Advanced 3D graphics for movies and games (NPGR010)

– Approximate global illumination computation

Jiří Vorba, MFF UK/Weta Digital

jirka@cgg.mff.cuni.cz

Slides by prof. Jaroslav Křivánek

Review

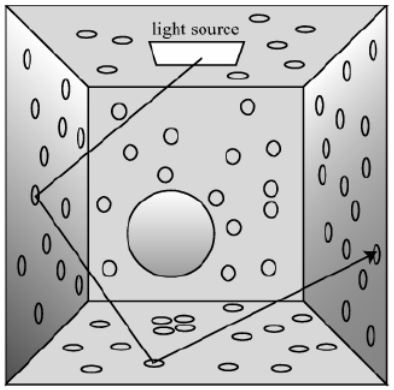
Photon mapping – SDS paths

© H.W.Jensen

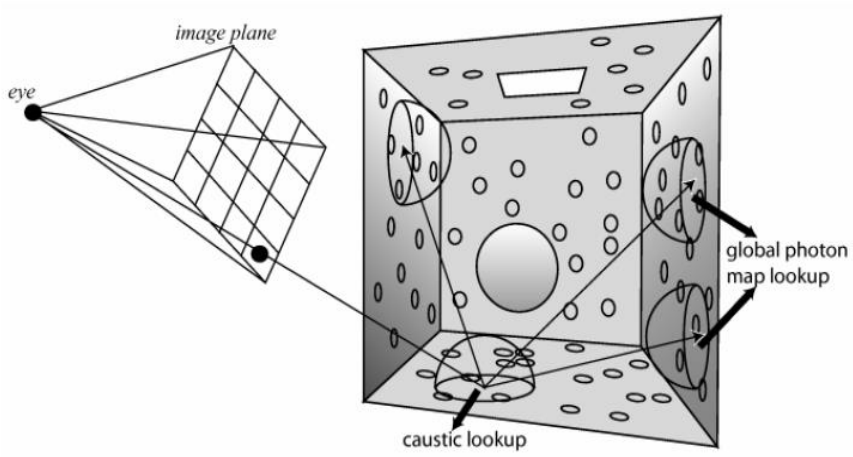
© Wojciech Jarosz

Photon mapping – Steps

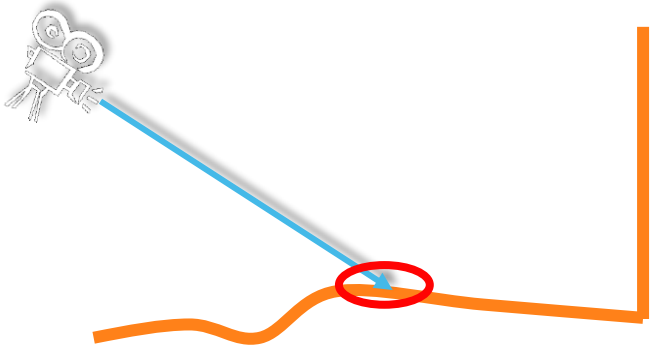
1. **Photon tracing**

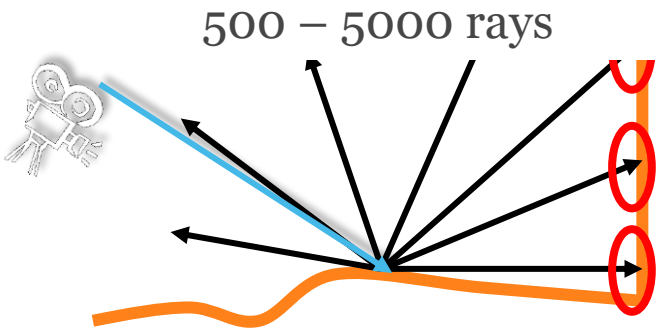


2. **Rendering with photon maps**



Final gathering?





information in the global photon map too inaccurate

inaccuracy in the global maps gets “averaged out”

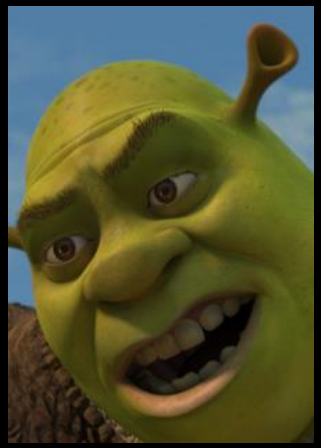
Progressive photon mapping

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

Approximate GI methods

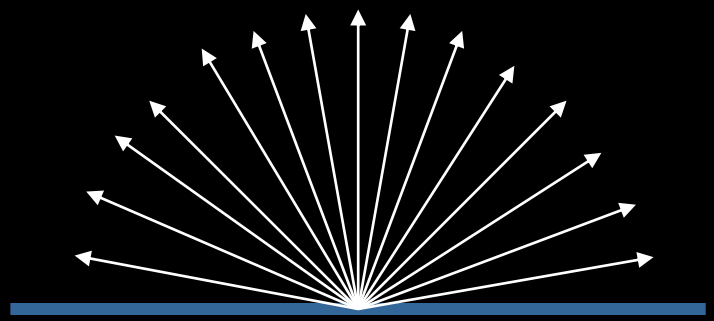
Irradiance caching

Jaroslav Křivánek
Charles University, Prague
Jaroslav.Krivanek@mff.cuni.cz



Motivation

- **Distribution path tracing** (DPT)
  Final gathering (FG)
  - Estimate illumination integral at a point by tracing many rays (500-5000)
  - Costly computation



- Irradiance caching accelerates DPT/FG for diffuse indirect illumination

Motivation

- Spatial coherence
  - Diffuse indirect illumination changes slowly over surfaces

Indirect irradiance – changes slowly

Irradiance caching

- Sparse locations for full DRT computation
- Resulting irradiance stored in a cache
- Most pixels interpolated from cached records

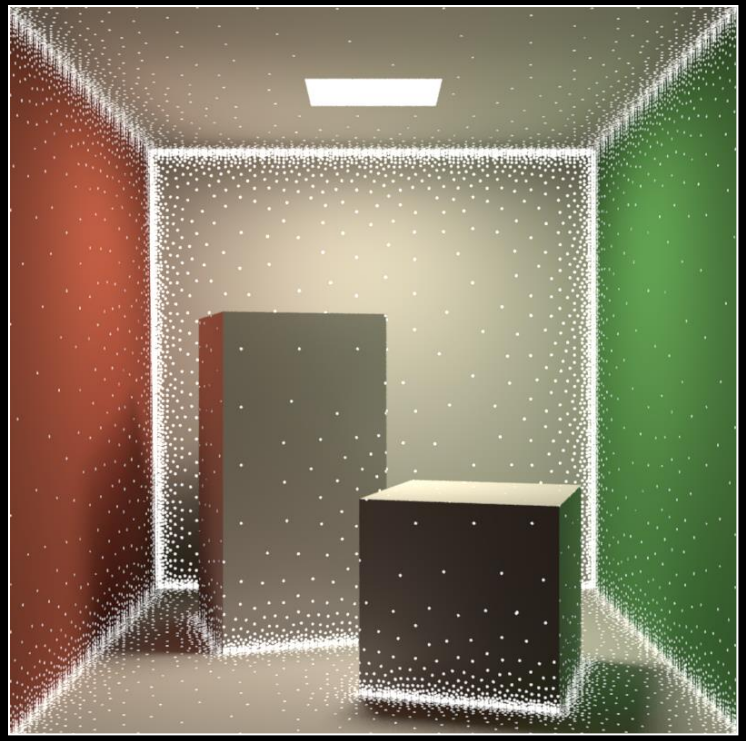


Image credit: Okan Arikan

Irradiance caching

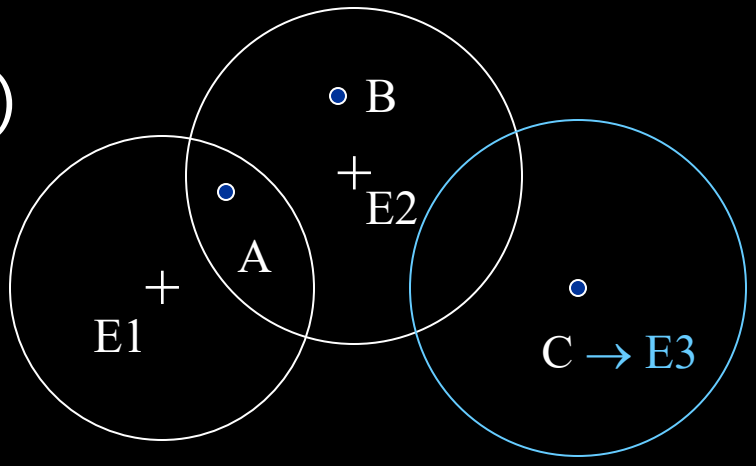
- Faster computation of the *diffuse component* of indirect illumination
- Diffuse reflection

$$L_o(\mathbf{p}) = E(\mathbf{p}) * \rho_d(\mathbf{p}) / \pi$$

- View-independence
  - Outgoing radiance independent of view direction
  - Total irradiance is all we need => cache irradiance

Irradiance caching

- Lazy evaluation of new irradiance values
  - Only if cannot be interpolated from existing ones
- Example: Values E1 and E2 already stored
  - Interpolate at A (fast)
  - Extrapolate at B (fast)
  - Add new record at C (slow)

