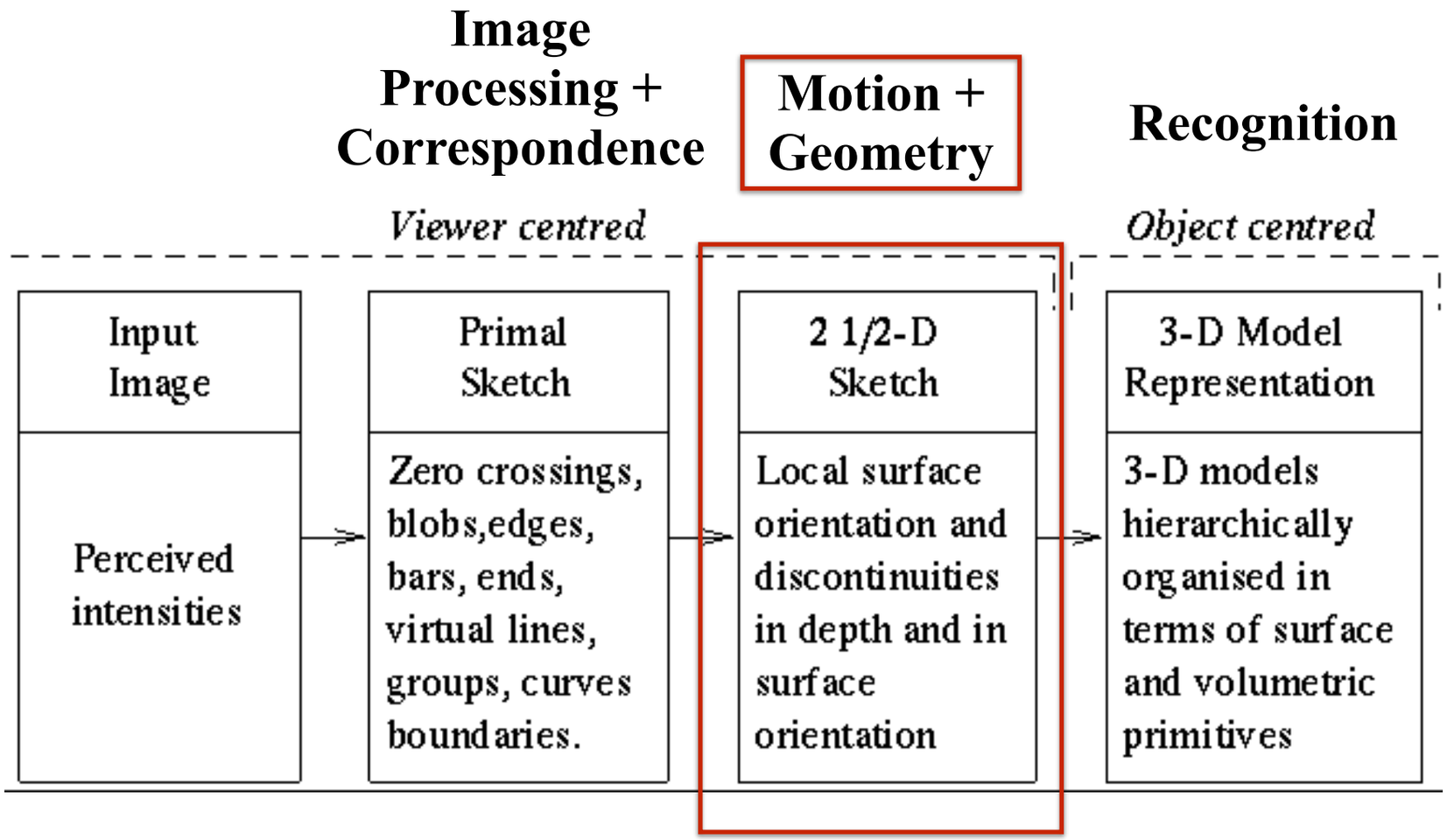
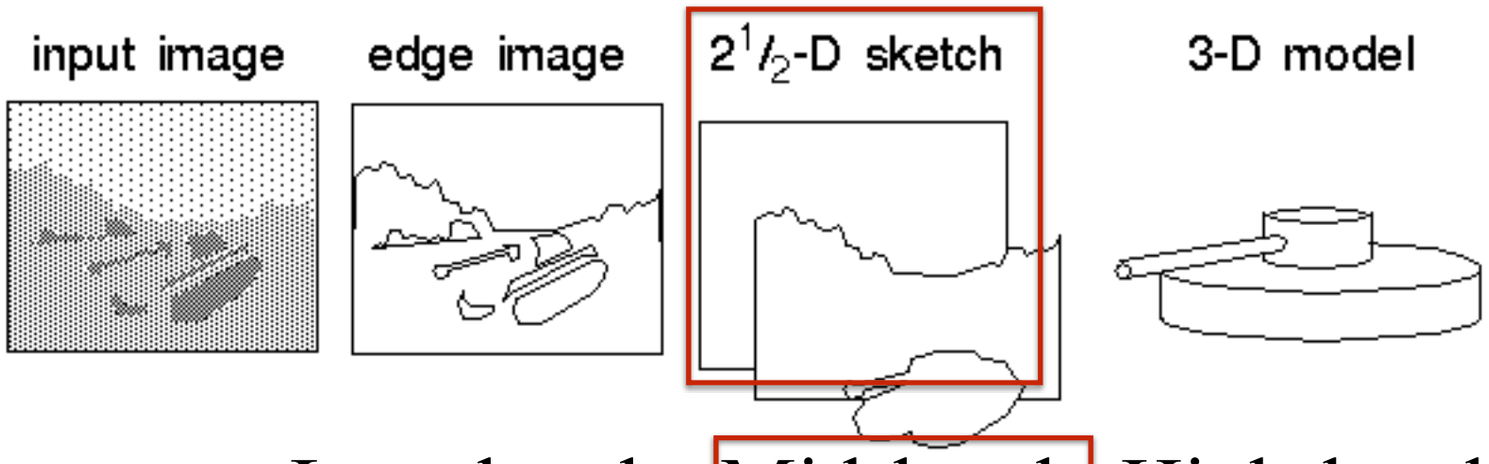
Multi-view Stereo

