

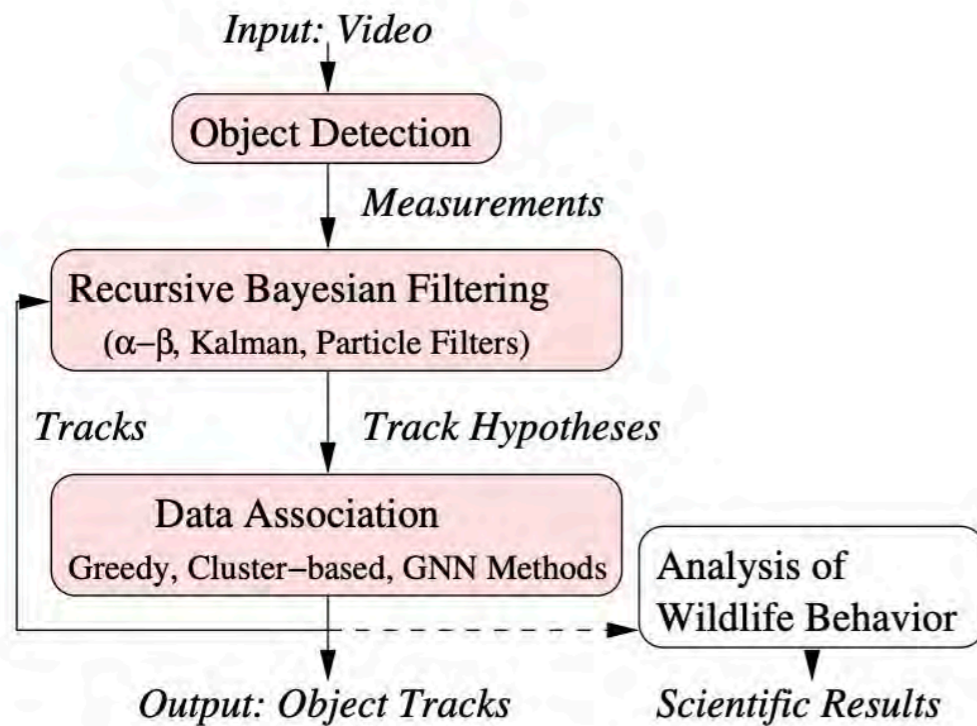
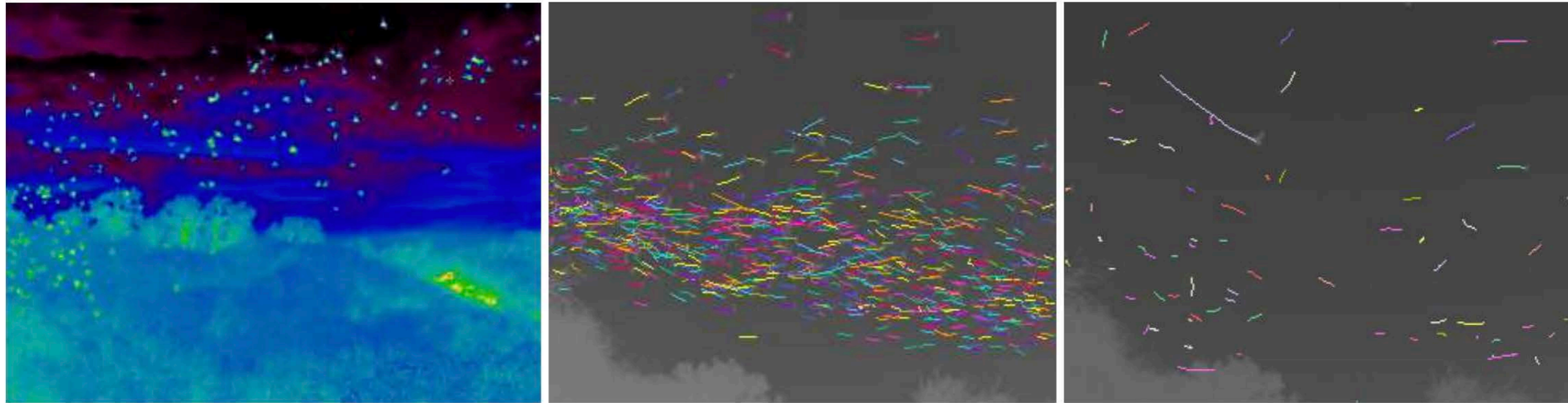
Dense Methods I: Stereo

CSE P576

Vitaly Ablavsky

These slides were developed by Dr. Matthew Brown for CSEP576 Spring 2020 and adapted (slightly) for Fall 2021
credit → Matt
blame → Vitaly

Computer Vision: The Halloween Edition



Results video: <https://youtu.be/3BiK4AgfIRo>

Dense Methods I: Stereo

- Stereo matching, local + global optimization
- Multi-view stereo, geometry representations
- Photometric Stereo

Application: Photo Collections → 3D

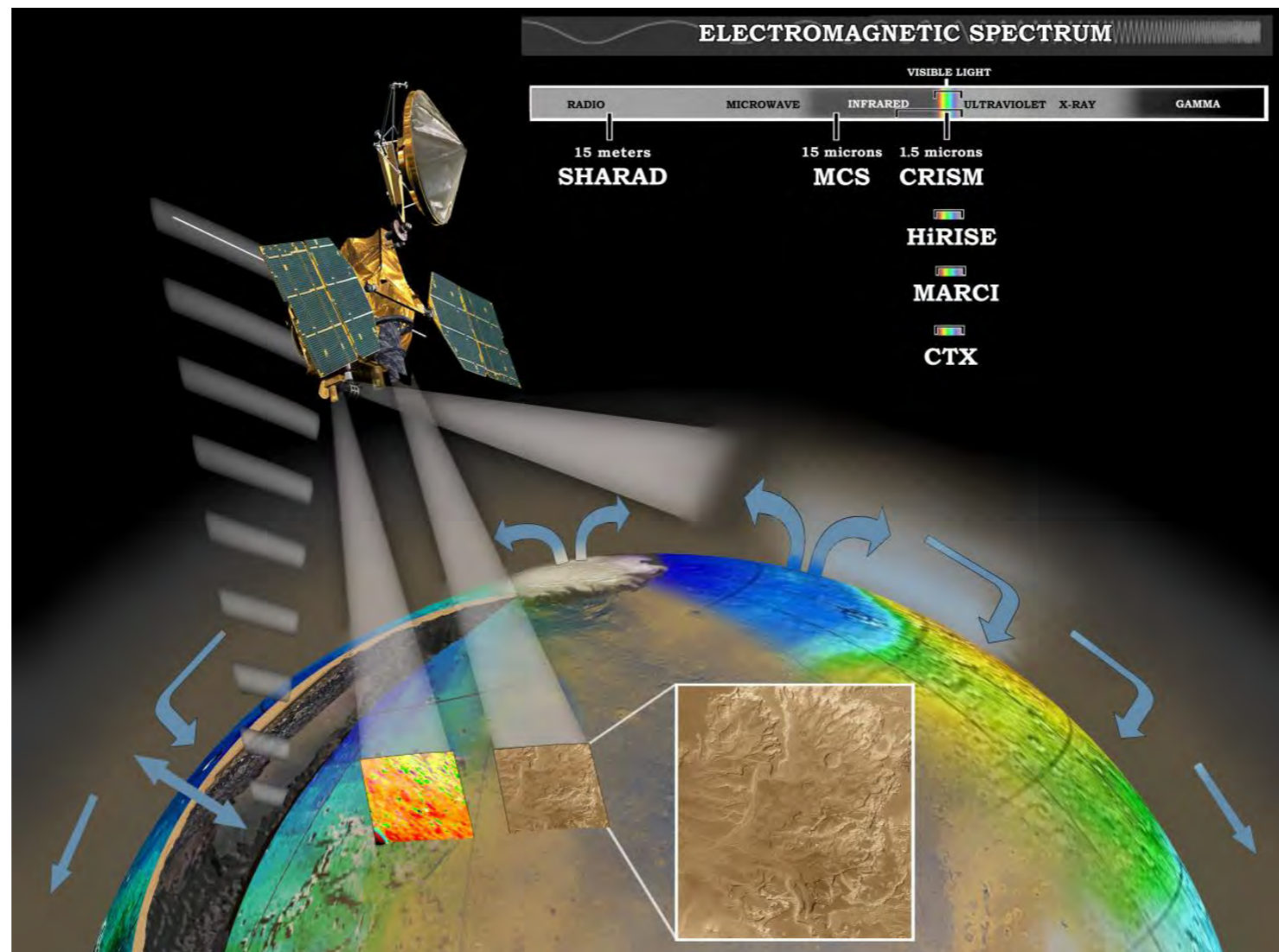


- Generate detailed 3D model (e.g., depth values at every pixel in input images)

[Y. Furukawa PMVS]



Application: Remote Sensing

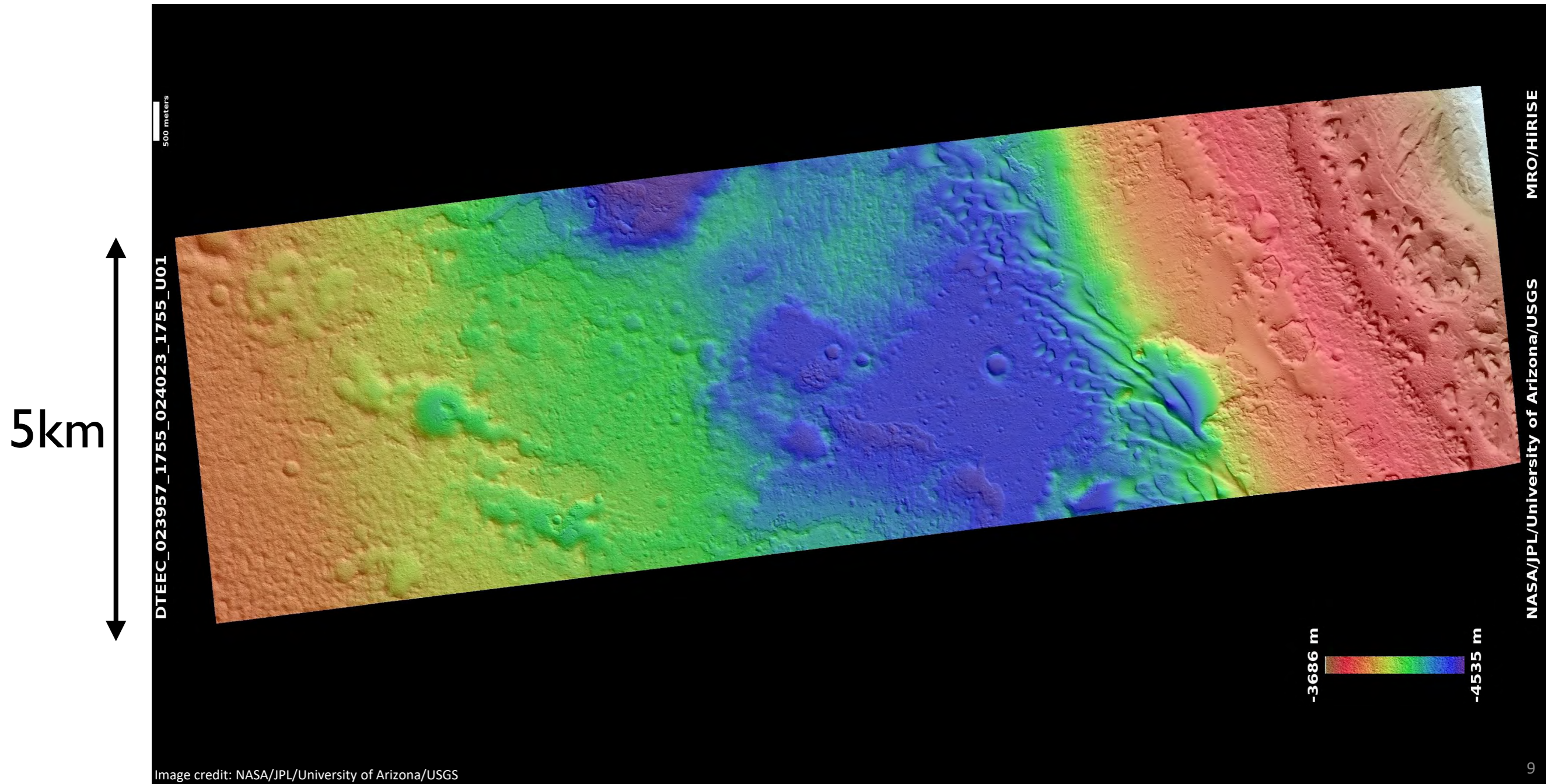


[NASA/JPL]

- Mars Reconnaissance Orbiter
- Launched 2005, ~13 orbits / earth day
- HiRISE camera pixels are 1 μ radian (0.3m at 300km)
- MARCI camera has 5 visible + 2 UV bands, lower res

