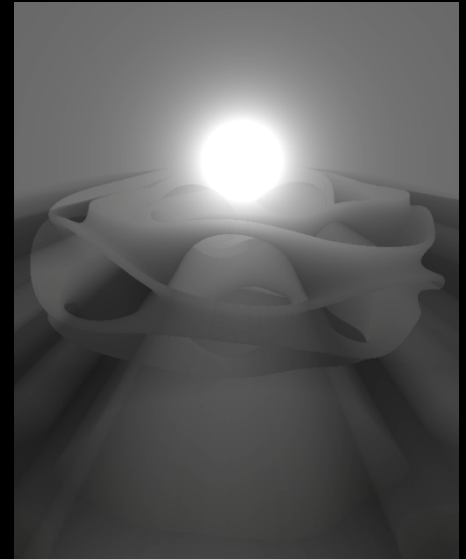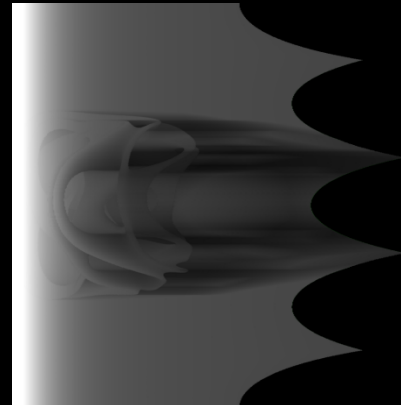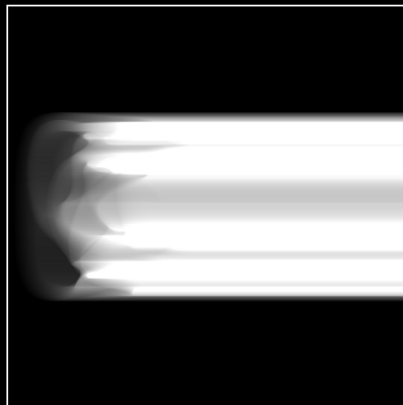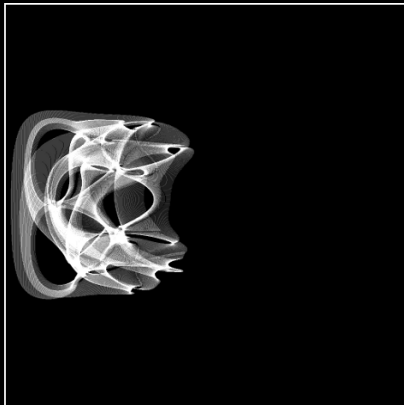# Voxelized Shadow Volumes

Chris Wyman

Department of Computer Science

University of Iowa

Problem: Visibility In Volumes

- Quickly solve the query:
  - Which points in volume see some point P?

- Example application: volumetric lighting
  - When integrating scattering with shadows

- Without loss of generality:
  - Use volume lighting for concrete examples
  - Our algorithm is not limited to this problem

Imagine: Want Volume Lighting

Creative Commons Image: Mila Zinkova

Why Is This Hard?

- Alternatively:
  - Why not use shadow maps?
  - Why not use shadow volumes?

- After all, many propose just that:
  - Shadow volumes [Max86] [Biri06]
  - Shadow maps [Dobashi02] [Englehardt10] [Chen11]
  - Combine both [Wyman08] [Billeter10]

THE UNIVERSITY OF IOWA
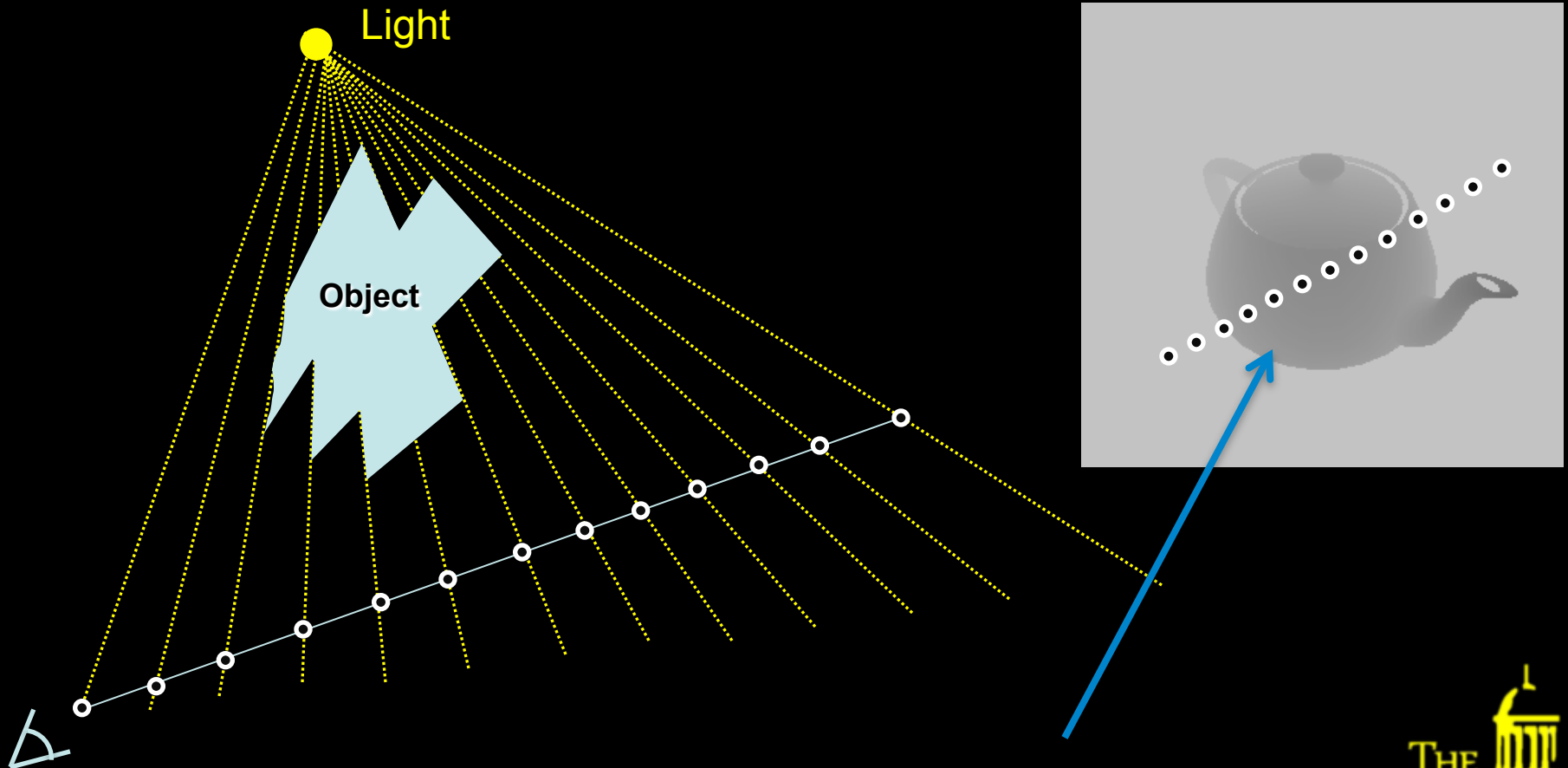
Why Is This Hard?

- Rendering without visibility:
  - [Sun et al. 05] takes << 1ms (~1000+ fps)
  - [Pegoraro et al. 09] takes < 3 ms (~300+ fps)

- Rendering with visibility:
  - [Chen et al. 11] takes 7-24 ms
  - [Billeter et al. 10] takes 9-100 ms
  - Older techniques even slower
  - Most slow with increased geometric complexity

- Obviously, visibility is quite costly!

Why Is This Hard?

