimented with the threshold elevation model of Ramasubramanian *et al.* [RPG99]. We decided not to use it, since visible artifacts were produced near edges, where the model predicts high threshold elevation. This is due to the fact that (ir)radiance caching artifacts have distinct structure that makes them more easily perceptible than unstructured noise for which the model has been designed.

5. Practical Issues

This section describes techniques that help making an implementation of radiance or irradiance caching practical.

5.1. Adapting Record Density by Gradient Magnitude

The source code of Radiance lighting simulation system [War94] contains a (never published) test that avoids a too low record density if the indirect illumination gradient is high. The test consists in comparing the upper bound of the translational irradiance gradient, derived from the “split sphere model” [WRC88], with the actual irradiance gradient, estimated from the hemisphere sampling [WH92]. Should the magnitude of the actual irradiance gradient exceed the supposed upper bound, the actual gradient magnitude will be used in the error metric instead of the upper bound. Technically, this is done by clamping the value of the harmonic mean distance R_i when a new record is being created:

$$\textbf{if} \quad \frac{|\nabla_t E_i|}{E_i} > \frac{1}{R_i}, \quad \textbf{then} \quad R_i := \left(\frac{|\nabla_t E_i|}{E_i}\right)^{1}. \qquad (2)$$

The use of this test substantially improves the image quality produced by irradiance caching.

We use a similar test in radiance caching:

$$\textbf{if} \quad \Delta_i > 1/R_i, \quad \textbf{then} \quad R_i := 1/\Delta_i,$$

where Δ_i is the ratio of the radiance gradient L2-norm to the L2-norm of the radiance itself:

$$\Delta_i = \frac{\sqrt{\|\frac{\partial \Lambda_i}{\partial x}\|^2 + \|\frac{\partial \Lambda_i}{\partial y}\|^2}}{\|\Lambda_i\|}.$$

This test works aggregately on the whole coefficient vector which represents the result of the whole hemisphere sampling. Therefore, the test is neither view dependent nor it takes the BRDF sharpness into account. It helps to detect the most serious cases of high gradient change, but it leaves much space for improvement to our adaptive radiance caching.

5.2. Neighbor Clamping

In this section we propose *neighbor clamping*, a heuristic used to detect geometrically small sources of indirect illumination.

The radius of the i-th record’s influence area $\{\mathbf{p};\, w_i(\mathbf{p}) > 1/a_i\}$ on a flat surface is given by the product $a_i R_i$, which follows from the definition of $w_i(\mathbf{p})$. Here R_i is the harmonic mean of distances to visible surfaces from the record position, $\mathbf{p}_i$, computed from the ray lengths during hemisphere sampling. The closer the surrounding geometry, the smaller R_i, and the higher the record density. However, because the sample rays from which R_i is computed are only a random subset of all directions in the hemisphere, small features are likely to be missed. The computed value of R_i is, then, too large and produces a too low record density. If the missed features are sources of strong indirect illumination, interpolation artifacts in the image result. Because of the randomness of ray directions, the features missed by one record might not be missed by another, which even amplifies the noticeability of the image artifacts.

Examples of features most commonly missed are steps of a staircase, or windowsills on a facade, which may be too small to keep the harmonic mean of the ray lengths low, yet too important in terms of indirect illumination to be missed. The left column of Figure 4 shows the artifacts due to the insufficient record density around a geometry feature (the steps).

Our goal is to make sure that no relevant geometry features are missed. Additionally, a geometry feature detected by the rays from one record should be detected by all nearby records, too.

Tabellion and Lamorlette [TL04] address this problem by computing the R-value of a record as the minimum of the ray lengths during the hemisphere sampling, instead of taking the harmonic mean. This increases the probability of detecting small geometry features and indeed, the step-like features are missed very rarely. However, after some experimentation we decided not to use the minimum ray length for the following reasons. Firstly, using the minimum ray length is overly sensitive to extremely small features that hardly have any importance for indirect illumination. Secondly, and more importantly, the minimum ray length on concave objects is very small, as the rays near the equator are usually extremely short. This produces an excessive record density on those objects. Tabellion solves this problem by excluding

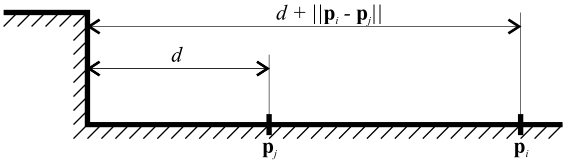


Figure 3: *The maximum possible difference of the two records' distance from the step is given by their mutual distance.*

rays near the equator, hitting other surfaces at grazing angles, from the computation of the minimum distance [Tab05]. But even with this modification, we found it difficult to obtain a decent distribution of records on concave surfaces of varying curvature by using the minimum ray length for computing R_i.

We have thus decided to compute R_i as the harmonic mean of the ray lengths, and to prevent random missing of geometry features by imposing an additional constraint on the difference of the R-values of neighboring records. The constraint stems from a basic observation on geometrical coherence in a scene. Consider a record j at position $\mathbf{p}_j$ located on a floor at a distance d from a step (see Figure 3). Now consider another record at position $\mathbf{p}_i$. By triangle inequality, the maximum possible distance of $\mathbf{p}_i$ from the step, in terms of d and the distance between the two records is $d + \|\mathbf{p}_i - \mathbf{p}_j\|$. Motivated by this observation, we never allow the R-values of two nearby records to differ by more than their mutual distance.

Technically, when a new record i is being added to the cache, we locate all nearby records j and clamp the new record's R_i value to $R_i := \min\{R_i, R_j + \|\mathbf{p}_i - \mathbf{p}_j\|\}$. After that we similarly clamp the nearby records' R-values by the new record's R_i-value. A consequence of this clamping is that a too large R-value of a record, caused by missing a geometrical feature, is clamped down by the R-value of some of the neighboring records that did not miss that feature.

This heuristic, which we call *neighbor clamping*, is fully justified when using the minimum ray length for computing R_i, but it gives very good results even for the harmonic mean. The features are almost never missed, and the overall distribution of records in the scene behaves well. Furthermore, we do not experience the problem with excessive record density on concave objects. Figure 4 demonstrates how neighbor clamping (right column) helps to detect small, step-like geometry features. Without neighbor clamping (left column), those features are often missed and artifacts appear in the image (see the detail of the stairs). Both images were rendered using approximately the same number of records (7750). Without neighbor clamping, at least 20,000 records were required to get rid of the image artifacts on the stairs.

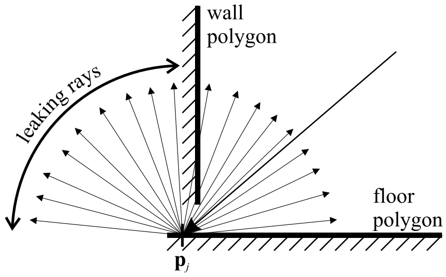


Figure 5: *Inaccurate connection of polygons may result in ray leaking.*

5.3. Ray Leaking

Irradiance and radiance caching are quite sensitive to imperfections in scene modeling; a typical example in which they break down is an inaccurate connection of adjacent edges of two polygons. This may be produced *e.g.* by an insufficient number of significant digits when a scene is exported to a text file.

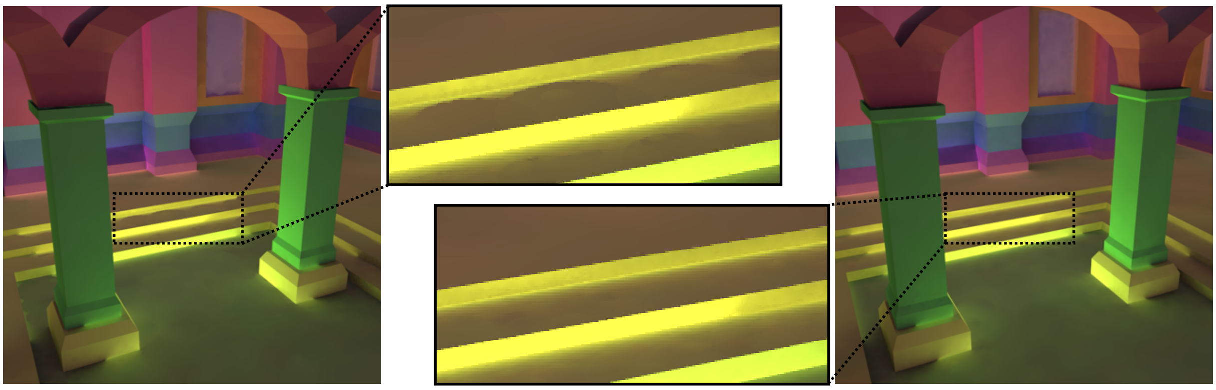
Consider the situation in Figure 5, where there is a small gap between the floor polygon and the wall polygon. If a primary ray hits this gap, its intersection with the floor polygon can be found *behind* the wall polygon. As a consequence, rays that are supposed to hit the wall now *leak* either to the neighboring room or to infinity. The outcome of such an event is quite disastrous:

- The computed irradiance or incoming radiance is wrong.
- The harmonic mean of the ray lengths is much grater than it should be; therefore, the wrong (ir)radiance is extrapolated on a very large area.

The resulting image artifacts are shown in Figure 6 on the left. The following paragraphs discuss two possible solutions to this problem, among which the more successful is to use neighbor clamping.

The first solution is to shift the origin of sampling rays away from polygon edges [TL04, Tab05]. The ray origin shifting suppresses ray leaking in some case, but is not dependable.

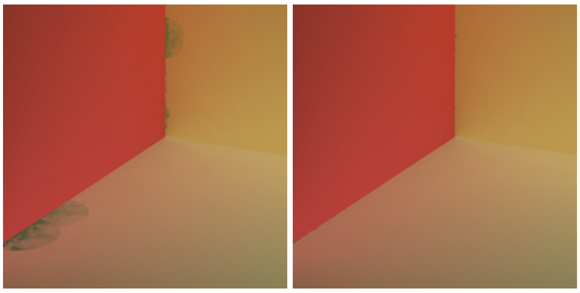
A better solution consists in using neighbor clamping. Records suffering from ray leaking have disproportionately bigger R-value than their neighbors not having this problem, thereby breaking the assumption of geometrical coherence used to derive the neighbor clamping heuristic. Therefore, ray leaking is reliably detected by the use of neighbor clamping and its consequences are alleviated by the reduction of the erroneous R-value. Figure 6 on the right shows how neighbor clamping detects and suppresses the effects of ray leaking. Besides small feature detection, ray leaking detection is another purpose that neighbor clamping serves well.



no neighbor clamping

with neighbor clamping

Figure 4: *Without our neighbor clamping, small, step-like geometry features are often missed. Missing geometry features that are strong sources of indirect illumination produces disturbing image artifacts (see the detail of the steps in the left column). Neighbor clamping helps to detect the small geometry features and suppresses the image artifacts (see the right column). The images show only indirect illumination.*



no neighbor clamping

with neighbor clamping

Figure 6: *Serious image artifacts caused by ray leaking (left) are significantly reduced by using the proposed neighbor clamping heuristic (right).*

6. Results and Discussion

Adaptive Radiance Caching. Results of adaptive radiance caching are shown in Figures 7 to 10; the rendering statistics, measured on a PC with Intel Pentium M 1.5GHz and 512MB RAM, are given in Table 1. Notice that the extra overhead introduced by adaptive caching is very small. For each scene, we rendered one image with adaptive caching and another without adaptive caching using approximately the same number of records. We compare the quality of the two renderings. Apart from the rendered images, the figures show the map of the approximation error a and the record locations. Notice how the value of a and the record density adapt to the rate of change of indirect illumination with our adaptive caching.

We do not provide a comparison with a reference solution because our error metric promotes smooth, artifact-free images, but does not ensure any error bound for the image. Our goal is to generate images that look good and are plausible, but not necessarily physically correct.

The floor of the box in Figure 7 uses the Lafortune BRDF [LFTG97] fitted to measured metallic BRDF data [Wes00]. Only direct illumination and first bounce indirect illumination for the floor is used. Non-adaptive radiance caching has problems capturing the change of indirect illumination caused by the edges of the box and the pyramid. On the other hand, the record density behind the two objects is unnecessarily high. Adaptive caching produces a record density that reproduces all the details of the indirect illumination. To obtain an artifact-free image with non-adaptive caching, at least 3,300 records were required, which is 3 times more than with adaptive caching.

The scenes in Figures 8 and 9 use four indirect bounces and both irradiance and radiance caching. The teapot in Figure 8 has a Phong BRDF with the exponent of 16. Non-adaptive caching runs into troubles capturing the reflection of the spout on the teapot body, which adaptive caching renders faithfully. In this scene, adaptive caching also increases record density on the teapot rim, where there are no noticeable artifacts even with the non-adaptive algorithm. This suggests that a better perceptual metric, which possibly takes the projected record size into account, would probably allow even lower number of records without deteriorating image quality. The teapot scene required at least 2,800 records for an artifact-free image without adaptive caching, that is 1.7 times more than with adaptive caching.

In the Walt Disney Hall scene (Figure 9), the adaptive record density is useful not only for radiance caching (cutouts 1 and 2), but also for irradiance caching. Adaptive irradiance caching nicely captures the caustics formed on the floor at the entrance to the building (cutout 3), and also delivers a better image quality on the staircase, where the non-adaptive caching leaves some smudges. Notice the slight visible discontinuity on the metal wall at the very right of the image. This is actually a discontinuity of first order, caused

	Cornell Box 800 × 800		Teapot 800 × 800		Walt Disney Hall 1280 × 800	
	adaptive ($a = 0.55$ - 0.05)	non-adaptive ($a = 0.23$)	adaptive ($a = 0.6$ - 0.05)	non-adaptive ($a = 0.15$)	adaptive ($a = 0.35$ - 0.05)	non-adaptive ($a = 0.18$)
# records	990	1010	1589	1695	4078	4144
Cache filling [s]	68.6	68.7	94.3	94.8	195	196
Overhead [s]	2.18	1.6	4.18	2.81	11.2	9.34

Table 1: *Rendering statistics for adaptive radiance caching example scenes.* Cache filling *is the time spent on creating radiance cache records (hemisphere sampling).* Overhead *is the time spent on interpolating from cache records and evaluating the perceptual error criterion. Notice that the extra overhead introduced by adaptive caching is very small.*

by the inversion of the indirect illumination gradient and by the change of the tint from brownish to bluish. Our current discernability metric does not capture those aspects of discontinuity perception. For an image quality similar to that produced by adaptive radiance caching, the non-adaptive radiance caching required as many as $29,000$ records (7 times more than adaptive caching). For non-adaptive *irradiance* caching, $17,500$ records were sufficient, which is only 1.3 times more than with adaptive irradiance caching.

Figure 10 demonstrates how record density automatically adapts to the sharpness of a BRDF. The floor is assigned the isotropic Ward-Duer BRDF [Due05, War92] with $\rho_d = 0$, $\rho_s = 0.9$ and $\alpha = 0.60$, 0.30 and 0.15 (from left to right). With increasing BRDF sharpness, the record density automatically grows to reproduce the sharper features of the indirect illumination, without requiring any user intervention. The number of records is 335, 516, and 993, respectively.

Discussion of the Radiance Caching Results. Figures 7, 8 and 9 show that adaptive radiance caching mostly pays off if there is an indirect illumination feature that requires a locally high record density. In the non-adaptive caching, the only solution to render such a feature is to increase the record density in the whole image by manually altering the global value of the approximation error a. This is inefficient, since most of the image already has a high enough record density. We would like to point out that on glossy surfaces, the localized features almost always appear; therefore, adaptive radiance caching is very profitable.

In a walkthrough animation with adaptive caching, one might observe slight popping artifacts as refinements occurs at new locations. To suppress the artifacts in high quality rendering, we first let the Algorithm 1 converge for each frame, and then perform the actual image synthesis.

Adaptive Irradiance Caching Results and Discussion. To test our adaptive algorithm with irradiance caching, we rendered a number of scenes, of which two are shown in Figure 11. We have found that the number of records and the rendering times for both adaptive and non-adaptive irradiance caching are similar. This is caused by the fact that localized indirect illumination features rarely appear on diffuse surfaces. (A notable exception are caustics, usually handled by other means than irradiance caching.) Thus, the adaptive record density does not bring as much profit to irradiance caching as it does to radiance caching.

The irradiance caching results demonstrate that extending the combination of the original Ward's error criterion [WRC88] and the adaptation by the gradient magnitude (Section 5.1) with the proposed neighbor clamping heuristic (Section 5.2) results in a robust, dependable interpolation error criterion. Within the irradiance caching framework, it would probably be difficult to find an error criterion that computes the indirect illumination with a significantly fewer records, without missing any illumination features.

7. Conclusion and Future Work

In this paper we described an adaptive algorithm for controlling record density in radiance and irradiance caching. The algorithm is based on detecting visible discontinuities caused by inadequate sampling of indirect illumination. If a discontinuity is detected, record density is locally increased in order to produce smooth, artifact free images. As a result, record density automatically adapts to the complexity of indirect illumination—fewer records are used for smoothly varying illumination and more records are generated in areas of abrupt illumination changes. Moreover, the record density adapts to the BRDF sharpness and the viewing direction. The adaptation is automatic, without the need for user intervention. We demonstrated on a number of scenes that the image quality produced by adaptive caching is superior to that generated by the error criterion of radiance caching.

We also proposed the neighbor clamping heuristic, which significantly reduces the probability of missing small geometry features. Neighbor clamping also helps to detect, and alleviate the consequences of, ray leaking—a serious problem of irradiance and radiance caching.

Adaptive caching in combination with neighbor clamping represent a step forward towards more robust radiance and irradiance caching—the algorithms are now more reliable and much less tweaking is needed for high quality results.

In future work, it would be interesting to use the results of Durand *et al.*'s frequency analysis of light transport [DHS*05] to design a more complete interpolation error criterion for glossy surfaces. Using wavelets in radiance caching for the incoming radiance representation would allow interpolation on surfaces with higher-frequency BRDFs.

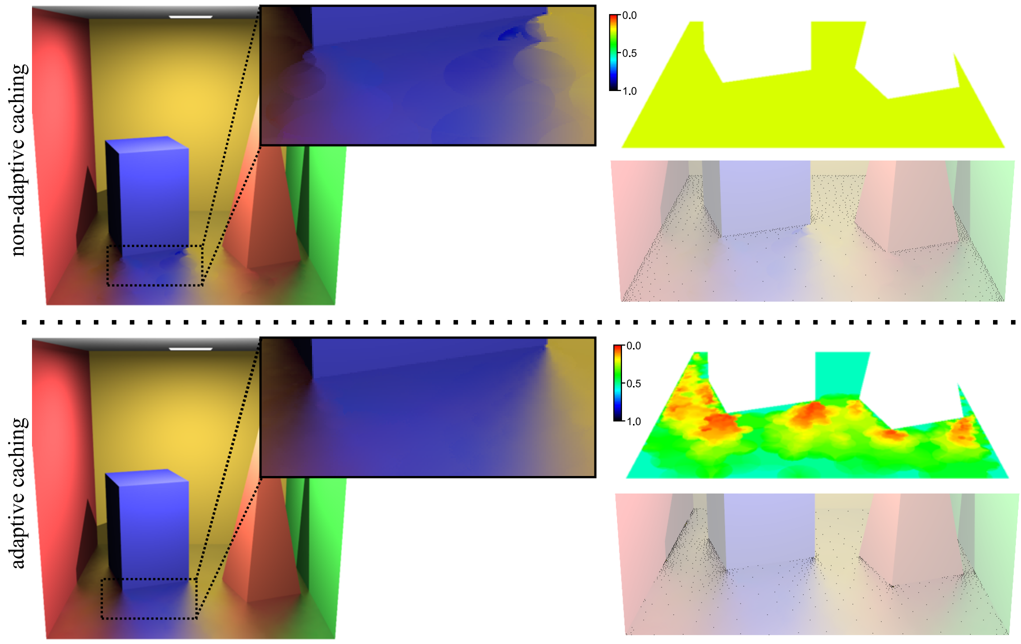


Figure 7: *Comparison of non-adaptive (top) and adaptive (bottom) caching for the Cornell box scene. The figure shows the rendered images, cutouts from the rendered images, color coded values of the approximation error a and record positions.*

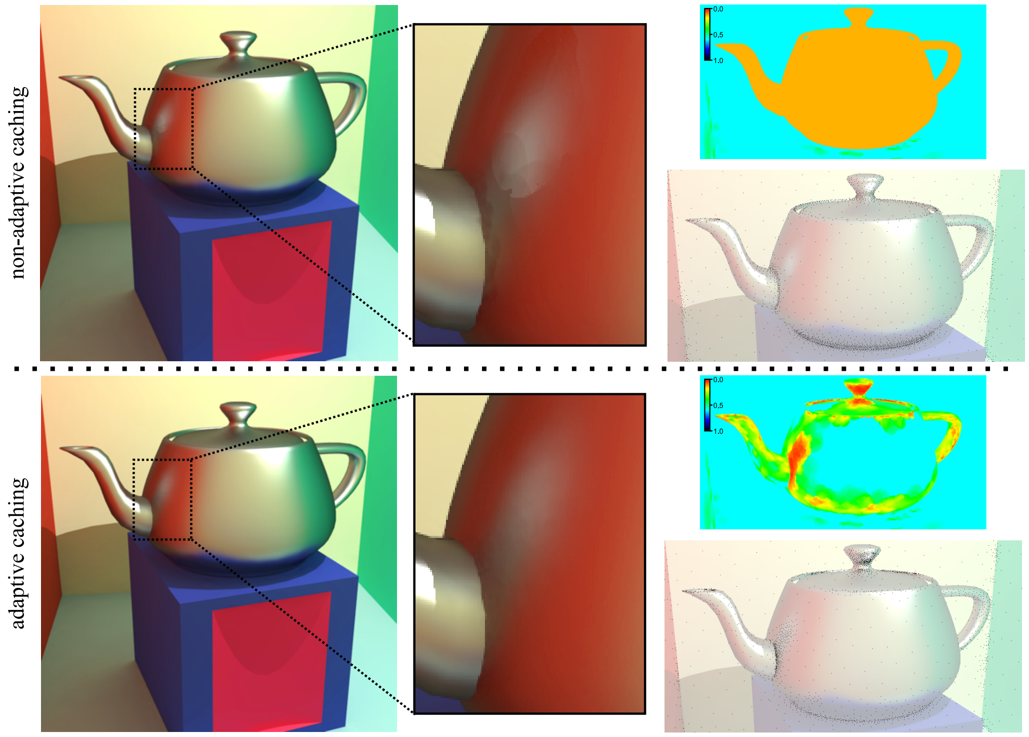


Figure 8: *Comparison of non-adaptive (top) and adaptive (bottom) caching for the Teapot scene. The figure shows the rendered images, cutouts from the rendered images, color coded values of the approximation error a and record positions.*

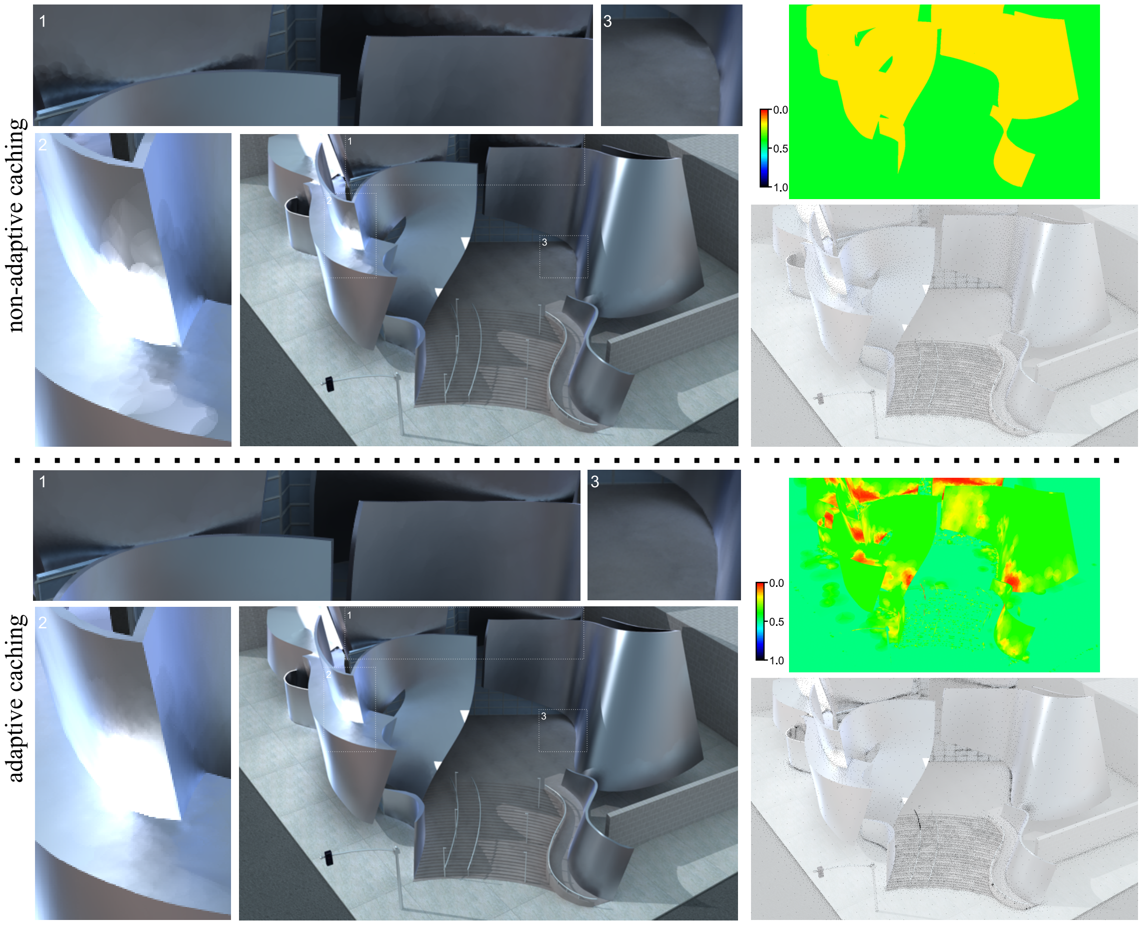


Figure 9: *Comparison of non-adaptive (top) and adaptive (bottom) caching for the Walt Disney Hall. The figure shows the rendered images, cutouts from the rendered images, color coded values of the approximation error a and record positions.*

Acknowledgements

This work has been supported by France Telecom R&D, Rennes, France, the Ministry of Education of the Czech Republic under the research programs LC-06008 (Center for Computer Graphics), and MSM 6840770014.