David Marr's Taxonomy of Vision





Low-level Mid-level High-level

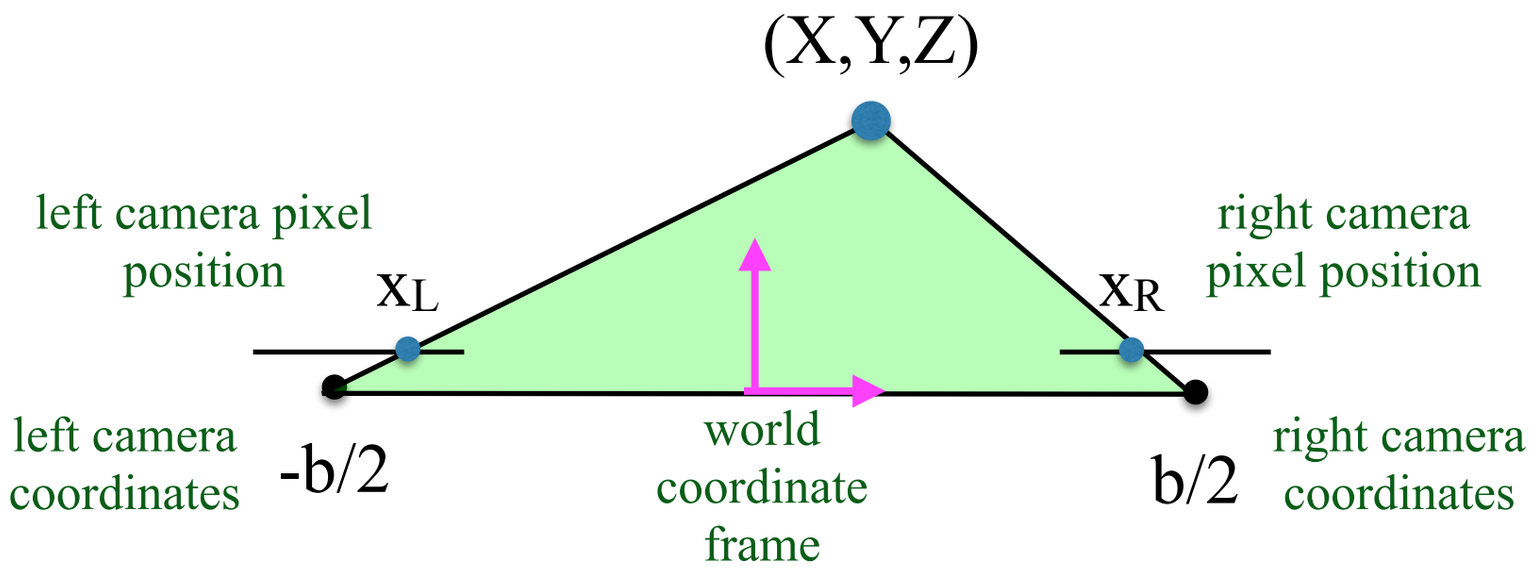
Multi-view geometry

Multiview stereo

- **triangulation**
- correspondence (dynamic programming)
- active illumination
- volumetric models / visiblity reasoning
- patch-based methods

Triangulation for Rectified Stereo Pairs

Top-down view where world coordinates are centered between cameras



left camera pixel position $x_L = f\frac{X + \frac{b}{2}}{Z}$ $x_R = f\frac{X - \frac{b}{2}}{Z}$ right camera pixel position

$d = x_L - x_R = \frac{bf}{Z}$ is the **disparity** between corresponding left and right image points