Application: Remote Sensing

- Martian surface elevation map



Application: Remote Sensing

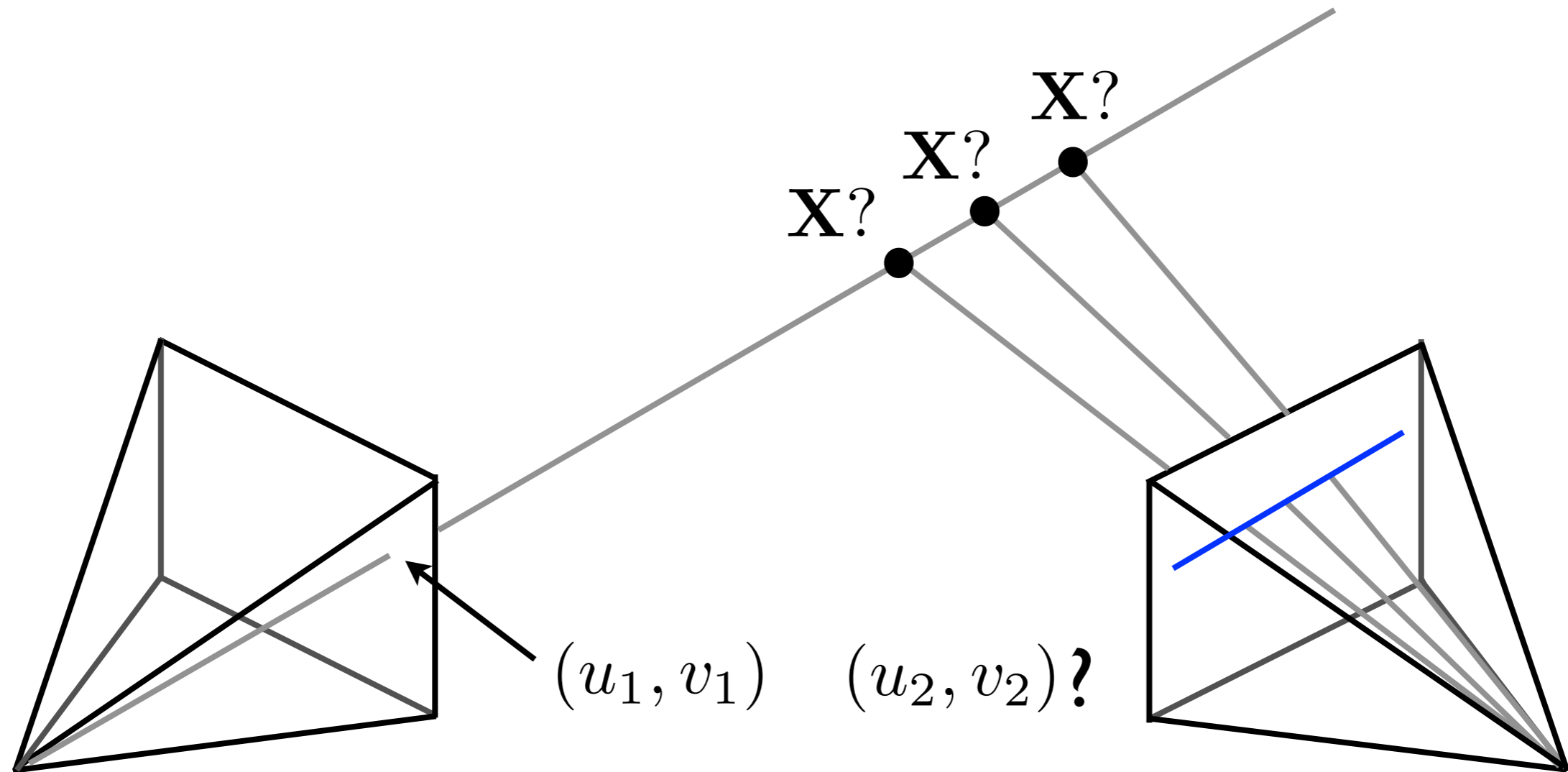
- Martian surface detail



Image credit: NASA / JPL-Caltech / UA / Kevin M. Gill

Epipolar Geometry

- A point in one view may lie on a line in the 2nd



Position in image 2 depends on the **depth** of the 3D point

2-view Stereo

- Camera motion only, points constrained to epipolar lines



ID Search

Stereo Camera Configuration

- Humans and many stereo cameras have parallel optical axes

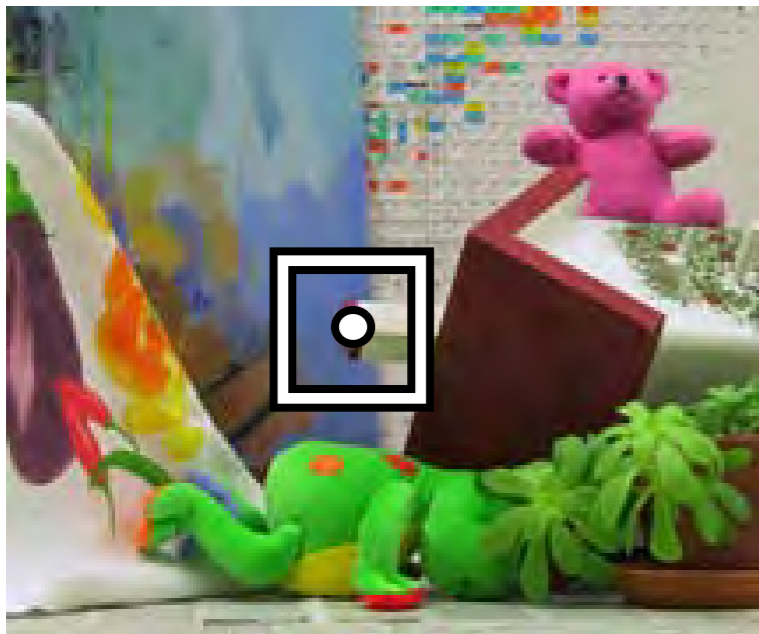


Axis Aligned Stereo

- A common stereo configuration has camera optical axes aligned, with cameras related by a translation in the x direction

Stereo Matching

- In a standard stereo setup, where cameras are related by translation in the x direction, epipolar lines are horizontal



- Stereo algorithms search along scanlines for matches
- Distance along the scanline (difference in x coordinate) for a corresponding feature is called **disparity**

Disparity and Depth: R



Disparity and Depth: L



Disparity and Depth: R+L



Effect of Window Size

- Larger windows \rightarrow smoothed result



$W=3$



$W=11$



$W=25$

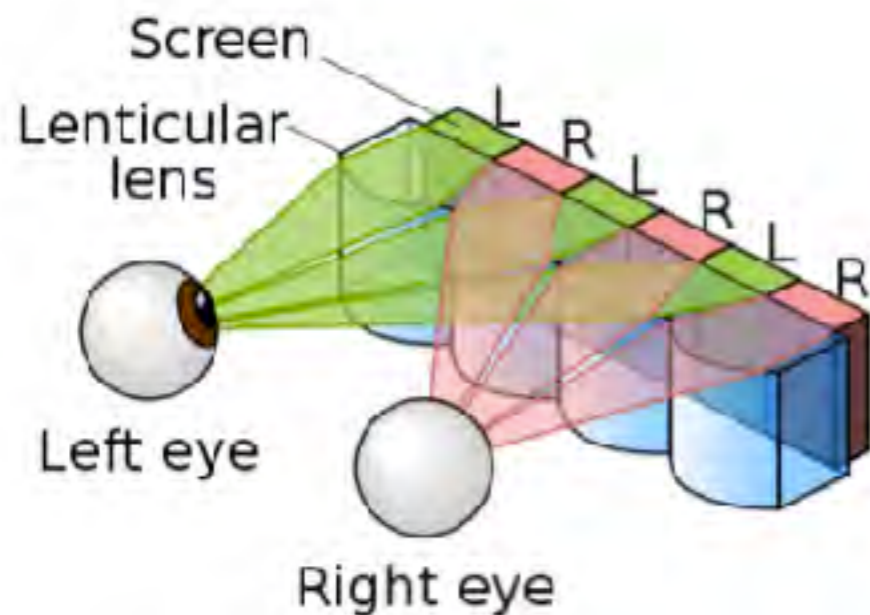
Anaglyph

- Stereo pair with images encoded in different color channels



Stereo Displays

- Field sequential (shutter) glasses transmit alternate left/right image at 120Hz



Lenticular lenses send different images directly to each eye, without the need for glasses

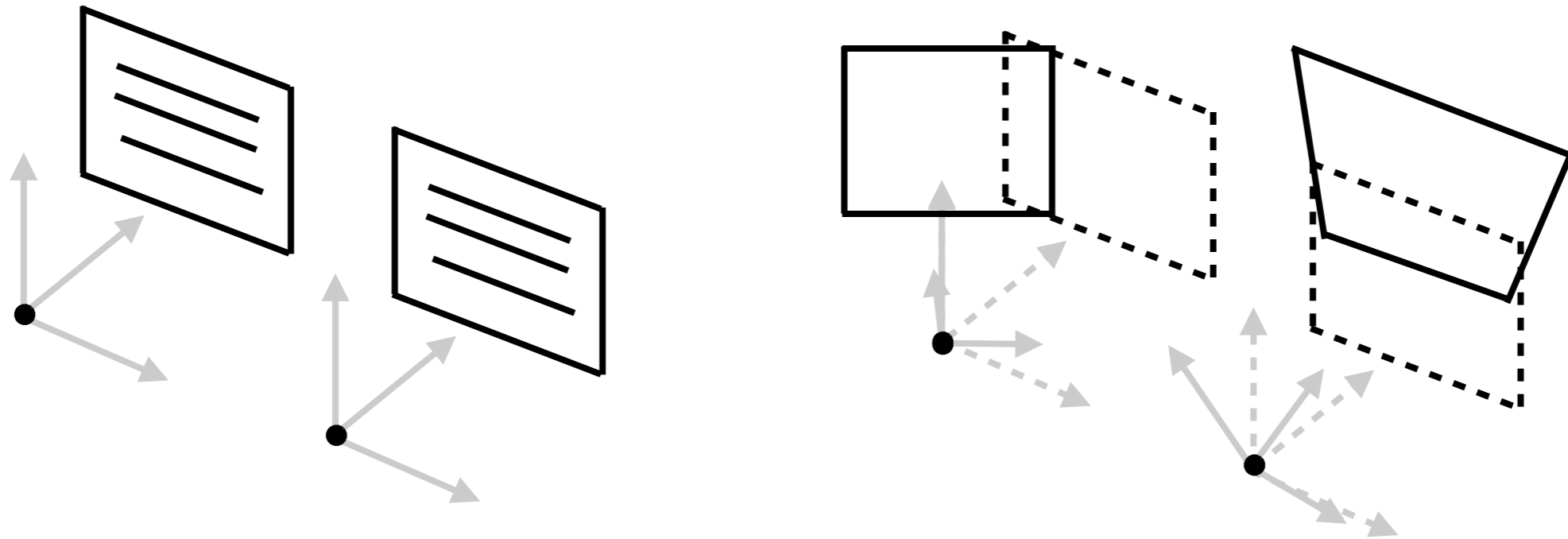
Stereo Displays

- VR headsets send L/R images directly to each eye



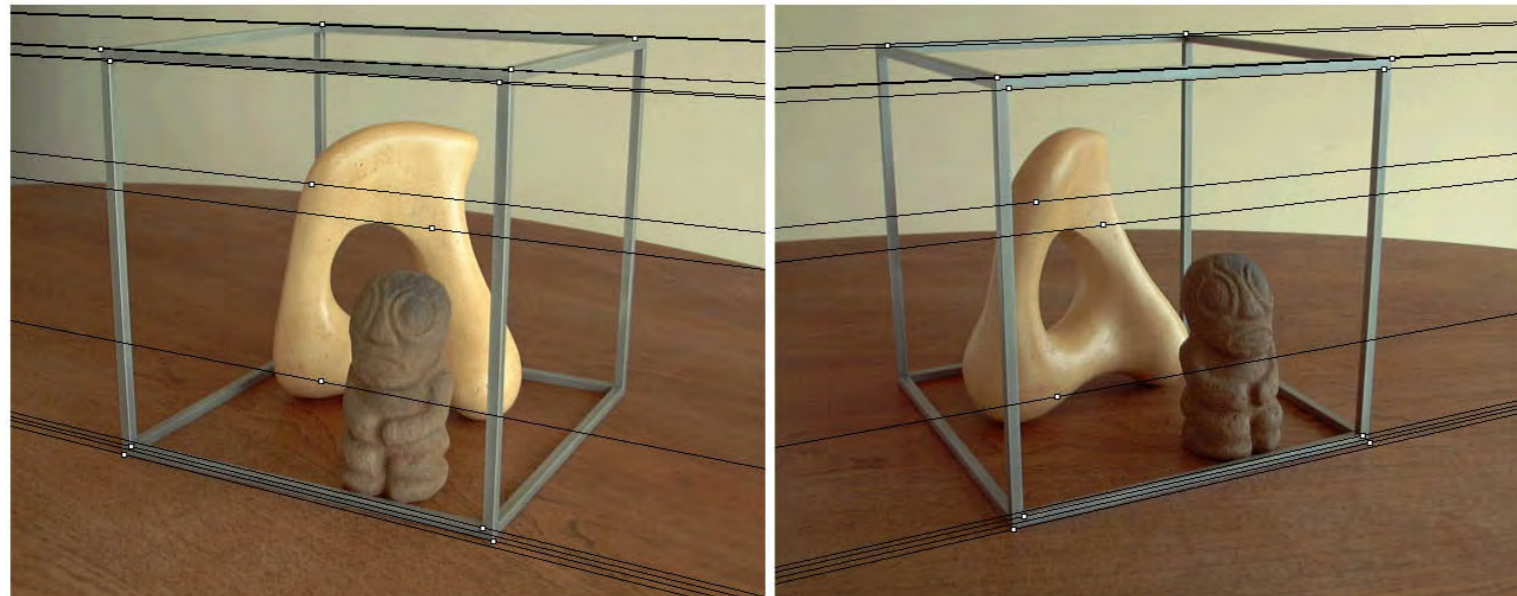
Stereo Rectification

- If the optical axes are not aligned, we can rotate the images (homography) until they are perpendicular to the baseline



Stereo Rectification

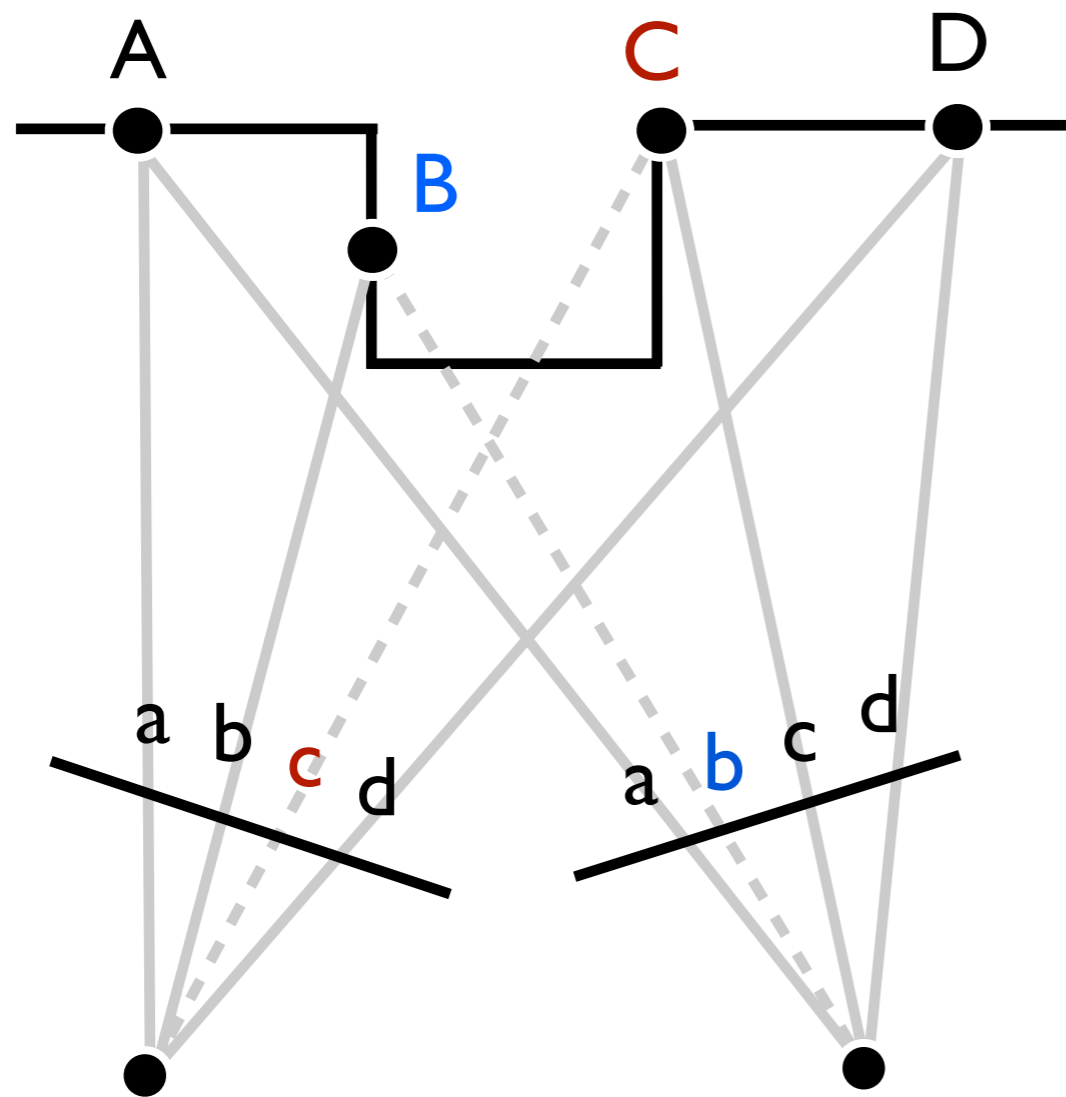
- Transform (rotate) images so that epipolar lines are horizontal



[Loop Zhang 1999]

Occlusions

- Sometimes a point in image 1 does not appear in image 2, or vice-versa (this is called an **occlusion**)



- Occlusions cause gaps in the stereo reconstruction
- + Matching is difficult nearby as aggregation windows often overlap the occluded region