- Both SMs and SVs designed for surfaces
  - Shadow volumes scale more naturally
    - As they bound regions
    - Require significant fill rate
    - Fill rate increases with higher frequency shadows
  - Shadow maps simply sample more
    - Sample at points in volume, not just on surfaces
    - Leads to *incoherent* memory access
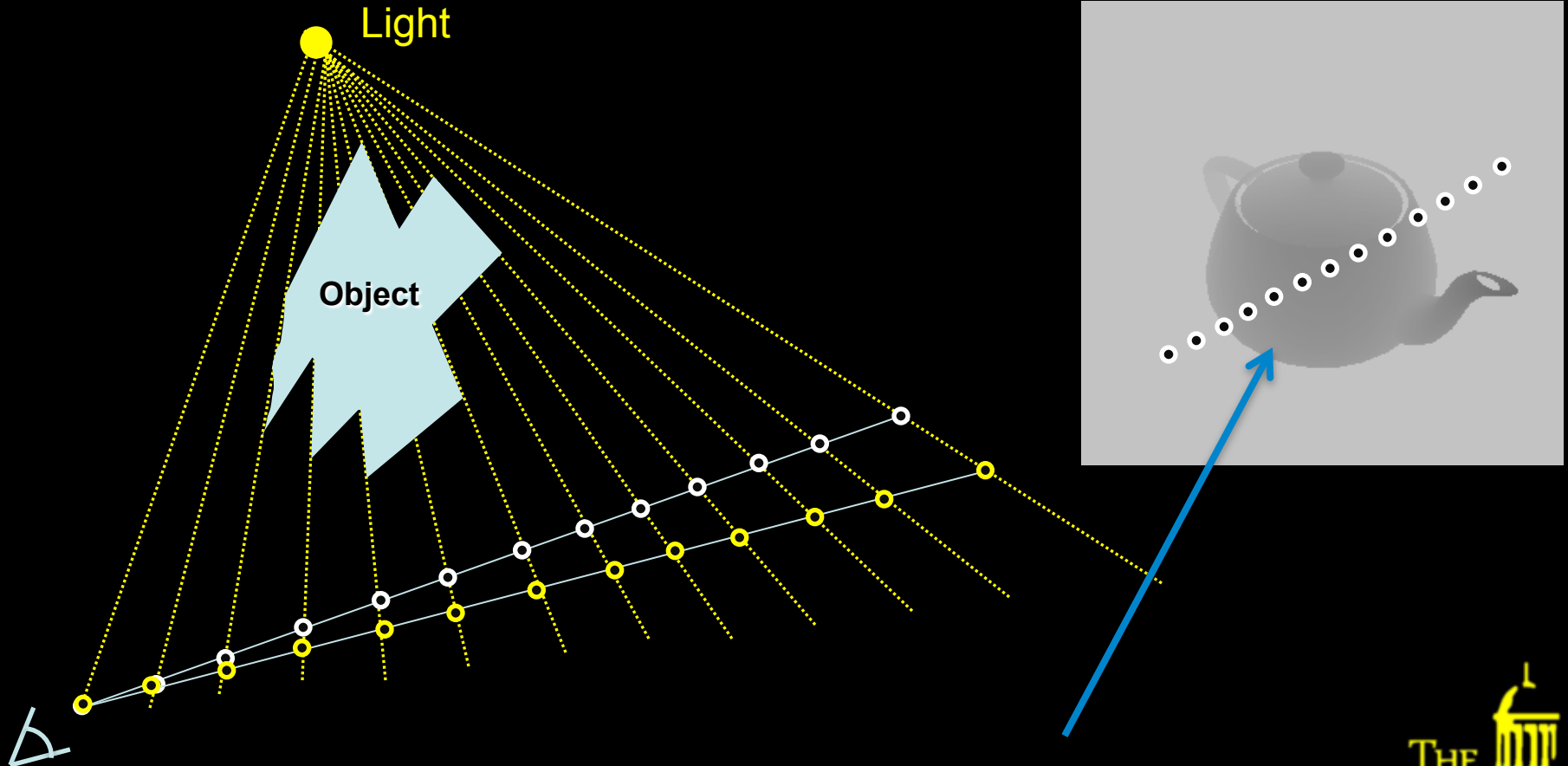   - Leads to *redundant* memory access

Why Is This Hard?

Light

Object

Shadow map samples likely have poor cache coherence.

Why Is This Hard?

Light

Object

Independently computed samples along adjacent rays look into *same* set of shadow map texels. And there is no guarantee values are still in the texture cache.

THE UNIVERSITY OF IOWA

Voxelized Shadow Volume Goals

- Develop a *cache coherent* visibility lookup
  - Eye ray lookups should be efficient
  - Lookups for nearby eye rays near in texture

- Eliminate redundancy between lookups
  - Less important
  - But falls out naturally

What Are VSVs? (The Basics)

- ## Imagine voxelizing
  - To 3D voxel grid

- ## Parallel scan
  - Along a grid axis
  - Use an bitwise OR
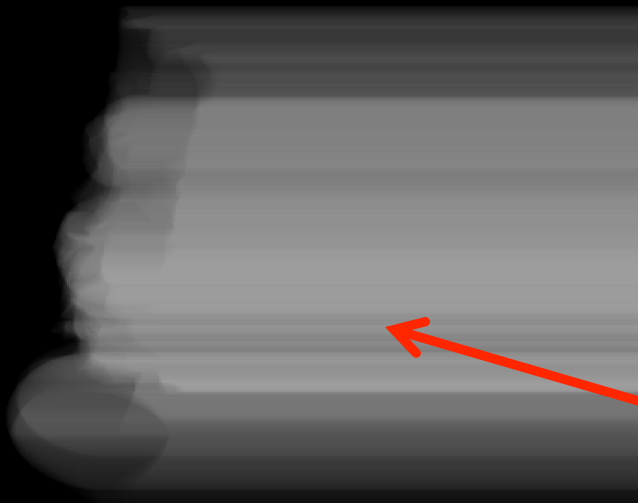
Light direction

THE UNIVERSITY OF IOWA
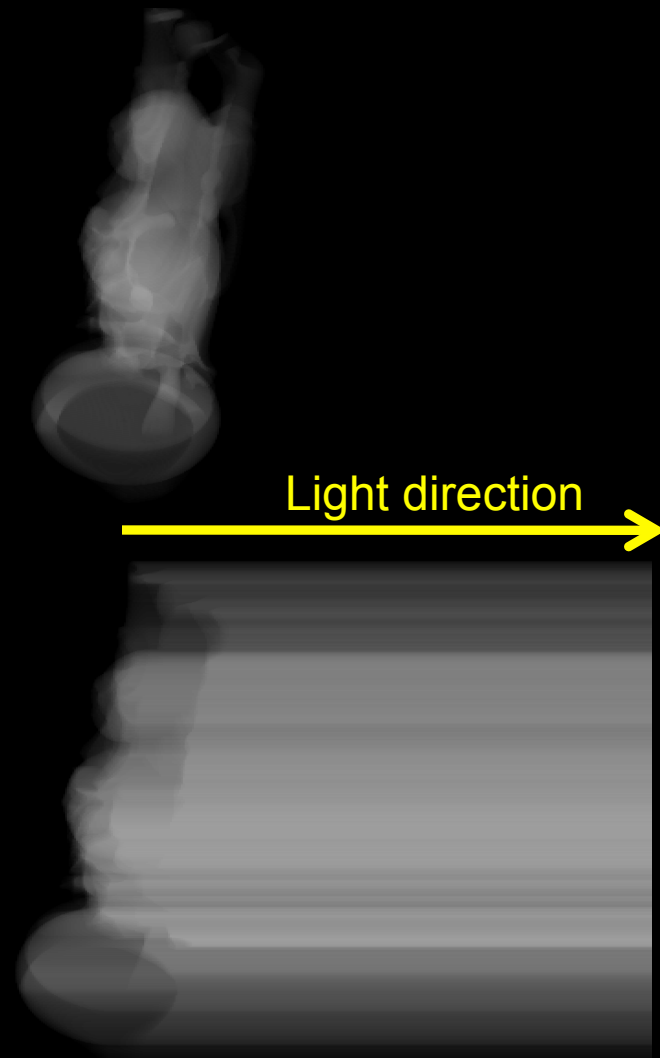
What Are VSVs? (The Basics)

- Each pixel corresponds to:
  - Set of binary voxels, where
    - 1 → inside geometry
    - 0 → outside geometry

- Each pixel corresponds to:
  - Set of binary voxels, where
    - 1 → inside shadow volume
    - 0 → outside shadow volume

A voxelized shadow volume!

