Voxel-based Global Illumination

Jin Hur Junhyuk Yoon

Computer Graphics and Image Processing Laboratory, SNU

Paper Info

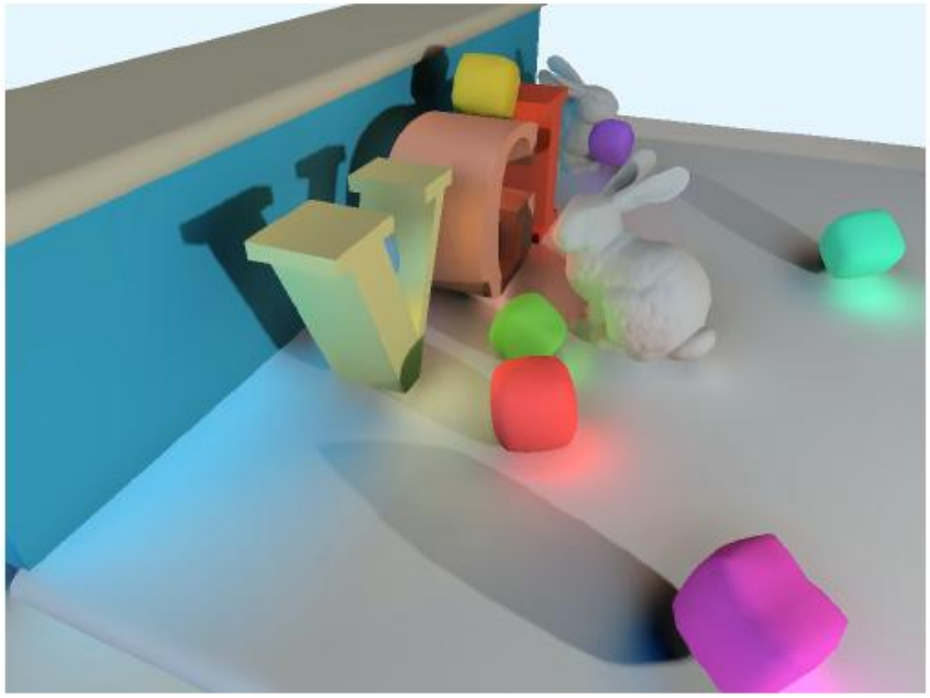
- Voxel-based Global Illumination
- Sinje Thiedemann*, Niklas Henrich*, Thorsten Grosch†, Stefan Müller*

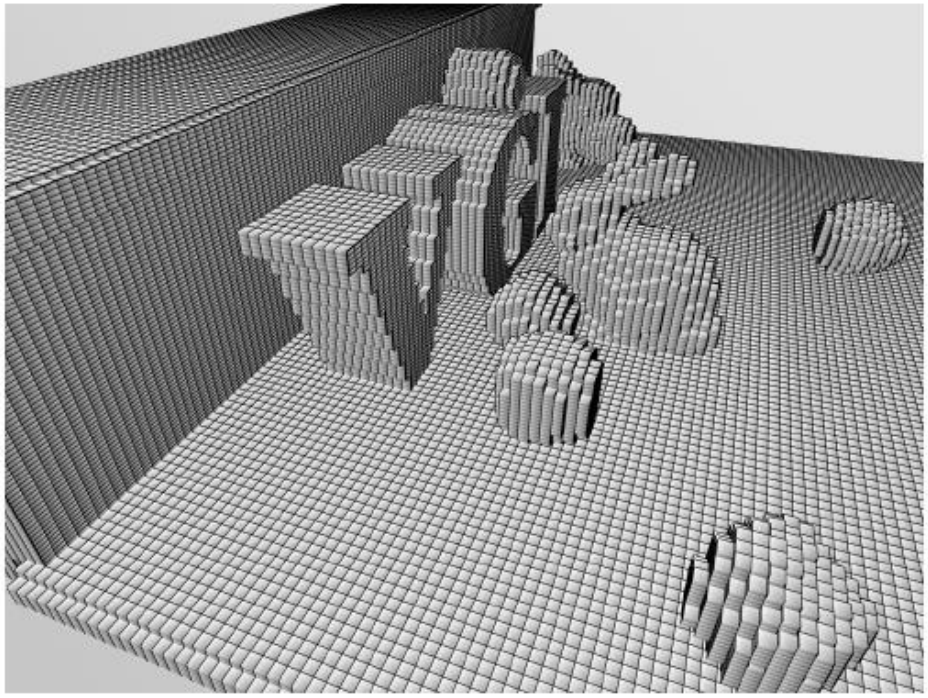
  ** University of Koblenz-Landau, Germany*

  † University of Magdeburg, Germany
- I3D '11 Symposium on Interactive 3D Graphics and Games

Goal

- Computing global illumination in real-time, given a large and dynamic scene





Why voxel-based model?

- The original polygon-based scene description is too time-consuming for computing the light transport

- There are several fast screen-space illumination for it, but they have limitations since they can only simulate what is visible in the camera image

Contribution

Jin Hur

- A new atlas-based voxelization method
- An improved ray/voxel intersection test

Junhyuk Yoon

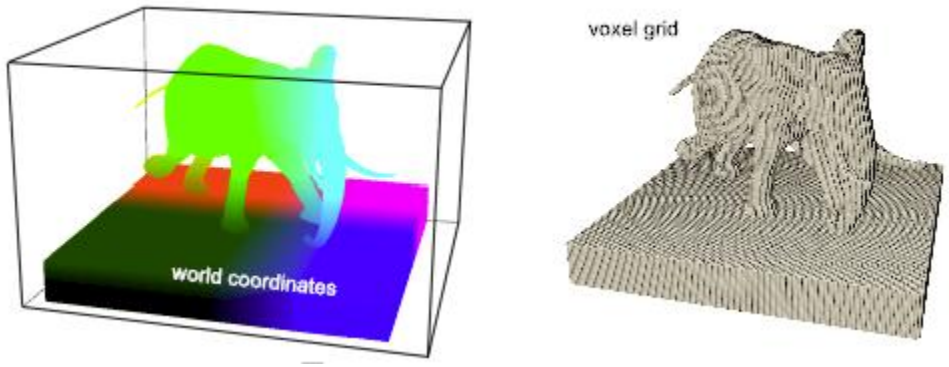
- Real-time near-field illumination with voxel visibility
- Interactive global illumination with voxel visibility

Jin Hur

PART 1. VOXELIZATION

What is voxelization ?