Edge Aware Stereo

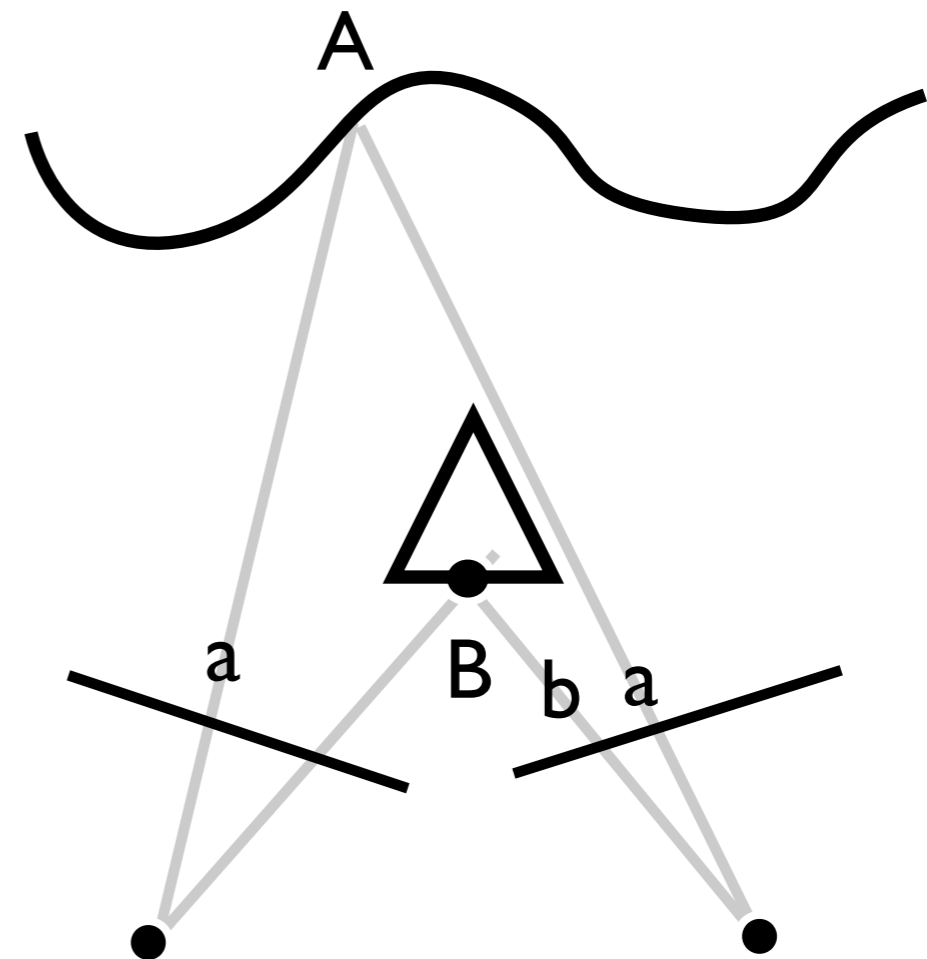
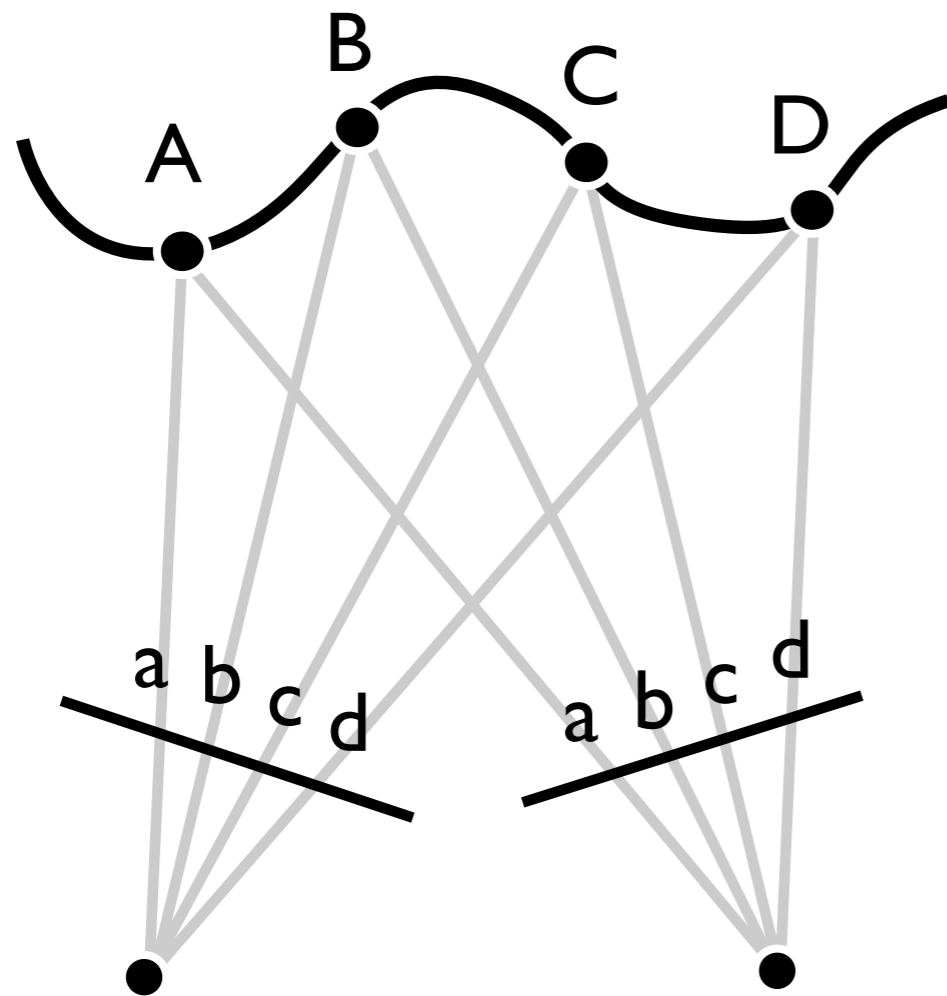
- Occlusions and depth discontinuities cause problems for stereo matching, as aggregation windows overlap multiple depths



- Segmentation-based stereo approaches aim to solve this by trying to guess the depth edges (e.g., joint segmentation and depth estimation [Taguchi et al 2008])

Ordering Constraint

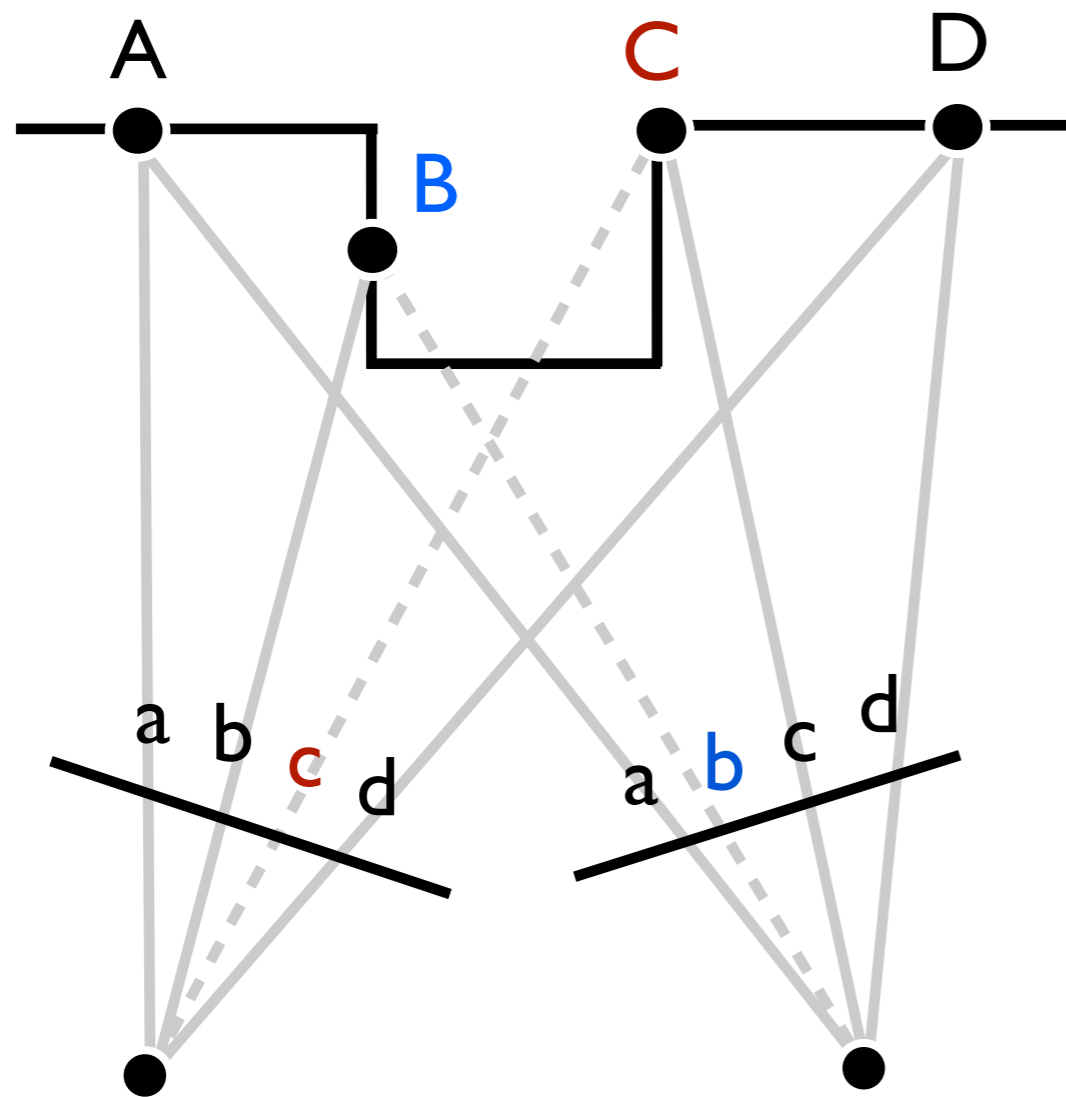
- If point B is to the right of point A in image 1, the same is *usually* true in image 2



Not always, e.g., if an object is wholly within the ray triangle generated by A

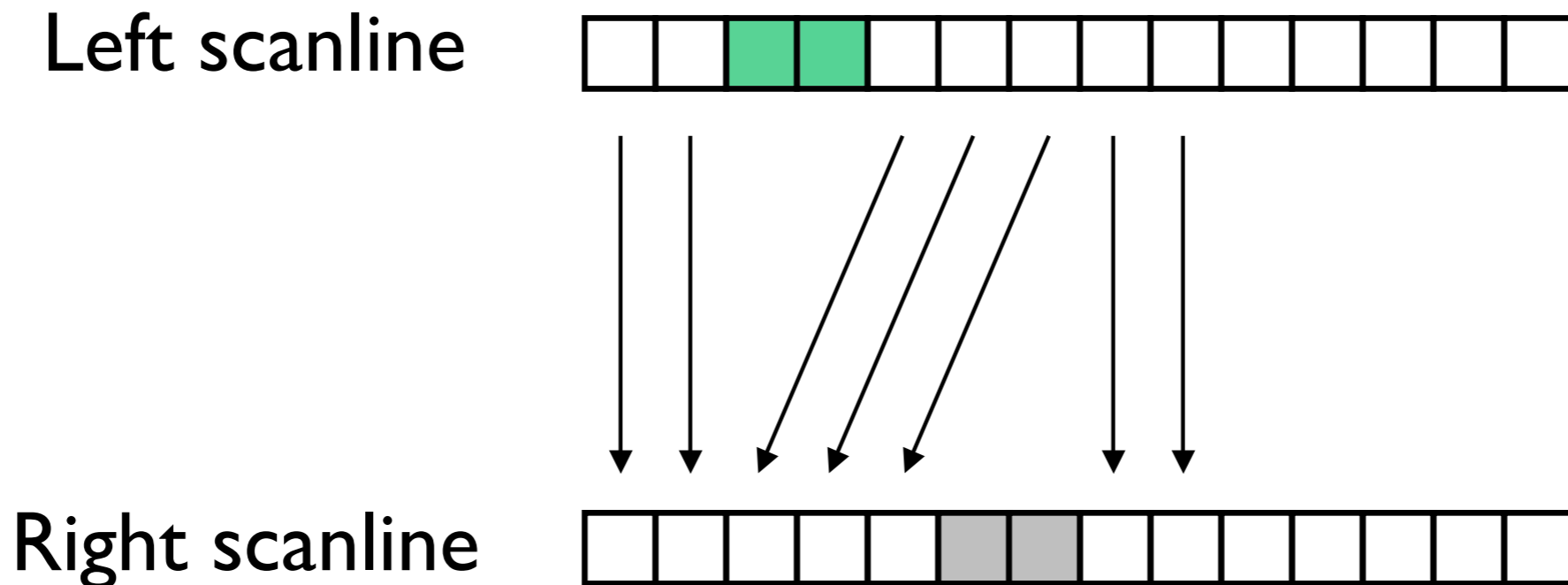
Occlusions + Ordering

- Note that the ordering constraint is still maintained in the presence of occlusions



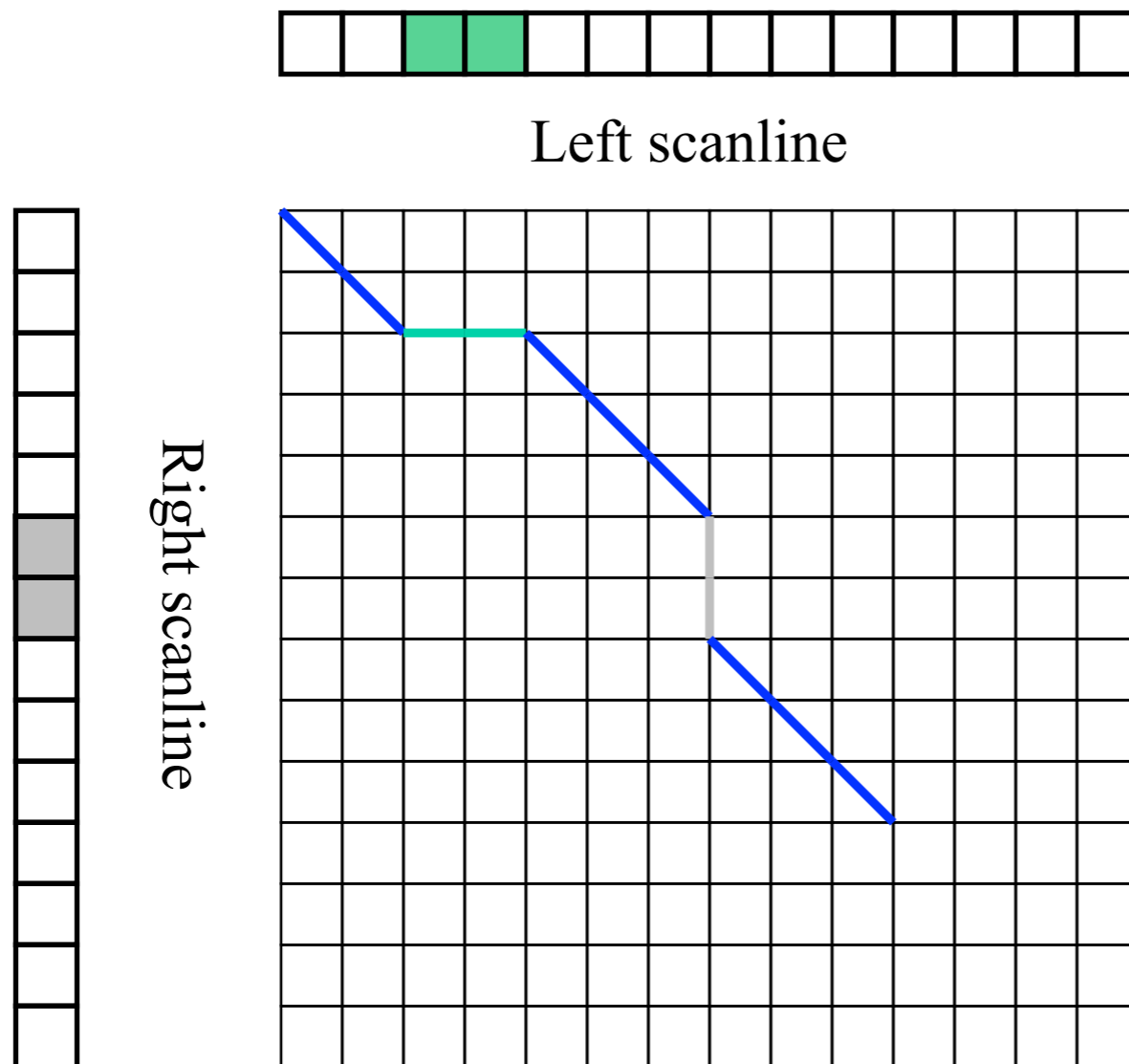
Optimal Scanline Mapping

- We can imagine a mapping between left and right scanlines
- If we assume the ordering constraint this must be monotonic, but there may be step changes (due to occlusions)
- How can we find the best monotonic sequence mapping left to right scanlines?



