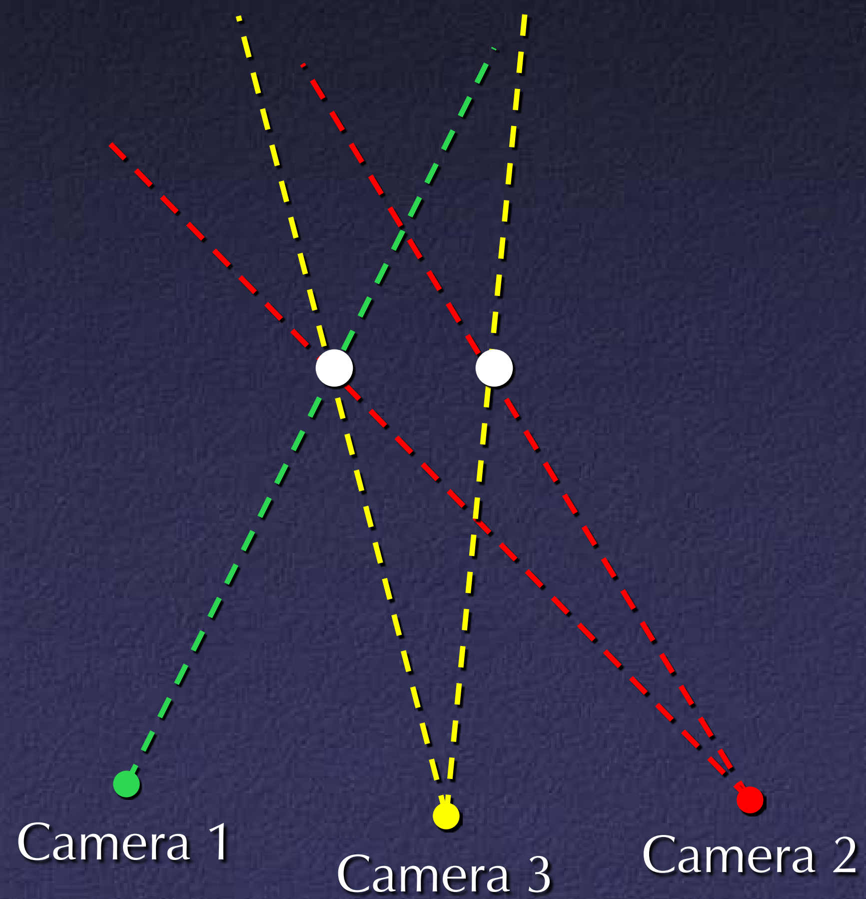
Multiview Reconstruction

Why More Than 2 Views?

- Baseline
  - Too short – low accuracy
  - Too long – matching becomes hard

Why More Than 2 Views?

- Ambiguity with 2 views



Trinocular Stereo

- Straightforward approach to eliminate bad correspondences
  - Pick 2 views, find correspondences
  - For each matching pair, reconstruct 3D point
  - Project point into 3^{rd} image
  - If can't find correspondence near predicted location, reject

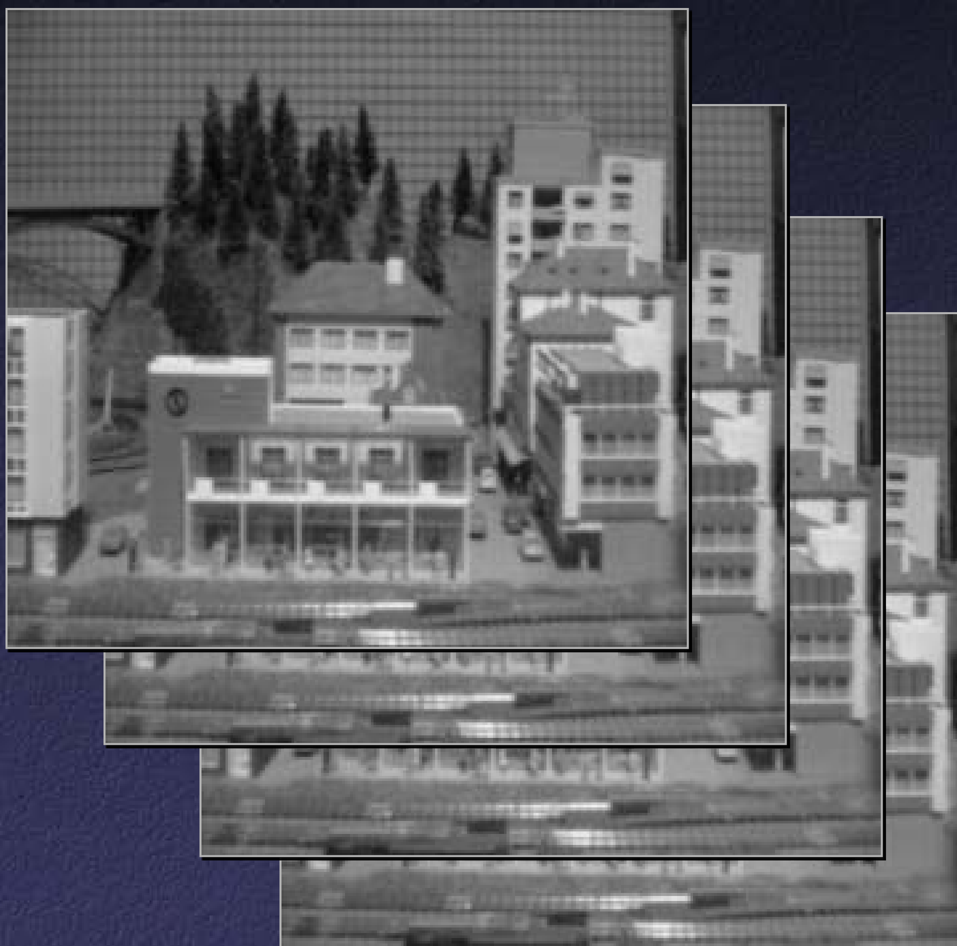
Trinocular Stereo

- Trifocal geometry: relations between points in three camera views
- Trifocal tensor: analogue of essential matrix
  - 3x3x3 trilinear tensor (3D cube of numbers)
  - Given lines in 2 views, predict lines in the 3rd

Multibaseline Stereo

- Slightly different algorithm for *n* cameras:
- Pick one reference view
- For each candidate depth
  - Compute sum of squared differences to all other views, assuming correct disparity for view
- Resolves ambiguities: only correct depths will "constructively interfere"