- FROM
  - A scene representation consisting of discrete geometric entities
- TO
  - A three-dimensional regular spaced grid

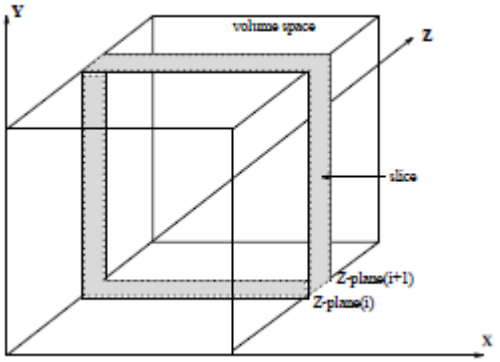


Type of voxelization

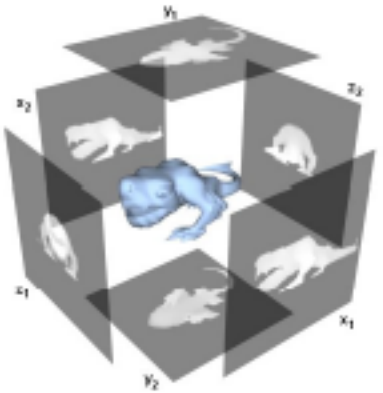
- *Binary voxelization*
  - A cell stores whether geometry is present in this cell or not
- *Multi-valued voxelization*
  - A cell can also stores arbitrary other data like materials or normals

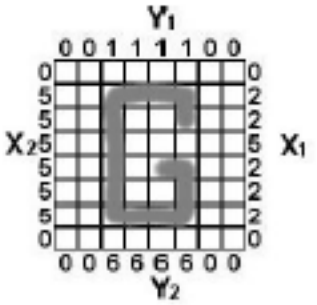
- *Boundary voxelization*
  - encodes the object surfaces only
- *Solid voxelization*
  - captures the interior of a model

Other methods



< Slicing based voxelization >





< Depth-peeling based voxelization >

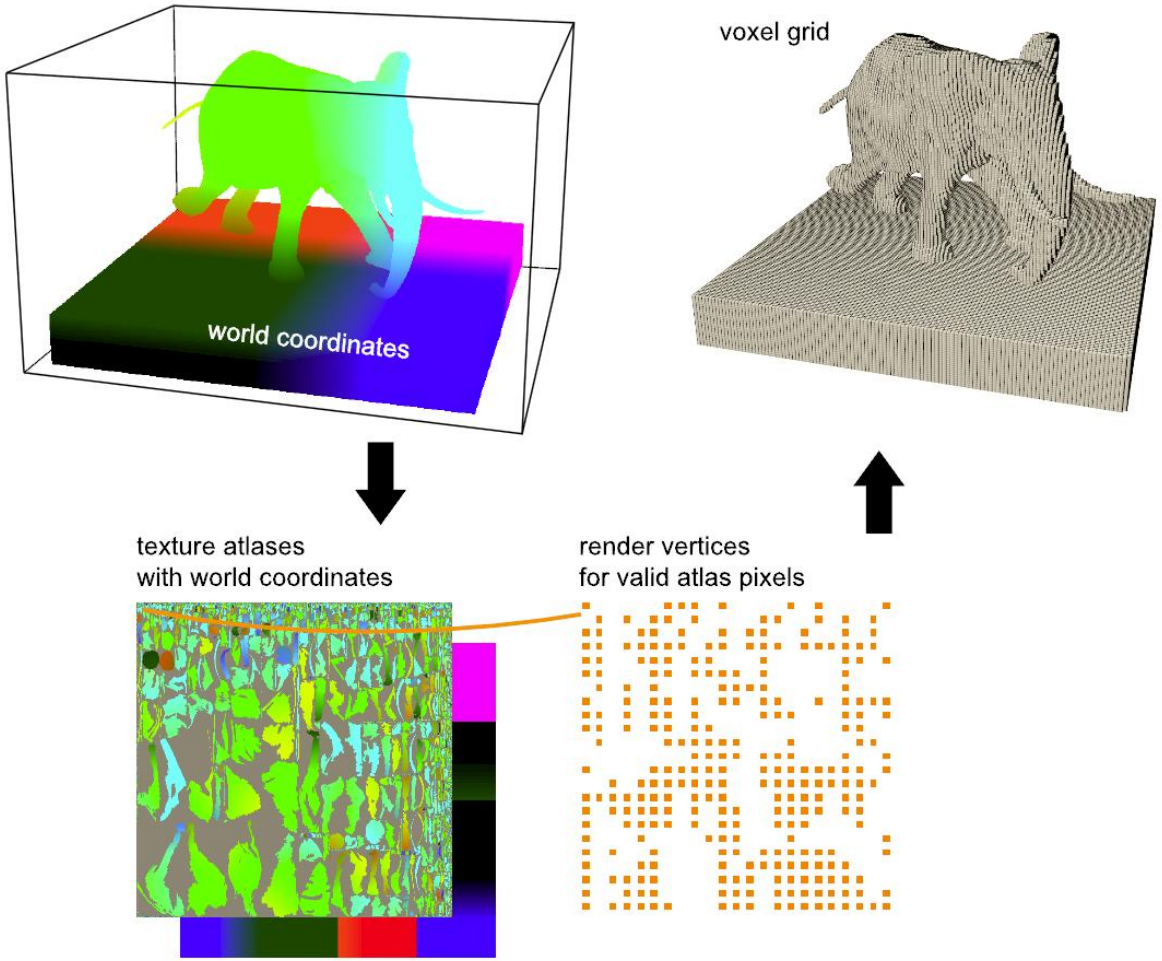
Paper’s method

Atlas-based voxelization

Hierarchical ray/voxel intersection test

Atlas-based voxelization (1/6)

- Algorithm



Atlas-based voxelization (2/6)

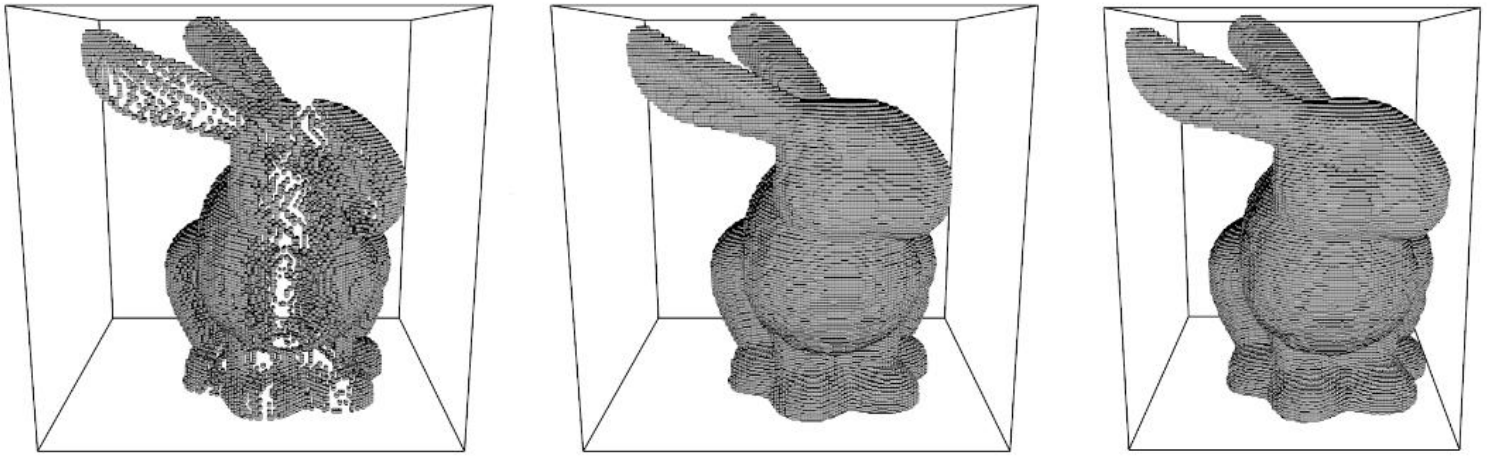
- Atlas texture
  - Binary
    - 2D texture
    - The bits of the RGBA channels of a 2D texture are used to encode the voxels
  - Multi-valued
    - 3D texture
    - One texel per one voxel

Atlas-based voxelization (3/6)