Dynamic Programming

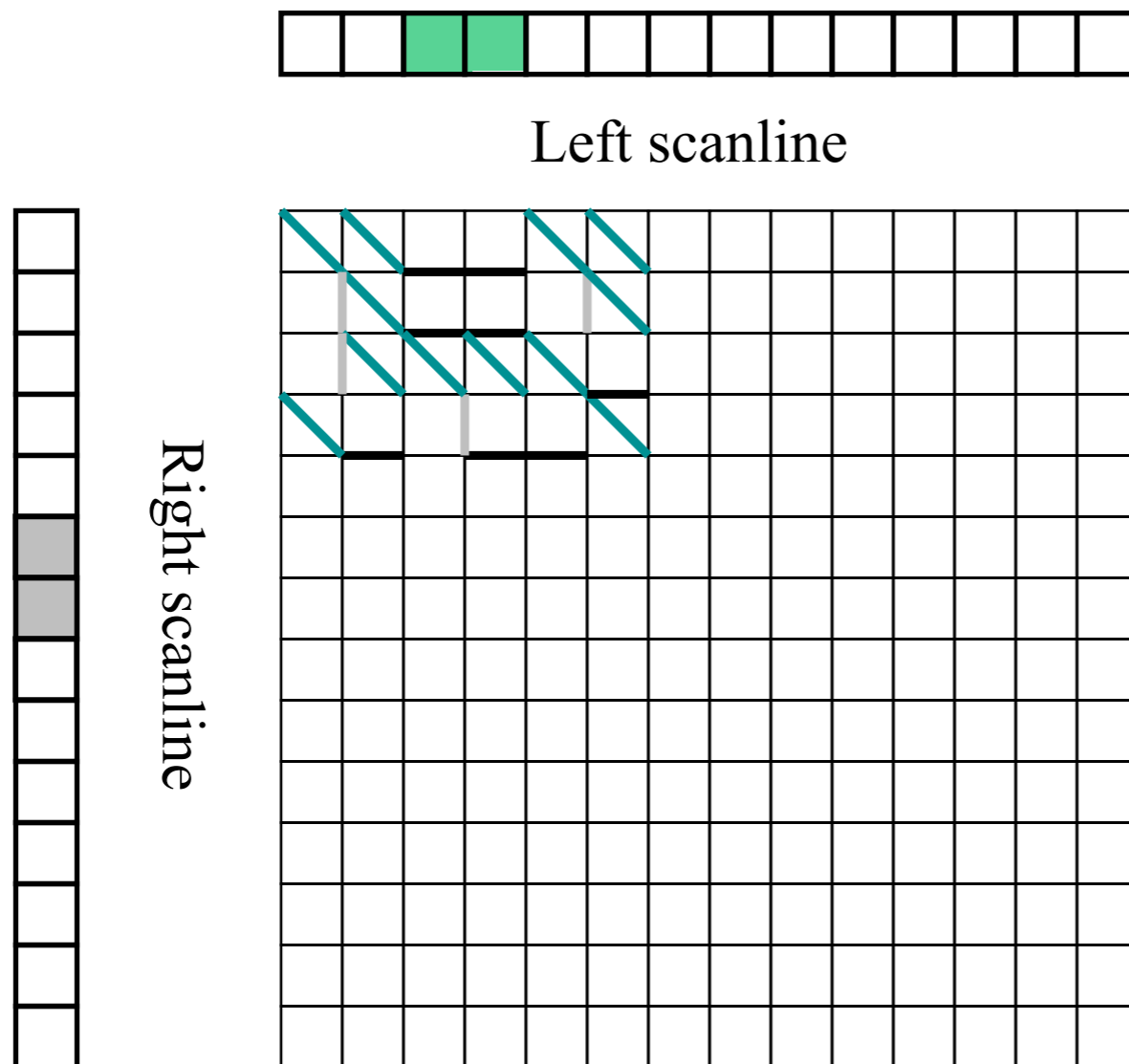
- At each point, we may make one of 3 moves: left/right occlusion (higher cost), or sequential correspondence (lower cost based on patch SSD)



Look for a path from top left to bottom right with the lowest cost

Dynamic Programming

- We need not consider all possible paths, as we only need to know the lowest cost path for reaching each state



Scan over grid finding
min-cost paths and
backtrack from the end

Stereo Cost Functions

- Energy function for stereo matching based on disparity $d(x,y)$
- Sum of data and smoothness terms

$$E(d) = E_d(d) + \lambda E_s(d)$$

- Data term is cost of pixel x,y allocated disparity d (e.g., SSD)

$$E_d(d) = \sum_{(x,y)} C(x, y, d(x, y))$$

- Smoothness cost penalises disparity changes with robust $\rho(\cdot)$

$$E_s(d) = \sum_{(x,y)} \rho(d(x, y) - d(x + 1, y)) + \rho(d(x, y) - d(x, y + 1))$$

- This is a Markov Random Field (MRF), which can be solved using techniques such as Graph Cuts

Stereo Comparison

- Global vs Scanline vs Local optimization



Ground
truth



Graph Cuts
[Kolmogorov
Zabih 2001]



Dynamic
Programming



SSD 21px
aggregation

Multiview Stereo

- Plane sweep, volumetric, depth map merging

Multiview Stereo



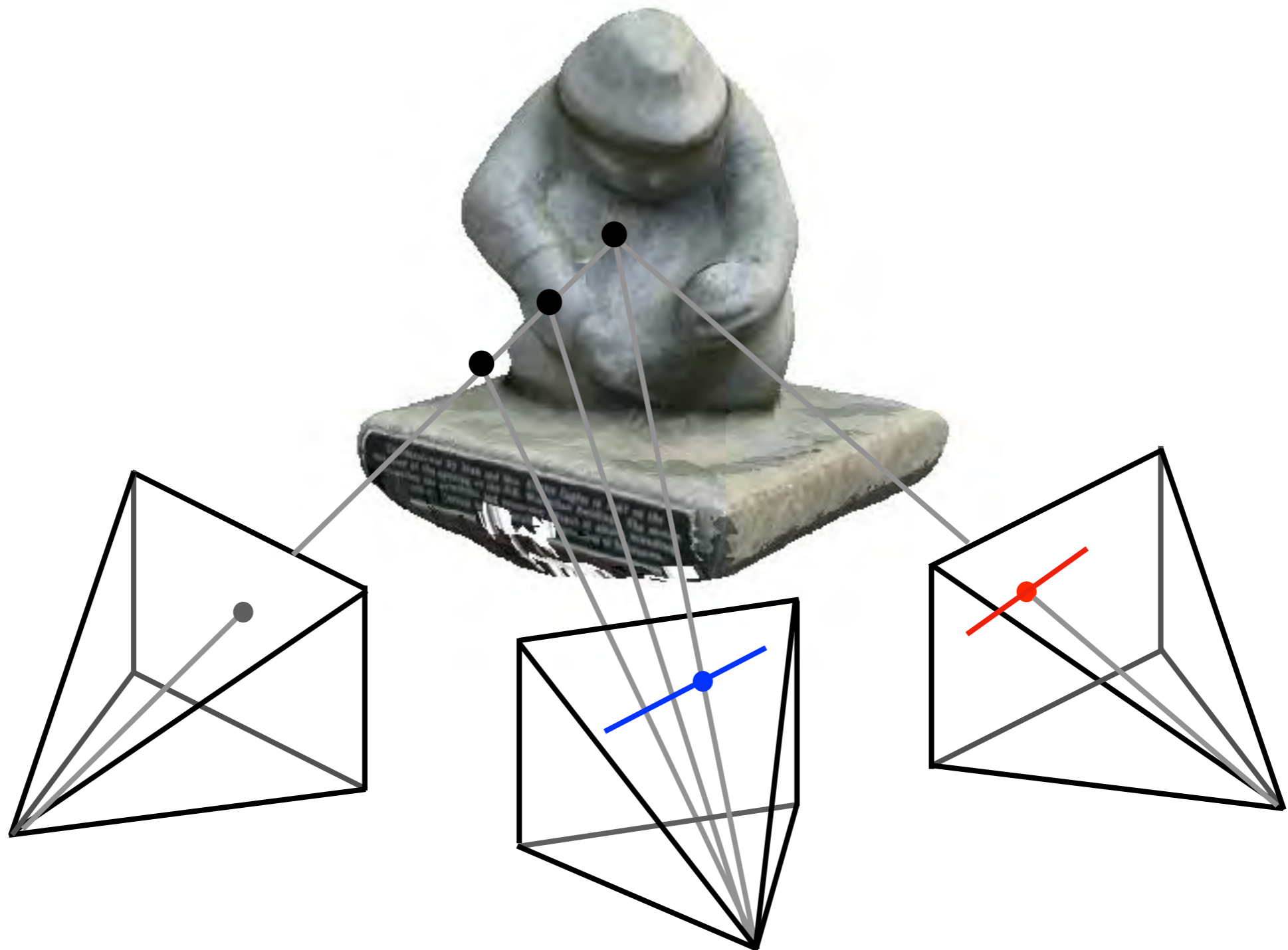
- Use information from $N > 2$ views to form a dense 3D reconstruction

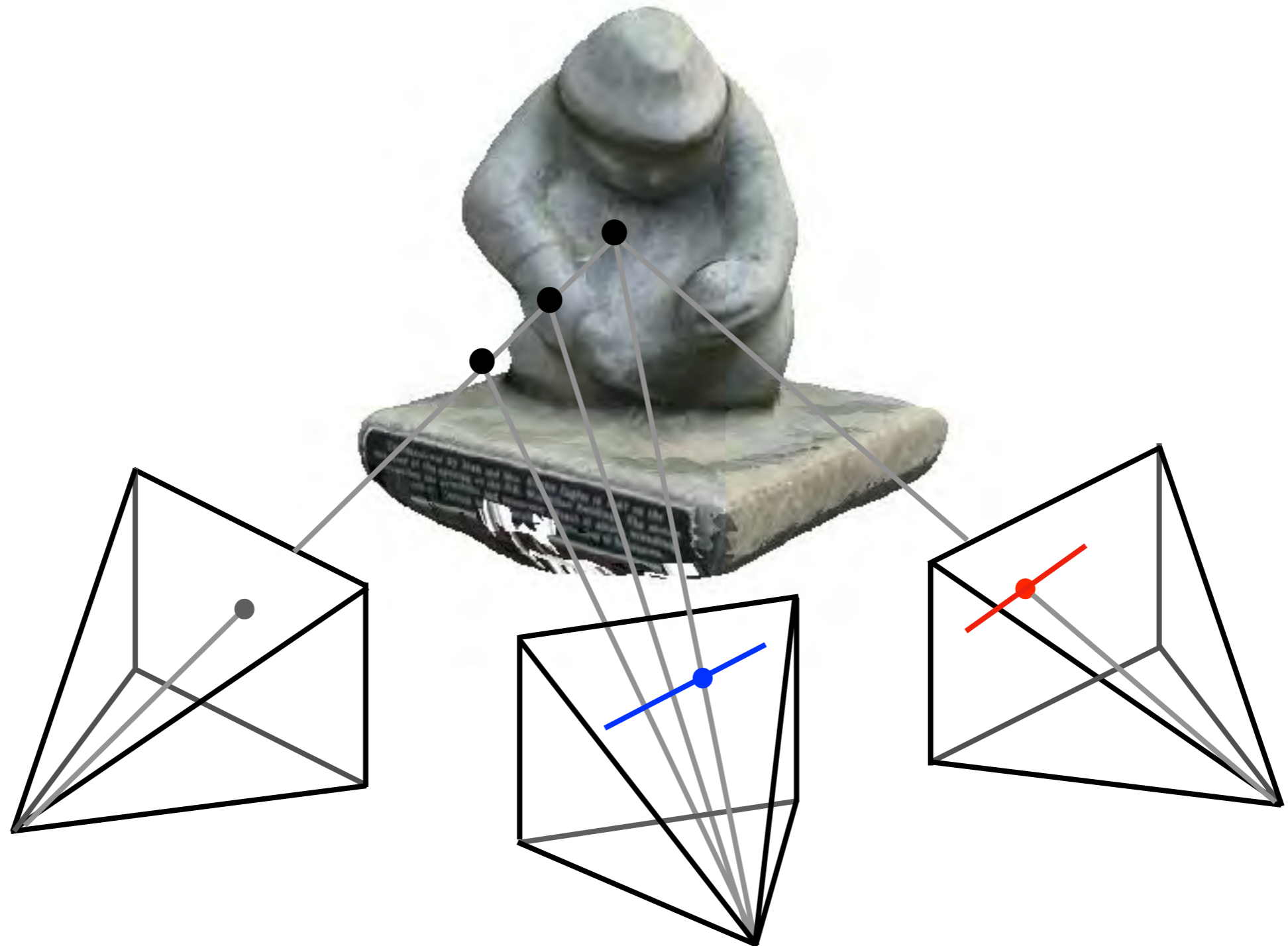
[Y. Furukawa PMVS]



