한국컴퓨터그래픽스학회 2014

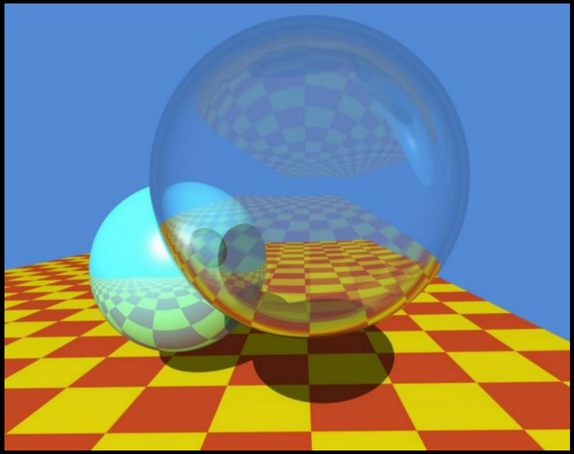
몬테카를로 광선 추적법을 위한 가속화 기술

(Acceleration Techniques for Monte Carlo Ray Tracing)

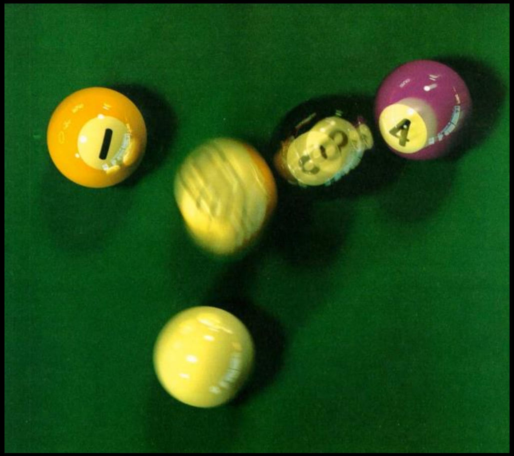
문보창 (Moon, Bochang)

KAIST

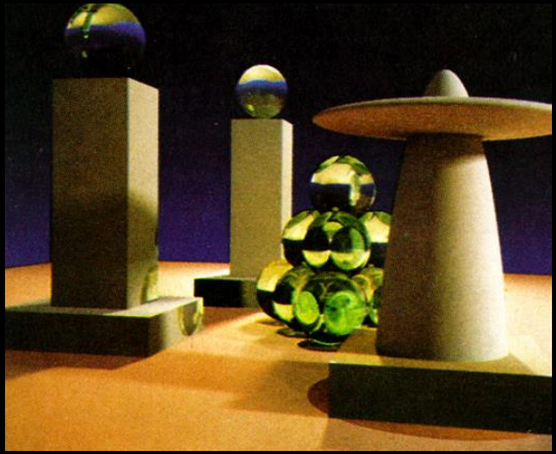
Ray Tracing (광선 추적법)



Ray tracing
[Whitted 1980]



Distributed ray tracing
[Cook 1984]



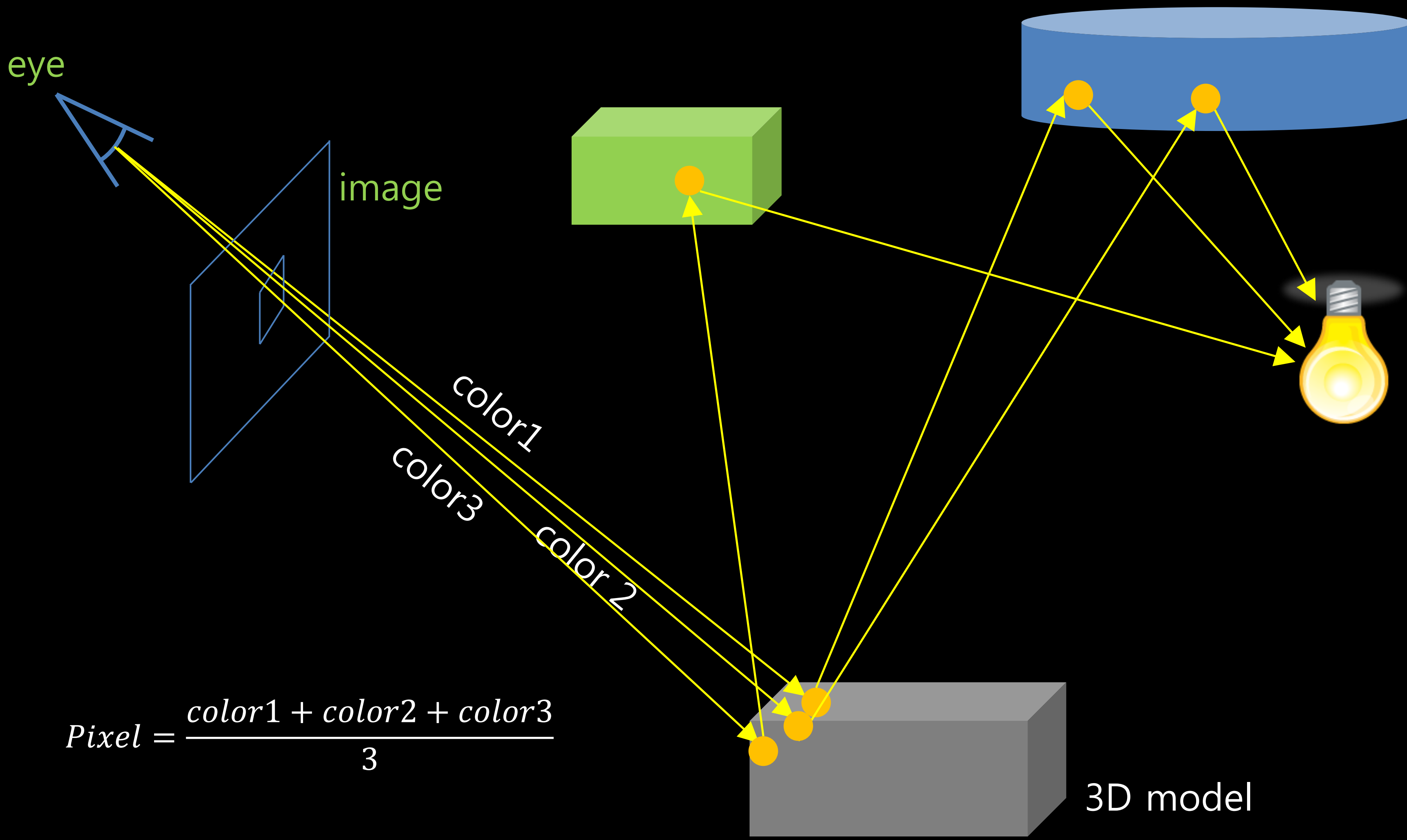
Path tracing
[Kajiya 1986]

Monte Carlo ray tracing

Quake 4: Ray Traced [www.q4rt.de]

Pixar's Cars, 2006

Monte Carlo Ray Tracing



$$Pixel = \frac{color1 + color2 + color3}{3}$$


Playback of captured images

Challenges

104 M triangles (12.8 GB)

Rendering time: 2 days

3 GHz processor
4 GB main memory
(2009 년 당시 최고급)

Challenges

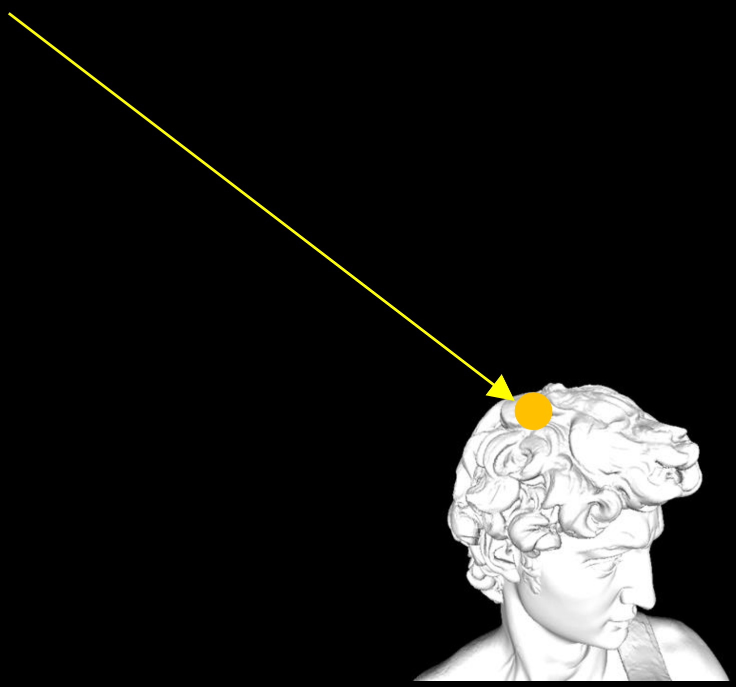
Short animation (200 frames)