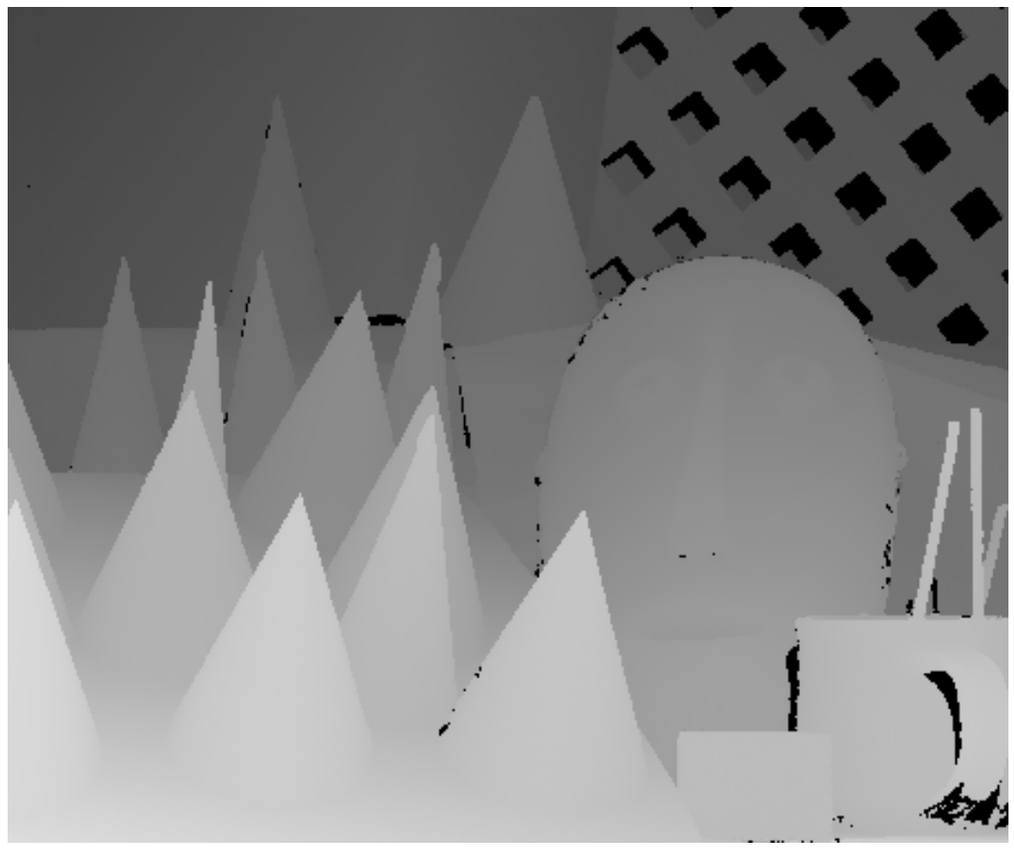
- inverse proportional to depth Z
- disparity increases with baseline b

$$\Rightarrow \quad Z = \frac{bf}{(x_L - x_R)} \qquad X = \frac{b(x_L + x_R)}{2(x_L - x_R)} \qquad Y = \frac{b(y_L + y_R)}{2(x_L - x_R)}$$

Disparity Maps

$$d = x_L - x_R = \frac{bf}{Z}$$

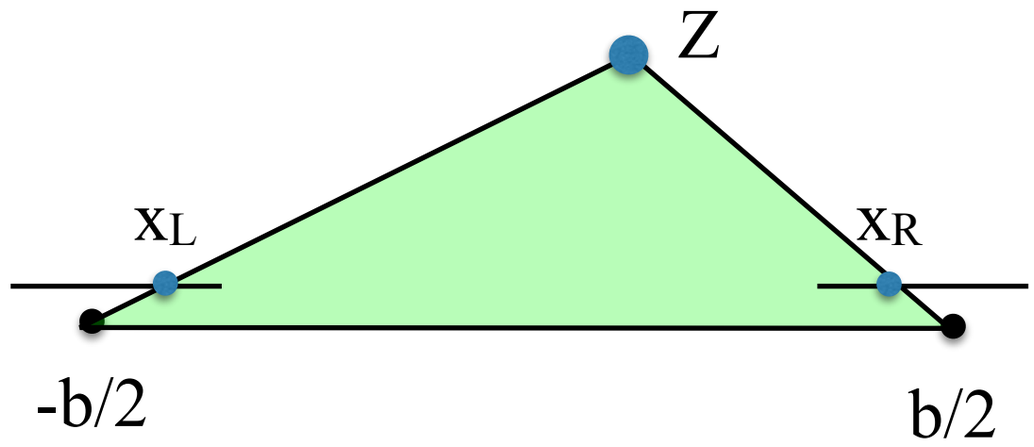
Disparity values (0-64)



Note how disparity is larger (brighter) for closer surfaces.

It is often more convenient to work with disparity (units of pixels) rather than depth (units of meters that depend on intrinsics)

Numerical stability



$$d = x_L - x_R = \frac{bf}{Z}$$

Scene + camera variables: Z,f,b

Dependent variable: d = function(Z,f,b)

How does a small error in disparity d result in a change in the estimated depth Z in terms of scene + camera variables?