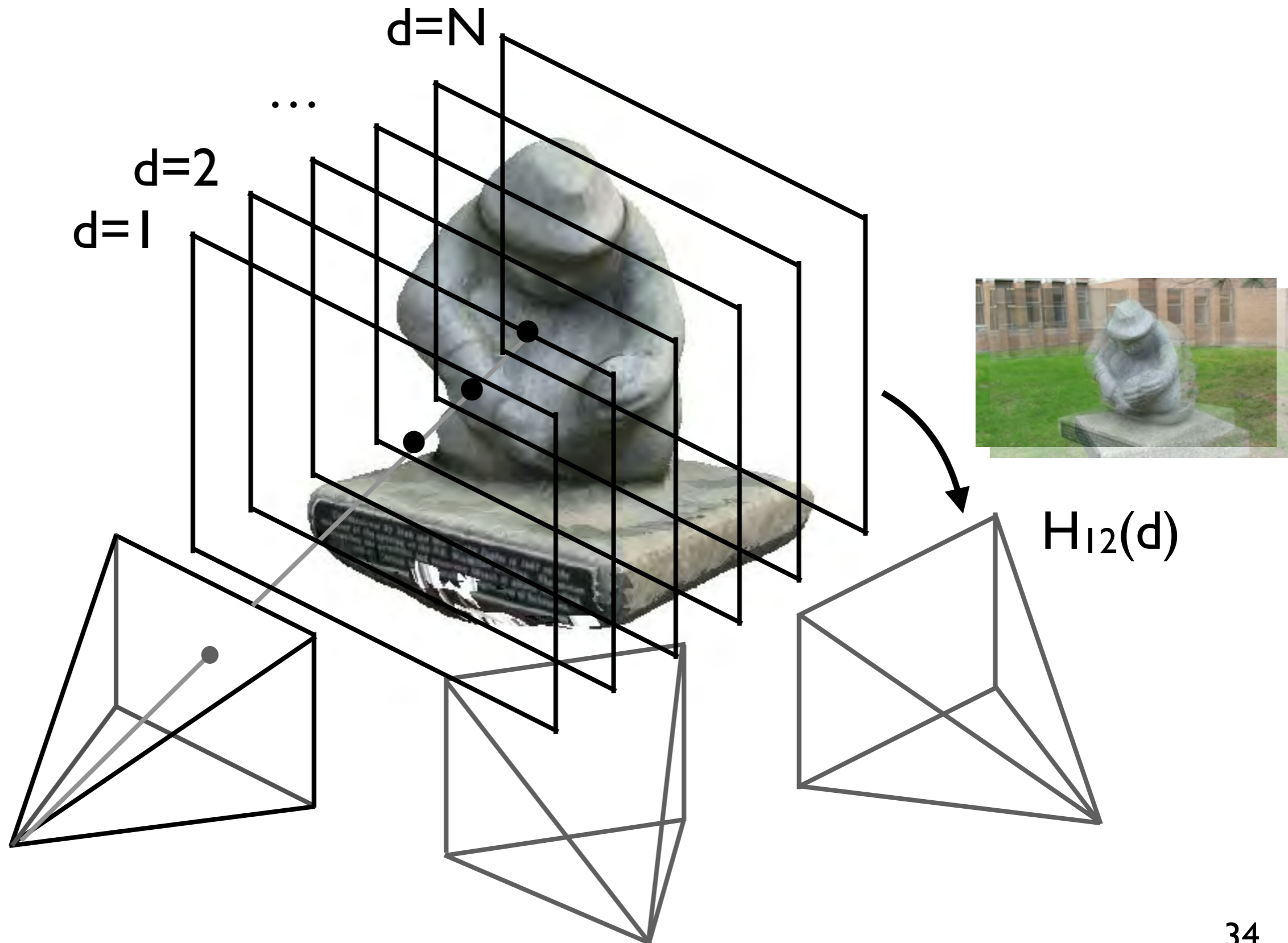
Multiview Stereo

- Search along epipolar lines to find good matches in N views



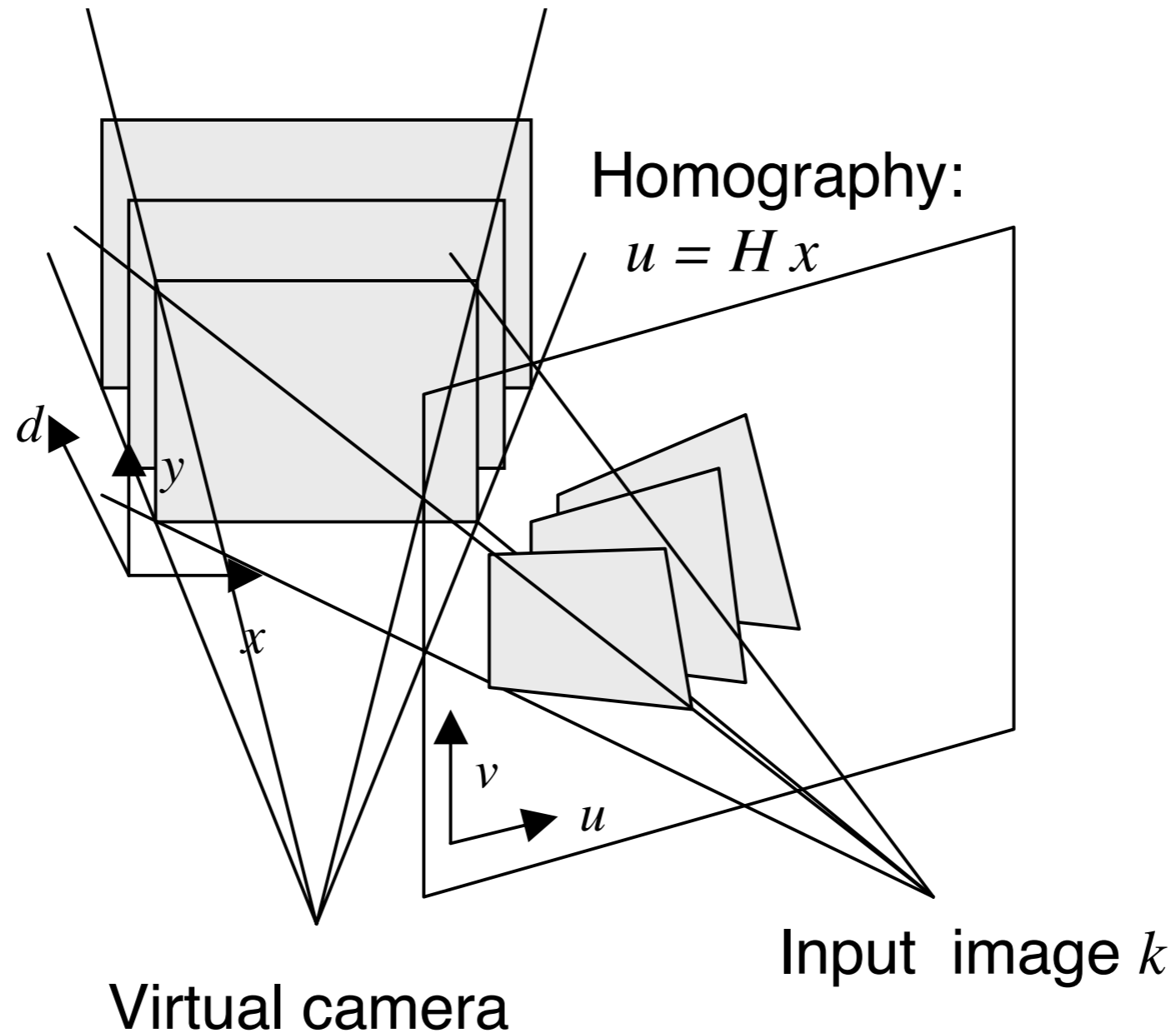


Plane Sweep Stereo




Plane Sweep Stereo

- Warp images using a set of planes in front of the camera



Plane Sweep Stereo

- Warp images using a set of planes in front of the camera
-  Try out `PlaneSweep.ipynb` from the course webpage

Plane Sweep Stereo Jupyter Notebook

```
[63]: show_images(images, titles=['left', 'right'], cols=2)
```

left



right



Plane Sweep

```
[8]: disp_search = np.linspace(1.0/16.0, 1.0/2.0, 128)
diffs = np.zeros((I_l.shape[0], I_l.shape[1], len(dispatch)))

# Sweep over image in inverse depth (/disparity)
for d in range(0, len(dispatch)):
    # Depth of plane
    z = 1.0 / disp_search[d]
    # We'll use fronto parallel planes with the following normal:
    n = np.array([0, 0, 1])
    # Since these images are rectified, H_rl ends up as a trivial translation!
    H_rl = np.matmul(np.matmul(Kr, R_rl - np.outer(l_r, n/z)), np.linalg.inv(Kl))
    # Synthesize an image of I_L from I_r induced by plane n/z
    Iwarp_l = transform.warp(I_r, H_rl)
    # Compare our synthetic image to our observed left image
    diff3 = I_l - Iwarp_l
    diffs[:, :, d] = np.sqrt(diff3[:, :, 0]**2 + diff3[:, :, 1]**2 + diff3[:, :, 2]**2)
```

```
[9]: print(z), print(n/z); print(H_rl)
```

```
2.0
[0.  0.  0.5]
[[ 1.         0.        -74.28570619]
 [ 0.         1.         0.         ]
 [ 0.         0.         1.         ]]
```

Application of Plane+Parallax: Camera Arrays

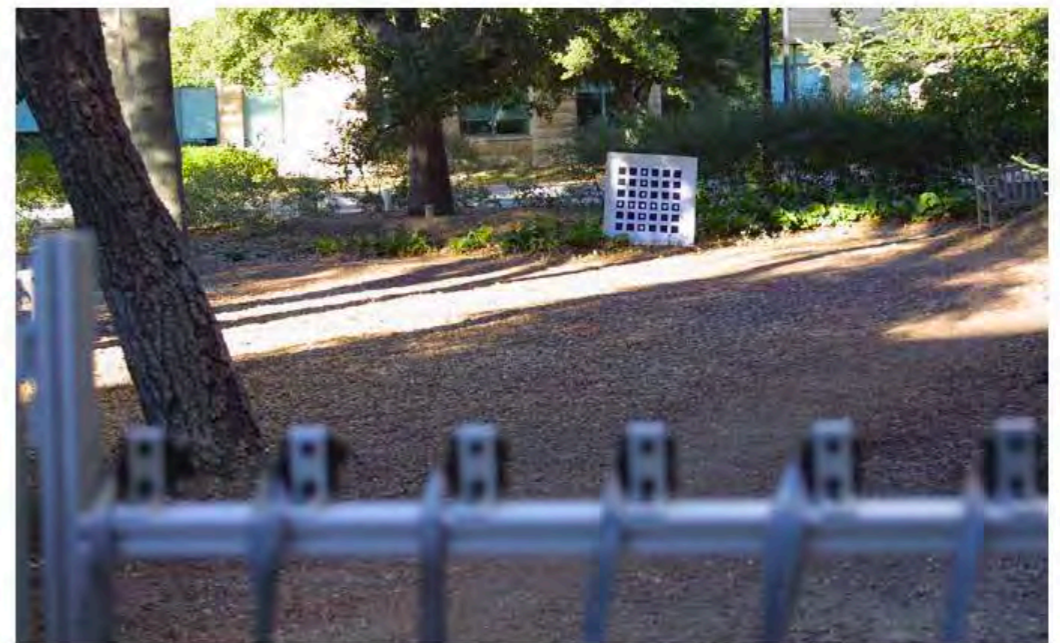
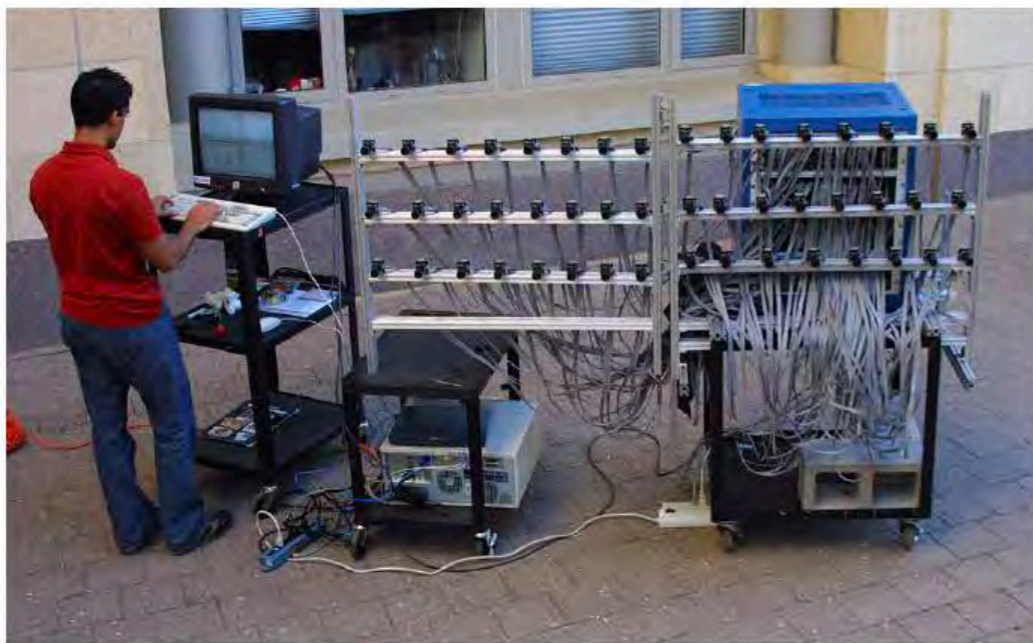
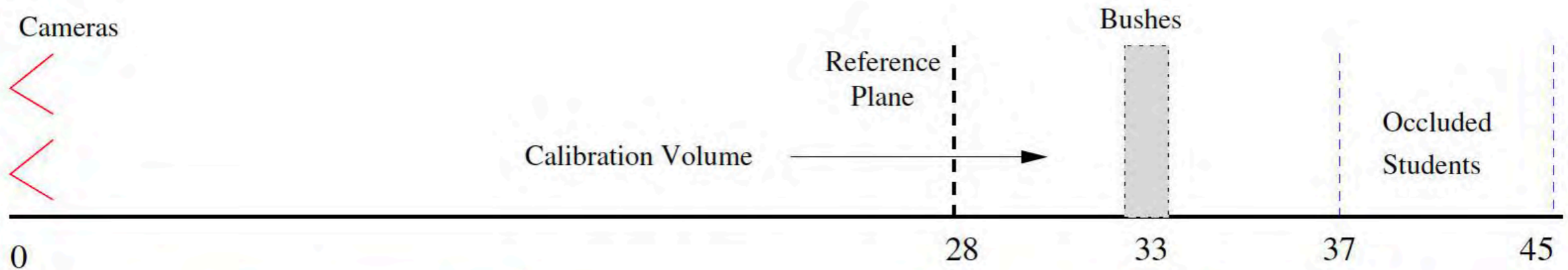


Figure 3: Experimental setup for synthetic aperture photography. Top: scene layout and distances from camera array (meters). Left: Our 45-camera array on a mobile cart, controlled by a standalone 933 MHz Linux PC. Right: The view from behind the array, showing the calibration target in front of bushes. The goal was to try and see people standing behind the bushes using synthetic aperture photography.

Calibration via Rank-1 Constraint on Parallax

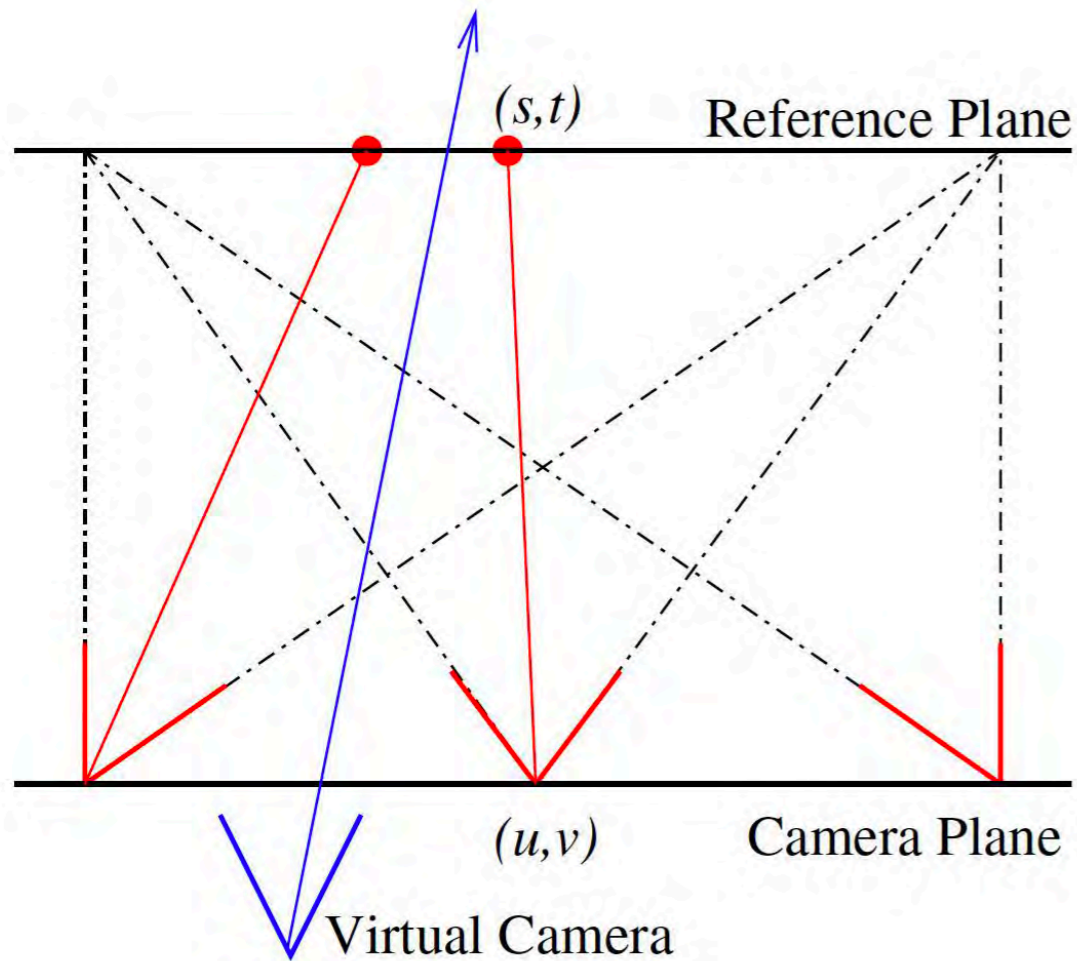


Figure 1: Light field rendering using the two-plane parametrization. The image viewpoints lie on the camera plane, and images are aligned on a parallel reference plane. Rays are parametrized by their intersection (u, v) with the camera plane and (s, t) with the reference plane. Rays desired for a novel view are computed by applying a 4-D reconstruction filter to the nearest samples in (u, v, s, t) space.