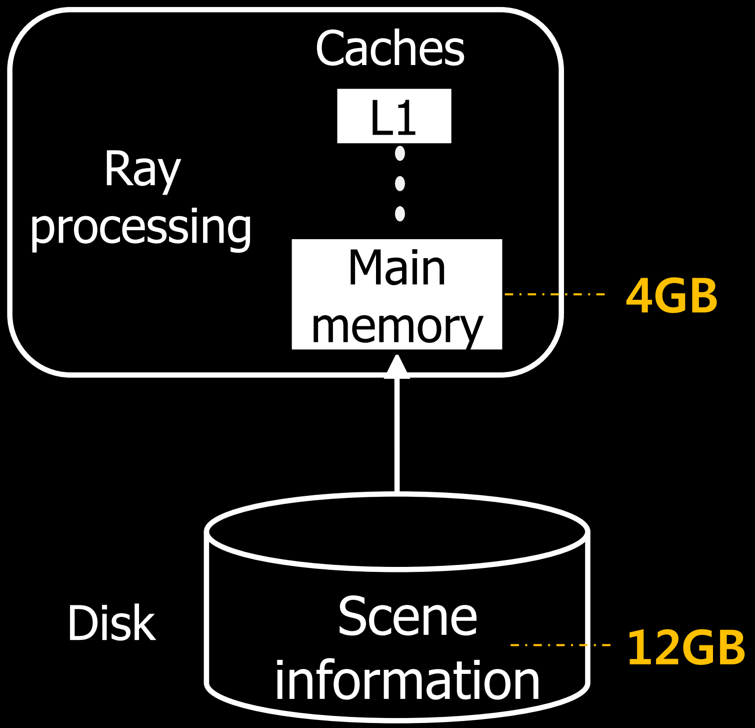
Estimated time

= 2 days x 200 frames
= 400 days

Challenges

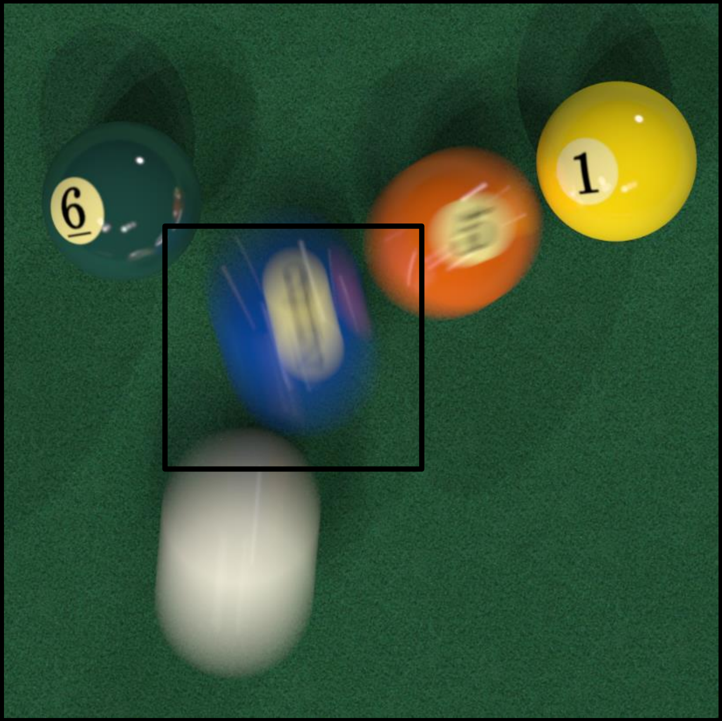
1) Computation time of processing a ray can be large





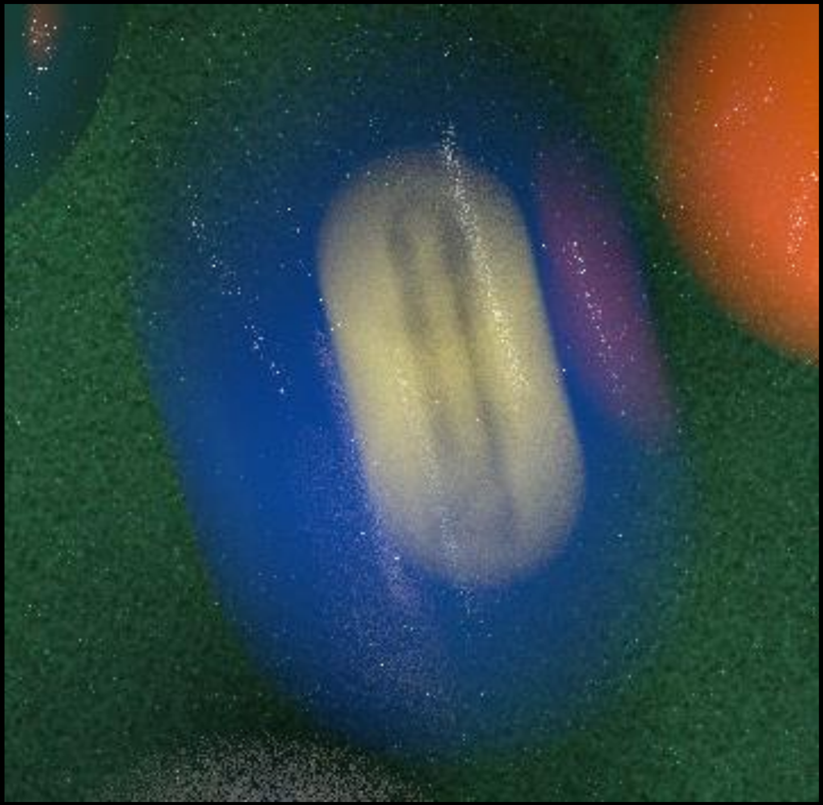
Challenges

1) Computation time of processing a ray can be large
2) A large number of rays should be traced



Challenges

1) Computation time of processing a ray can be large
2) A large number of rays should be traced



N = 53

N = 64,000

Goal

- **To accelerate Monte Carlo ray tracing**
  1) Achieve a high cache utilization
  2) Reduce the required number of ray samples

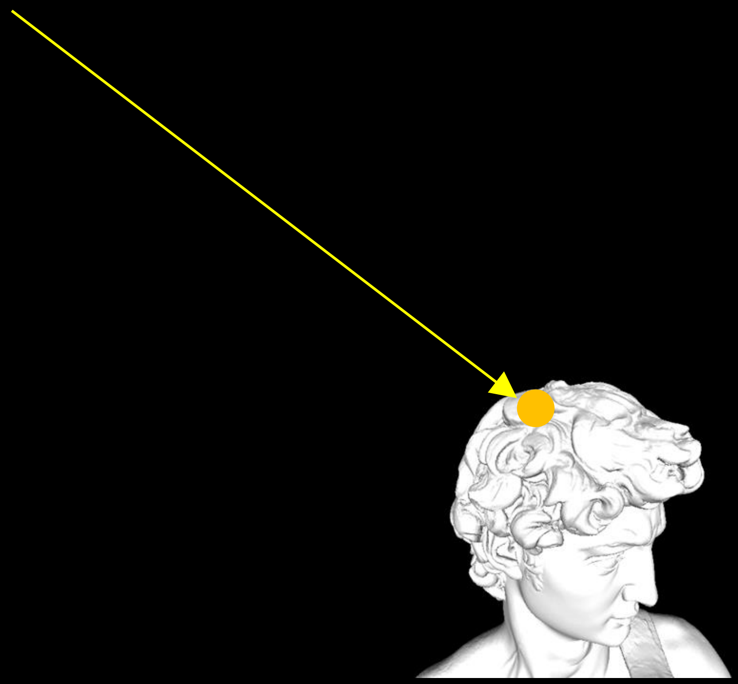
Outline

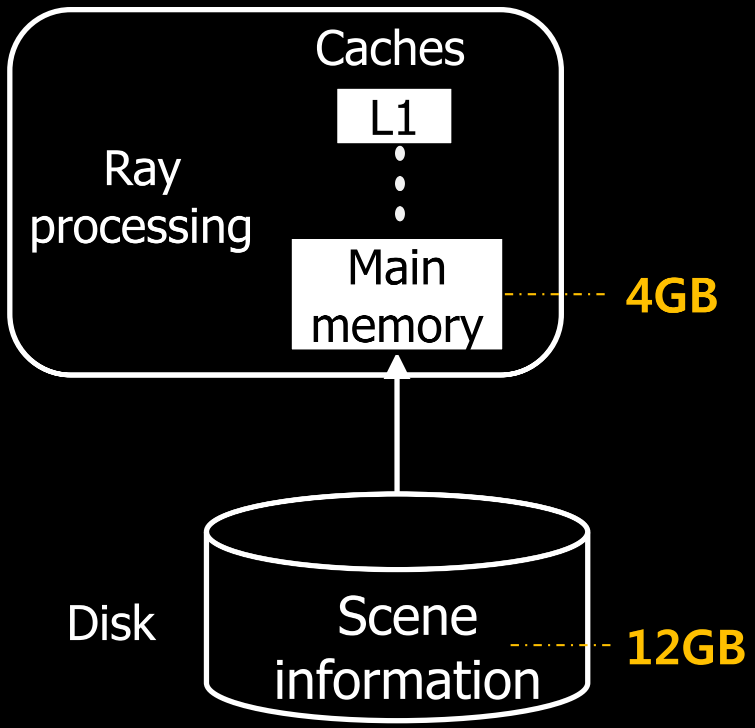
- **To accelerate Monte Carlo ray tracing**
  1) Achieve a high cache utilization
     - Cache-oblivious reordering (TOG)
  2) Reduce the required number of ray samples
     - Virtual flash image based filtering (CGF)
     - Local regression based adaptive rendering (TOG)