- Pros
  - No restrictions to the objects
  - Applicable for dynamic rigid bodies and moderately deforming models
- Cons
  - The objects should be stored in a texture atlas and an appropriate mapping should be generated for the models already
  - Not allow strong deformations of the object

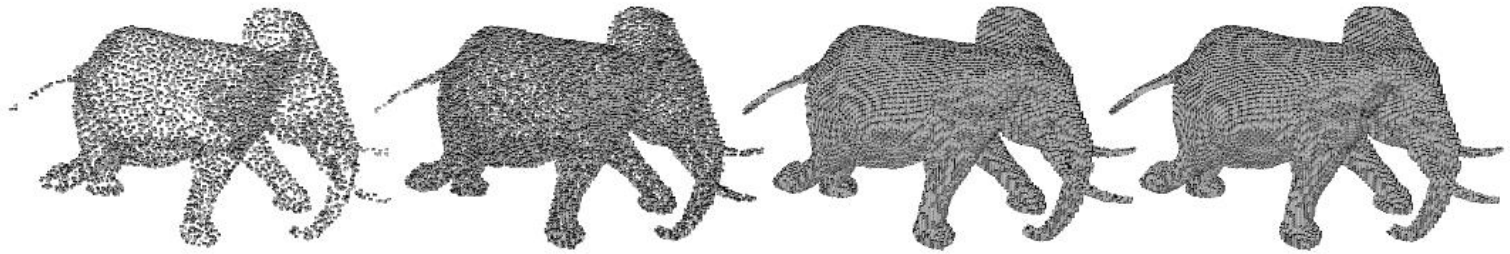
Atlas-based voxelization (4/6)

- Comparison with other methods
  - Problems with polygons which are viewed from a grazing angle



Atlas-based voxelization (5/6)

- Performance is directly related to the number of rendered vertices
- It is needed to choose sufficient atlas resolutions



Atlas-based voxelization (6/6)

- Environment
  - GeforceGTX295, Intel Core2Duo 3.16 GHz, 4GB RAM
- Performance

Voxel-grid resolution	Time (ms)	Vertices	Atlas resolution
$64^2 \times 128$	0.52	15k	176×176
$128^2 \times 128$	0.69	65k	368×368
$256^2 \times 128$	1.48	285k	768×768
$512^2 \times 128$	3.37	791k	1280×1280

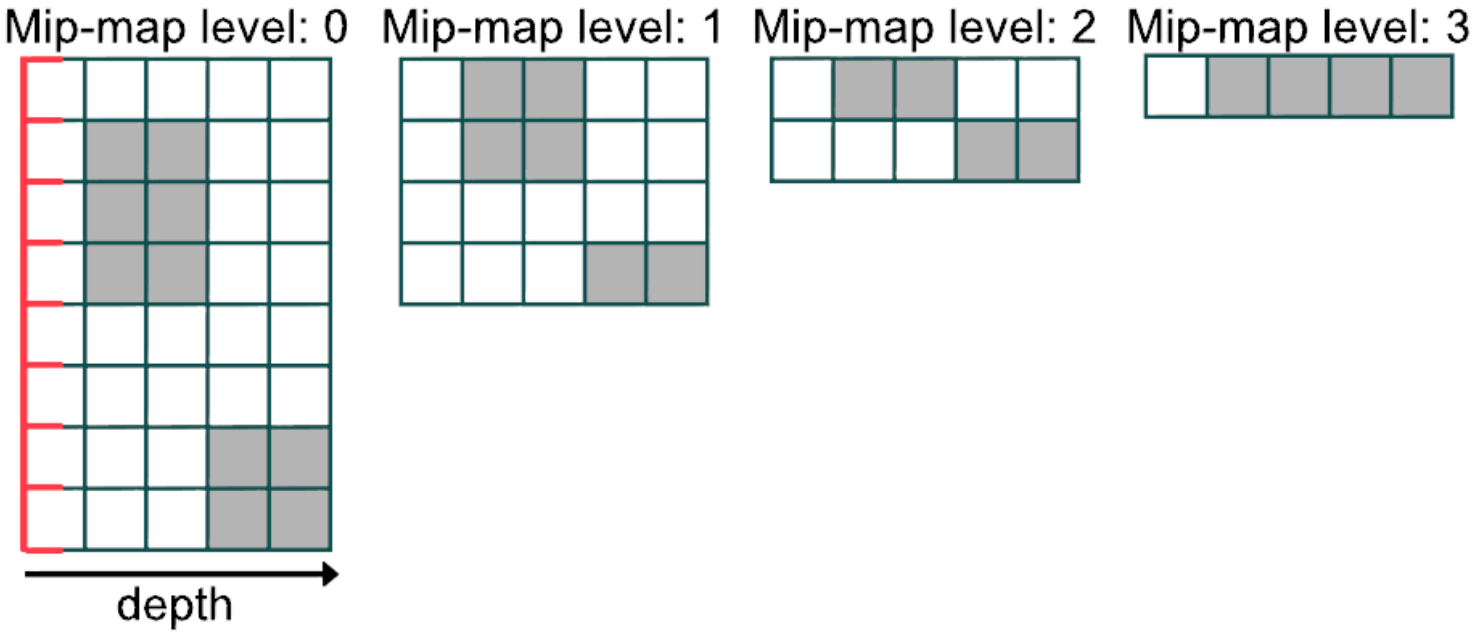
< Binary voxelization >

Voxel-grid resolution	Time (ms)	Vertices	Atlas resolution
$64^2 \times 128$	1.23	15k	176×176
$128^2 \times 128$	2.01	65k	368×368
$256^2 \times 128$	4.29	285k	768×768

< Multi-valued voxelization >

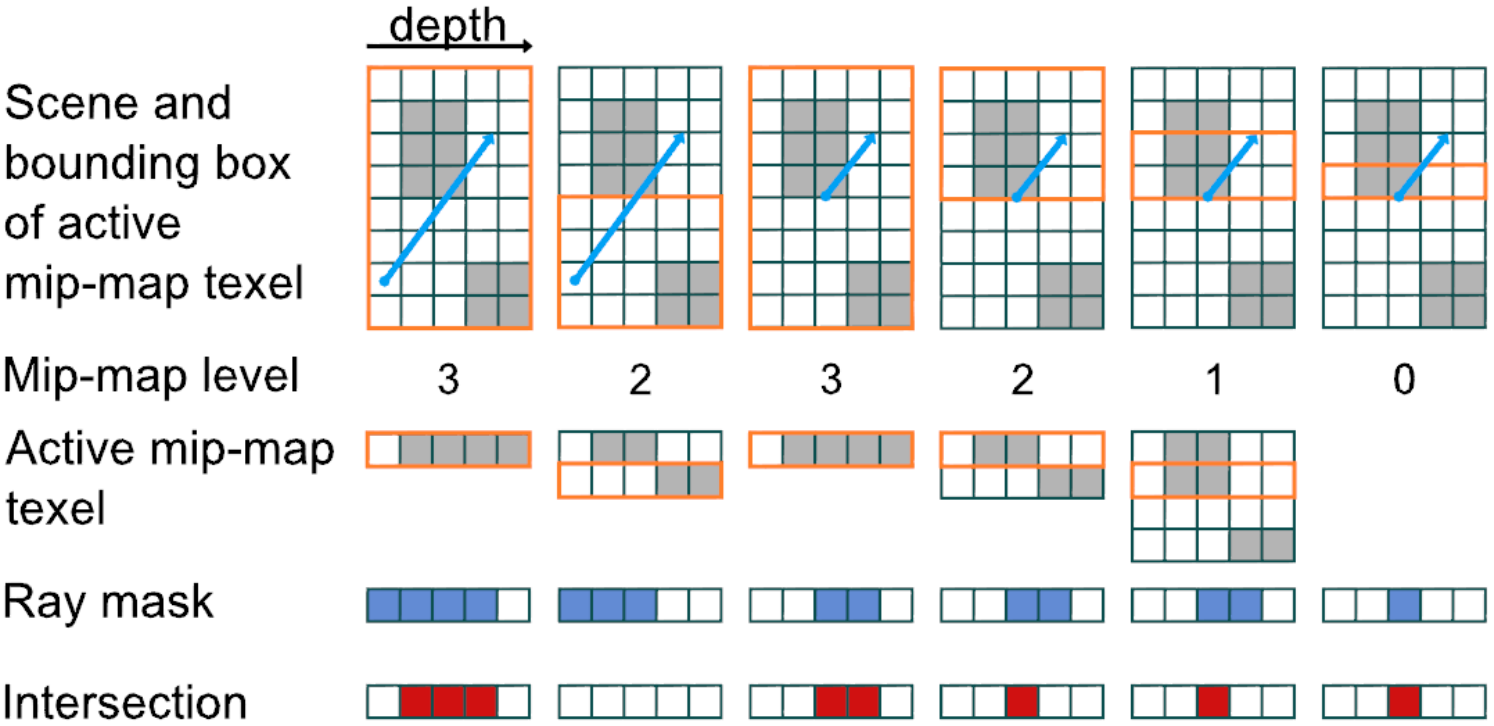
Ray-voxel intersection test (1/2)