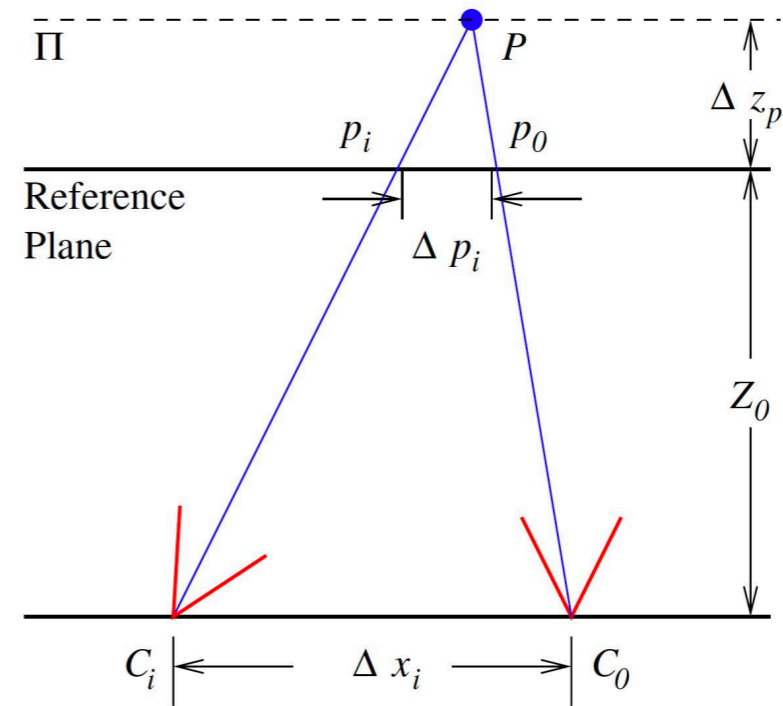


Figure 2: Planar parallax for light fields. A point P not on the reference plane has distinct images p_0, p_i in cameras C_0, C_i . The parallax between these two depends on the relative camera displacement Δx_i and relative depth $\frac{\Delta z_p}{\Delta z_p + Z_0}$.

$$\Delta p_i = \Delta x_i \frac{\Delta z_p}{\Delta z_p + Z_0} \quad (1)$$

$$\begin{bmatrix} \Delta p_{11} & \dots & \Delta p_{1n} \\ \vdots & \ddots & \vdots \\ \Delta p_{m1} & \dots & \Delta p_{mn} \end{bmatrix} = \begin{bmatrix} \Delta x_1 \\ \vdots \\ \Delta x_m \end{bmatrix} \begin{bmatrix} d_1 & \dots & d_n \end{bmatrix}$$

$$\Delta \mathbf{P} = \Delta \mathbf{X} \mathbf{D} \quad (2)$$

Seeing Through "Occlusions"



(a) Two images from a light field of students standing behind bushes.



(b) Synthetic aperture sequence, with the focal plane moving away from the array computed using our parallax based calibration method. We get sharp focus at different depths, ranging from the bushes to the building facade.

Seeing Through "Occlusions"



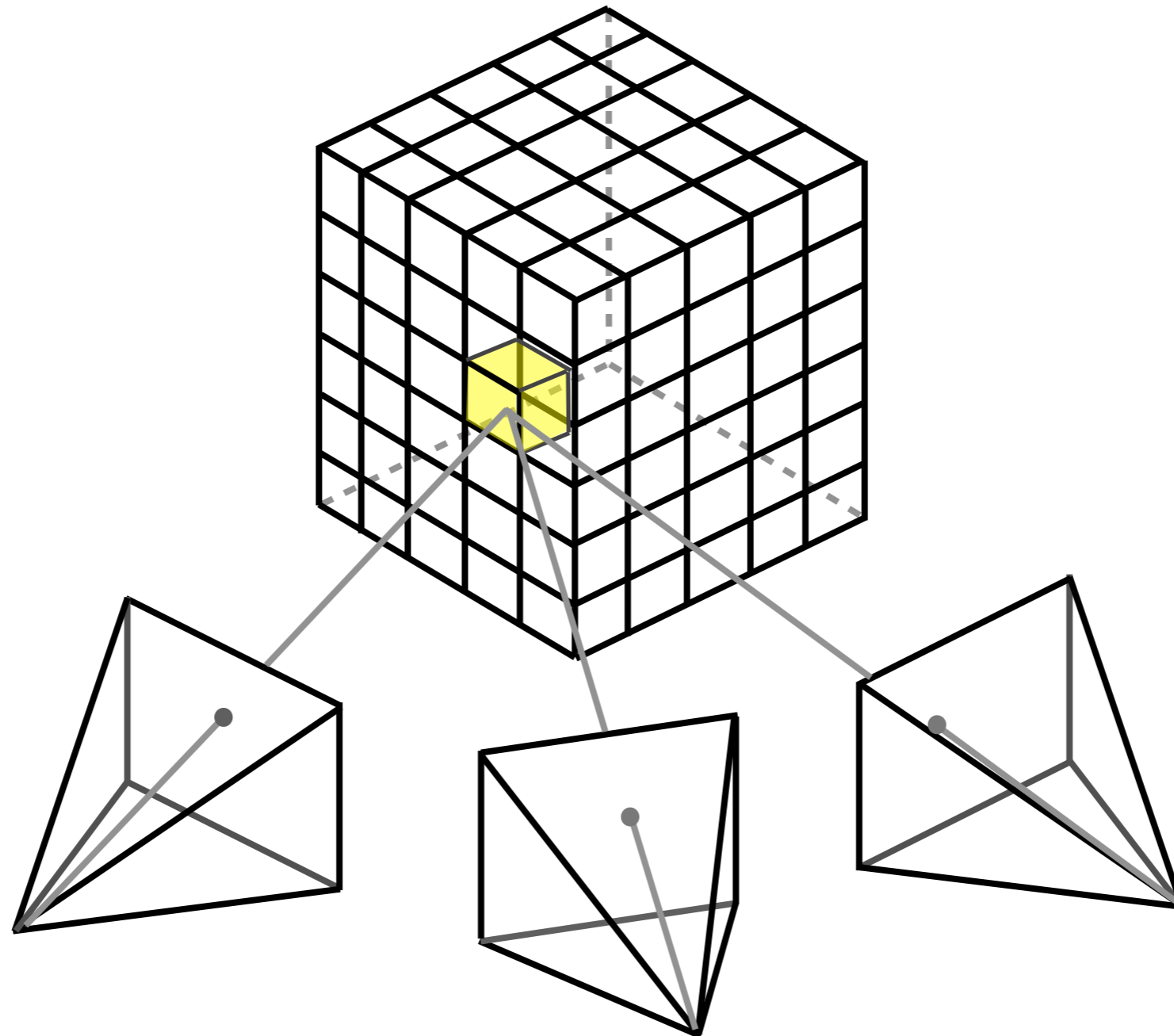
(a) Two images from a light field of students standing behind bushes.

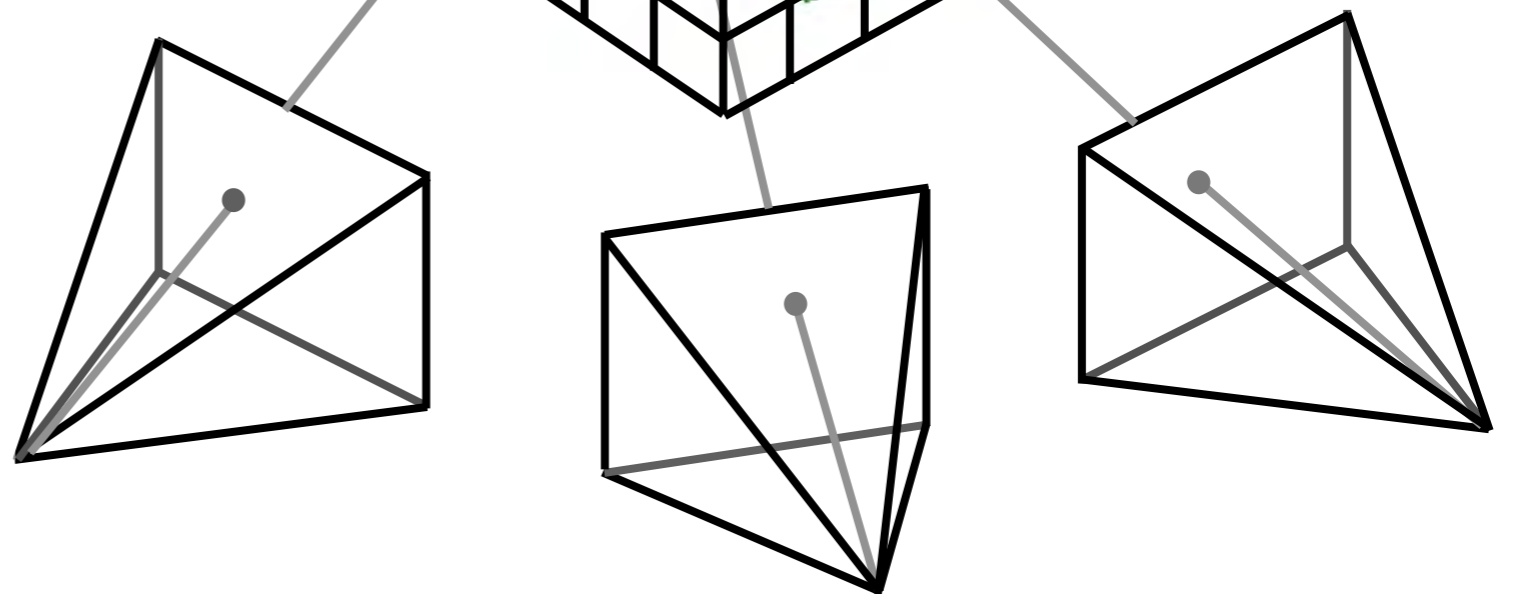
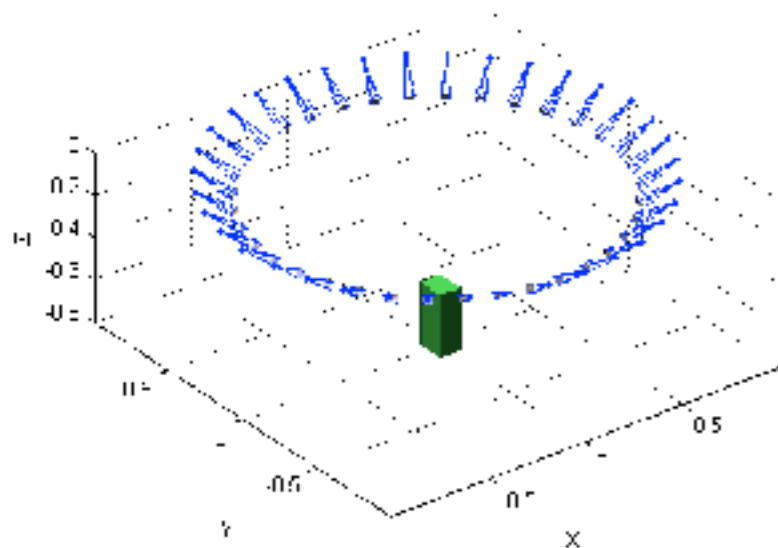
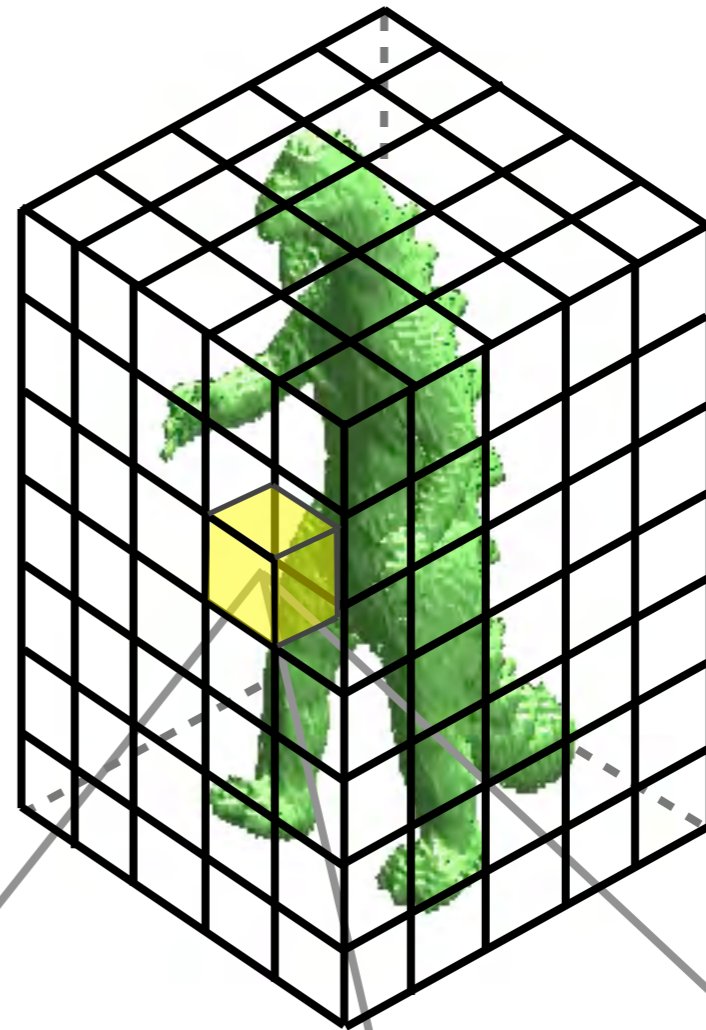


(b) Synthetic aperture sequence, with the focal plane moving away from the array computed using our parallax based calibration method. We get sharp focus at different depths, ranging from the bushes to the building facade.

Volumetric Stereo

- Discretise the scene using a grid of **voxels**
- Infer occupancy and colour of voxels by projecting to images





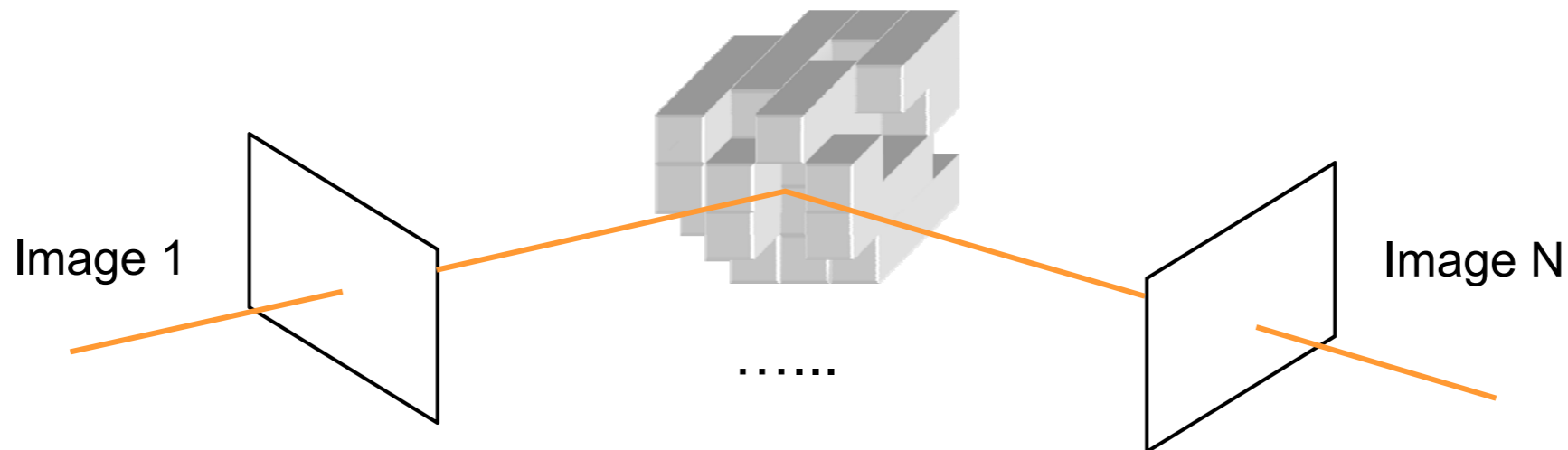
- Idea: visit all voxels in order, keep only photo-consistent voxels