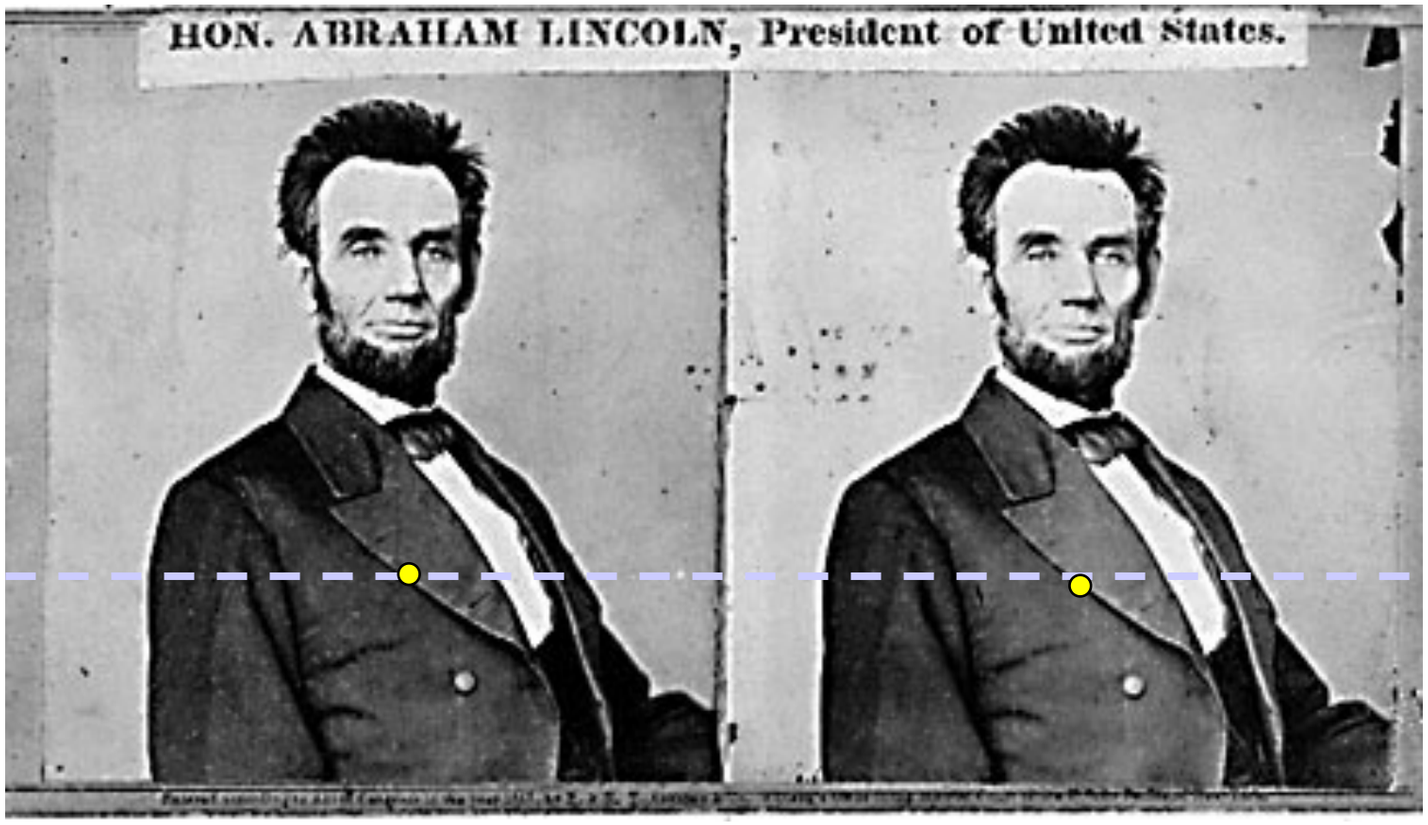
$$Z = \frac{bf}{x_L - x_R} = \frac{bf}{d} \quad \Rightarrow \quad \frac{\partial Z}{\partial d} = -\frac{bf}{d^2} = -\frac{Z^2}{bf}$$

1. Error inversely proportional to baseline (larger baselines increase numerical stability)
2. If we measure the baseline in pixel coordinates, then increasing the camera resolution will increase the baseline! (more sensitivity to small disparities)
3. Error increases quadratically with depth (hard to reconstruct far away points)

Multiview stereo

- triangulation
- **correspondence (dynamic programming)**
- active illumination
- volumetric models / visiblity reasoning
- patch-based methods