Seems Too Simple…

- Why has it never been done?
  - Need to scan along grid axis → very limiting

- But, if possible… What advantages?
  - Voxelization & render resolution independent
  - Voxelization occurs in different pass
  - Implies:
    - Shadow volume fill rate decouples from geometric complexity & screen resolution
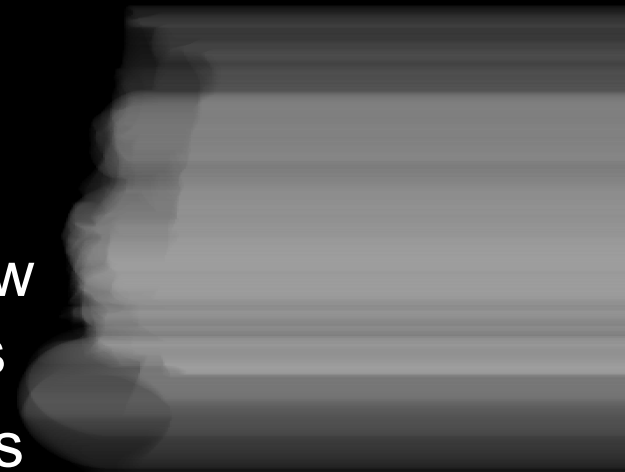
Seems Too Simple…

- ## Other advantages?
  - Lookup logistics:
    - One "pixel" (at right) gives shadow visibility at many volume samples
    - Store 128 binary visibility samples in a texel on GPU
    - Significantly reduces lookups for dense visibility sampling

   - Used anywhere visibility needed

How Do We Generalize?

- Need parallel scan to run along grid axis
  - I.e., light direction runs along axis
  - Similar idea from ray tracing literature
    - [Hunt 08] Perspective-space accel structures

- Need per-pixel lookup along grid axis
  - To stash row of visibility samples in a texel
  - Done frequently in GPU computing
    - [Eisemann 06] Screen-space voxel grid

- Novel: Need them simultaneously

THE UNIVERSITY OF IOWA

Creating VSVs

- We propose *epipolar space voxelization*
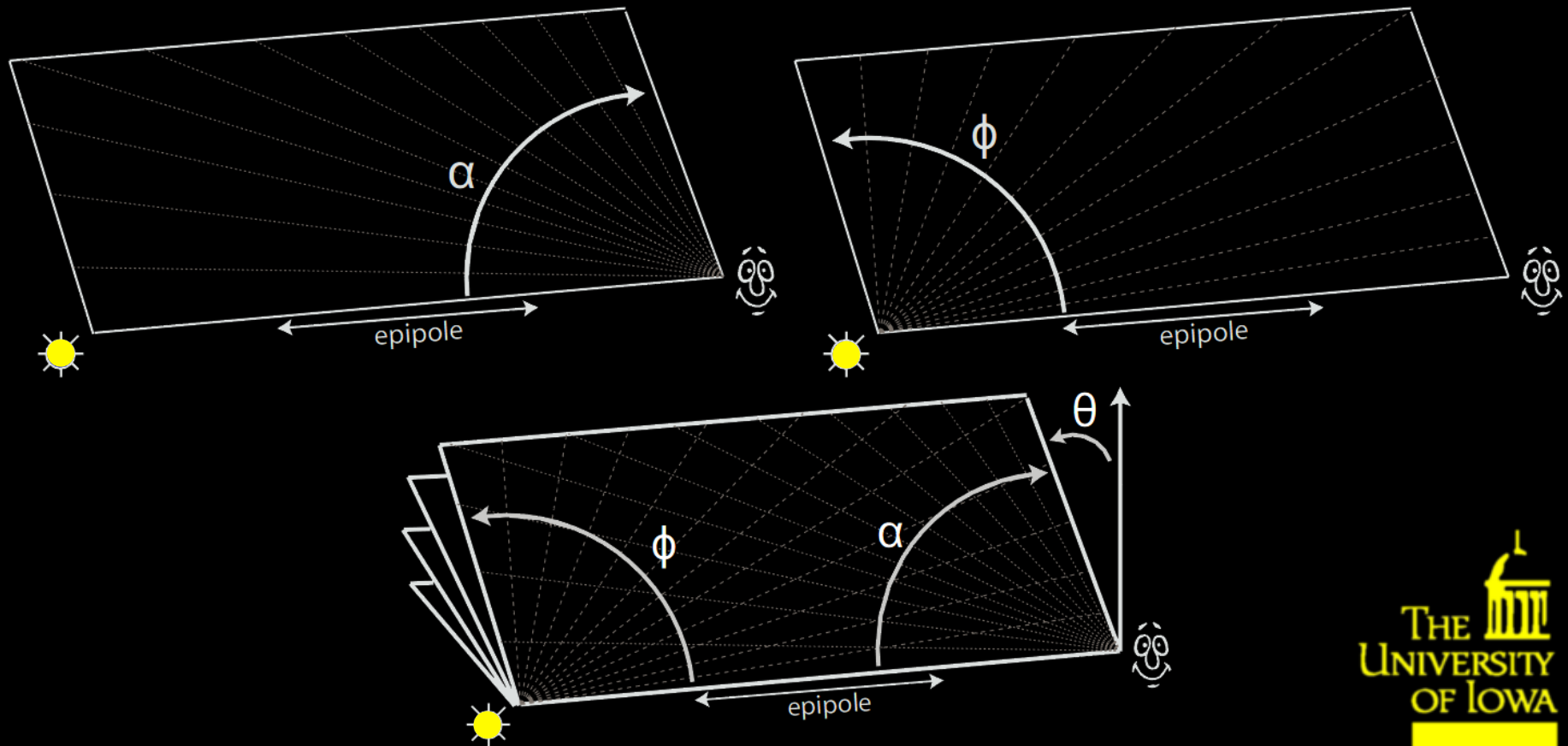
Creating VSVs

- Straightforward parameterization:

**Given eye-space light and vertex position:** `esLPos, esVPos`

```
vec3 toLight = normalize( esLPos );
vec3 toVert = normalize( esVPos );
vec3 upVec = normalize( cross( toLight, vec3(0,0,-1) ) );
vec3 forwardVec = cross( upVec, toLight );
```

$$\text{float } \alpha = acos(dot(toLight, toVert));$$
$$\text{float } \theta = atan(dot(forwardVec, toVert), dot(upVec, toVert));$$
$$\text{float } \phi = acos(dot (-toLight, normalize(esVPos-esLPos)));$$

- Satisfies our constraints:
  - One axis (constant α) parallel to view rays
  - One axis (constant Φ) parallel to light rays