- Use a binary voxelized scene representation
- Build a mip-map hierarchy



Ray-voxel intersection test (2/2)

- Algorithm



Junhyuk Yoon

PART 2. ILLUMINATION

Illumination Methods

- Indirect Illumination using voxels
  - Using voxelized scene representation
  - With optimized intersection test
- Using Monte-Carlo Integration

$$L_o(\mathbf{x}, \omega_o) = \int_\Omega f(\mathbf{x}, \omega_o, \omega_i)\, L_i(\mathbf{x}, \omega_i)\, \cos\theta\, d\omega_i$$

$$L_o(\mathbf{x}, \omega_o) \approx \frac{1}{N} \sum_{i=1}^{N} \frac{f(\mathbf{x}, \omega_o, \omega_i)\, \widetilde{L_i}(\mathbf{x}, \omega_i)\, \cos\theta}{p(\omega_i)}$$

$\widetilde{L_i}$: 'approximated' incoming radiance based on voxel model

Illumination Methods

- **Real-Time Near-Field Single Bounce Indirect Light**
  - Keeping the ray-length short
    → near-field single bounce in real-time
- **Voxel Path Tracing**
  - Need ‘multi-valued voxelization’
  - Can compute multiple bounces interactively
- Voxelization Procedure
  - At startup, static scenes are voxelized
  - For each frame, only the **dynamic elements** need to be voxelized additionally

Ambient Occlusion

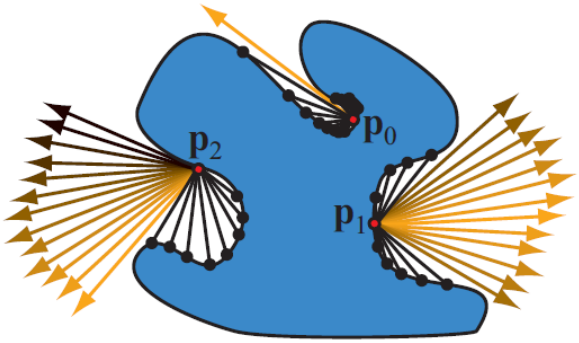
- Adding Realism with Ambient Occlusion

Environment lighting only | Adds ambient occlusion | Adds indirect lighting

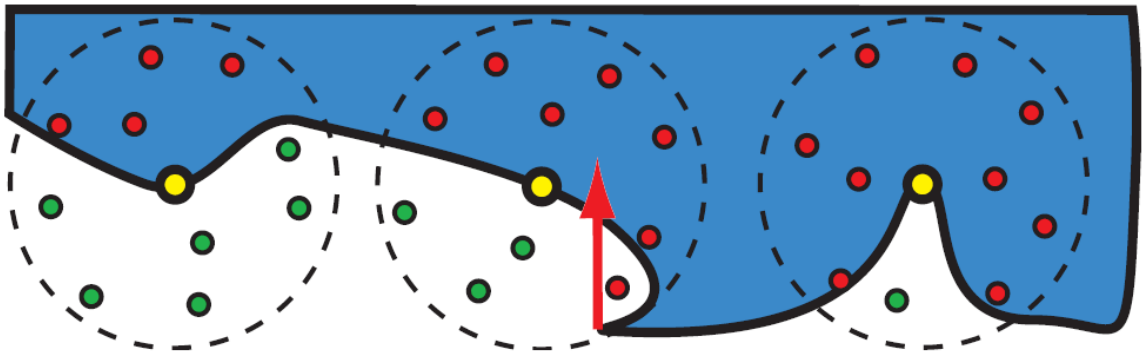
<From GPU Gems2 ch14.>

SSAO: Screen-Space Ambient Occlusion

- Ray-shooting vs. Screen-space method



Shooting rays at each positions



Using the Z-buffer:
Green samples : pass the z-test
Red samples : fail the z-test

<From Real-time Rendering book>

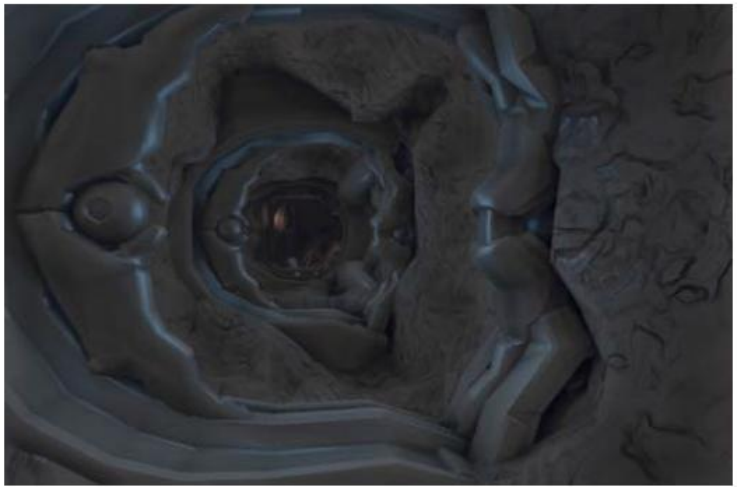
SSAO: Screen-Space Ambient Occlusion

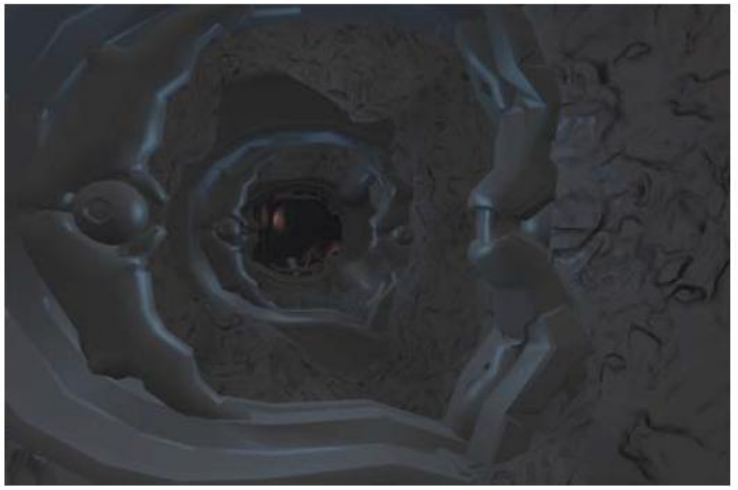
- Crytek's ambient occlusion examples

Figure 15. Screen-Space Ambient Occlusion in a complete ambient lighting situation (note how occluded areas darken at any distance)