Multibaseline Stereo



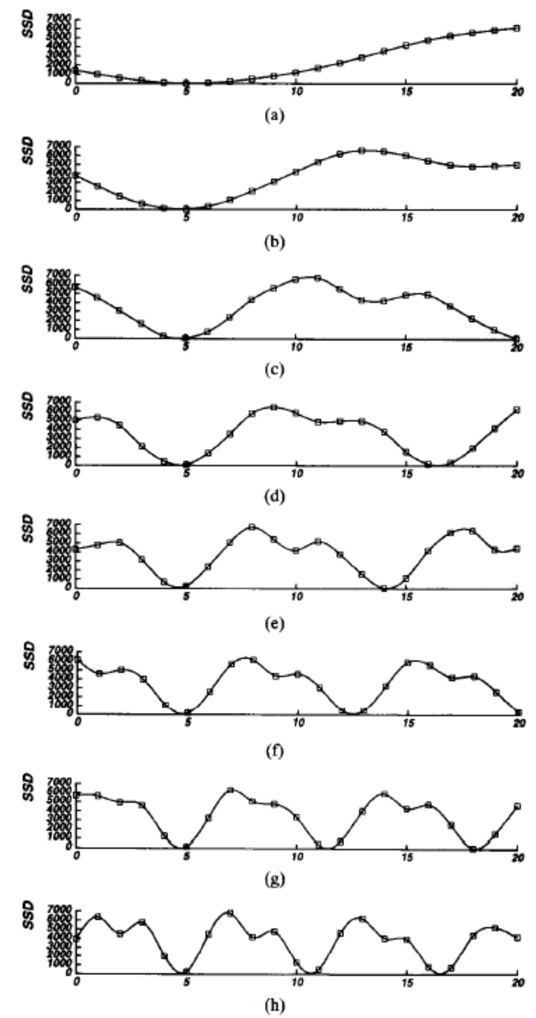
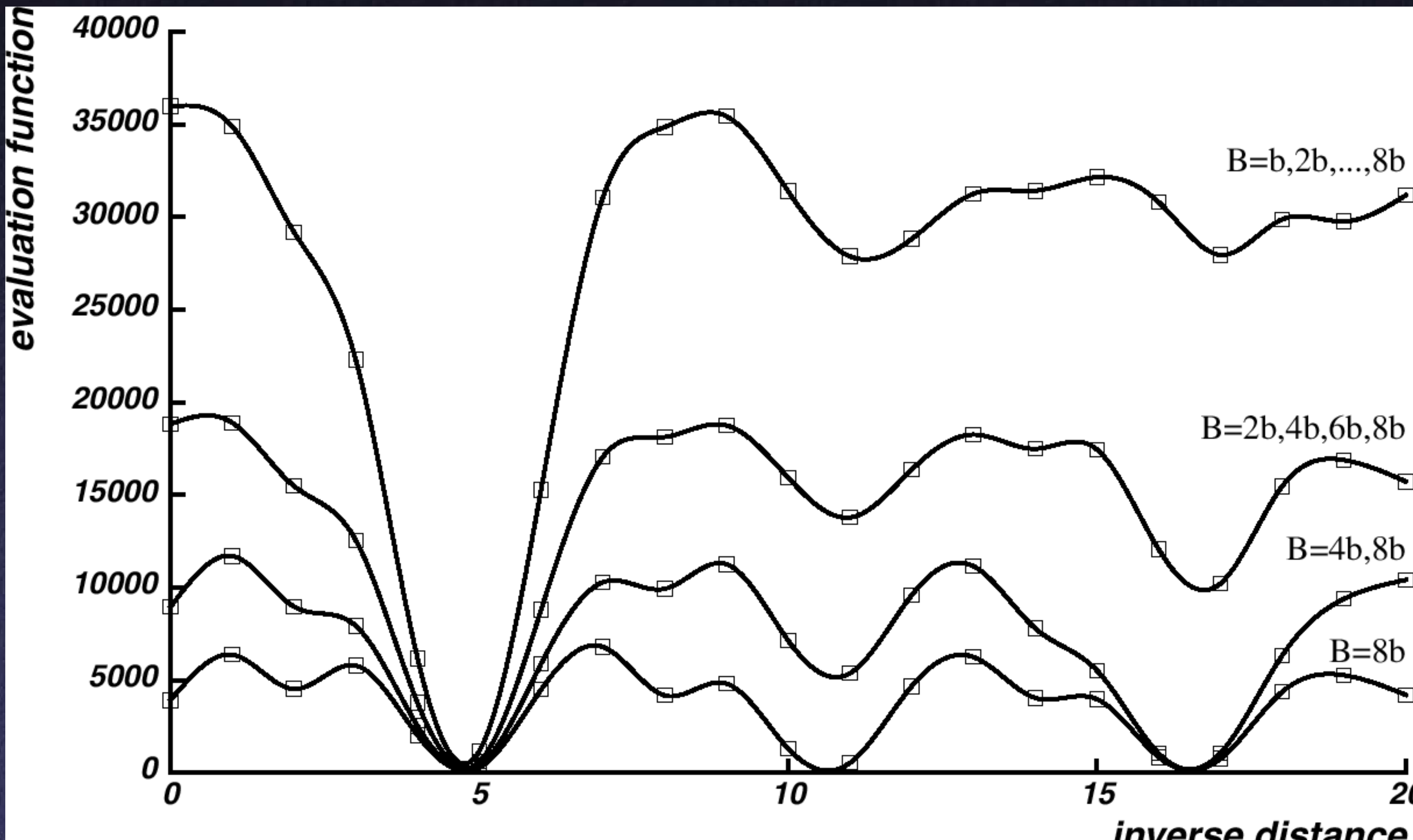


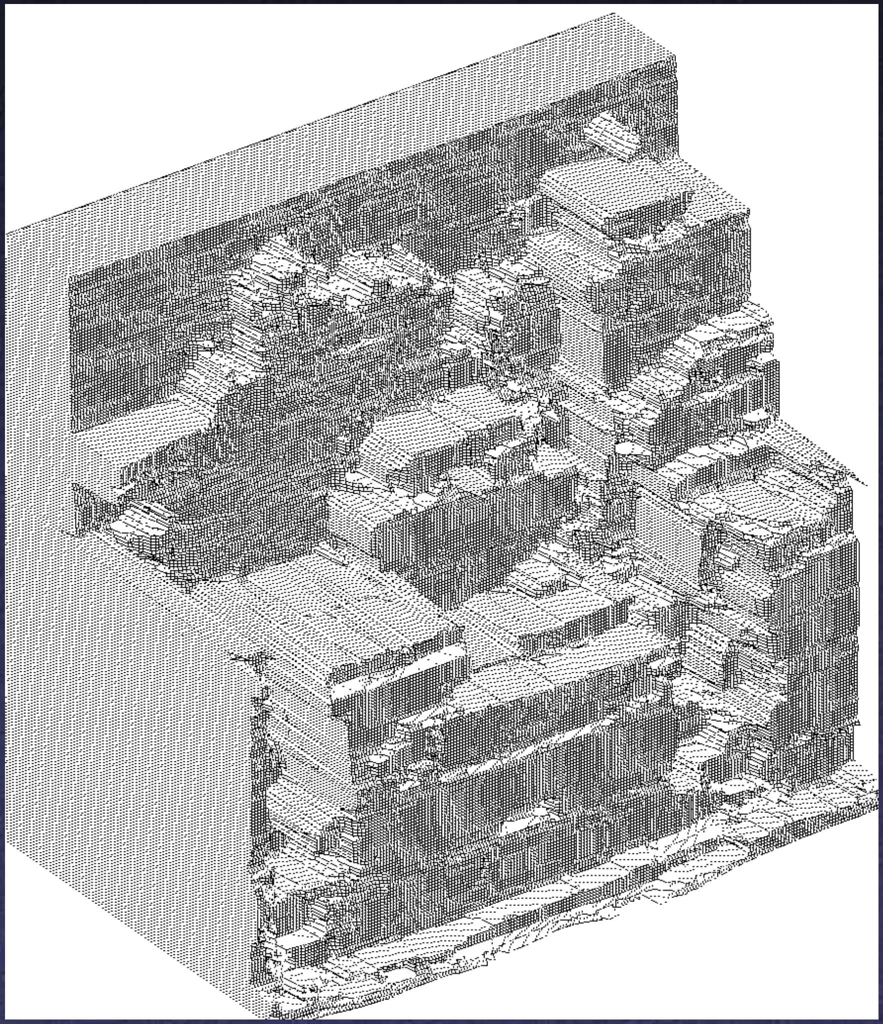
Fig. 5. SSD values versus inverse distance: (a) $B = b$; (b) $B = 2b$; (c) $B = 3b$; (d) $B = 4b$; (e) $B = 5b$; (f) $B = 6b$; (g) $B = 7b$; (h) $B = 8b$. The horizontal axis is normalized such that $8bF = 1$.

Multibaseline Stereo

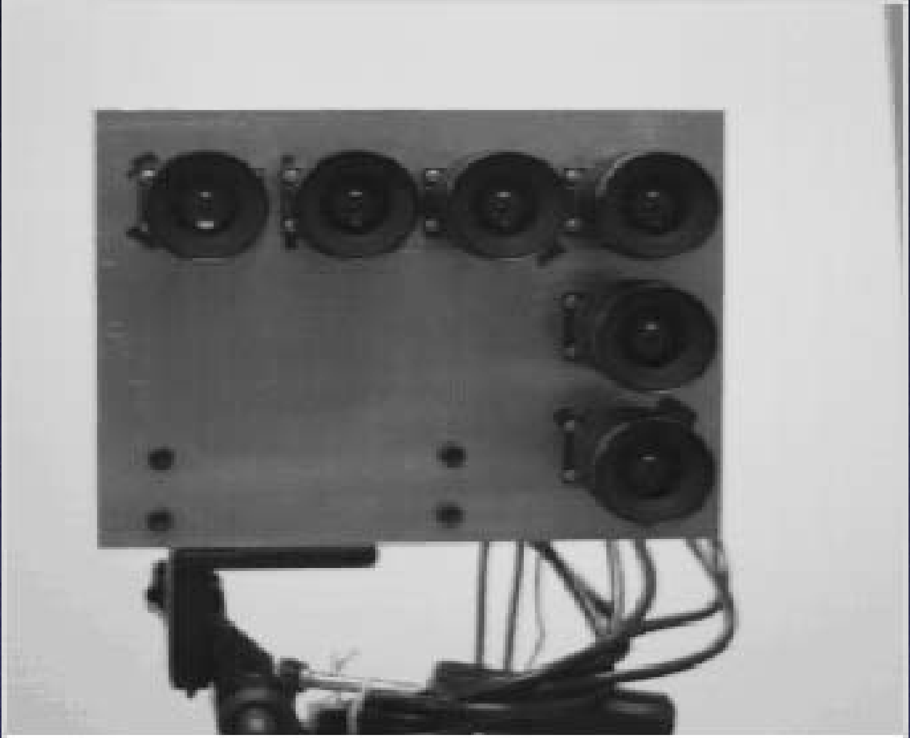


[Okutami & Kanade]