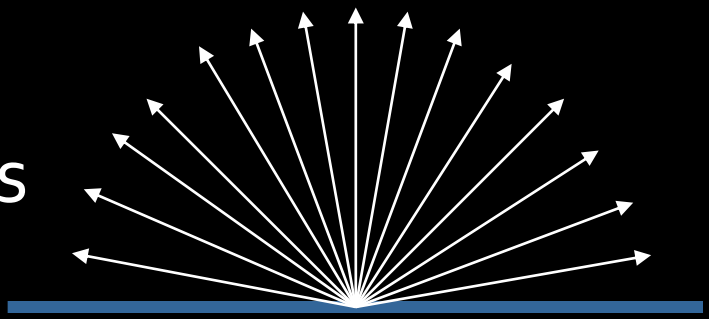


Irradiance caching pseudocode

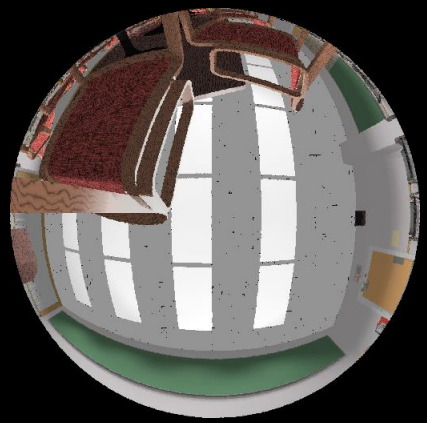
```
GetIrradiance(p):
   Color E = InterpolateFromCache(p);
   if( E == invalid )
      E = SampleHemisphere(p);
      InsertIntoCache(E, p);
   return E;
```

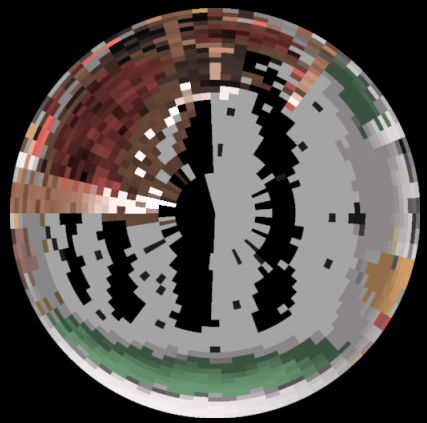
Indirect irradiance calculation

```
E = SampleHemisphere(p);
```



- Cast 500-5000 secondary rays (user-specified)
- Compute illumination at intersection
  - Direct illumination only, or
  - Path tracing, or
  - Photon map radiance estimate, or
  - Query in (another) irradiance cache



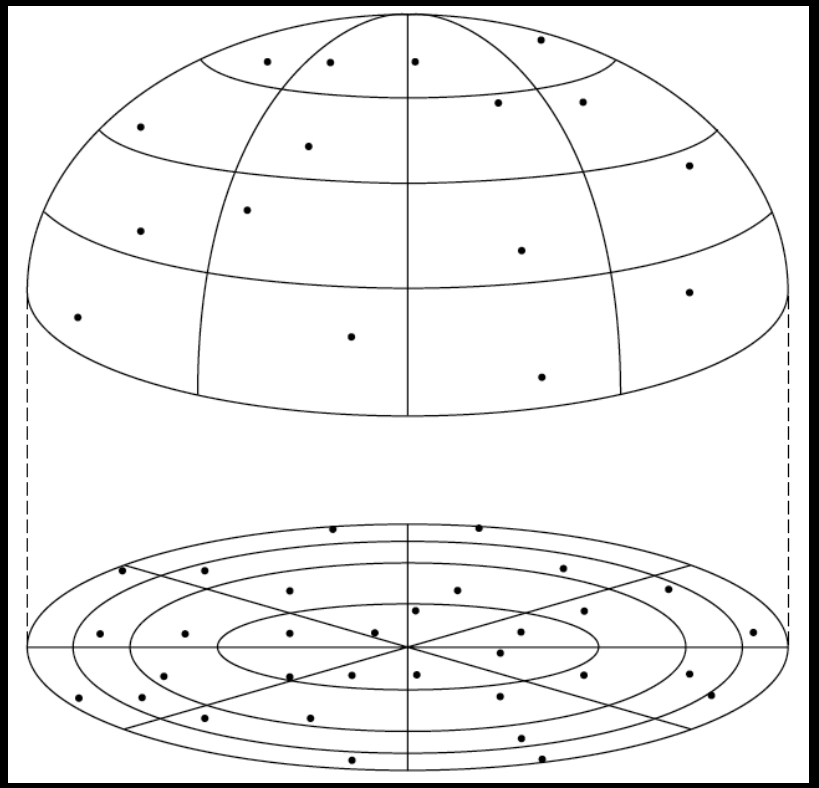


  - No emission taken into account!

Indirect irradiance calculation

```
E = SampleHemisphere(p);
```

- *Stratified Monte Carlo* hemisphere sampling
  - Subdivide hemisphere into cells
  - Choose a random direction in each cell and trace ray



Indirect irradiance calculation

```
E = SampleHemisphere(p);
```

- Estimating irradiance at **p**:

$$E(\mathbf{p}) = \int L_i(\mathbf{p}, \omega_i) \cos\theta_i \, \mathrm{d}\omega_i$$

- General form of the stratified estimator

$$E(\mathbf{p}) \approx \frac{1}{MN} \sum_{j=0}^{M-1} \sum_{k=0}^{N-1} \frac{f(\theta_{j,k}, \phi_{j,k})}{p(\theta_{j,k}, \phi_{j,k})}$$

Indirect irradiance calculation

```
E = SampleHemisphere(p);
```

- For irradiance calculation, the integrand is:

$$L(\theta, \phi) \cos\theta$$

- PDF:

$$p(\theta, \phi) = \frac{\cos\theta}{\pi}$$

Indirect irradiance calculation

- Irradiance estimator for IC:

$$E(\mathbf{p}) \approx \frac{\pi}{MN} \sum_{j=0}^{M-1} \sum_{k=0}^{N-1} L_{j,k}$$

- $L_{i.k}$ … radiance sample from direction:

$$(\theta_{j,k}, \phi_{j,k}) = \left(\arccos \sqrt{1 - \frac{j + \zeta_{j,k}^1}{M}}, 2\pi \frac{k + \zeta_{j,k}^2}{N} \right)$$

- M, N … number of divisions along θ and ϕ
- $\zeta_{j,k}^1, \zeta_{j,k}^2$ … random numbers from R(0,1)

Irradiance caching pseudocode

```
GetIrradiance(p):
    Color E = InterpolateFromCache(p);
    if( E == invalid )
        E = SampleHemisphere(p);
        InsertIntoCache(E, p);
    return E;
```

Record spacing

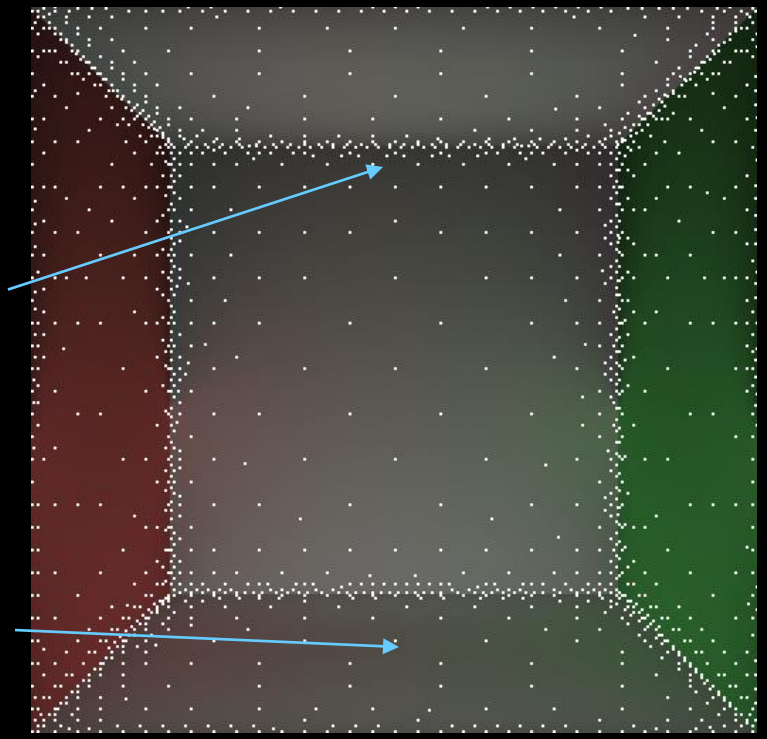
- If $E(\mathbf{p})$ changes slowly => interpolate more
- If $E(\mathbf{p})$ changes quickly => interpolate less

- What is the upper bound on rate of change (i.e. gradient) of irradiance?
- Answer from the "worst case" analysis (omitted)

Record spacing

- Near geometry
  → dense spacing
  - Geometry = source of indirect illumination

- Open spaces
  → sparse sampling



Record spacing

Irradiance interpolation

```
E = InterpolateFromCache(p)
```

- Weighted average: $$E(\mathbf{p}) = \frac{\sum_{i \in S(\mathbf{p})} E_i(\mathbf{p}) w_i(\mathbf{p})}{\sum_{i \in S(\mathbf{p})} w_i(\mathbf{p})},$$