w/ SSAO

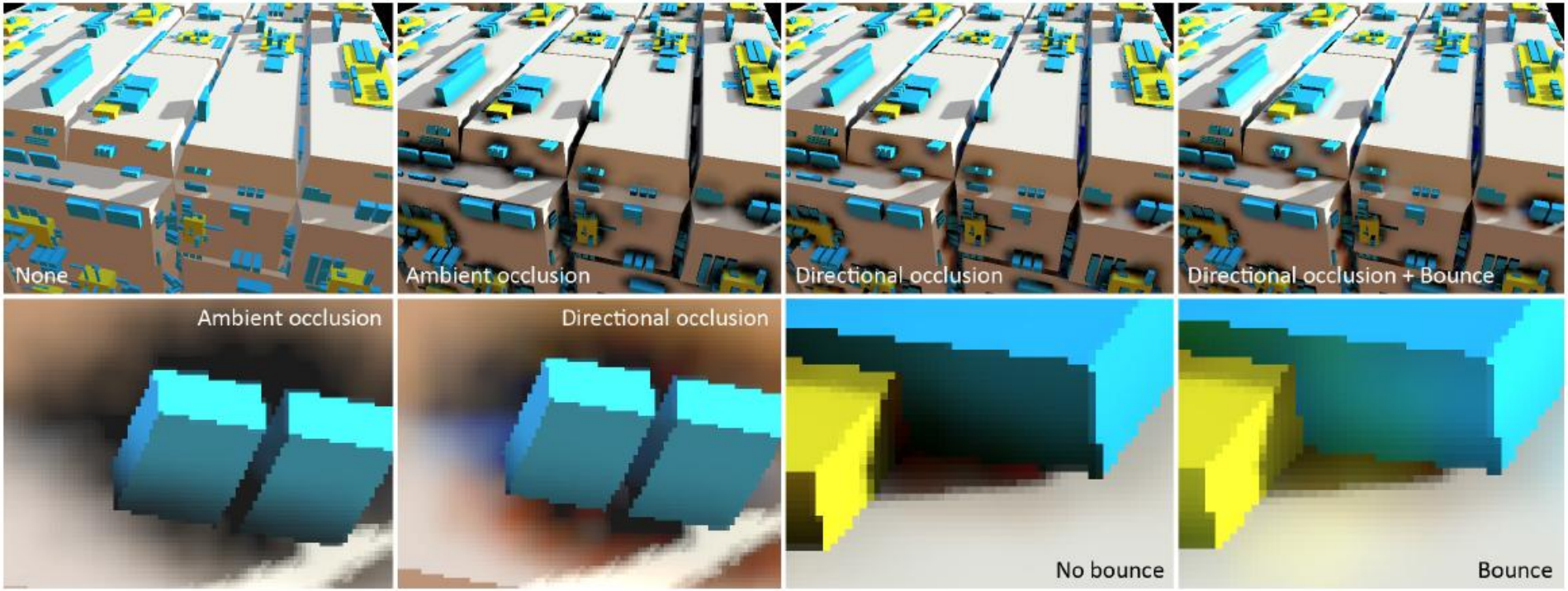
wo/ SSAO





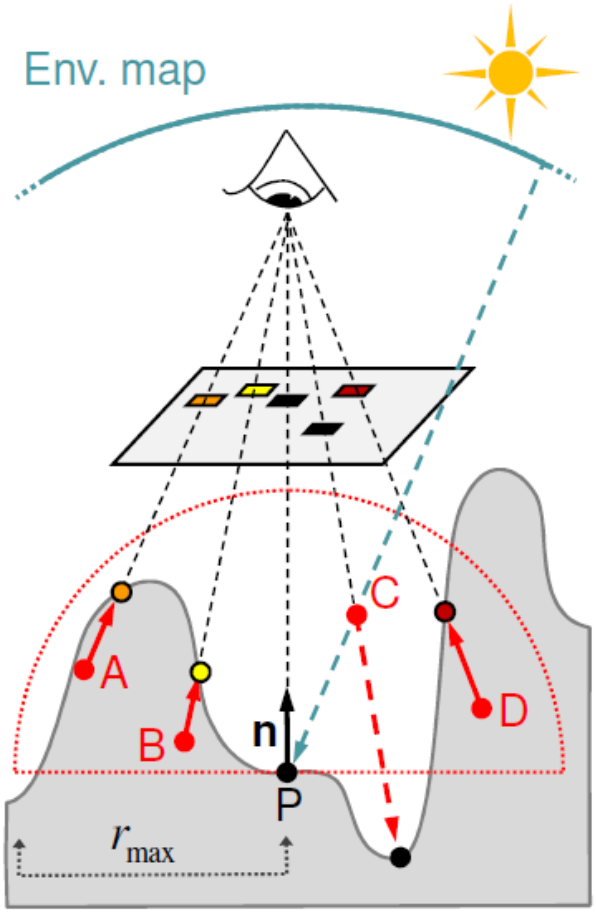
SSDO: Screen-Space Directional Occlusion

- Generalize ambient occlusion to directional occlusion

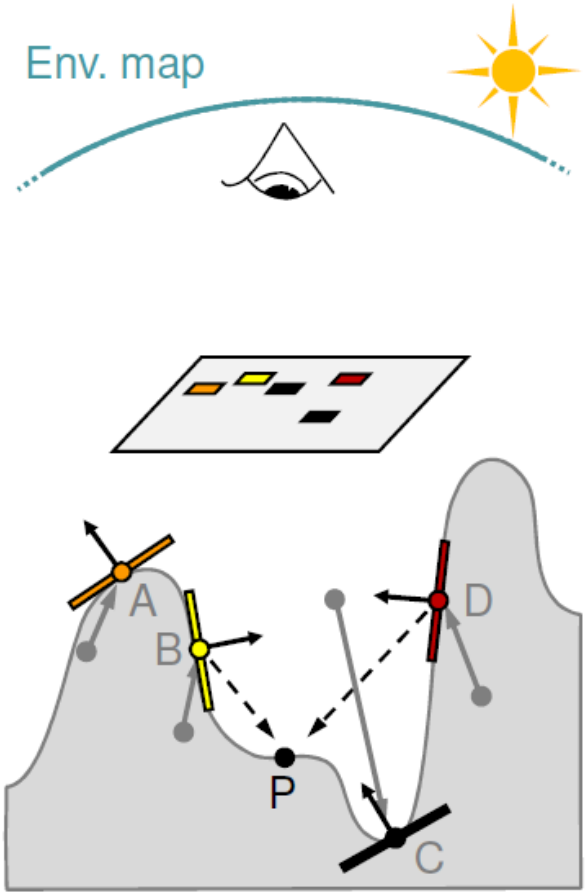


RITSCHEL, T., GROSCH, T., AND SEIDEL, H.-P. 2009. Approximating dynamic global illumination in image space. In *I3D '09: Proceedings of the 2009 Symposium on Interactive 3D Graphics and Games*, ACM, 75–82.