Multibaseline Stereo Reconstruction



Multibaseline Stereo



(a)

(b)

Figure 7: The CMU Video-Rate Stereo Machine Prototype System: (a) camera head; (b) processor boards

Problems with Multibaseline Stereo

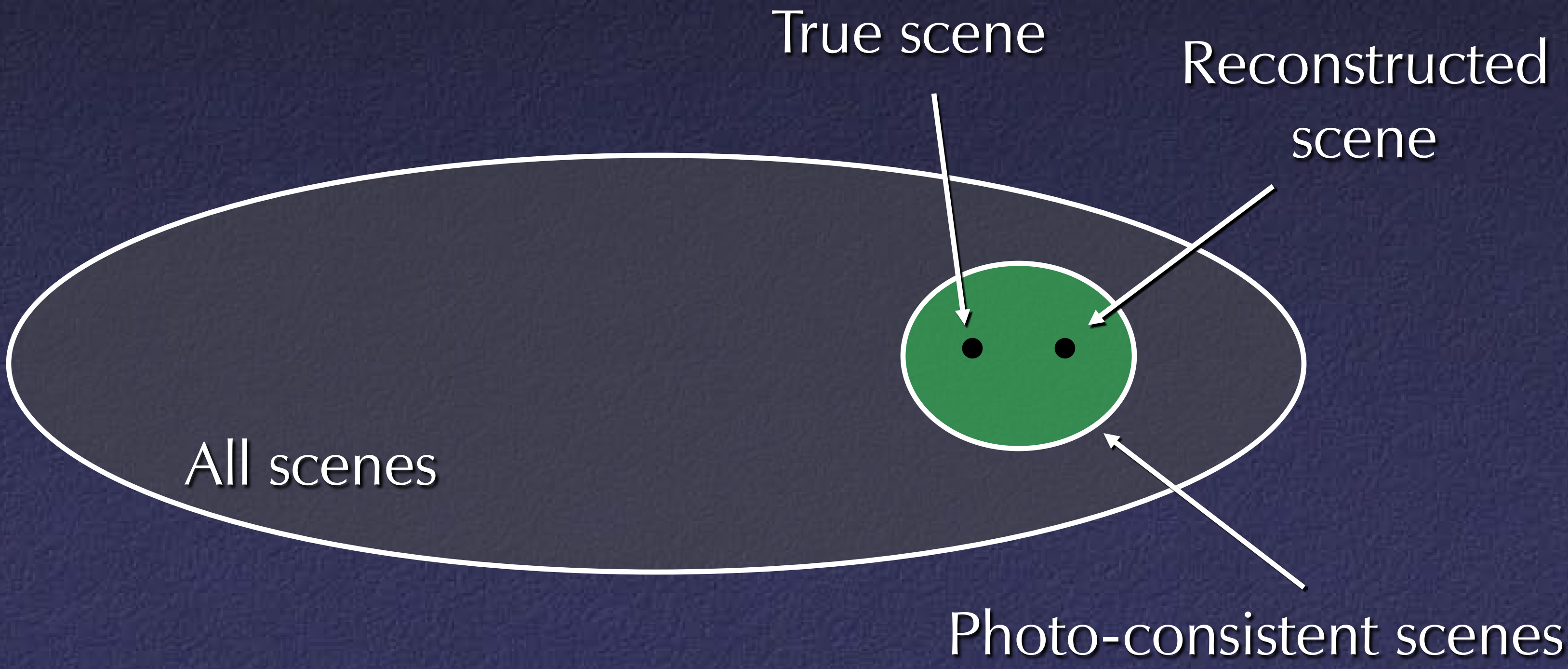
- Have to pick a reference view
- Occlusion
  - With many cameras / large baseline, occlusion becomes likely
  - Contributes incorrect values to error function

Volumetric Multiview Approaches

- Goal: find a model consistent with images
- "Model-centric" (vs. image-centric)
- Typically use discretized volume (voxel grid)
- For each voxel, compute occupied / free
  (for some algorithms, also color, etc.)

Photo Consistency

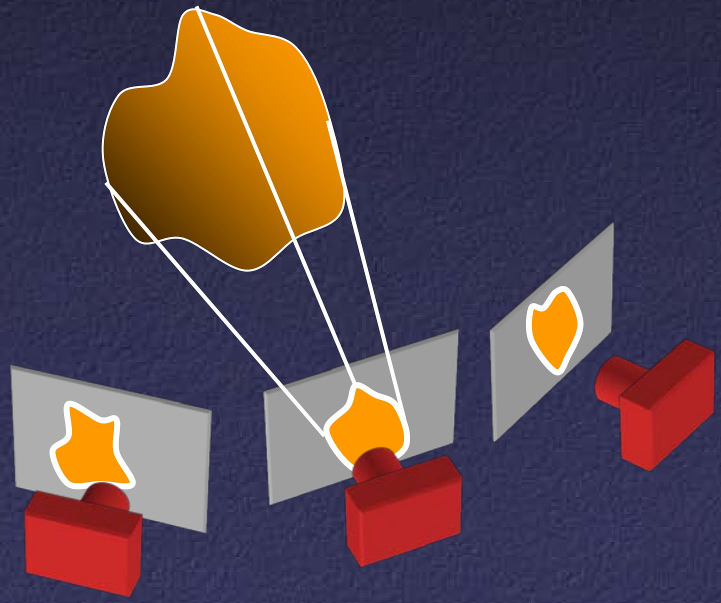
- Result: not necessarily correct scene
- Many scenes produce the same images



Silhouette Carving

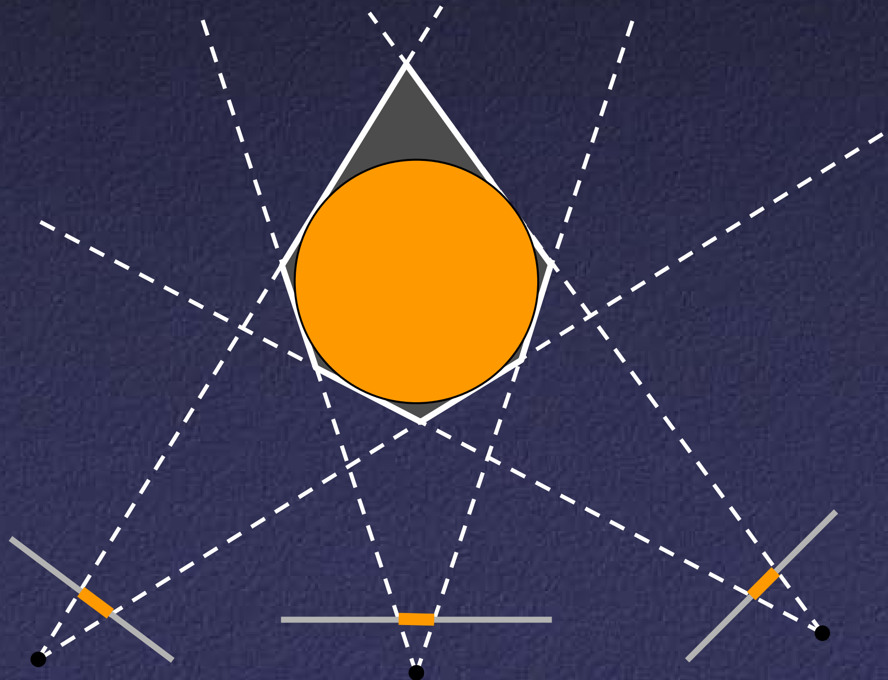
- Find silhouettes in all images
- Exact version:
  - Back-project all silhouettes, find intersection

Binary Images →



Silhouette Carving

- Find silhouettes in all images
- Exact version:
  - Back-project all silhouettes, find intersection