Cache-Oblivious Ray Reordering

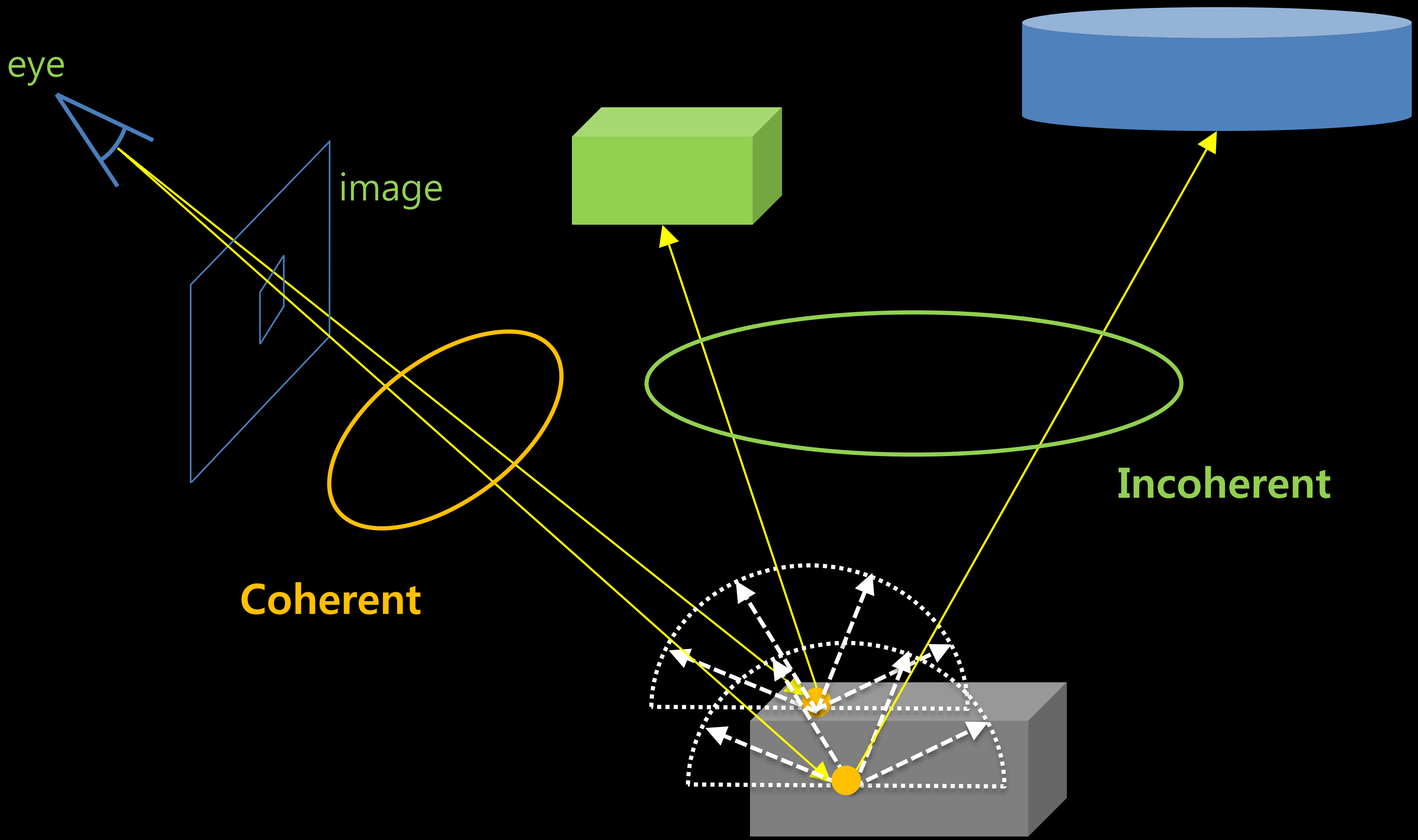
ACM Transactions on Graphics 2010
Presented at SIGGRAPH 2011