Basic Stereo Algorithm



For each epipolar line

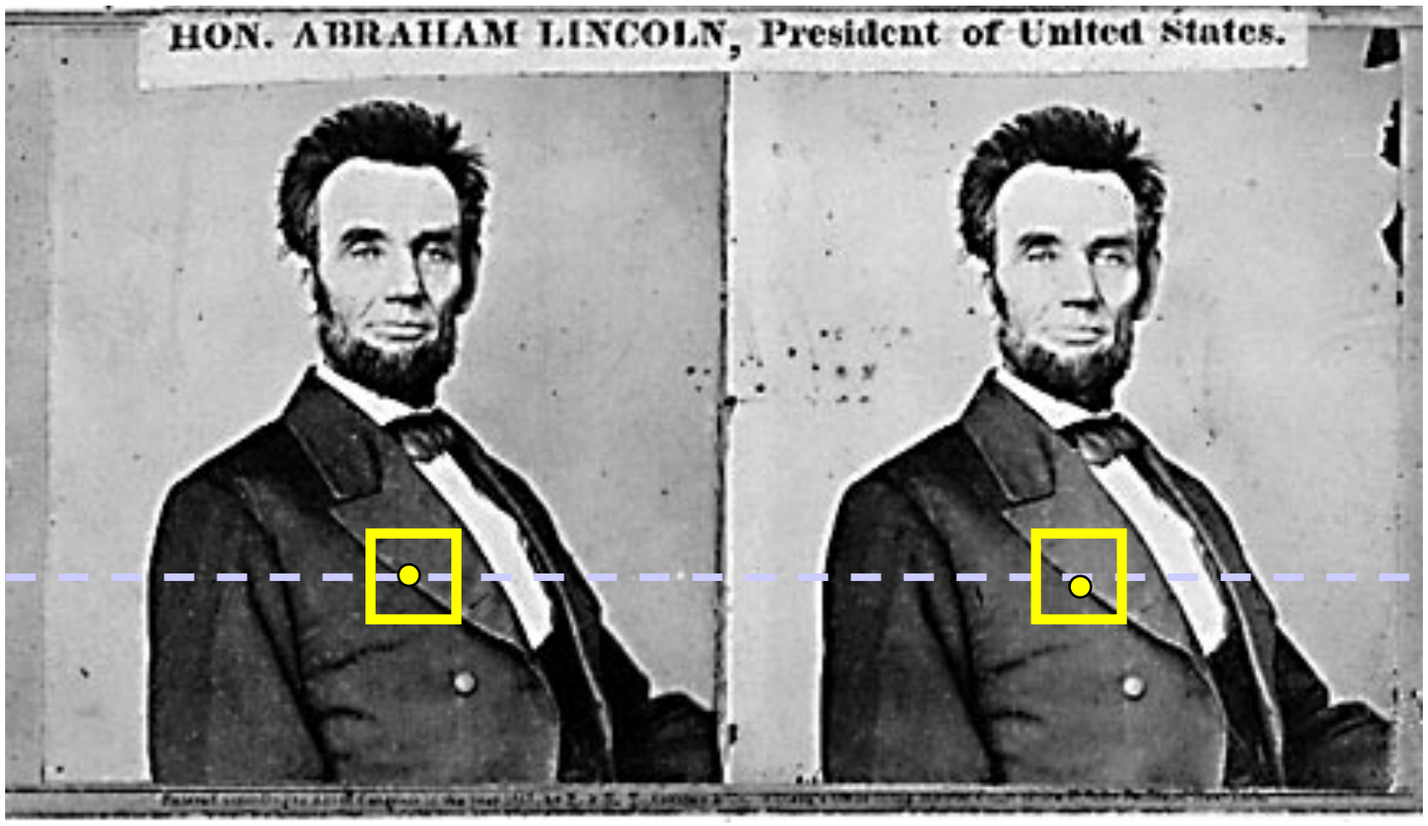
For each pixel in the left image

- compare with every pixel on same epipolar line in right image
- pick pixel with minimum match cost

Improvement: match ***windows***

- (Normalized) Correlation, Sum of Squared Difference (SSD), Sum of Absolute Differences (SAD), etc…

Basic Stereo Algorithm



For each epipolar line

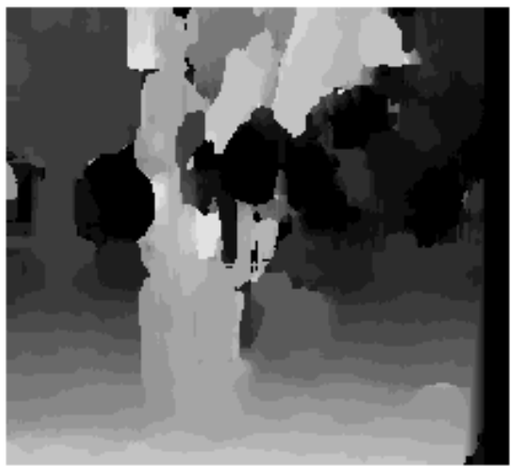
For each pixel in the left image

- compare with every pixel on same epipolar line in right image
- pick pixel with minimum match cost

Improvement: match ***windows***

- (Normalized) Correlation, Sum of Squared Difference (SSD), Sum of Absolute Differences (SAD), etc…