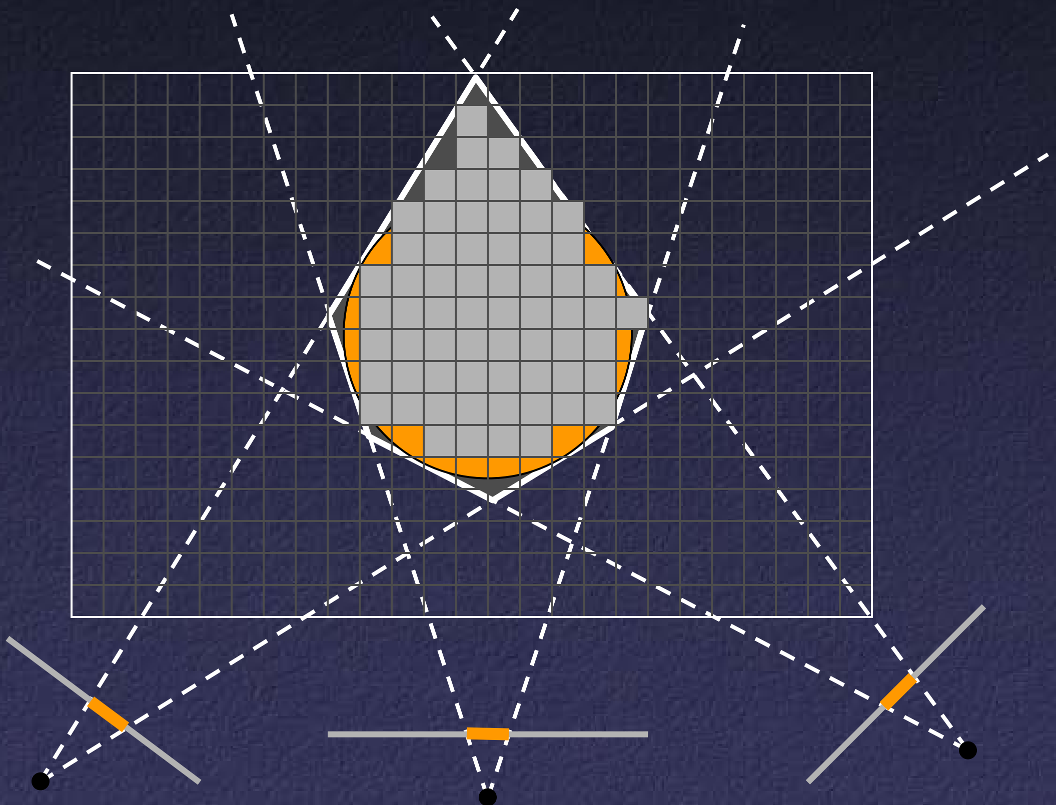
Silhouette Carving

- Limit of silhouette carving is *visual hull* or *line hull*
- Complement of lines that don't intersect object
- In general not the same as object
  - Can't recover "pits" in object
- Not the same as convex hull

Silhouette Carving

- Discrete version:
  - Loop over all voxels in some volume
  - If projection into images lies inside all silhouettes, mark as occupied
  - Else mark as free

Silhouette Carving



Voxel Coloring

- Seitz and Dyer, 1997
- In addition to free / occupied, store color at each voxel
- Explicitly accounts for occlusion

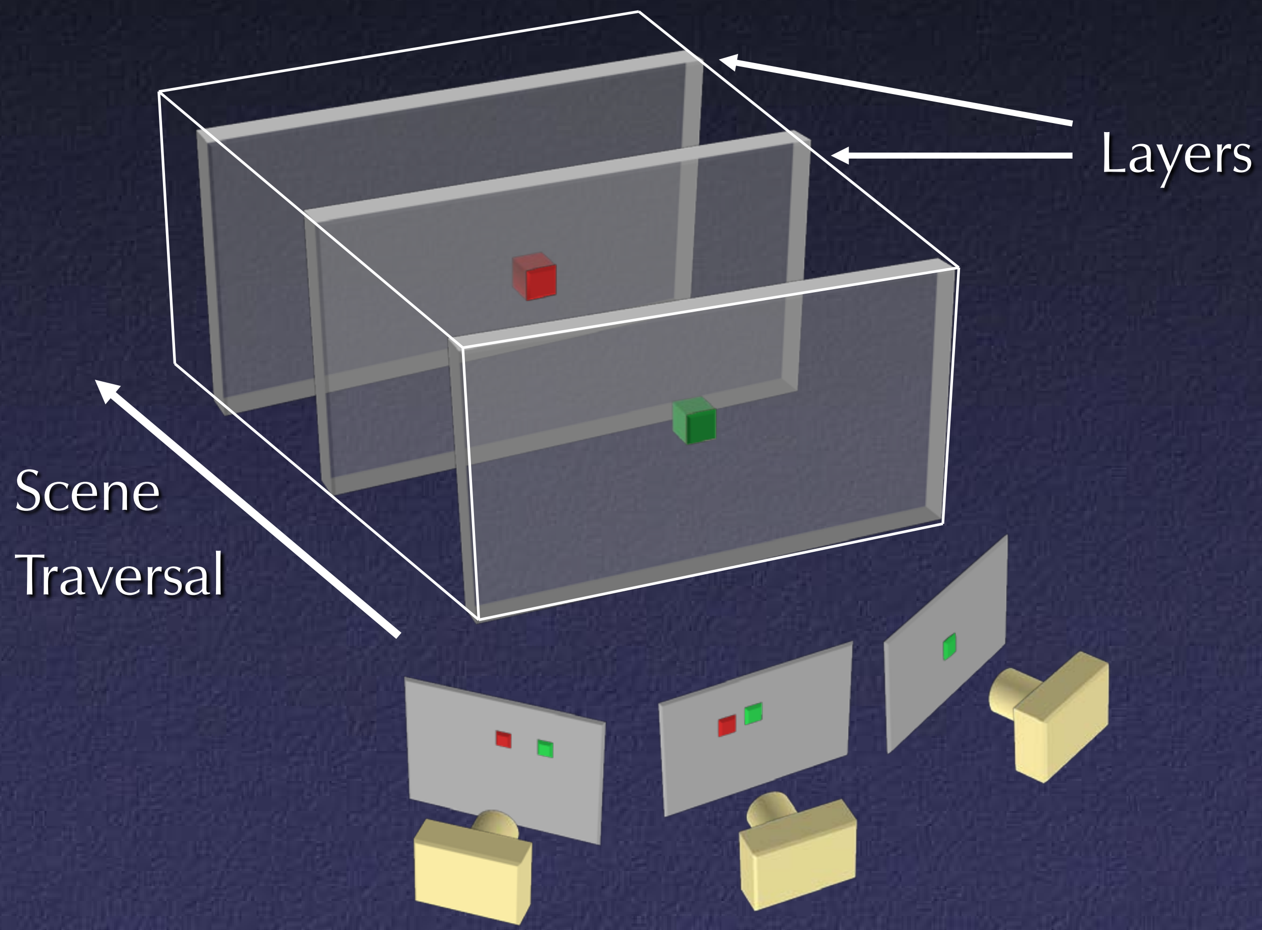
Voxel Coloring

- Basic idea: sweep through a voxel grid
  - Project each voxel into each image in which it is visible
  - If colors in images agree, mark voxel with color
  - Else, mark voxel as empty
- Agreement of colors based on comparing standard deviation of colors to threshold

Voxel Coloring and Occlusion

- Problem: which voxels are visible?
- Solution, part 1: constrain camera views
  - When a voxel is considered, necessary occlusion information must be available
  - Sweep occluders before occludees
  - Constrain camera positions to allow this sweep