Incoherent Secondary Rays



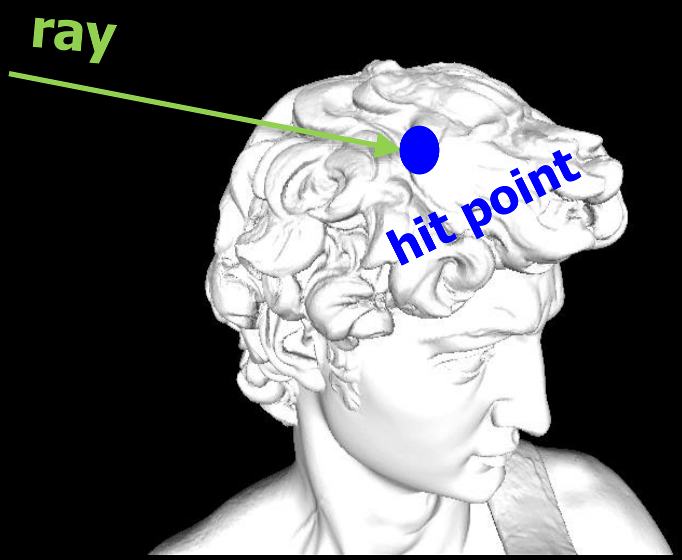


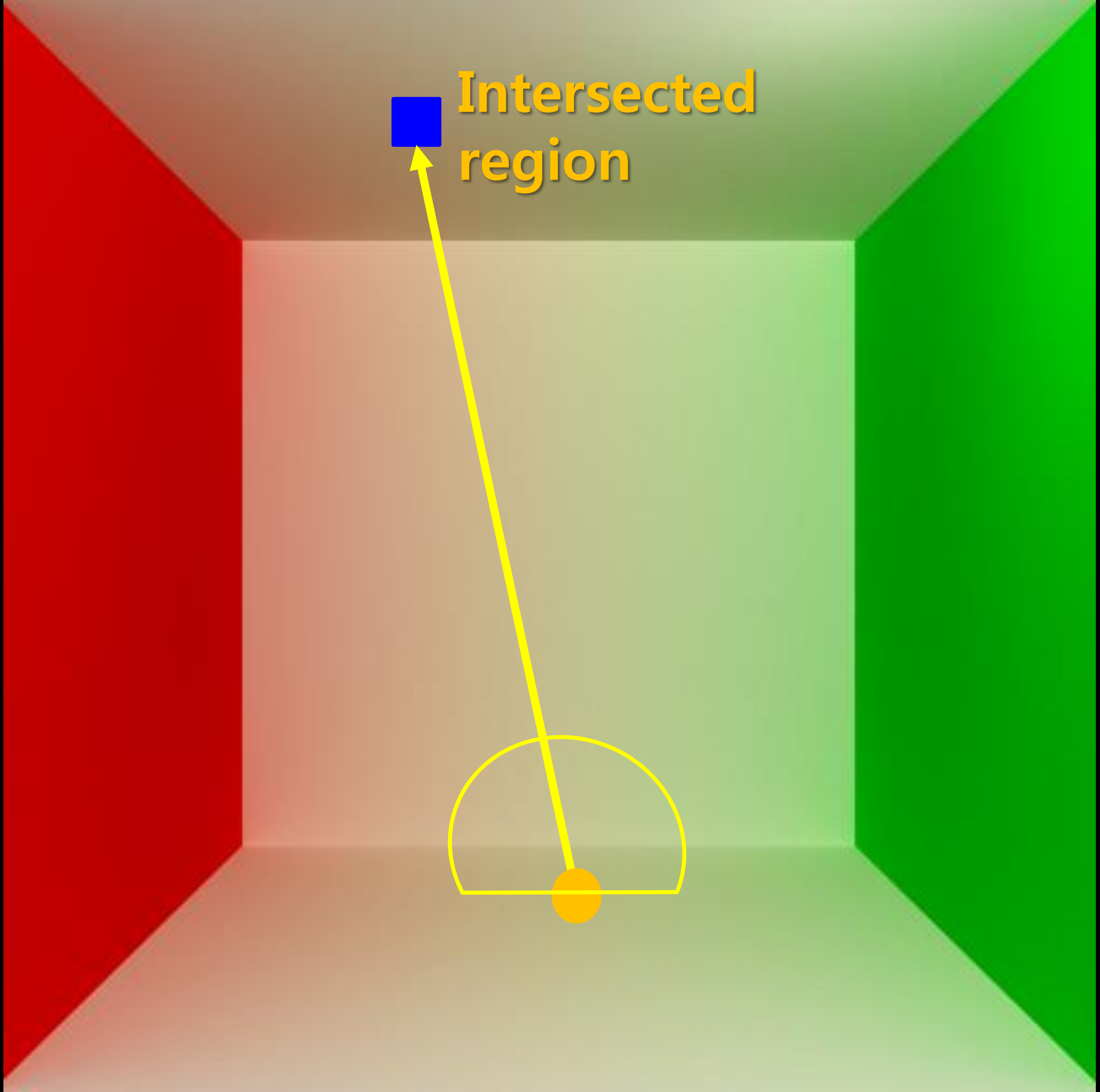
Incoherent Secondary Rays

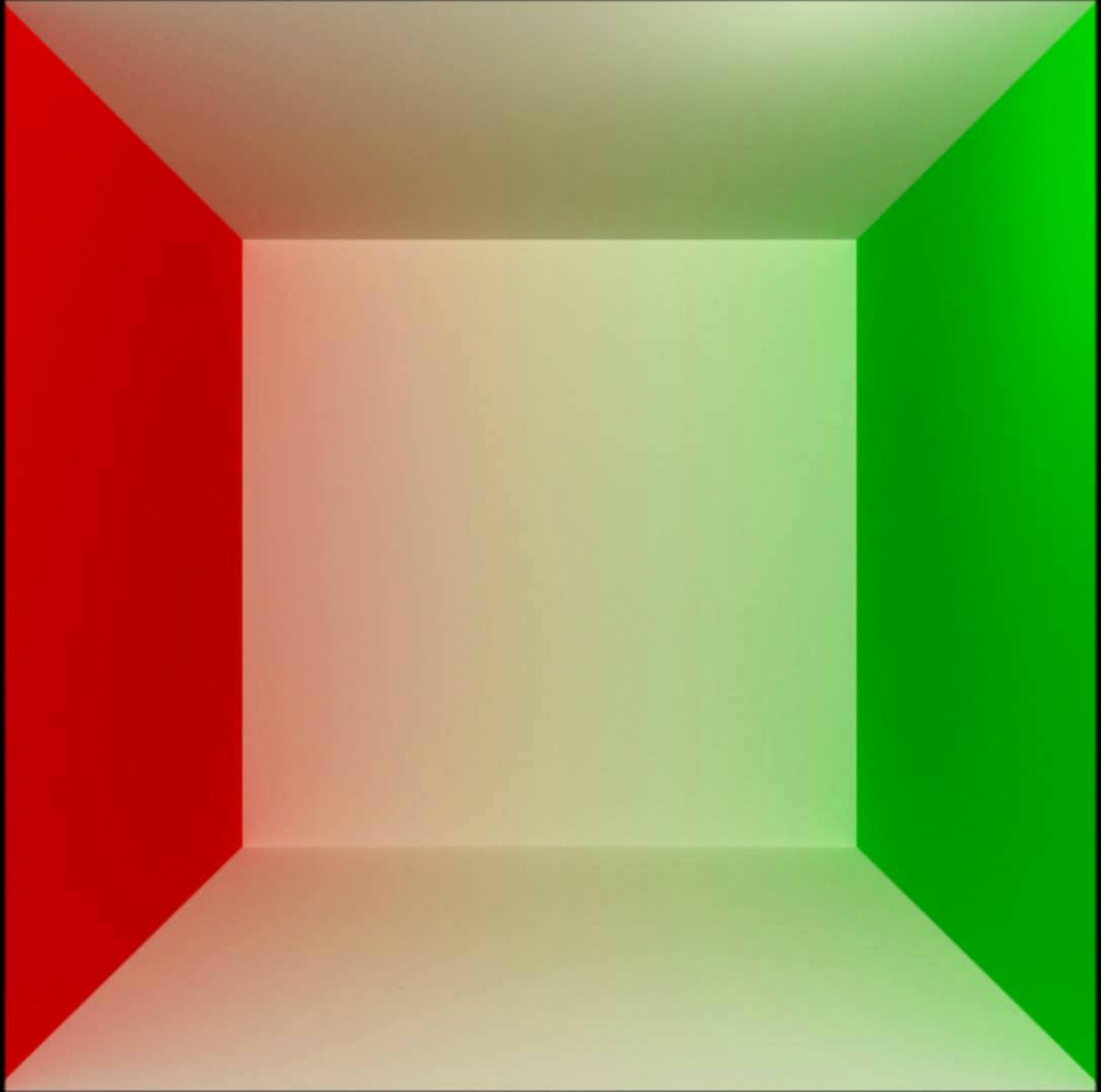


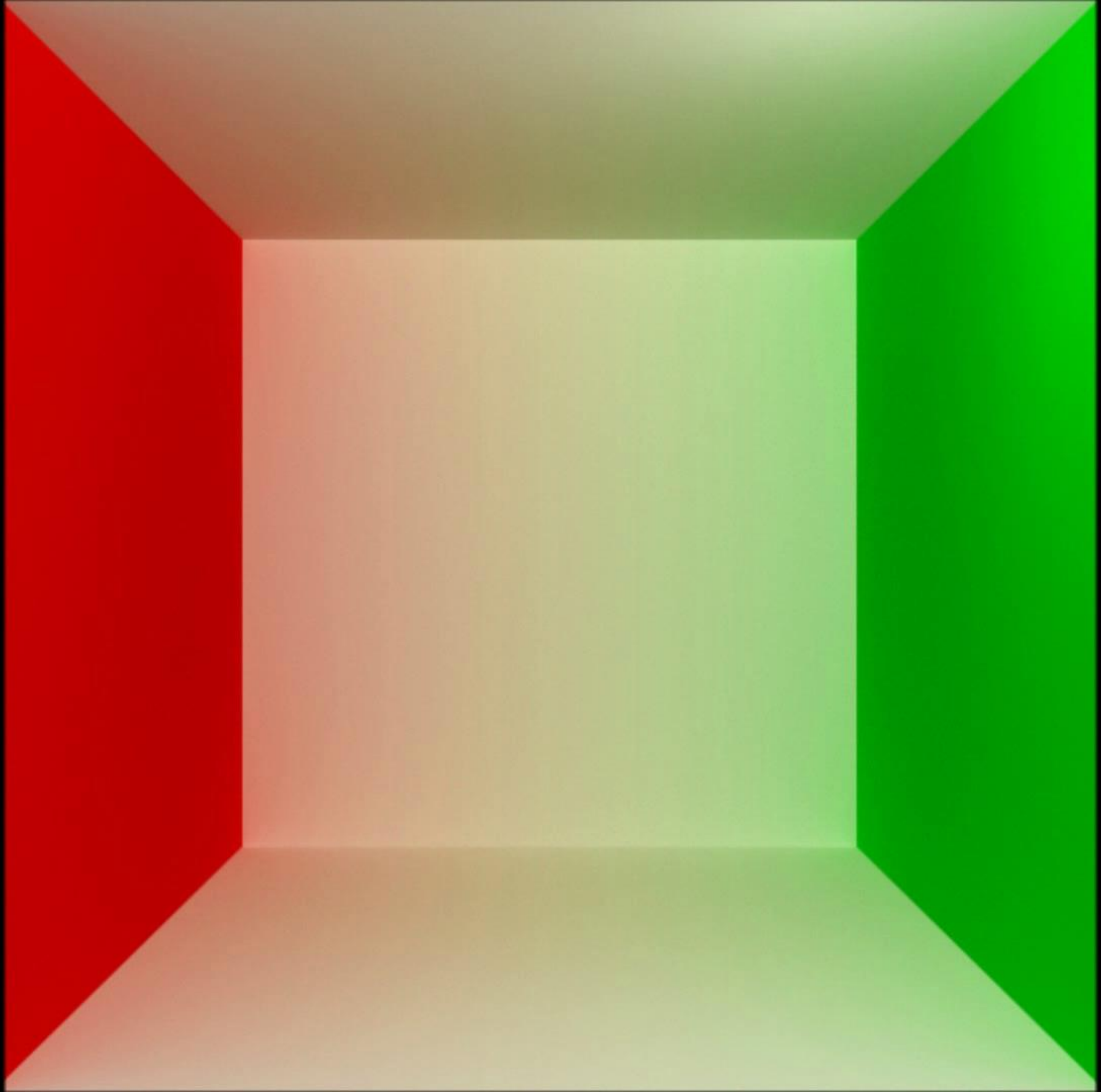
Contributions

- **Propose a novel *hit point heuristic* (HPH) to compute a coherent ordering of rays**