Creating VSVs

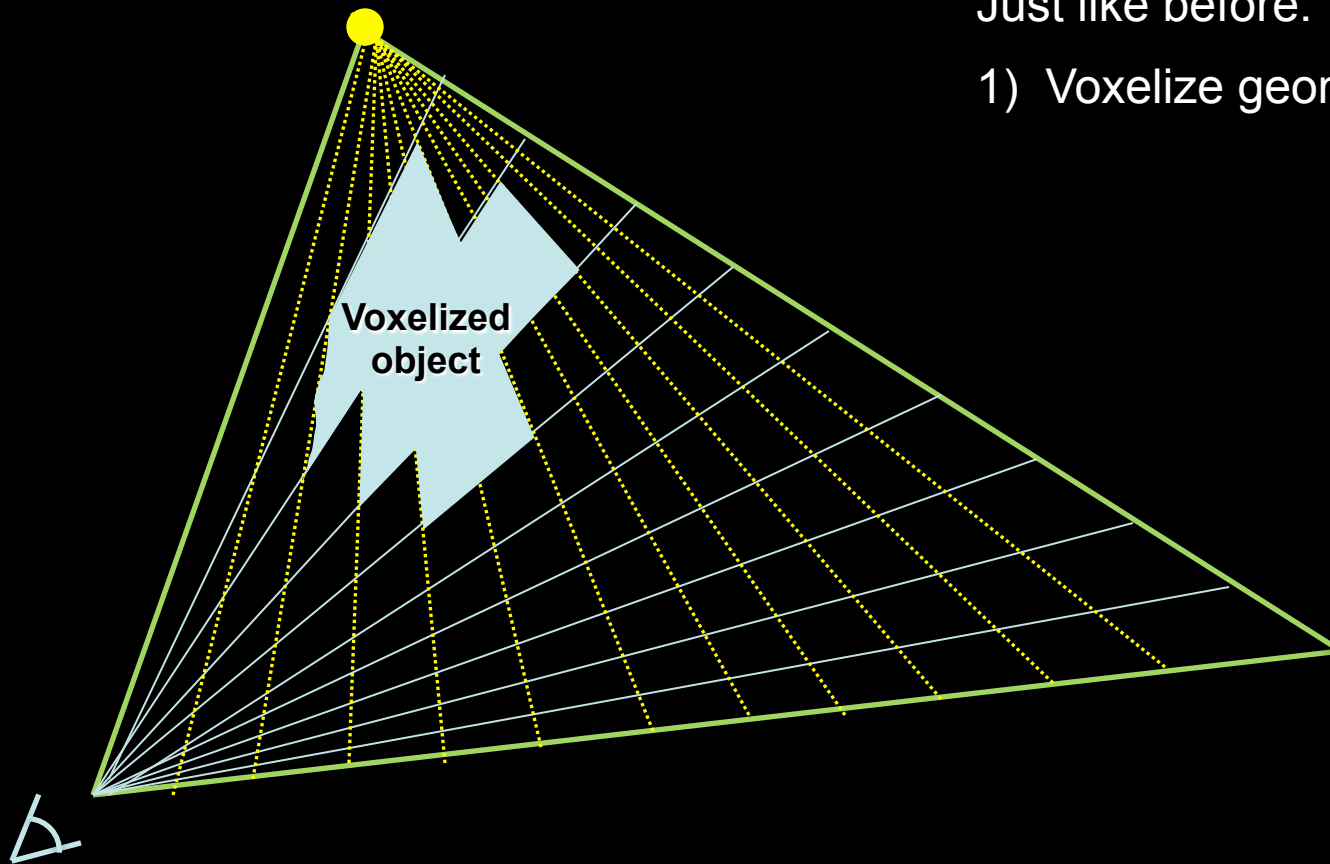Consider an epipolar plane
(i.e., plane of constant θ)
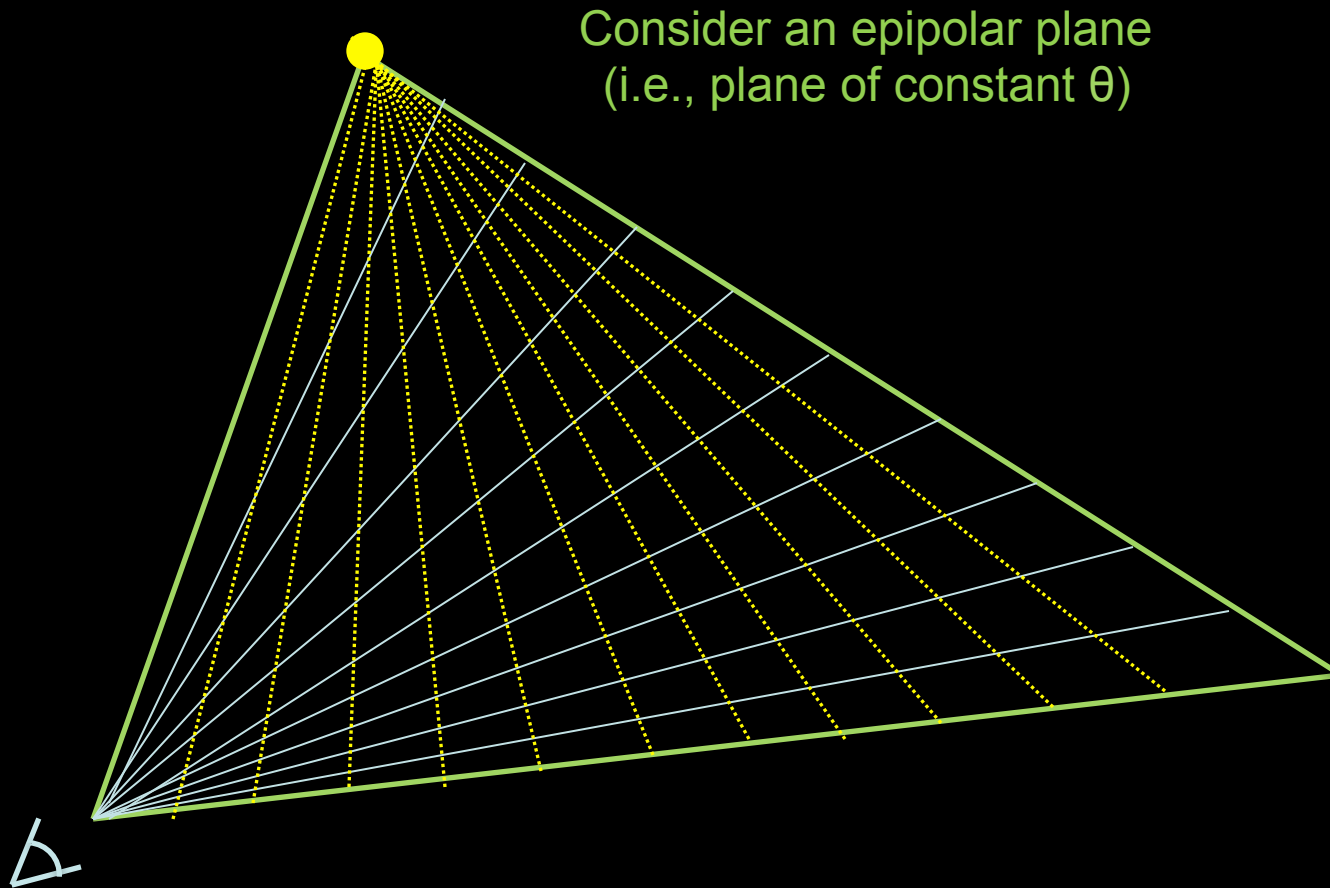
Creating VSVs

Just like before:

1) Voxelize geometry

Voxelized object

Create VSVs

Just like before:

