Optimizing Realistic Rendering with Many-Light Methods

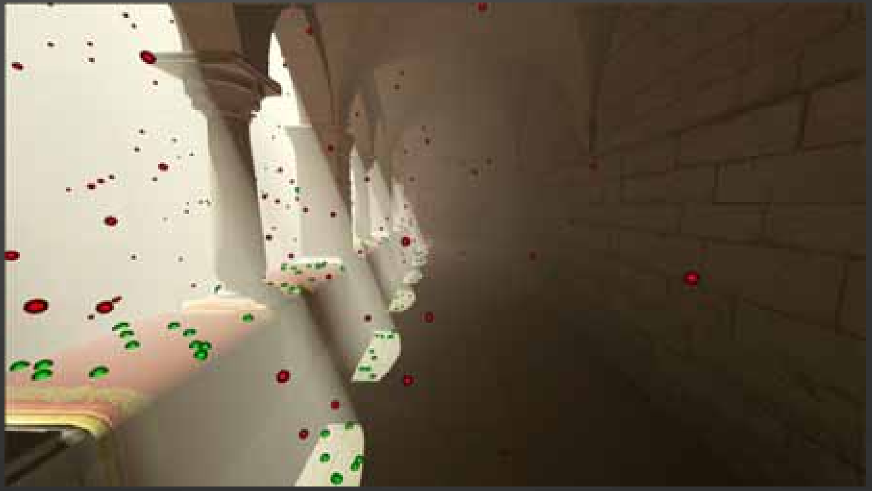
Real-Time Many-Light Rendering

Carsten Dachsbacher
Computer Graphics Group
Karlsruhe Institute of Technology

Real-time Many-light Rendering

Outline

- main difference to offline-methods is visibility computation
  - rasterization instead of raycasting
  - VPL generation
  - lighting and shadowing from VPLs
- high-quality rendering
  - bias compensation in screen-space
  - approximate compensation in participating media rendering

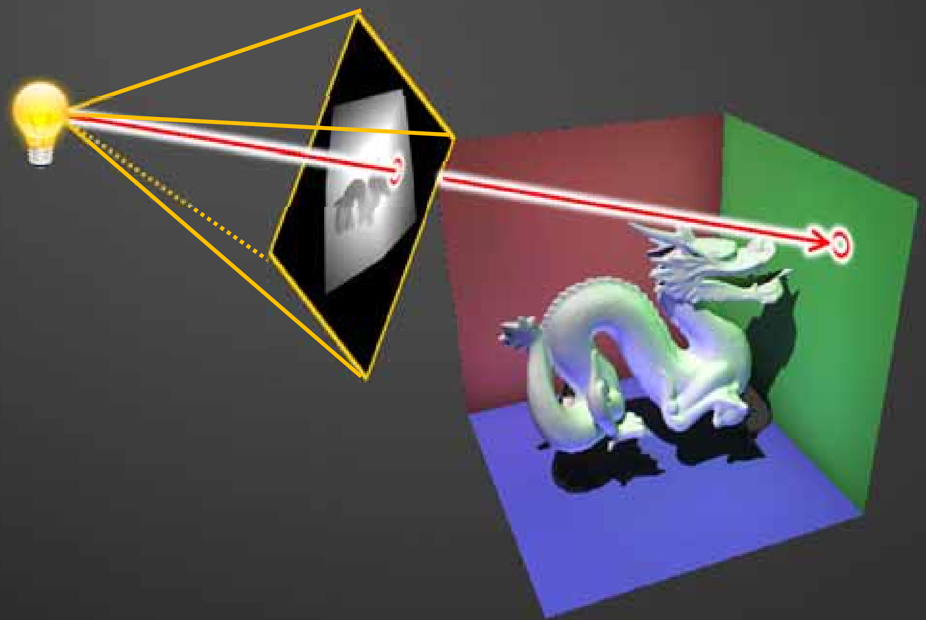


Real-time Many-light Rendering

Visibility Computation for VPL Generation

- real-time rendering ↔ mostly diffuse scenes ↔ relatively few VPLs (~10^3)
- if acceleration structure available use ray casting

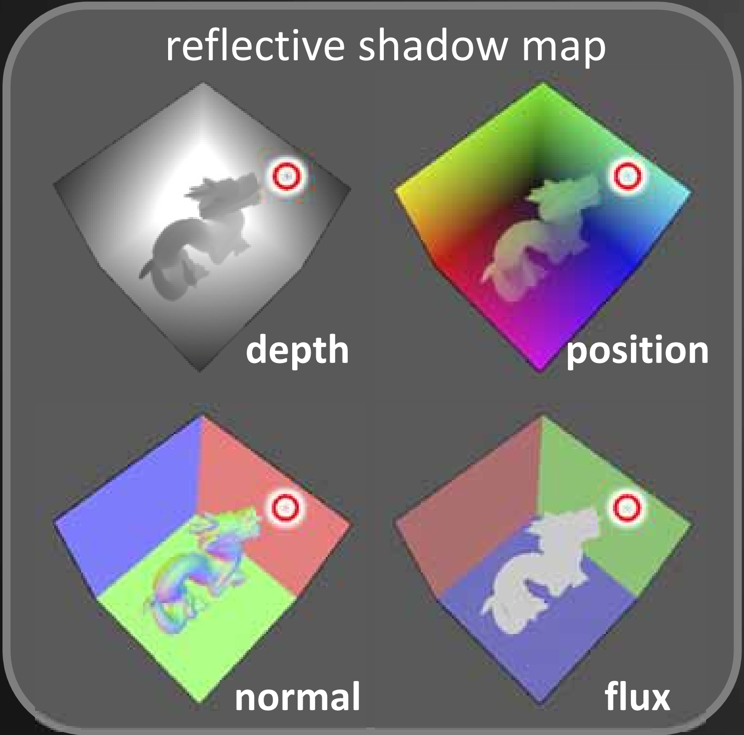
- VPL generation with rasterization
  - render scene from light
  - observation: visible surfaces = first intersection of light path

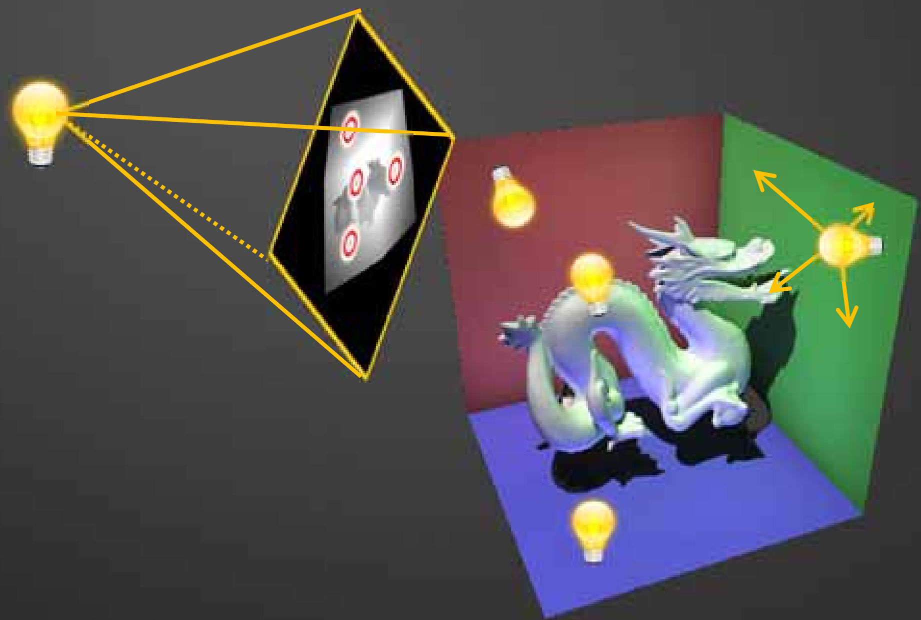


Real-time Many-light Rendering

VPL Generation with Rasterization

- render scene from light into reflective shadow map [DS05]: all information available for creating VPLs and continuing paths
  - single bounce indirect illumination by directly sampling the RSM
  - importance sampling can easily be added [DS06]
- proceed recursively by rendering another RSM

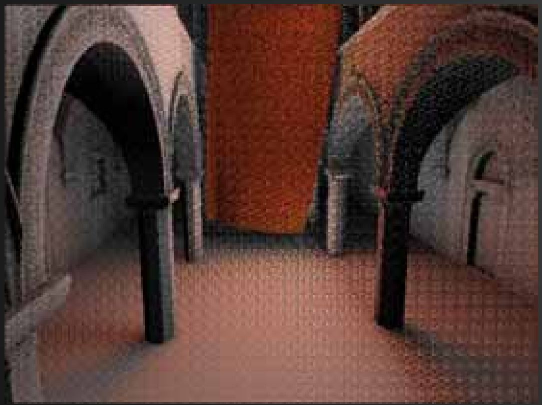




Rendering with VPLs

Lighting and Shadowing

- many lights can be handled with deferred shading
  - interleaved sampling (problem: detailed normals/geometry) [Seg06]
  - hierarchical shading [NW10]
  - accumulate and filter incident light [SW09]
  - clustered deferred and forward shading [OBA12]

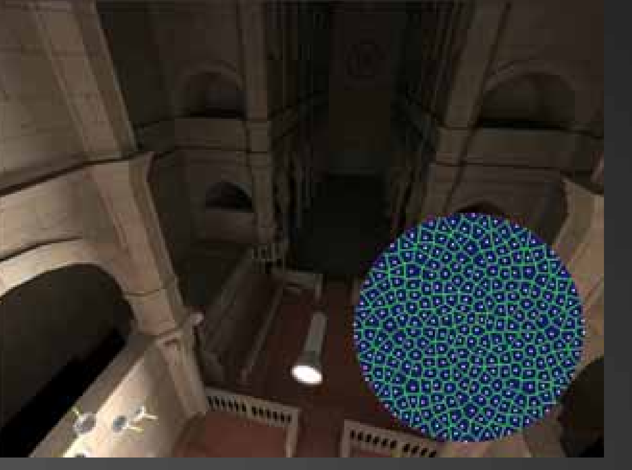


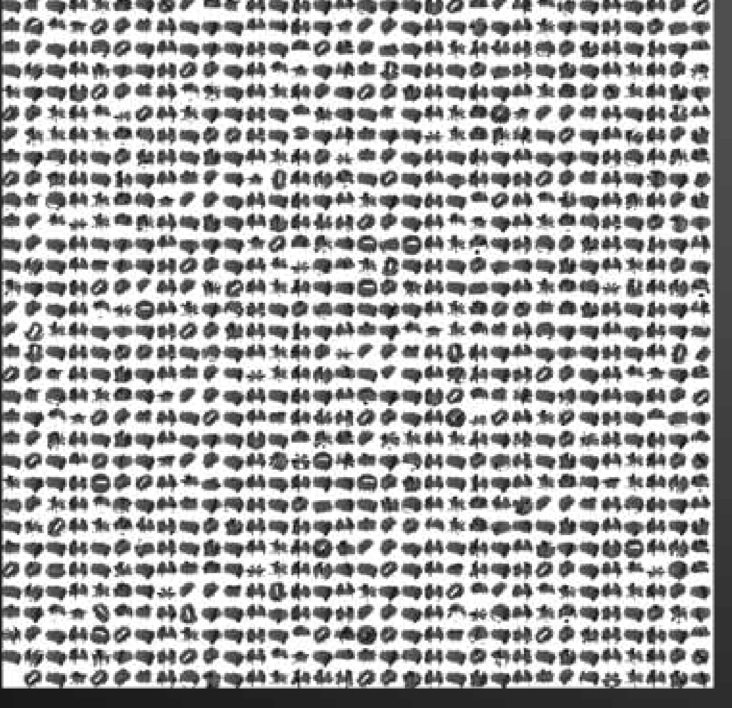

- **bottleneck: shadow computation**

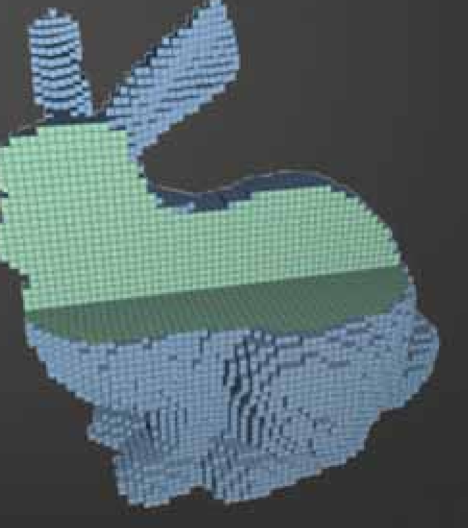
Rendering with VPLs

Shadow Computation

- …is the real bottleneck with instant radiosity / many lights methods
  - exploit temporal coherency [LSKLA07]
  - sampled visibility
    - voxelization, e.g. [SS10]
    - faster shadow maps