References

[CW93] Cohen M. F., Wallace J. R.: *Radiosity and Realistic Image Synthesis*. Morgan Kaufmann, San Francisco, CA, 1993.

[DHS*05] Durand F., Holzschuch N., Soler C., Chan E., Sillion F. X.: A frequency analysis of light transport. *ACM Trans. Graph. (Proceedings of SIGGRAPH 2005) 24*, 3 (2005).

[DS04] Dmitriev K., Seidel H.-P.: Progressive path tracing with lightweight local error estimation. In *Vision, modeling, and visualization 2004 (VMV-04)* (2004).

[Due05] Duer A.: On the ward model for global illumination. Unpublished material, 2005.

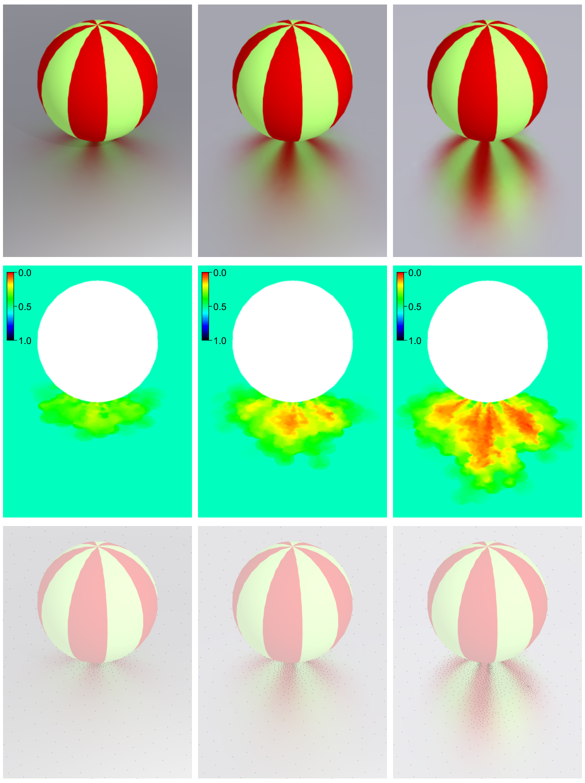
[GH97] Gibson S., Hubbold R.: Perceptually-driven radiosity. *Computer Graphics Forum 16*, 2 (June 1997).

[GKBP05] Gautron P., Křivánek J., Bouatouch K., Pattanaik S. N.: Radiance cache splatting: A GPU-friendly global illumination algorithm. In *Rendering Techniques 2005* (June 2005).

[GKPB04] Gautron P., Křivánek J., Pattanaik S. N., Bouatouch K.: A novel hemispherical basis for accurate and efficient rendering. In *Rendering Techniques 2004* (June 2004), pp. 321–330.

[HHS05] Havran V., Herzog R., Seidel H.-P.: Fast final gathering via reverse photon mapping. *Computer Graphics Forum 24* (2005).

[Jen01] Jensen H. W.: *Realistic Image Synthesis Using Photon Mapping*. A. K. Peters Ltd., Natick, MA, 2001.



$\alpha = 0.60$, #records = 335 $\alpha = 0.30$, #records = 516 $\alpha = 0.15$, #records = 993

Figure 10: *Automatic adaptation of record density to the sharpness of the BRDF. The floor is assigned the Ward-Dür BRDF with* $\rho_d = 0$*,* $\rho_s = 0.9$ *and (left to right)* $\alpha = 0.60$, 0.30*, and* 0.15.

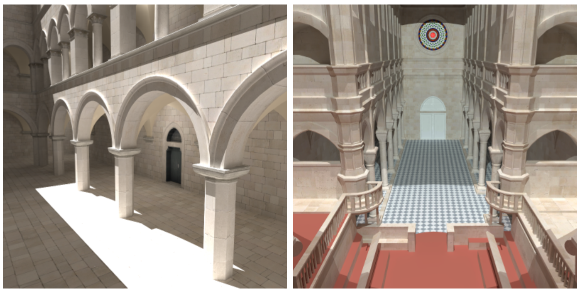


Figure 11: *Scenes used to test adaptive irradiance caching.*

[KGPB05] Křivánek J., Gautron P., Pattanaik S., Bouatouch K.: Radiance caching for efficient global illumination computation. *IEEE TVCG 11*, 5 (2005).

[Kři05] Křivánek J.: *Radiance Caching for Global Illumination Computation on Glossy Surfaces*. PhD thesis, Université de Rennes 1 and Czech Technical University, December 2005.

[LFTG97] Lafortune E. P. F., Foo S.-C., Torrance K. E., Greenberg D. P.: Non-linear approximation of reflectance functions. In *SIGGRAPH '97* (1997).

[MPT97] Martin I., Pueyo X., Tost D.: An image-space refinement criterion for linear hierarchical radiosity. In *Graphics Interface '97* (1997), pp. 26–36.

[RPG99] Ramasubramanian M., Pattanaik S. N., Greenberg D. P.: A perceptually based physical error metric for realistic image synthesis. In *SIGGRAPH '99* (1999).

[SM02] Smyk M., Myszkowski K.: Quality improvements for indirect illumination interpolation. In *Proceedings of the International Conference on Computer Vision and Graphics* (2002).

[SP94] Sillion F., Puech C.: *Radiosity and Global Illumination*. Morgan Kaufmann, 1994.

[Suy02] Suykens - De Laet F.: *On Robust Monte Carlo Algorithms for Multi-pass Global Illumination*. PhD thesis, Katholieke Universiteit Leuven, September 2002.

[Tab05] Tabellion E.: E-mail communication, 2005.

[TL04] Tabellion E., Lamorlette A.: An approximate global illumination system for computer generated films. *ACM Trans. Graph. 23*, 3 (2004).

[War92] Ward G. J.: Measuring and modeling anisotropic reflection. In *SIGGRAPH '92* (1992).

[War94] Ward G. J.: The Radiance lighting simulation and rendering system. In *SIGGRAPH '94* (1994).

[Wes00] Westin S. H.: Lafortune BRDF for RenderMan. http://www.graphics.cornell.edu/~westin/lafortune/lafortune.html, 2000.

[WFA*05] Walter B., Fernandez S., Arbree A., Bala K., Donikian M., Greenberg D. P.: Lightcuts: a scalable approach to illumination. *ACM Trans. Graph. 24*, 3 (2005).

[WH92] Ward G. J., Heckbert P. S.: Irradiance gradients. In *Proceedings of the Third Eurographics Workshop on Rendering* (1992), pp. 85–98.

[WHSG97] Walter B., Hubbard P. M., Shirley P., Greenberg D. P.: Global illumination using local linear density estimation. *ACM Trans. Graph. 16*, 3 (1997).

[WRC88] Ward G. J., Rubinstein F. M., Clear R. D.: A ray tracing solution for diffuse interreflection. In *Proceedings of SIGGRAPH '88* (1988), pp. 85–92.

Eurographics Symposium on Rendering (2005)
Kavita Bala, Philip Dutré (Editors)

Radiance Cache Splatting: A GPU-Friendly Global Illumination Algorithm

Pascal Gautron [†] Jaroslav Křivánek [‡] Kadi Bouatouch [§] Sumanta Pattanaik [¶]

Abstract

Fast global illumination computation is a challenge in several fields such as lighting simulation and computer-generated visual effects for movies. To this end, the irradiance caching algorithm is commonly used since it provides high-quality rendering in a reasonable time. However this algorithm relies on a spatial data structure in which nearest-neighbors queries and data insertions are performed alternately within a single rendering step. Due to this central and permanently modified data structure, the irradiance caching algorithm cannot be easily implemented on graphics hardware. This paper proposes a novel approach to global illumination using irradiance and radiance cache: the radiance cache splatting*. This method directly meets the processing constraints of graphics hardware since it avoids the need of complex data structure and algorithms. Moreover, the rendering quality remains identical to classical irradiance and radiance caching. Our renderer shows an implementation of our algorithm which provides a significant speedup compared to classical irradiance caching.*

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism - Rendering, Global Illumination

1. Introduction

The aim of global illumination computation is to simulate multiple interreflections of light in a scene.As computers become more and more powerful, high-quality global illumination computation gets employed in a growing number of fields, such as architectural design, cinema and video games. Generally, the computation is performed using ray tracing and Monte Carlo sampling, and is very costly. A number of methods have been proposed to reduce the computational cost of global illumination. Several approaches have been proposed to render globally illuminated scenes in real-time, such as [WBS03, GWS04, TPWG02, WS99]. However, interactive methods based on ray tracing rely on parallel processing using several computers to maintain a reasonable frame rate. An efficient approach to global il-

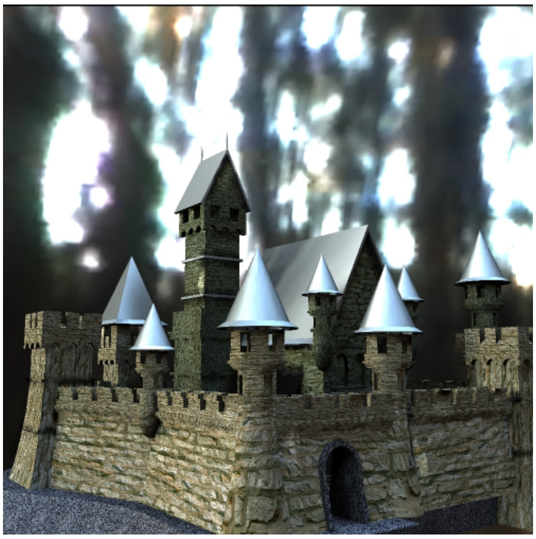


Figure 1: *The Castle scene (58K triangles) illuminated by an environment map. Our renderer computes first-bounce glossy global illumination in 10.1 s at resolution* 1000 × 1000.

[†] IRISA (Université de Rennes 1), Rennes, France - University of Central Florida, Orlando, FL, USA

[‡] IRISA (Université de Rennes 1), Rennes, France - University of Central Florida, Orlando, FL, USA - Czech Technical University , Prague, Czech Republic

[§] IRISA (Université de Rennes 1), Rennes, France

[¶] University of Central Florida, Orlando, FL, USA

lumination using ray tracing is the irradiance and radiance caching [WRC88, KGPB05]. The irradiance caching method is being employed for architectural design using the Radiance software [War04, War94].

In this paper, we propose a new method for irradiance and radiance caching which leverages graphics hardware and computes global illumination at a time order of magnitude faster than currently available caching-based methods. As shown in [TL04], first bounce global illumination takes into account most of the light transfer in a scene, and provides realistic results in most cases. Moreover, Tabellion et al. [TL04] show that a coarsely tessellated scene is sufficient to compute an accurate indirect component of the global illumination solution. Therefore, this paper focuses on the computation of first bounce global illumination in moderately complex scenes. Our method provides a fast and simple way of computing indirect lighting in a simplified geometry, while the direct lighting can be computed independently by an offline renderer using a detailed geometry.

We reformulate the irradiance and radiance caching algorithms by defining a fast image-space method based on splatting. This method makes an extensive use of graphics hardware and can be used for fast, high quality rendering or interactive visualization of globally illuminated scenes. Unlike [PBMH02, PDC*03], we avoid the need of representing and traversing complex data structures on the Graphics Processing Unit (GPU).

This paper is organized as follows: Section 2 presents previous techniques used for fast global illumination using graphics hardware. After an overview of the irradiance and radiance caching algorithms in Section 3, Section 4 presents our rendering algorithm. Section 5 details the implementation of our GPU-based renderer, and Section 6 discusses the results obtained with our algorithm in both high-quality rendering and interactive walkthroughs.

2. Related Work

Much research has been done in GPU-accelerated global illumination computation during the past years. This section describes related GPU-based global illumination algorithms.

Radiosity [GTGB84] is a classical method for global illumination computation. This approach is based on the calculation of energy transfer between all surface elements in a scene. Therefore, many visibility tests are required to perform an accurate computation, making this method very costly. Many attempts to hardware acceleration for radiosity have been developed in the last decades. Among them the hemi-cube approach [CG85, SP89] uses graphics hardware to identify the patches visible from a given patch in the scene. More recently, [CHL04] and [CHH03] propose methods for GPU-based radiosity. The former relies on texturing and visibility testing, whereas the latter uses the GPU to process the radiosity matrix.

The method described in [TPWG02] makes use of graphics hardware to display the results of global illumination computation at interactive rates. In this approach, the scene is adaptively tesselated, and the incoming radiance is computed for each vertex using parallel ray tracing. Unlike in our method, the GPU is only used for interpolating the incoming radiance across triangles using Gouraud shading. Unfortunately, high quality rendering requires to tesselate each surface into many triangles, yielding performance drop. Moreover, this method focuses only on interactive visualization: the rendering time for high quality global illumination is not improved by this approach.

In [NPG04, NPG03], Nijasure et al. propose a method for non diffuse global illumination computation using graphics hardware. The incoming radiance function at a number of locations *a priori* selected is sampled and projected into the spherical harmonics basis. Then the incoming radiance at any surface point is estimated by interpolating the incoming radiance at nearby sample locations. Although the authors demonstrate real-time performance, the main drawback of this method is the choice of sample points. In [NPG04, NPG03] these points are placed on a regular grid inside the volume of the scene, therefore not adapting to the lighting complexity.

In the Precomputed Radiance Transfer (PRT) approaches, the radiance transfer between surfaces of an object is precomputed offline and represented using spherical harmonics [SKS02, SHHS03] or wavelets [LSSS04, WTL04]. Using this precomputed information, the global illumination solution can be computed and displayed at interactive rates using the GPU. Although the PRT approaches allow real-time, high quality relighting, they rely on a costly precomputation which prevents from using them easily in complex dynamic scenes.

Wand and Straßer [WS03] describe a GPU-based method for real-time caustics computation. This algorithm relies on the selection of sample points on glossy surfaces. Each sample point is considered as a pinhole camera that projects the incoming light on diffuse receiver surfaces. This method handles several dynamic light sources and objects at interactive rates, but speed drops quickly as quality improves.

The Reflective Shadow Maps method [DS05] aims at computing first-bounce global illumination in realtime. This approach is based on an extension of shadow mapping, in which each element of the shadow map stores the incoming light flux. Although the this method offers realtime performance, it does not consider occlusion for the computation of indirect lighting, then yielding physically incorrect results.

Several attempts have been made to compute global illumination using GPU-based ray tracing. These methods rely on the versatility of programmable graphics hardware and use fragment shaders to perform ray-primitive intersections [CHH02, PBMH02]. The work described in [PBMH02] has been extended to photon map [Jen01] rendering [PDC*03].

In addition, another photon map rendering method is presented in [MM02]. Those approaches suffer from the same drawback: the GPU architecture does not allow to handle complex data structures such as trees, which are commonly used in ray tracing optimization and photon map storage. Therefore, the photon map is stored in a regular grid [PDC*03], or in a costly hash table [MM02]. The related nearest-neighbors queries have been simplified to meet the data structure and GPU constraints, yielding quality or performance drop.

Three other approaches for GPU-accelerated photon map rendering have been proposed. Larsen et al. [LC04] use graphics hardware to perform the costly final gathering: the photon map is built on the Central Processing Unit (CPU) using the classical method defined in [Jen01]. For each surface, an "approximate illumination map" is built using the data contained in the photon map. The GPU is used to perform final gathering and caustics filtering. In this paper, we take advantage of this approach to accelerate global illumination computation. The approaches presented in [SB97] and [LP03] use the GPU for irradiance reconstruction: each photon is rendered as a textured quadrilateral. The corresponding texture represents the kernel function for the photon. Although those methods show encouraging results, they are bounded by the large number of photons required to render a high quality image.

Besides hardware acceleration, many other methods have been proposed to speed up global illumination computation. Among them, the approaches based on the storage and the interpolation of incoming radiance provide fast and accurate results. Such methods include the photon maps [Jen01] and the shading cache [TPWG02]. The irradiance caching [WRC88] provides a fast and accurate way of computing indirect diffuse interreflections. [KGPB05] proposes the radiance caching, an extension of irradiance caching for the computation of indirect glossy lighting. This latter uses hemispherical harmonics [GKPB04] to represent the incoming radiance function and account for view-dependent BRDFs.

The method proposed in this paper reformulates the irradiance and radiance caching algorithms to allow an easy and efficient GPU implementation.

3. Irradiance and Radiance Caching Overview

Due to the similarity between irradiance and radiance caching, we refer to these algorithms as *(ir)radiance* caching in the remainder of this document. The (ir)radiance caching algorithms are based on the following observation: "the indirect illuminance tends to change slowly over a surface" [WRC88]. Therefore, these methods exploit spatial coherence by sparsely sampling and interpolating indirect incoming radiance. For each sample point, an *(ir)radiance record* stores the sampled incoming radiance. The records are stored in the *(ir)radiance cache*. If a point $\mathbf{p}$ in the scene is surrounded with a set of (ir)radiance records S_r, the indirect incoming lighting at point $\mathbf{p}$, $E(\mathbf{p})$, can be estimated by Eq. (1) [WRC88].

$$E(\mathbf{p}) = \frac{\sum_{k \in S_r} w_k(\mathbf{p}) E_k}{\sum_{k \in S_r} w_k(\mathbf{p})} \tag{1}$$

where E_k is the computed incoming lighting at $\mathbf{p}$ and $w_k(\mathbf{p})$ is the weighting function of record k evaluated at $\mathbf{p}$ (see the next section for the definition of w_k). In the case of irradiance cache, $E(\mathbf{p})$ represents the irradiance at point $\mathbf{p}$. For radiance cache, $E(\mathbf{p})$ stands for the incoming radiance function. The record set S_r is computed by querying the (ir)radiance cache. In order to optimize the rendering speed, a spatial data structure such as an octree is used to represent the (ir)radiance cache. More details on (ir)radiance caching and incoming radiance interpolation can be found in [WH92, War94, KGPB05, KGBP05].

4. Our Algorithm

In this paper, our aim is to reformulate the (ir)radiance caching algorithm to take advantage of the GPU computing power for first-bounce global illumination. The GPUs are SIMD (Single Instruction Multiple Data) processors. Such processors cannot handle efficiently complex data structures such as octrees. Therefore, we propose a fast rendering algorithm which avoids the need for nearest-neighbors queries and spatial data structures in the (ir)radiance caching. Moreover this approach aims at reducing the CPU workload by performing most of the computation on the GPU. The core of our approach is the *radiance cache splatting*, which determines the contribution of each record to the indirect lighting of visible objects. The radiance cache splatting and the whole rendering algorithm are described hereafter. For the sake of clarity, our algorithm is first presented in the case of irradiance caching. If necessary, specific details about the extension to radiance caching are given at the end of each subsection.

4.1. Radiance Cache Splatting

As described in Section 3, the irradiance caching algorithm relies on the computation and the interpolation of irradiance records. For a point visible through a pixel, the irradiance caching determines which records contribute to the indirect lighting of this point. The radiance cache splatting uses the opposite approach: for a given record, our algorithm determines which visible points it contributes to by splatting the record on the image plane. The result of radiance cache splatting is stored in the *radiance splat buffer*, which has the same size as the frame buffer. Each pixel $SPLATBUFF(x,y)$ of the radiance splat buffer is a pair (L_o, w), where L_o is the sum of the weighted contribution of each record, and w is the cumulated weight.

As described above, the radiance cache splatting (Algorithm 1) is designed for the computation of the contribution of an irradiance record to the indirect lighting of visible points. Our approach is based on the equation used in the irradiance caching interpolation scheme (Eq. (1)). The weight allocated to record k at point $\mathbf{p}$ with normal $\mathbf{n}$ is defined in [WRC88] as:

$$w_k(\mathbf{p}) = \frac{1}{\frac{\|\mathbf{p}-\mathbf{p}_k\|}{R_k} + \sqrt{1-\mathbf{n}\cdot\mathbf{n}_k}} \qquad (2)$$

where $\mathbf{p}_k$, $\mathbf{n}_k$ and R_k are respectively the location of record k, its normal and the harmonic mean distance to the objects visible from $\mathbf{p}_k$. The user-defined value a represents the accuracy of the computation. This value is used to threshold the weighting function: record k contributes to the estimate of the outgoing radiance at point $\mathbf{p}$ if and only if

$$w_k(\mathbf{p}) \geq \frac{1}{a} \qquad (3)$$

Substituting Eq. (3) into Eq. (2) and assuming $\mathbf{n} = \mathbf{n}_k$, one can see that record k can contribute to the estimate of the outgoing radiance at point $\mathbf{p}$ only if:

$$\|\mathbf{p}-\mathbf{p}_k\| \leq aR_k \qquad (4)$$

Therefore, Eq. (4) guarantees that a record k cannot contribute to the outgoing radiance of a point outside a sphere I_k centered at $\mathbf{p}_k$, with radius $r_k = aR_k$.

Algorithm 1 Radiance cache splatting

```
Let $k = \{\mathbf{p}_k, \mathbf{n}_k, E_k, R_k\}$ be the considered record
Determine the bounding box of $I_k$ on the image plane
for all pixel $P(x,y) = \{\mathbf{p}, \mathbf{n}, \rho_d\}$ in the bounding box do
    // Evaluate weighting function at $\mathbf{p}$
    $w = \frac{1}{\frac{\|\mathbf{p}-\mathbf{p}_k\|}{R_k} + \sqrt{(1-\mathbf{n}\cdot\mathbf{n}_k)}}$
    if $w \geq \frac{1}{a}$ then
        //Compute the contribution of record k at point $\mathbf{p}$
        $E'_k = E_k(1 + \mathbf{n}_k \times \mathbf{n} \cdot \nabla_r + (\mathbf{p}-\mathbf{p}_k) \cdot \nabla_t)$
        // Compute the outgoing radiance
        $L_o = \rho_d E'_k$
        // Accumulate into the radiance splat buffer
        $SPLATBUFF(x,y).L_o += wL_o$
        $SPLATBUFF(x,y).w += w$
    end if
end for
```

Given a camera, the *radiance cache splatting* splats the sphere I_k onto the image plane (Figure 2). The weighting function (Eq. (2)) is evaluated for each point visible through pixels covered by I_k. Then, the weight is tested against the accuracy value (Eq. (3)). For each pixel passing this test, our algorithm computes the contribution of record k to the outgoing radiance estimate.