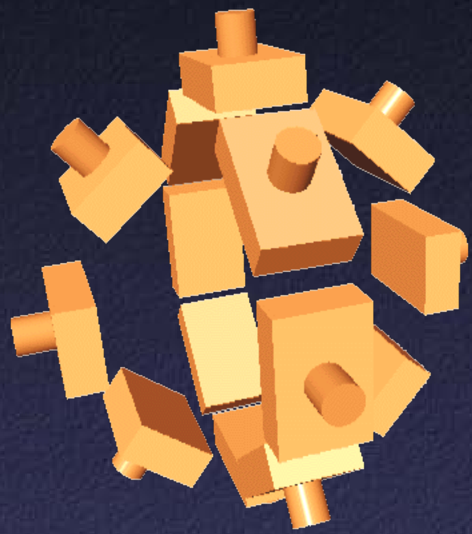
Voxel Coloring Sweep Order



Seitz

Voxel Coloring Camera Positions

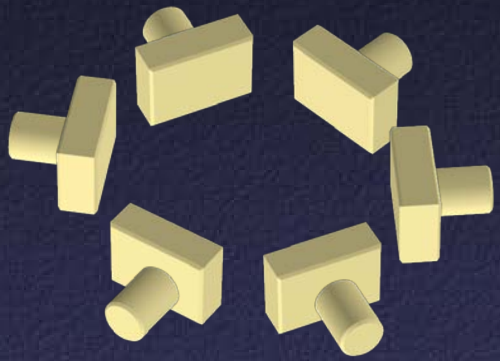
Inward-looking
Cameras above scene



Outward-looking
Cameras inside scene

Seitz

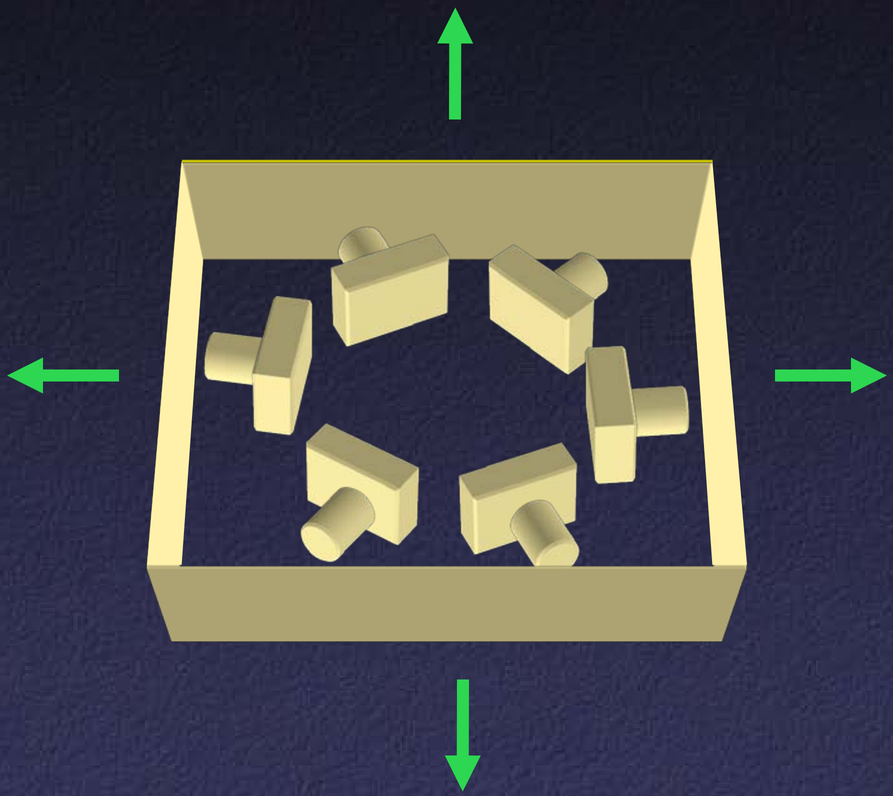
Panoramic Depth Ordering



- Cameras oriented in many different directions
- Planar depth ordering does not apply