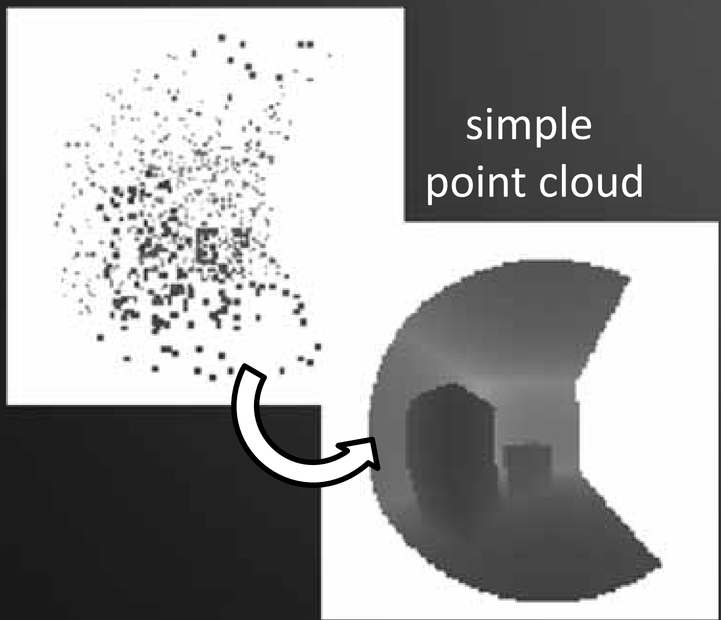




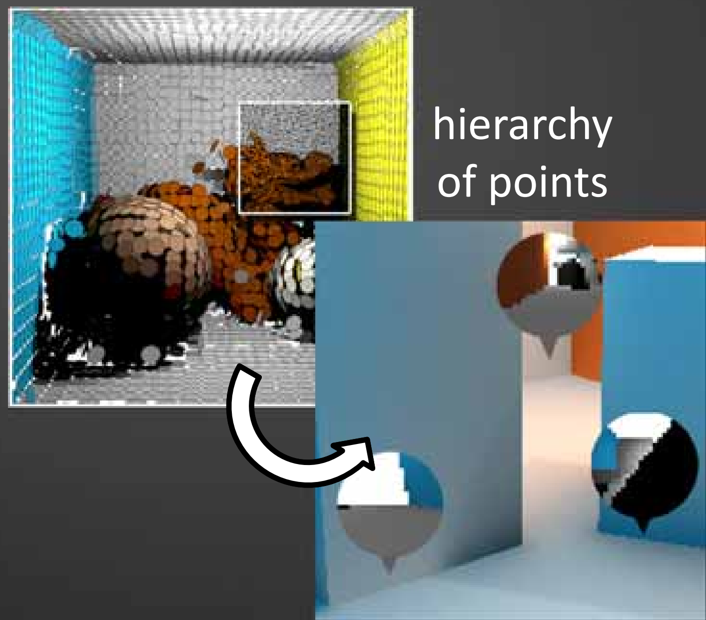
Shadow Mapping for VPLs

Problem Setting

- need **many shadow maps** of **low/moderate resolution**
- rendering the scene many times (transformation, …) is costly
  - what we need is level-of-detail rendering
  - point representations are well-suited for fast, approximate renderings
  - two approaches: simple LOD with no connectivity and water-tight rendering with point hierarchy



simple point cloud

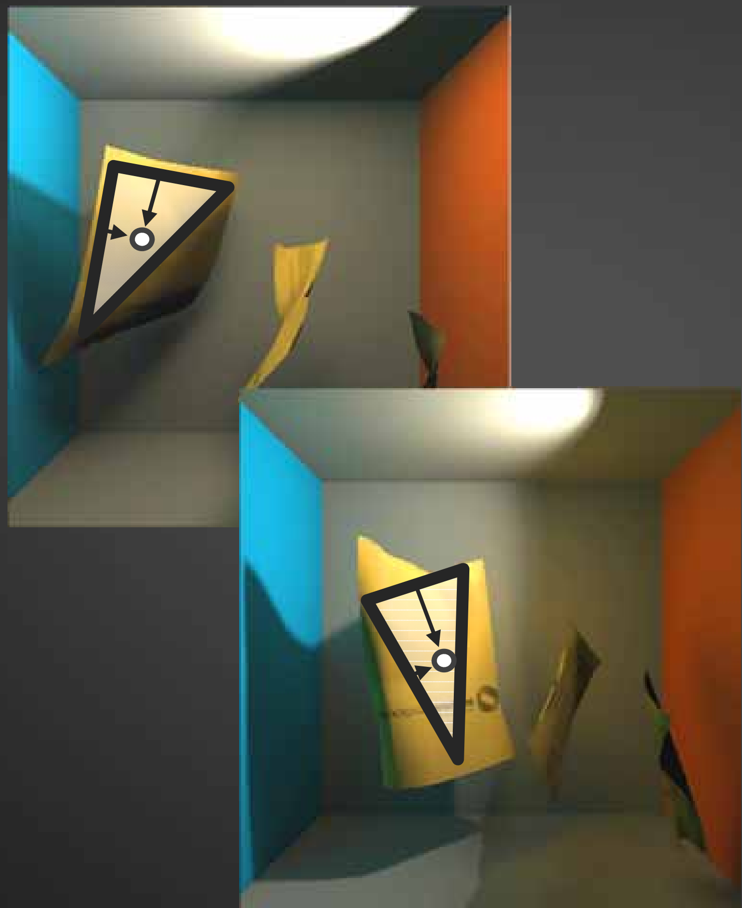


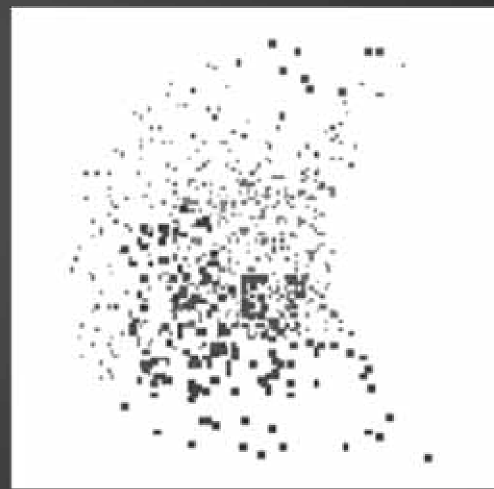
hierarchy of points

Shadow Mapping for VPLs

Imperfect Shadow Maps

- create random sets of point samples (triangle ID + barycentric coords)
- 4k to 16k points per “shadow map” (global parameter)

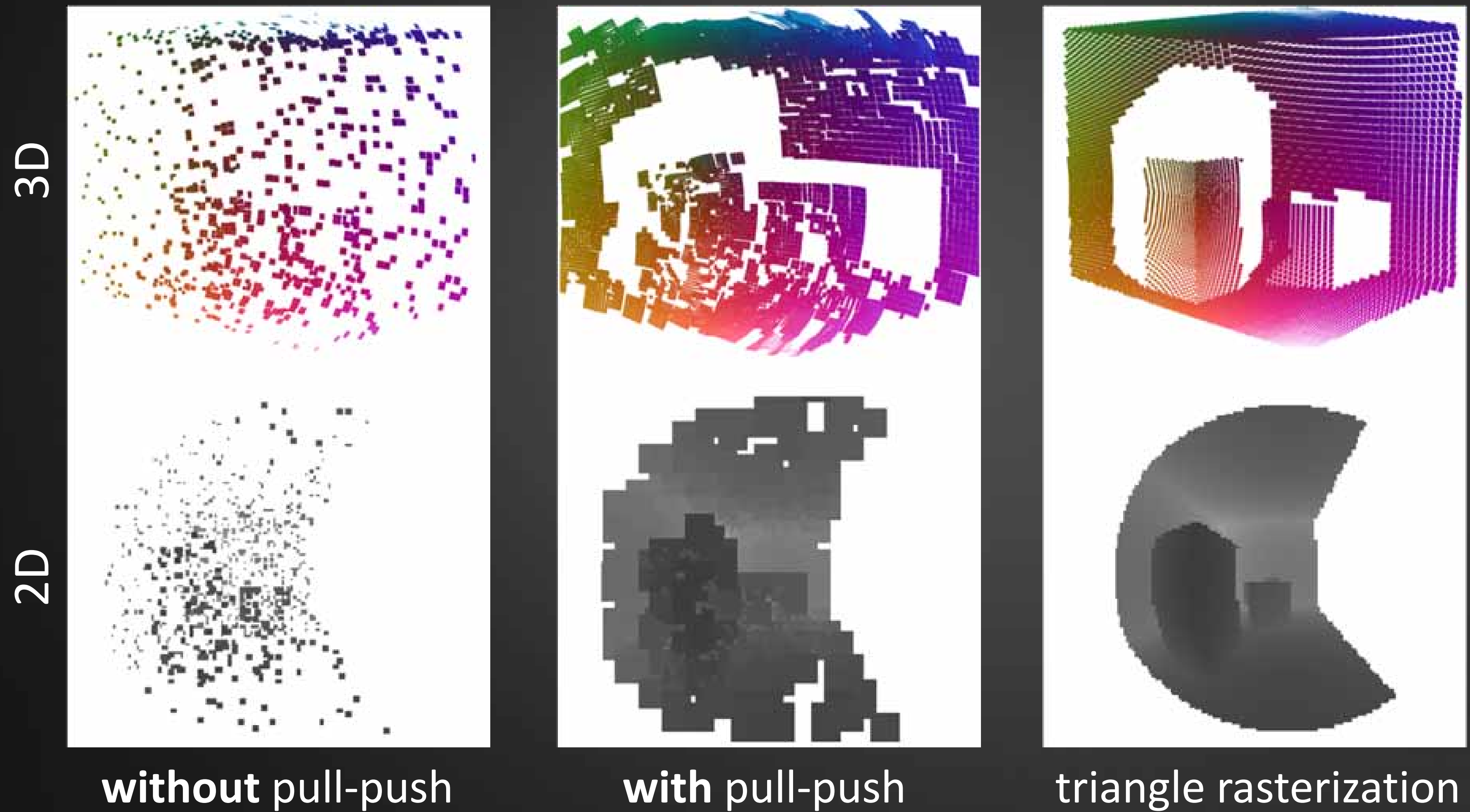




Shadow Mapping for VPLs

Imperfect Shadow Maps

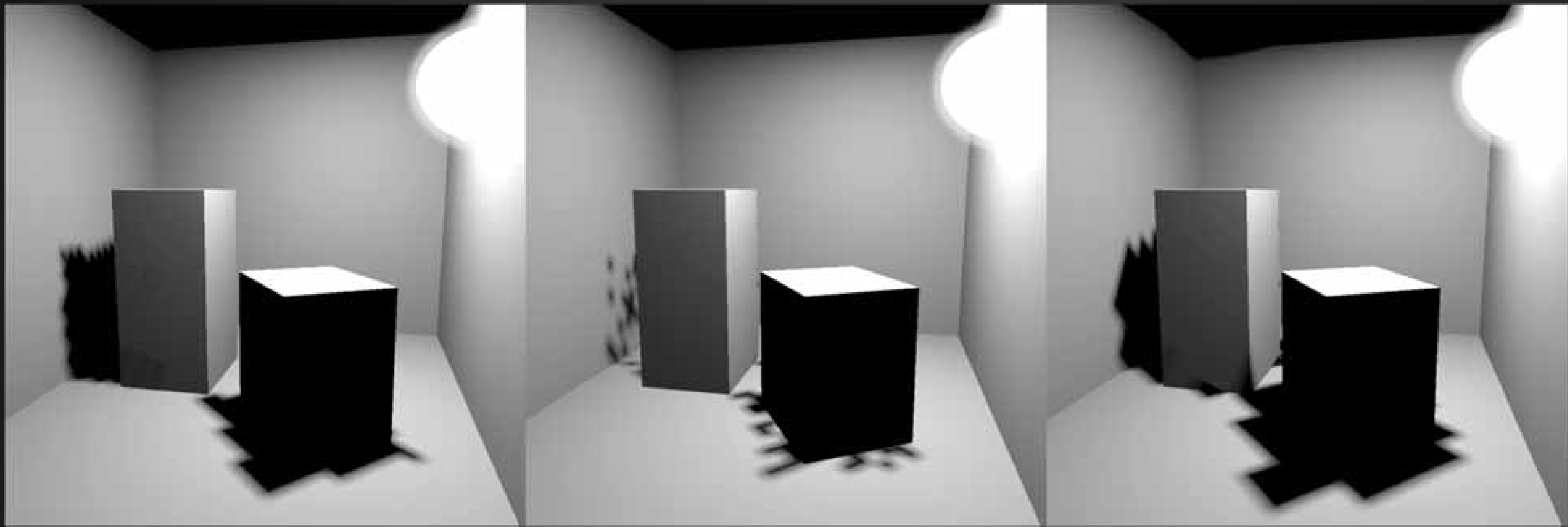
- 4k to 16k points per “shadow map” (global parameter)
- heuristic to reconstruct the surfaces from point samples