The outgoing radiance contribution L_o to a point $\mathbf{p}$ as seen through a pixel is obtained by evaluating the following integral:

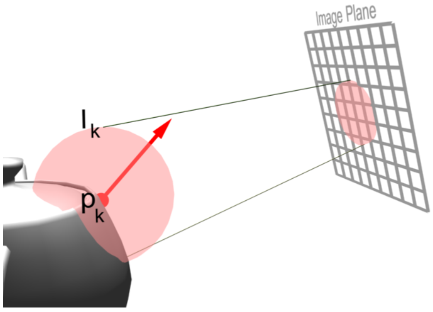


Figure 2: *The sphere I_k around the position $\mathbf{p}_k$ of the record k is splatted on the image plane. For each point within the sphere splat, the contribution of record k is accumulated into the radiance splat buffer.*

$$L_o(\mathbf{p}, \omega_o) = \int_H L_i(\mathbf{p}, \omega_i) f(\omega_o, \omega_i) \cos(\theta_i) d\omega_i \qquad (5)$$

where ω_i and ω_o are respectively the incoming and outgoing directions. $L_i(\mathbf{p}, \omega_i)$ is the radiance incoming at $\mathbf{p}$ from direction ω_i. $f(\omega_o, \omega_i)$ is the surface BRDF evaluated for the directions ω_i and ω_o. In the case of irradiance caching, we only consider diffuse interreflections. Therefore, Eq. (5) simplifies to:

$$L_o(\mathbf{p}) = \rho_d \int_H L_i(\mathbf{p}, \omega_i) \cos(\theta_i) d\omega_i = \rho_d E(\mathbf{p}) \qquad (6)$$

where ρ_d is the diffuse surface reflectance, and $E(\mathbf{p})$ is the irradiance at point $\mathbf{p}$. Therefore, the contribution of record k to the outgoing radiance at point $\mathbf{p}$ is

$$L_o = \rho_d E'_k(\mathbf{p}) \qquad (7)$$

where $E'_k(\mathbf{p})$ is the irradiance estimate of record k at point $\mathbf{p}$. This estimate is obtained using irradiance gradients:

$$E'_k = E_k(1 + \mathbf{n}_k \times \mathbf{n} \cdot \nabla_r + (\mathbf{p}-\mathbf{p}_k) \cdot \nabla_t) \qquad (8)$$

where ∇_r and ∇_t are respectively the rotational and translational gradients for record k. The computation of irradiance gradients is detailed in [WH92, KGBP05].

Note that our splatting approach can be used with any weighting function containing a distance criterion. In this paper we focus on the weighting function defined in [WRC88], although the function proposed in [TL04] could also be employed.

Extension to Radiance Caching The BRDFs of glossy surfaces are view-dependent. Therefore, Eqs. (6) and (8) cannot be used in this case. As described in [KGPB05], both the incoming radiance contribution and cosine-weighted BRDF are represented using hemispherical harmonics. Due to the basis functions orthonormality [GKPB04], Eq. (5) reduces

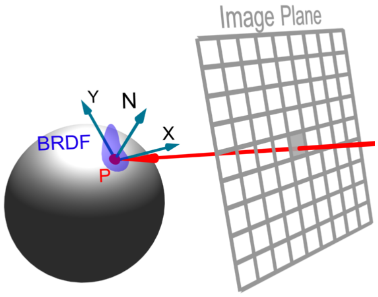


Figure 3: *The radiance cache splatting requires per-pixel information about geometry and materials: the hit point, local coordinate frame, and BRDF.*

to:

$$\begin{aligned} L_o(\mathbf{p},\omega_o) &= \int_H L_i(\mathbf{p},\omega_i) f(\omega_o,\omega_i)\cos(\theta_i)d\omega_i \\ &= \sum_{l=0}^{n-1}\sum_{m=-l}^{l} c_l^m(\omega_{out})\lambda_l^{m'}(\mathbf{p}) \qquad (9) \end{aligned}$$

where l is the order of the projection used for BRDF and incoming radiance representation. c_l^m and $\lambda_l^{m'}(\mathbf{p})$ are respectively the projection coefficients of the BRDF and the projection coefficients of the incoming radiance of record k interpolated at point $\mathbf{p}$. As described in [KGPB05, KGBP05] this estimate is obtained by applying translational gradients to the incoming radiance stored in record k. Then, since the local coordinate frames at points $\mathbf{p}_k$ and $\mathbf{p}$ might differ, we use hemispherical harmonics rotation to align the incoming radiance estimate with the local frame at $\mathbf{p}$. See [GKPB04] and [KGPB05, KGBP05] for details about hemispherical harmonics rotation and incoming radiance estimation.

Using the radiance cache splatting, the contribution of a record to the outgoing radiance estimate at points visible from the current camera is computed independently of other records. The outgoing radiance contribution of each cache record is accumulated in the radiance splat buffer. The process of generating the final image uses the contents of the radiance splat buffer to display the global illumination solution.

4.2. Indirect Lighting Rendering

The final indirect lighting image is generated in five main steps (Algorithm 2). Given a camera, the first step consists in obtaining per-pixel information about viewed objects: their position, local coordinate frame and BRDF (Figure 3).

In the second and third steps, the rendering process determines where new (ir)radiance records are necessary to achieve the user-defined accuracy of indirect illumination computation. In Step 2, each existing record (possibly computed for previous frames) is splatted onto the splat buffer using the procedure described in Section 4.1. Step 3 consists in reading back the radiance splat buffer into the CPU

Algorithm 2 Indirect lighting rendering

```
// Step 1
Generate geometric and reflectance data of objects viewed through pixels (GPU)
Clear the splat buffer
// Step 2
for all cache records do
    // The radiance cache is empty for the first image,
    // and non empty for subsequent images
    Algorithm 1: splat the records onto the radiance splat buffer (GPU)
end for
// Step 3
Read back the radiance splat buffer from GPU to CPU
// Step 4
for all pixels $(x,y)$ in the radiance splat buffer do
    if $SPLATBUFF(x,y).w < a$ then
        Compute a new incoming radiance record at corresponding hit point (GPU/CPU): see Section 5.2 for technical details
        Apply Algorithm 1: splat the new record (CPU)
    end if
end for
// Step 5
for all cache records do
    Apply Algorithm 1: splat all the newly generated records (GPU)
end for
//Normalize the radiance splat buffer (GPU)
for all pixels $(x,y)$ in the radiance splat buffer do
    $SPLATBUFF(x,y).L_o / = SPLATBUFF(x,y).w$
end for
Combine the radiance splat buffer with direct lighting (GPU)
```

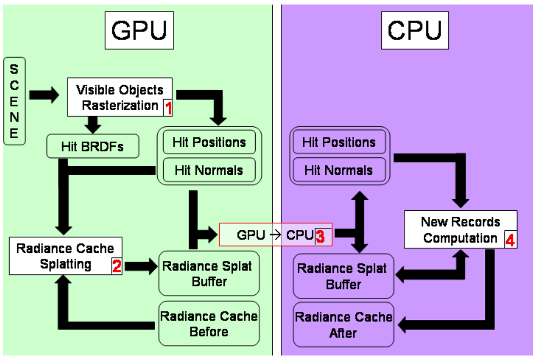


Figure 4: *The irradiance cache filling process. The numbers show the steps defined in Algorithm 2. During this process, the irradiance cache stored on the CPU is updated whereas the copy on the GPU remains untouched.*

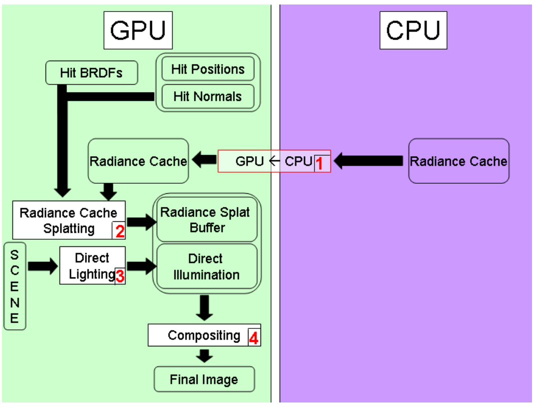


Figure 5: *The final rendering task. The numbers show the processing order described in steps 4 and 5 of Algorithm 2.*

memory. In Step 4, the algorithm traverses the radiance splat buffer to determine where new irradiance records are required to achieve the user-defined accuracy. For each pixel (x,y) in the radiance splat buffer, the cumulated weight is tested against the accuracy value a:

$$SPLATBUFF(x,y).w < a \tag{10}$$

If a pixel (x,y) passes this test, the existing cache records are insufficient to achieve the required accuracy. Therefore, a new record is generated at the location visible from pixel (x,y), and is splatted by the CPU onto the radiance splat buffer (Figure 4).

It should be noted that unlike previous approaches, the radiance cache splatting method do not rely on a specific traversal algorithm to ensure smooth interpolation. Therefore, the traversal of the radiance splat buffer can be performed linearly without introducing interpolation artifacts.

Once $SPLATBUFF(x,y).w \geq a$ for every pixel, the data stored in the cache can be used to display the indirect illumination according to the accuracy constraint. At that time in the algorithm, the irradiance cache stored on the CPU memory differs from the cache stored on the GPU: the copy on the GPU represents the cache before the addition of the records described above, while the copy on the CPU is up-to-date.

The last rendering step is the generation of the final image using the cache contents (Figure 5). The (ir)radiance cache on the GPU is updated, then the radiance cache splatting algorithm is applied on each newly generated cache record. Hence the radiance splat buffer contains the cumulated record weight and outgoing radiance contribution of all the (ir)radiance records. Then, as described in Eq. (1), the cumulated contribution of each pixel cumulated weight. This process yields an image of the indirect lighting in the scene from the current point of view. Combined with direct lighting, this method generates a fast, high quality global illumination solution.

As described above, our algorithm no longer relies on complex data structure for cache record storage. Moreover, the nearest-neighbors queries are replaced by simple sphere splatting and weighting function evaluation. These properties make the radiance cache splatting well-suited for GPU implementation.

5. Implementation

Our algorithm has been implemented using OpenGL and the OpenGL Shading Language (GLSL). This section gives technical information about the implementation of (ir)radiance cache splatting. After detailing the overall rendering algorithm, we discuss the use of GPU for (ir)radiance records computation.

5.1. Rendering Algorithm

As shown in Figures 4 and 5, our rendering algorithm makes intensive use of the GPU computational power. In this implementation, we have chosen to rely on the basic capability provided by graphics hardware: fast geometric primitive rasterization along with vertex and fragment processing. Hence the data required by our algorithm (Figure 3) is generated by rasterizing the scene geometry on the GPU using dedicated shaders. Figure 3 shows that this step includes the storage of per-pixel BRDF. In the case of diffuse BRDFs, the information consists of the diffuse reflectance of the material (i.e. one RGB value). In the case of glossy BRDFs, the per-pixel information is only one identifier. During splatting, this identifier is used to fetch the corresponding BRDF. The BRDFs are stored in a texture using the method described in [KSS02].

The (ir)radiance cache splatting can be performed either using the GPU or the CPU. As described in 4.2, our implementation uses both depending on the current context. The (ir)radiance cache splatting on the GPU is performed by drawing a quadrilateral tightly bounding the splatted sphere on the image plane. The position and size of the quadrilateral are computed using a vertex shader. Then, each fragment is processed so that its value represents the record's weighted contribution. The fragment is accumulated in the radiance splat buffer using the floating-point blending capabilities provided by graphics hardware. The final normalization (i.e. the division by the cumulated weight) is performed using a fragment shader in an additional pass, for the final display. The GPU implementation is either used to splat the whole (ir)radiance cache before adding new records, or to display the final image.

The traversal of the radiance splat buffer is performed on the CPU. Consequently, the GPU-based splatting cannot be used efficiently during the record addition step, since it would require to read back the radiance splat buffer from GPU memory once per added record. Hence our program needs a CPU implementation of (ir)radiance cache splatting. While this implementation is designed the same way as for the GPU, Figure 3 shows that the (ir)radiance cache splatting requires information about the visible objects. This information is computed on the GPU, then read back to the CPU's memory once per frame. The overhead introduced by the data transfer from GPU to CPU turns out to be very small when using PCI-Express hardware.

Once all necessary records are computed and splatted on the radiance splat buffer, the final picture containing both direct and indirect lighting has to be generated. In our implementation, the direct lighting computation is carried out by the GPU: a fragment shader evaluates the BRDFs per-pixel, while shadow maps [Wil78,BP04] are used to simulate shadowing effects. The normalization of the radiance splat buffer and the combination with direct lighting is finally done in a fragment shader before displaying.

5.2. Record computation

The incoming (ir)radiance associated with a record is generated using both CPU and GPU. As in [SP89, LC04], our renderer uses the rasterization engine to sample the hemisphere above each record position in order to compute the incoming radiance and gradients. To compensate the incomplete hemisphere coverage Larsen et al. [LC04] divide the obtained irradiance by the plane coverage ratio. We propose a more accurate method, in which border pixels are extended so that they fill the remaining solid angle (Figure 6). Hence the incoming radiance is considered constant within the extension of each pixel. In our test scenes, this method yields more accurate results than the approach of Larsen et al.(Table 1). Moreover, a key aspect of this method is that the directional information of the incoming radiance remains plausible. Therefore, indirect glossy lighting can also be rendered correctly.

	Plane sampling	[LC04]	Our method
RMS Error	18.1%	10.4%	5.8%

Table 1: *RMS error of 10000 irradiance values computed in the Sibenik Cathedral scene. Our method yields more accurate results than previous approaches, while preserving the directional information of the incoming radiance.*

As shown in [LC04] the irradiance is defined by a weighted sum of the pixels of the sampling plane. This sum can be calculated either using the GPU and automatic mipmap generation [LC04], or using frame buffer readback and CPU-based summation (Figure 7). Due to the high transfer rates provided by PCI-Express, CPU-based computation turned out to be faster on the computer we used. However, the PCI-Express architecture does not avoid pipeline stalls: the GPU remains idle during the readback and during the computation of the records on the CPU. Future work will consider multithreading and scheduling for enhanced performance.

The same approach is used to compute the incoming radiance function for radiance cache records. Instead of computing the irradiance, we project the pixel values onto the hemispherical harmonics basis using the CPU.

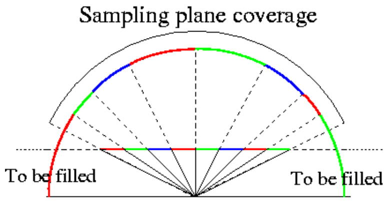


Figure 6: *The hemisphere sampling is replaced by rasterizing the scene geometry on a sampling plane. Since this plane does not cover the whole hemisphere, we use a border compensation method to account for the missing directions. Border pixels are extended to avoid zero lighting coming from grazing angles, yielding more plausible results.*

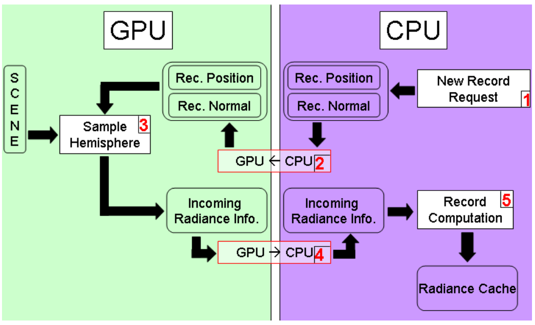


Figure 7: *New record computation process. The numbers show the order in which the tasks are processed.*

6. Results

This section discusses the results obtained using our implementation of radiance cache splatting. The images and timings presented here have been generated using an NVIDIA Quadro FX 3400 PCI-E and a 3.6 GHz Pentium 4 CPU with 1 GB RAM.

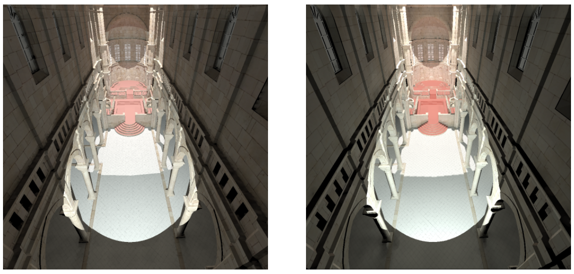
6.1. Fast High Quality Rendering

In this section, our aim is non-interactive, high quality rendering of global illumination. First, we compare the results obtained with our GPU-based renderer with the well-known Radiance [War04] software in the context of irradiance caching. Second, we discuss the results of radiance caching in glossy environments using our renderer.

We have compared the rendering speed of the Radiance software and our renderer in diffuse environments: the *Sibenik Cathedral* and the *Sponza Atrium* (Figure 9). The images are rendered at a resolution of 1000 × 1000 and use a 64 × 64 resolution for hemisphere rasterization. The results are discussed hereafter, and summarized in Table 2.

a) Sibenik Cathedral This scene contains 80K triangles, and is lit by two light sources. The image is rendered with an accuracy parameter of 0.15. At the end of the rendering process, the irradiance cache contains 4076 irradiance records. The radiance cache splatting on the GPU is performed in 188 ms. The Radiance software rendered this scene in 7 min 5 s while our renderer took 14.3 s, yielding a speedup of about 30×.

b) Sponza Atrium This scene contains 66K triangles and two light sources. Using an accuracy of 0.1, this image is generated in 13.71 s using 4123 irradiance records. These records are splatted on the GPU in 242.5 ms. Using the Radiance software with the same parameters, a comparable image is obtained in 10 min 45 s. In this scene, our renderer proves about 47× faster than the Radiance software.



(a) Radiance (b) Our renderer

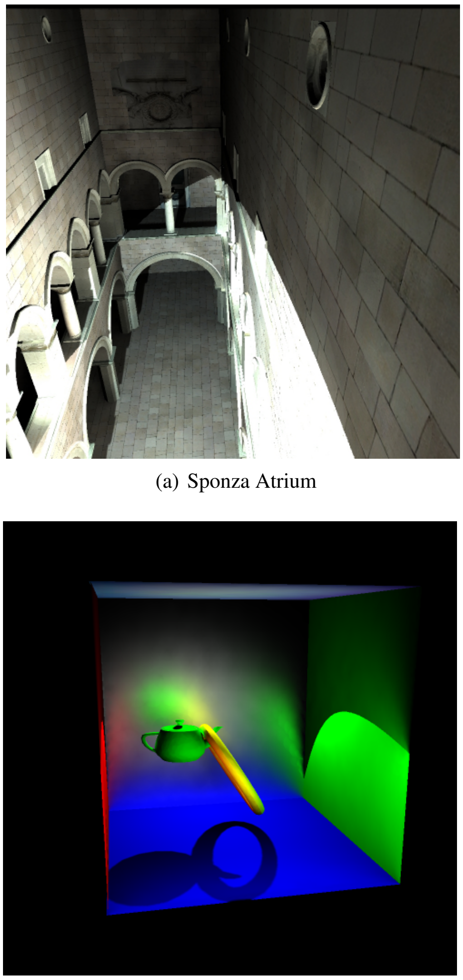
Figure 8: *The Sibenik Cathedral scene (80K triangles). The images show first bounce global illumination computed with Radiance (a) and our renderer (b)*

	Sibenik	Sponza
Triangles	80K	66K
Accuracy	0.15	0.1
Radiance time (s)	425	645
Our renderer time (s)	14.3	13.7
Speedup	29.7	47.1

Table 2: *Rendering times obtained using Radiance and our renderer for high quality rendering in diffuse environments. Each image is rendered at resolution* 1000 × 1000*.*

Our renderer also contains an implementation of radiance caching, hence supporting the computation of global illumination in glossy environments. If the environment also contains diffuse surfaces, our renderer uses either radiance or irradiance caching depending on the considered surface.

The *Cornell Box* scene presented in Figure 9(b) contains a glossy back wall (Phong BRDF, exponent 20), while the other objects are diffuse. The glossy BRDF and incoming radiance function are projected into the hemispherical basis using order 10 representation. The accuracy parameters are 0.25 for both radiance and irradiance caching. Figure 9(b) was rendered in 12.18 s using 3023 irradiance records and 869 radiance records. The GPU-based splatting for the irradiance cache is performed in 65.91 ms. The radiance cache is splatted in 935.5 ms.



(a) Sponza Atrium

(b) Cornell Box

Figure 9: *Images obtained with our renderer. The Sponza Atrium (66K triangles) contain only diffuse surfaces. The Cornell Box (1K triangles) contains a glossy back wall.*

Figure 1 shows an example of radiance cache splatting in a more complex glossy environment: the *Castle* scene contains about 57K triangles. In this scene, the glossiness of the roofs is obtained using a Phong BRDF with exponent 15. The BRDF and incoming radiance functions are represented by order 5 projection on the hemispherical harmonics basis. The accuracy parameter we used is 0.25 for irradiance caching and 0.2 for radiance caching. At the end of the rendering process, the irradiance and radiance cache respectively contain 3204 and 1233 records, computed in 10.1 s. The splatting on the GPU is performed in 58.6 ms for the irradiance cache. The radiance cache records are splatted in 493.7 ms.

The results presented above show that our algorithm is able to render fast high-quality glossy global illumination. The simplicity of our algorithm also allows its use for progressive rendering in interactive applications.

6.2. Interactive Visualization of Global Illumination

An important aspect of (ir)radiance caching is that the values of the records do not depend on the viewpoint. Therefore, records computed for a given frame can be reused in subsequent frames. Hence the radiance cache splatting approach can also be used in the context of interactive visualization of global illumination, and progressive rendering. Since the direct lighting is computed independently, the user can walk through the environment while the irradiance and radiance caches are filled on the fly. Figure 10 shows sequential images of *Sam* scene (63K triangles) obtained during an interactive session with an accuracy parameter of 0.5 and resolution 512 × 512. The global illumination is computed progressively, by adding at most 100 new records per frame. Our renderer provides an interactive frame rate (between 5 and 32 fps) during this session, allowing the user to move even if the global illumination computation is not completed. The accompanying videos presents interactive walkthroughs in diffuse and glossy environments: *Sam*, the *Sibenik Cathedral* and the *Castle*.

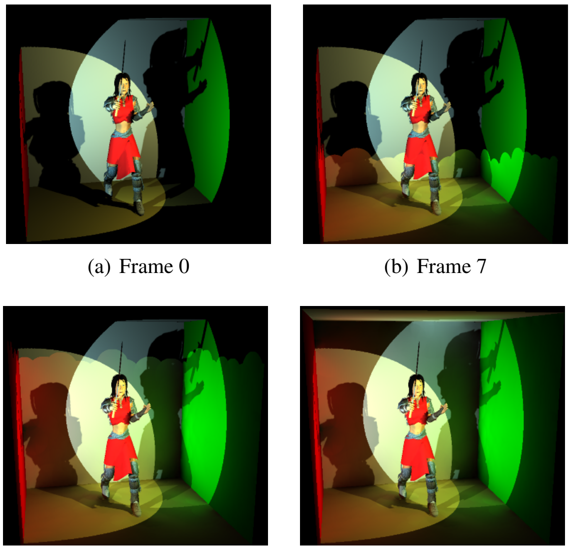
6.3. Discussion: Speedup Analysis

Previous subsections show that our method achieves significant speedup compared to the Radiance software by using the GPU for both record computation and final rendering.

In the (ir)radiance caching algorithm, the first and most expensive rendering step consists in computing the value of the cache records using hemisphere sampling. In this paper we propose to reduce this cost by using a sampling plane along with an accurate, novel compensation method. Therefore, the incoming radiance values are computed accurately using fast GPU rasterization and shadow maps.

Once all the needed (ir)radiance records are computed, the final rendering in Radiance relies on ray tracing and nearest-neighbors queries to calculate the outgoing radiance for each pixel. Although the cost involved is not dominant compared to the records computation, the speedup due to the simplicity of radiance cache splatting allows to display a globally illuminated scene at interactive rates.

Therefore, our global illumination method leverages graphics hardware using both novel and previous approaches, yielding a significant overall speedup while reducing the CPU workload.



(a) Frame 0

(b) Frame 7

(c) Frame 11

(d) Frame 14

Figure 10: *A progressive rendering session for interactive visualization of the* Sam *scene (63K triangles). Our renderer computes at most 100 new records per frame, hence maintaining an interactive frame rate (5 fps) during the global illumination computation. When the (ir)radiance cache is full, the global illumination solution is displayed at 32 fps.*

7. Conclusion and Future Work

In this paper, we proposed a reformulation of the (ir)radiance caching algorithm, by defining the radiance cache splatting. This method takes advantage of the latest graphics hardware to perform both the computation of irradiance and radiance records and the final rendering of global illumination. Our renderer shows a speedup of more than 29× compared to the Radiance software for high quality rendering. Moreover, we show interactive performance for global illumination visualization in moderately complex scenes. To our knowledge, the radiance cache splatting is the first implementation of irradiance and radiance caching using programmable graphics hardware. We believe that our method could be integrated into film production renderers for fast and accurate computation of indirect illumination.