Matching window

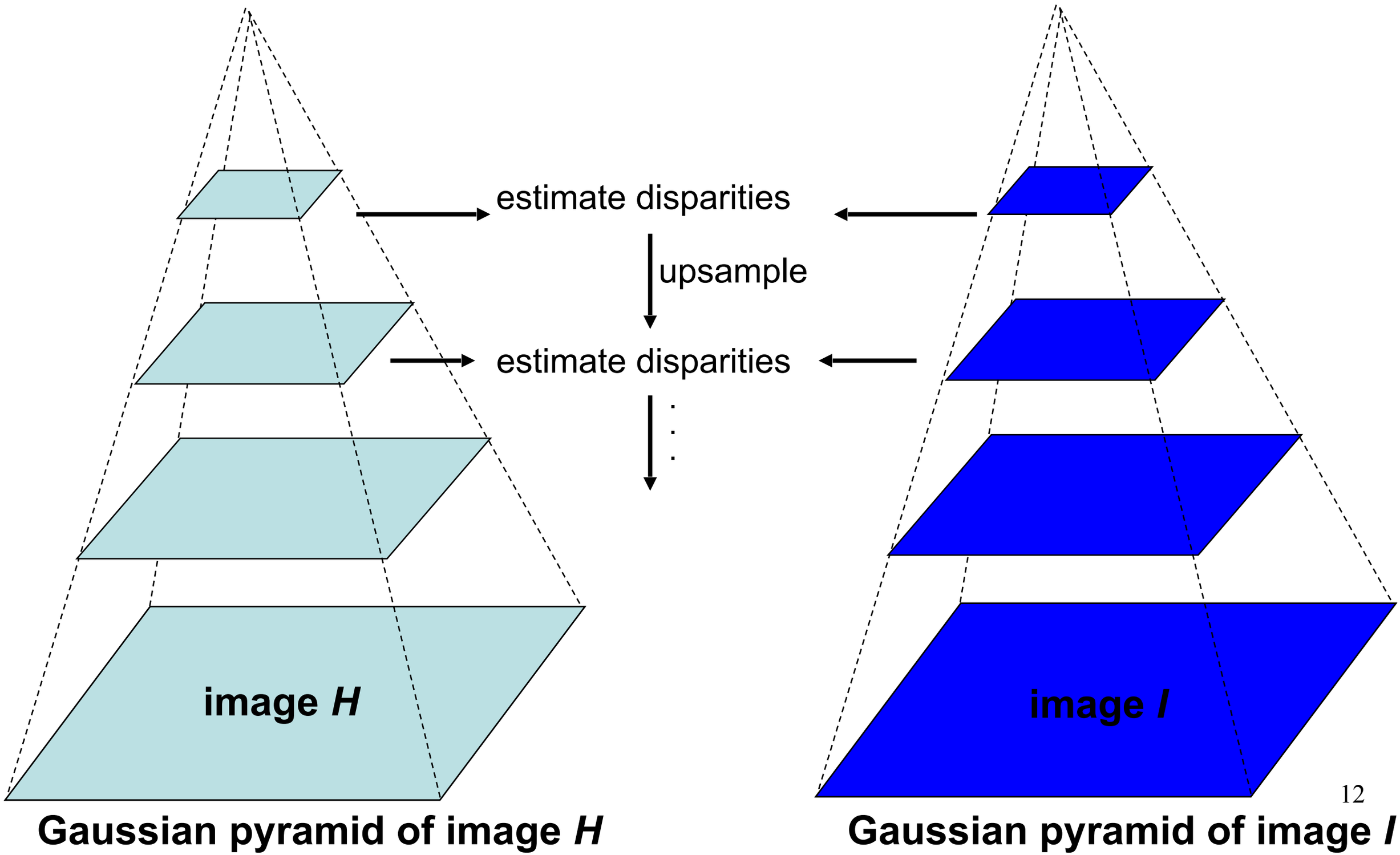


Small matching window (better localization but noisy)

Large matching window (fewer errors but less precise)

How to get the best of both worlds?

Coarse-to-fine stereo



Gaussian pyramid of image *H*

Gaussian pyramid of image *I*

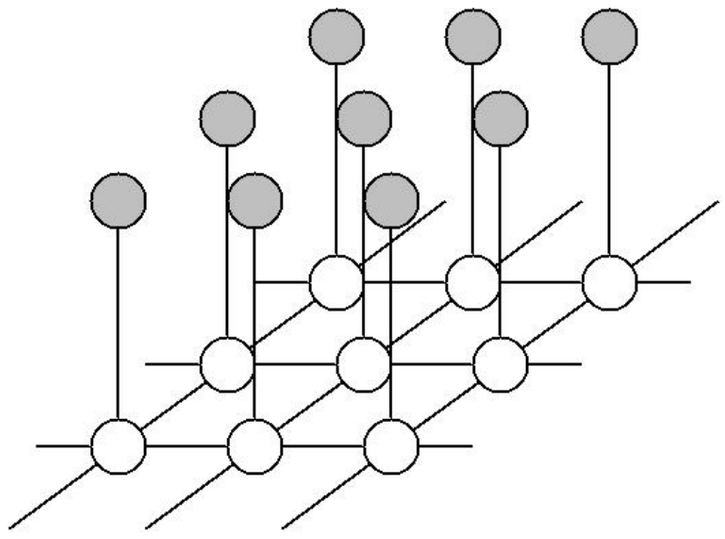
But there is a relationship between disparity of neighboring pixels