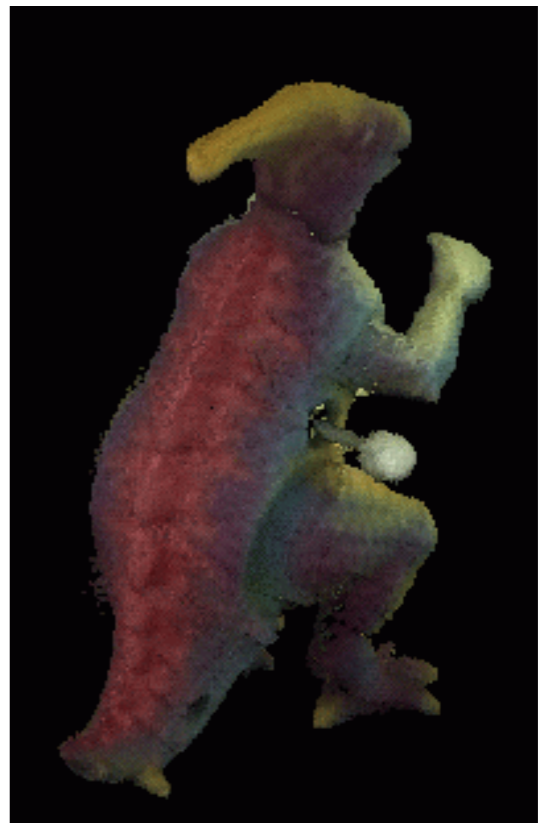
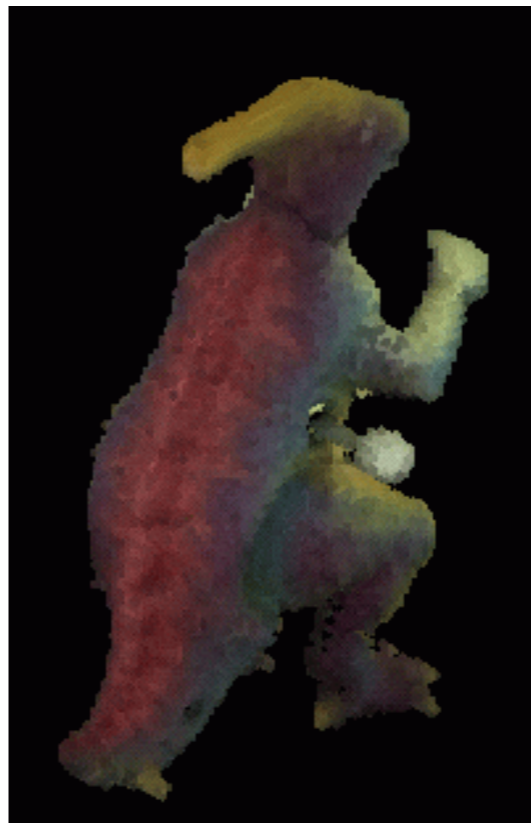
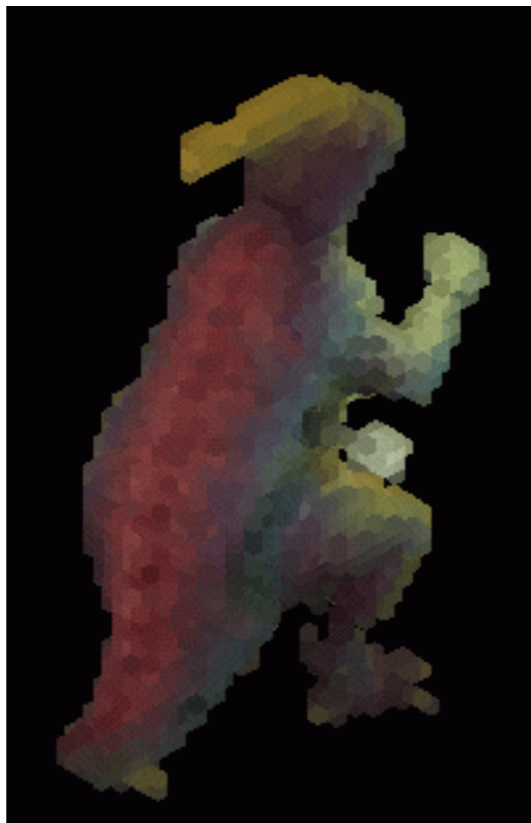
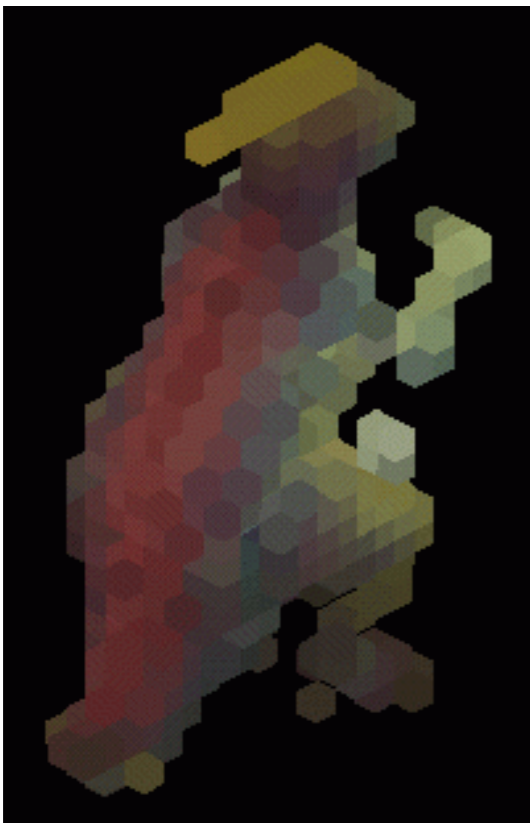
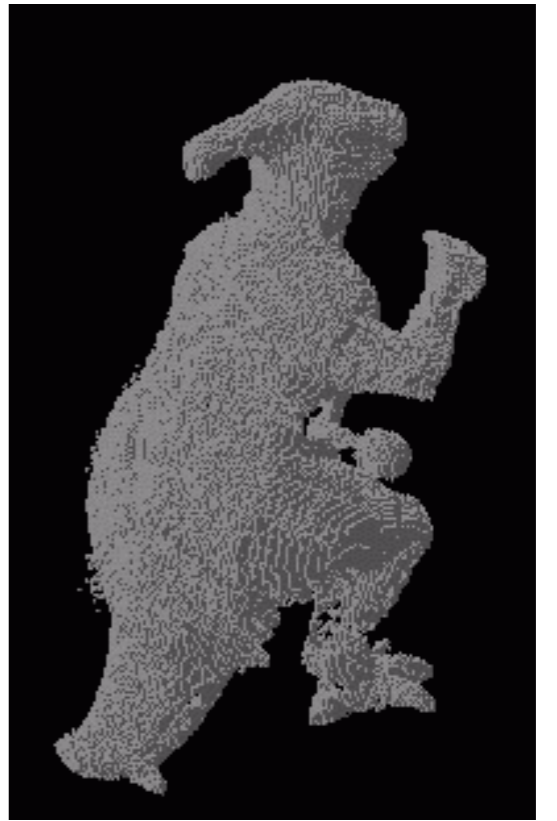
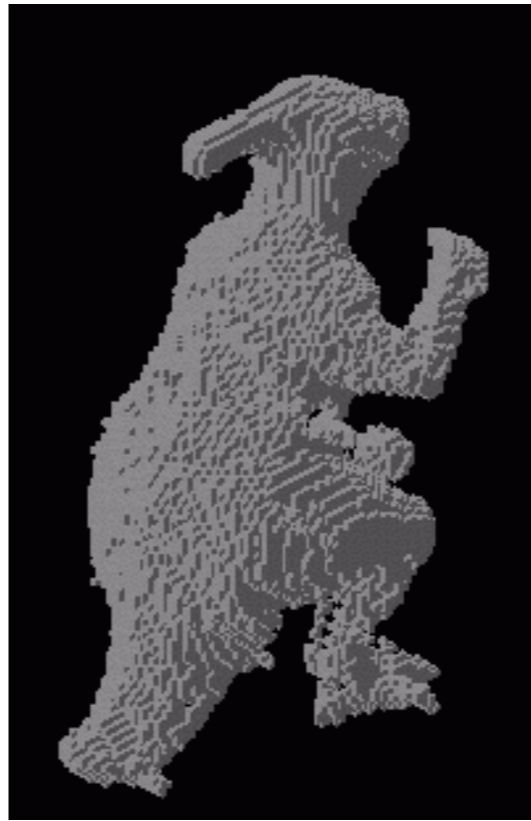
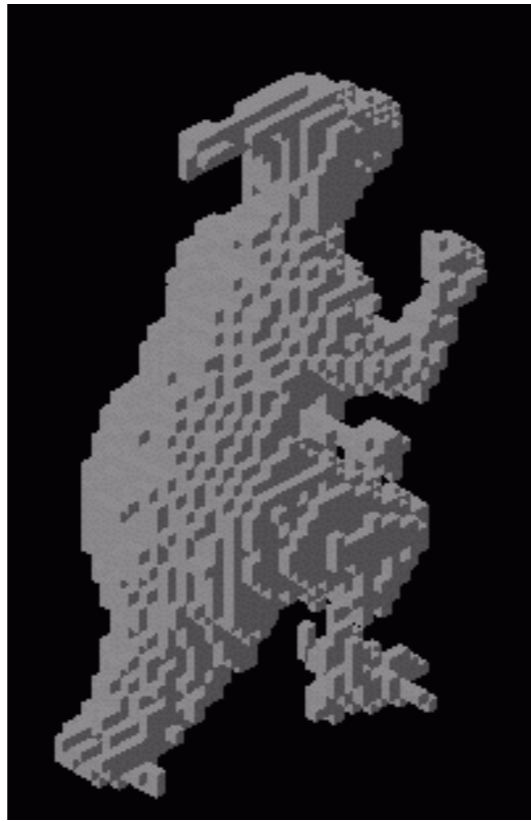
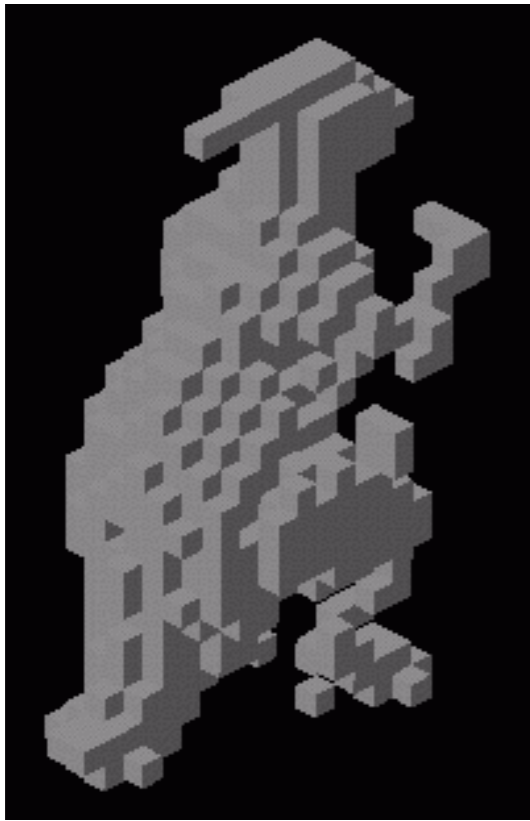
What is wrong with this idea?

Space Carving

- Space carving finds a voxel reconstruction that is consistent with the input images, taking into account visibility



- Initialise a volume containing the true scene
- Choose a voxel v on the surface
- Project v to all views where visible
- If v is not photo-consistent, remove it from the volume
- Repeat until all voxels are photo-consistent

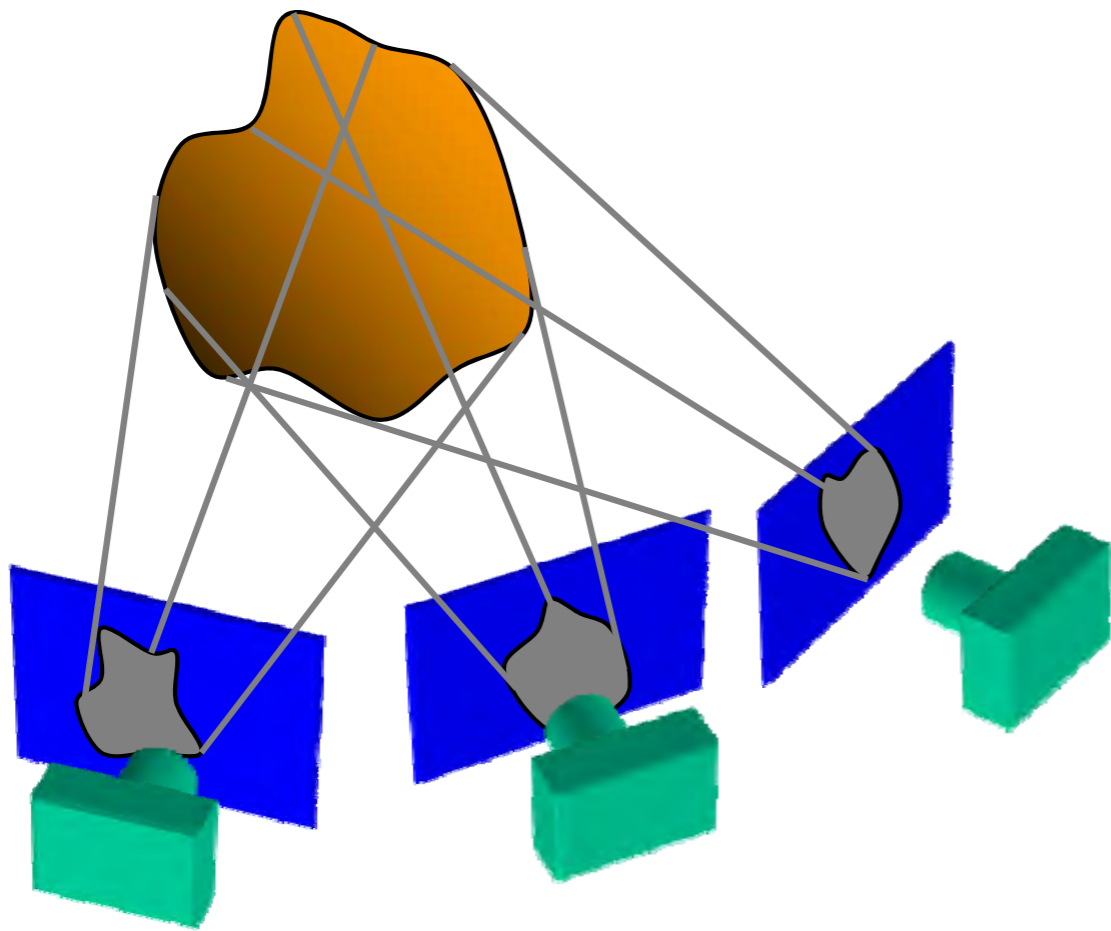


~1k voxels

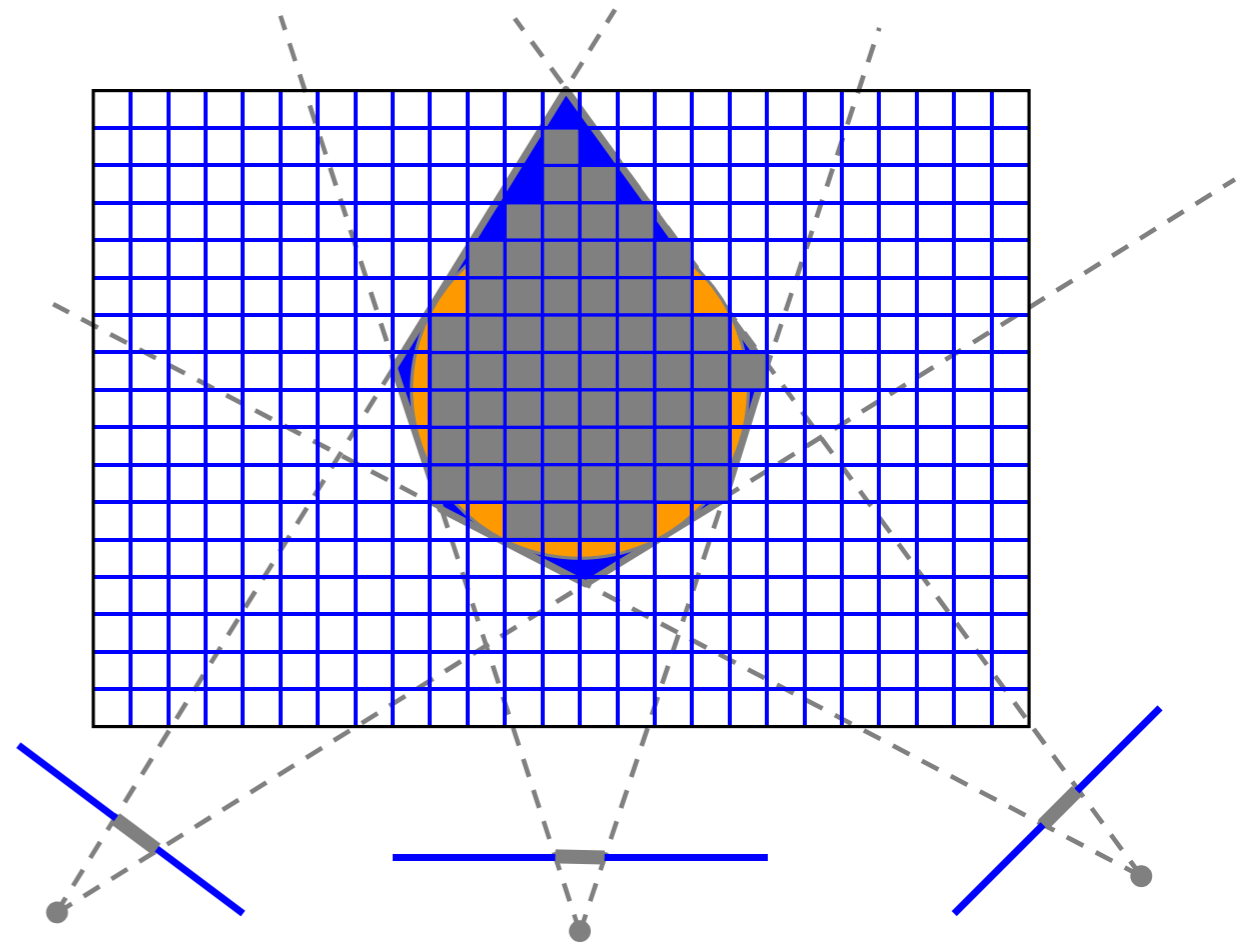
~70k voxels

Silhouette Intersection

- Consider the case of binary images (silhouettes)
- Voxel is part of the object if it lies in the silhouette in all views