In the future, we plan to extend this method in several directions. Among them, some challenging improvements are the handling of multiple light bounces and higher frequency BRDFs. Moreover, future work will consider the extension of our algorithm to highly complex models using GPU-based optimizations such as in [BWPP04].

Even though our method shows significantly faster results than previous approaches, high quality global illumination is still not computed interactively. Therefore, we would also like to speed up the rendering process to reach real-time performance.

References

[BP04] Bunnell M., Pellacini F.: *GPU Gems: Shadow map antialiasing*, 1 ed. Addison Wesley, 2004, pp. 185–192.

[BWPP04] Bittner J., Wimmer M., Piringer H., Purgathofer W.: Coherent hierarchical culling: Hardware occlusion queries made useful. In *Proceedings of Eurographics* (2004), pp. 615–624.

[CG85] Cohen M., Greenberg D. P.: The hemi-cube: A radiosity solution for complex environments. In *Proceedings of SIGGRAPH* (1985), vol. 19, pp. 31–40.

[CHH02] Carr N. A., Hall J. D., Hart J. C.: The ray engine. In *Proceedings of SIGGRAPH/Eurographics Workshop on Graphics Hardware* (2002), pp. 37–46.

[CHH03] Carr N. A., Hall J. D., Hart J. C.: GPU algorithms for radiosity and subsurface scattering. In *Proceedings of SIGGRAPH/Eurographics Workshop on Graphics hardware* (2003), pp. 51–59.

[CHL04] Coombe G., Harris M. J., Lastra A.: Radiosity on graphics hardware. In *Proceedings of Graphics Interface* (2004), pp. 161–168.

[DS05] Dachsbacher C., Stamminger M.: Reflective shadow maps. In *Proceedings of the Symposium on Interactive 3D graphics and games* (2005), pp. 203–231.

[GKPB04] Gautron P., Křivánek J., Pattanaik S., Bouatouch K.: A novel hemispherical basis for accurate and efficient rendering. In *Proceedings of Eurographics Symposium on Rendering* (2004), pp. 321–330.

[GTGB84] Goral C. M., Torrance K. E., Greenberg D. P., Battaile B.: Modelling the interaction of light between diffuse surfaces. In *Proceedings of SIGGRAPH* (1984), vol. 18, pp. 212–222.

[GWS04] Günther J., Wald I., Slusallek P.: Realtime caustics using distributed photon mapping. In *Proceedings of Eurographics Symposium on Rendering* (2004), pp. 111–121.

[Jen01] Jensen H. W.: *Realistic Image Synthesis Using Photon Mapping*. AK Peters, July 2001.

[KGBP05] Křivánek J., Gautron P., Bouatouch K., Pattanaik S.: Improved radiance gradient computation. In *Proceedings of SCCG* (2005), pp. 149–153.

[KGPB05] Křivánek J., Gautron P., Pattanaik S., Bouatouch K.: Radiance caching for efficient global illumination computation. *To appear in IEEE Transactions on Visualization and Computer Graphics* (2005).

[KSS02] Kautz J., Sloan P.-P., Snyder J.: Fast, arbitrary brdf shading for low-frequency lighting using spherical harmonics. In *Proceedings of Eurographics workshop on Rendering* (2002), Eurographics Association, pp. 291–296.

[LC04] Larsen B. D., Christensen N.: Simulating photon mapping for real-time applications. In *Proceedings of Eurographics Symposium on Rendering* (2004), pp. 123–131.

[LP03] Lavignotte F., Paulin M.: Scalable photon splatting for global illumination. In *Proceedings of GRAPHITE* (2003), pp. 1–11.

[LSSS04] Liu X., Sloan P.-P., Shum H.-Y., Snyder J.: All-frequency precomputed radiance transfer for glossy objects. In *Proceedings of Eurographics Symposium on Rendering* (2004), pp. 337–344.

[MM02] Ma V. C. H., McCool M. D.: Low latency photon mapping using block hashing. In *Proceedings of SIGGRAPH/Eurographics Workshop on Graphics Hardware* (2002), pp. 89–98.

[NPG03] Nijasure M., Pattanaik S., Goel V.: Interactive global illumination in dynamic environments using commodity graphics hardware. In *Proceedings of Pacific Graphics* (2003), pp. 450–454.

[NPG04] Nijasure M., Pattanaik S., Goel V.: Real-time global illumination on the GPU. *To appear in Journal of Graphics Tools* (2004).

[PBMH02] Purcell T. J., Buck I., Mark W. R., Hanrahan P.: Ray tracing on programmable graphics hardware. In *Proceedings of SIGGRAPH* (2002), pp. 703–712.

[PDC*03] Purcell T. J., Donner C., Cammarano M., Jensen H. W., Hanrahan P.: Photon mapping on programmable graphics hardware. In *Proceedings of Graphics Hardware* (2003), pp. 41–50.

[SB97] Sturzlinger W., Bastos R.: Interactive rendering of globally illuminated glossy scenes. In *Proceedings of Eurographics Workshop on Rendering* (1997), pp. 93–102.

[SHHS03] Sloan P.-P., Hall J., Hart J., Snyder J.: Clustered principal components for precomputed radiance transfer. In *Proceedings of SIGGRAPH* (2003), pp. 382–391.

[SKS02] Sloan P.-P., Kautz J., Snyder J.: Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. *Proceedings of SIGGRAPH* (2002), 527–536.

[SP89] Sillion F., Puech C.: A general two-pass method integrating specular and diffuse reflection. In *Proceedings of SIGGRAPH* (1989), vol. 23, pp. 335–344.

[TL04] Tabellion E., Lamorlette A.: An approximate global illumination system for computer generated films. In *Proceedings of SIGGRAPH* (2004), pp. 469–476.

[TPWG02] Tole P., Pellacini F., Walter B., Greenberg D. P.: Interactive global illumination in dynamic scenes. In *Proceedings of SIGGRAPH* (2002), pp. 537–546.

[War94] Ward G. J.: The Radiance lighting simulation and rendering system. In *Proceedings of SIGGRAPH* (1994), pp. 459–472.

[War04] Ward G. J.: *Radiance Synthetic Imaging System*. http://radsite.lbl.gov/radiance, 2004.

[WBS03] Wald I., Benthin C., Slusallek P.: Interactive global illumination in complex and highly occluded environments. In *Proceedings of Eurographics Symposium on Rendering* (2003), pp. 74–81.

[WH92] Ward G. J., Heckbert P. S.: Irradiance gradients. In *Proceedings of Eurographics Workshop on Rendering* (1992), pp. 85–98.

[Wil78] Williams L.: Casting curved shadows on curved surfaces. In *Proceedings of SIGGRAPH* (1978), pp. 270–274.

[WRC88] Ward G. J., Rubinstein F. M., Clear R. D.: A ray tracing solution for diffuse interreflection. In *Proceedings of SIGGRAPH* (1988), pp. 85–92.

[WS99] Ward G., Simmons M.: The holodeck ray cache: an interactive rendering system for global illumination in nondiffuse environments. *ACM Trans. Graph. 18*, 4 (1999), 361–368.

[WS03] Wand M., Strasser W.: Real-time caustics. In *Proceedings of Eurographics* (2003), pp. 611–620.

[WTL04] Wang R., Tran J., Luebke D.: All-frequency relighting of non-diffuse objects using separable BRDF approximation. In *Proceedings of Eurographics Symposium on Rendering* (2004), pp. 345–354.

Temporal Radiance Caching

Pascal Gautron, Kadi Bouatouch, *Member*, *IEEE*, and Sumanta Pattanaik, *Member*, *IEEE*

Abstract—We present a novel method for fast high-quality computation of glossy global illumination in animated environments. Building on the irradiance caching and radiance caching algorithms, our method leverages temporal coherence by sparse temporal sampling and interpolation of the indirect lighting. In our approach, part of the global illumination solution computed in previous frames is reused in the current frame. Our reusing scheme adapts to the change of incoming radiance by updating the indirect lighting only where there is a significant change. By reusing data in several frames, our method removes the flickering artifacts and yields a significant speedup compared to classical computation in which a new cache is computed for every frame. We also define temporal gradients for smooth temporal interpolation. A key aspect of our method is the absence of any additional complex data structure, making the implementation into any existing renderer based on irradiance and radiance caching straightforward. We describe the implementation of our method using graphics hardware for improved performance.

Index Terms—Global illumination, animation, temporal coherence, graphics processors.

1 Introduction

Efficient computation of global illumination in complex animated environments is one of the most important research challenges in computer graphics. Achievements in this field have many applications, such as visual effects for computer-assisted movies and video games. Many approaches for such computation have been proposed in the last 20 years. Most of these methods are based on radiosity, such as those in [1], [2], [3], and [4]. However, radiosity methods are prone to visual artifacts due to undersampling or high memory consumption and computational cost due to high-quality sampling. Therefore, other approaches, such as an extension of photon mapping [5] and irradiance caching [6], [7], have been proposed.

We propose a simple and accurate method based on temporal caching for efficient computation of global illumination effects in animated environments. Our method provides rapid generation of image sequences for environments in which viewer, objects, and light sources move.

Our approach focuses on a temporal optimization for lighting computation based on the irradiance caching [8] and radiance caching [9] techniques. These caching-based techniques use sparse sampling and interpolation in the spatial domain to reduce the computational cost of indirect illumination. Our extension introduces sparse sampling and interpolation in the temporal domain to reduce the computational cost in dynamic environments. We define a temporal weighting function and temporal radiance gradients for accurate temporal interpolation, which are based on an estimate of the change of incoming radiance in the course of time. This estimate requires a basic knowledge of the incoming radiance at future time steps. To this end, we propose an inexpensive graphics processing unit (GPU)-based method using reprojection to compute this estimate.

Unlike many previous approaches, our method does not introduce any new data structure and adds very little overhead to the existing memory requirements. Since the same record in the cache can be used in several frames, the computational cost related to the computation of the records is significantly reduced. Furthermore, the records used to render an animation segment can be kept within a small memory space. Once the records are computed in the scene, our GPU-based renderer can display the dynamic globally illuminated scene in real time.

This paper is organized as follows: Section 2 presents some significant previous works in the field of global illumination for animations in dynamic scenes. Section 3 describes the key aspects of the irradiance and radiance caching algorithms. In Section 4, we highlight the problems related to using those algorithms for animation rendering and present our main contributions: the temporal weighting function, the estimation of future incoming radiance by reprojection, and the temporal gradients (TGs). Section 5 contains implementation details for efficient computation using graphics hardware. Our results are presented in Section 6.

2 Previous Work

This section describes the most significant approaches to the computation of high-quality global illumination in dynamic scenes. We also describe several methods aiming at computing approximate solutions at interactive frame rates. A thorough description of many existing methods can be found in [4]. Most significant algorithms for global illumination computation are detailed in [10], [11].

- *P. Gautron is with France Telecom, R&D Division, TECH/IRIS Lab, 4, rue du Clos Courtel, 35512 Cesson-Sévigné, France. E-mail: pgautron@irisa.fr.*
- *K. Bouatouch is with IRISA, Campus de Beaulieu, 35042 Rennes Cedex, France. E-mail: kadi@irisa.fr.*
- *S. Pattanaik is with the University of Central Florida, School of Electrical Engineering and Computer Science, Computer Science Building, Room 251, Orlando, FL 32816-2362. E-mail: sumant@cs.ucf.edu.*

Manuscript received 1 June 2006; revised 10 Nov. 2006; accepted 24 Jan. 2007; published online 28 Feb. 2007.
Recommended for acceptance by P. Dutre.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org, and reference IEEECS Log Number TVCG-0078-0606.
Digital Object Identifier no. 10.1109/TVCG.2007.1039.

1077-2626/07/$25.00 © 2007 IEEE Published by the IEEE Computer Society

2.1 Interactive Methods

The methods described in this section focus on interactive approximate rendering in which the quality may be compromised to enhance interactivity. The Render Cache [12], [13] is based on the high coherence of the points visible between two subsequent frames: Most of the points visible in frame n are also visible in frame $n+1$. Therefore, the render cache stores the points visible in frame n and reprojects them onto frame $n+1$. As the visibility may change between two successive frames, some pixels may not have any corresponding visible point stored in the cache. In that case, those pixels are either filled by computing the actual corresponding hit point or by hole filling. However, the artifacts due to missing information make the Render Cache usable only in the context of fast approximate rendering. Note that we use a similar reprojection technique to estimate the temporal change of incoming lighting.

Another approach based on interactive ray tracing is Frameless Rendering [14]. This method is based on parallel asynchronous ray tracing. Each processor is assigned a fixed set of pixels to render continuously. The computation of each pixel updates the resulting image immediately. The computational power of several processors working in parallel allows interactive frame rates. This method has been enhanced by Dayal et al. [15] for using temporal coherence. As in our approach, the authors compute TGs. However, those gradients are computed in image space.

In the Shading Cache method [16], the irradiance at the vertices of a tesselated scene is adaptively computed and cached using a parallel computing cluster. The scene is rendered and displayed at interactive frame rates using hardware-based interpolation. The vertices within the region affected by object and camera movement are localized using an image-space sampling scheme and updated. The rendering shows ghosting artifacts in regions affected by moving objects; for example, color bleeding due to an object may remain behind even though the object has moved away. These artifacts eventually disappear if the viewer and objects remain static for several seconds.

A similar drawback is present in the work by Dmitriev et al. [5]. The authors present a density estimation technique using photon mapping [17] and quasi Monte Carlo. The photon map is updated when the user moves an object. For each frame, the algorithm detects which photon paths should be recomputed. The update consists in using quasi Monte Carlo to trace a group of photons following a similar path. This method is well suited for interactive manipulation of objects but suffers from delays between the object motion and the update of the photon map.

2.2 Irradiance Caching and Final Gathering

Several approaches have been proposed to accelerate the final gathering process used to render photon maps in dynamic scenes. The methods described in this section exploit temporal coherence in the context of final gathering using irradiance caching [8].

Tawara et al. [18] propose a two-step method for computing indirect lighting in dynamic scenes. In the first step, they compute an irradiance cache using only the static objects of the scene. Then, for each frame, they update the irradiance cache by introducing the dynamic objects at their frame-specific positions. This method shows significant speedup but is restricted to animations during which "lighting conditions do not change significantly" [18]. Otherwise, the static irradiance cache has to be recomputed from scratch, leading to the temporal artifacts inherent in this method.

Another approach by Tawara et al. [6] is based on the detection of the incoming radiance changes related to the displacement of objects. The rays traced for computing irradiance records are stored and updated using either a user-defined update rate or an adaptive rate based on the number of rays hitting a dynamic object. This method requires a large amount of memory to store the rays, making it difficult to use in complex scenes where many records have to be computed.

The method described in [7] is based on a data structure called *anchor*. Using this structure, the algorithm detects changes in visibility and incoming radiance for each irradiance record during an animation sequence. Therefore, the irradiance values stored in the cached records can be efficiently updated. However, even though this method provides high-quality results, it requires the explicit storage of the rays used to compute each irradiance record and, hence, is highly memory intensive.

In this paper, we present a method exploiting temporal coherence for using irradiance and radiance caching [8], [9] in dynamic scenes. Section 3 presents a brief overview of irradiance and radiance caching.

3 Irradiance and Radiance Caching: An Overview

The irradiance and radiance caching algorithms are based on the following observation: "The indirect illuminance tends to change slowly over a surface" [8]. These methods take advantage of spatial coherence by sparsely sampling and interpolating indirect incoming radiance. Irradiance caching [8] is designed for the computation of indirect diffuse lighting. Radiance caching [9] extends the approach proposed by Ward et al. [8] to glossy interreflections.

The irradiance and radiance caching algorithms are very similar. In irradiance caching, a record stores the irradiance at a given point on a surface. In radiance caching, a record stores the projection coefficients of the incoming radiance for the hemispherical harmonics basis [19]. The interpolation schemes are also similar: The irradiance stored in the irradiance cache and the coefficients stored in the radiance cache undergo weighted interpolation. The method described in this paper is similarly applicable to both caching techniques. For the simplicity of explanation, we only consider irradiance caching in most of the following discussions. Specific details about radiance caching are given in Section 4.6.

Irradiance Interpolation. Let us consider a point $\mathbf{p}$ in a scene. In [8], an estimate of the irradiance $E(\mathbf{p})$ at point $\mathbf{p}$ is given by

$$E(\mathbf{p}) = \frac{\sum_{K \in S_r} w_K(\mathbf{p}) E_K(\mathbf{p})}{\sum_{K \in S_r} w_K(\mathbf{p})}, \qquad (1)$$

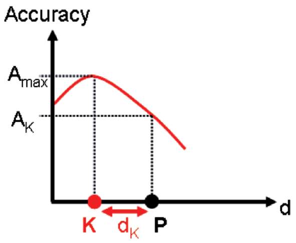


Fig. 1. The *accuracy* of the reconstructed lighting is maximum at the location of actual computation of record $\mathbf{K}$. The accuracy decreases when the lighting is extrapolated at a point $\mathbf{p}$ in the neighborhood of $\mathbf{K}$.

where S_r represents the set of irradiance records surrounding $\mathbf{p}$. For each record $K \in S_r$, $w_K(\mathbf{p})$ is the weighting function of record K evaluated at point $\mathbf{p}$. $E_K(\mathbf{p})$ is the contribution of record K to the computed incoming lighting at point $\mathbf{p}$. In the next two paragraphs, we describe the irradiance weighting function and the computation of the contribution of the records using irradiance gradients.

Weighting Function. In the irradiance caching algorithm, a given record contributes to the indirect lighting of points with similar lighting. If the incoming lighting at the record location changes rapidly in the neighborhood of the record, its weight should be small in the surrounding area. Conversely, if the lighting changes slowly, the weight of the record should be high. Therefore, the weighting function is based on an approximation of the potential change in incoming radiance with respect to displacement and rotation.

The weighting function is chosen as the inverse of this estimated change. At a point $\mathbf{p}$ with normal $\mathbf{n}$, the weighting function of record K is defined as

$$w_K(\mathbf{p}) = \frac{1}{\frac{\|\mathbf{p}-\mathbf{p_K}\|}{R_K} + \sqrt{1 - \mathbf{n} \cdot \mathbf{n_K}}}, \qquad (2)$$

where $\mathbf{p_K}$, $\mathbf{n_K}$, and R_K are, respectively, the location of record K, its normal, and the harmonic mean distance to the objects visible from $\mathbf{p_K}$.

This weighting function is not only used for interpolation but also to determine the set of records $S_r(\mathbf{p})$ contributing to the incoming radiance at point $\mathbf{p}$:

$$S_r(\mathbf{p}) = \{K | w_K(\mathbf{p}) \geq a\}, \qquad (3)$$

where a is a user-defined accuracy value.

Irradiance Gradients for Accurate Extrapolation. The irradiance contribution of record K to the irradiance at point $\mathbf{p}$ depends on the translation and rotation gradients of the irradiance. The gradient computations proposed in [20], [9], and [21] present several methods to estimate the change in incoming radiance with respect to translation and rotation. Figs. 1a and 1b in [20] illustrate the quality improvement obtained using gradients. Ward and Heckbert [20] define the contribution of record K to point $\mathbf{p}$ with normal $\mathbf{n}$ as

$$E_K(\mathbf{p}) = E_K + (\mathbf{n_K} \times \mathbf{n}) \cdot \nabla_\mathbf{r} + (\mathbf{p} - \mathbf{p_K}) \cdot \nabla_\mathbf{p}, \qquad (4)$$

where E_K, $\nabla_\mathbf{r}$, and $\nabla_\mathbf{p}$ are, respectively, the irradiance, the rotation gradient, and the translation gradient stored in record K.

The irradiance and radiance caching algorithms provide an accurate and efficient way of computing global illumination in static scenes. Furthermore, several methods have been proposed to enhance the quality and efficiency of these algorithms [22], [23]. However, problems arise when using this approach for global illumination computation in dynamic environments. Section 4 presents the typical problems encountered when using the irradiance caching algorithm in dynamic scenes and describes our temporal optimization method.

4 Temporal Irradiance Caching

4.1 Quality Measurement

Both our method and the (ir)radiance caching algorithms rely on sparse sampling and approximations (interpolations and extrapolations) for fast global illumination computation. Intuitively, the quality of the reconstructed lighting is high at the location and time of actual computation and decreases as the extrapolation point and time get away from the record. In this paper, the quality of the reconstructed lighting is named *accuracy*. The maximum (100 percent) accuracy is obtained when the lighting is explicitly computed (that is, at the location and time of record creation) and decreases as the point of interest gets away from the record (Fig. 1).

4.2 Irradiance Caching in Dynamic Scenes

As described in the previous section, irradiance caching leverages spatial coherency of indirect lighting and, thus, reduces the computation time of global illumination. In dynamic scenes, a simple and commonly used approach is to discard all the cached records and start with an empty cache at the beginning of each frame. This indiscriminate discard of records amounts to significant waste of computational effort. Additionally, the resulting animation video quality may be poor. The reason for this is that the distributions of record location in frames n and $n+1$ are likely to be different. Fig. 2 illustrates the consequence of the change of record distribution: Since the gradients extrapolate the change of incoming radiance, they are not completely accurate compared to the ground truth. Therefore, the accuracy of the lighting reconstructed by irradiance caching is not constant over a surface: The accuracy is maximum at the record location and decreases as the extrapolation point gets away from the record. Therefore, changing the distribution of records also changes the distribution of the accuracy, yielding high-frequency flickering artifacts. Thus, the simple use of irradiance caching in dynamic scenes gives rise to poor animation quality.

In this paper, we propose a simple and unified framework for view-dependent global illumination in dynamic environments with predefined animation in which objects, light sources, and cameras can move.

4.3 Overview of Temporal Radiance Caching

In [8], [20], the interpolation scheme of Ward et al. converts the high-frequency noise of Monte Carlo path tracing into low frequency error. As shown above, this result is achieved by sparse sampling, extrapolation, and interpolation in space. In this paper, our aim is to convert the high-

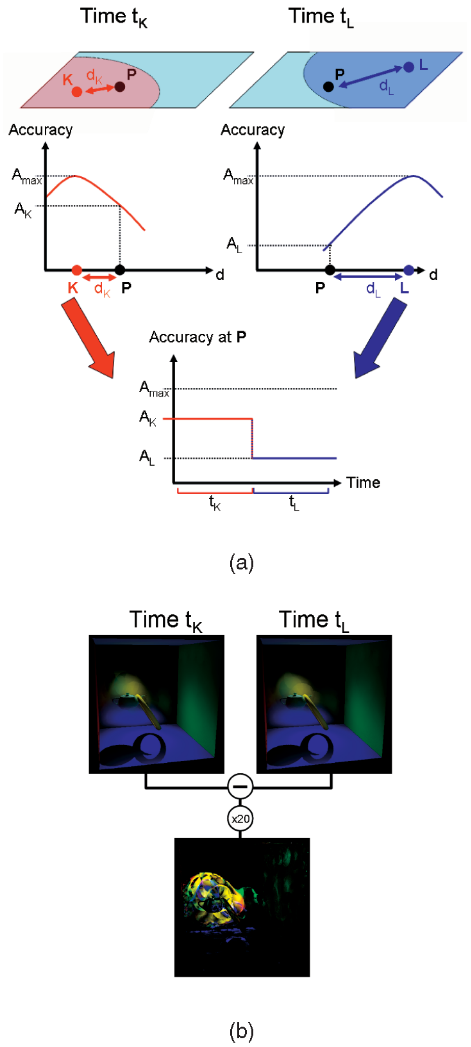


Fig. 2. (a) Changing the location of the records between successive frames can significantly modify the accuracy of the lighting at a point $\mathbf{p}$. (b) Therefore, the incoming radiance at this point can change noticeably between two frames, yielding flickering artifacts (b). Note that the maximum accuracy A_{max} is obtained at the point and time of actual computation of the incoming radiance.

frequency temporal noise (that is, the flickering artifacts) into low-frequency temporal errors by sparse sampling, extrapolation, and interpolation in the temporal domain. More precisely, we amortize the cost of record computation over several frames by performing a sparse temporal sampling and temporal interpolation of the irradiance (Algorithm 1). When a record K is created at frame n, the future incoming lighting is estimated. This estimate is first used to compute the ratio between the current and future lightings. In the spirit of [8], we define our temporal weighting function w_K^t as the inverse of the temporal change. Hence, the number of frames in which K can contribute is inversely proportional to the future change of incoming radiance. Since the actual computation of the future incoming lighting may be expensive, we use a simple reprojection technique for estimating the future lighting using the data sampled at the current frame. As the irradiance at a point is extrapolated using the irradiance and gradients of neighboring records, we propose TGs for smooth temporal interpolation and extrapolation of the irradiance.

```
Algorithm 1 Temporal Radiance Caching
  for all frames $n$ do
    for all existing records $K$ do
      if $w_K^t(n)$ is greater than a threshold then
        Use $K$ in frame $n$
      end if
    end for
    for all points $\mathbf{p}$ where a new record is needed do
      Sample the hemisphere above $\mathbf{p}$
      Estimate the future incoming lighting (Section 4.7)
      Generate $w_K^t$ (Section 4.4)
      Compute the TGs (Section 4.5)
      Store the record in the cache
    end for
end for
```

4.4 Temporal Weighting Function

The temporal weighting function expresses the confidence on a given record as a function of time. Using a derivation similar to that in [8], we define the temporal change ϵ^t of incoming radiance between time t and t_0 as

$$\epsilon^t = \frac{\partial E}{\partial t}(t_0)(t - t_0). \tag{5}$$

This derivative $\frac{\partial E}{\partial t}(t_0)$ can be approximated using estimates of incoming radiance at two successive times t_0 and t_1, denoted E_0 and E_1:

$$\begin{aligned} \frac{\partial E}{\partial t}(t_0) &\approx \frac{E_1 - E_0}{t_1 - t_0} && (6)\\ &= \frac{\tau E_0 - E_0}{t_1 - t_0} \qquad \text{where } \tau = E_1/E_0 && (7)\\ &= E_0 \frac{\tau - 1}{t_1 - t_0}. && (8) \end{aligned}$$

In our method, the time range of the animation is discretized into integer frame indices. Therefore, we always choose $t_1 - t_0 = 1$, that is, E_1 and E_0 represent the estimated irradiance at two successive frames.