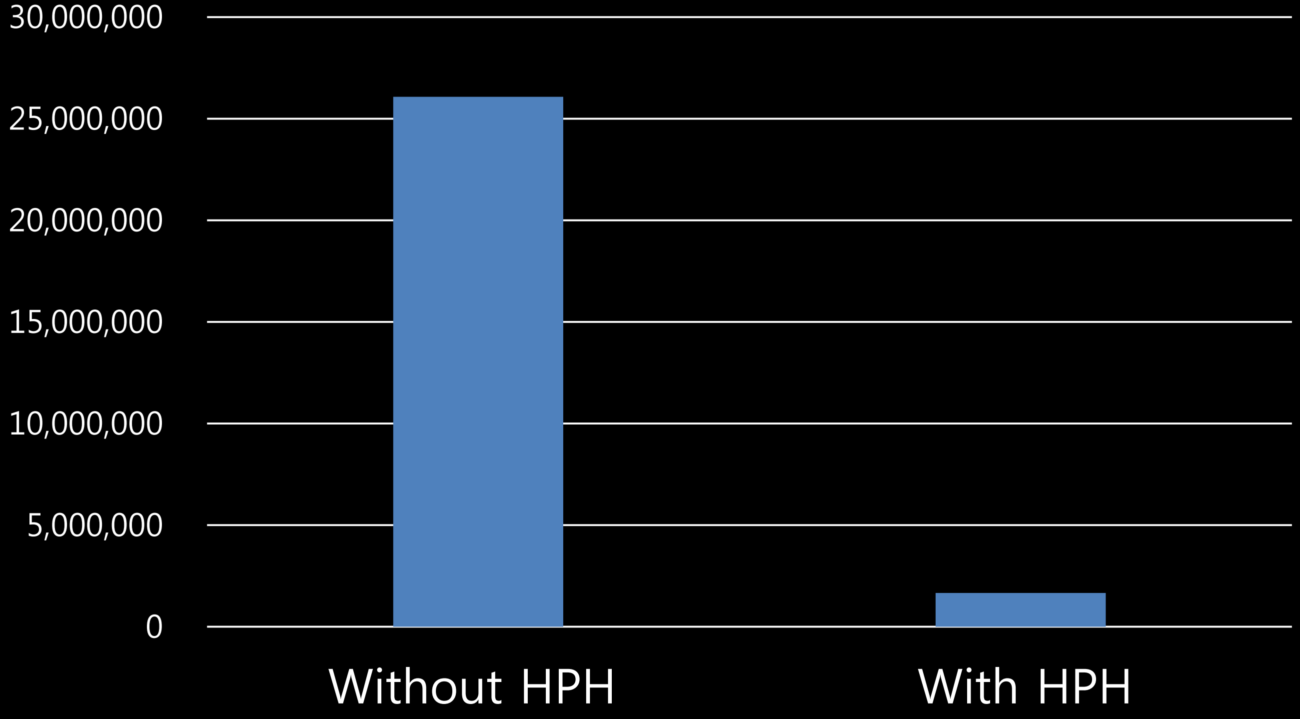



Intersected
region





Disk I/O


30,000,000
25,000,000
20,000,000
15,000,000
10,000,000
5,000,000
0
Without HPH
With HPH

Results

104 M triangles (12.8 GB)

- **Rendering time**
  - 2 days per frame
  - 3 hours with HPH
- **Animation**
  - 2 days x 200 frames = more than 1 year
  - 3 hours x 200 frames = 25 days

Outline

- **To accelerate Monte Carlo ray tracing**
  1) Achieve a high cache utilization
     - Cache-oblivious reordering (TOG)
  2) Reduce the required number of ray samples
     - Virtual flash image based filtering (CGF)
     - Local regression based adaptive rendering (TOG)

Robust Image Denoising using a Virtual Flash Image for Monte Carlo Ray Tracing

Computer Graphics Forum (2013)

Presented at EGSR 2014

Monte Carlo ray tracing result (N = 10,000)

Motivation

Straightforward approach

N = 64
6 minutes

N = 10,000
16 hours

Results

N = 64
6 minutes

Filtering

N = 64
6 minutes
Overhead (2 sec.)