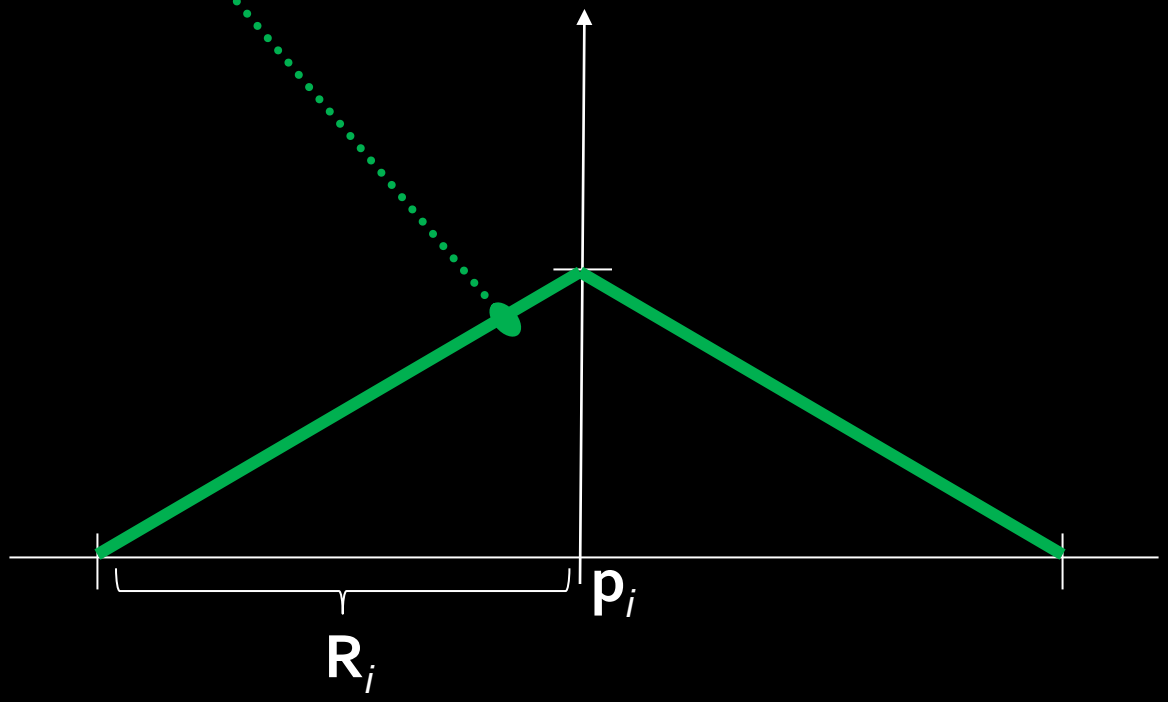
- Records used for interpolation:

$$S(\mathbf{p}) = \{i;\ w_i(\mathbf{p}) > 0\}$$

Weighting function

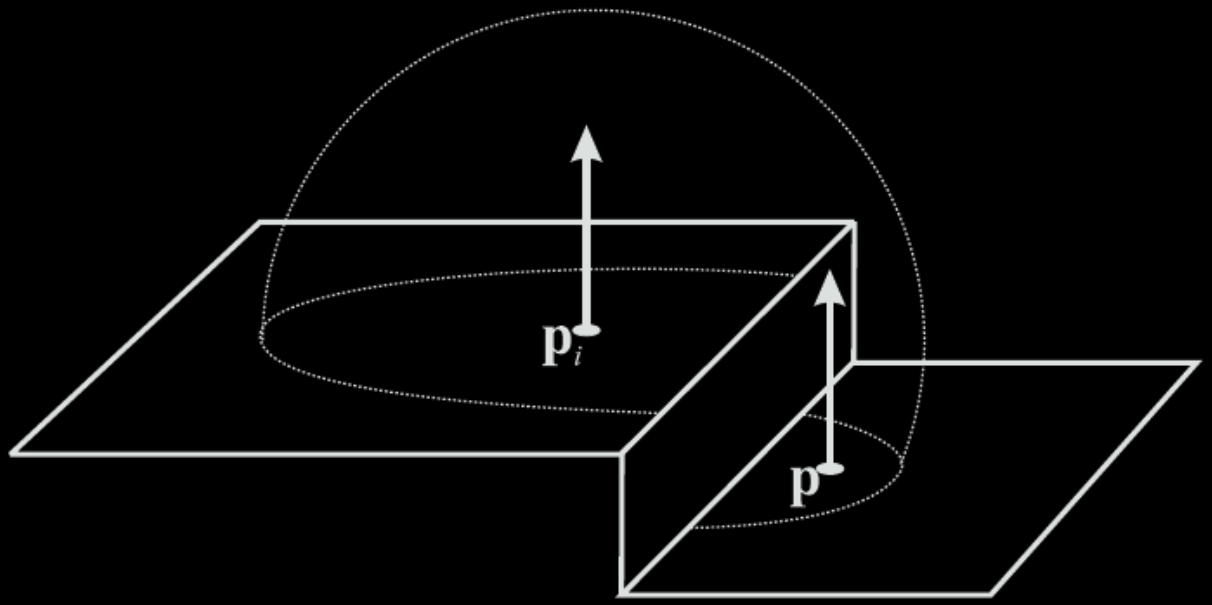
[Tablellion and Lamorlette 04]

$$w_i(\mathbf{p}) = 1 - \kappa \max\left\{ \frac{\|\mathbf{p} - \mathbf{p}_i\|}{\mathrm{clamp}(2R_i, R_{\min}, R_{\max})}, \quad \frac{\sqrt{1 - \mathbf{n} \cdot \mathbf{n}_i}}{\sqrt{1 - \cos 10^\circ}} \right\}$$



Heuristic "behind" test

- Record at $\mathbf{p}_i$ rejected from interpolation at $\mathbf{p}$ if $\mathbf{p}$ is "behind" $\mathbf{p}_i$



Irradiance caching pseudocode

```
GetIrradiance(p):
    Color E = InterpolateFromCache(p);
    if( E == invalid )
        E = SampleHemisphere(p);
        InsertIntoCache(E, p);
    return E;
```

Irradiance cache record

```
InsertIntoCache(E, p);
```

• Vector3 position	Position in space
• Vector3 normal	Normal at `position'
• float R	Validity radius
• Color E	Stored irradiance
• Color dEdP[3]	Gradient w.r.t. translation
• Color dEdN[3]	Gradient w.r.t. rotation

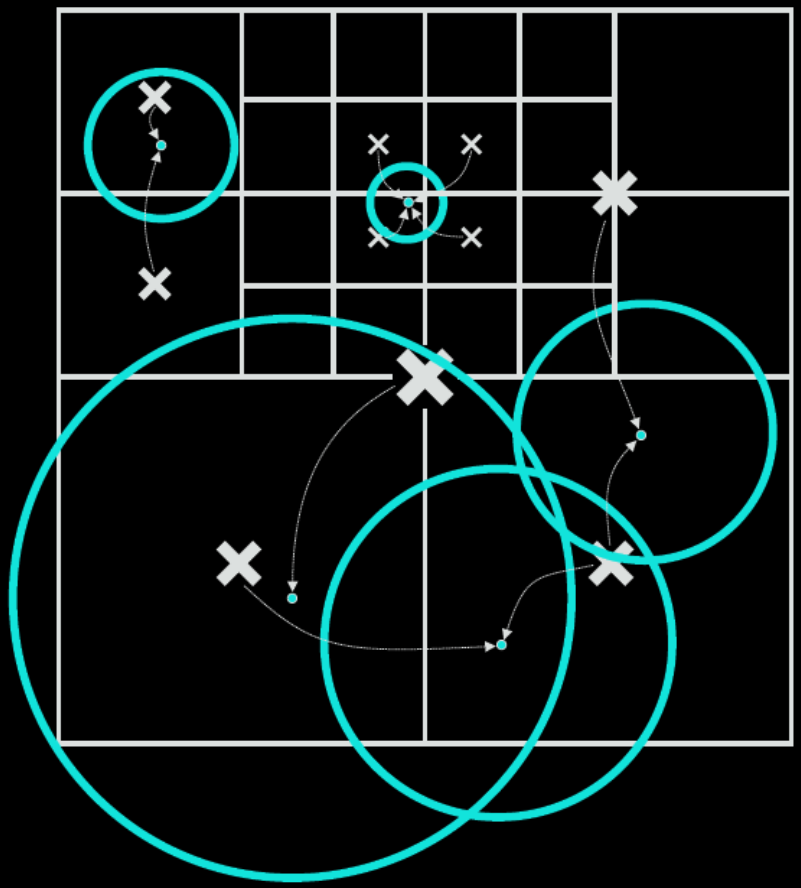
Irradiance cache data structure

```
InsertIntoCache(E, p);
```

- Requirements
  - Fast incremental updates (records stored on the fly)
  - Fast query for all records (spheres) overlapping a given point **p**