As in [8], we define the temporal weighting function as the inverse of the change, excluding the term E_0:

$$w_K^t(t) = \frac{1}{(\tau - 1)(t - t_0)}, \tag{9}$$

where $\tau = E_1/E_0$ is the *temporal irradiance ratio*.

This function can be evaluated and tested against a user-defined accuracy value a^t. A record K created at t_0 is allowed to contribute to the image at t if

$$w_K^t(t) \geq 1/a^t. \tag{10}$$

The temporal weighting function is used to adjust the time segment during which a record is considered valid. Since a given record can be reused in several frames, the computational cost can be significantly reduced.

However, (7) shows that if the environment remains static starting from frame t_0, we obtain $\tau = 1$. Therefore, (10) shows that w_K^t is infinite for any frame and, hence, record K is allowed to contribute at any time $t > t_0$. However, since part of the environment is dynamic, the inaccuracy becomes

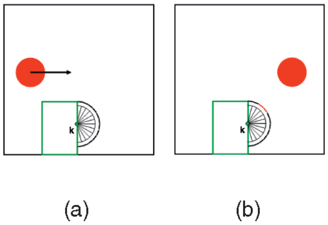


Fig. 3. When record K is created at time t_K, the surrounding environment is static ($\tau_K = 1$). However, the red sphere is visible from the K n frames later. The user-defined value $\delta_{t_{max}}$ prevents the record from contributing if $n > \delta_{t_{max}}$, then reducing the risk of using obsolete records. (a) Time t_K. (b) Time $t_K + n$.

significant when $t - t_0$ gets high (see Fig. 3). This is a limitation of our technique for estimating the temporal change of incoming radiance, which determines the lifespan of a record by considering only the change between E_t and E_{t+1}. Therefore, we introduce a user-defined value $\delta_{t_{max}}$ limiting the length of the validity time segment associated with each record. If (11) does not hold, we decide that the record cannot be reasonably used:

$$t - t_0 < \delta_{t_{max}}. \tag{11}$$

This reduces the risk of using obsolete records, which allows the user to control the artifacts due to residual global illumination effects also known as "ghosts," which commonly appear in interactive methods. However, as a and a^t, this value must be user-defined by trial and error to obtain the best results. If $\delta_{t_{max}}$ is too low, many records may be recomputed unnecessarily. Setting $\delta_{t_{max}} = 1$ implies the recomputation of each record for each frame. In this case, the resulting performance would be similar to the classical per-frame computation. Nevertheless, this would completely avoid the ghosting artifacts, since the indirect lighting is continually computed from scratch. If $\delta_{t_{max}}$ is set too high, the same records might be reused in too many frames. Hence, artifacts due to the residual global illumination effects are likely to appear in the vicinity of the moving objects, degrading the quality of the rendered frame. Consequently, such a high value significantly reduces the rendering time at the detriment of the temporal accuracy.

However, abrupt discarding of a record reintroduces the flickering problem described in Section 4.2. As proposed in [18], we avoid this problem by keeping track of the location of the records over time. Let us consider a record K located at point $\mathbf{p_K}$. If K was allowed to contribute to the previous frame and cannot be reused in current frame, a new record l is created at the same location, that is, $\mathbf{p_l} = \mathbf{p_K}$ (Fig. 4b). Since the location of visible records remains constant in space, the accuracy at a given point tends to be constant over time, hence reducing the flickering artifacts. Note that the locations of records remain constant even though they lie on dynamic objects (Fig. 5).

The temporal weighting function provides a simple and adaptive way of leveraging temporal coherence by introducing an aging method based on the change of incoming radiance. Nevertheless, Fig. 4c shows that the accuracy of the computation is still not continuous in the course of time. Replacing obsolete records with new ones creates a discontinuity of accuracy, which causes the visible flickering artifacts. Therefore, we propose TGs to generate a smooth and less noticeable transition between successive records.

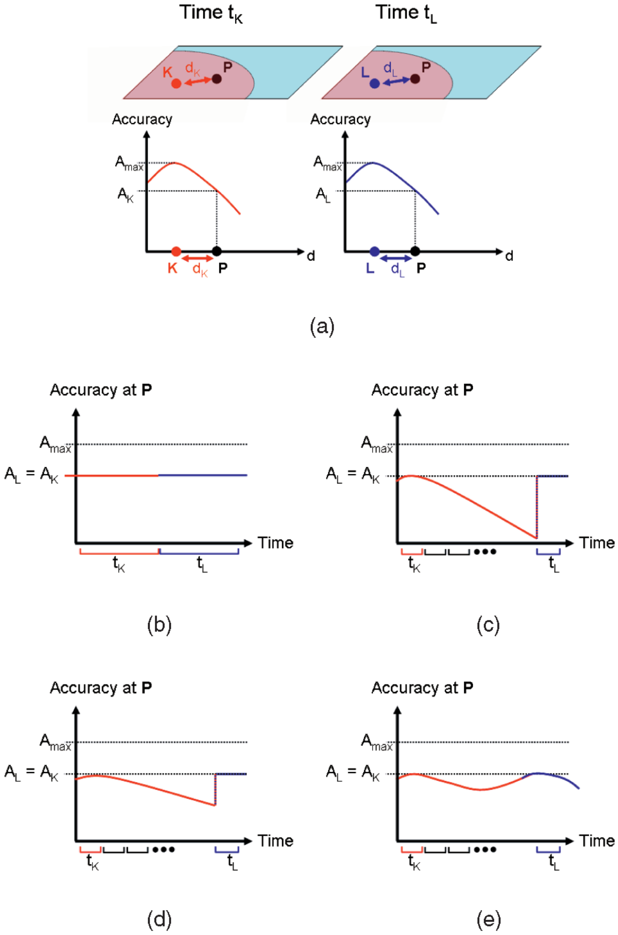


Fig. 4. Empirical shape of the accuracy (a) at a fixed time with respect to space and (b), (c), (d), (e) at a fixed location $\mathbf{p}$ with respect to time. (b) New record after $\delta = 1$. (c) New record after $\delta = n$ frames with no temporal gradient. (d) New record after $\delta = n$ frames with extrapolated temporal gradient. (e) New record after $\delta = n$ frames with an interpolated temporal gradient. If the records are located at the same point between successive frames as in (a), the temporal accuracy is improved as shown in (b). However, as shown in (c), flickering artifacts due to the temporal discontinuities of accuracy may appear when a record is reused in several frames, then recomputed. (d) Extrapolated TGs decrease the amplitude of the discontinuity in one pass, reducing flickering. (e) Interpolated TGs use two passes to eliminate the discontinuity by smoothing out the temporal changes.

4.5 Temporal Gradients

TGs are conceptually similar to classical irradiance gradients. Instead of representing the incoming radiance change with respect to translation and rotation, those gradients represent how the incoming radiance gets altered over time.

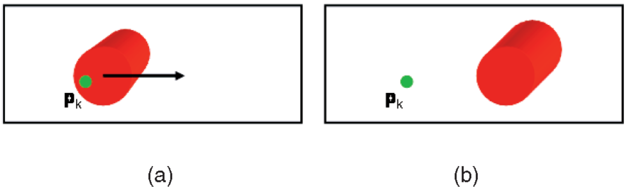


Fig. 5. Record K created at time t_K remains at point $\mathbf{p_K}$ even though it lays on a dynamic object. (a) Time t_K. (b) Time $t_K + n$.

In the context of irradiance caching, (4) shows that the irradiance at point $\mathbf{p}$ is estimated using rotation and translation gradients. The temporal irradiance gradient of record K at a given point $\mathbf{p}$ with normal $\mathbf{n}$ is derived from (4) as

$$\boldsymbol{\nabla}^{\mathbf{t}}(\mathbf{p}) = \frac{\partial}{\partial t}(E_K + (\mathbf{n_K} \times \mathbf{n}) \cdot \boldsymbol{\nabla}_{\mathbf{r}} + (\mathbf{p} - \mathbf{p_K}) \cdot \boldsymbol{\nabla}_{\mathbf{p}}), \tag{12}$$

where

- $\boldsymbol{\nabla}_{\mathbf{r}}$ and $\boldsymbol{\nabla}_{\mathbf{p}}$ are the rotation and translation gradients.
- $\mathbf{p_K}$ and $\mathbf{n_K}$ are the location and normal of record K.

As described in Section 4.4, we keep the location of all records constant over time. We choose any point of interest $\mathbf{p}$ with normal $\mathbf{n}$, which is also constant over time. Therefore, $\mathbf{n_K} \times \mathbf{n}$ and $\mathbf{p} - \mathbf{p_K}$ are constant with respect to time. The equation for TGs becomes

$$\boldsymbol{\nabla}^{\mathbf{t}}(\mathbf{p}) = \boldsymbol{\nabla}^{\mathbf{t}}_{\mathbf{E_K}} + (\mathbf{n_K} \times \mathbf{n}) \cdot \boldsymbol{\nabla}^{\mathbf{t}}_{\nabla_{\mathbf{r}}} + (\mathbf{p} - \mathbf{p_K}) \cdot \boldsymbol{\nabla}^{\mathbf{t}}_{\nabla_{\mathbf{p}}}, \tag{13}$$

where

- $\boldsymbol{\nabla}^{\mathbf{t}}_{\mathbf{E_K}} = \frac{\partial E_K}{\partial t}$ is the *TG of irradiance*.
- $\boldsymbol{\nabla}^{\mathbf{t}}_{\nabla_{\mathbf{r}}} = \frac{\partial \nabla_{\mathbf{r}}}{\partial t}$ is the *TG of rotation gradient*.
- $\boldsymbol{\nabla}^{\mathbf{t}}_{\nabla_{p}} = \frac{\partial \nabla_{\mathbf{p}}}{\partial t}$ is the *TG of translation gradient*.

Using (13), the contribution of record K created at time t_K to the incoming radiance at point $\mathbf{p}$ at time t is estimated by

$$\begin{aligned} E_K(\mathbf{p}, t) = {} & E_K + \boldsymbol{\nabla}^{\mathbf{t}}_{\mathbf{E_K}}(t - t_K) \\ & + (\mathbf{n}_{\mathrm{K}} \times \mathbf{n}) \cdot (\boldsymbol{\nabla}_{\mathbf{r}} + \boldsymbol{\nabla}^{\mathbf{t}}_{\nabla_{\mathbf{r}}}(t - t_K)) \\ & + (\mathbf{p}_{\mathrm{K}} - \mathbf{p}) \cdot (\boldsymbol{\nabla}_{p} + \boldsymbol{\nabla}^{\mathbf{t}}_{\nabla_{\mathbf{p}}}(t - t_K)). \end{aligned} \tag{14}$$

This formulation represents the temporal change of the incoming radiance around p_K as three vectors. These vectors represent the change of the incoming radiance at point $\mathbf{p_K}$ and the change of the translation and rotation gradients over time. The values of these vectors can be easily computed using the information generated in Section 4.7. Since our method estimates the incoming radiance at time $t+1$ using the information available at time t, we define *extrapolated* TGs as

$$\boldsymbol{\nabla}^{\mathbf{t}}_{\mathbf{E_K}} \approx E(t_K + 1) - E(t_K) \tag{15}$$

$$\boldsymbol{\nabla}^{\mathbf{t}}_{\nabla_{\mathbf{r}}} \approx \boldsymbol{\nabla}_{\mathbf{r}}(t_K + 1) - \boldsymbol{\nabla}_{\mathbf{r}}(t_K) \tag{16}$$

$$\boldsymbol{\nabla}^{\mathbf{t}}_{\nabla_{\mathbf{p}}} \approx \boldsymbol{\nabla}_{\mathbf{p}}(t_K + 1) - \boldsymbol{\nabla}_{\mathbf{p}}(t_K). \tag{17}$$

However, as illustrated in Fig. 4d, these TGs do not remove all the discontinuities in the animation. When a record K is replaced by record l, the accuracy of the result exhibits a possible discontinuity, yielding some flickering artifacts. As explained in Section 4.4, this problem can be avoided by keeping track of the history of the records: When record K becomes obsolete, a new record l is created at the same location. Since $t_l > t_K$, we can use the value of incoming radiance stored in l to compute *interpolated* TG for record K:

$$\boldsymbol{\nabla}^{\mathbf{t}}_{\mathbf{E_K}} \approx (E_l - E_K)/(t_l - t_K) \tag{18}$$

$$\boldsymbol{\nabla}^{\mathbf{t}}_{\nabla_{\mathbf{r}}} \approx (\boldsymbol{\nabla}_{\mathbf{r}}(t_l) - \boldsymbol{\nabla}_{\mathbf{r}}(t_K))/(t_l - t_K) \tag{19}$$

$$\boldsymbol{\nabla}^{\mathbf{t}}_{\nabla_{\mathbf{p}}} \approx (\boldsymbol{\nabla}_{\mathbf{p}}(t_l) - \boldsymbol{\nabla}_{\mathbf{p}}(t_K))/(t_l - t_K). \tag{20}$$

As illustrated in Fig. 4e, the TGs enhance the continuity of the accuracy, hence removing the flickering artifacts. However, the gradients only account for the first derivative of the change of incoming lighting, which temporally smoothes the changes. Although this method proves accurate in scenes with smooth changes, it should be noted that the gradient-based temporal interpolation may introduce ghosting artifacts when used in scenes with very sharp changes of illumination. In this case, a^t and $\delta_{t_{max}}$ must be reduced to obtain a sufficient update frequency.

4.6 Extension to Radiance Caching

4.6.1 Temporal Weighting Function

Our temporal weighting function is based on an estimate of the ratio between the current and future incoming radiances. In the context of radiance caching, the directional information of the incoming radiance must be very accurate. Therefore, we define the temporal radiance ratio as

$$\tau_{radiance} = max\{\lambda_1^i / \lambda_0^i, 0 \geq i < n\}, \tag{21}$$

where λ_0^i and λ_1^i, are respectively the ith projection coefficient of the current and future incoming lighting. By using the maximum ratio, our method can account for directional change of incoming radiance without loss of accuracy.

4.6.2 Temporal Gradients

As described in [9], the rotational gradient is not necessary in the radiance caching algorithm. The incoming radiance function and the translation gradient being represented using projection coefficients, we compute the TG of each coefficient independently. Thus, the problem reduces to the computation of the above equations for each coefficient.

However, as shown in (7), the determination of our temporal weighting function and extrapolated gradients relies on the knowledge of the incoming radiance at the next time step, E_{t_0+1}. Since the explicit computation of E_{t_0+1} would introduce a significant computational overhead, we propose a simple and accurate estimation method based on reprojection.

4.7 Estimating E_{t_0+1}

We use the reprojection and hole-filling approach proposed by Walter et al. [12]. However, it must be noted that, while Walter et al. use reprojection for interactive visualization using ray tracing, our aim is to provide a simple and reliable estimate of the incident radiance at a given point at time $t_0 + 1$ by using the data acquired at time t_0 only. This estimate will be used to determine the lifespan of the records by evaluating our temporal weighting function.

In the context of predefined animation, the changes in the scene are known and accessible at any time. When a record K is created at time t_0, the hemisphere above $\mathbf{p_K}$ is sampled (Fig. 6a) to compute the incoming radiance and gradients at this point. Since the changes between times t_0 and $t_0 + 1$ are known, it is possible to reproject the points visible at time t_0 to obtain an estimate of the visible points at time $t_0 + 1$ (Fig. 6b). The outgoing radiance of the reprojected visible points can be estimated by accounting for the rotation and displacement of both objects and light

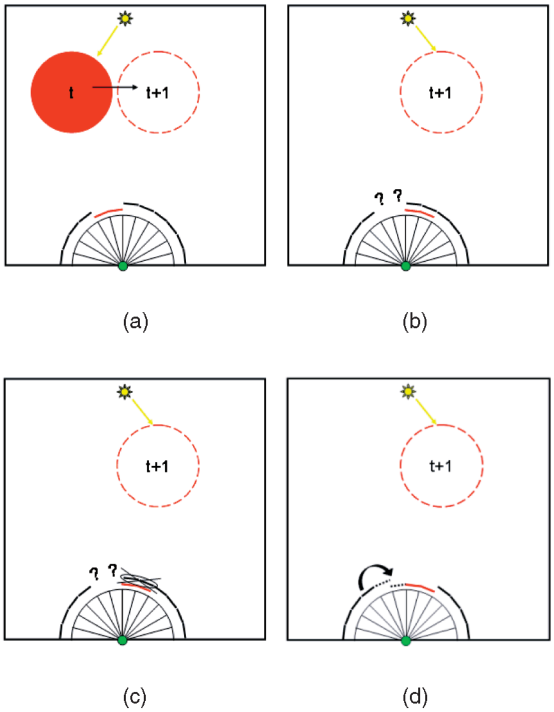


Fig. 6. (a) The hemisphere is sampled at time t as in the classical irradiance caching process. (b) For each ray, our method determines where each visible point will be located at time $t+1$ by reprojection. (c) Distant overlapping points are removed using a depth test, whereas (d) resulting holes are filled using the farthest neighboring values.

sources. In overlapping areas, a depth test accounts for the occlusion change (Fig. 6c).

However, some parts of the estimated incoming radiance may be unknown (holes) due to displacement and overlapping of visible objects (Fig. 6d). As proposed in [12], we use a simple hole-filling method: Each hole is filled using the background values, yielding a plausible estimate of the future indirect lighting.

As shown in Fig. 7 and Table 1, the reprojection reduces the error in the estimate of the future incoming lighting. Those errors were measured by comparing the irradiances at time $t+1$ with the irradiances estimated using records created at time t.

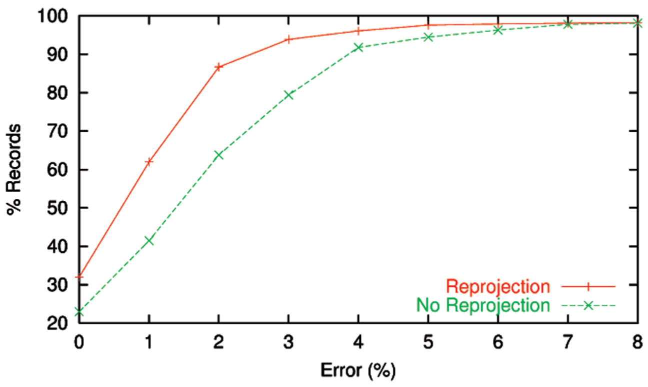


Fig. 7. Error between the actual lighting at $t+1$ and the lighting at $t+1$ estimated with and without reprojection. This plot represents the percentage of records for which the estimate of the future incoming lighting is below a given RMS error level. The reprojection reduces the overall error in the estimate compared to a method without reprojection (that is, where the lighting is considered temporally constant). Errors are computed using 4,523 records.

TABLE 1
RMS Error Between the Actual Lighting at $t+1$ and the Lighting at $t+1$ Estimated with and without Reprojection

Error	Reprojection	No Reprojection
Min	0%	0%
Max	30%	32%
Mean	2.9%	3.7%
Median	1.6%	2.4%

(Based on 4,523 values).

Our method provides a simple way of extending irradiance caching to dynamic objects and dynamic light sources by introducing a temporal weighting function and TGs. In Section 5, we discuss the implementation of our method on a GPU for increased performance.

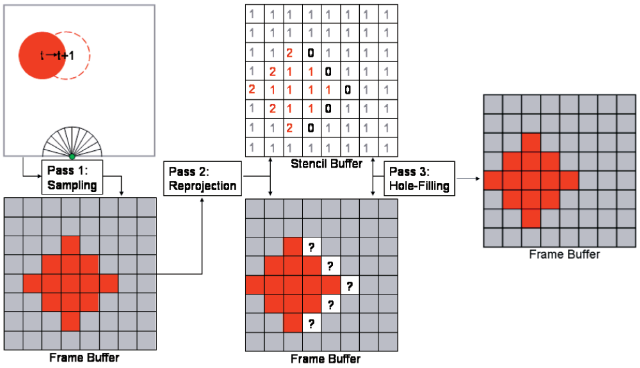
5 GPU Implementation

Our method has been implemented within a GPU-based renderer for irradiance and radiance caching. First, we detail the implementation of the incoming radiance estimate by reprojection (Section 4.7). Then, we describe how the GPU can be simply used in the context of radiance cache splatting to discard useless records and avoid their replacement.

Reprojection of incoming radiance. As shown in Section 4.4, the computation of the temporal weighting function and TGs for a given record K requires an estimate of the radiance reaching point $\mathbf{p_K}$ at the next time step. This estimate is obtained through reprojection (Section 4.7), provided that the position of the objects at the next time step is known. Therefore, for a given vertex v of the scene and a given time t, we assume that the transformation matrix corresponding to the position and normal of v at time $t+1$ is known. We assume that such matrices are available for light sources as well. Using the method described in [24], a record K can be generated by rasterizing the scene on a single plane above point $\mathbf{p_K}$. In a first pass, during the rasterization at time t, shaders can output an estimate of the position and incoming lighting at time $t+1$ of each point visible to $\mathbf{p_K}$ at time t. This output can be used to reconstruct an estimate of the incoming radiance function at time $t+1$.

This estimate is obtained in a second pass: Each projected visible point generated in the first pass is considered a vertex. Each of those vertices is sent to the graphics pipeline as a pixel-sized point. The result of the rasterization process is an estimate of the incoming radiance function at time $t+1$. Since the size of the sampling plane is usually small (typically, 64×64), this process is generally much faster than resampling the whole scene.

During the reprojection process, some fragments may overlap. Even though the occlusion can be simply solved by classical Z-Buffer, the resulting image may contain holes (Fig. 6d). These holes are created at the location of dynamic objects. Since the time shift between two successive frames is very small, the holes are also small. As described in Section 4.7, we use a third pass to fill the holes using the local background (that is, the neighboring value with the



1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	2	0	1	1	1	1
1	2	1	1	0	1	1	1
2	1	1	1	1	0	1	1
1	2	1	1	0	1	1	1
1	1	2	0	1	1	1	1
1	1	1	1	1	1	1	1

Fig. 8. Reprojection using the GPU. The first pass samples the scene to gather the required information. Then, the visible points are reprojected to their estimated position at next time step. During this pass, each rendered fragment increments the stencil buffer. Finally, the holes (that is, where the stencil value is 0) are filled using the deepest neighboring values.

highest depth). This computation can be performed efficiently on the GPU using the stencil buffer with an initial value of 0. During the reprojection process, each rasterized point increments the stencil buffer. Therefore, the hole-filling algorithm must be applied only on pixels where the stencil buffer is still 0. The final result of this algorithm is an estimate of the incoming radiance at time $t+1$, generated entirely on the GPU (Fig. 8). This estimate is used in the extrapolated TGs and the temporal weighting function. As shown in (10), this latter defines a maximum value of the lifespan of a given record and triggers its recomputation. However, this recomputation is not always necessary.