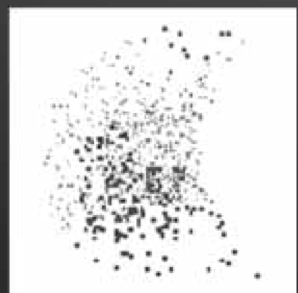
Shadow Mapping for VPLs

Imperfect Shadow Maps

- comparison of shadow maps for a single point light



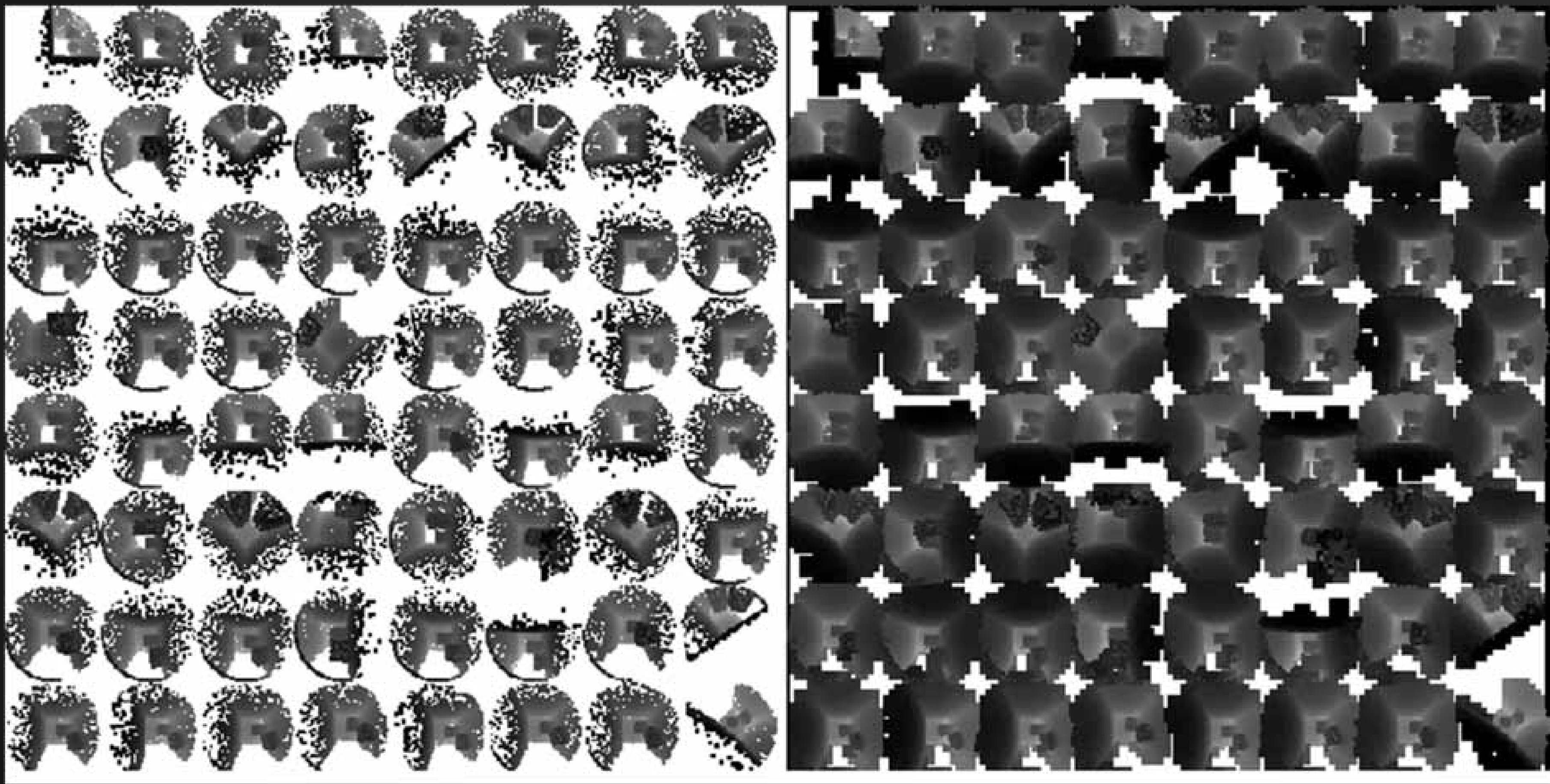
triangle rasterization | **without** pull-push | **with** pull-push



Shadow Mapping for VPLs

Imperfect Shadow Maps

- pull-push in image-space: parallel for thousands of shadow maps



without pull-push   **with** pull-push

Shadow Mapping for VPLs

Imperfect Shadow Maps

- … can render thousands of shadow maps in 100ms
- … work because errors average out
- … require playing with parameters

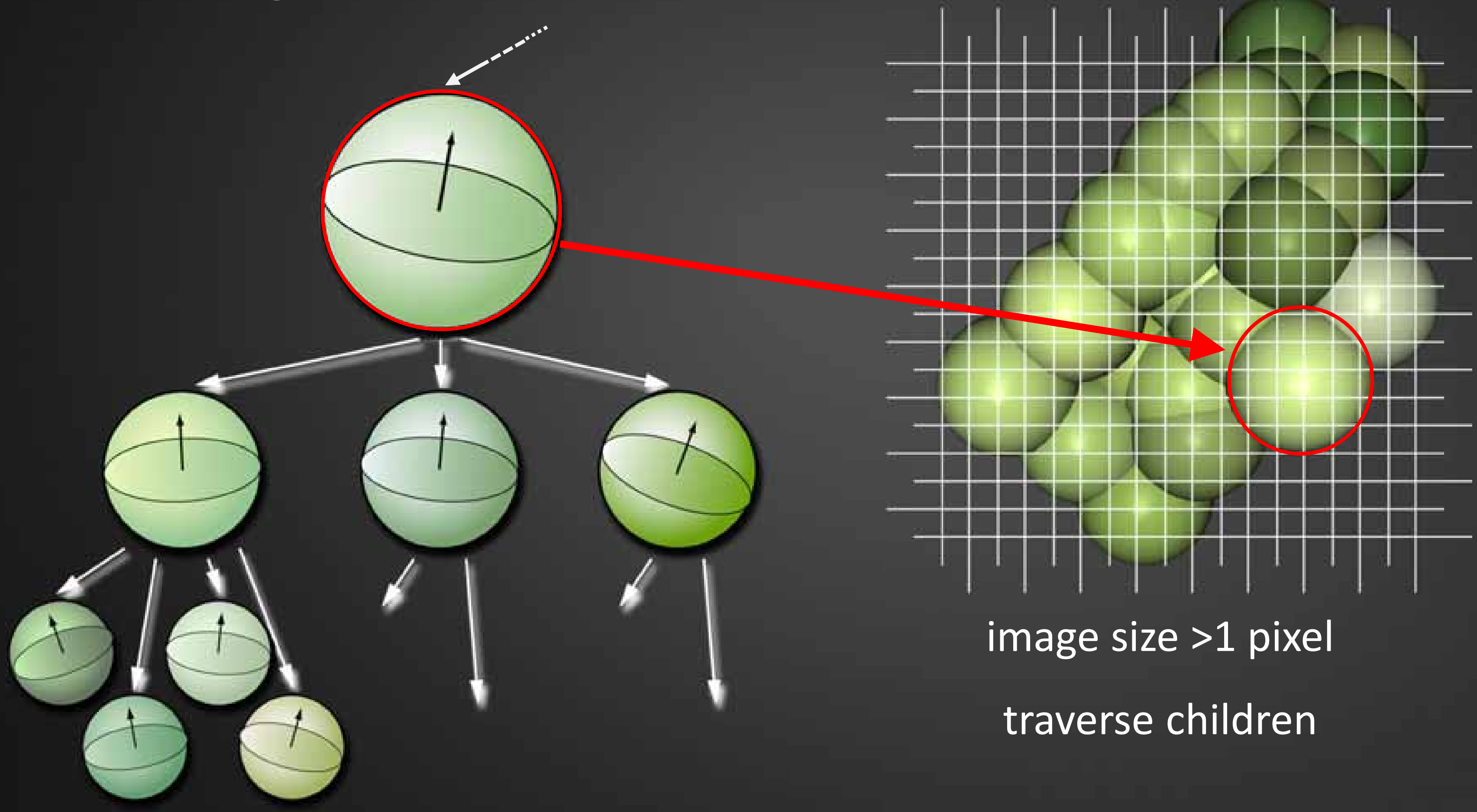
"perfect" shadow maps

imperfect shadow maps

Shadow Mapping for VPLs

High-Quality Point-based Rendering

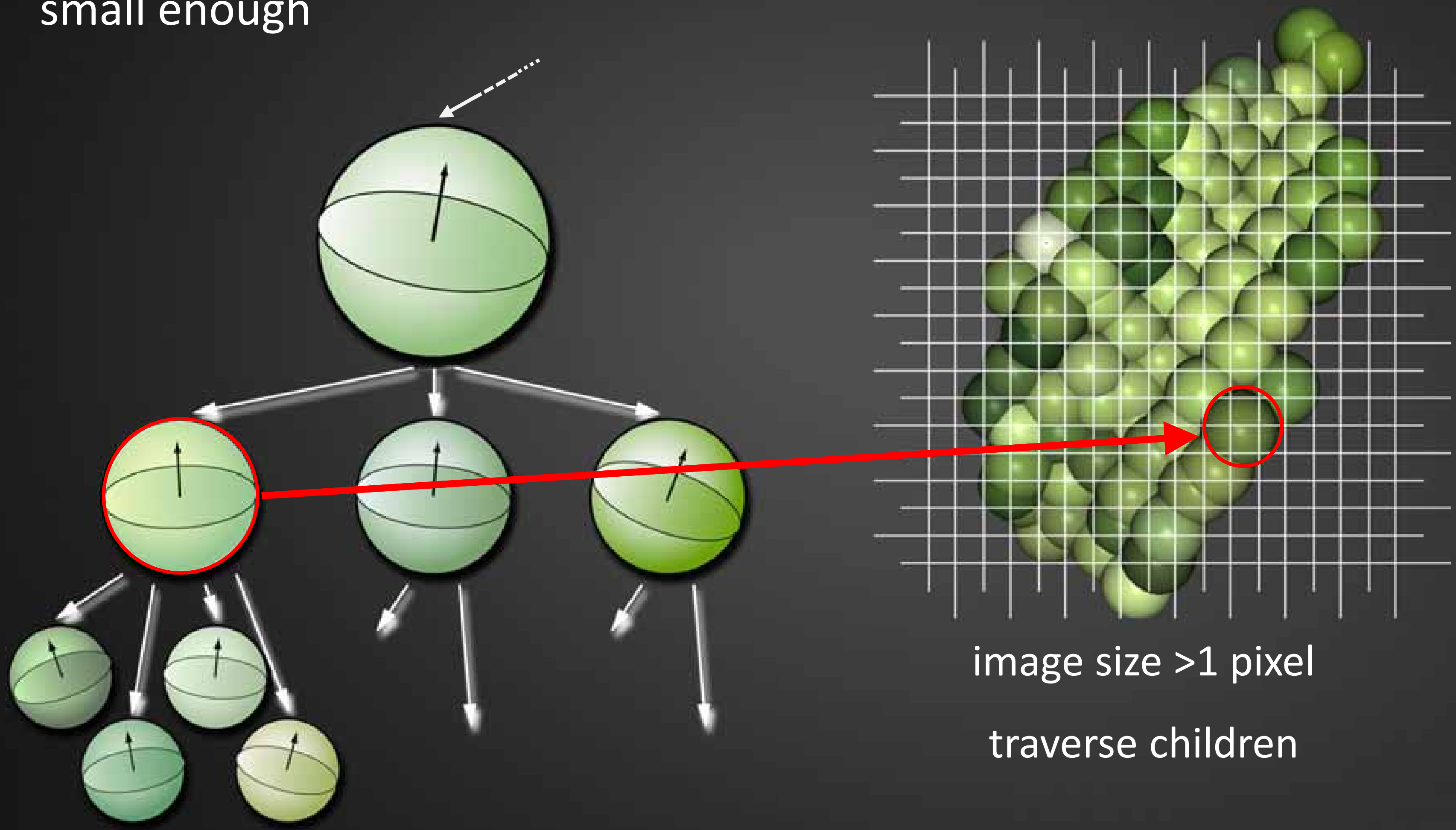
- create random points on surfaces and create hierarchy
- idea of Qsplat: traverse hierarchy until projected size of point primitive is small enough



Shadow Mapping for VPLs

High-Quality Point-based Rendering

- create random points on surfaces and create hierarchy
- idea of Qsplat: traverse hierarchy until projected size of point primitive is small enough



Shadow Mapping for VPLs

High-Quality Point-based Rendering

- create random points on surfaces and create hierarchy
- idea of Qsplat: traverse hierarchy until projected size of point primitive is small enough