Data structure: Octree

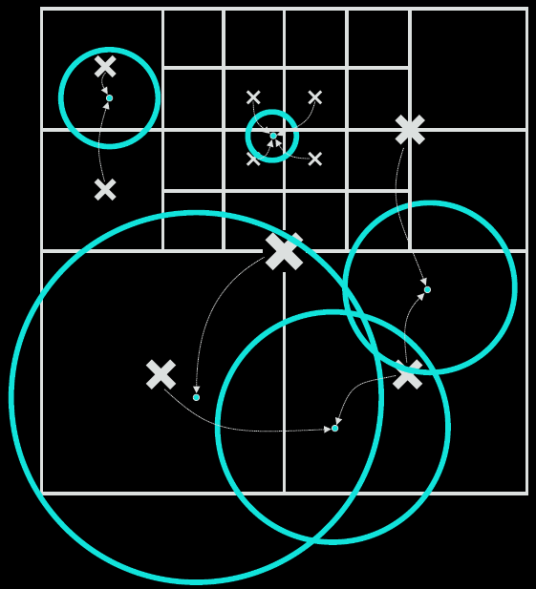
```
InsertIntoCache(E, p);
```



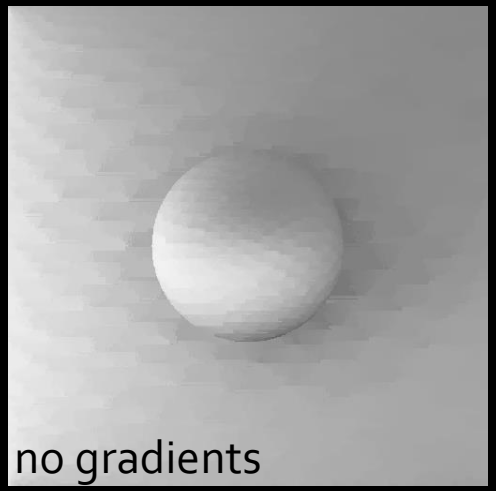
Data structure: Octree

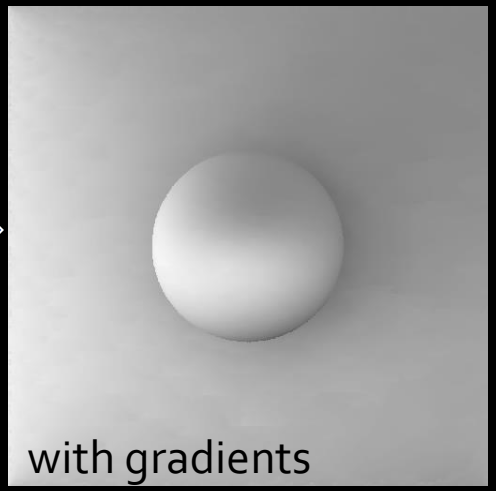
back to … `E = InterpolateFromCache(p)`

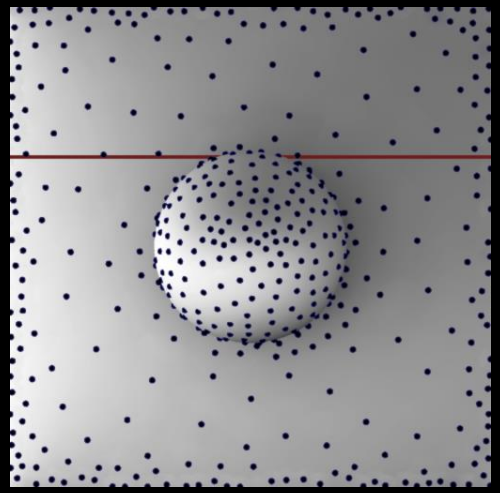
```
procedure LookUpRecordsMR(p, n)
    node ← root
    while node ≠ NULL do
        for all records i stored in node do
            if (w_i(p) > 0) and (p_i not in front of p) then
                Include record in S(p).
            end if
        end for
        node ← child containing p
    end while
end procedure
```

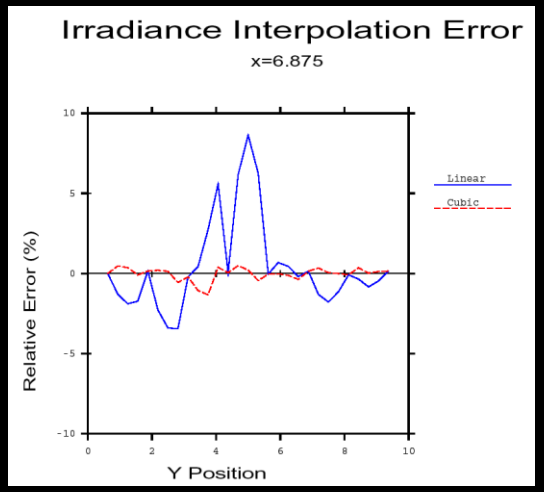


Irradiance gradients









Irradiance gradients

- Essential for smooth interpolation
- Calculated during hemisphere sampling
  - i.e. no extra rays, little overhead
- Stored as a part of the record in the cache
- Used in interpolation

Rotation gradient

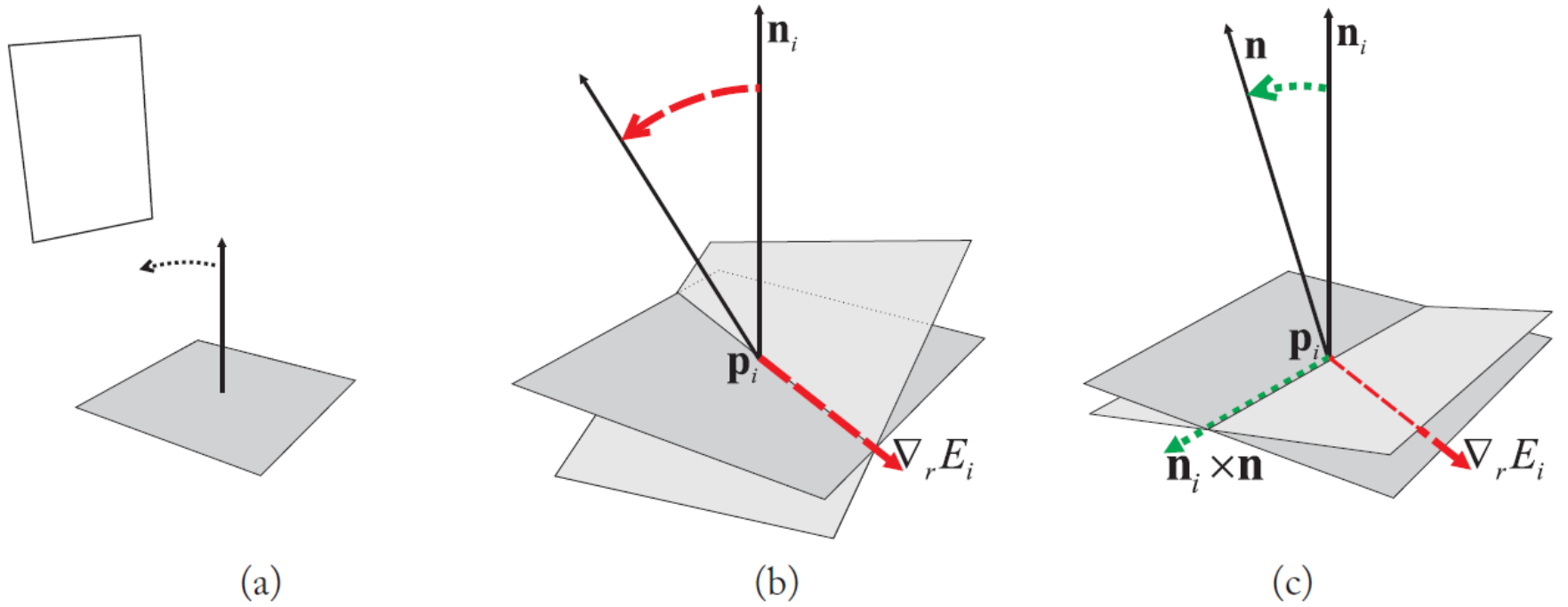
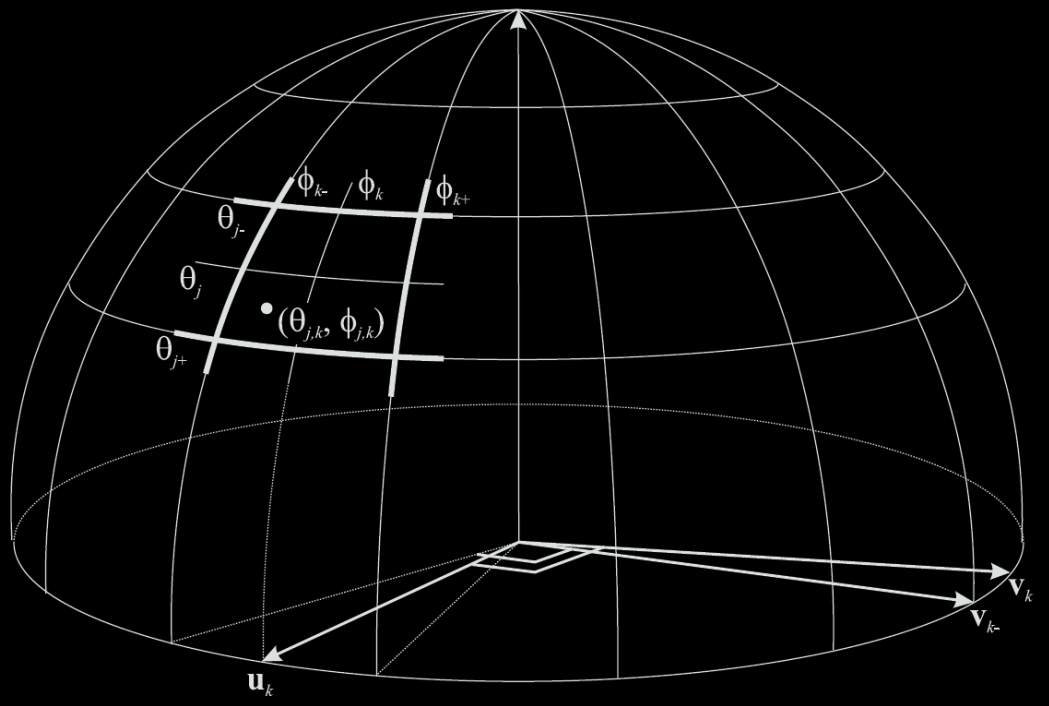


Figure 2.4: (a) As the surface element is rotated towards the bright surface, irradiance increases. (b) The rotation gradient $\nabla_r E_i$ of cache record i gives the axis of rotation that produces maximum increase in irradiance. The gradient magnitude is the irradiance derivative with rotation around that axis. (c) When the surface element is rotated around any arbitrary axis (in our example determined by the change in surface normal as $\mathbf{n}_i \times \mathbf{n}$) the irradiance derivative is given by the dot product of the axis of rotation and the rotation gradient: $(\mathbf{n}_i \times \mathbf{n}) \cdot \nabla_r E_i$.