Seitz

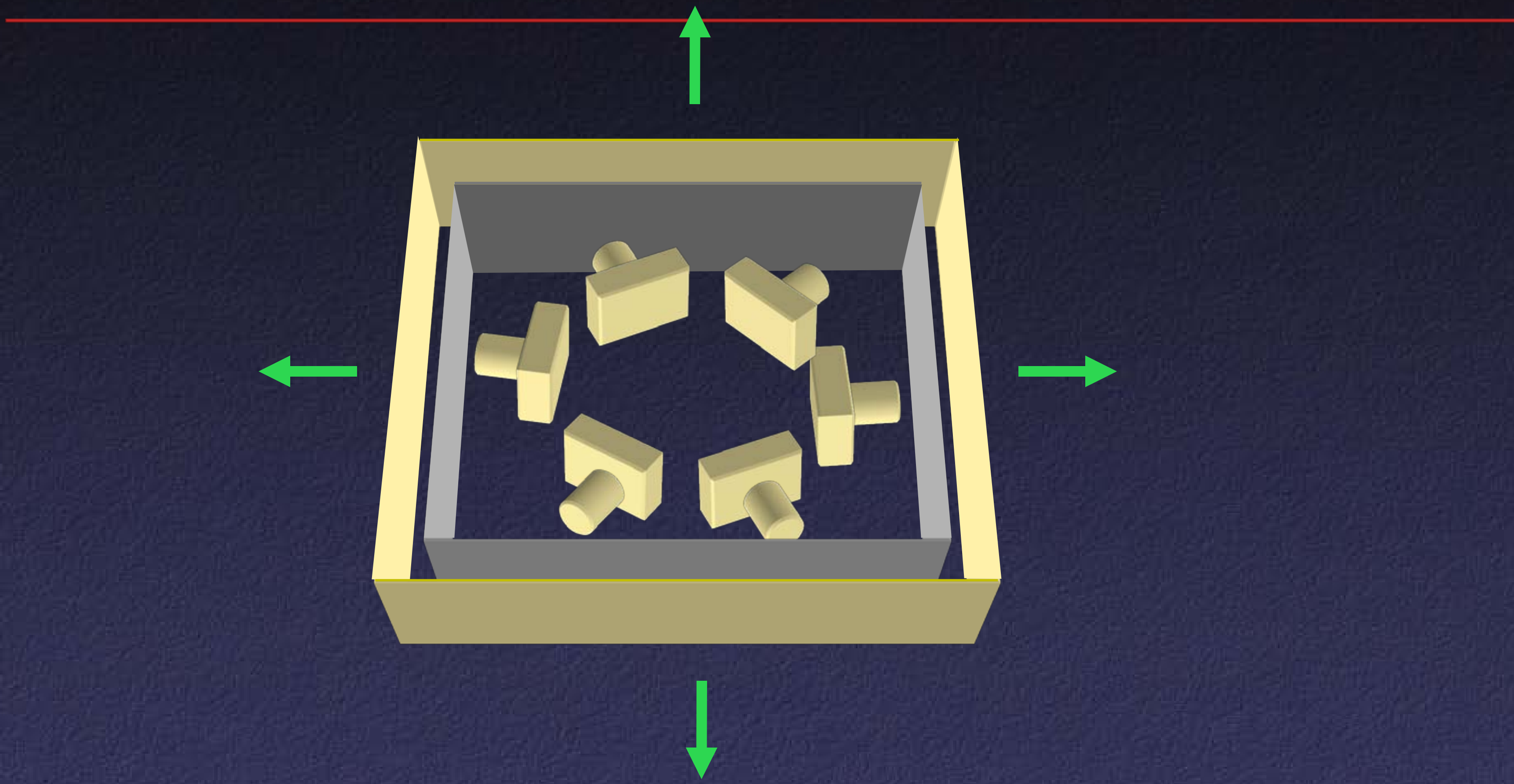
Panoramic Depth Ordering



Layers radiate outwards from cameras

Seitz

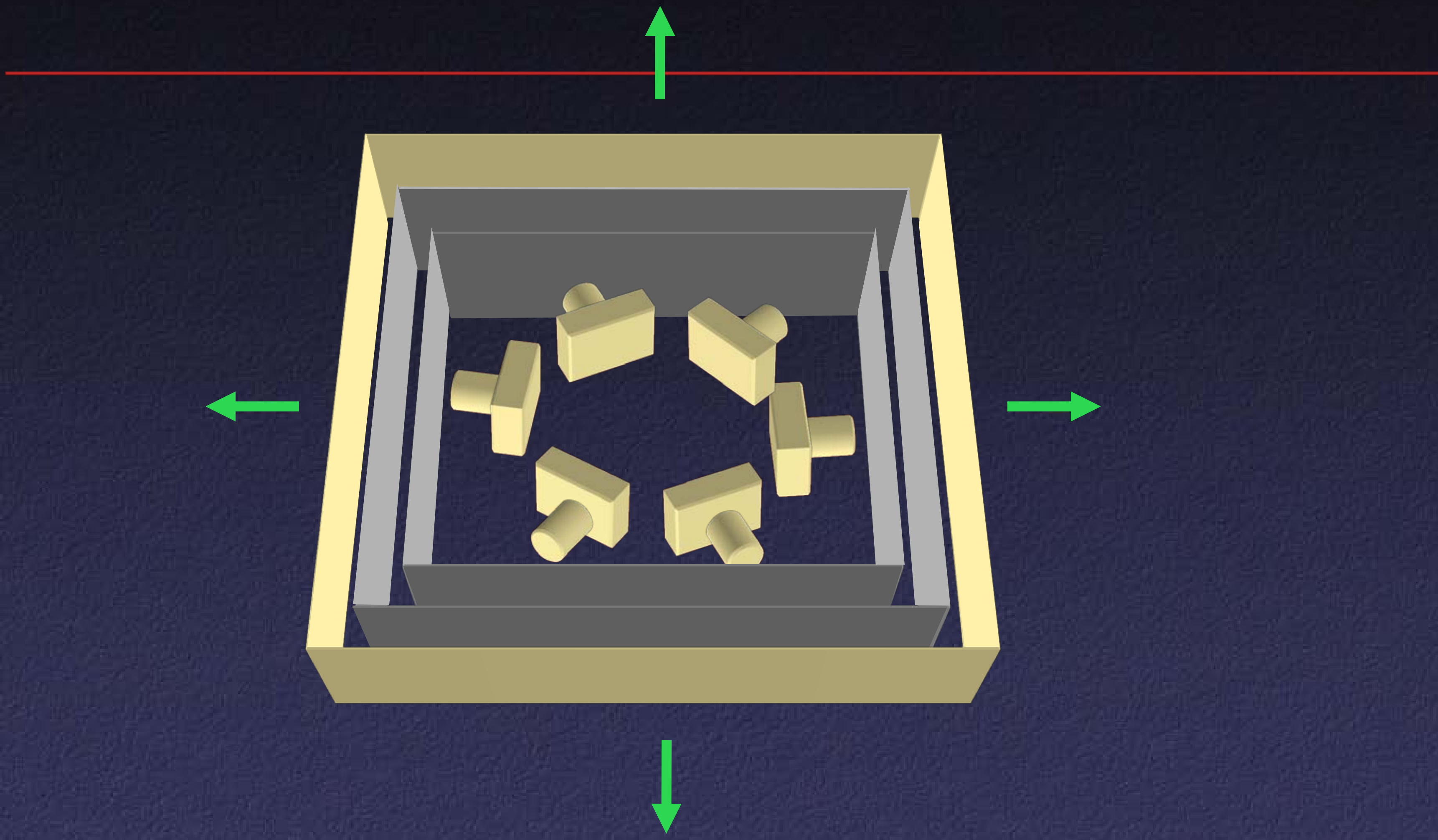
Panoramic Depth Ordering



Layers radiate outwards from cameras

Seitz

Panoramic Depth Ordering



Layers radiate outwards from cameras

Seitz

Voxel Coloring and Occlusion

- Solution, part 2: per-image mask of which pixels have been used
  - Each pixel only used once
  - Mask filled in as sweep progresses

Image Acquisition

Selected Dinosaur Images

Selected Flower Images

- Calibrated Turntable
- 360° rotation (21 images)

Seitz

Voxel Coloring Results

Dinosaur Reconstruction
72 K voxels colored
7.6 M voxels tested
7 min. to compute
on a 250MHz SGI

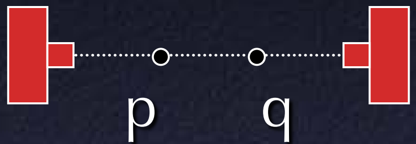
Flower Reconstruction
70 K voxels colored
7.6 M voxels tested
7 min. to compute
on a 250MHz SGI

Seitz

Voxel Coloring Results

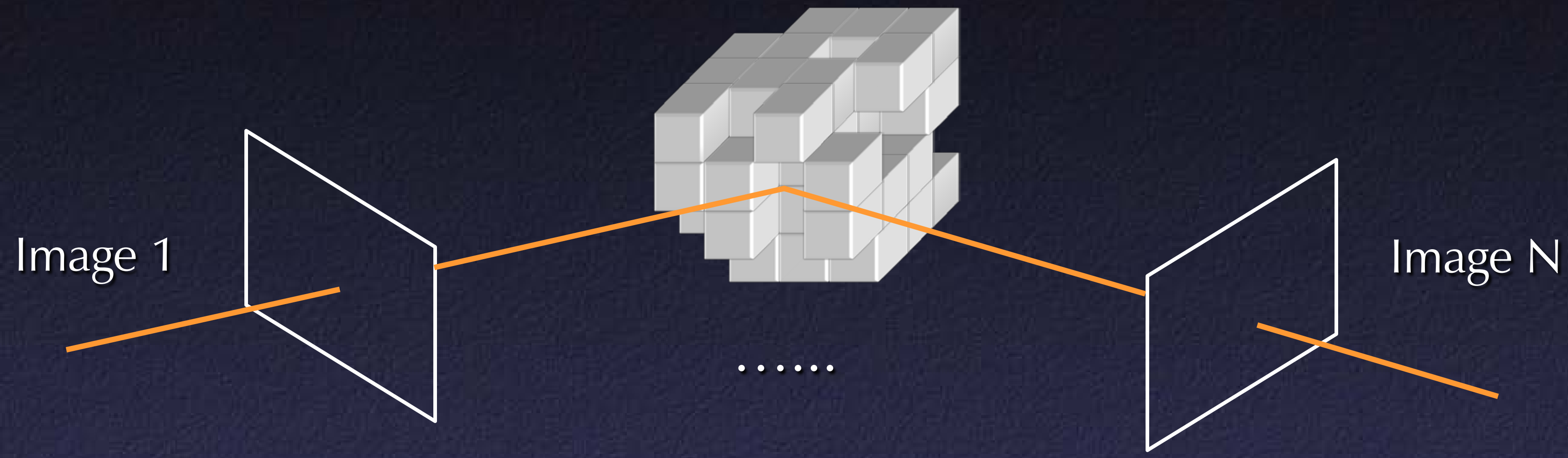
- With texture: good results
- Without texture: regions tend to “bulge out”
  - Voxels colored at earliest time at which projection into images is consistent
  - Model good for re-rendering: image will look correct for viewpoints near the original ones

Limitations of Voxel Coloring



- A view-independent depth order may not exist
- Need more powerful general-case algorithms
  - Unconstrained camera positions
  - Unconstrained scene geometry/topology

Space Carving



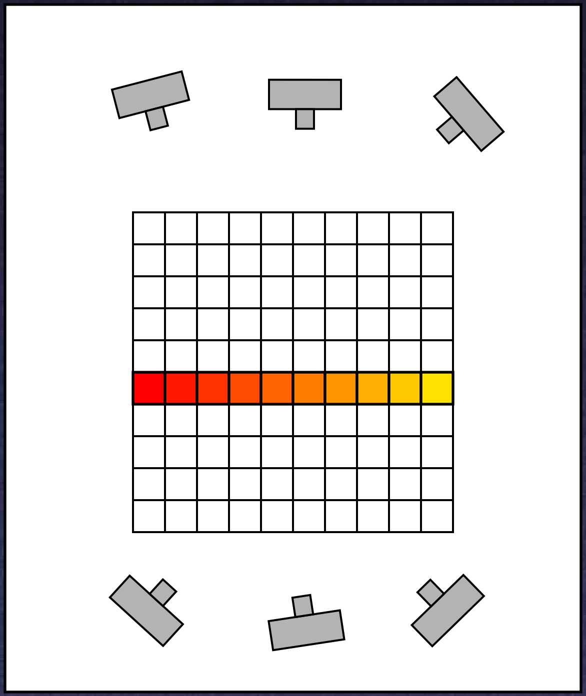
Initialize to a volume V containing the true scene
Choose a voxel on the current surface
Project to visible input images
Carve if not photo-consistent
Repeat until convergence

Kutulakos & Seitz

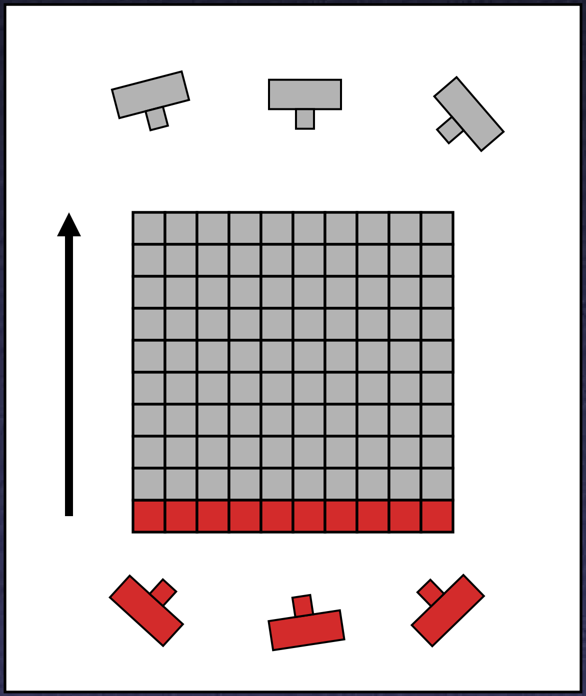
Multi-Pass Plane Sweep

- Faster alternative:
  - Sweep plane in each of 6 principal directions
  - Consider cameras on only one side of plane
  - Repeat until convergence