Radiance cache splatting. The radiance cache splatting method [24] is based on a simple observation of the spatial weighting function described in [8]: An (ir)radiance record K cannot contribute to the lighting of points located outside a sphere of influence centered at p_K. Therefore, the indirect lighting at visible points can be obtained by splatting the sphere corresponding to record K on the image plane. This splatting is performed on graphics hardware by rasterizing a quadrilateral tightly bounding the sphere. For each fragment within this quadrilateral, a fragment program evaluates the weighting function for record K and verifies whether record K is allowed to contribute to the indirect lighting of the visible point corresponding to this fragment (see (3)). The weighted average described in (1) is computed using simple hardware blending. Using this method, high-quality global illumination can be displayed at interactive frame rates.

Replacement/deletion method. As described in the previous sections, the flickering artifacts of the lighting come from the temporal discontinuities of the accuracy. Therefore, if a record cannot contribute to the current image (that is, it is out of the view frustum or occluded), it can be simply deleted instead of being replaced by a novel, up-to-date record. This avoids the generation and update of a "trail" of records following dynamic objects (Fig. 9), hence reducing the memory and computational costs. In the context of radiance cache splatting [24], this decision can be easily made using hardware occlusion queries: During the last frame of the lifespan of record K, an occlusion query is issued as the record is rasterized. In the next frame, valid records are first rendered. If a record K is now obsolete, the result of the occlusion query is read from the GPU. If the number of covered pixels is 0, the record is discarded. Otherwise, a new record l is computed at location $\mathbf{p_l} = \mathbf{p_K}$.

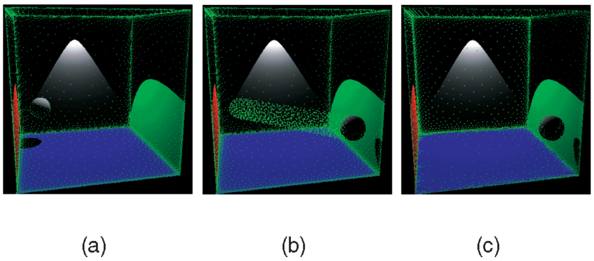


Fig. 9. The sphere moves from the left to the right of the Cornell Box. (a) At time 1, records (represented by green points) are generated to compute the global illumination solution. When the sphere moves, new records are created to evaluate the incoming radiance on the sphere. (b) If every record is permanently kept up to date, a "trail" of records lies on the path of the dynamic sphere. (c) Using our method, only useful records are updated.

The hardware occlusion queries are very useful, but they suffer from high latency. However, in our method, the result of a query is not needed immediately. Between the query issue and the reading of the record coverage, the renderer renders the other records, then switches the scene to next frame and renders valid records. In practice, the average latency appeared to be negligible (less than 0.1 percent of the overall computing time). Besides, in our test scenes, this method reduces the storage and computational costs by up to 25-30 percent.

6 Results

This section discusses the results obtained using our method and compares them with the classical method in which a new cache is computed for each frame. This latter method is referred to as *per-frame computation* in the remainder of this section. The images, videos, and timings have been generated using a 3.8 GHz Pentium 4 with 2 Gbytes of RAM and an nVidia GeForce 7800 GTX with 512 Mbytes. The scene details and timings are summarized in Table 2. The animations are presented in the accompanying video.

Cube in a box. This very simple diffuse scene (Fig. 12a) exhibits high flickering when no TGs are used. Along with a significant speedup, our method reduces the flickering artifacts by using extrapolated TGs. Such artifacts are unnoticeable with interpolated gradients. The animations are generated using $a^t = 0.05$ and a maximum lifespan of

TABLE 2
Test Scenes and Timings

Scene	Nb. Poly	Nb. Frames	Per-Frame Comp.	Our Method	Speedup
Cube in a Box	24	400	2048s	269s	7.62
Moving Light	24	400	2518s	2650s	0.95
Flying Kite	28K	300	5109s	783s	6.52
Japanese Interior	200K	750	13737s	7152s	1.9
Spheres	64K	200	3189s	753s	4.24

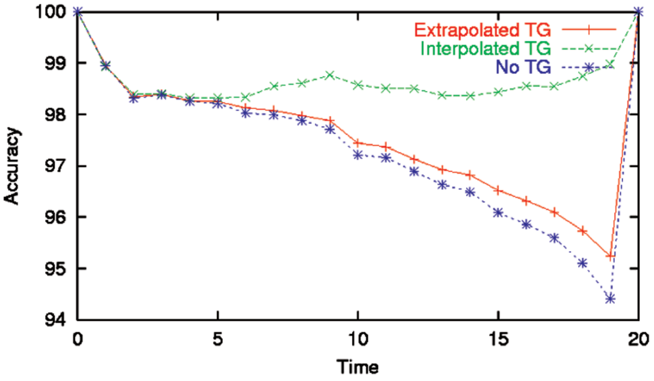


Fig. 10. Plot of the temporal accuracy as a function of time obtained in scene Cube in a Box by creating records at time 0 and extrapolating their value until time 19. The accuracy value at a time t is obtained by computing the error between the extrapolated values and the actual lighting at time t. At time 20, the existing records are discarded and replaced by up-to-date records with maximum accuracy. The TGs provide a better approximation compared to the approach without those gradients. Using interpolated gradients, the accuracy is continuous and remains above 98 percent.

20 frames. The per-frame computation requires 772,000 records to render the animation. In our method, only 50,000 records are needed, yielding a memory load of 12.4 Mbytes. Fig. 10 shows the accuracy values obtained with and without TGs. The remaining flickering of the extrapolated TGs are due to the discontinuities of accuracy. Since our aim is high-quality rendering, the following results focus on interpolated TGs that avoid discontinuities.

Moving light. A similar scene (Fig. 12b) illustrates the behavior of our algorithm in the context of dynamic light sources. The bottom of the box is tiled to highlight the changes of indirect lighting. Due to the highly dynamic indirect lighting, the lifespan of the records is generally very short, yielding frequent updates of irradiance values. Compared to per-frame computation, our method renders the animation with higher quality in a comparable time.

Flying kite. In a more complicated textured scene (Fig. 12c), our algorithm also provides a significant quality improvement while drastically reducing the computation time. In the beginning of the animation, the change of indirect lighting is small, and, hence, the records can be reused in several frames. However, when the kite comes down, its dynamic reflection on the ceiling and wall is clearly noticeable. Using our temporal weighting function, the global illumination solution of this zone is updated at a fast pace, avoiding ghosts in the final image (Fig. 11).

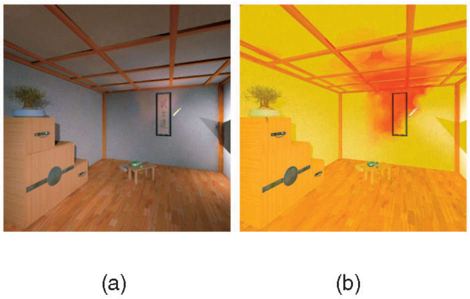


Fig. 11. (a) Actual sampling frame. When computing the global illumination solution for the current frame, our method estimates where the lighting changes. (b) Records lifespan. The lifespan of each generated record is computed by estimating the future change of lighting. Green and red colors respectively represent long and short lifespans.

Japanese interior. In this more complex scene (Fig. 12d), the glossy and diffuse objects are rendered using, respectively, the radiance and irradiance caching algorithms. The animation illustrates the features of our method: Dynamic nondiffuse environment and important changes of indirect lighting. In the beginning of the animation, the scene is lit by a single dynamic light source. In this case, TGs suppress the flickering artifacts present in the per-frame computation but do not provide a significant speedup ($1.25\times$). In the remainder of the animation, most of the environment is static, even though some dynamic objects generate strong changes in the indirect illumination. Our TGs take advantage of this situation by adaptively reusing records in several frames. The result is the elimination of flickering and a significant speedup (up to $9\times$) compared to per-frame computation. During the generation of this animation, the average latency introduced by occlusion queries is 0.001 percent of the overall rendering time.

Spheres. This scene features complex animation with 66 diffuse and glossy bouncing spheres and a glossy back wall (Fig. 12e). Our method eliminates the flickering while reducing the computational cost of a factor 4.24. We used a temporal accuracy value $a^t = 0.05$ and a maximum record lifespan $\delta_{t_{max}} = 5$.

Comparison with Monte Carlo path tracing. As the (ir)radiance caching algorithms introduce low-frequency spatial errors, our method introduces low-frequency temporal errors. Therefore, the obtained images contain both spatial and temporal errors. In a sequence of 100 images of the *Cube in a Box* scene, the average root-mean-square (RMS) error between our method and the reference solution is 0.139

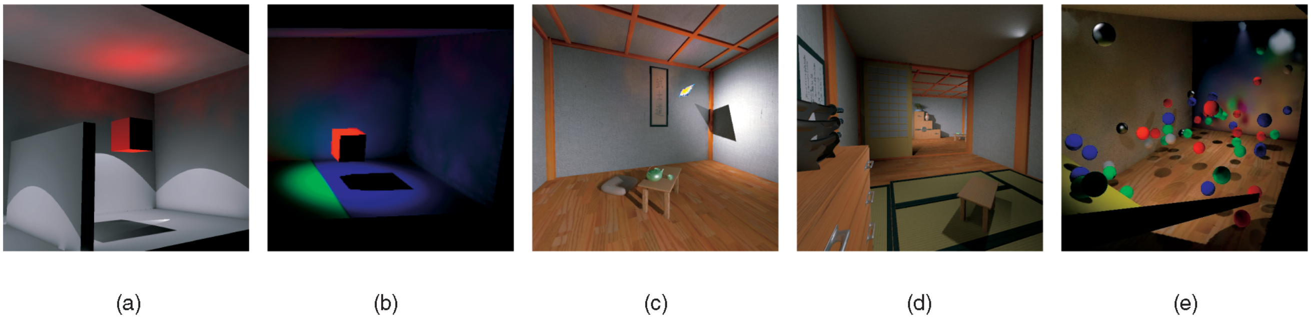


Fig. 12. Images of scenes discussed in Section 6. (a) Cube in a Box. (b) Moving light. (c) Flying kite. (d) Japanese Interior. (e) Spheres.

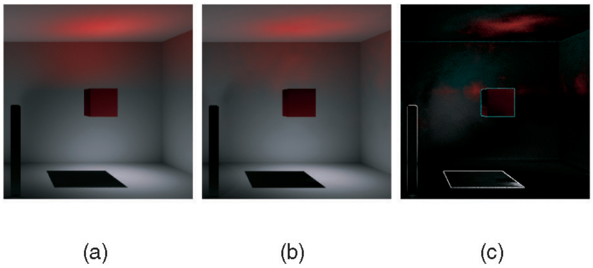


Fig. 13. Images obtained using (a) Monte Carlo path tracing (16,000 rays per hemisphere at each pixel) and (b) our method. (c) The image obtained by differencing (a) and (b). The pixel values in (c) are multiplied by 5 to highlight the differences.

(Fig. 13). Even though the results obtained using our method exhibit differences compared to the reference solution, the images obtained are a reliable estimate of the global illumination solution.

Computational overhead of reprojection. During the computation of a record, our method evaluates the value of the incoming lighting for both current and next time steps. As shown in Section 4.7, the estimation of the future incoming lighting is performed by simple reprojection. Therefore, the related computational overhead is independent of the scene geometry. In our tests, each record was computed at resolution 64×64. In our system, the reprojection is performed in approximately 0.46 ms. For comparison, the time required to compute the actual incoming lighting at a given point in our 200,000 polygon scene is 4.58 ms. In this case, the overhead due to the reprojection is only 10 percent of the cost of the actual hemisphere sampling. Even though this overhead is not negligible, our estimate enables us reduce the overall rendering time by reusing the records in several frames.

7 Conclusion

In this paper, we presented a novel method for exploiting temporal coherence in the context of irradiance and radiance caching. We proposed an approach for sparse sampling and interpolation of the incoming radiance in the temporal domain. We defined a temporal weighting function and temporal gradients, allowing a simple and accurate temporal interpolation of incoming radiance values. The results show both a significant speedup and an increased quality compared to per-frame computation. Due to our sparse temporal sampling, the incoming radiance values for the entire animation segment can be stored within the main memory. As our method provides a significant quality improvement and is easy to implement, we believe that our approach can be integrated in production renderers for efficient rendering of animated scenes.

Future work includes the design of a more accurate estimation method for extrapolated temporal gradients. Such a method will find use in on-the-fly computation of indirect lighting during interactive sessions. Another improvement would consist in designing an efficient method for faster aging of the records located near newly created records for which important changes have been detected. This would avoid the need for a user-defined maximum validity time while guaranteeing the absence of global illumination ghosts.

References

[1] D.R. Baum, J.R. Wallace, M.F. Cohen, and D.P. Greenberg, "The Back-Buffer Algorithm: An Extension of the Radiosity Method to Dynamic Environments," *The Visual Computer,* vol. 2, no. 5, pp. 298-306, 1986.

[2] X. Pueyo, D. Tost, I. Martín, and B. Garcia, "Radiosity for Dynamic Environments," *J. Visualization and Computer Animation,* vol. 8, no. 4, pp. 221-231, 1997.

[3] G. Besuievsky and X. Pueyo, "Animating Radiosity Environments through the Multi-Frame Lighting Method," *J. Visualization and Computer Graphics,* vol. 12, pp. 93-106, 2001.

[4] C. Damez, K. Dmitriev, and K. Myszkowski, "Global Illumination for Interactive Applications and High-Quality Animations," *Proc. Ann. Conf. European Assoc. Computer Graphics (Eurographics '02),* pp. 55-77, Sept. 2002.

[5] K. Dmitriev, S. Brabec, K. Myszkowski, and H.-P. Seidel, "Interactive Global Illumination Using Selective Photon Tracing," *Proc. Eurographics Workshop Rendering,* pp. 25-36, 2002.

[6] T. Tawara, K. Myszkowski, and H.-P. Seidel, "Exploiting Temporal Coherence in Final Gathering for Dynamic Scenes," *Proc. Computer Graphics Int'l Conf.,* pp. 110-119, June 2004.

[7] M. Smyk, S.-I. Kinuwaki, R. Durikovic, and K. Myszkowski, "Temporally Coherent Irradiance Caching for High Quality Animation Rendering," *Proc. Ann. Conf. European Assoc. for Computer Graphics (Eurographics '05),* vol. 24, no. 3, pp. 401-412, 2005.

[8] G.J. Ward, F.M. Rubinstein, and R.D. Clear, "A Ray Tracing Solution for Diffuse Interreflection," *Proc. Int'l Conf. Computer Graphics and Interactive Techniques (SIGGRAPH '88),* pp. 85-92, 1988.

[9] J. Křivánek, P. Gautron, S. Pattanaik, and K. Bouatouch, "Radiance Caching for Efficient Global Illumination Computation," *IEEE Trans. Visualization and Computer Graphics,* vol. 11, no. 5, pp. 550-561, Sept.-Oct. 2005.

[10] P. Dutre, P. Bekaert, and K. Bala, *Advanced Global Illumination.* AK Peters, 2003.

[11] M. Pharr and G. Humphreys, *Physically Based Rendering.* Morgan Kaufmann, 2004.

[12] B. Walter, G. Drettakis, and S. Parker, "Interactive Rendering Using the Render Cache," *Proc. Eurographics Workshop Rendering,* pp. 235-246, 1999.

[13] B. Walter, G. Drettakis, and D.P. Greenberg, "Enhancing and Optimizing the Render Cache," *Proc. Eurographics Workshop Rendering,* pp. 37-42, 2002.

[14] G. Bishop, H. Fuchs, L. McMillan, and E.J.S. Zagier, "Frameless Rendering: Double Buffering Considered Harmful," *Proc. Int'l Conf. Computer Graphics and Interactive Techniques (SIGGRAPH '94),* pp. 175-176, 1994.

[15] A. Dayal, C. Woolley, B. Watson, and D. Luebke, "Adaptive Frameless Rendering," *Proc. Eurographics Workshop Rendering,* pp. 265-276, 2005.

[16] P. Tole, F. Pellacini, B. Walter, and D.P. Greenberg, "Interactive Global Illumination in Dynamic Scenes," *Proc. Int'l Conf. Computer Graphics and Interactive Techniques (SIGGRAPH '02),* pp. 537-546, 2002.

[17] H.W. Jensen, *Realistic Image Synthesis Using Photon Mapping.* AK Peters, 2001.

[18] T. Tawara, K. Myszkowski, and H.-P. Seidel, "Localizing the Final Gathering for Dynamic Scenes Using the Photon Map," *Proc. Vision, Modeling, and Visualization Conf. (VMV '02),* 2002.

[19] P. Gautron, J. Křivánek, S. Pattanaik, and K. Bouatouch, "A Novel Hemispherical Basis for Accurate and Efficient Rendering," *Proc. Eurographics Symp. Rendering,* pp. 321-330, 2004.

[20] G.J. Ward and P.S. Heckbert, "Irradiance Gradients," *Proc. Eurographics Workshop Rendering,* pp. 85-98, 1992.

[21] J. Křivánek, P. Gautron, K. Bouatouch, and S. Pattanaik, "Improved Radiance Gradient Computation," *Proc. Spring Conf. Computer Graphics (SCCG '05),* pp. 149-153, 2005.

[22] E. Tabellion and A. Lamorlette, "An Approximate Global Illumination System for Computer-Generated Films," *Proc. Int'l Conf. Computer Graphics and Interactive Techniques (SIGGRAPH '04),* Aug. 2004.

[23] J. Křivánek, K. Bouatouch, S.N. Pattanaik, and J. Žára, "Making Radiance and Irradiance Caching Practical: Adaptive Caching and Neighbor Clamping," *Proc. Eurographics Symp. Rendering,* 2006.

[24] P. Gautron, J. Křivánek, K. Bouatouch, and S. Pattanaik, "Radiance Cache Splatting: A GPU-Friendly Global Illumination Algorithm," *Proc. Eurographics Symp. Rendering,* June 2005.

[25] C.M. Goral, K.E. Torrance, D.P. Greenberg, and B. Battaile, "Modelling the Interaction of Light Between Diffuse Surfaces," *Proc. Int'l Conf. Computer Graphics and Interactive Techniques (SIGGRAPH '84),* pp. 212-222, July 1984.

[26] C. Damez, "Simulation Globale de l'Eclairage Pour des Sequences Animees Prenant en Compte La Coherence Temporelle," PhD dissertation, Univ. Joseph Fourier, 2001.

[27] C. Damez, F.X. Sillion, and N. Holzschuch, "Space-Time Hierarchical Radiosity with Clustering and Higher-Order Wavelets," *Proc. Ann. Conf. European Assoc. for Computer Graphics (Eurographics '01),* pp. 129-141, Sept. 2001.

[28] G. Besuievsky and M. Sbert, "The Multi-Frame Lighting Method: A Monte Carlo-Based Solution for Radiosity in Dynamic Environments," *Proc. Eurographics Workshop Rendering,* pp. 185-194, 1996.

[29] I. Martín, X. Pueyo, and D. Tost, "Frame-to-Frame Coherent Animation with Two-Pass Radiosity," *IEEE Trans. Visualization and Computer Graphics,* vol. 9, no. 1, pp. 70-84, Jan.-Mar. 2003.

[30] G. Drettakis and F.X. Sillion, "Interactive Update of Global Illumination Using a Line-Space Hierarchy," *Proc. Int'l Conf. Computer Graphics and Interactive Techniques (SIGGRAPH '97),* vol. 31, no. 3, pp. 57-64, 1997.

[31] X. Granier and G. Drettakis, "A Final Reconstruction Approach for a Unified Global Illumination Algorithm," *ACM Trans. Graphics,* vol. 23, no. 2, pp. 163-189, 2004.

Pascal Gautron received the PhD degree from the Institut de Recherche en Informatique et Systèmes Aléatoires/Institut National de Recherche en Informatique et en Automatique (IRISA/INRIA) Rennes and in collaboration with the University of Central Florida. His research focused on the development of fast GPU-accelerated global illumination computation methods based on the irradiance caching algorithm. He is now a postdoctoral researcher at France Telecom R&D Rennes, and his current research work is toward the rendering of photorealistic virtual agents in real time.

Kadi Bouatouch received the degree in electronics and automatic systems engineering from Ecole Nationale Superiore d'Electricité et Mécanique (ENSEM) in 1974, the PhD degree in 1977, and the higher doctorate degree in computer science in the field of computer graphics in 1989. He is currently a professor at the University of Rennes 1, France, and a researcher at the Institut de Recherche en Informatique et Systèmes Aléatoires (IRISA). He is working on global illumination, lighting simulation for complex environments, parallel radiosity, and augmented reality. He is a member of Eurographics, ACM, and the IEEE. He was a member of the program committees of several conferences and workshops and referee for several computer graphics journals, such as *The Visual Computer*, *IEEE Computer Graphics and Applications*, *IEEE Transactions on Visualization and Computer Graphics*, *IEEE Transactions on Image Processing*, and so forth. He has also acted as a referee for many conferences and workshops.

Sumanta Pattanaik received the PhD degree in computer science from Birla Institute of Technology and Science (BITS), Pilani. He is an associate professor of computer science at the University of Central Florida. His research interests include realistic image synthesis and real-time realistic rendering. He is a member of the IEEE, ACM SIGGRAPH, and Eurographics. He is the graphics category editor of *ACM Computing Reviews*.

Fast and Detailed Approximate Global Illumination by Irradiance Decomposition

Okan Arikan
okan@cs.berkeley.edu
University of California, Berkeley

David A. Forsyth
daf@cs.uiuc.edu
University of Illinois, Urbana-Champaign

James F. O'Brien
job@cs.berkeley.edu
University of California, Berkeley

Abstract

In this paper we present an approximate method for accelerated computation of the final gathering step in a global illumination algorithm. Our method operates by decomposing the radiance field close to surfaces into separate far- and near-field components that can be approximated individually. By computing surface shading using these approximations, instead of directly querying the global illumination solution, we have been able to obtain rendering time speed ups on the order of 10 compared to previous acceleration methods. Our approximation schemes rely mainly on the assumptions that radiance due to distant objects will exhibit low spatial and angular variation, and that the visibility between a surface and nearby surfaces can be reasonably predicted by simple location- and orientation-based heuristics. Motivated by these assumptions, our far-field scheme uses scattered-data interpolation with spherical harmonics to represent spatial and angular variation, and our near-field scheme employs an aggressively simple visibility heuristic. For our test scenes, the errors introduced when our assumptions fail do not result in visually objectionable artifacts or easily noticeable deviation from a ground-truth solution. We also discuss how our near-field approximation can be used with standard local illumination algorithms to produce significantly improved images at only negligible additional cost.

Keywords: Final gather, approximate global illumination, spherical harmonics.

CR Categories: I.3.3 [COMPUTER GRAPHICS]: Picture / Image Generation, I.3.7 [COMPUTER GRAPHICS]: Three Dimensional Graphics and Realism

1 Introduction

Current rendering techniques can produce highly compelling, beautiful images that realistically capture a variety of interesting effects such as color bleeding between nearby surfaces or indirect shadows due to strong reflected illumination. Capturing these effects requires computing how light propagates and reflects within the scene environment, and the term global illumination refers to the class of algorithms that perform these computations.

Unfortunately, computing a global illumination solution is a fundamentally difficult task, requiring solutions to integral equations that recursively express the radiance leaving a surface as a function of the radiance leaving other surfaces. The algorithms available for computing these solutions must either discretize the environment or employ some form of stochastic sampling. However, producing solutions free of objectionable discretization artifacts and sampling noise often demands impractically large computation times and/or storage space.