Rotation gradient formula

$$\nabla_r E \approx \frac{\pi}{MN} \sum_{k=0}^{N-1} \left(\mathbf{v}_k \sum_{j=0}^{M-1} -\tan\theta_j \cdot L_{j,k} \right)$$



Translation gradient

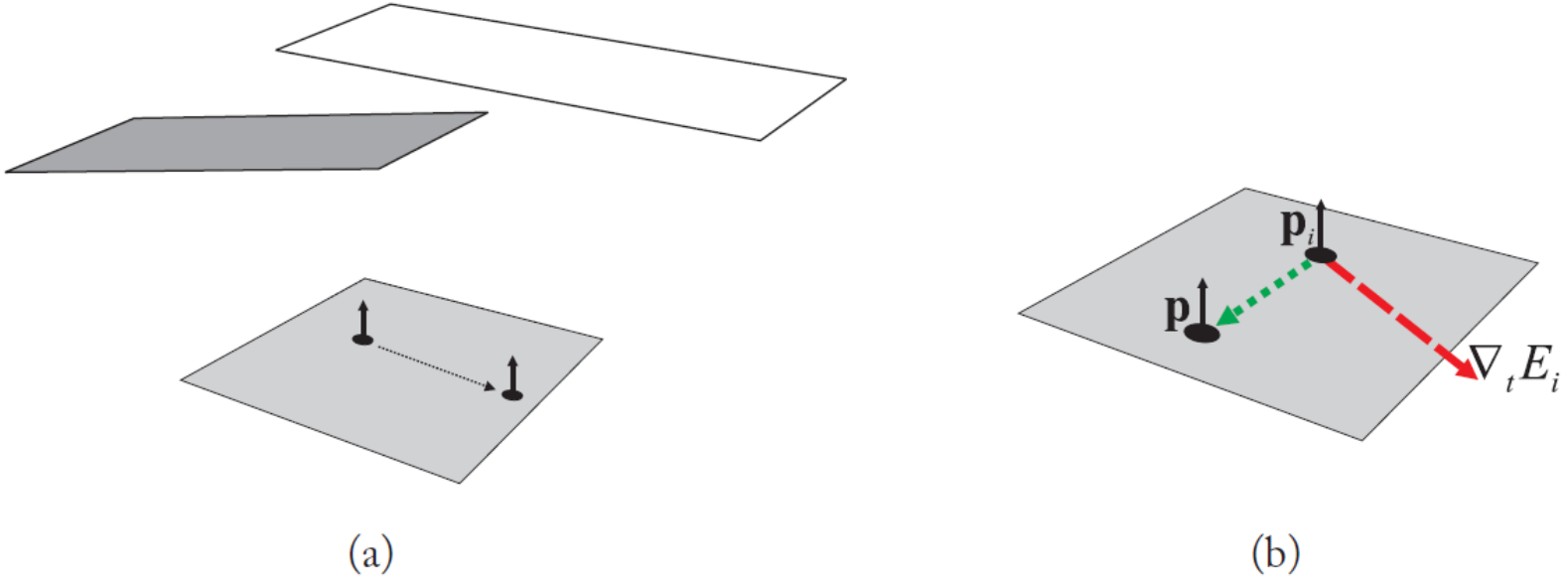
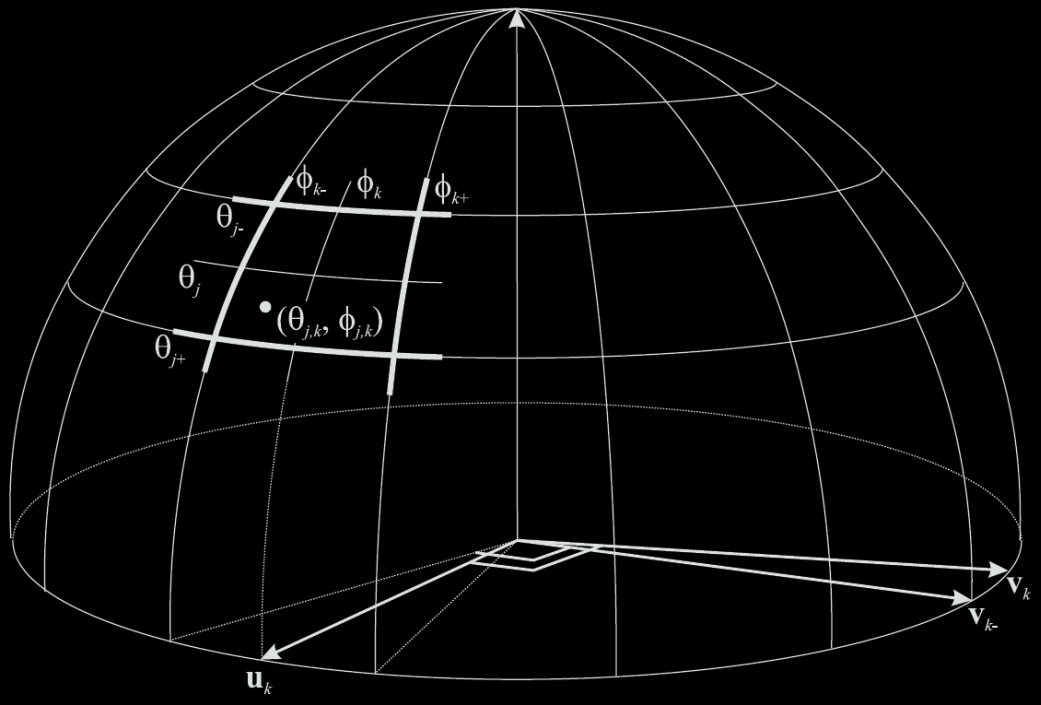


Figure 2.6: (a) As the surface element is translated, it becomes more exposed to the bright surface, and irradiance increases. (b) The translation gradient $\nabla_t E_i$ of record i gives the direction of translation that produces the maximum increase in irradiance. The gradient magnitude is the irradiance derivative with respect to translation along that direction. When a surface element is translated along any arbitrary direction, a first-order approximation of the change in irradiance is given by the dot product of the translation vector and the translation gradient: $(\mathbf{p} - \mathbf{p}_i) \cdot \nabla_t E_i$.

Translation gradient formula

$$\nabla_t E \approx \sum_{k=0}^{N-1} \Bigg[\mathbf{u}_k \frac{2\pi}{N} \sum_{j=1}^{M-1} \frac{\cos^2\theta_{j-} \sin\theta_{j-}}{\min\{r_{j,k}, r_{j-1,k}\}} (L_{j,k} - L_{j-1,k}) + \mathbf{v}_{k-} \sum_{j=0}^{M-1} \frac{\cos\theta_j (\cos\theta_{j-} - \cos\theta_{j+})}{\sin\theta_{j,k} \min\{r_{j,k}, r_{j,k-1}\}} (L_{j,k} - L_{j,k-1}) \Bigg]$$



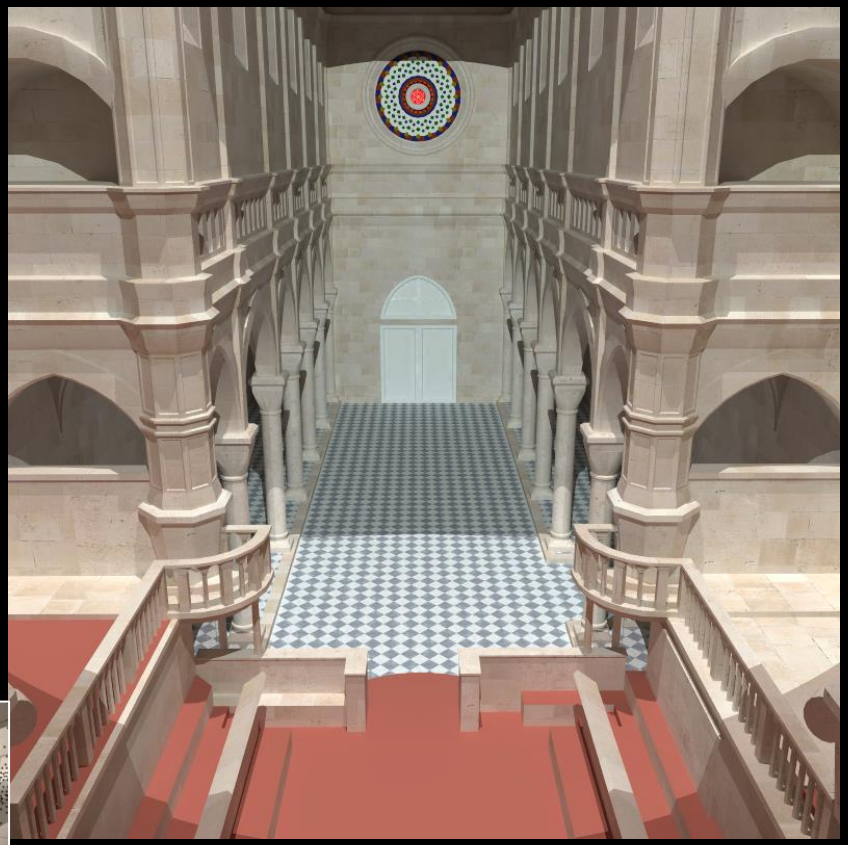
Irradiance interpolation w/ grads

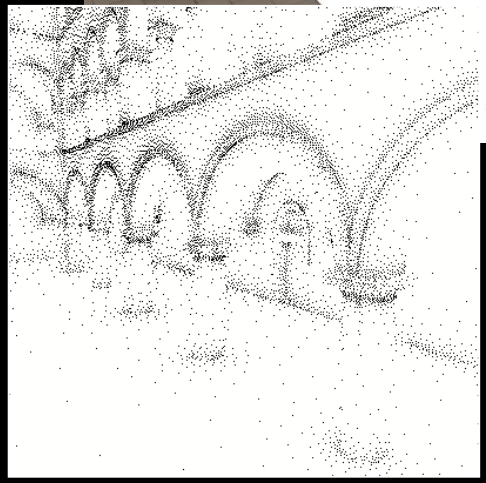
```
E = InterpolateFromCache(p)
```

- Weighted average: $$E(\mathbf{p}) = \frac{\sum_{i \in S(\mathbf{p})} E_i(\mathbf{p}) w_i(\mathbf{p})}{\sum_{i \in S(\mathbf{p})} w_i(\mathbf{p})},$$

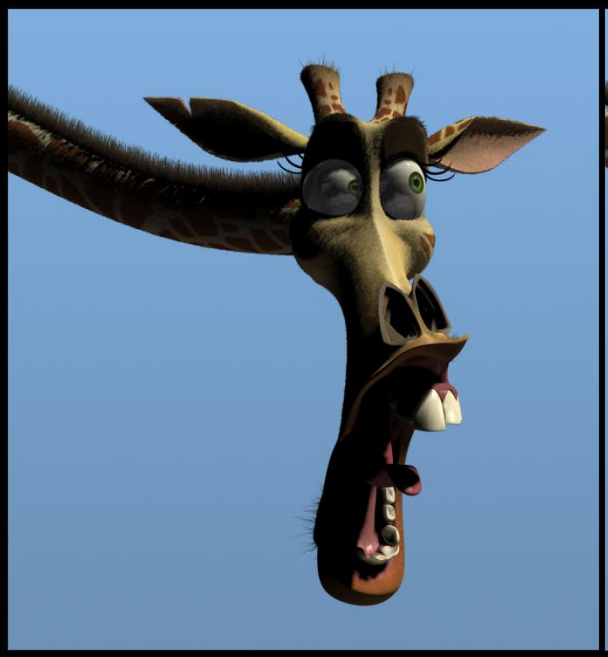
$$E_i(\mathbf{p}) = E_i + (\mathbf{n}_i \times \mathbf{n}) \cdot \nabla_r E_i + (\mathbf{p} - \mathbf{p}_i) \cdot \nabla_t E_i$$