z_i = disparity

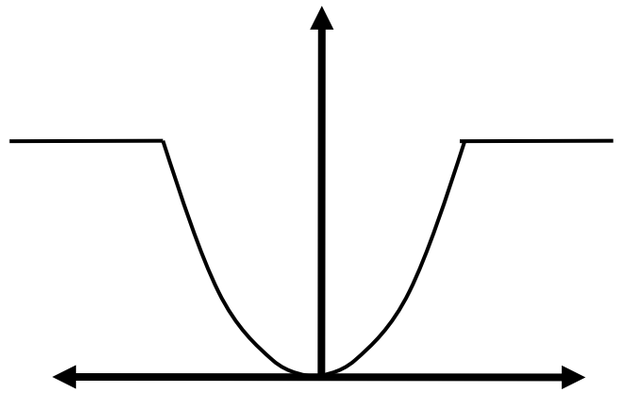
Find the disparity z_i that produces the best match across images

Nearby pixel disparities are similar

$$E(z) = \sum_{i \in V} \phi_i(z_i) + \sum_{ij \in E} \psi_{ij}(z_i, z_j)$$

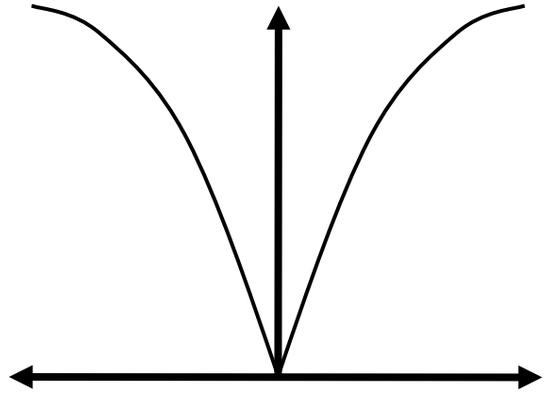


Robust error functions to handle “outliers”



truncated quadratic

$$\rho_{\alpha,\lambda}(x) = \begin{cases} \lambda x^2 & \text{if } |x| < \frac{\sqrt{\alpha}}{\sqrt{\lambda}} \\ \alpha & \text{otherwise} \end{cases}$$



lorentzian

$$\rho_\sigma(x) = log\left(1 + \frac{1}{2}(\frac{x}{\sigma})^2\right)$$

Discrete disparity estimation via graph optimization

z_i = disparity

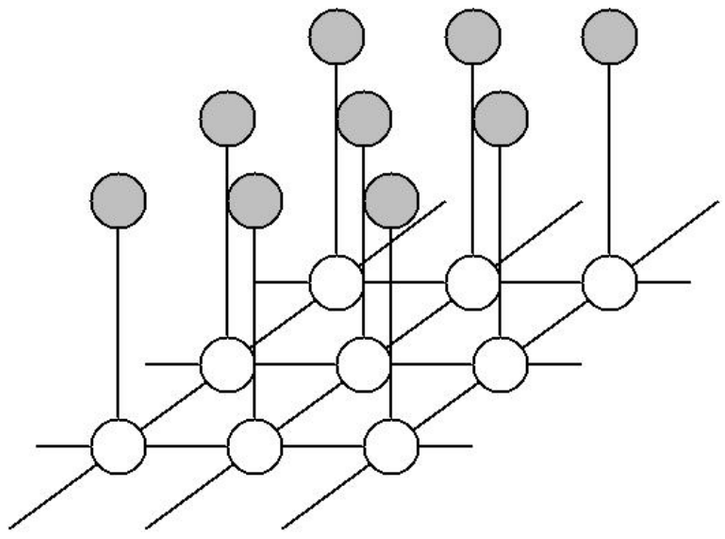
$$\phi_i(z_i) = \rho(||I_2(x_i + z_i, y_i) - I(x_i, y_i)||)$$
$$\psi_{ij}(z_i, z_j) = \rho(z_i - z_j)$$

Use robust error function
ρ to handle discontinuities

Find the disparity z_i that produces the best match across images

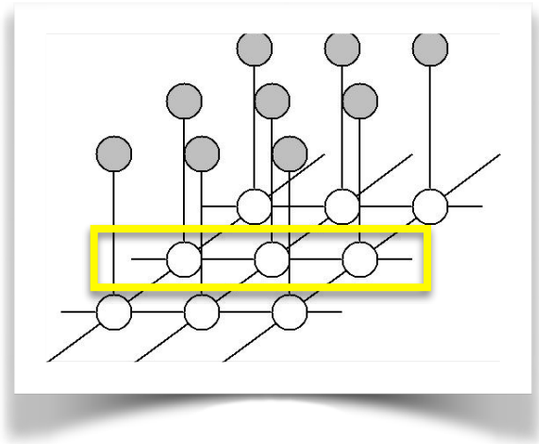
Nearby pixel disparities are similar

$$E(z) = \sum_{i \in V} \phi_i(z_i) + \sum_{ij \in E} \psi_{ij}(z_i, z_j)$$



Solve with graph optimization algorithms (e.g., mincost / maxflow graphcuts)