SSDO: Screen-Space Directional Occlusion



Illuminated only from C : colored shadow

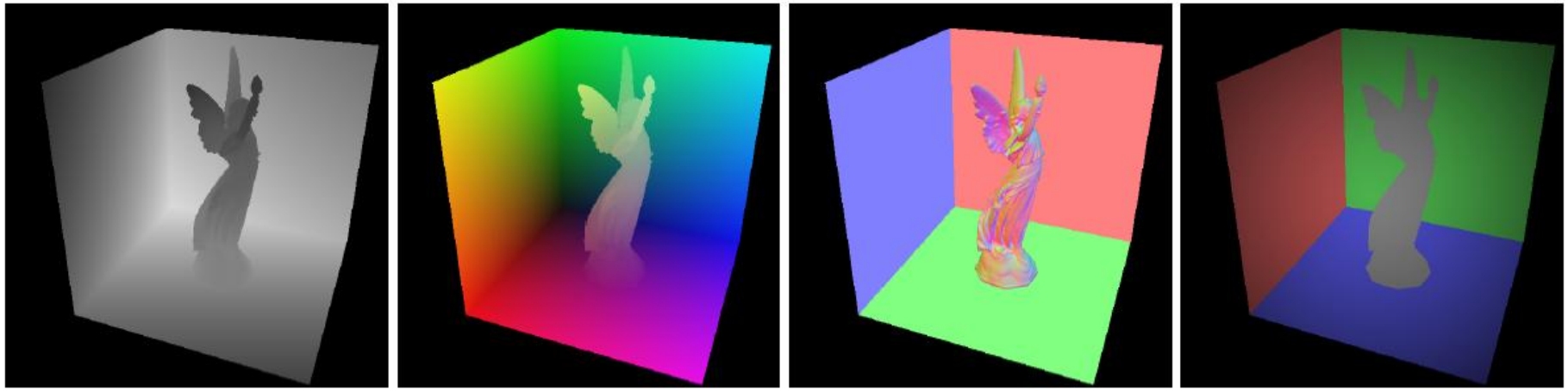


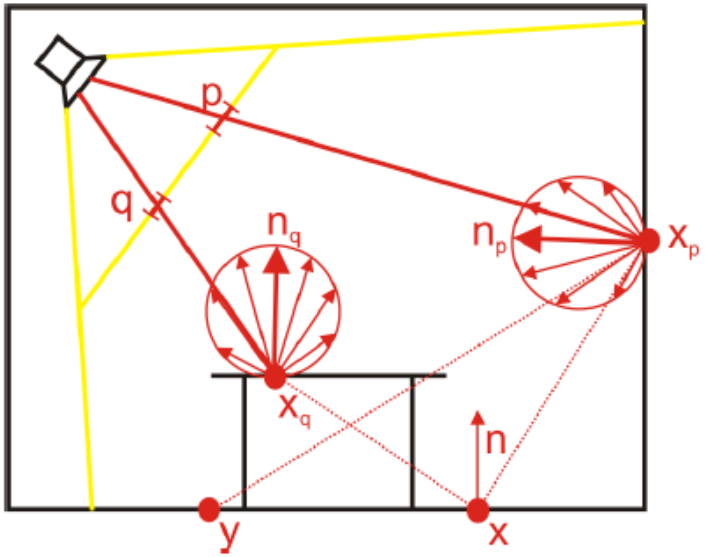
Indirect bounce : from B and D

RITSCHEL, T., GROSCH, T., AND SEIDEL, H.-P. 2009. Approximating dynamic global illumination in image space. In *I3D '09: Proceedings of the 2009 Symposium on Interactive 3D Graphics and Games*, ACM, 75–82.

RSM: Reflective Shadow Map

Shadow map (depth, world space coordinates, normal, flux)



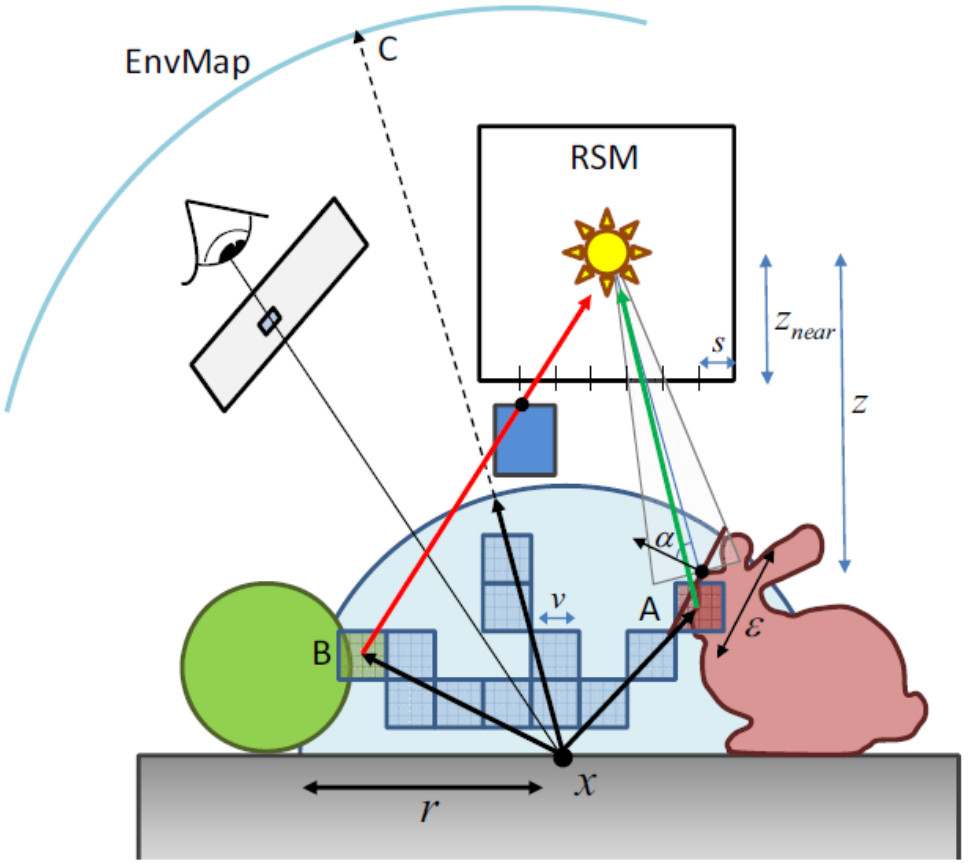


Each pixel in shadow map → small area light source

DACHSBACHER, C., AND STAMMINGER, M. 2005. Reflective shadow maps. In I3D 2005.