1) Voxelize geometry
2) Scan along light axis

Voxelized object
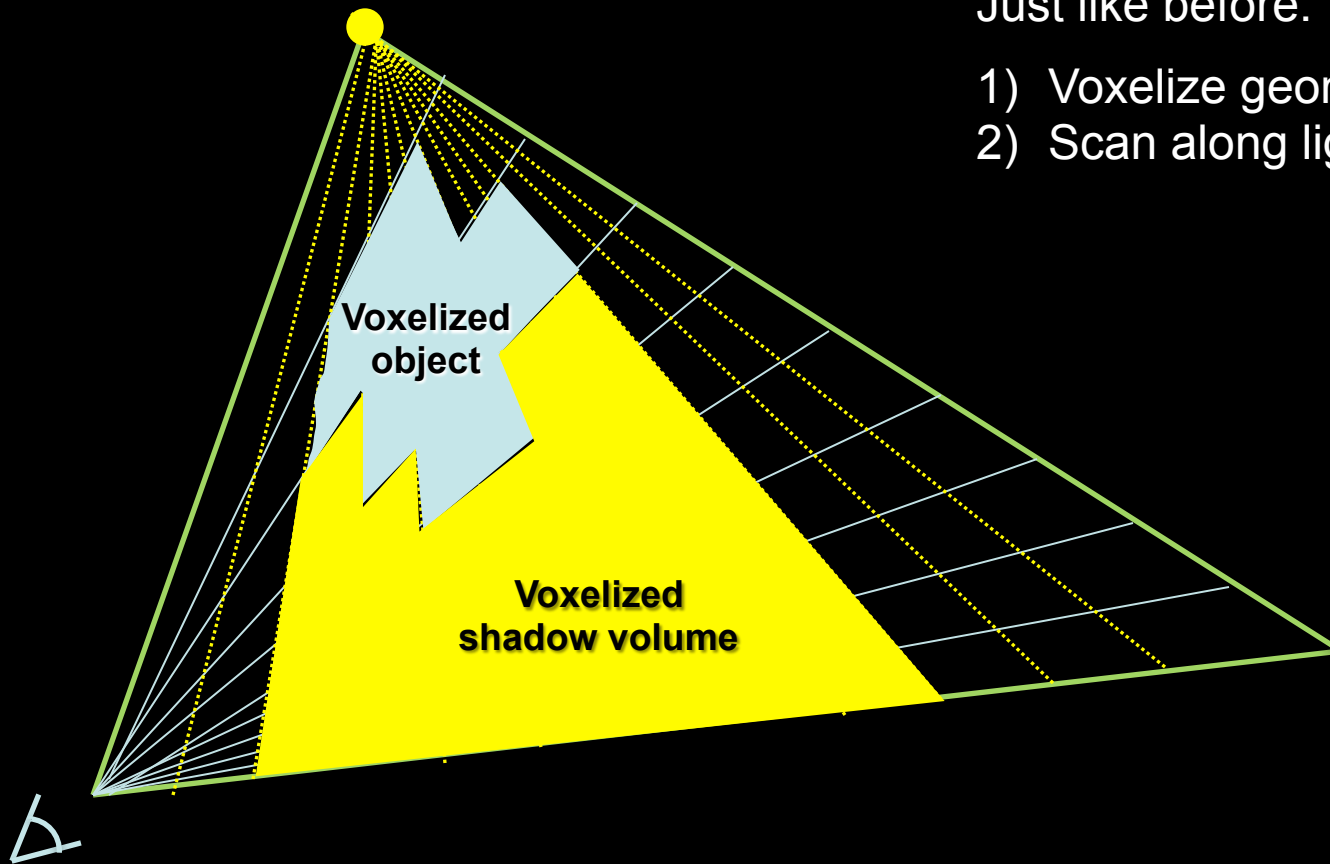
Voxelized shadow volume

Using Voxel Shadow Volumes

Just like before:

1) Voxelize geometry
2) Scan along light axis
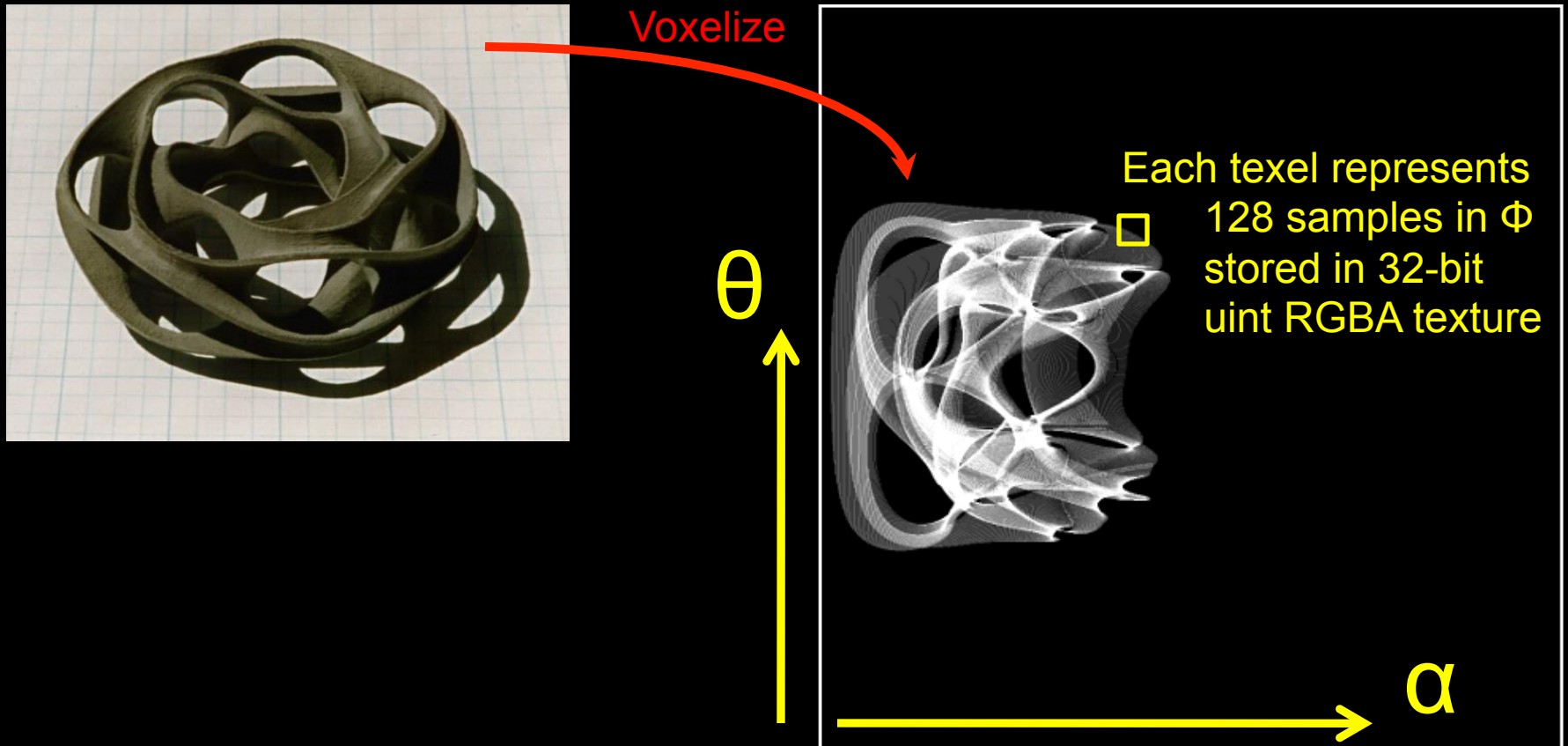3) Look up visibility
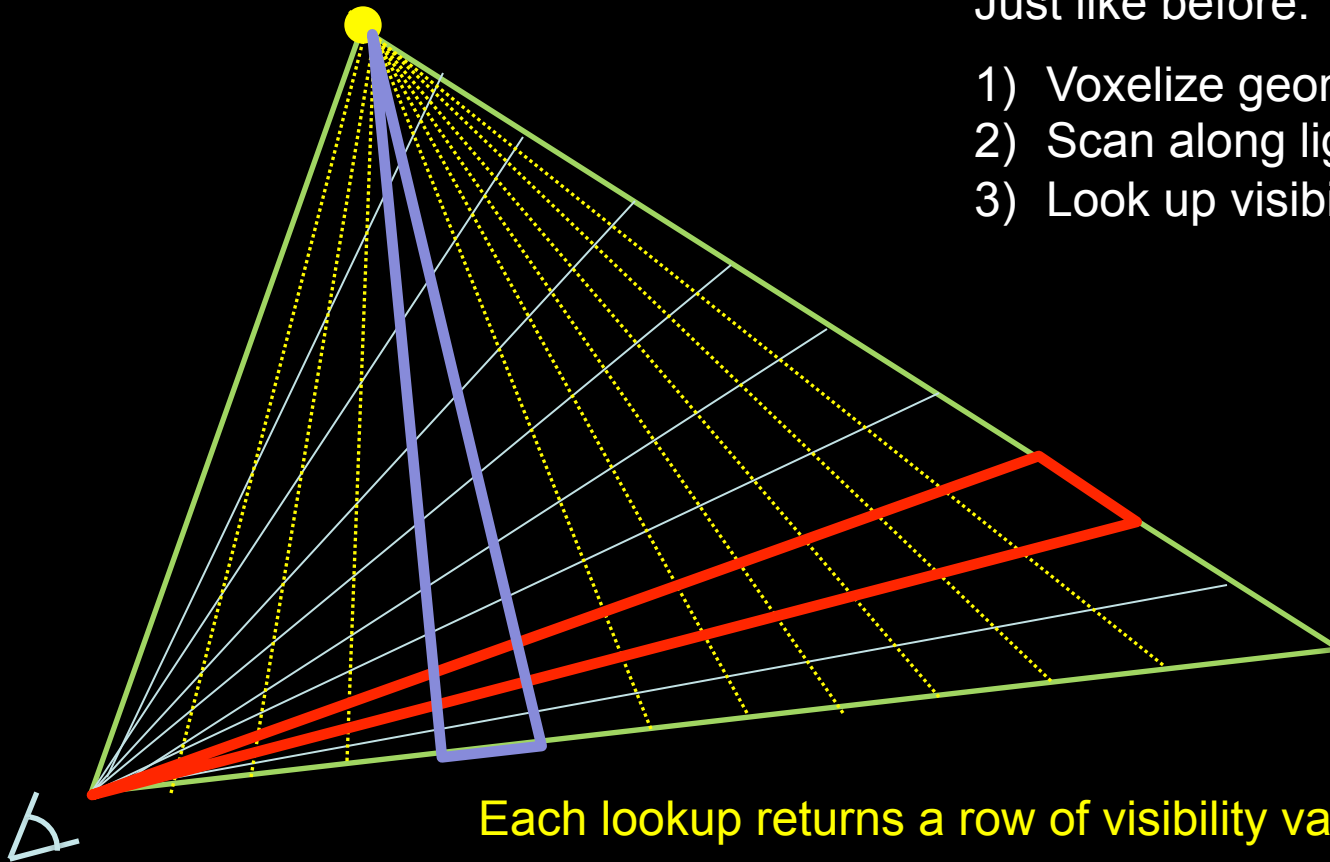
Each lookup returns a row of visibility values (red)