Challenges

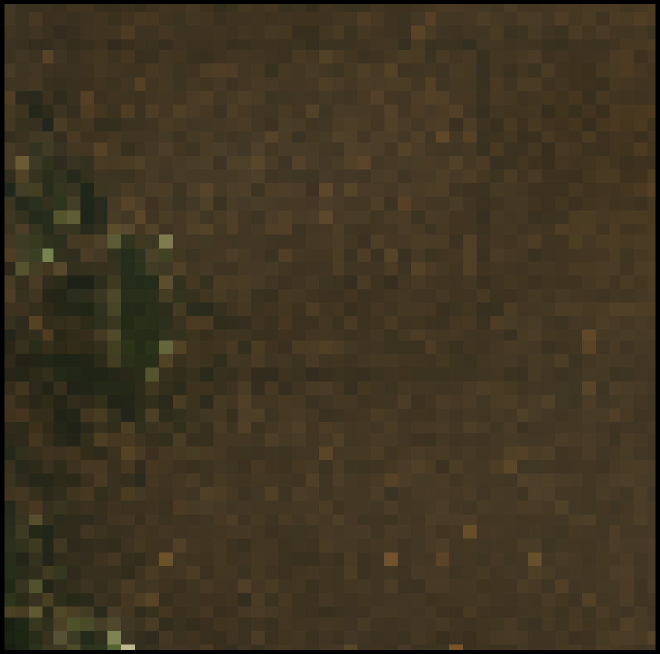
N = 64
6 minutes

Gaussian filter

Challenges

N = 64
6 minutes

Bilateral filter



Contributions

- **Propose a novel edge-stopping function,** ***virtual flash image***

Input image
N = 64

Virtual flash image

Output image

Flash Image

- **Motivated by flash photography [Petschingg 04, Eisemann 04]**

Input image

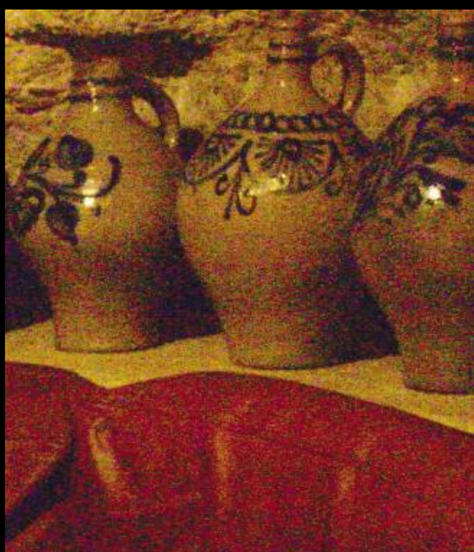
Flash Image

- **Motivated by flash photography [Petschingg 04, Eisemann 04]**

Flash image

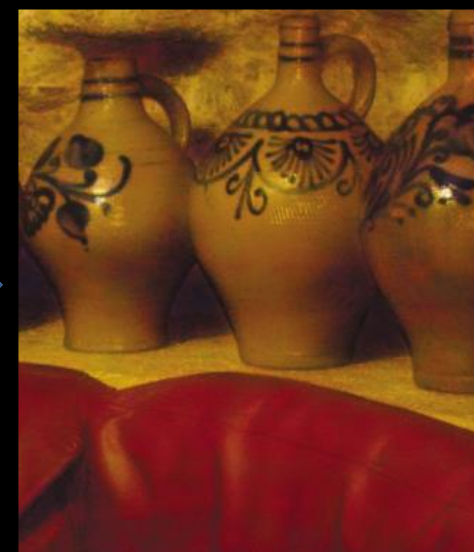
Flash Image

- **Motivated by flash photography [Petschingg 04, Eisemann 04]**



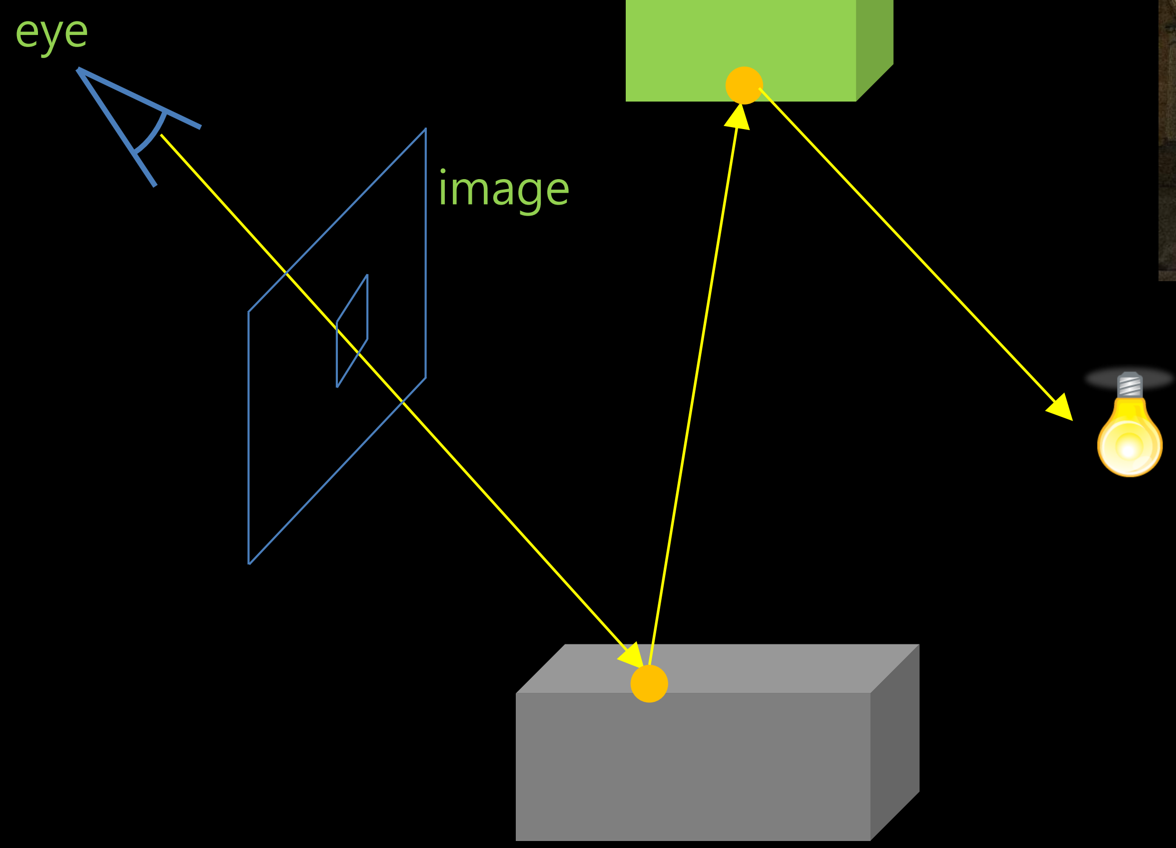
Input image

Flash image



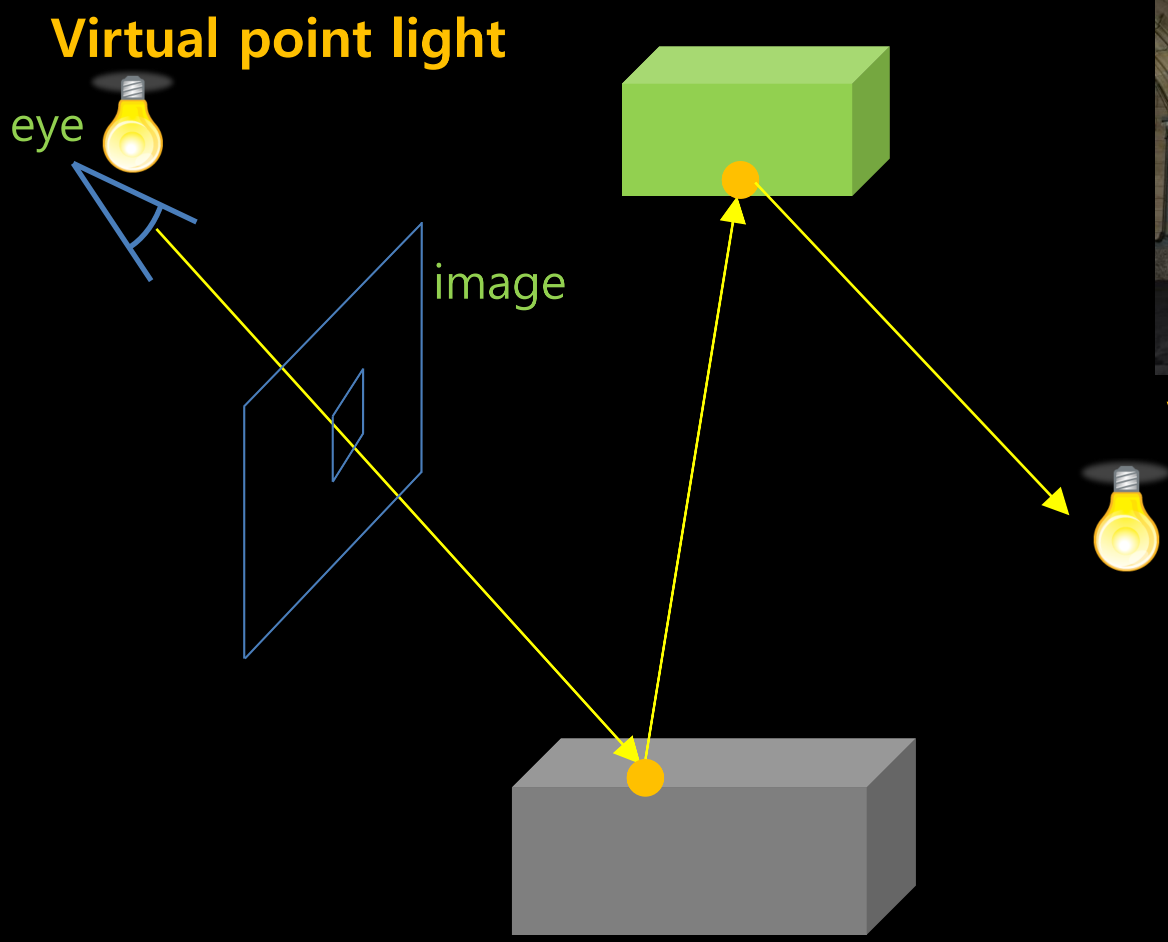
Result

Virtual Flash Image



Input image

Virtual Flash Image



Virtual flash image

Input image

Virtual flash image

Output image


Xu et al. 05
Ours

Outline

- **To accelerate Monte Carlo ray tracing**
  1) Achieve high cache utilization
     - Cache-oblivious reordering (TOG)
  2) Reduce the required number of ray samples
     - Virtual flash image based filtering (CGF)
     - Local regression based adaptive rendering (TOG)

Adaptive Rendering based on Weighted Local Regression

ACM Transaction on Graphics 2014

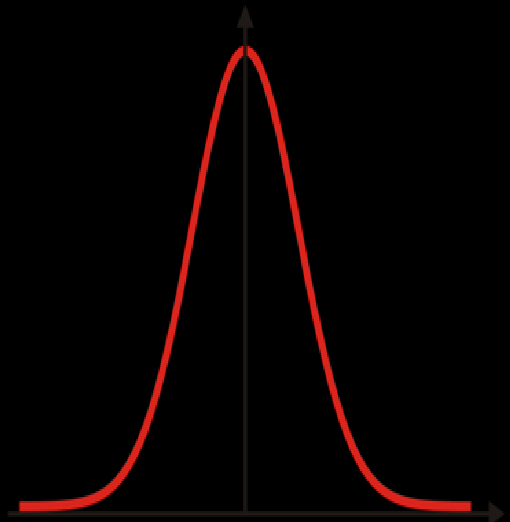
Will be presented as a SIGGRAPH talk 2014

(Will be presented at SIGGRAPH 2015)

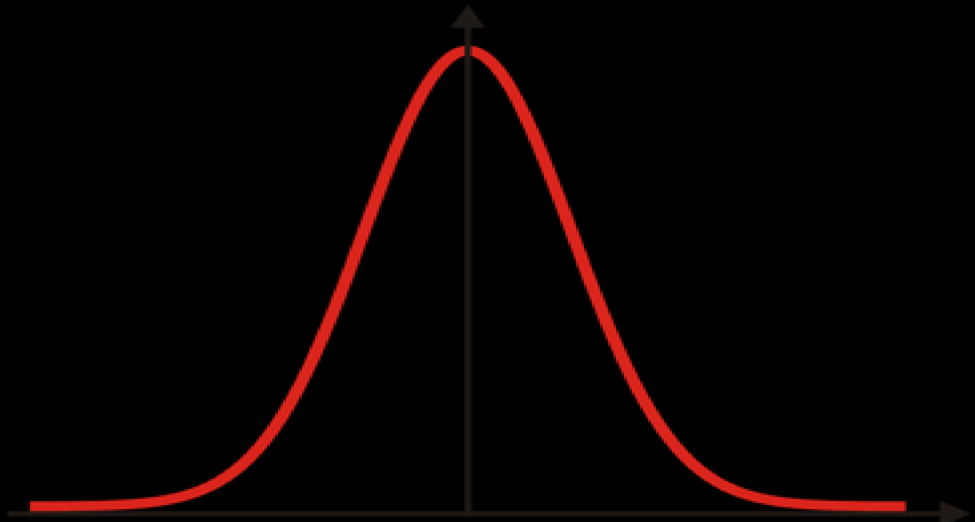
Image Filtering

Gaussian filter: $\hat{f}_h(x) = \frac{1}{2\pi h^2} e^{-\frac{||x - x_c||^2}{2h^2}}$