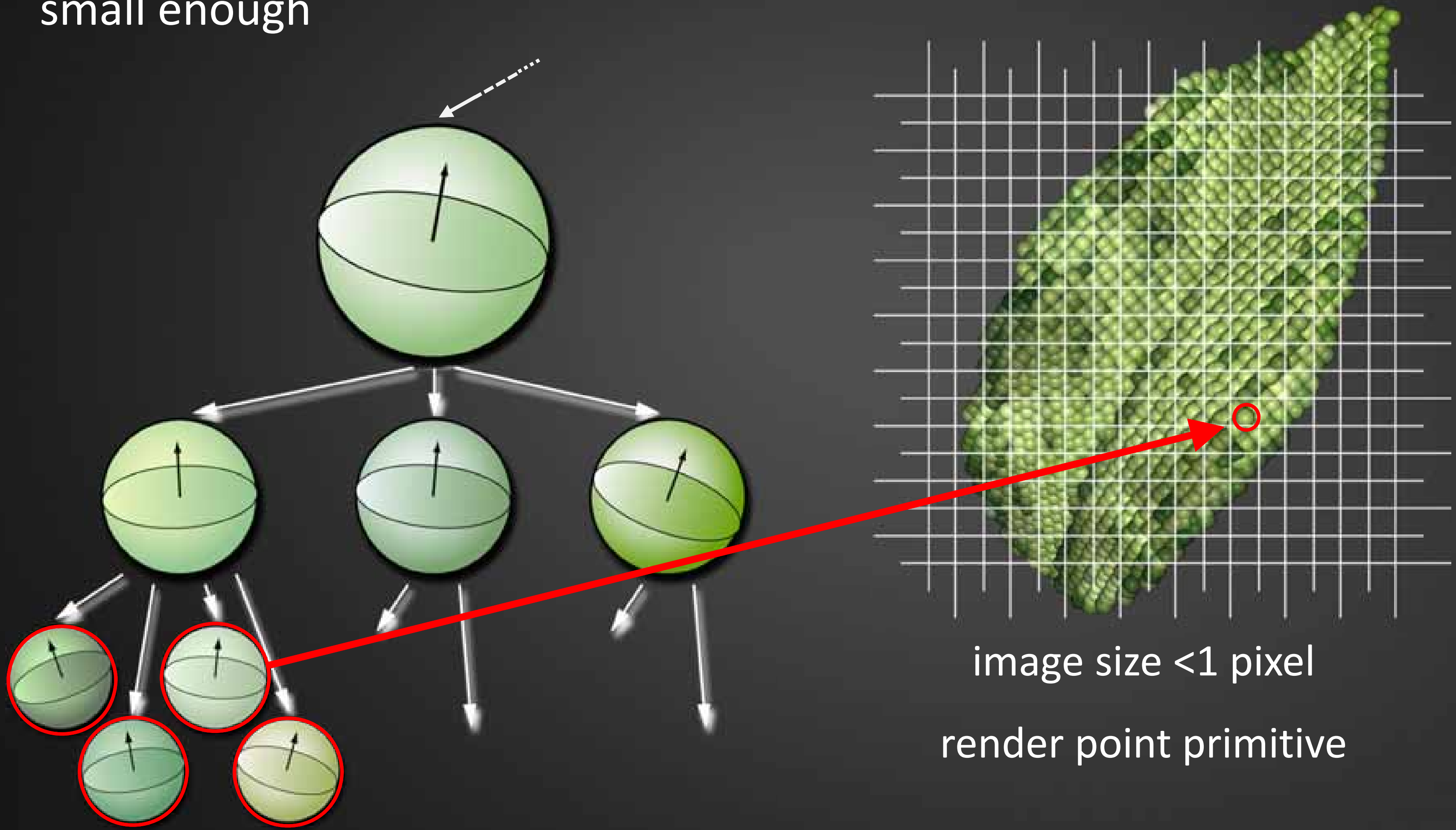


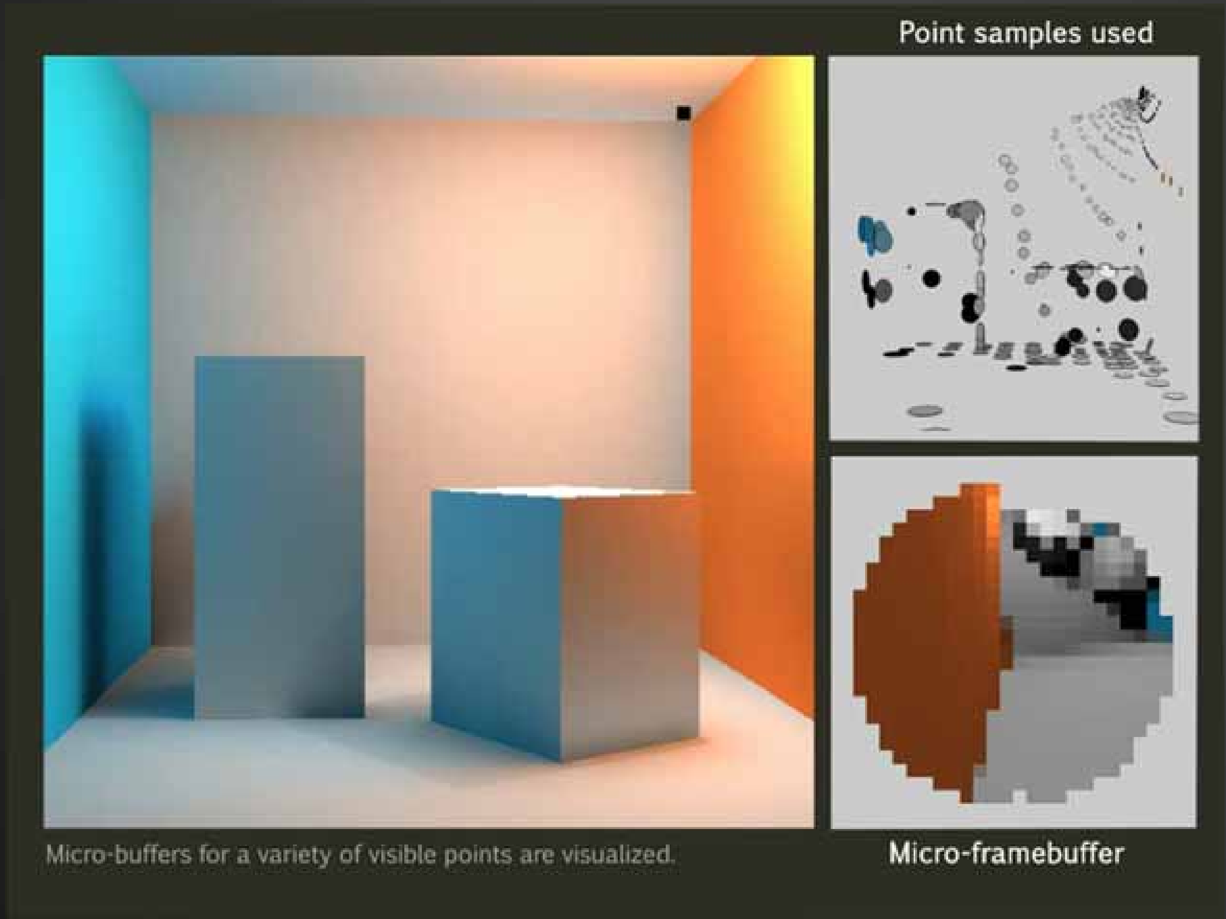
image size <1 pixel

render point primitive

Shadow Mapping for VPLs

Micro-Rendering

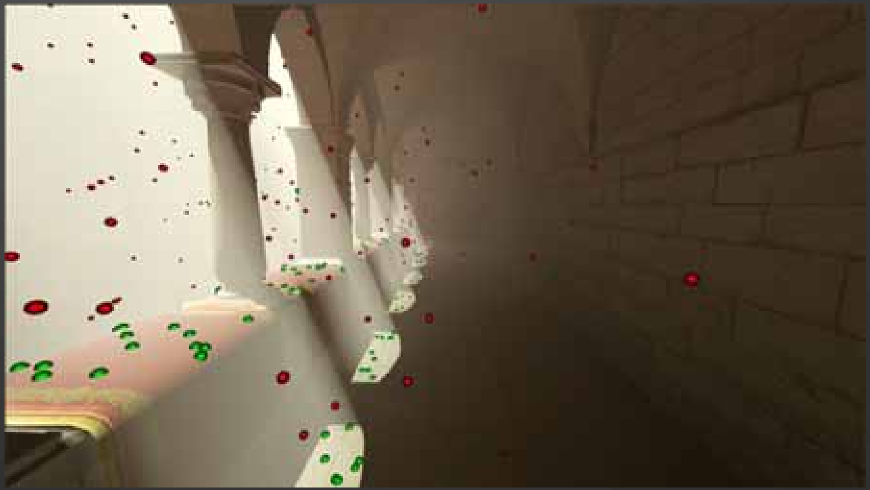
- renders accurate environment maps / depth buffers from point hierarchy
- actually developed for final gathering, using CUDA/OpenCL
- can be used to create (R)SMs (in 2009: ~16k in 100 ms, each 24^2 pixels)



Real-time Many-light Rendering

Outline

- main difference to offline-methods is visibility computation
  - rasterization instead of raycasting
  - VPL generation
  - lighting and shadowing from VPLs
- high-quality rendering
  - bias compensation in screen-space
  - approximate compensation in participating media rendering

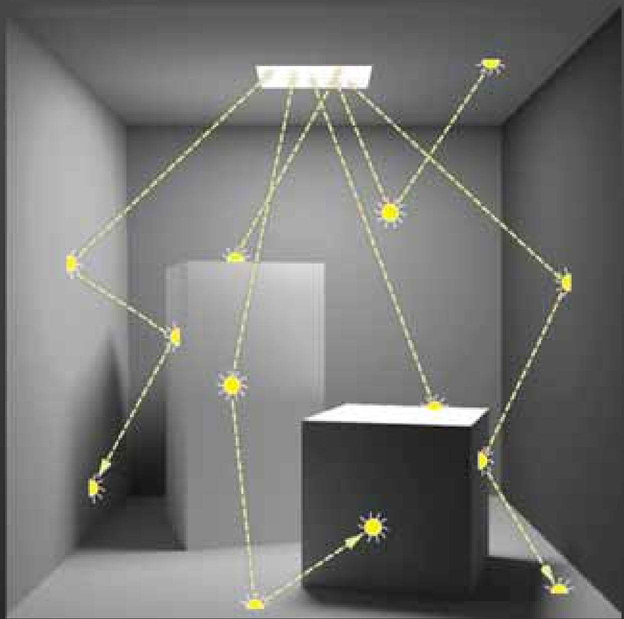


Singularities and Bias Compensation

- so far: VPL generation, shading and shadowing
- we assume to use VPLs to approximate indirect illumination $\hat{L}$ only

$$L = L_e + \mathbf{T}L$$

$$\underbrace{L = L_e}_{\text{direct emission}} + \underbrace{\mathbf{T}L_e}_{\text{direct illumination}} + \underbrace{\mathbf{T}\hat{L}}_{\text{indirect illumination}}$$

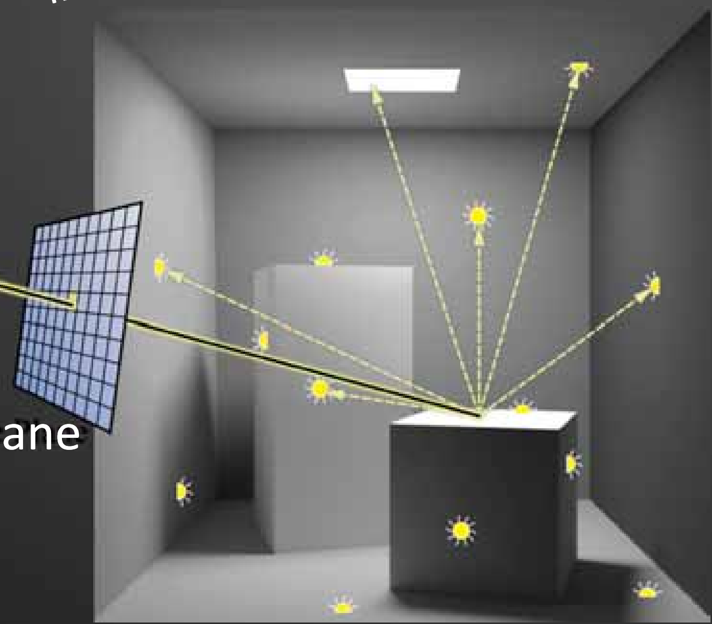


Singularities and Bias Compensation

- so far: VPL generation, shading and shadowing
- we assume to use VPLs to approximate indirect illumination $\hat{L}$ only

$$L = L_e + \mathbf{T}L$$

$$L = \underbrace{L_e}_{\text{direct emission}} + \underbrace{\mathbf{T}L_e}_{\text{direct illumination}} + \underbrace{\mathbf{T}\hat{L}}_{\text{indirect illumination}}$$

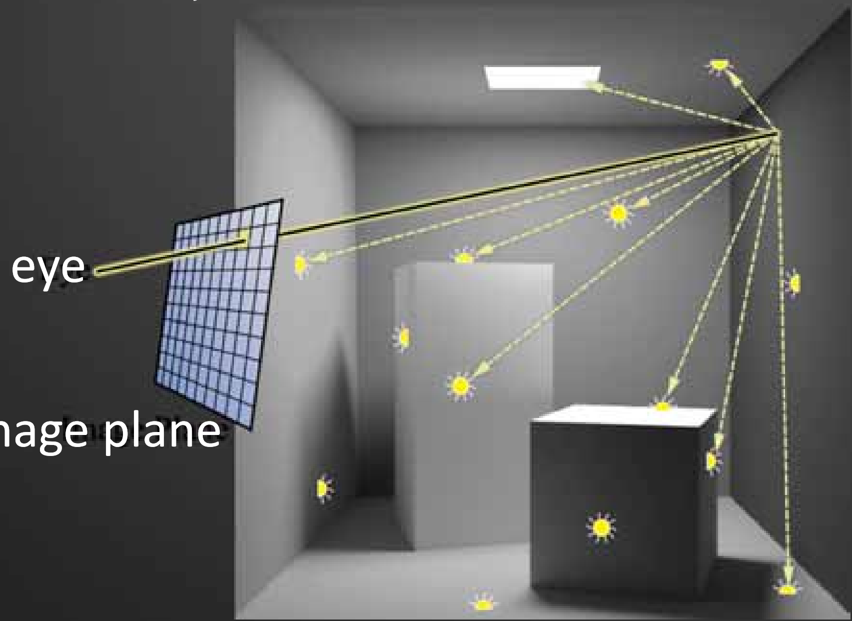


Singularities and Bias Compensation

- so far: VPL generation, shading and shadowing
- we assume to use VPLs to approximate indirect illumination $\hat{L}$ only

$$L = L_e + \mathbf{T}L$$

$$L = \underbrace{L_e}_{\text{direct emission}} + \underbrace{\mathbf{T}L_e}_{\text{direct illumination}} + \underbrace{\mathbf{T}\hat{L}}_{\text{indirect illumination}}$$