Each bit returned in lookup corresponds to visibility at one sample (blue) along view ray.

VSV Implementation

- Map epipolar space into a 2D GPU texture
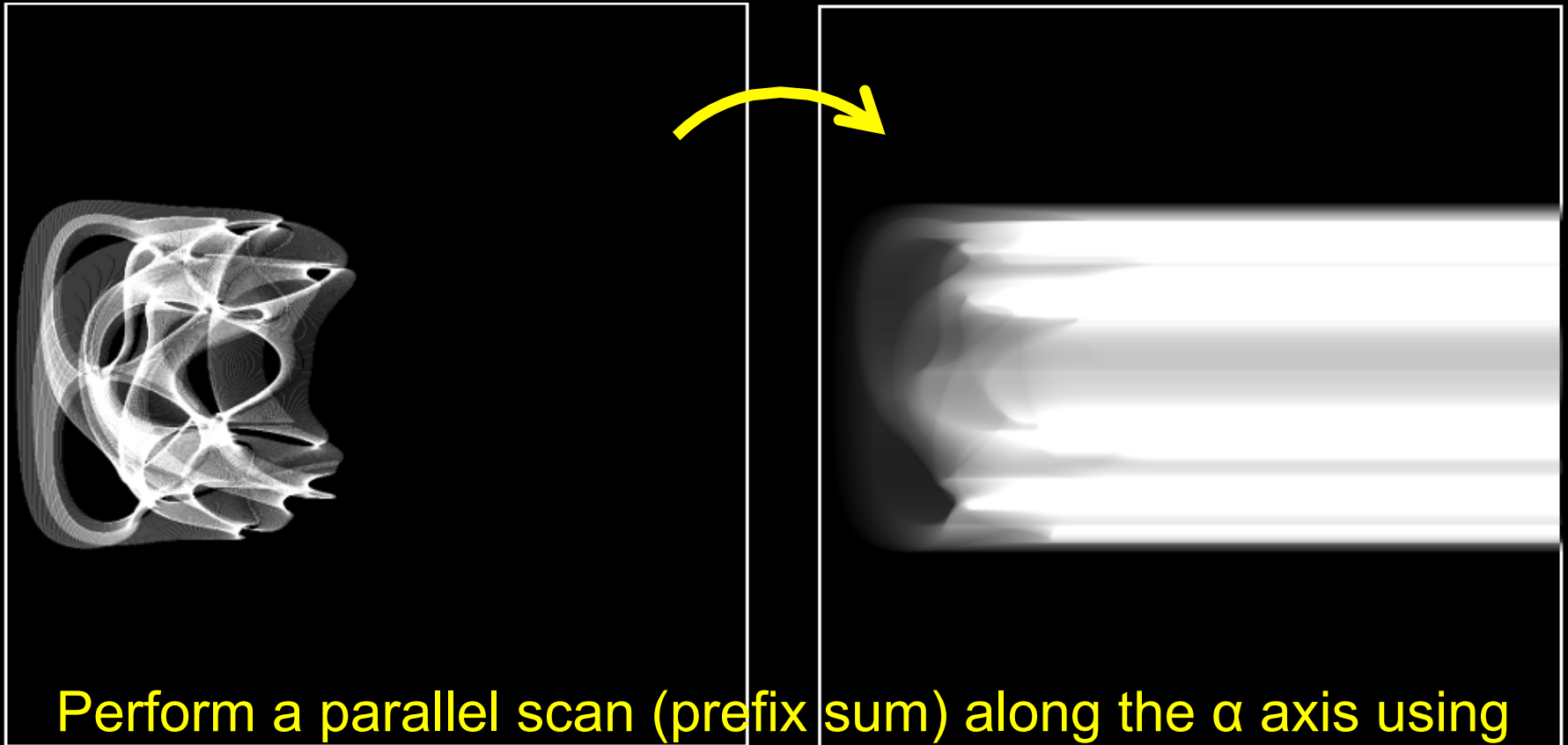