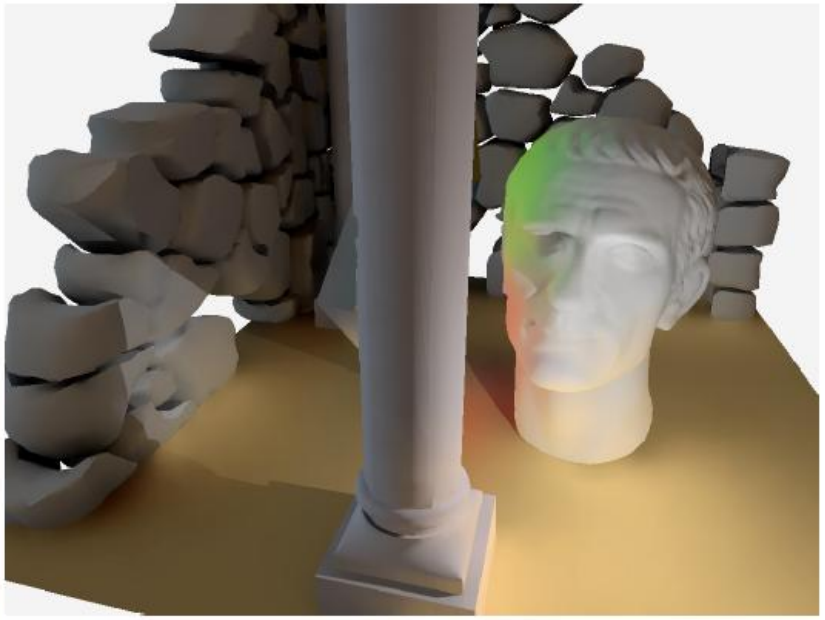
Real-Time Near-Field Single Bounce Indirect Light

- Generate **RSM** for fast near-filed illumination
- Shoot N rays from x
- Find first intersection point using **binary voxelization**
- Gather direct radiance $\widetilde{L_i}$ from RSM



Real-Time Near-Field Single Bounce Indirect Light

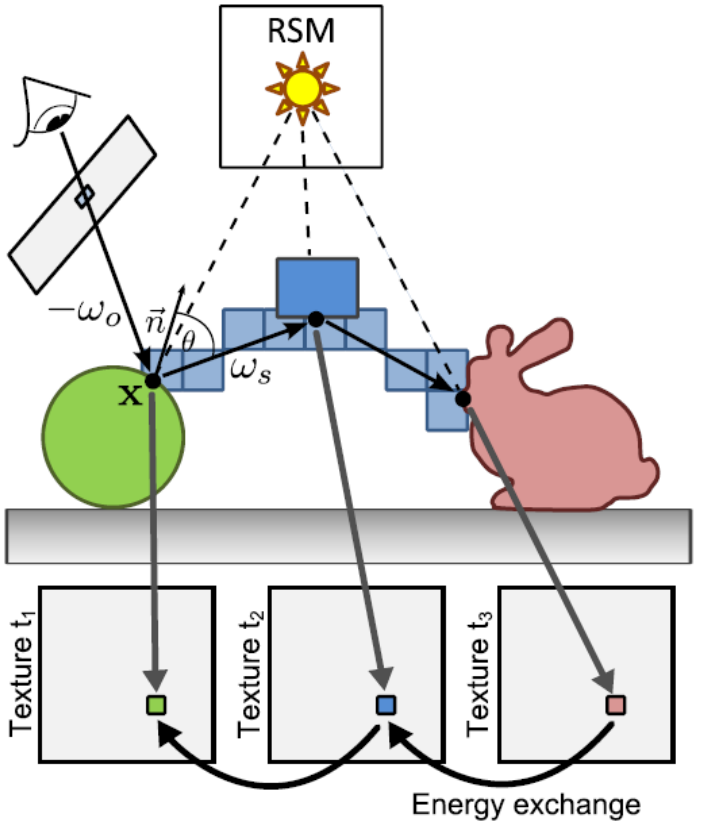
- SSDO w/ one bounce vs. Proposed method



- Senders and blockers which are invisible in the camera image are always detected.

Voxel Path Tracing

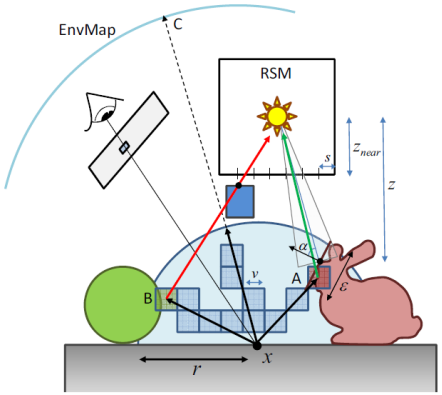
- Generate **RSM** for direct illumination term
- For each x, shoot a ray (using importance sampling of BRDF)
- If shadowed, fetch normal & BRDF from **multi-valued voxelization**
- If not shadowed, fetch from RSM
- Store the hit-position in a texture
- **Propagate the energy backwards**

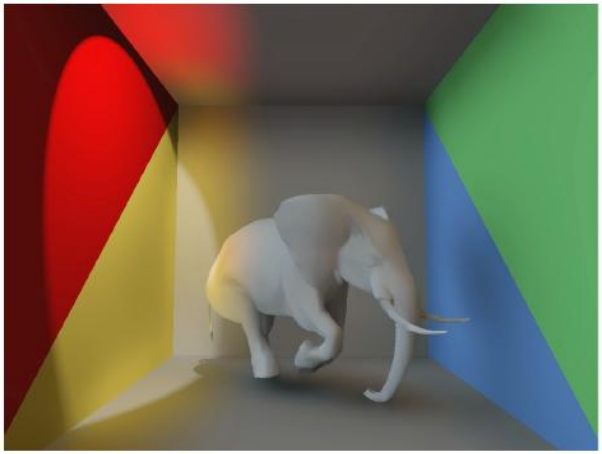


Results (movie)