Singularities and Bias Compensation

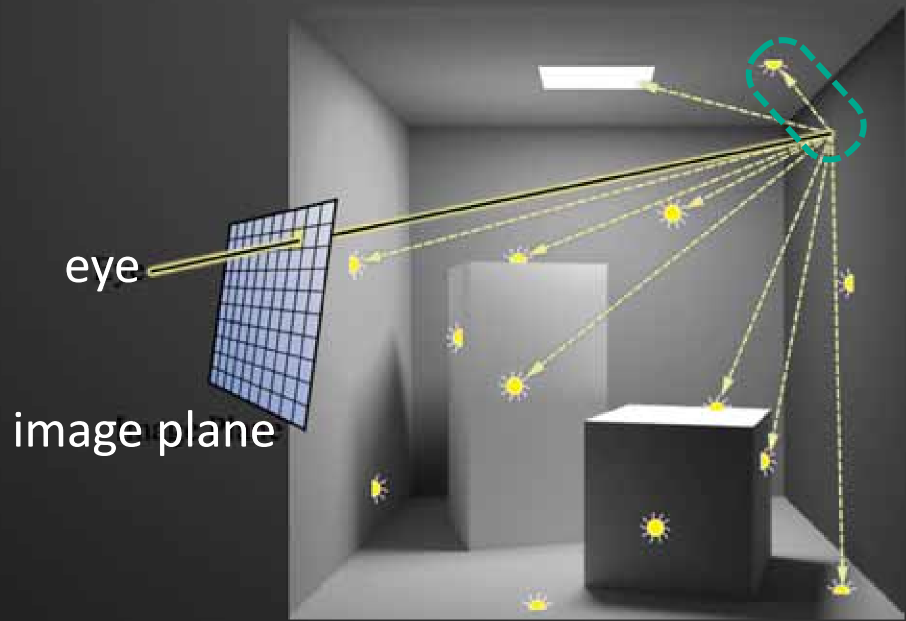
$$L = L_e + \mathbf{T}L_e + \mathbf{T}\hat{L}$$

transport operator:

$$(\mathbf{T}\hat{L})(\mathbf{x}\leftarrow\mathbf{y}) = \sum_{i=1}^{N} f_r(\mathbf{x}\leftarrow\mathbf{y}\leftarrow\mathbf{z}_i)\,G(\mathbf{y}\leftrightarrow\mathbf{z}_i)\,V(\mathbf{y}\leftrightarrow\mathbf{z}_i)\,\hat{L}(\mathbf{y}\leftarrow\mathbf{z}_i)$$

geometry term:

$$G(\mathbf{y}\leftrightarrow\mathbf{z}_i) = \frac{\cos^+(\theta_{\mathbf{y}})\cos^+(\theta_{\mathbf{z}_i})}{\|\mathbf{y}-\mathbf{z}_i\|^2}$$



Singularities and Bias Compensation

reference (slow) rendering

fast rendering with few VPLs

clamping VPLs’ contribution

clamping the contribution of nearby VPLs
by bounding the geometry term

Singularities and Bias Compensation

reference (slow) rendering | **DIFFERENCE** | clamping VPLs' contribution

clamping removes short distance light transport.

How do we restore the missing energy?

Bounded and Residual Light Transport

full LT: $L_e + \mathbf{T}L_e + \mathbf{T}\hat{L}$

$$\mathbf{T}\hat{L} = \sum_{i=1}^{N} f_r \, G \, V \, \hat{L}$$

bounded indirect LT: $L_e + \mathbf{T}L_e + \mathbf{T}_b\hat{L}$

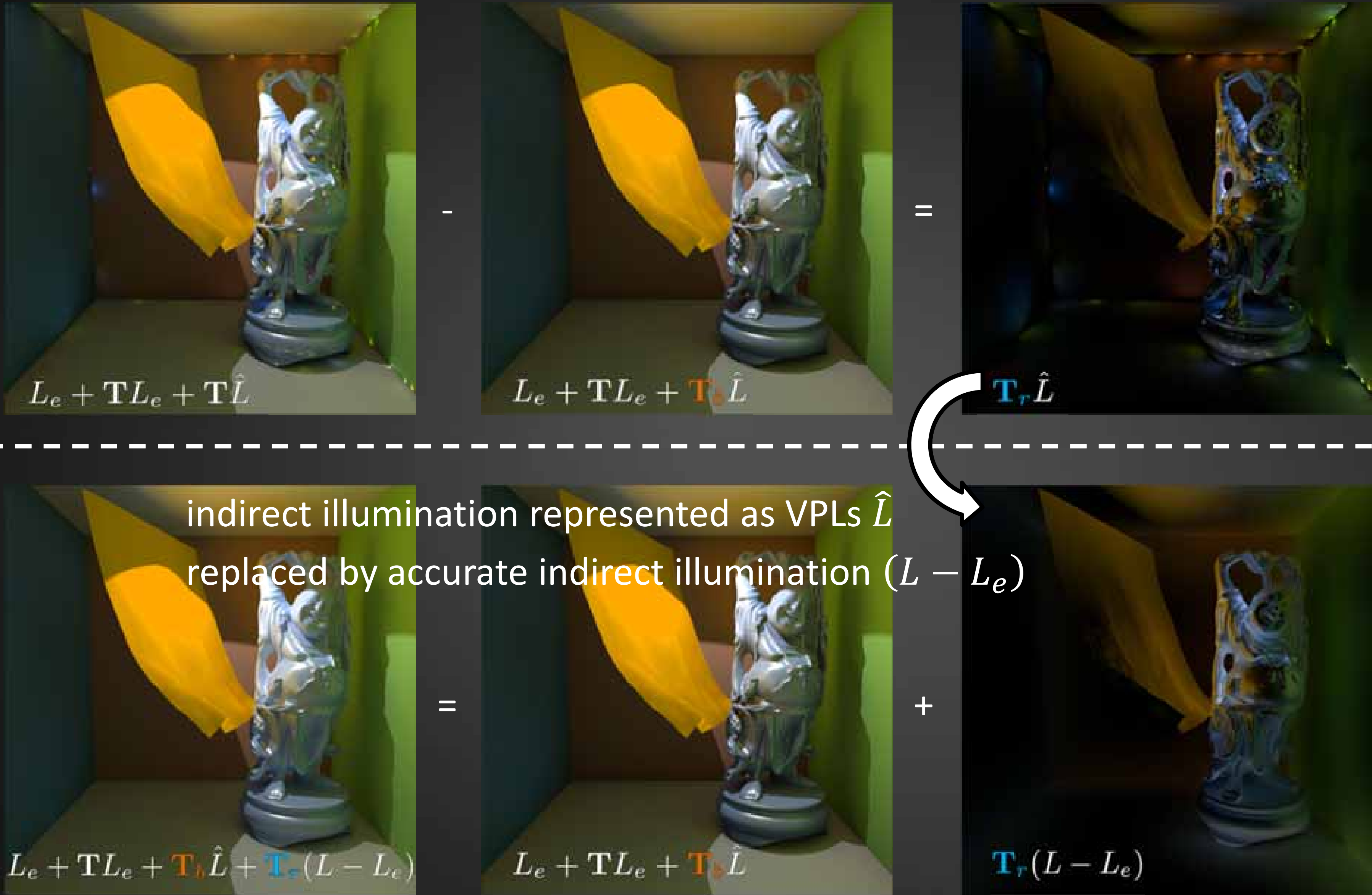
$$\mathbf{T}_b\hat{L} = \sum_{i=1}^{N} f_r \, \min(G, b) \, V \, \hat{L}$$

residual indirect LT: $\mathbf{T}_r\hat{L}$

$$\mathbf{T}_r\hat{L} = \sum_{i=1}^{N} f_r \, \max(G - b, 0) \, V \, \hat{L}$$

b: user-defined bound

Bounded and Residual Light Transport



indirect illumination represented as VPLs $\hat{L}$
replaced by accurate indirect illumination $(L - L_e)$

Bias Compensation

Bias Compensation [KK04]

- $\mathrm{T}_r(L - L_e)$ computed with MC integration
- can degenerate to path tracing: too expensive for real-time rendering

Reformulated Bias Compensation

- re-use the existing (clamped) solution
- iteratively apply the residual transport

$(L - L_e)$

$$L = L_e + \mathbf{T}L_e + \mathbf{T}_b\hat{L} + \mathbf{T}_r(L - L_e)$$

recursive expansion

$$L = L_e + \sum_{i=0}^{\infty} \mathbf{T}_r{}^i (\mathbf{T}L_e + \mathbf{T}_b\hat{L})$$

apply iteratively

compute once

design choice: compute and apply in screen-space

Screen-Space Bias Compensation

Algorithm Overview

- precomputation
  1. distribute VPLs (as before)
  2. create an imperfect shadow map for every VPL

- rendering
  1. create deferred shading buffers
  2. apply deferred direct and bounded VPL lighting $\mathbf{T}L_e + \mathbf{T}_b\hat{L}$
  3. N-times in screen-space:
     compute residual transport and add it to the image

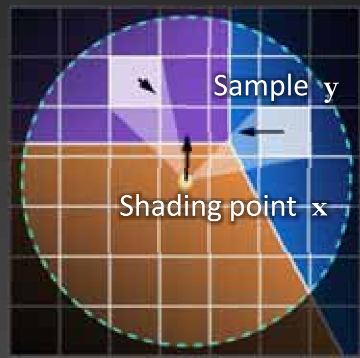
$$\sum_{i=0}^{\infty} \mathbf{T}_r{}^i(\mathbf{T}L_e + \mathbf{T}_b\hat{L})$$

Screen-Space Bias Compensation

Residual Transport Integration (1 iteration)

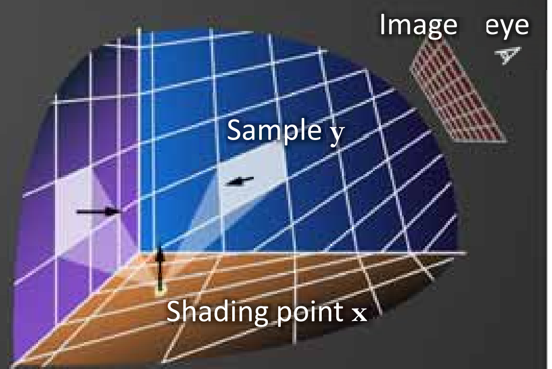
- **FOR EACH** pixel:
  - iterate over neighboring pixels
    - IF $G(\mathbf{x}\leftrightarrow\mathbf{y}) > b$
      - add contribution (with information in G-buffer)

$$G(\mathbf{x}\leftrightarrow\mathbf{y}) = \frac{\cos^{+}(\theta_{\mathbf{x}})\cos^{+}(\theta_{\mathbf{y}})}{\|\mathbf{x}-\mathbf{y}\|^{2}}$$



camera view

- clamping occurs in a close neighborhood only: close in world space = close in screen-space
- we can conservatively estimate a bounding radius and restrict the integration to it



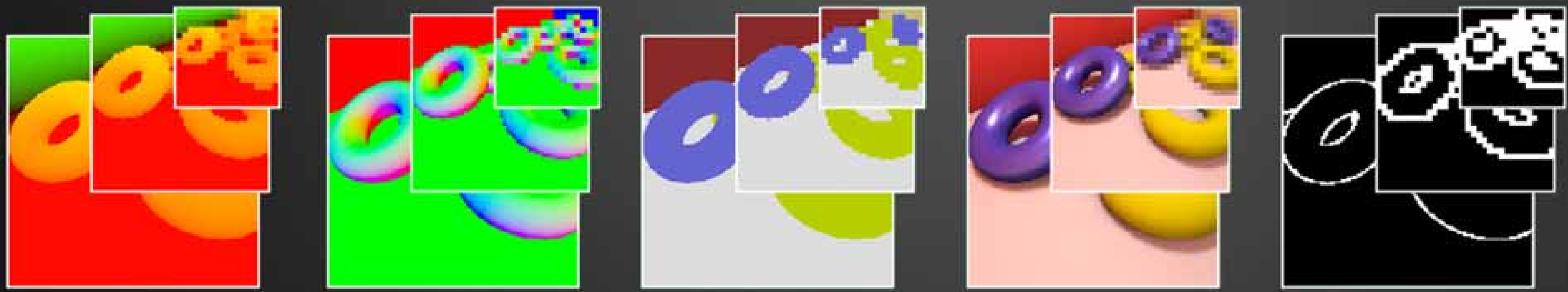
side view

Screen-Space Bias Compensation

Hierarchical Integration

- still too many samples (even with the bounding radius)
- multi-resolution top-down integration (in spirit of [NW09])