Project volumes from each silhouette back into scene and intersect

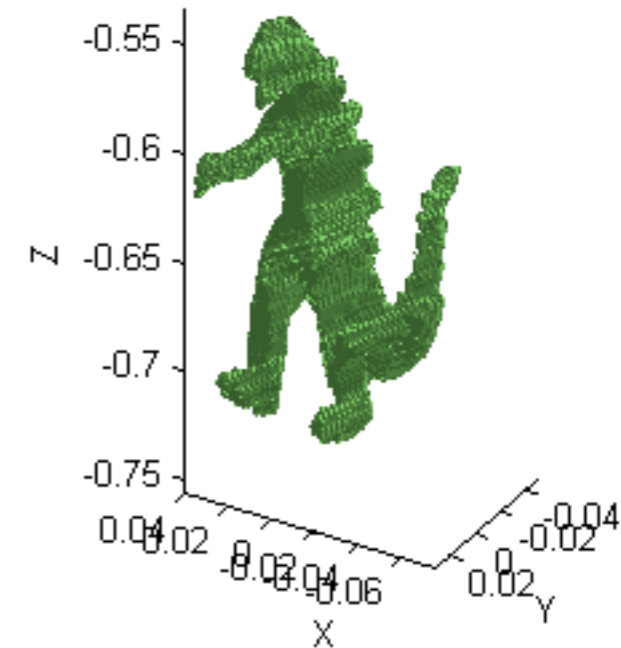
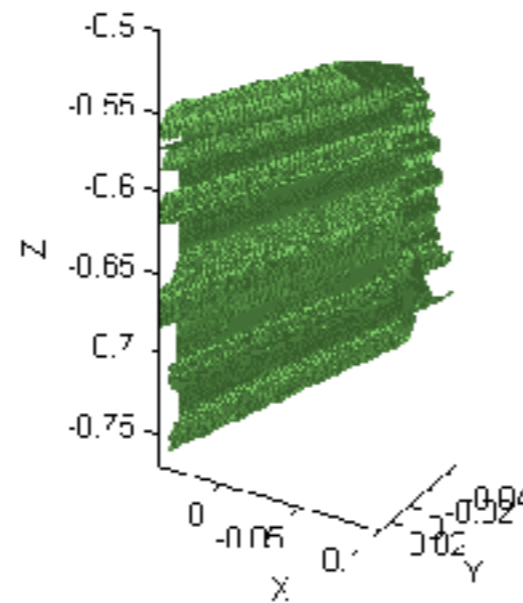
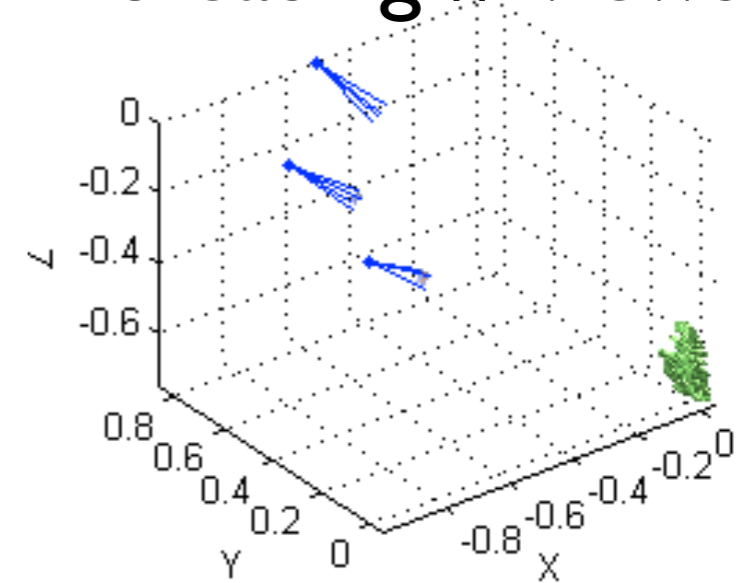
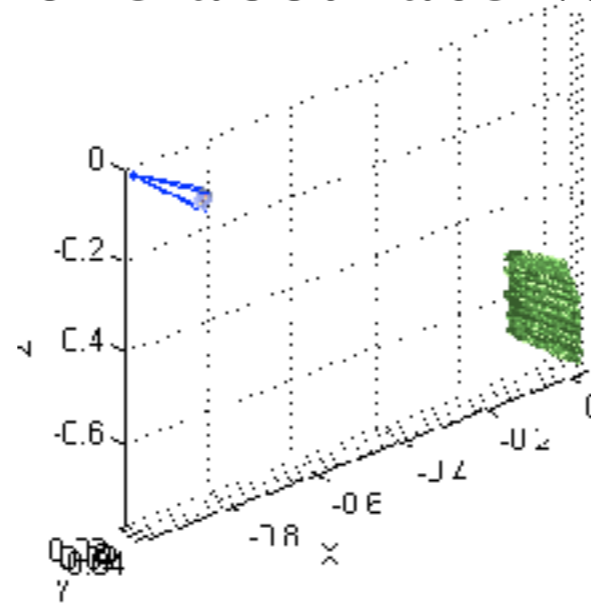


Voxel reconstruction is larger than object

[Seitz / Lazebnik] 41

Silhouette Intersection

- The intersection of back-projected silhouettes is called the **visual hull**, it is more accurate with increasing # views



[Ben Tordoff /
Mathworks]

1 camera

3 cameras

Depth Map Merging

- Idea: Nearby images have the most reliable stereo matches
- If we have a lot of images/pixels, we may not need to perform wide baseline matching



Depth Map Merging

- Select subsets of images and compute high confidence depth maps (e.g., keep only low SSD matches)
- Merge depth maps using robust fusion, e.g., using signed distance functions [Curless Levoy 1996]

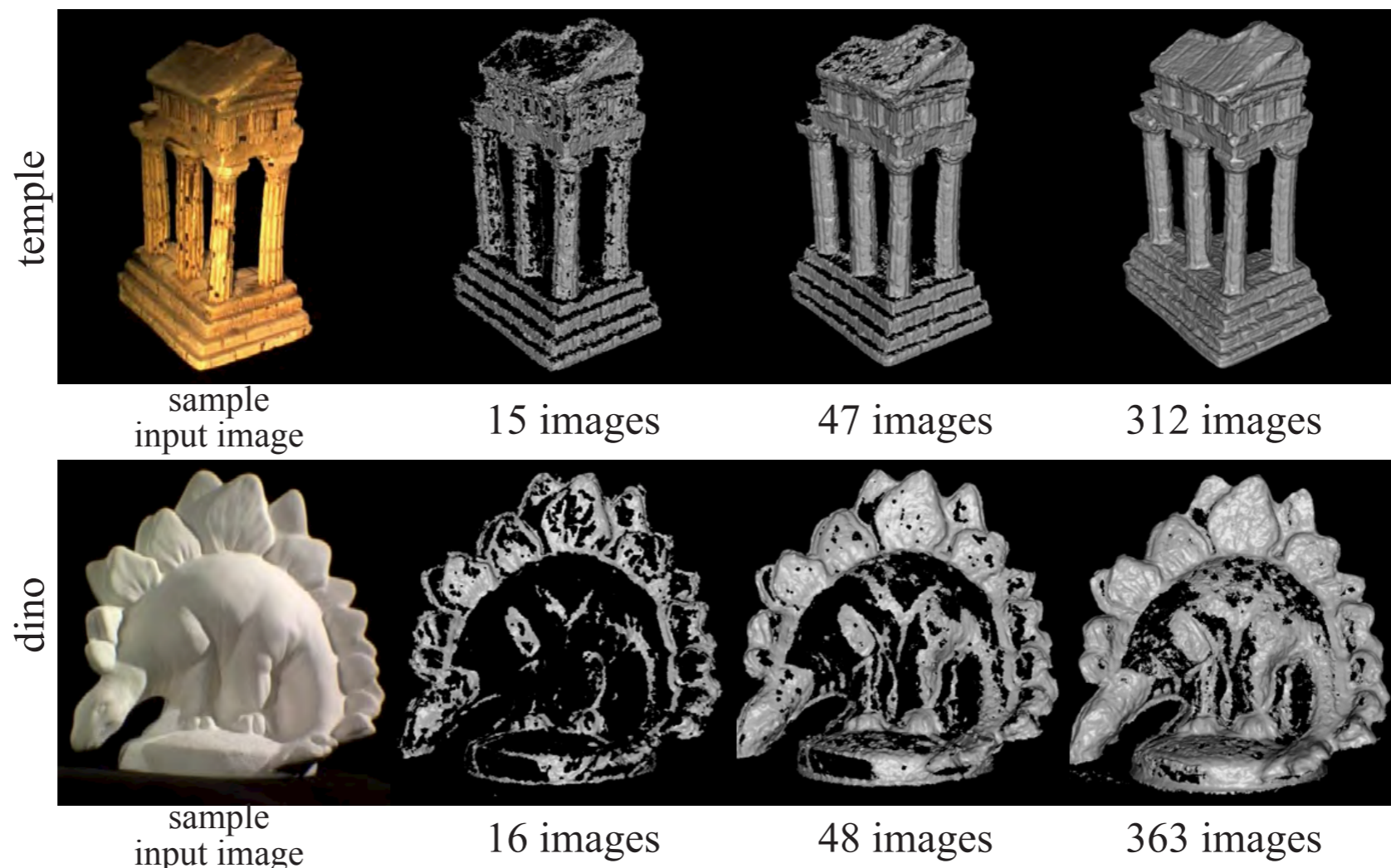
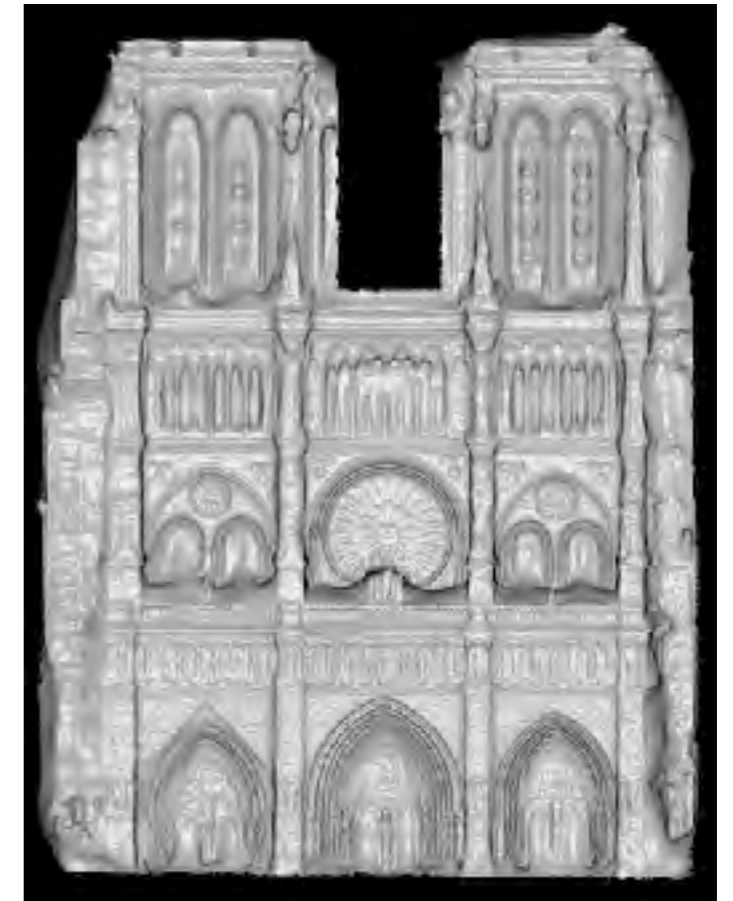
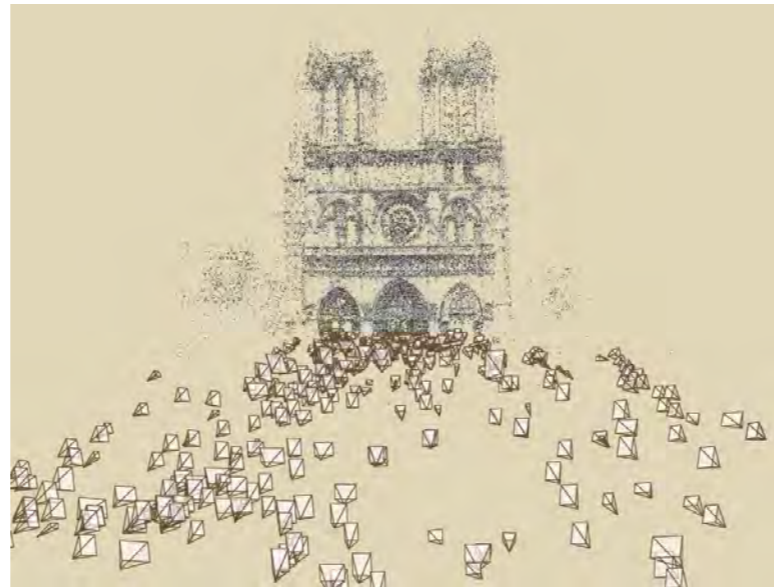
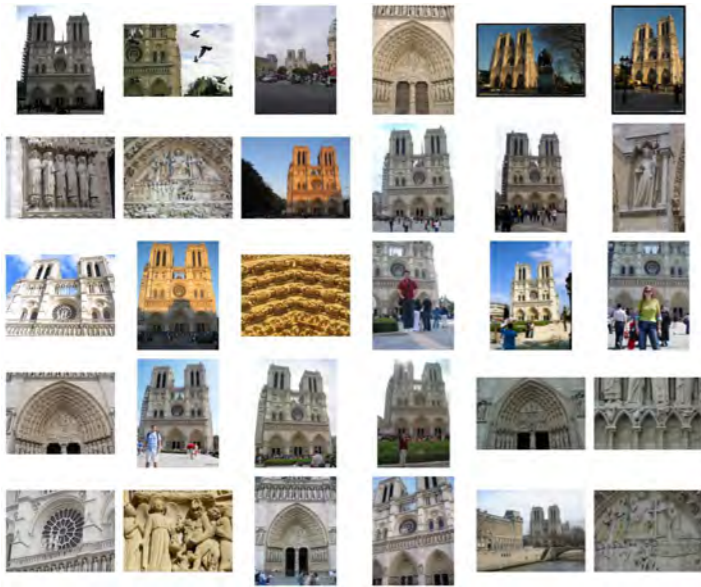


Photo Collections → 3D



- Depth map merging is practical for photo collections:
- Adaptable to complex geometry and large-scale scenes
- Robust to varied imagery and noise — select only subsets with good matches (don't try to match everything)

Neural Scene Representation

- Neural Radiance Fields, ~10s of input views



Photometric Stereo

- We can also get 3D information about the scene using one camera and **multiple lights**



- The most straightforward case of photometric stereo is to assume Lambertian reflectance

Photometric Stereo by Example

- Use object of known geometry, match colour patterns



Non-rigid Photometric Stereo with Colored Lights

C. Hernández¹, G. Vogiatzis¹, G.J. Brostow²,
B. Stenger¹ and R. Cipolla²

Toshiba Research Cambridge¹

University of Cambridge²

Next Lecture

- Depth, Flow