Irradiance caching examples





Irradiance caching examples



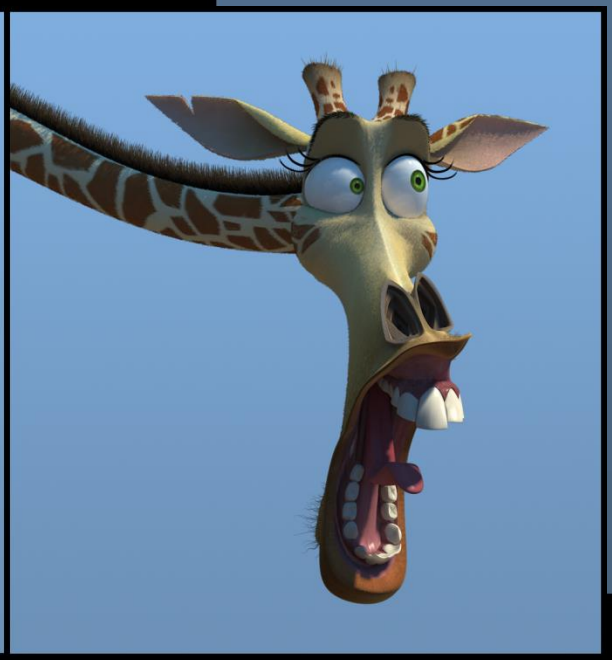


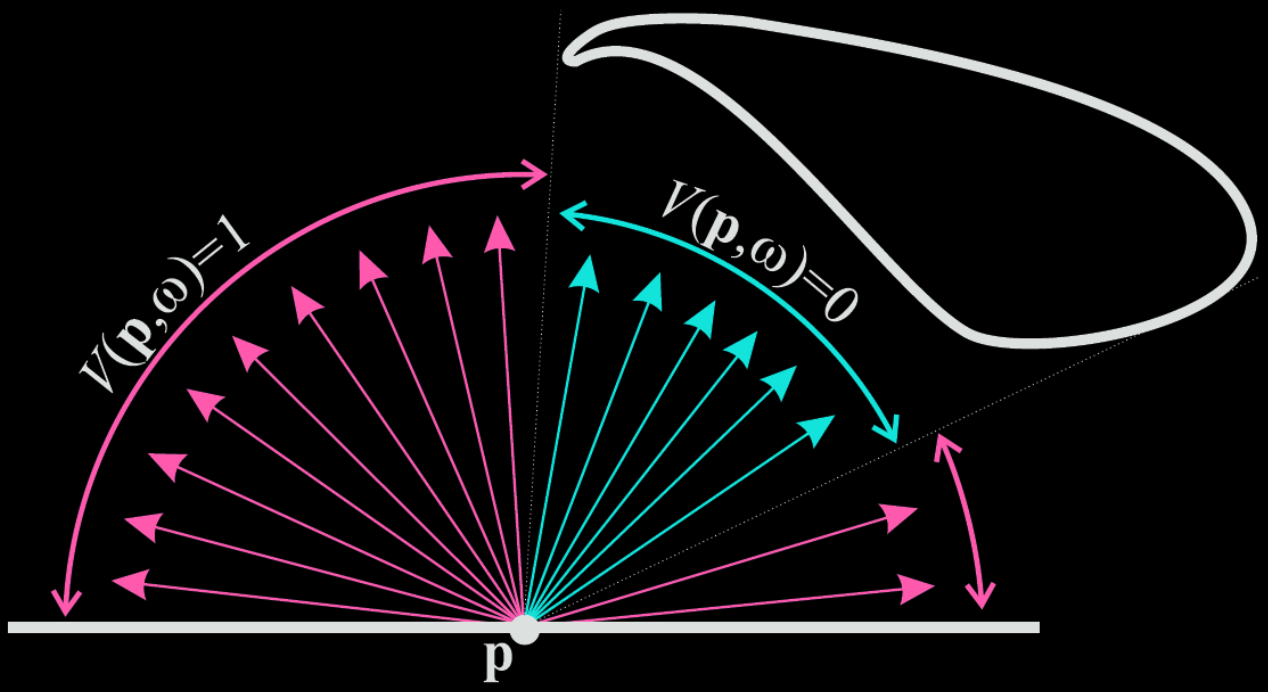
Image credit: Eric Tabellion, PDI DreamWorks

Irradiance caching examples

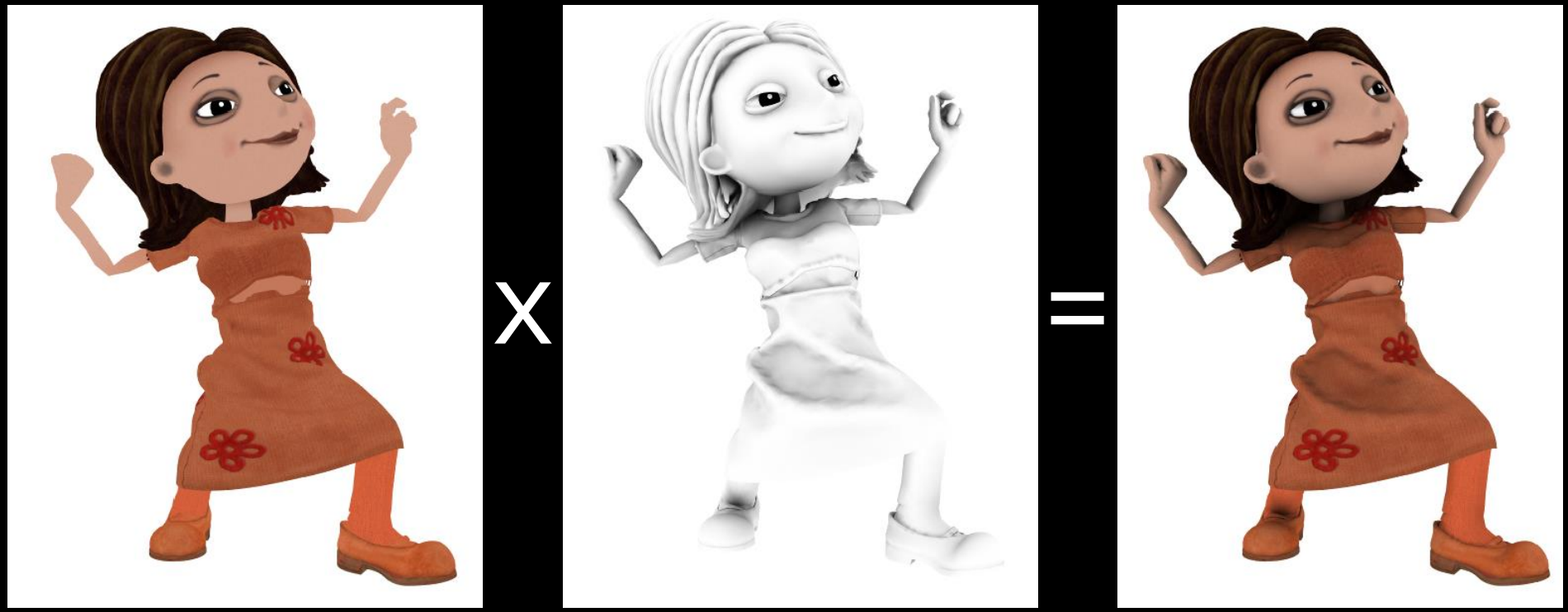
Image credit: Eric Tabellion, PDI DreamWorks

Ambient occlusion

$$A(\mathbf{p}) = \frac{1}{\pi} \int_{H^+} V(\mathbf{p}, \omega) \cos\theta \, \mathrm{d}\omega$$



Ambient occlusion



Ambient occlusion caching

Conclusion

- Fast indirect illumination of diffuse surfaces
  - Sparse sampling & fast interpolation
- Biased
- Not consistent
- Tons of implementation details that I did not discuss here