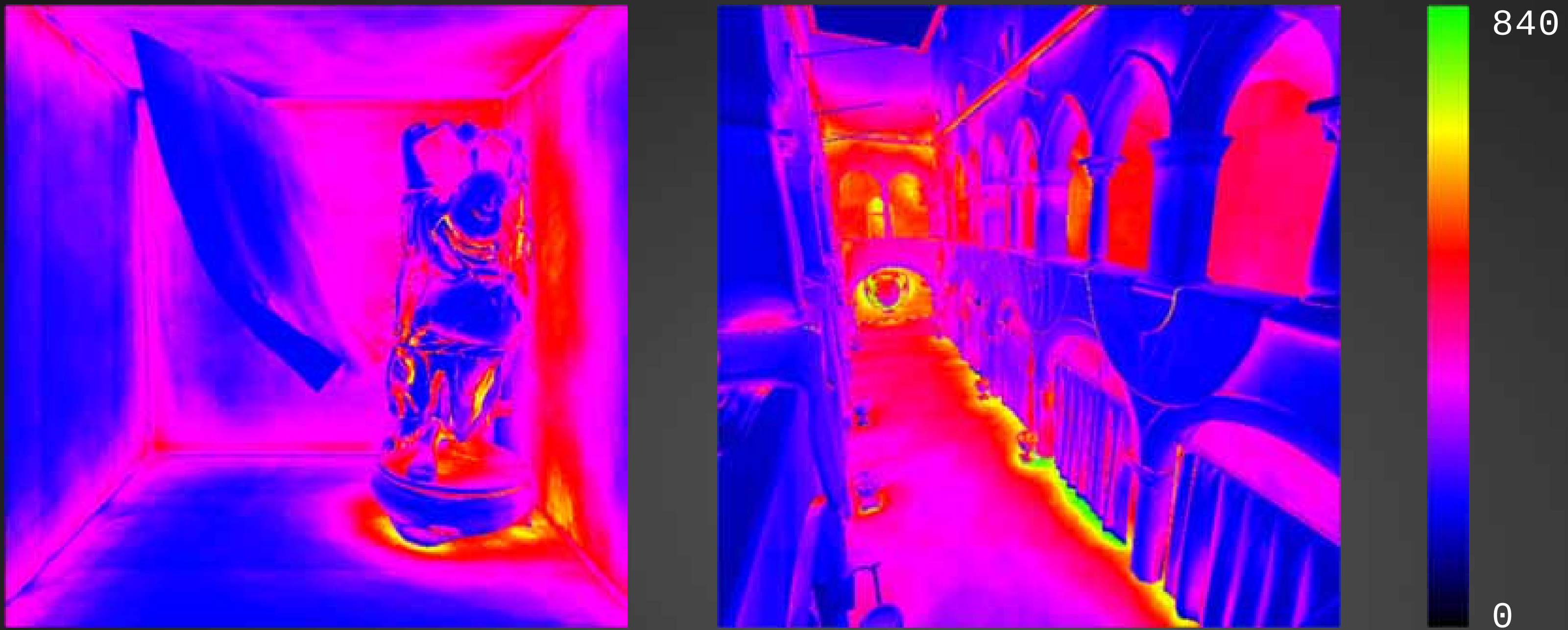
- hierarchical approach requires
  - mip-map chain of the G-Buffer and bounded illumination
  - discontinuity buffer



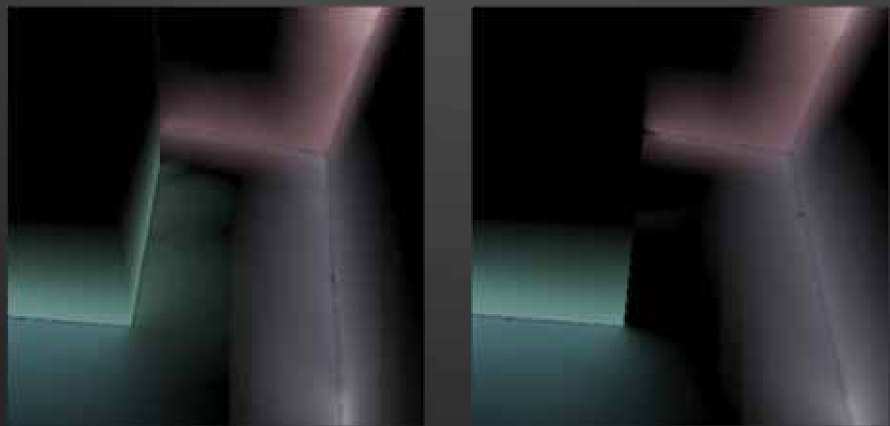
deferred shading buffers   clamped solution   discontinuity buffer

SSBC: Hierarchical Integration

number of samples (per pixel)

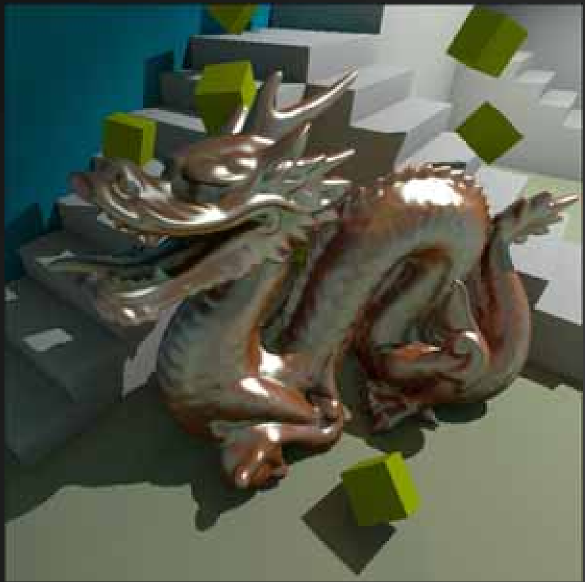


screen space always means: no information on hidden surfaces



Screen Space Bias Compensation

bounded light transport

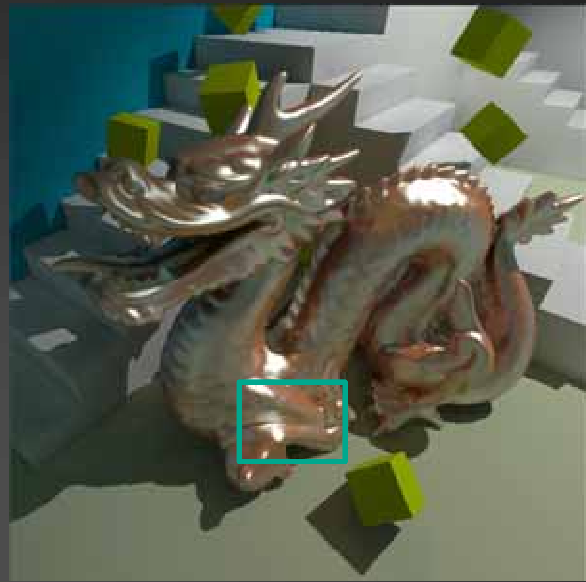


+

residual light transport

=

result



rendered with:
1024x768 at:
(ATI Radeon HD 5870)

no SSBC
10.3 FPS

1 iteration SSBC
8.2 FPS

2 iterations SSBC
6.4 FPS

Comparison to Ground Truth

compensation only | result

bias compensation [KK04]

CPU ~ 10.9 hours
(8-core, 4GB RAM)

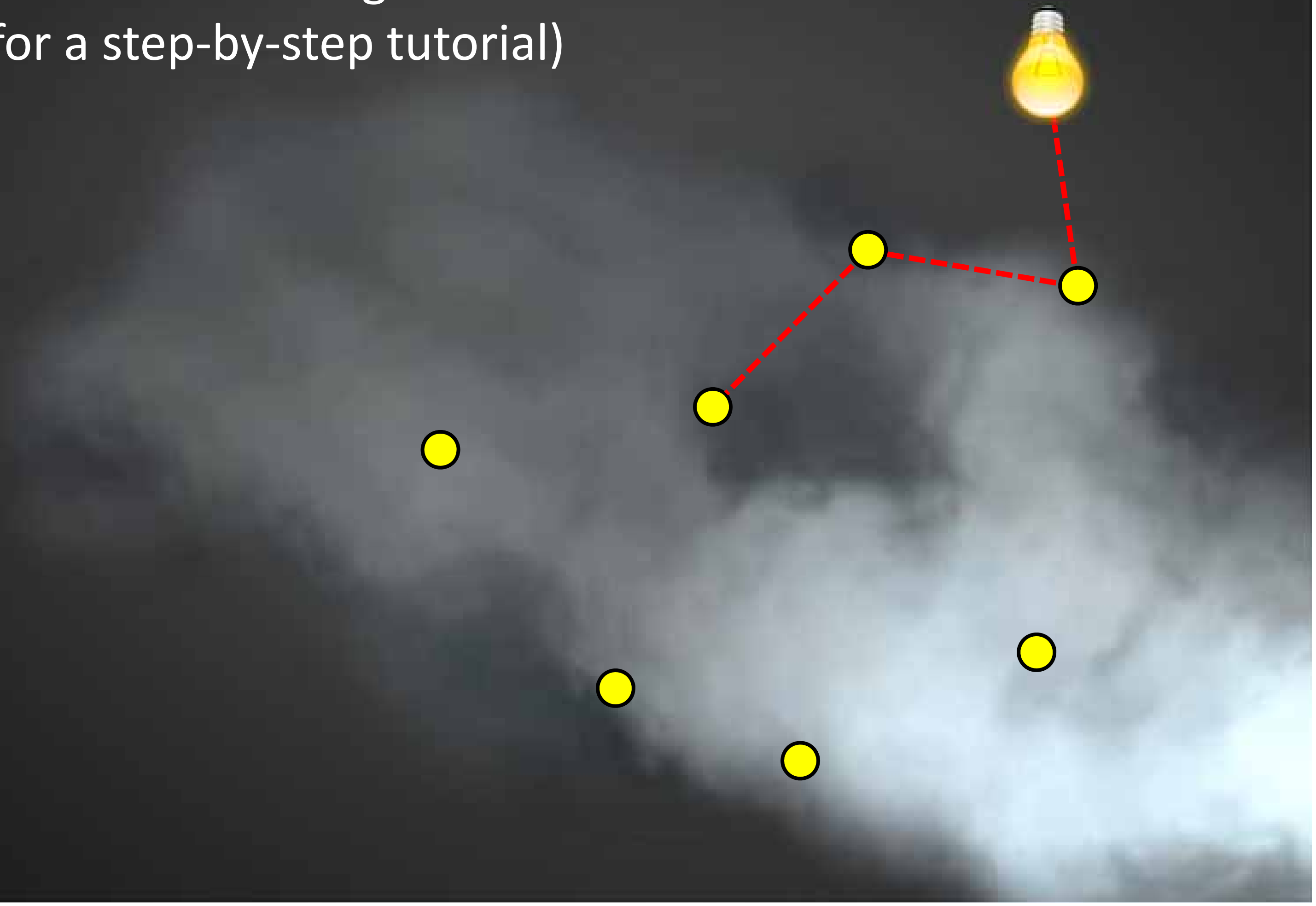
screen-space
bias compensation
(3 steps)

GPU ~ 550 ms
(ATI Radeon HD 5870)

Participating Media with Many-Lights

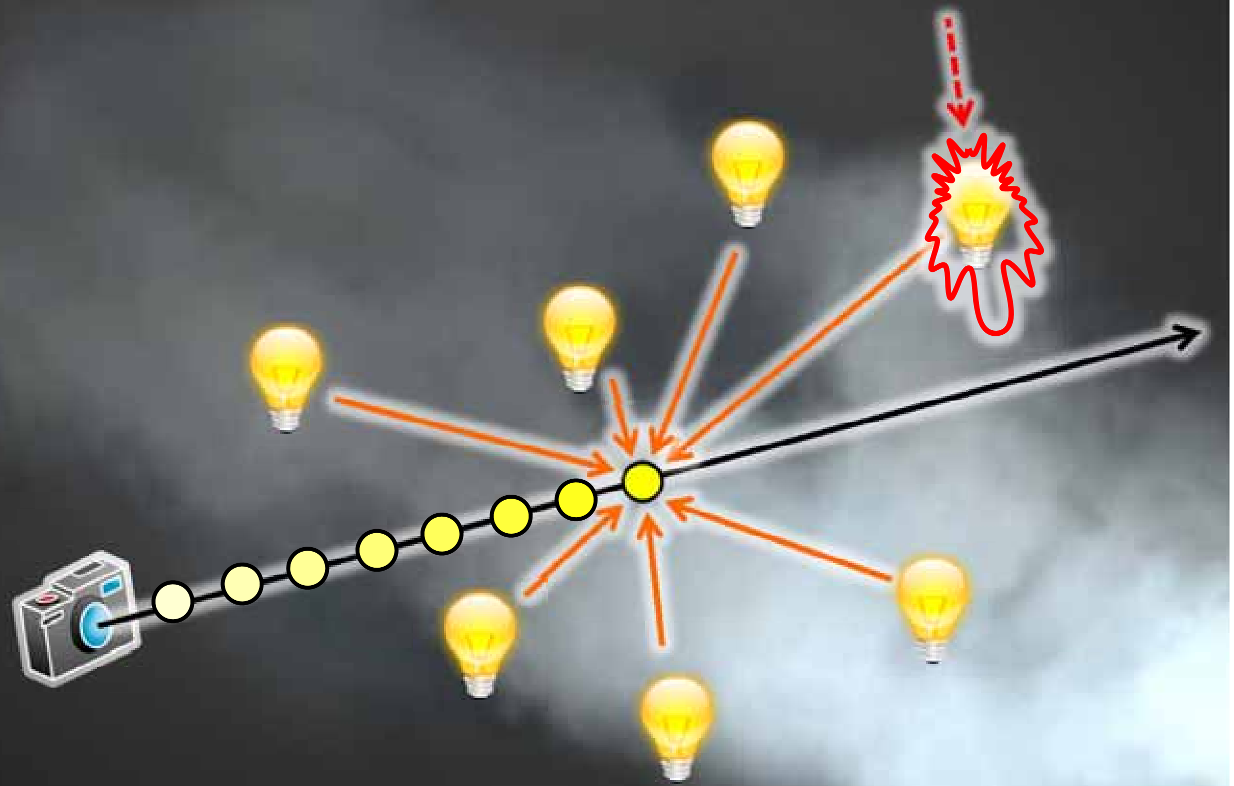
Light Transport in Participating Media

- direct light from surface VPLs and
- single-scattering from media VPLs (emit according to phase function)
- VPLs also generated at scattering events in media
  (see [ENSD12] for a step-by-step tutorial)