Voxelize

Each texel represents 128 samples in Φ stored in 32-bit uint RGBA texture

θ

α

VSV Implementation
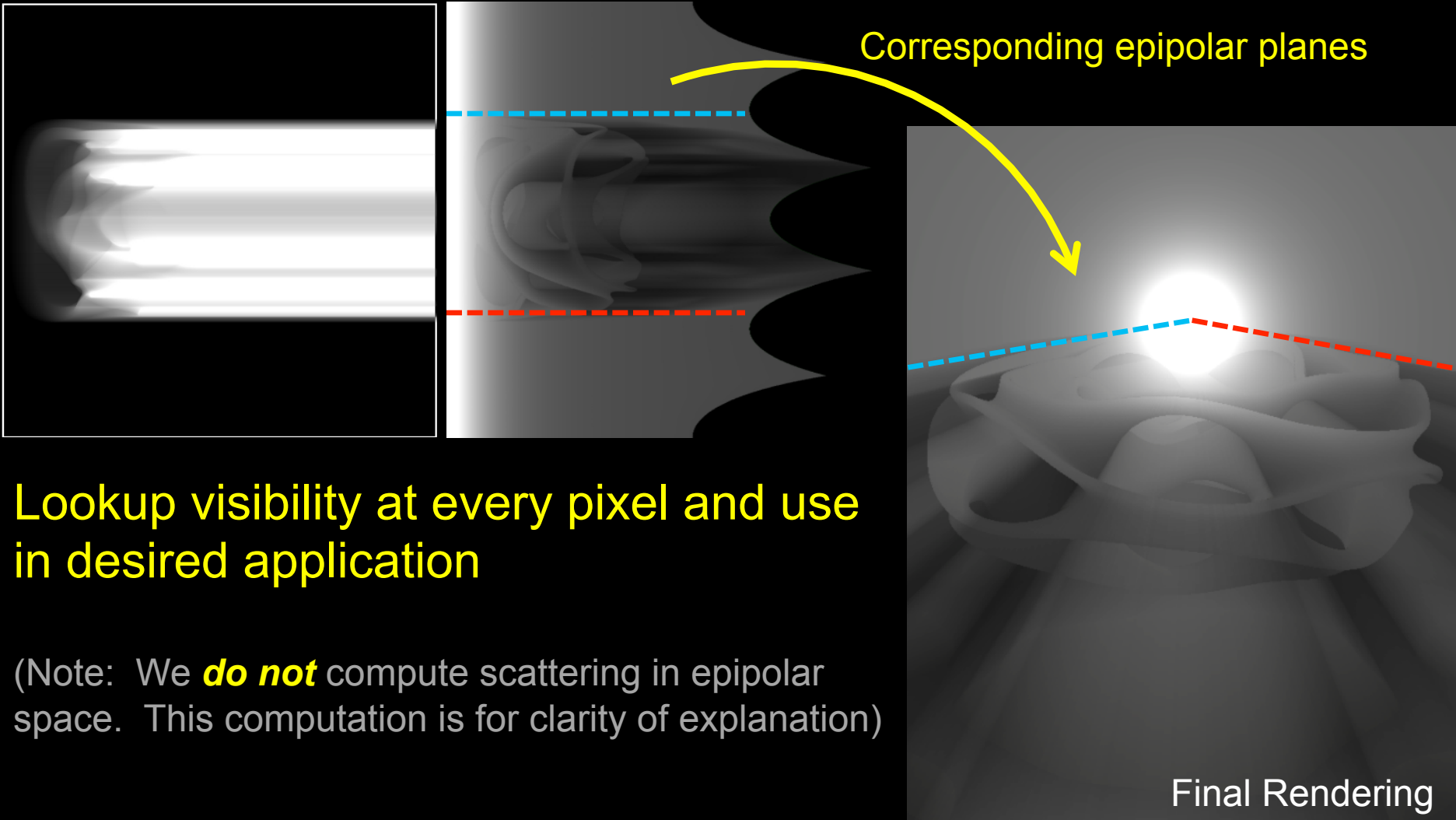

Perform a parallel scan (prefix sum) along the α axis using a bitwise OR, rather than a + operator, to create the VSV.

VSV Implementation

Corresponding epipolar planes

Lookup visibility at every pixel and use in desired application

(Note: We **do not** compute scattering in epipolar space. This computation is for clarity of explanation)

Final Rendering

Tricky Details

- How to voxelize into epipolar space?
  - Paper proposes 3 approaches:
    - [Eisemann 06] screen space voxelization
      - Blazing fast, requires watertight models
      - Requires fixes to handle epipolar singularities
    - [Schwarz 10] conservative voxelization
      - Significantly slower, better quality for thin geometry
      - Requires fixes to handle epipolar singularities
    - Resampling shadow map to epipolar space
      - Similar to approach used by [Chan 11]
      - Blazing fast, naively handles all singularities.

THE UNIVERSITY OF IOWA

Tricky Details

- See paper for details on first 2 methods

- Voxelization via resampling:
  - For each (θ_i, Φ_j) in epipolar space
    - Create the light ray in direction (θ_i, Φ_j)
    - Locate the corresponding shadow map texel
    - Lookup nearest surface, compute its α_{ij} value
    - Set the bit at $(\theta_i, \Phi_j, \alpha_{ij})$

Results

Total frame cost: 10.1 ms

Total frame cost: 6.3 ms

Shadow map: 0.8 ms
Voxelization: 3.1 ms   Other rendering: 3.5 ms
Parallel scan: 2.7 ms

Shadow map: 0.3 ms
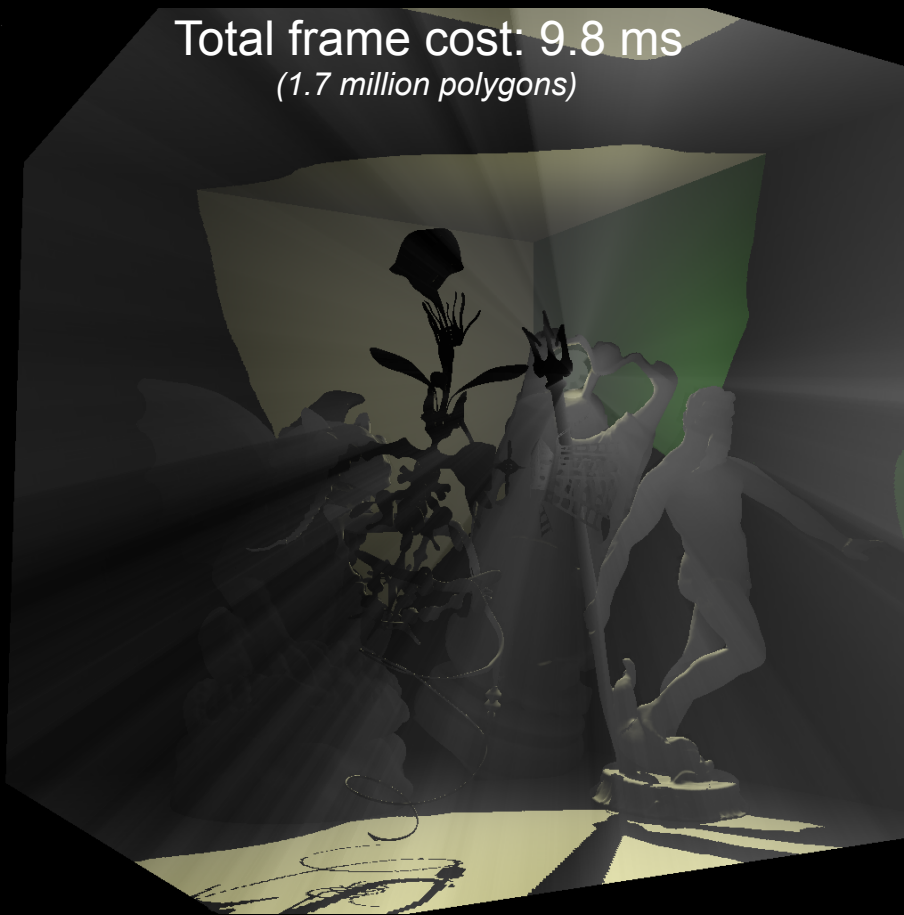Voxelization: 1.4 ms   Other rendering: 1.9 ms
Parallel scan: 2.7 ms