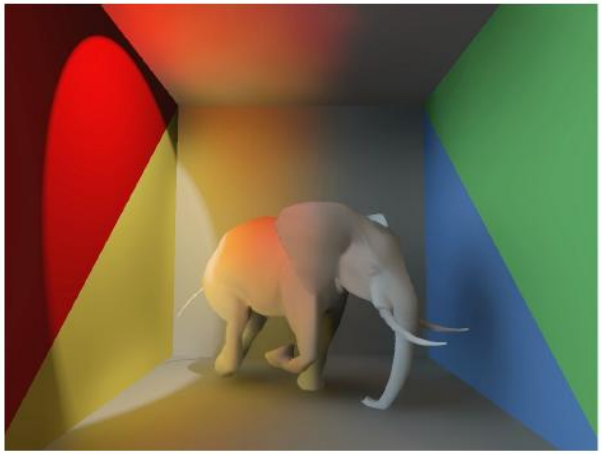
Results

- Voxel-based single bounce illumination with different radii r

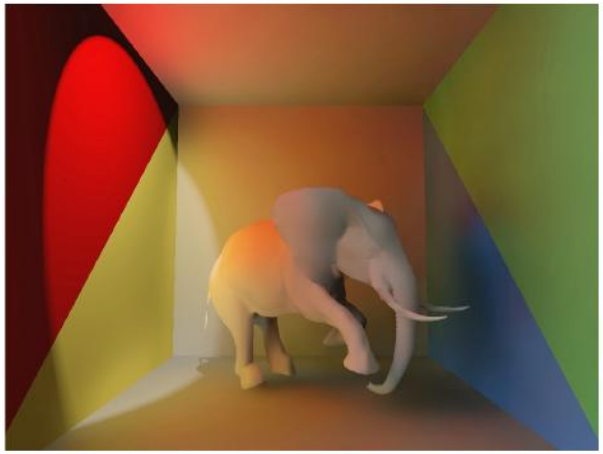




30 fps



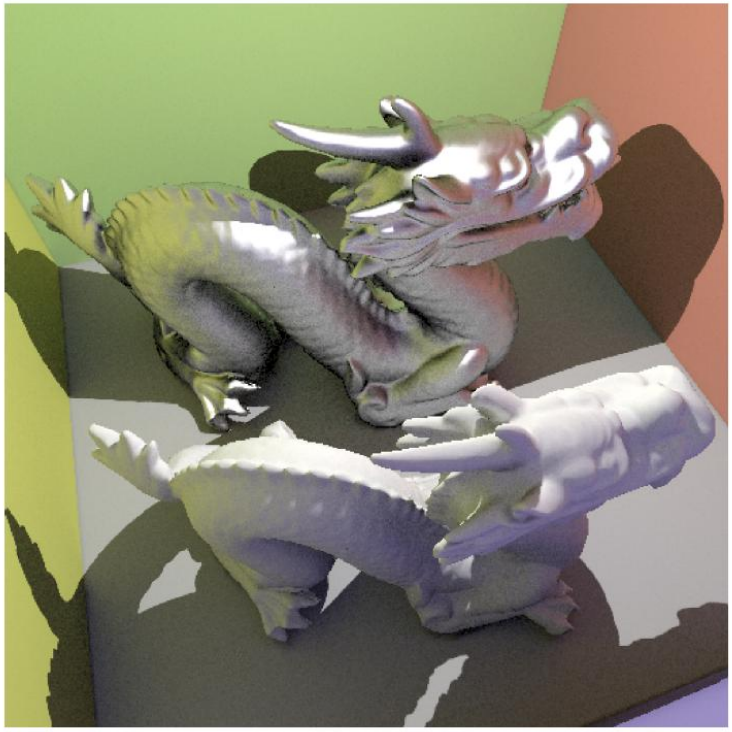
27.7 fps

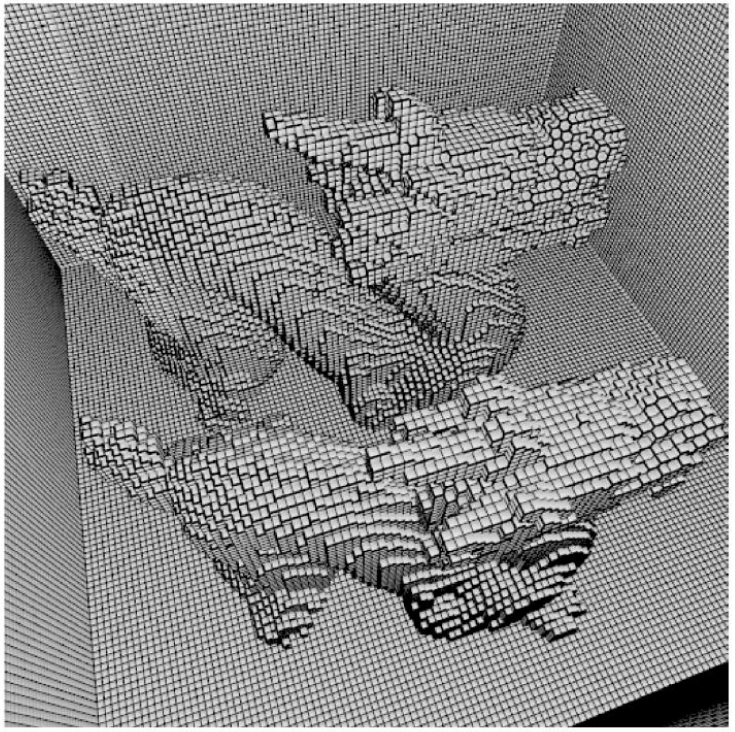


25 fps

Results

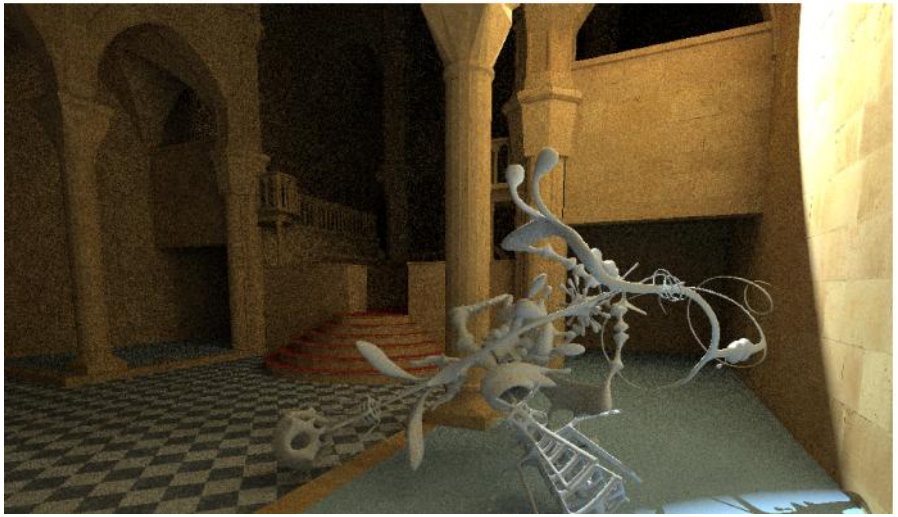
- Path tracing with voxel-based visibility
  - 32 directions per pixel, 1 bounce, 3.5 fps
  - Voxel grid resolution : 128^3



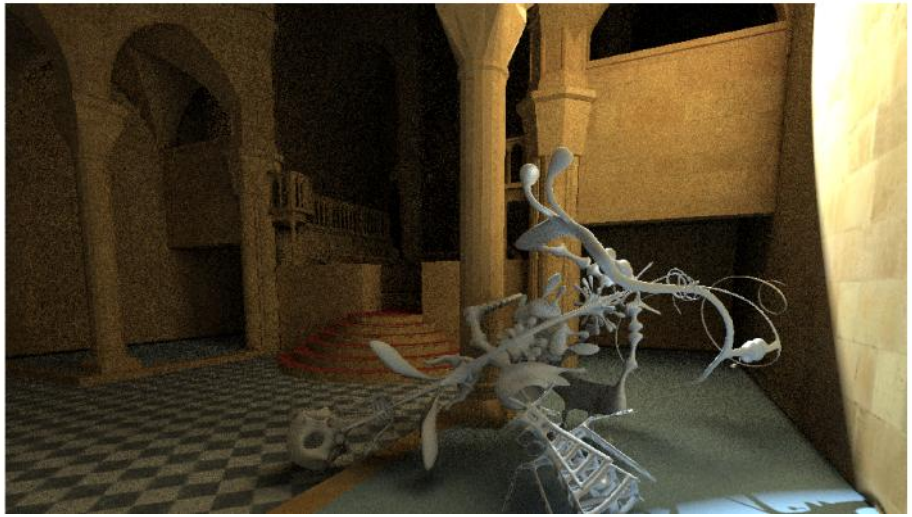


Results

- Path tracing with voxel-based visibility
  - 64 samples per pixel, two inter-reflections
  - Overall comparable, except thin structures (over-darkening from coarse voxelization)



Conventional path tracing (28 min.)



Proposed method (2.2 sec., voxel resolution: 128^3)

Results

- Scene with animated horse (28 fps)
  - 2 spot lights/RSMs, 128^3 voxel resolution, 20 samples per pixel
  - Indirect light rendered at ¼ x ¼ image → up-scaled to 1024x768