Light Transport in Participating Media

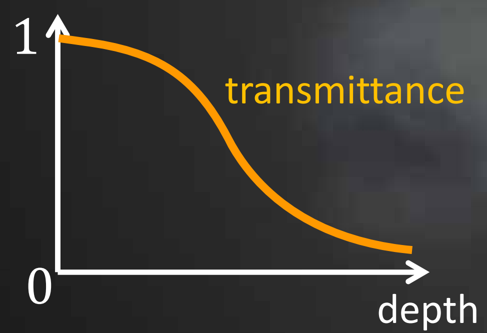
- direct light from surface VPLs and
- single-scattering from media VPLs (emit according to phase function)
- VPLs also generated at scattering events in media
  (see [ENSD12] for a step-by-step tutorial)

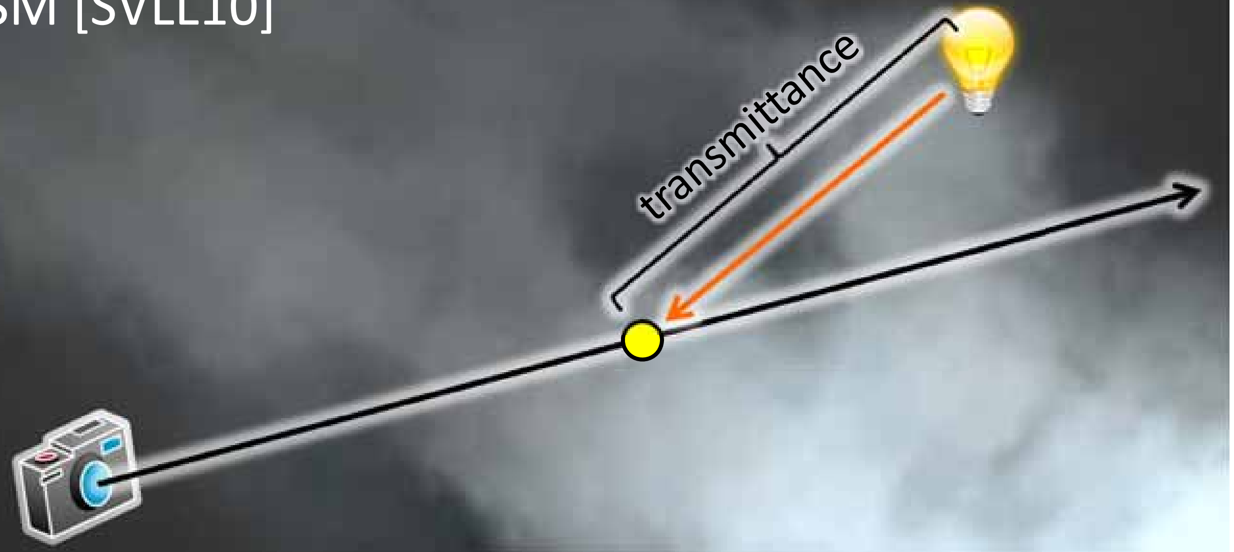


Participating Media with Many-Lights

Visibility and Transmittance

- homogeneous media:
  - standard shadow map per VPL (compute transmittance)
- heterogeneous media:
  - shadow map plus ray marching or
  - deep shadow maps [LV00] or
  - adaptive volumetric SM [SVLL10]





Rendering Strategies for Participating Media

Light Transport in Participating Media

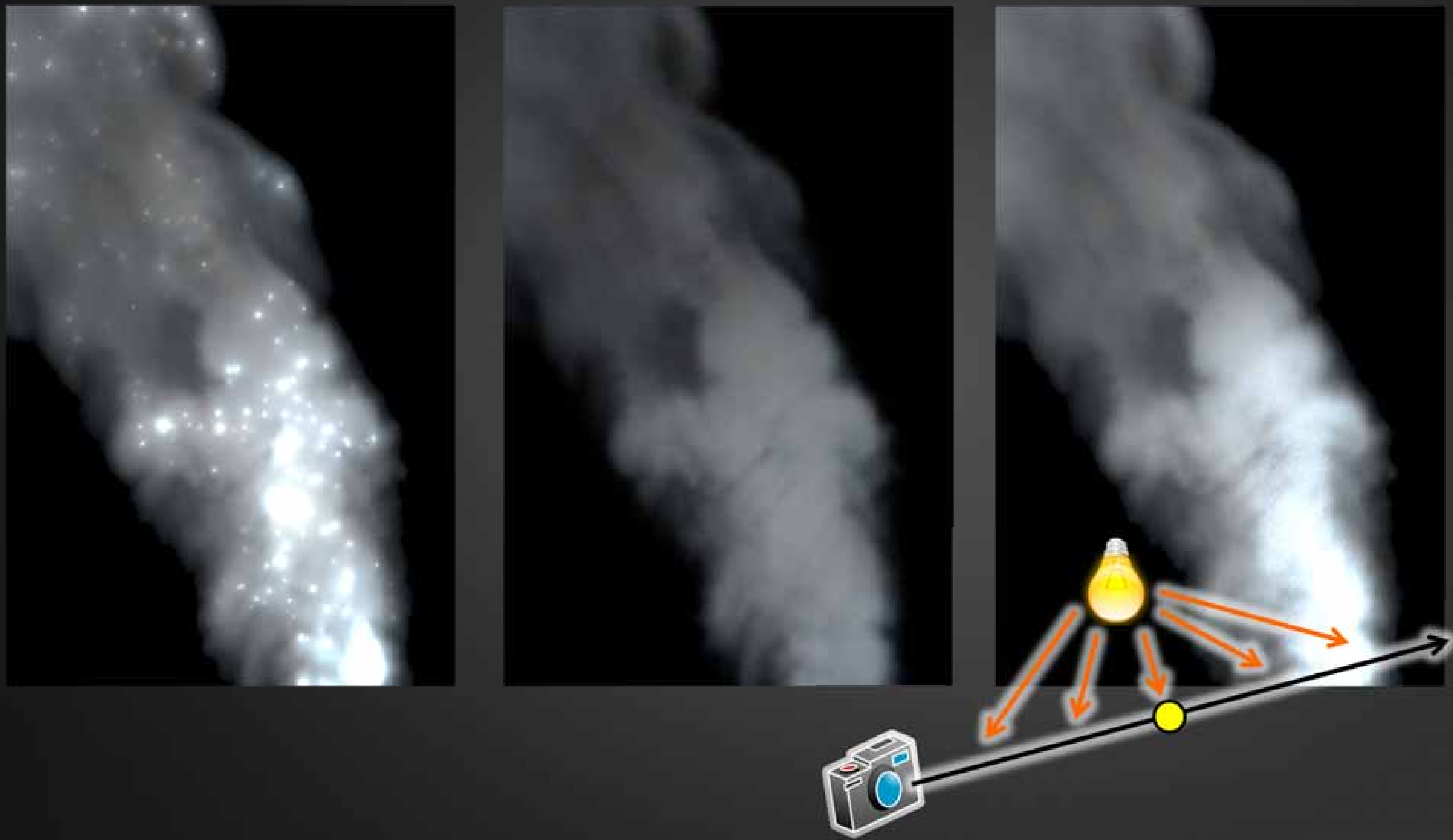
- direct light from surface VPLs and
- single-scattering from media VPLs (emit according to phase function)
- increased cost for visibility/transmittance computation

- observations to speed up bias compensation
  - how many compensation steps
  - heterogeneity vs. homogeneity
  - assumptions on visibility
  - **approximate bias compensation** without ray casting!

Participating Media with Many-Lights

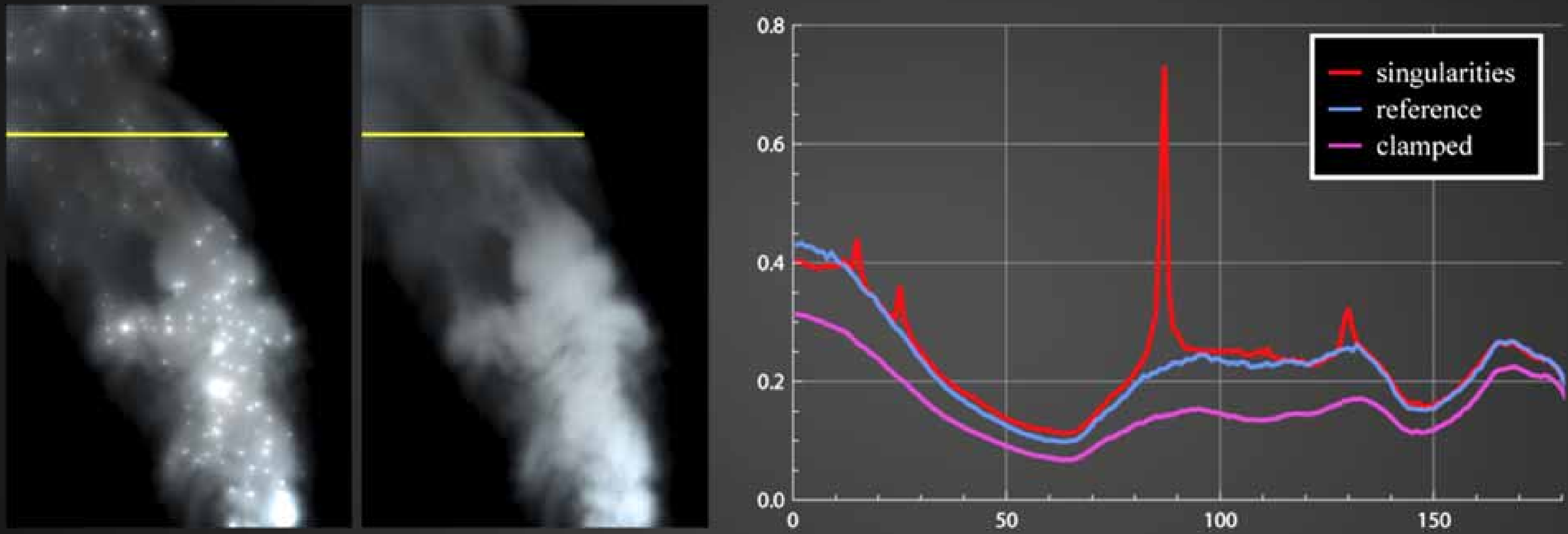
Bias Compensation

no clamping | clamping | (approximate) bias compensation



Participating Media with Many-Lights

Bias Compensation

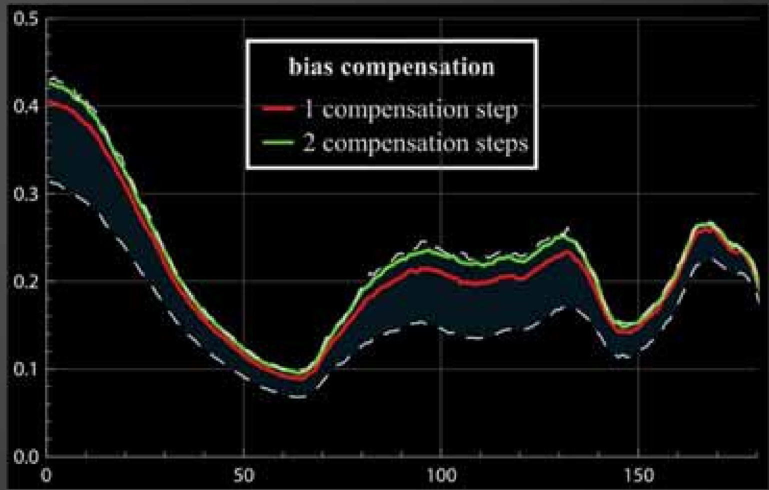


Participating Media with Many-Lights

Bias Compensation

- classic bias compensation [RSK08] if prohibitively expensive
- similar to surface case: magnitudes of compensation steps drop quickly

computed with path tracing (Raab et al.'s method)

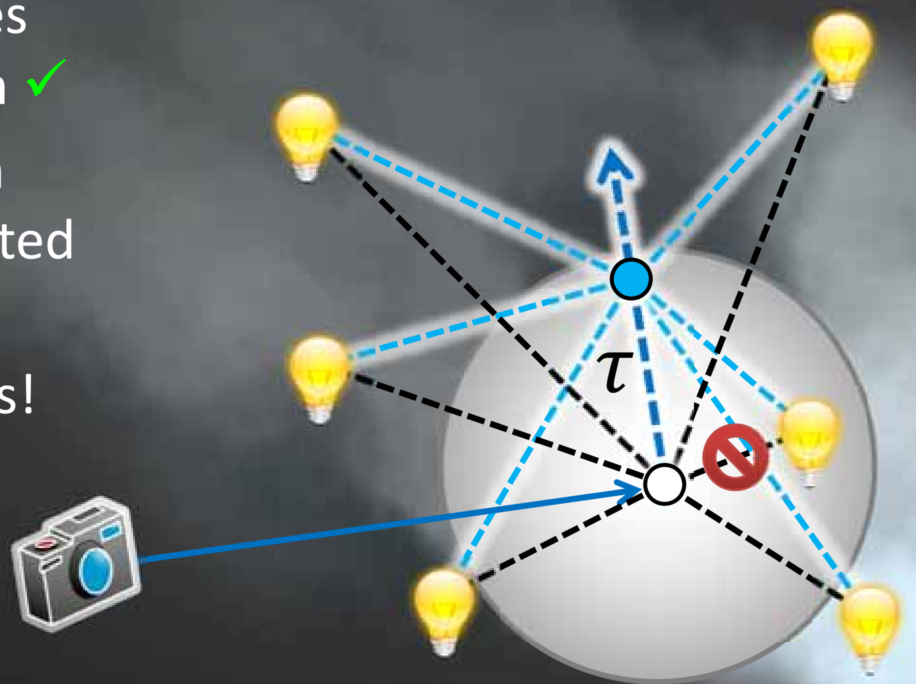


Participating Media with Many-Lights

Path Vertex Generation

- goal: create new path vertices **inside bounding region**
- heterogeneous media: Woodcock tracing (rejection sampling) might create vertices that have to be omitted