Multi-Pass Plane Sweep

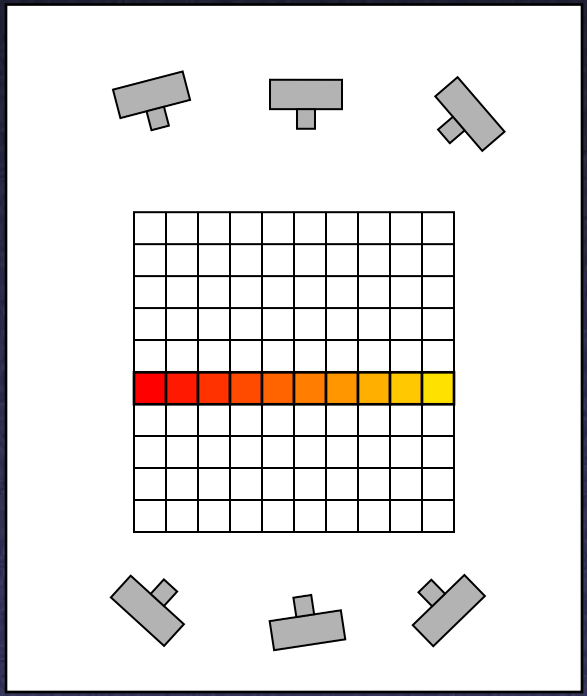


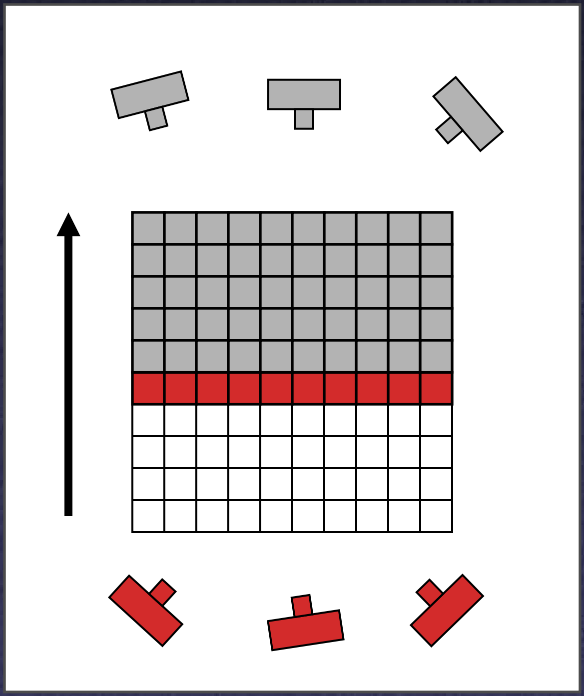
True Scene



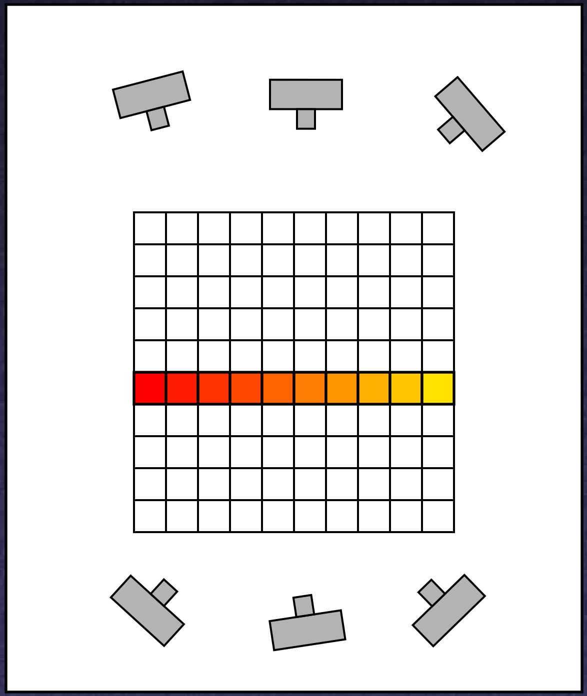
Reconstruction

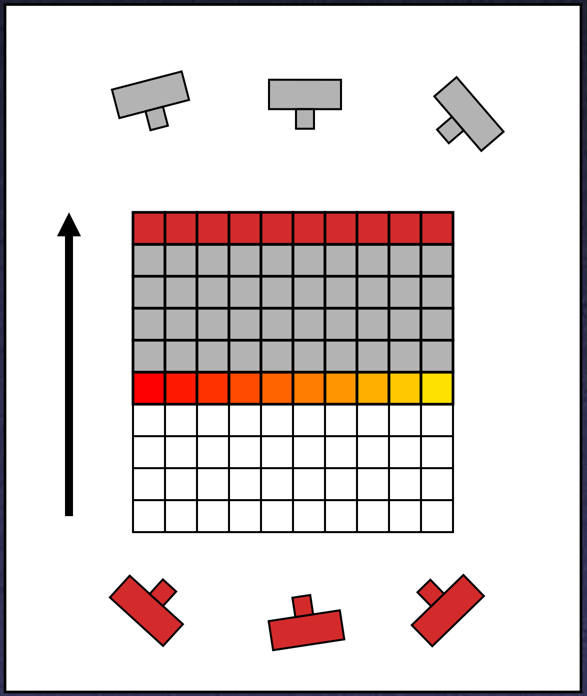
Multi-Pass Plane Sweep



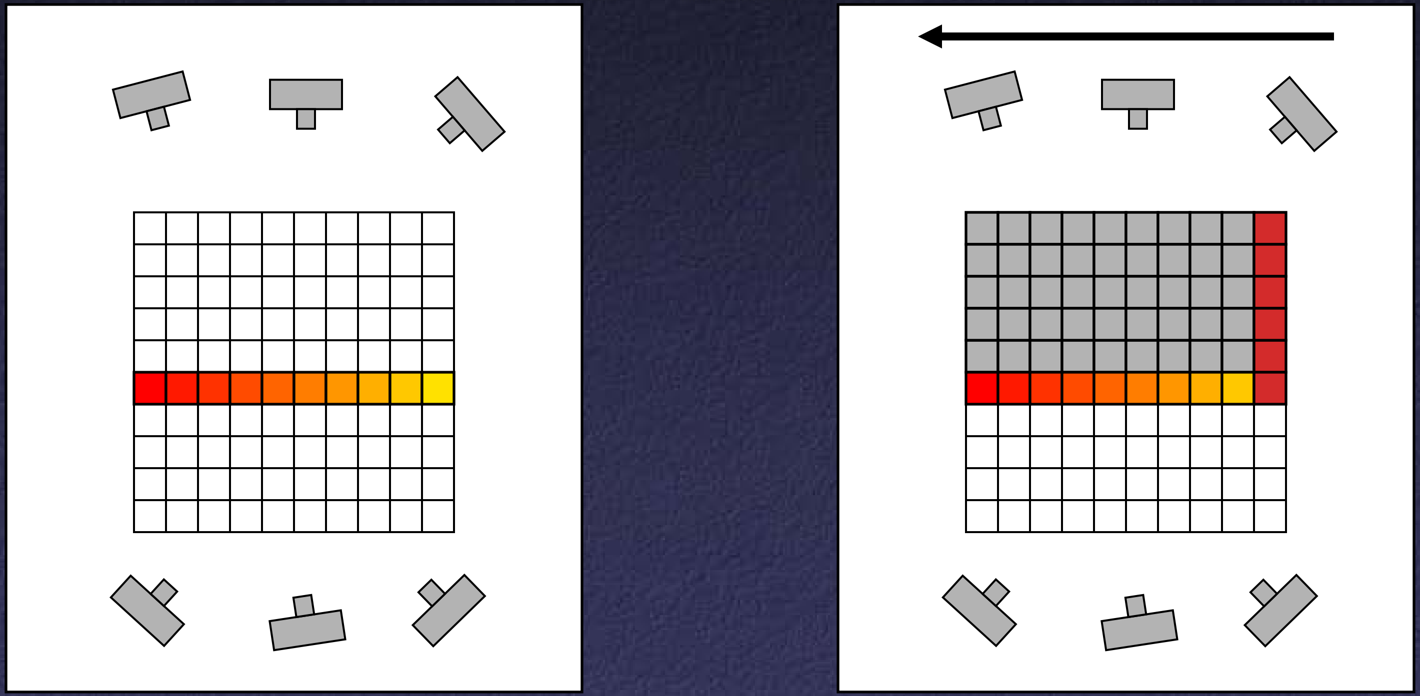


Multi-Pass Plane Sweep

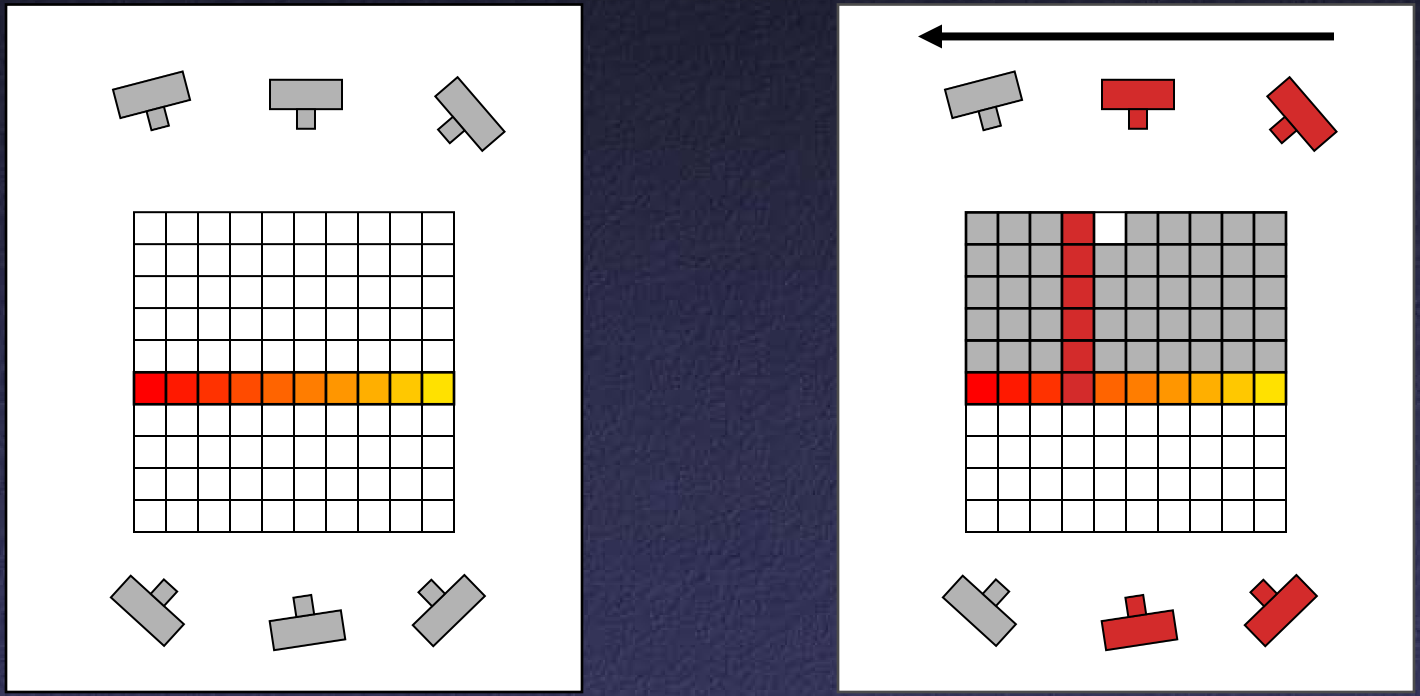