Results

Total frame cost: 9.8 ms
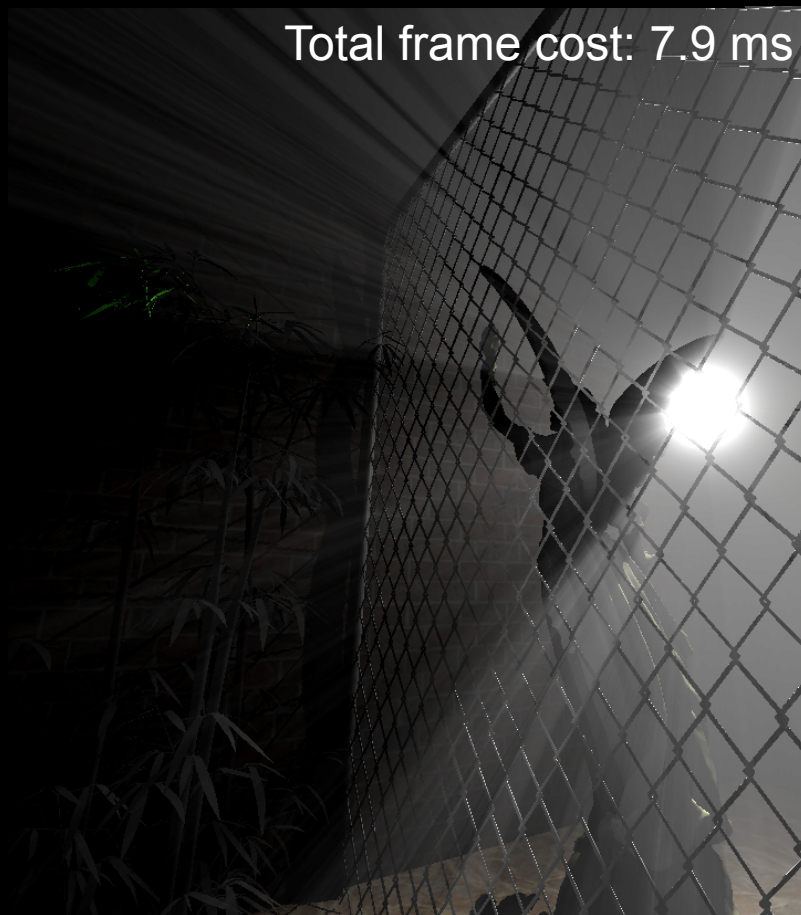*(1.7 million polygons)*

Total frame cost: 7.9 ms

Shadow map: 2.6 ms
Voxelization: 1.9 ms   Other rendering: 2.6 ms
Parallel scan: 2.7 ms

Shadow map: 1.1 ms
Voxelization: 2.0 ms   Other rendering: 2.1 ms
Parallel scan: 2.7 ms

Video / Demo

Aliasing Issues

- Aliasing occurs, as with all sampling
  - Can focus on important samples
    - Paper talks about selecting good α, Φ, and θ ranges
  - Can do adaptive sampling
    - Paper explores briefly, but more work needed
  - Can brute force by adding more samples
    - Our performance → 512 x 2048 x 512 volumes
    - Parallel scan decreases roughly linearly in size
  - Filtering in epipolar space
    - Perhaps similar to PCF, needs more exploration.

Aliasing Issues

- Our worst aliasing occurs @ singularity
  - Geometry seen occluding light
 - Geometry seen behind light (less problematic)

Summary

- Proposed *voxelized shadow volumes* for visibility queries
  - Voxelize using a new epipolar space parameterization
  - Prefix sum along light rays in epipolar space
  - Gives a discrete sampling of shadow volumes
  - Lookup 128 binary visibility samples with single texture lookup

- Advantages
  - Decouples geometric complexity & visibility cost
  - Cache coherent lookups
  - Drop into existing participating media techniques for visibility

- Disadvantages
  - Some aliasing near singularity
  - Some care in implementation details for robustness (see paper)

THE UNIVERSITY OF IOWA

Acknowledgements

- People who have listened to this idea in progress
  - My students (Greg Nichols, Rajeev Penmatsa, and Thomas Hansen) and too many others to list…

- Funding sources:
  - DARPA: HR001-09-1-0027
  - ARO: W911NF-10-1-0338

- Hardware donations:
  - NVIDIA Corporation

THE
UNIVERSITY
OF IOWA
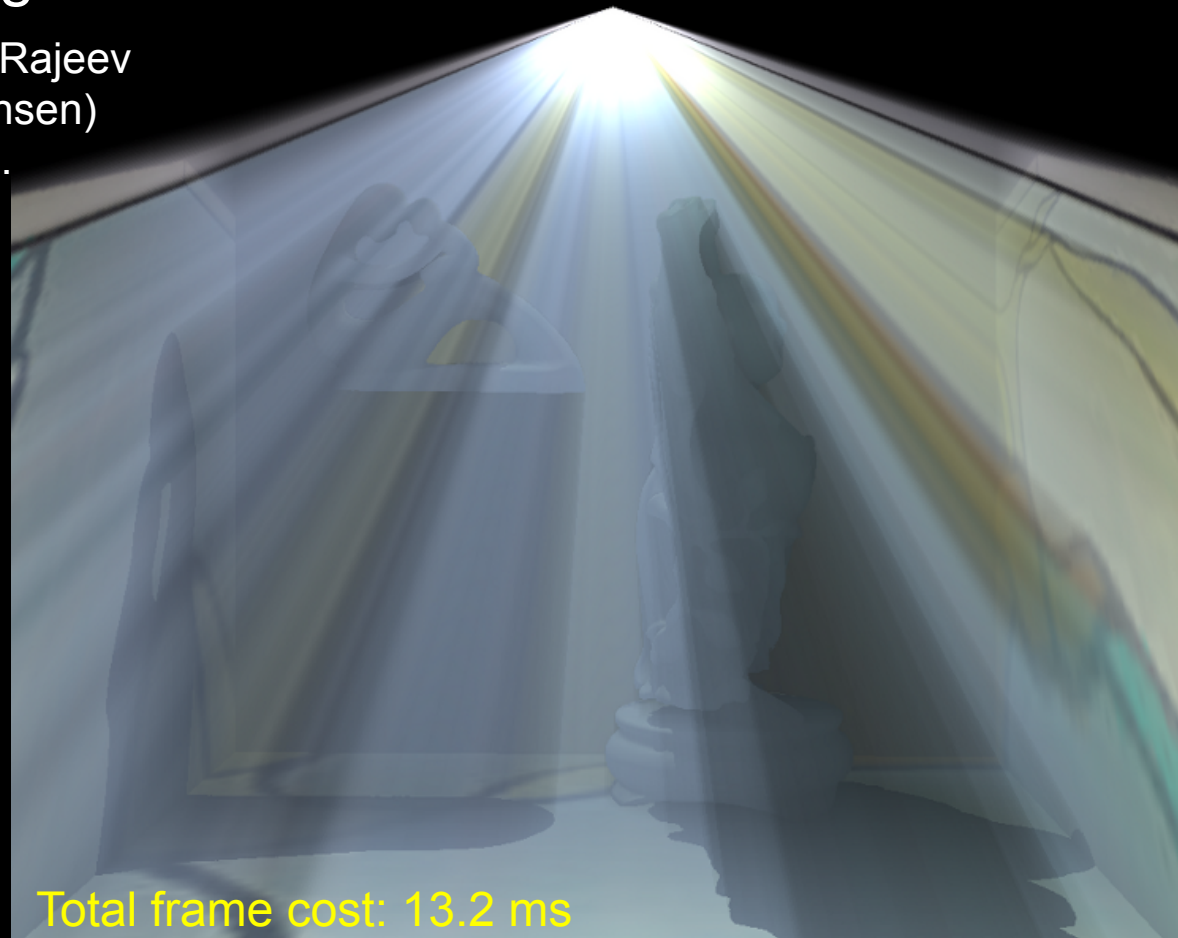
Total frame cost: 13.2 ms

Comparison To Ground Truth

VSV visibility

Ray traced visibility