Further reading

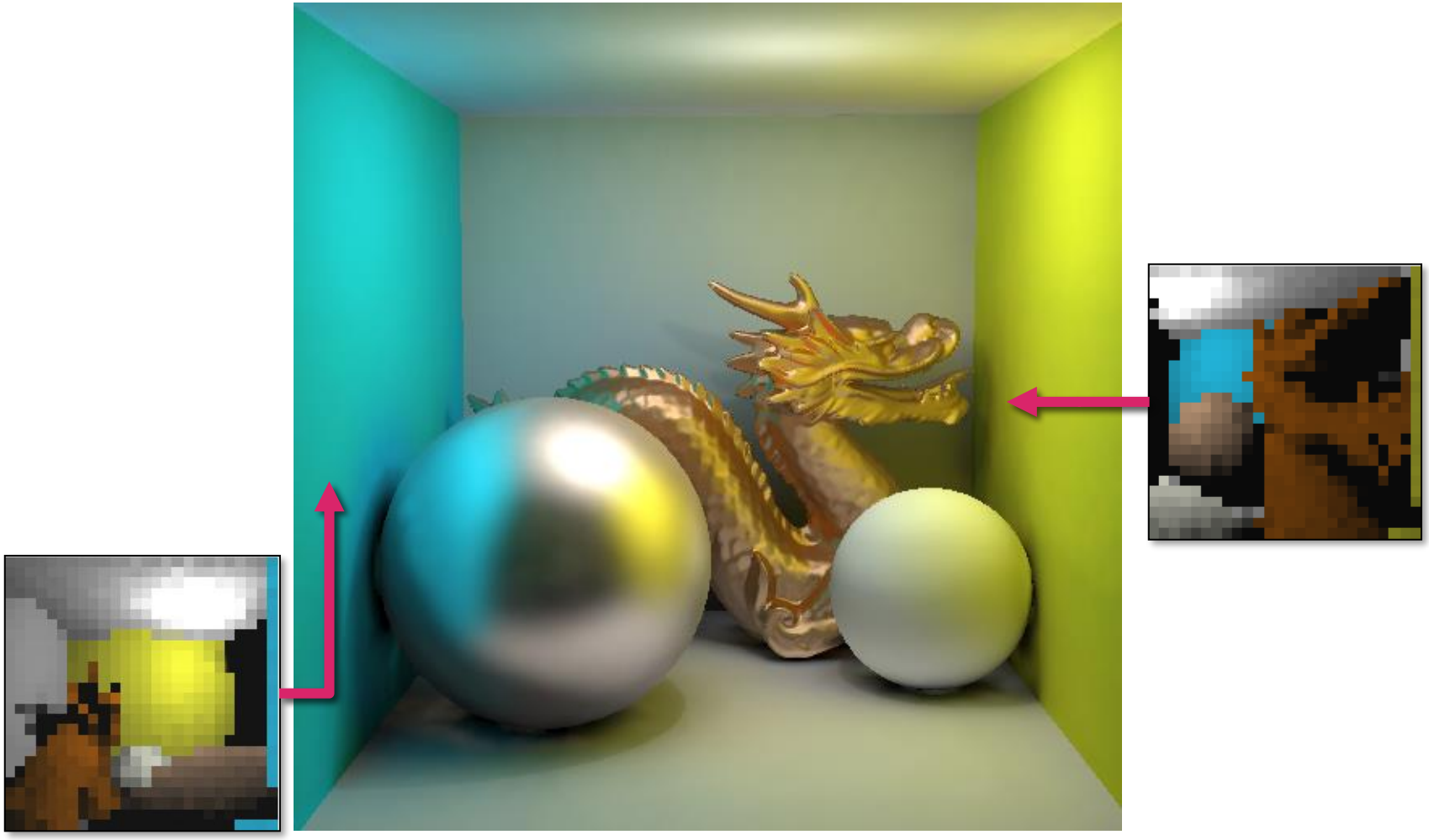
- Practical Global Illumination with Irradiance Caching
  - SIGGRAPH Course: 2008, Křivánek et al.
  - Book, 2009, Křivánek & Gautron
  - Both give references to further resources

Point-based Global Illumination

Point-based global illumination

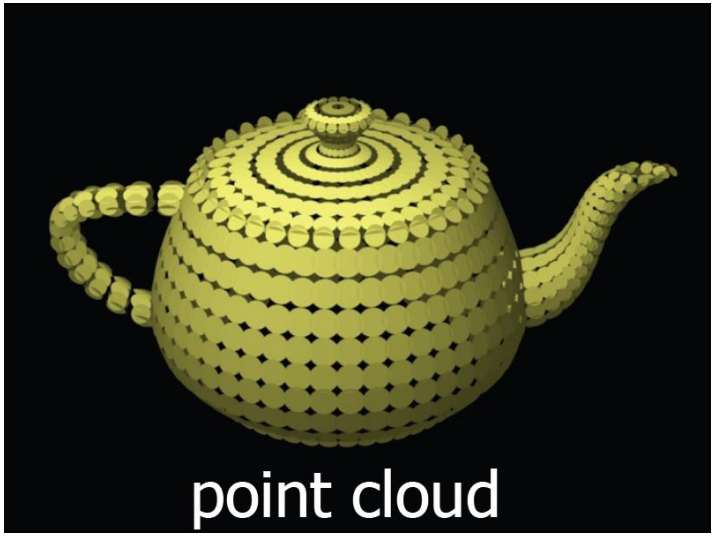
- **Original idea**
  - M. Bunnell, “Dynamic ambient occlusion and indirect lighting”, GPU Gems 2
- **Application in movie production**
  - P. Christensen, “Point-based approximate color bleeding”, Pixar tech memo #08-01
- **Real-time implementation** (CUDA)
  - T. Ritschel et al, “Micro-rendering for scalable, parallel final gathering”, SIGGRAPH Asia 2009

Point-based global illumination



Slide credit: Tobias Ritschel

Point-based global illumination



Slide credit: Per Christensen

References

- Křivánek et al.: **Global Illumination Across Industries**, SIGGRAPH 2010 course. http://cgg.mff.cuni.cz/~jaroslav/gicourse2010/
  - *Point-based Global Illumination for Film Production* (Per Christensen, PIXAR)
  - *Ray Tracing vs. Point-based GI for Animated Films* (Eric Tabellion, PDI Dreamworks)

- Ritschel et al. **Microrendering for Scalable, Parallel Final Gathering**, SIGGRAPH Asia 2009. http://www.mpi-nf.mpg.de/~ritschel/Microrendering/

References

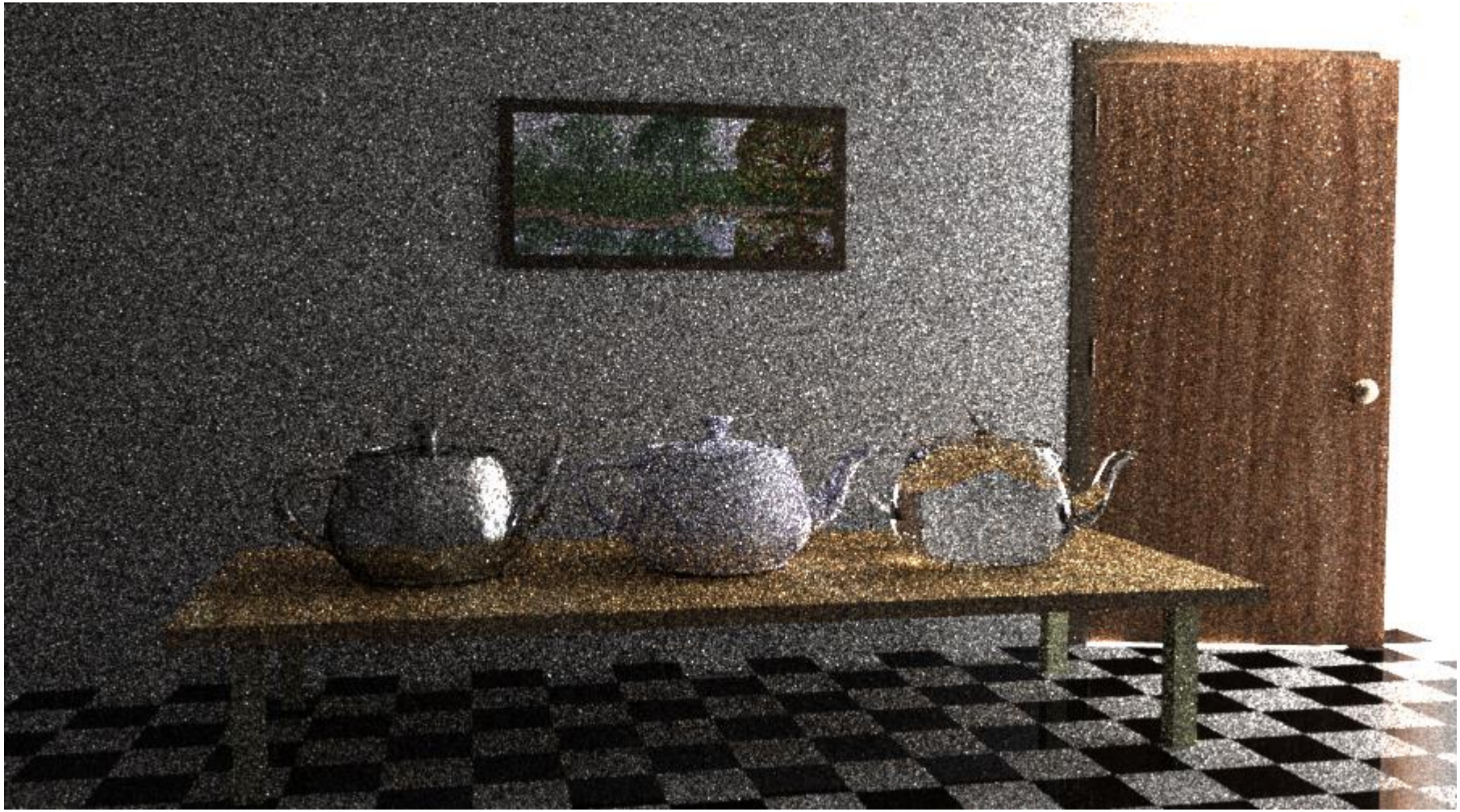
- Křivánek et al.: **Global Illumination Across Industries**, SIGGRAPH 2010 course. http://cgg.mff.cuni.cz/~jaroslav/gicourse2010/
  - *Ray Tracing Solution in Film Production Rendering* (Marcos Fajardo, SolidAngle)

What did we not cover?

In fact, many things…

- Metropolis Light Transport
- Virtual point lights / Many-light methods
- Precomputed radiance transfer
- Participating media + subsurface scattering
- Real-time GI
- Hair rendering
- Appearance measurement and modeling

Metropolis Light Transport



(a) Bidirectional path tracing with 40 samples per pixel.

Image credit: Eric Veach

Metropolis Light Transport

(b) Metropolis light transport with an average of 250 mutations per pixel [the same computation time as (a)].