$h - filtering\ parameter\ (i.e., bandwidth)$

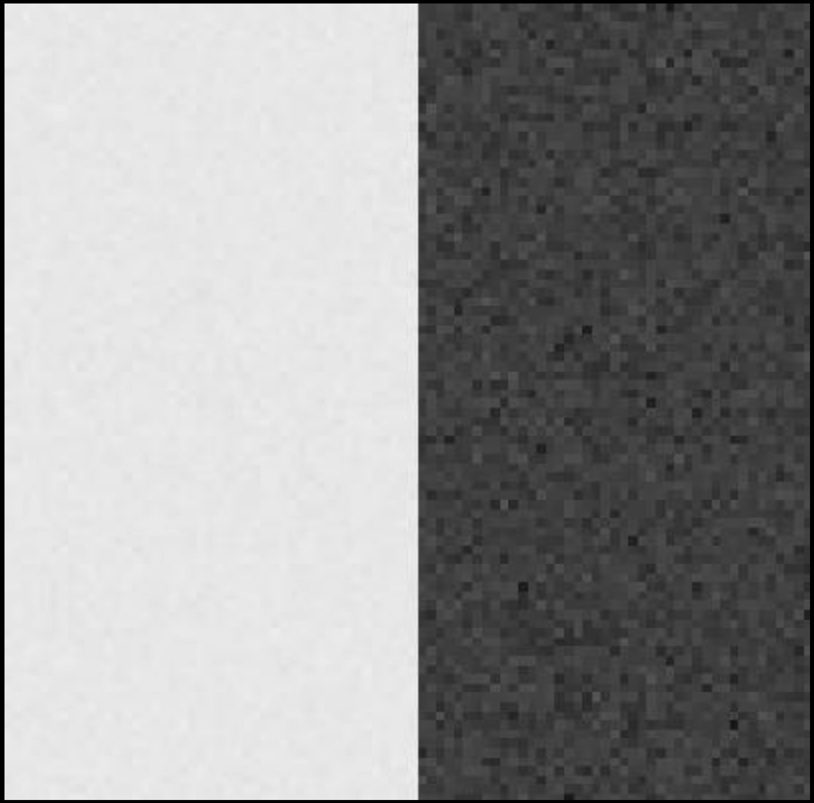


$small\ \ h$



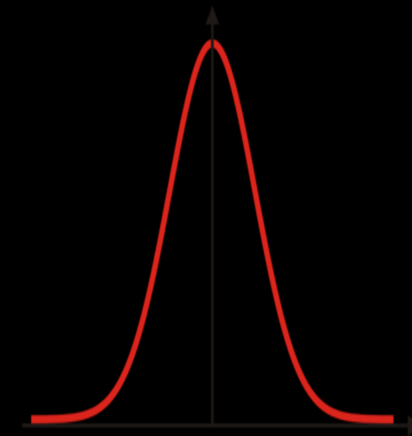
$large\ \ h$

Image Filtering

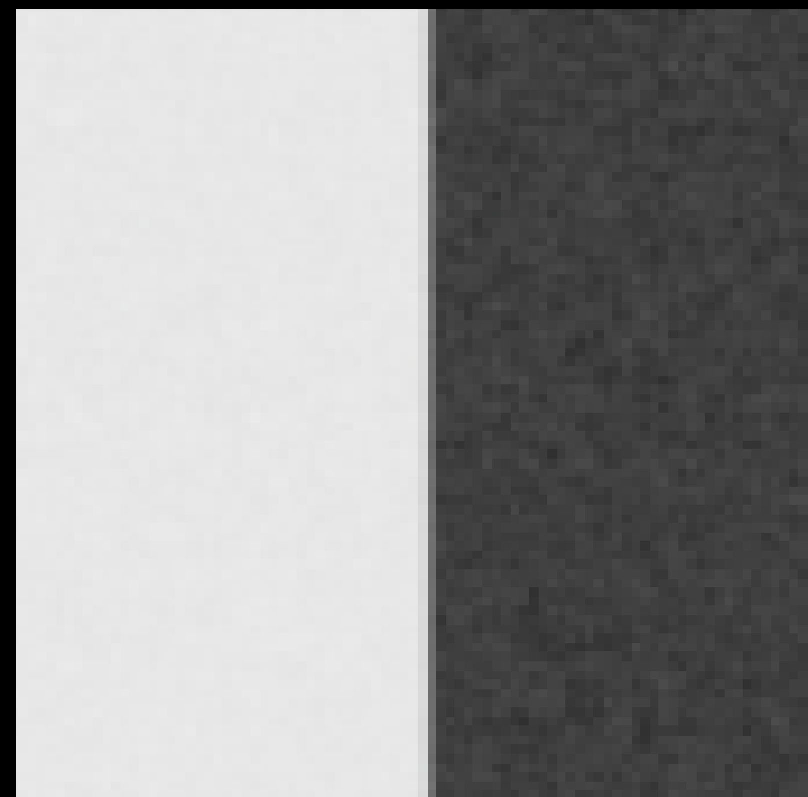


Input image

Image Filtering

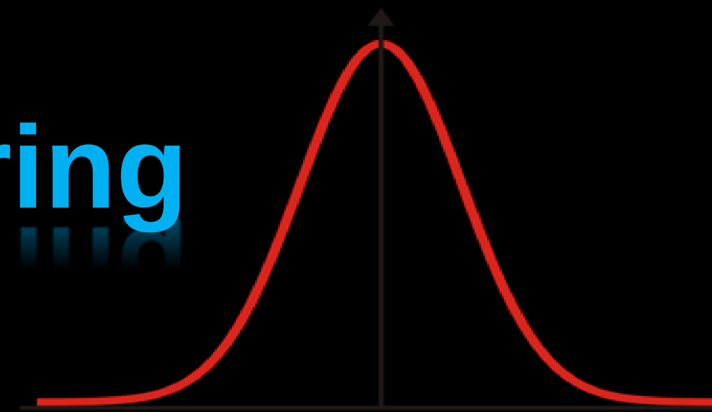


Edge regions	Smooth regions
✓	✗



result with small h

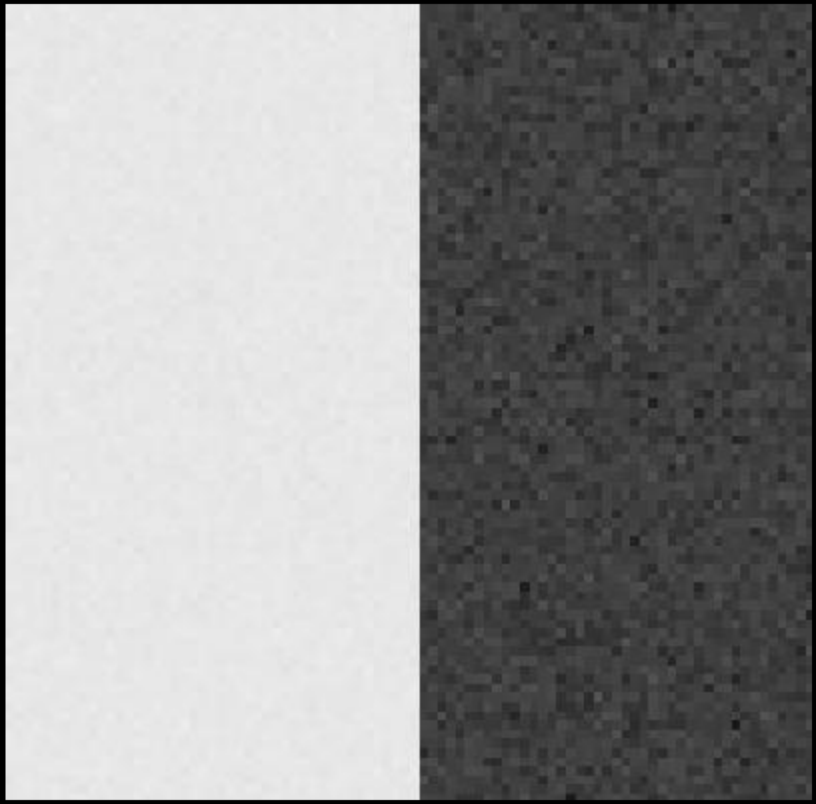
Image Filtering



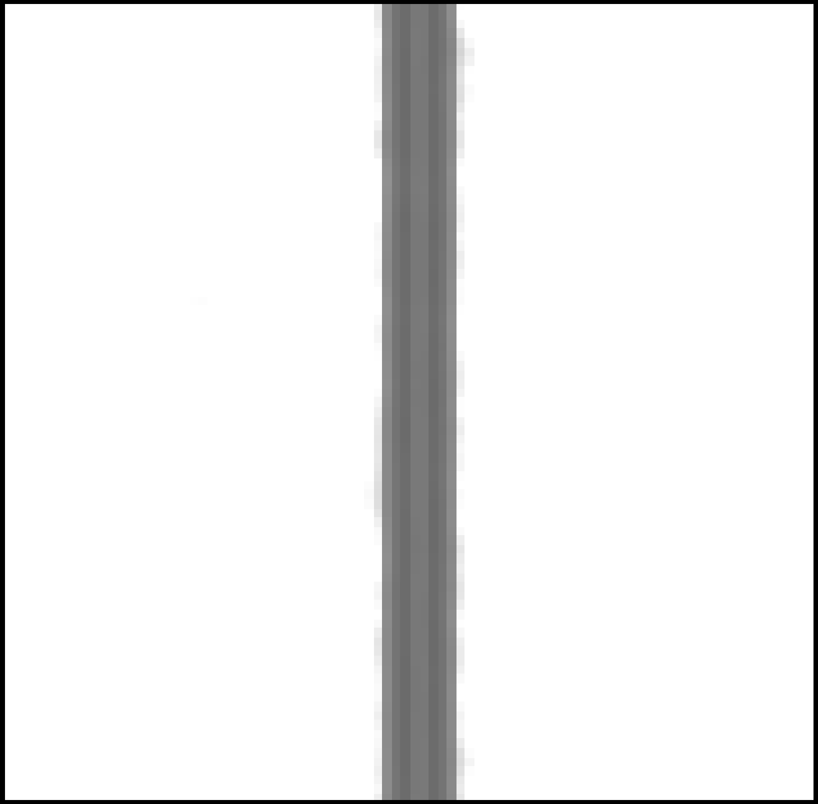
Edge regions	Smooth regions
	

result with large h

Our Approach (Adaptive Filtering)