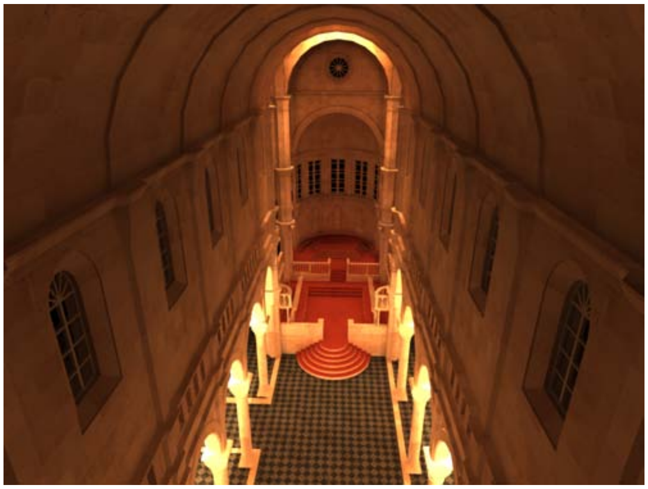


Figure 1: An image rendered with our algorithm. Notice the fast change in global illumination near surface relief. This image takes about 5 minutes to render (from end to end) on a desktop PC.

A widely adopted technique, called final gathering, approximates a high quality global illumination solution by combining a relatively coarse global illumination solution with high resolution sampling of select surfaces. First, a standard global illumination method, such as photon mapping, is used to compute a rough solution. Afterward, an image is generated using Monte Carlo techniques to integrate indirect irradiance from the rough solution. Monte Carlo integration is performed only for surfaces that are directly visible, or clearly reflected. The result captures illumination features of the global solution, but because diffuse and glossy surfaces blur reflections, objectionable artifacts present in the rough solution do not appear in the final image. Although final gathering provides a large overall improvement in total rendering time (for an equal quality result), one typically finds that the cost of the final gather dominates the cost of computing the coarse global illumination solution. As a result, improving the speed of the final gather is an effective way of reducing overall rendering time.

One such approach is irradiance caching, which takes advantage of the smoothing property of diffuse surfaces by caching the value of the irradiance integral at points on surfaces. The cached values are then re-used for nearby points using scattered data interpolation. Details, such as precise sampling regime and the interpolation scheme vary from implementation to implementation.

However, irradiance caches are not efficient around geometric detail because they must maintain a denser set of samples to account

ACM 0730-0301/05/0700-1108

for the potentially rapidly changing irradiance. One can arbitrarily limit the sample density, but doing so often obliterates perceptually important lighting variations that reveal surface detail and geometric texture.

In this paper we propose an approximate method for accelerated final gathering that splits the irradiance integral into separate far- and near-field components that can then be approximated individually. As the name suggests, the far-field term accounts for the power coming from distant surfaces, while the near-field term accounts for the reflections and occlusions due to nearby geometry. Our approximation schemes rely mainly on the assumptions that radiance due to distant objects will exhibit low spatial and angular variation, and that the visibility between a surface and nearby surfaces can be reasonably predicted by simple location- and orientation-based heuristics. Motivated by these assumptions, our far-field scheme uses scattered-data interpolation with spherical harmonics to represent spatial and angular variation, and our near-field scheme employs an aggressively simple visibility heuristic. The assumptions underlying our far field scheme are similar to those which motivate irradiance caching, except we cache radiance, rather than irradiance. Since only the radiance due to distant objects is cached our required sampling density is not impacted by local geometry.

Using this decomposition, we are able to approximate final gathering in complex scenes displaying a variety of global illumination effects (see Figure 1). Our tests show that the method uses far fewer visibility queries (ray traces) and achieves an order of magnitude speedup on average compared to irradiance caching. For our test scenes, the errors introduced at locations where the underlying assumptions fail do not result in visually objectionable artifacts or easily noticeable deviation from a ground-truth solution.

We also discuss how our near-field approximation can be used with standard local illumination algorithms to produce significantly improved images at only negligible additional cost. This local enhancement should be particularly useful in production environments where artists often employ unrealistic light placement to achieve a desired look. Like ambient occlusion shading, our local enhancement greatly enhances the realistic appearance of an object but it is significantly cheaper to compute.

2 Related Work

Global illumination has been one of the most heavily researched branches of computer graphics. A thorough review of available methods cannot be presented here, but [Sillion and Puech 1994] and [Dutré et al. 2003] provide a good overview of finite element and Monte Carlo based global illumination techniques. We use photon mapping [Jensen 2001], to generate the global illumination solutions used with our final gathering algorithm.

Originally, final gathering was developed for use with radiosity methods to obtain visually pleasing results from blocky piecewise-constant solutions. [Lischinski et al. 1993] introduced an object space refinement method for increasing the visual quality of the solution of a hierarchical radiosity step. [Rushmeier 1988], [Rushmeier et al. 1993] use a coarse geometric model for a radiosity solution which is queried by a Monte Carlo algorithm to create the final picture. [Zimmerman and Shirley 1995] decrease the variance caused by Monte Carlo integration by re-classifying bright reflecting surfaces in a coarse radiosity solution as light sources to ensure that those surfaces are sampled. [Scheel et al. 2001] demonstrated that a final gather could be accelerated by using the link information from a prior radiosity step to identify important senders. [Scheel et al. 2002] also uses the link information to find the senders whose power contribution can be interpolated. A similar idea has also been explored by [Bekaert et al. 1998] where per pixel final gather was obtained using a Monte Carlo integration scheme which uses radiosity solution for importance sampling. All of these methods generate a visually pleasing result from a coarse global illumination solution. However, they still result in an expensive final gather step that must compute expensive irradiance for each pixel in an image.

In an influential paper, [Ward et al. 1988] introduced the concept of performing very accurate final gather on surfaces only at scattered points and then interpolating those values. Known as irradiance caching, their method adapts the sample locations based on how quickly the sampled final gather result changes. A later paper [Ward and Heckbert 1992] refined the algorithm to decrease the discontinuities due to the insertion of new samples while rendering.

Instead of caching irradiance, our far-field approximation caches incident radiance. Our near-field correction then adjusts the irradiance integral obtained from the cached representation by explicitly accounting for nearby geometry. This will allows us to cache the incident radiance at a relatively small number of locations and still approximate the high-frequency detail in indirect illumination. A similar idea for extending irradiance caches by caching the incident radiance has been explored by [Krivanek et al. 2005]. However, their method aims at handling glossy surfaces whereas our method attacks oversampling. [Tabellion and Lamorlette 2004] presents an improvement over irradiance caches, but they must also sample more densely around geometric detail and their limit on the sampling density can lead to smoothed interpolation. Similar to our work, [Greger et al. 1998] stores incident radiance in space, but their method suffers from the same dense sampling problem of irradiance caches.

The behavior of diffuse surfaces in close proximity to other surfaces (such as corners) has been explored in [Rathsfeld 1999] and [Atkinson 2000]. Their results confirm the existence of high-frequency change, or light reflexes, around corners.

The quality of the results from many of the methods above can be explained by research on the relationship between the appearance of diffuse surfaces and incident illumination. [Epstein et al. 1995] and [Basri and Jacobs 2000] showed that the color of a diffuse surface was a function of the low dimensional representation of the incident illumination. [Ramamoorthi and Hanrahan 2001b] used low order spherical harmonics, and demonstrated harmonics to be a good and compact representation. [Gautron et al. 2004] extend the idea onto hemispherical basis functions which provide a more accurate representation for functions defined on hemispheres. Spherical harmonics have also been successfully used in [Ramamoorthi and Hanrahan 2001a] to represent environment maps for diffuse surfaces. This idea has been expanded in [Sloan et al. 2002] with the introduction of transfer functions to take self occlusions or interreflections into account. However such methods require expensive precomputation to obtain the transfer functions.

3 Overview

The final gather in a rendering process computes shading for the surfaces that are either directly visible, or visible by a mirror-like reflection. Presuming that the BRDF of these surfaces can be separated into diffuse and specular components, we observe that the directional dependence of the specular component makes it well suited for fast computation using standard sampling methods while, in contrast, computing the diffuse reflection by sampling can be quite costly. Accordingly, our method for accelerating final gathering focuses on speeding up this bottleneck by replacing the costly irradiance integral used for computing diffuse illumination with a fast approximation.

Let P be a point on some surface we wish to shade, with normal vector N. The incoming radiance at P from a direction ω is $L(P,\omega)$. If we define ρ as the diffuse reflectivity (albedo) of the surface, the diffusely reflected light, or *radiosity*, is defined as follows:

$$B = \frac{\rho}{\pi} \int_{\Omega} (N \quad \omega)\, L(P,\omega) d\omega \tag{1}$$

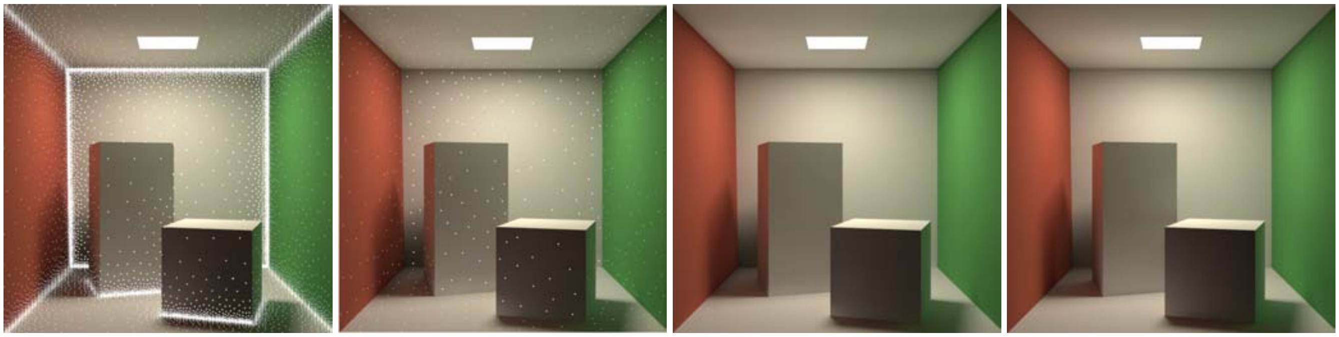


Figure 2: The leftmost image shows the locations of the irradiance cache samples as white dots. The samples are concentrated around the corners to accurately capture the swift change in illumination. The center-left image shows the locations of our spherical harmonic samples. Since the high-frequency changes in diffuse brightness around corners are captured explicitly by our correction term I_N, we require far fewer samples. The image on the center-right shows the sample points removed to show the final image computed using our technique, and the rightmost image shows the ground truth, which is obtained by estimating the irradiance integral (equation 1) using Monte Carlo techniques.

The integral in this equation computes the power that P receives and is called *irradiance*.

The definition of irradiance is implicitly recursive: Finding the incoming radiance $L(P,\omega)$ involves computing the irradiance integral at the source of the radiance. One could perform the computation using path integration (*Monte Carlo* variants) or discretize the scene and convert the problem into a linear system (*radiosity* variants, not to be confused with the term for diffusely reflected light). However, these methods are often too expensive to obtain high quality solutions by themselves.

As we have discussed, the common practice of final gathering uses a Monte Carlo technique to compute the irradiance integral at visible surfaces, but it avoids costly recursion by using a precomputed coarse global illumination solution as a proxy for $L(P,\omega)$. This practice is called Final Gathering, because it "gathers" from the coarse global illumination solution.

Final gathering is expensive because it involves shooting many sample rays for each point to be shaded. Although computing the irradiance integral is expensive, it can often be re-used: If the value of the integral is not changing very rapidly on a surface, it can be computed sparsely, and be interpolated using scattered data interpolation techniques. This technique is often called irradiance caching [Ward et al. 1988; Ward and Heckbert 1992].

As a point moves across a surface, far away surfaces move less on the incident hemisphere of the point than nearby surfaces (parallax effect). This means the irradiance integral in Equation 1 does not change very rapidly (as P moves on a surface) if the source of the incoming radiance $L(P,\omega)$ is far away. Similarly, if there are other surfaces nearby, then these surfaces can reflect and obstruct light and create rapid changes in the irradiance integral. For this reason, irradiance caching methods are forced to form spatially dense caches in regions of geometric detail (as shown in Figure 2.). Even a phenomenon as familiar as a corner attracts many cache samples to capture potential changes in the irradiance.

Let us designate all the visible points that are farther from P than a distance threshold α as *distant*, and all other points as *nearby*. The incident hemisphere Ω over P can then be partitioned into Ω_D and Ω_N ($\Omega = \Omega_D \cup \Omega_N$) according to whether a distant or near point projects onto a given portion of the hemisphere. Given this distinction, we can split the irradiance integral into two components: Power coming from distant surfaces and power coming from nearby surfaces. Power coming from distant surfaces can be handled easily, because it does not change rapidly on surfaces. Power coming from nearby surfaces can also be handled easily, because it is by definition, a local computation. Formally, we write:

$$\begin{aligned} B &= \frac{\rho}{\pi}\int_{\Omega_D}(N\ \ \omega)L_D(P,\omega)d\omega \\ &+ \frac{\rho}{\pi}\int_{\Omega_N}(N\ \ \omega)L_N(P,\omega)d\omega \qquad (2) \end{aligned}$$

Here we denote the incident radiance from distant points with L_D and from nearby points with L_N. In practice, it is difficult to determine Ω_D directly. However finding Ω_N is relatively straightforward by looking at the nearby scene triangles and finding the solid angle they subtend. These properties suggest that we define $L_D^\star$ — an extension of L_D from Ω_D to Ω by ignoring any occlusions due to nearby surfaces. Now $L_D^\star = L_D$ on Ω_D so we can rewrite the integral above as follows:

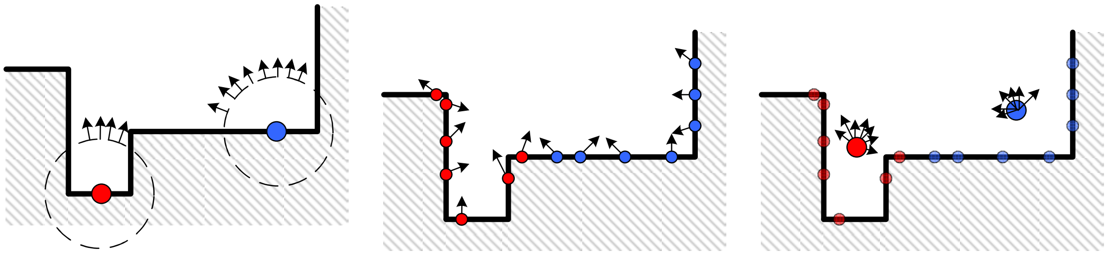
$$\begin{aligned} B &= \frac{\rho}{\pi}\int_{\Omega}(N\ \ \omega)L_D^\star(P,\omega)d\omega \\ &+ \frac{\rho}{\pi}\int_{\Omega_N}(N\ \ \omega)[L_N(P,\omega)\quad L_D^\star(P,\omega)]d\omega \\ &= \frac{\rho}{\pi}(I_D+I_N) \qquad (3) \end{aligned}$$

In equation 3, the I_D can be thought as the irradiance due to distant surfaces (*distant term*) and I_N can be thought as a correction term that accounts for the power being reflected or being occluded due to nearby surfaces (*near term*). Sections 4 and 5 will explain how these terms respectively can be approximated efficiently. Section 7 will discuss the effect of the parameter α which distinguishes distant from nearby surfaces.

4 Distant Term

The distant illumination $I_D = \int_\Omega(N\ \ \omega)L_D^\star(P,\omega)d\omega$ does not change rapidly over surfaces. We could cache I_D sparsely and use scattered data interpolation techniques, an approach that would be similar to irradiance caching but with the difference of separating irradiance due to far and near surfaces. However, rather than caching the value of I_D (irradiance), we cache a compact representation of $L_D^\star$ (radiance). Caching $L_D^\star$ instead of I_D will allow us to compute the second term of the correction term which subtracts the portion of I_D that corresponds to distant surfaces obscured by nearby ones. It also facilitates using bump or displacements maps where rapid variation in the surface normal causes rapid changes in I_D but not $L_D^\star$. [Krivanek et al. 2005] demonstrates additional advantages of storing the incident radiance and describes how the information can be used to efficiently handle glossy surfaces.

$L_D^\star$ is a function of position and direction: It gives us the incident distant radiance from a given direction arriving at a given position. Spherical harmonics provide a nice representation for such functions defined over a sphere. As demonstrated by [Ramamoorthi and Hanrahan 2001b], for the diffuse component of a BRDF,



(a) (b) (c)

Figure 3: (a) Constructing spherical harmonic samples by shooting rays from a single location can undersample certain directions. Since we only consider sources of incident radiance that are farther away from the sample point than α, a sample computed at the red point would only have samples coming from a very narrow range of directions. The blue sample also has a similar problem because it does not contain any rays shooting towards the right. (b) To avoid this situation, we spread the origin of our rays onto surfaces. We still consider the intersections that are far enough away. But when we compute the harmonic coefficients, we assume all the rays originate from the geometrical average of the origins of the rays. (c) This way when we compute the harmonic sample for the red point we do not suffer from the undersampling problems.

only 9 coefficients provide a sufficient representation of the incoming radiance, and so we use these 9 coefficients for representing $L_D^\star$ at a fixed point. Given a query point, the distant incident radiance is obtained by interpolating spherical harmonic coefficients from nearby samples. Note that our query points will be points on surfaces where we wish to perform a shading calculation, but the locations where we cache $L_D^\star$ may be arbitrary points in space, though presumably they will be near surfaces we wish to shade. Because spherical harmonics provide an orthonormal basis, I_D can be efficiently evaluated using a dot product of harmonic coefficients for $max(N \quad \omega, 0)$ and $L_D^\star(P,\omega)$.

We compute the projection of $L_D^\star$ onto the harmonic basis using a sampling fit procedure. The incident hemisphere Ω of P is randomly sampled and incoming radiance at these sample points are obtained by ray-cast queries into the coarse global illumination solution. Rays whose intersections are closer to P than α are discarded. The spherical harmonic coefficients approximating $L_D^\star$ are computed by a least-squares fit to these incident radiance samples.

4.1 Sampling

Irradiance caching algorithms typically cache irradiance values at specific points on the visible surfaces. The corresponding approach would be for us to select a set of surface points, sample the incoming radiance over a hemisphere centered at each of those points, and then store the resulting harmonic coefficients at each point. However, we have noticed that doing so can create ugly bias artifacts if our implicit assumptions about the smoothness of distant illumination fails. Instead we opt for a procedure that will produce a more benign smoothed solution.

We start with the set of all shading points that will need the global illumination computation[1]. We then attempt to fit a spherical harmonic to the incoming radiance for these points by shooting rays from random points in this set in random directions (in the incident hemisphere of the selected points). A single harmonic sample is not good enough to represent the incoming distant radiance if any of these rays intersects a surface within the bounding sphere of the samples. If this is the case, we split (using k-means) the set of shading points into two, and attempt to fit harmonics for each of the subsets recursively. A set is not split if the diameter of the bounding sphere for the shading points in the set is less than α, or if the rays do not intersect other surfaces within the bounding sphere. This method ensures that the harmonics samples will not be closer to each other than α and that the places without geometric detail will not create lots of harmonic samples. Spawning rays from random shading points corresponds to spreading the origins of the rays used to compute the harmonics onto the surfaces. (See Figure 3.)

The number of rays that we use to compute a harmonic sample is equivalent to the number of rays that irradiance caches use to compute a sample. In practice, we use 512 rays per harmonic in all of our examples. Once a harmonic sample is computed using least squares fit to the incident radiances coming along these rays, it is stored at the geometric center of the shading points in the set. For each sample, we also store the radius of the bounding sphere of the shading points for which the sample was collected. The samples are interpolated using scattered data interpolation, similar to irradiance caches: For a query point, we locate all the harmonic samples that are near the query point and perform a normalized distance weighted sum of the harmonic coefficients.

5 Near Term

While distant illumination generally exhibits slow variation over surfaces, the illumination arriving from, and occluded by, near surfaces can vary relatively rapidly. In fact, as illustrated by Figure 7, lighting variation induced by near surfaces often provides many of the cues that reveal the geometric structure of an object. As a result, the smoothness-dependent interpolation method we used for distant terms cannot work well for estimating the irradiance from near surfaces.

Instead we compute an approximation of I_N directly at each shading point. By definition, this computation is local and only involves surfaces within distance α. However, even limited to this small neighborhood, we have found the required visibility tests to be expensive. So rather than actually computing the visibility terms we make use of a fast, aggressive heuristic that approximates visibility based on the relative location and orientation of the surfaces.

First we rewrite the nearby correction term of equation 3 as a sum over all nearby triangles:

$$I_N \qquad \sum_{t \in T} FF(t)[B(t)/\pi \quad L_D^\star(P,\omega_t)] \tag{4}$$

In this equation T is the set of triangles whose centroids are closer to the shading point than α. We find such triangles using an octree decomposition of the scene. ω_t is the direction towards the centroid of the triangle, $FF(t)$ and $B(t)$ respectively stand for the form factor of the triangle t to P and its radiosity, which we assume is constant over the triangle.

To avoid problems caused by large triangles that have significant portions within and without a distance α, triangles larger than α are split into smaller ones. Splitting triangles also justifies treating the radiosity as constant over a triangle, as does the fact that we are referring to radiosity from the coarse global illumination solution. A third benefit of splitting triangles is that it prevents shading discontinuities that would be generated if a large triangle's center moved from inside α to outside α as we moved a small distance on

[1] If rendering a 640 by 480 image with one sample per pixel, we have $640 \times 480 = 307200$ shading points.

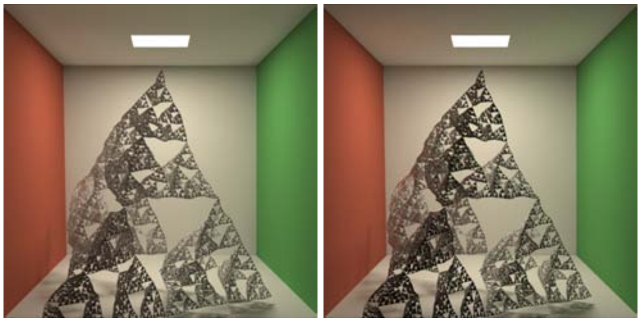


Figure 5: This scene demonstrates a deformed fractal in a Cornell box. The image on the left is the ground truth obtained using Monte Carlo integration. The image on the right is the result of our algorithm. The fractal violates our visibility assumption, because it contains many small triangles that reflect/occlude power from far away surfaces. Even in this contrived scenario, the errors introduced by our algorithm are not too distracting.

the rendered surface. For the scenes that we rendered, this process did not more than triple the number of scene triangles, because this splitting only attacks the big scene triangles.

Our visibility heuristic discards any triangles that are either backfacing to the shading point, P, or that are completely below the plane passing through P with normal N as those triangles cannot provide direct illumination to P. *We then assume that all other triangles within distance α are visible from P.* Although one can easily come up with example geometries where this aggressive, gross approximation would be wrong, we have found that it works surprisingly well in practice. Later, in section 7, we will discuss this approximation further.

Recall that $L_D^\star(P,\omega_t)$ is the approximation of incoming radiance from direction ω_t due to distant sources computed using our spherical harmonic-based interpolation. We also treat this quantity as constant over directions spanned by one of the split triangles, and justify the approximation with L_D's smoothness and the small size of the split triangles.

The remaining quantity that must be computed is the form factor between P and a triangle, $FF(t)$. It is this computation that would normally involve visibility information. Standard methods would render the triangles onto a hemi-cube or determine visibility by sampling. While we could easily use one of these methods in our form factor computation, doing so would reduce the speed improvement of our algorithm over irradiance caching. Instead, we assume that the triangles selected by our heuristic are all completely visible, and compute form factors using the analytical polygon to point form factor formula given by [Hottel and Saforim 1967].

6 The Whole Algorithm

Once a coarse global illumination pass has been completed, our algorithm constructs the image in two passes. In pass 1, we compute the spherical harmonics capturing the incident radiance due to distant surfaces. Because the spherical harmonic samples represent only the distant illumination and are used for the diffuse component of the BRDF, they can be scattered sparsely. The near-far threshold α determines the minimum spacing between these samples. In pass 1, we also compute the average radiosity value over all visible triangles and split them if necessary.

In pass 2, for every point we need to shade, we first interpolate the spherical harmonic coefficients to obtain a representation of the incident radiance due to distant surfaces ($L_D^\star$). We then compute the distant term (I_D) and the nearby correction term (I_N) by finding nearby triangles and computing the sum in equation 4. The final diffuse reflected power is obtained by equation 3.