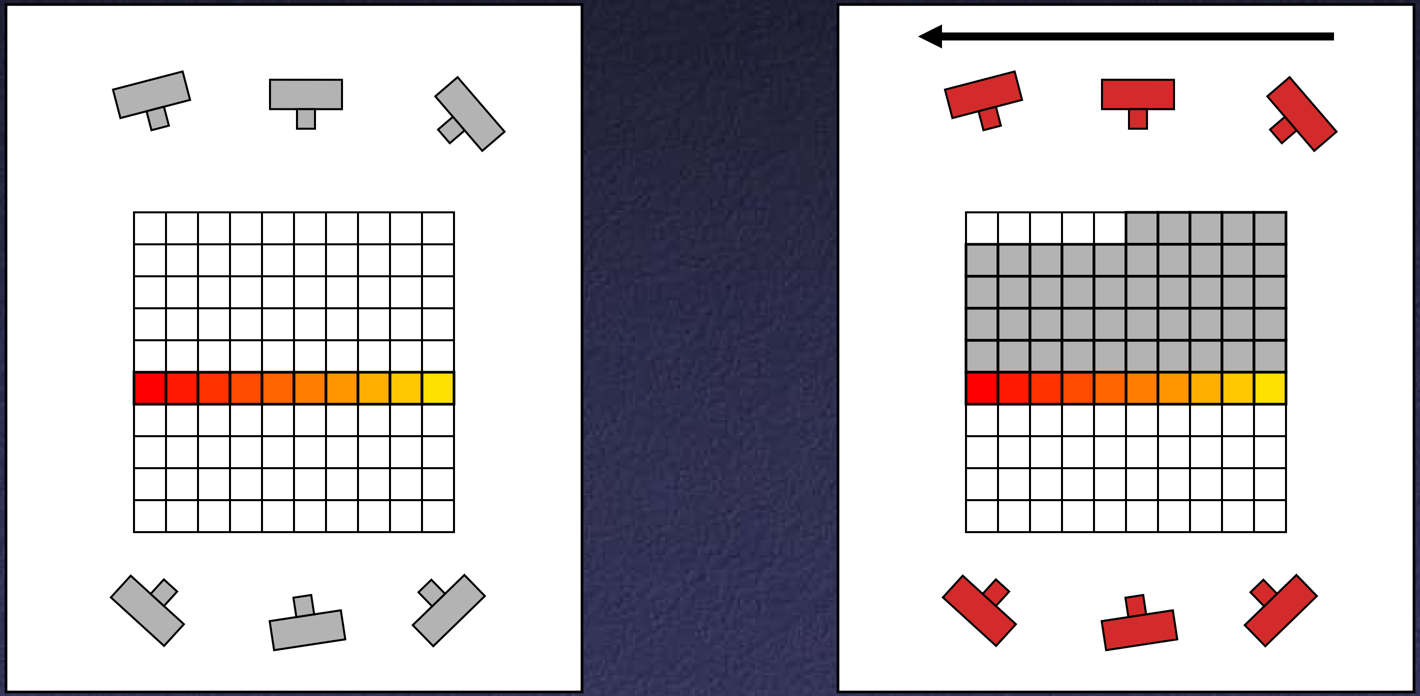
Multi-Pass Plane Sweep



Multi-Pass Plane Sweep



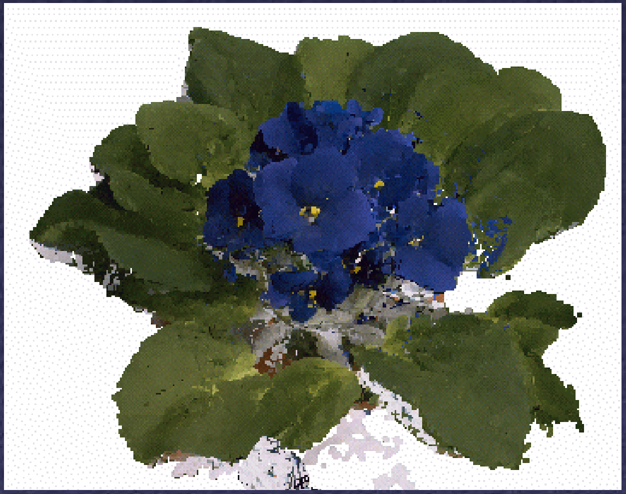
Multi-Pass Plane Sweep



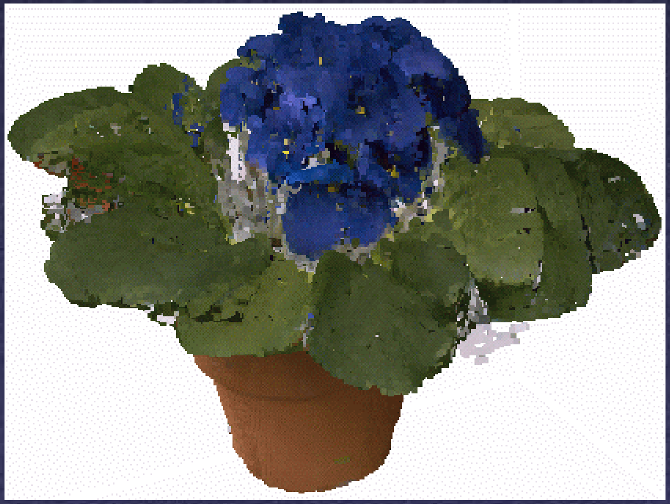
Space Carving Results: African Violet

Input Image (1 of 45)

Reconstruction



Reconstruction



Reconstruction

Space Carving Results: Hand

Input Image
(1 of 100)

Views of Reconstruction