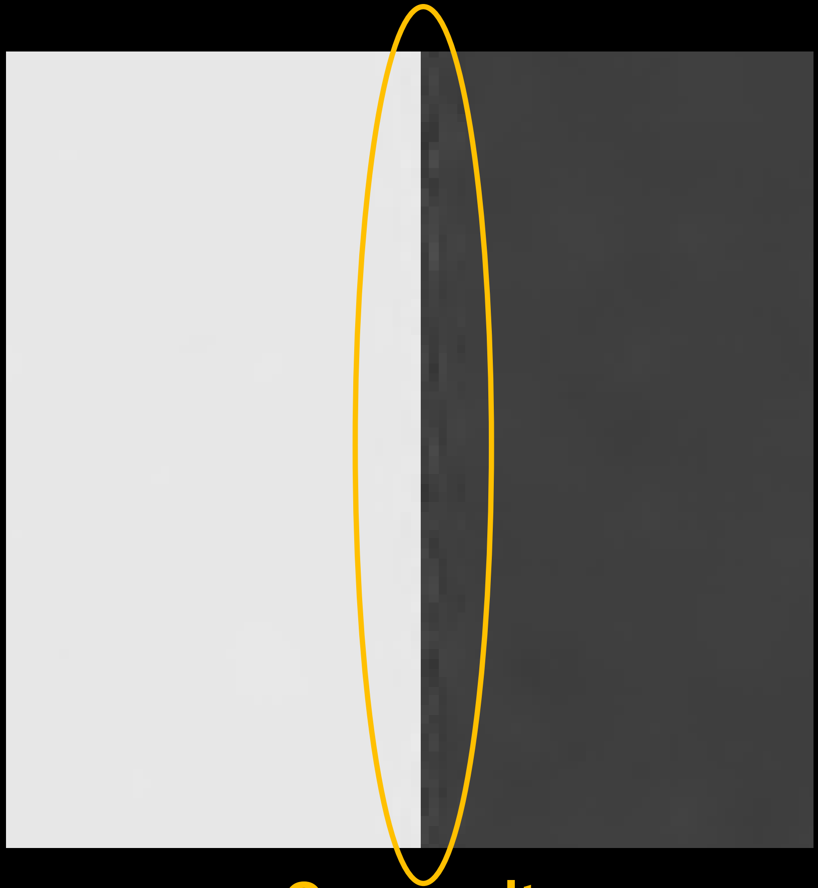
Input image



Our bandwidth map (adaptive h)

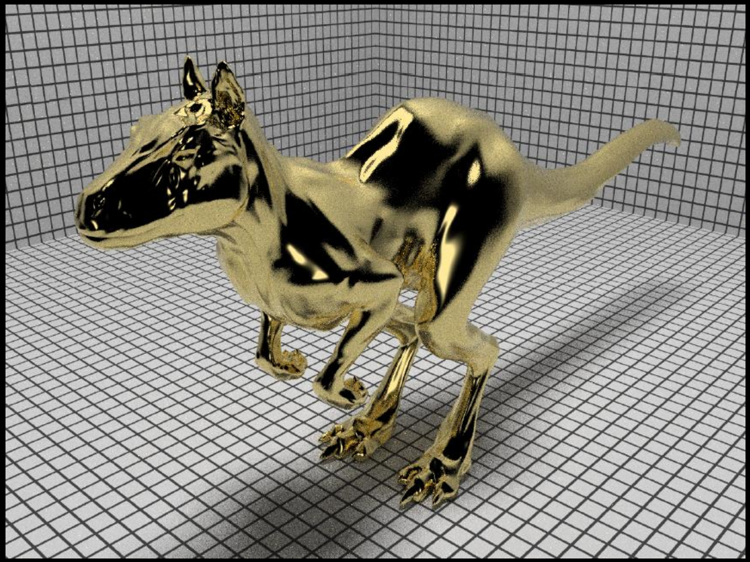
Our Approach (Adaptive Filtering)

Edge regions	Smooth regions
✓	✓

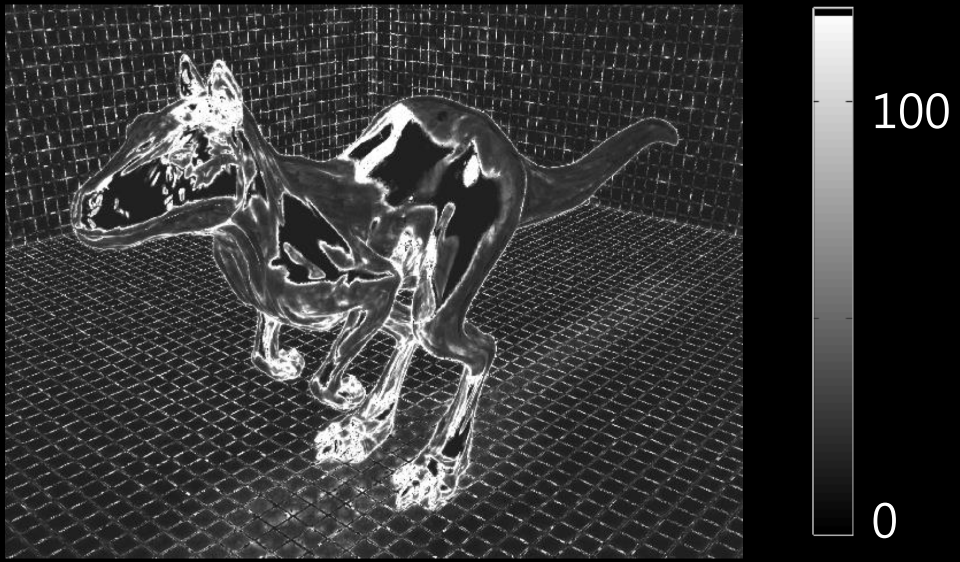


Our result

Our Approach (Adaptive Sampling)



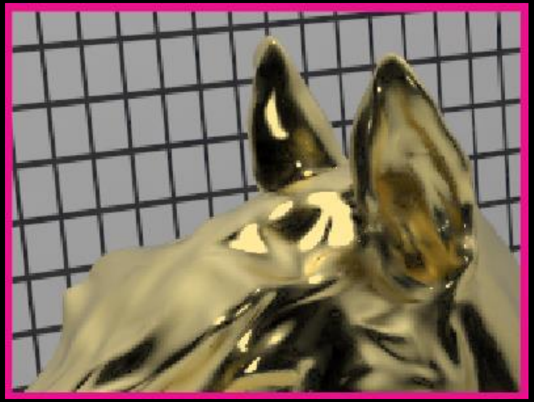
Input image
(N = 16)



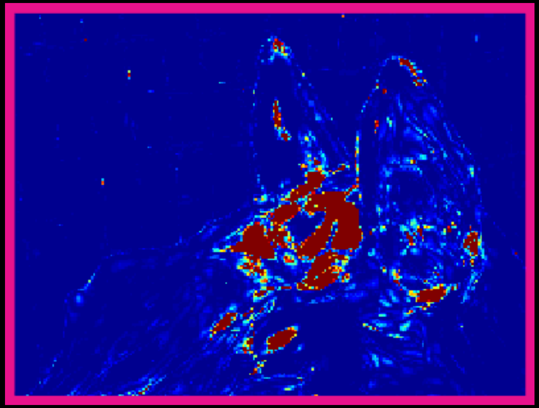
Our sampling map

Contributions

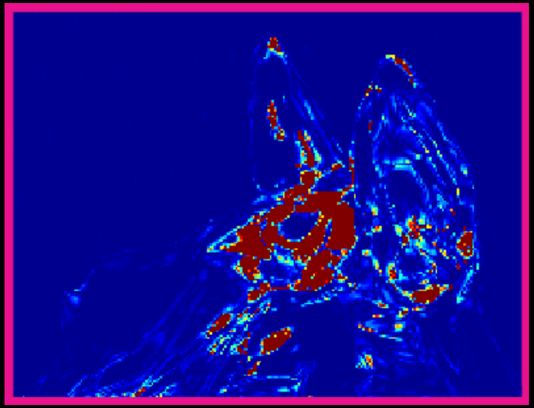
- **Propose a new image adaptive sampling and filtering based on weighted local regression**



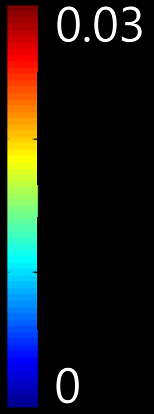
Filtered image



Our MSE estimation



Reference MSE



Results

Equal-time Comparisons

Low discrepancy sampling
(N = 143)

Our method
(N = 128)

Equal-time Comparison for Animation

[Rousselle et al. 2012]
(N = 136)

Our method
(N = 128)

Conclusion and Future Work

- **To accelerate Monte Carlo ray tracing**
  - ✓ Achieve a high cache utilization
  - ✓ Reduce the required number of ray samples

- **Future work**
  - Support for real-time applications (e.g., games)

Thank you