7 Nature of the Approximation

As with any approximation method, the far- and near-term approximations we use can introduce error into the rendered image. For the interpolation method we use for distant illumination, the potential difficulty is that it may smooth over and obscure some perceptible illumination feature. For our simple visibility heuristic, we expect that it may be wrong and treat an occluded triangle as visible.

The potential problems due to our radiance interpolation are generally similar as those that may occur for irradiance caching. However, we gain some advantages by interpolating radiance rather than irradiance. In particular, irradiance changes due to variation in surface orientation will not cause problem for interpolating radiance.

Radiance interpolation can still encounter difficulties for sharp changes in indirect illumination caused by a distant occluder. In practice this happens rarely as most indirect illumination sources tend to appear as area sources and so create soft shadows. Nevertheless, when the situation does arise, our sampling procedure should tend to cluster samples near the illumination boundary to help resolve it.

Our simplistic near-field visibility heuristic will certainly produce errors, and on first consideration one maybe surprised that it works at all. In fact, we originally experimented with fast methods to actually compute local visibility correctly, and if desired one could still use such a method in conjunction with our incident radiance interpolation. However, we have found that for general scenes our much cheaper heuristic results in significant computation saving with little perceptible error.

One possible explanation for why it works well in practice comes from considering which situations break our visibility assumption. The most obvious is a surface that is surrounded by many mutually occluding surfaces. One can easily contrive such an example, but except for some particular examples such as hair or tree leaves, we believe the situation does not often arise in real-world scenes. Additionally, most rendered scenes contain significantly less geometric detail than the real world. Furthermore the commonly occurring examples often require some special treatment that would in any case preclude most generic acceleration methods. (For example, special shadow structures for hair, translucency for leaves.) We also observe that it may be difficult to see surfaces that are surrounded by many occluders so that shading errors on those surfaces may be somewhat excusable.

Finally, there are user-tunable parameters in both the near- and far-field approximations and these parameters can impact the quality of the rendered image. The criteria used for placing interpolation samples and building clusters affects the ability to resolve fast changes in distant lighting. The parameter α changes at what scale occlusions are computed but lighting smoothed, and the scale at which we ignore occlusions but do not smooth illumination. The effect of varying α is illustrated in Figure 4.

In Figure 6 we include a magnified difference image between the our method and the ground truth. The errors we make are concentrated around the corners where the nearby correction term is approximated. The reader is encouraged to evaluate our results perceptually against the ground truth.

In the next section, we discuss the qualitative properties of our approximation and demonstrate our results.

8 Results

To demonstrate our method, we have rendered several scenes and compared them to ground truth results obtained by estimating the irradiance integral in equation 1 using Monte Carlo techniques with a high number of samples. We also compared the images rendered using our algorithm to irradiance caching. Irradiance caching variants represent the current state of art in approximate final gathering and are available in many commercial rendering packages.

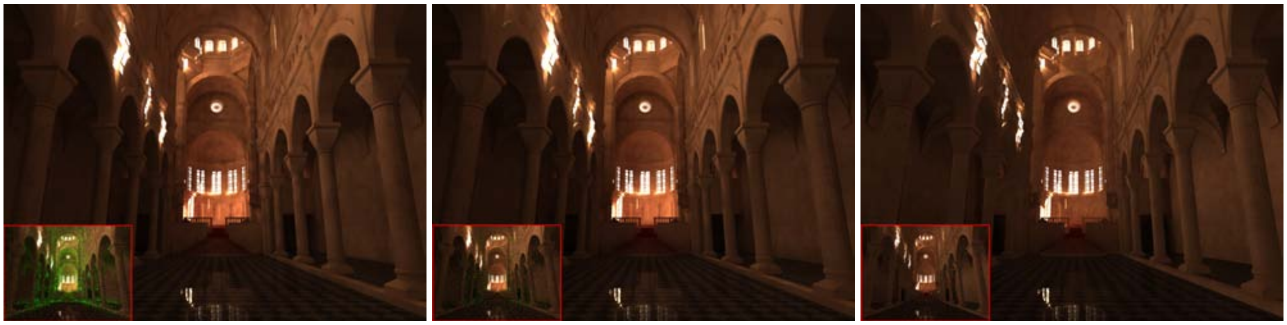


Figure 4: This figure demonstrates the effect of the α parameter (used by our approximation) on quality and rendering time. The left image is rendered with $\alpha = 0.25$ in 12 minutes. The overlaid thumbnail shows the locations of the spherical harmonic locations as green dots. The middle image is rendered using $\alpha = 1$ in 5 minutes. Notice that there are fewer samples which explains the speed difference. The rightmost image is rendered with $\alpha = 8$ in 6 minutes. This image uses even fewer spherical harmonic samples but takes longer. This is because more triangles are considered nearby due to the bigger α and the summation in equation 4 contributes to the rendering time. Some of the errors made by our approximation also become visible. For example, the undersides of the arches on the left side are brighter because of the illumination leaking from the brightly illuminated spots. This effort is due to the spherical harmonics being separated too far apart and not being able to represent the incident radiance for the points very accurately.

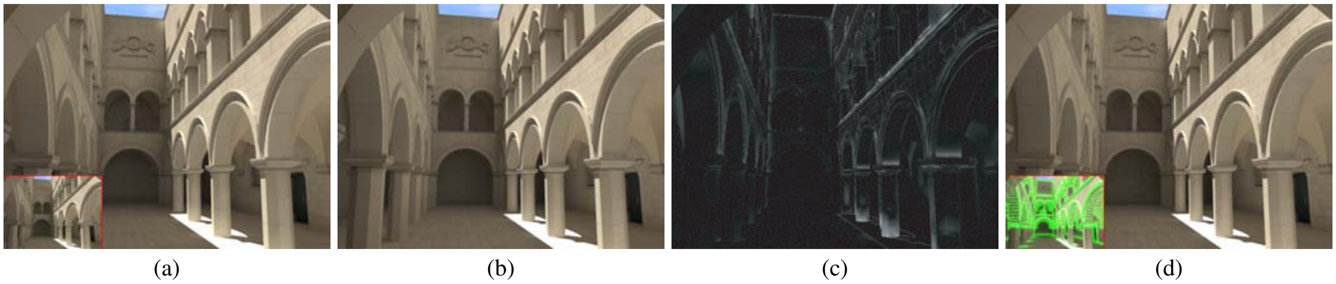


Figure 6: (a) shows the result of our algorithm. The overlaid thumbnail shows the locations of spherical harmonic samples as green dots. The minimum distance between the harmonic samples is our α parameter. (b) is the ground truth obtained using Monte Carlo integration of the irradiance integral. (c) shows a 4 times magnified difference image between (a) and (b). Most of the error is concentrated around high geometric detail. (d) shows the result obtained using irradiance caching. The overlaid thumbnail shows the irradiance cache samples as green dots. The cost of computing a spherical harmonic sample is about the same as obtaining one irradiance cache sample (the cost of the green dots in (a) and (d) is the same). The difference image between (b) and (d) can be found on the Siggraph 2005 DVD.

Figures 1, 4, 6, 8 demonstrate the quality of our approximation in complex environments, displaying complex diffuse interreflections[2]. Notice that high-frequency component of the diffuse interreflection is preserved. For example, in these figures, the niches are correctly shadowed by the nearby geometry. Similarly, the reflections due to nearby geometry are visible in Figure 6 where the pillars on the right touch the ground and around the top right corner of the atrium.

All our figures have been rendered on a dual Athlon 2.2+ system with 2GB of main memory. Only one rendering thread was used. All figures (except the Cornell box) have 1280x960 resolution with 4 samples per pixel. The source code for our method (and our implementation of irradiance caching) is available as a part of an open source renderer *Pixie* [3].

Figure 6 was rendered in 0:08 (photon mapping) + 1:24 (1^{st} pass) + 1:18 (2^{nd} pass) = 2:50 (mm:ss) using our method and in 37:04 using irradiance caching. This figure also compares our result against the ground truth. The same atrium model in Figure 8 was rendered in 10:01 (0:46 + 2:17 + 6:58) using our method and in 56:25 using irradiance caching. The time it takes to render this scene using only direct illumination is 8 minutes. The cathedral scene in Figure 4 was rendered in 5:51 (2:02 + 2:22 + 1:31) using our method and in 66:55 using irradiance caching. The night time version of this scene in Figure 1 was rendered in 7:41 (0:16 + 2:13 + 5:12) using our method and in 70:14 using irradiance caching.

Because our spherical harmonic samples are separated by at least α, we collect fewer samples than irradiance caching. The number of rays shot to compute one spherical harmonic sample is the same as the number of rays that we use to compute an irradiance cache sample. Thus the work of computing a spherical harmonic sample is equivalent to work done for each irradiance cache sample. Fewer samples translate to fewer rays traced and quicker results. On average our method is an order of magnitude faster than irradiance caching while providing the same quality images.

Our method computes the diffusely reflected power. This does not mean it is limited to diffuse surfaces however. Most BRDFs can be split into diffuse and specular lobes to capture reflections and refractions. The usually narrow specular lobes can be importance sampled efficiently. The difficult terms to compute are the diffuse and very glossy (wide) specular lobes. Our method attacks the common diffuse component of BRDF. We believe that by increasing the order of our spherical harmonics, the glossy specular surfaces can also be handled using our method as in [Krivanek et al. 2005].

9 Detail Enhancement

The nearby correction term itself is responsible for capturing the high-frequency reflection / refraction effects due to nearby scene geometry. The local effect can be added without global illumination to enhance the visual detail in a scene. The left portion of Figure 7

[2] The full resolution versions of all our figures are available on Siggraph 2005 DVD.

[3] `http://pixie.sourceforge.net`

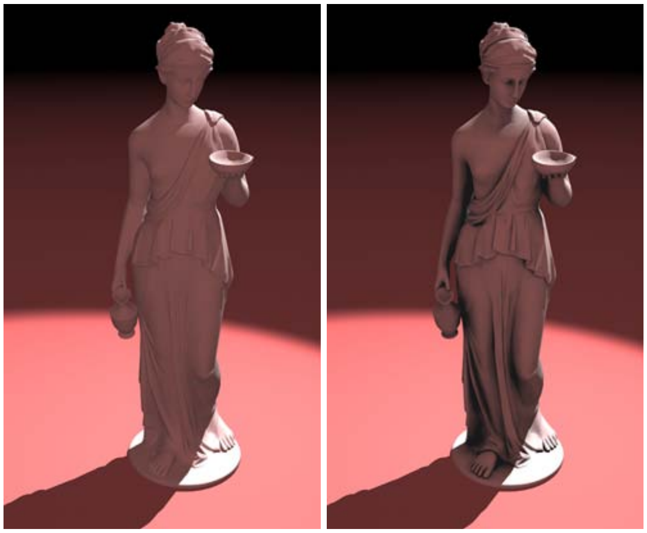


Figure 7: Our method can also be used without global illumination computations. On the left, the statue is illuminated with a spotlight from the top and a bounce light to simulate the light reflecting off the floor. The bounce light is a red point light source without a falloff placed underneath the floor. To compute the right image, we substituted the light contribution from the bounce light as $L_D^\star$ in equation 4. The computed I_N is then added on the diffuse color of the object computed using the main light and the bounce light.

shows a rendering of a statue on an infinite red plane. This scene is illuminated by a spotlight and a reddish bounce light, placed underneath the plane. The right image in Figure 7 has our correction term added. For every shading point, we use the illumination from the bounce light as $L_D^\star$ and compute the near-field correction term. We also set the radiosity of the triangles to zero. This essentially is used to approximate the portion of the power coming from the bounce light (which is itself not physically correct) that is occluded by nearby triangles. To compute this image, no rays were traced and the overhead of adding the nearby correction term was 20 seconds (less than 10% of the total rendering time). This technique can easily be used in a traditional pipeline where global illumination is often approximated using many physically incorrect light sources in order to enhance the surface relief and simulate occlusions due to nearby geometry.

10 Acknowledgements

We would like to thank Pixar and Wayne Wooden for providing us with their renderer and helping us with rendering. We also thank our reviewers for their constructive comments.

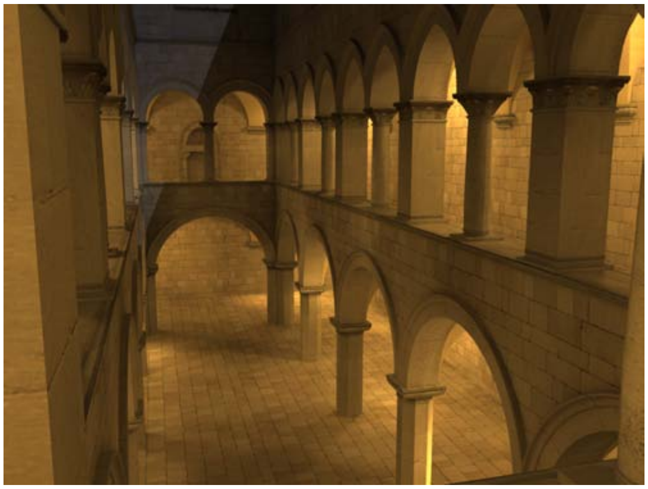


Figure 8: This figure demonstrates our results in a scene with many light sources. The scene is the same atrium model used in Figure 6. Rather than a single distant light source, this scene is illuminated with 16 yellow point lights placed on the ceilings and a single blue distant light. The overhead of adding the global illumination to this scene using our approximation was less than the time it took to render this scene using direct lighting only.

References

ATKINSON, K. E. 2000. The planar radiosity equation and its numerical solution. *IMA Journal of Numerical Analysis 20*, 303–332.

BASRI, R., AND JACOBS, D. 2000. Lambertian reflectance and linear subspaces. Tech. Rep. MCS00-21, 2000-172R, Waizmann Institute of Science, NEC Research Institude.

BEKAERT, P., DUTRE, P., AND WILLEMS, Y. D. 1998. Final radiosity gather step using a monte carlo technique with optimal importance sampling. Tech. Rep. CW275.

DUTRÉ, P., BEKAERT, P., AND BALA, K. 2003. *Advanced Global Illumination*. A. K. Peters Ltd.

EPSTEIN, R., HALLINAN, P. W., AND YUILLE, A. L. 1995. 5 ± 2 eigenimages suffice: An empirical investigation of low-dimensional lighting models. In *IEEE workshop on physics-based modeling in computer vision*, 108–116.

GAUTRON, P., KRIVANEK, J., PATTANAIK, S., AND BOUATOUCH, K. 2004. A novel hemispherical basis for accurate and efficient rendering. In *Rendering Techniques 2004: 15th Eurographics Workshop on Rendering*, 321–330.

GREGER, G., SHIRLEY, P., HUBBARD, P. M., AND GREENBERG, D. P. 1998. The irradiance volume. *IEEE Comput. Graph. Appl. 18*, 2, 32–43.

HOTTEL, H. C., AND SAFORIM, A. F. 1967. *Radiative Transfer*. McGraw Inc.

JENSEN, H. W. 2001. *Realistic Image Synthesis Using Photon Mapping*. A. K. Peters, Natick, MA.

KRIVANEK, J., GAUTRON, P., PATTANAIK, S., AND BOUATOUCH, K. 2005. Radiance caching for efficient global illumination computation. In *IEEE Transacations of Visualization and Comptuer Graphics*.

LISCHINSKI, D., TAMPIERI, F., AND GREENBERG, D. P. 1993. Combining hierarchical radiosity and discontinuity meshing. *Computer Graphics 27*, Annual Conference Series, 199–208.

RAMAMOORTHI, R., AND HANRAHAN, P. 2001. An efficient representation for irradiance environment maps. In *SIGGRAPH 2001, Computer Graphics Proceedings*, 497–500.

RAMAMOORTHI, R., AND HANRAHAN, P. 2001. The relationship between radiance and irradiance: Determining the illumination from images of a convex lambertian object. In *Journal of the Optical Society of America*.

RATHSFELD, A. 1999. Edge asymptotics for the radiosity equation over polyhedral boundaries. *Mathematical Methods in the Applied Sciences 22*, 3, 217–241.

RUSHMEIER, H. E., PATTERSON, C., AND VEERASAMY, A. 1993. Geometric simplification for indirect illumination calculations. In *Graphics Interface*.

RUSHMEIER, H. E. 1988. *Realistic Image Synthesis for Scenes with Radiatively Participating Media*. Ph.D. thesis.

SCHEEL, A., STAMMINGER, M., AND SEIDEL, H. 2001. Thrifty final gather for radiosity. In *Rendering Techniques 2001 (Proc. of Eurographics Workshop on Rendering 2001)*, Eurographics.

SCHEEL, A., STAMMINGER, M., AND SEIDEL, H. P. 2002. Grid based final gather for radiosity on complex scenes. In *EUROGRAPHICS 2002*, 547–555.

SILLION, F., AND PUECH, C. 1994. *Radiosity and Global Illumination*. Morgan Kaufmann, San Francisco, CA.

SLOAN, P. P., KAUTZ, J., AND SNYDER, J. 2002. Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. In *SIGGRAPH 2002, Computer Graphics Proceedings*, 527–536.

TABELLION, E., AND LAMORLETTE, A. 2004. An approximate global illumination system for computer generated films. In *SIGGRAPH 2004, Computer Graphics Proceedings*.

WARD, G. J., AND HECKBERT, P. 1992. Irradiance Gradients. In *Third Eurographics Workshop on Rendering*, 85–98.

WARD, G., RUBINSTEIN, F., AND CLEAR, R. 1988. A ray tracing solution for diffuse interreflectio. In *SIGGRAPH 1988, Computer Graphics Proceedings*.

ZIMMERMAN, K., AND SHIRLEY, P. 1995. A Two-Pass Realistic Image Synthesis Method for Complex Scenes. In *Rendering Techniques '95 (Proceedings of the Sixth Eurographics Workshop on Rendering)*, 284–295.

References

Annen, T., Kautz, J., Durand, F., and Seidel, H.-P. 2004. Spherical harmonic gradients for mid-range illumination. In *Rendering Techniques 2004, Eurographics Symposium on Rendering*, Eurographics Association, 331–336.

Arikan, O., Forsyth, D. A., and O'Brien, J. F. 2005. Fast and detailed approximate global illumination by irradiance decomposition. *ACM Trans. Graph. (Proceedings of ACM SIGGRAPH 2005) 24*, 3, 1108–1114.

Chelle, M., Andrieu, B., and Bouatouch, K. Nested radiosity for plant canopies. *Vis. Comput.*.

Durand, F., Holzschuch, N., Soler, C., Chan, E., and Sillion, F. X. 2005. A frequency analysis of light transport. *ACM Trans. Graph. (Proceedings of ACM SIGGRAPH 2005) 24*, 3, 1115–1126.

Fleming, R. W., Dror, R. O., and Adelson, E. H. 2003. Real-world illumination and the perception of surface reflectance properties. *Journal of Vision 3* (July), 347–368.

Foley, T., and Sugerman, J. 2005. KD-tree acceleration structures for a GPU raytracer. In *HWWS '05: Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware*, ACM Press, 15–22.

Gautron, P., Křivánek, J., Pattanaik, S. N., and Bouatouch, K. 2004. A novel hemispherical basis for accurate and efficient rendering. In *Rendering Techniques 2004, Eurographics Symposium on Rendering*, Eurographics Association, 321–330.

Gautron, P., Křivánek, J., Bouatouch, K., and Pattanaik, S. N. 2005. Radiance cache splatting: A GPU-friendly global illumination algorithm. In *Rendering Techniques 2005, Eurographics Symposium on Rendering*, Eurographics Association, 55–64.

Gautron, P., Bouatouch, K., and Pattanaik, S. 2007. Temporal radiance caching. *IEEE Transactions on Visualization and Computer Graphics (TVCG) 13*, 5 (September/October).

Gautron, P. 2006. *Cache de luminance et cartes graphiques : une approche pour la simulation d'clairage temps rel dans des scènes animes (Radiance caching and graphics hardware: and approach for real-time global illumnation in animated scenes.)*. PhD thesis, Université de Rennes 1.

Green, R. 2003. Spherical harmonic lighting: The gritty details. In *Game Developpers' Conference*.

Kajiya, J. T. 1986. The rendering equation. In *Proceedings of ACM SIGGRAPH'86*, ACM Press, 143–150.

Kautz, J., Sloan, P.-P., and Snyder, J. 2002. Fast, arbitrary BRDF shading for low-frequency lighting using spherical harmonics. In *Proceedings of the 13th Eurographics Workshop on Rendering*, Eurographics Association, 291–296.

Křivánek, J., Gautron, P., Bouatouch, K., and Pattanaik, S. 2005. Improved radiance gradient computation. In *SCCG '05: Proceedings of the 21st spring conference on Computer graphics*, ACM Press, 155–159.

Křivánek, J., Gautron, P., Pattanaik, S., and Bouatouch, K. 2005. Radiance caching for efficient global illumination computation. *IEEE Transactions on Visualization and Computer Graphics (TVCG) 11*, 5 (September/October).

Křivánek, J., Konttinen, J., Bouatouch, K., Pattanaik, S., and Žára, J. 2005. Fast approximation to spherical harmonic rotation. In *SCCG '06: Proceedings of the 22nd spring conference on Computer graphics*.

Křivánek, J., Bouatouch, K., Pattanaik, S. N., and Žára, J. 2006. Making radiance and irradiance caching practical: Adaptive caching and neighbor clamping. In *Rendering Techniques 2006, Eurographics Symposium on Rendering*.

Křivánek, J. 2005. *Radiance Caching for Global Illumination Computation on Glossy Surfaces*. PhD thesis, Université de Rennes 1 and Czech Technical University.

Larsen, B. D., and Christensen, N. 2004. Simulating photon mapping for real-time applications. In *Rendering Techniques 2004, Eurographics Symposium on Rendering*.

Larson, G. W., and Shakespeare, R. 1998. *Rendering with Radiance, The Art and Science of Lighting Visualization*. Morgan Kaufmann Publishers.

Ngan, A., Durand, F., and Matusik, W. 2005. Experimental analysis of brdf models. In *Rendering Techniques 2005, Eurographics Symposium on Rendering*, Eurographics Association, 117–226.

Purcell, T. J., Buck, I., Mark, W. R., and Hanrahan, P. 2002. Ray tracing on programmable graphics hardware. In *Proceedings of ACM SIGGRAPH '02*, ACM Press, 703–712.

Purcell, T. J., Donner, C., Cammarano, M., Jensen, H. W., and Hanrahan, P. 2003. Photon mapping on programmable graphics hardware. In *HWWS '03: Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware*, Eurographics Association, 41–50.

Ramamoorthi, R., and Hanrahan, P. 2001. An efficient representation for irradiance environment maps. In *Proceedings of ACM SIGGRAPH 2001*, ACM Press, 497–500.

Ramamoorthi, R., and Hanrahan, P. 2002. Frequency space environment map rendering. In *Proceedings of ACM SIGGRAPH 2002*, ACM Press, New York, NY, USA, 517–526.

Ramamoorthi, R., Mahajan, D., and Belhumeur, P. 2007. A first-order analysis of lighting, shading, and shadows. *ACM Trans. Graph. 26*, 1, 2.

Sloan, P.-P., Kautz, J., and Snyder, J. 2002. Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. In *Proceedings of ACM SIGGRAPH 2002*, ACM Press, 527–536.

Smyk, M., ichi Kinuwaki, S., Durikovic, R., and Myszkowski, K. 2005. Temporally coherent irradiance caching for high quality animation rendering. *Computer Graphics Forum (Proceedings of EUROGRAPHICS '05) 24*, 3.

Tabellion, E., and Lamorlette, A. 2004. An approximate global illumination system for computer generated films. *ACM Trans. Graph. (Proceedings of ACM SIGGRAPH 2004) 23*, 3, 469–476.

WALTER, B., DRETTAKIS, G., AND PARKER, S. 1999. Interactive rendering using render cache. In *Proceedings of the 13th Eurographics Workshop on Rendering*, 19–30.

WARD, G. J., AND HECKBERT, P. S. 1992. Irradiance gradients. In *Proceedings of the Third Eurographics Workshop on Rendering*, 85–98.

WARD, G. J., RUBINSTEIN, F. M., AND CLEAR, R. D. 1988. A ray tracing solution for diffuse interreflection. In *Proceedings of ACM SIGGRAPH '88*, ACM Press, 85–92.

WARD, G. J. 1994. The Radiance lighting simulation and rendering system. In *Proceedings of ACM SIGGRAPH '94*, ACM Press, 459–472.

WILLIAMS, L. 1978. Casting curved shadows on curved surfaces. In *Proceedings of ACM SIGGRAPH '78*, ACM Press, 270–274.