Image credit: Eric Veach

Metropolis Photon Tracing

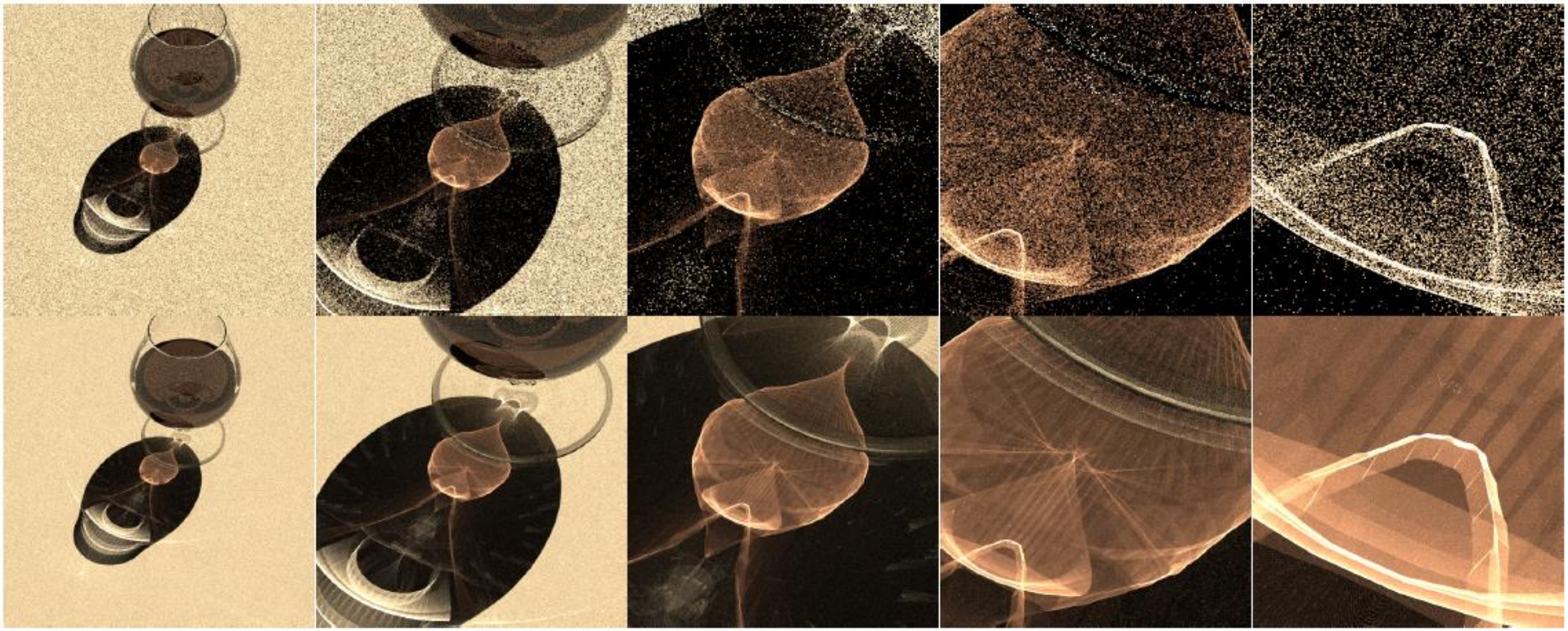


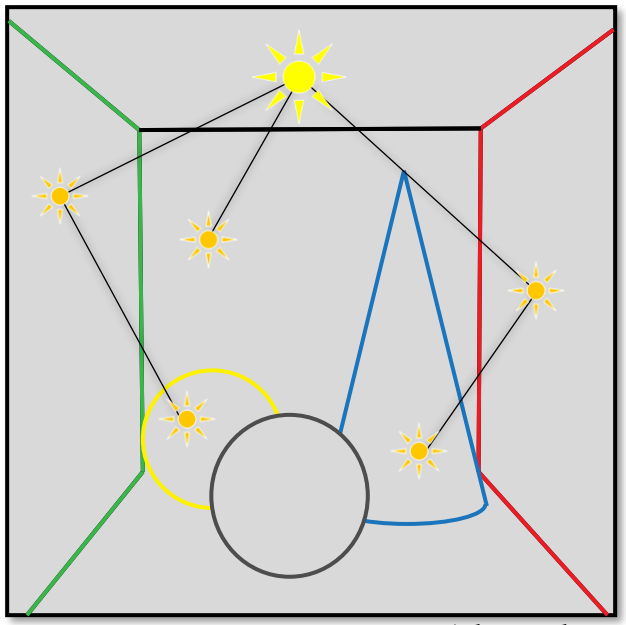
Image credit: Toshiya Hachisuka

Instant radiosity (VPL rendering)

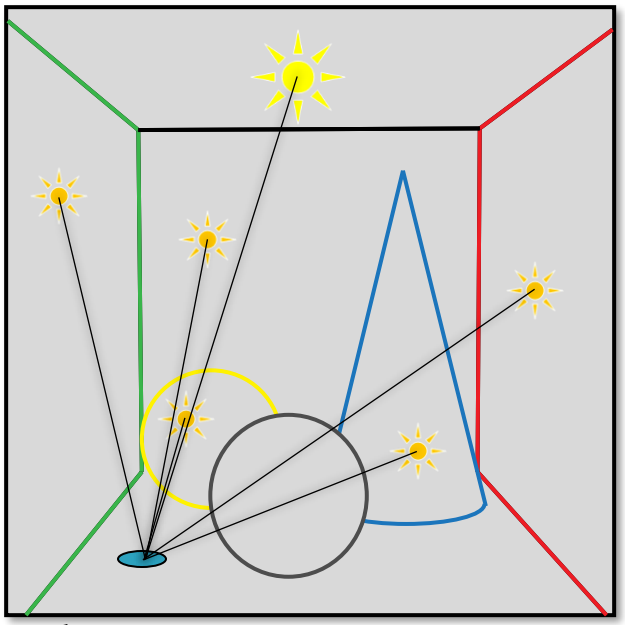
- [Keller 1997]
- Approximate indirect illumination by

Virtual Point Lights (VPLs)

1. Generate VPLs



2. Render with VPLs



Precomputed radiance transfer

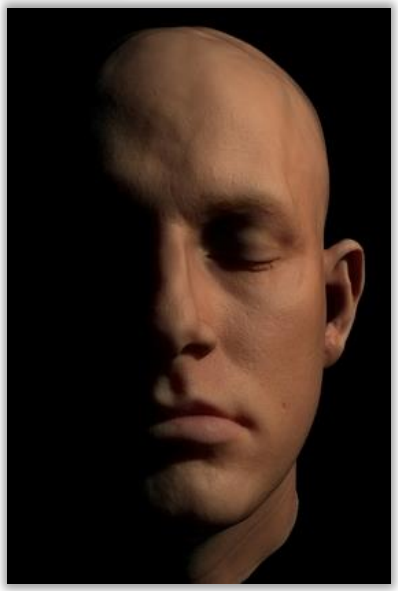
$$= \begin{bmatrix} T_{11} & T_{12} & \cdots & T_{1M} \\ T_{21} & T_{22} & \cdots & T_{2M} \\ T_{31} & T_{32} & \cdots & T_{3M} \\ \vdots & \vdots & \ddots & \vdots \\ T_{N1} & T_{N2} & \cdots & T_{NM} \end{bmatrix} \begin{bmatrix} L_1 \\ L_2 \\ \vdots \\ L_M \end{bmatrix}$$

Participating media

Subsurface scattering

Real

Simulated



Real-time GI

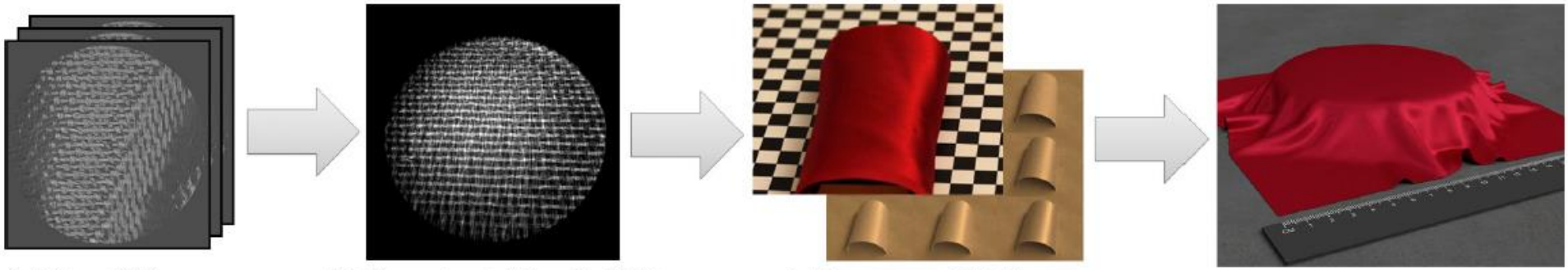
- VPL methods
- Screen-space methods
- Cone-tracing (Unreal Engine)
- Light propagation volumes (CryEngine)

Hair rendering

Single Scattering Only
(offline)
3 minutes

Path Tracing Reference
(offline)
22 hours

Appearance modeling



(a) Micro CT Images

(b) Reconstructed Density Field and Orientation Field

(c) Appearance Matching

(d) Rendered Results

Conclusion

Research challenges in rendering

- Existing algorithms are inherently bad for some practical scenes

- More work to do for rendering researchers

What else in CG

- Main general CG conferences
  - SIGGRAPH (ACM Transactions on Graphics – TOG)
  - SIGGRAPH Asia (ACM TOG)
  - Eurographics (Computer Graphics Forum)

- http://kesen.realtimerendering.com/

What else in CG

- Computational photography
- Appearance modeling & capture
- Animation (& capture)
- Dynamic simulation (hair, cloth, water, smoke, solids…)
- Visual perception
- Natural phenomena
- Non-photorealistic rendering
- Sound simulation
- Display technology
- Interaction technology
- Geometry modeling

General challenges in CG

- **Making CG usable:** UI design, collaboration
- Robust and efficient lighting simulation
- Virtual human
  - Hair modeling
  - **Animation**
  - Cloth
- Managing complexity
  - Natural environments etc
- Virtual Worlds (shared 3D graphics)
- …and more (the above is my random choice of "grand challenges")