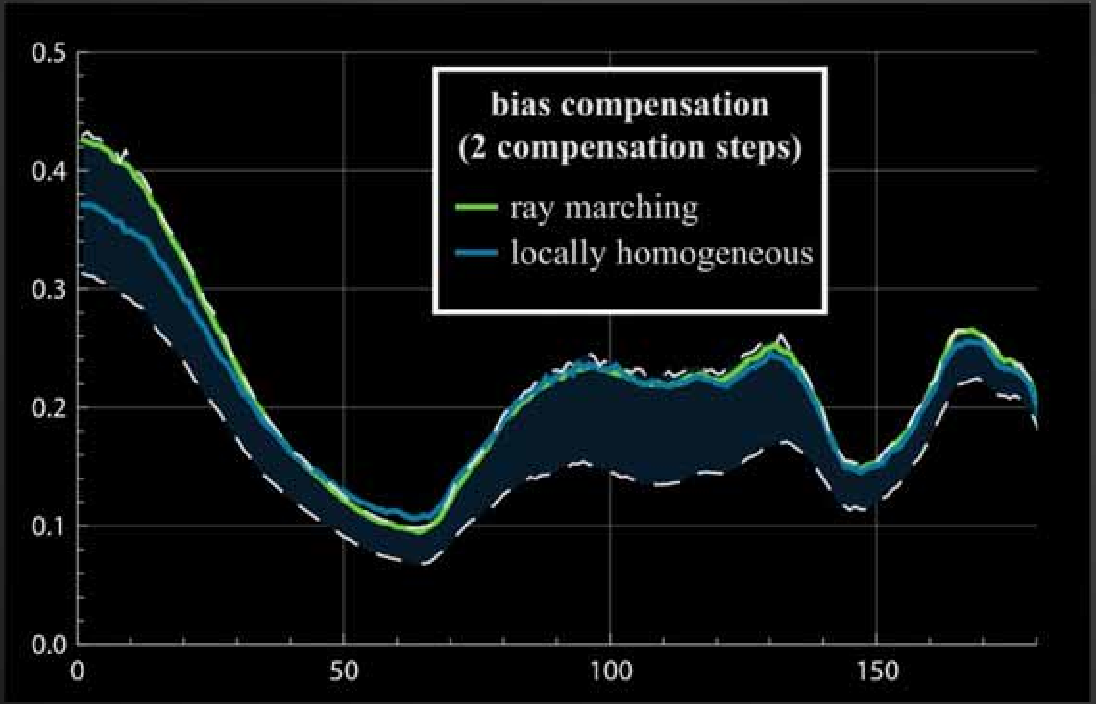
- assume locally homogeneous media
  (= similar scattering properties in some proximity)
  - simple to create vertices only in bounding region ✓
  - result still correct when transmittance τ computed with ray marching
  - see [ENSD12] for details!



Participating Media with Many-Lights

Path Vertex Generation

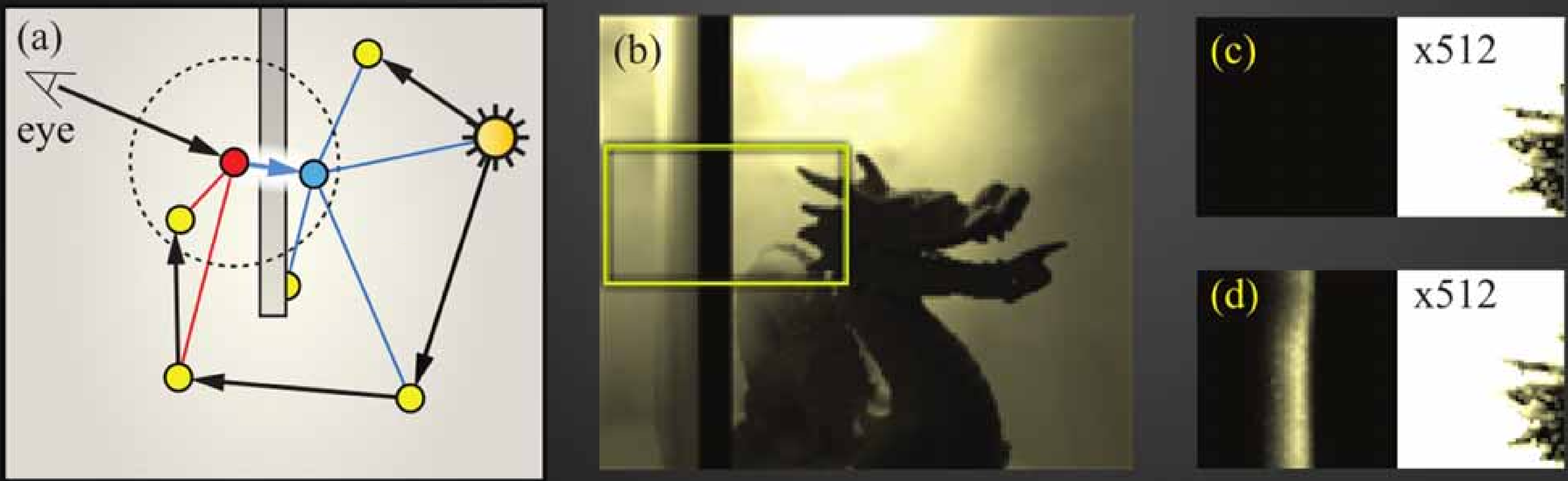
- assume media to be locally homogeneous
  - simple to create vertices only in bounding region ✓
- also compute transmittance using averaged scattering coefficients
  - not correct but very close



Participating Media with Many-Lights

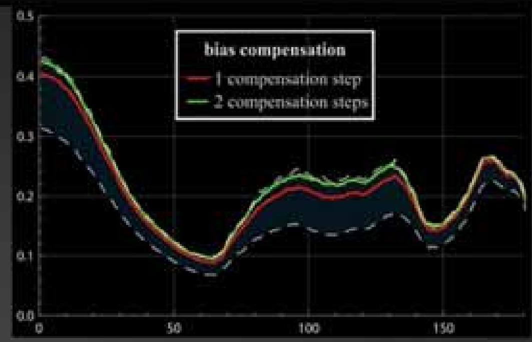
Do we have to compute visibility to newly created vertices?

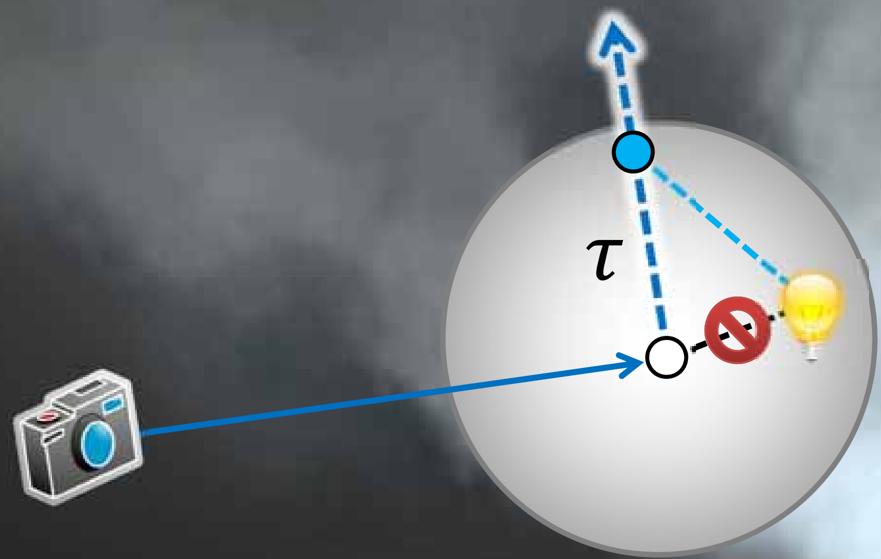
- new vertices are close to vertices requiring compensation
- what happens if we do not test mutual visibility?
- we tried to produce artifacts
  - vertices must be very close to a thin opaque object
  - medium must be thin (otherwise sampling through object unlikely)
  - quadratic decrease of compensation term



Approximate Bias Compensation

- VPL generation using ray casting
- two compensation steps only
- locally-homogeneous assumption
  - for creating new vertices without rejection
  - for computing transmittance to new vertices
- only transmittance τ but no visibility to new vertices
- more details in the paper [ENSD12]





Approximate Bias Compensation

Conclusions

Famous Last Words…

- many-lights methods work quite well in real-time
  - bias compensation is feasible for surfaces and media
  - glossiness for surfaces ↔ anisotropic phase functions for media
  - for mostly diffuse scenes, for scenes with moderate anisotropic media

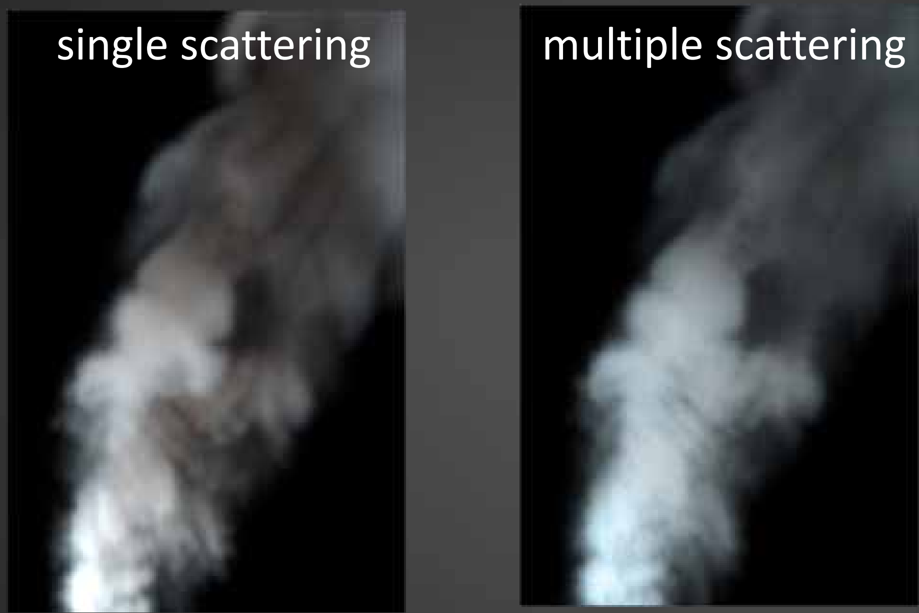
isotropic

moderate anisotropic

strong anisotropic

Conclusions

- … about participating media and multiple scattering (MS)
  - MS does not really add new visual details (single scattering does)
  - but MS contributes a lot to the total energy (clamping is no option)



- and finally: it's all about visibility computation
  - rasterization to resolve from-point visibility (VPL generation and use)
  - rasterization for screen space integration

Optimizing Realistic Rendering with Many-Light Methods

Real-Time Many-Light Rendering

Acknowledgements:

Some slides on SSBC have been created by Jan Novak. Tobias Ritschel provided images for ISM/MR.

References

- [LV00] Lokovic and Veach, Deep Shadow Maps, SIGGRAPH 2000
- [KK04] Kollig and Keller, Illumination in the Presence of Weak Singularities, 2004
- [DS05] Dachsbacher and Stamminger, Reflective Shadow Maps, I3D 2005
- [Seg06] Segovia et al., Non-interleaved Deferred Shading of Interleaved Sample Patterns, GH 2006
- [DS06] Dachsbacher and Stamminger, Splatting of Indirect Illumination, I3D 2006
- [LSKLA07] Laine et al., Incremental Instant Radiosity for Real-Time Indirect Illumination, EGSR 2007
- [RSK08] Raab et al., Unbiased global illumination with participating media, Monte Carlo and Quasi-Monte Carlo Methods, 2008
- [RGKSDK09] Ritschel et al., Imperfect Shadow Maps for Efficient Computation of Indirect Illumination, SIGGRAPH Asia 2008
- [SW09] Segovia and Wald, Screen Space Spherical Harmonics Filters for Instant Global Illumination, TechReport Intel, 2009
- [ED10] Engelhardt and Dachsbacher, Epipolar Sampling for Shadows and Crepuscular Rays in Participating Media with Single Scattering, I3D 2010
- [REGSKD10] Ritschel et al., Micro-Rendering for Scalable, Parallel Final Gathering, SIGGRAPH Asia 2009
- [SS10] Schwarz, Seidel, Fast Parallel Surface and Solid Voxelization on GPUs, SIGGRAPH Asia 2010
- [NW10] Nichols and Wyman, Interactive Indirect Illumination Using Adaptive Multiresolution Splatting, IEEE Transactions on Visualization and Computer Graphics 16(5), 2010
- [SVLL10] Salvi et al., Adaptive Volumetric Shadow Maps, EGSR 2010
- [NED11] Novak et al., Screen-Space Bias Compensation for Interactive High-Quality Global Illumination with Virtual Point Lights, I3D 2011
- [NNDJ12a] Novak et al., Virtual Ray Lights for Rendering Scenes with Participating Media, SIGGRAPH 2012
- [NNDJ12b] Novak et al., Progressive Virtual Beam Lights, EGSR 2012
- [ENSD12] Engelhardt et al., Approximate Bias Compensation for Rendering Scenes with Heterogeneous Participating Media, Pacific Graphics 2012
- [OBA12] Olsson et al., Clustered Deferred and Forward Shading, High Performance Graphics 2012