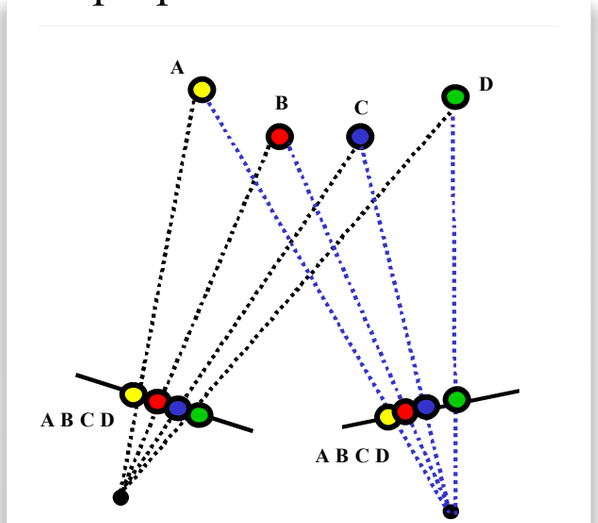
Special case: 1D single-scan-line consistency



Left Image

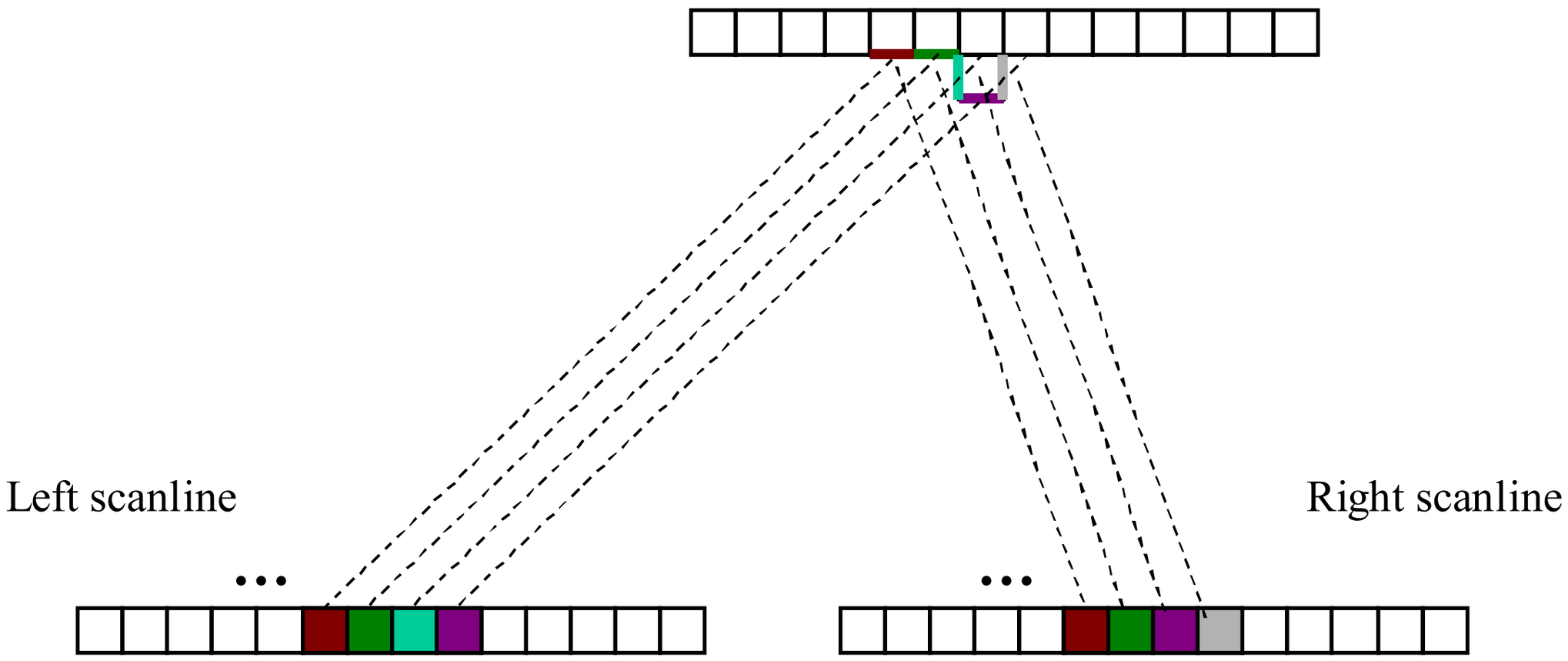
Right Image

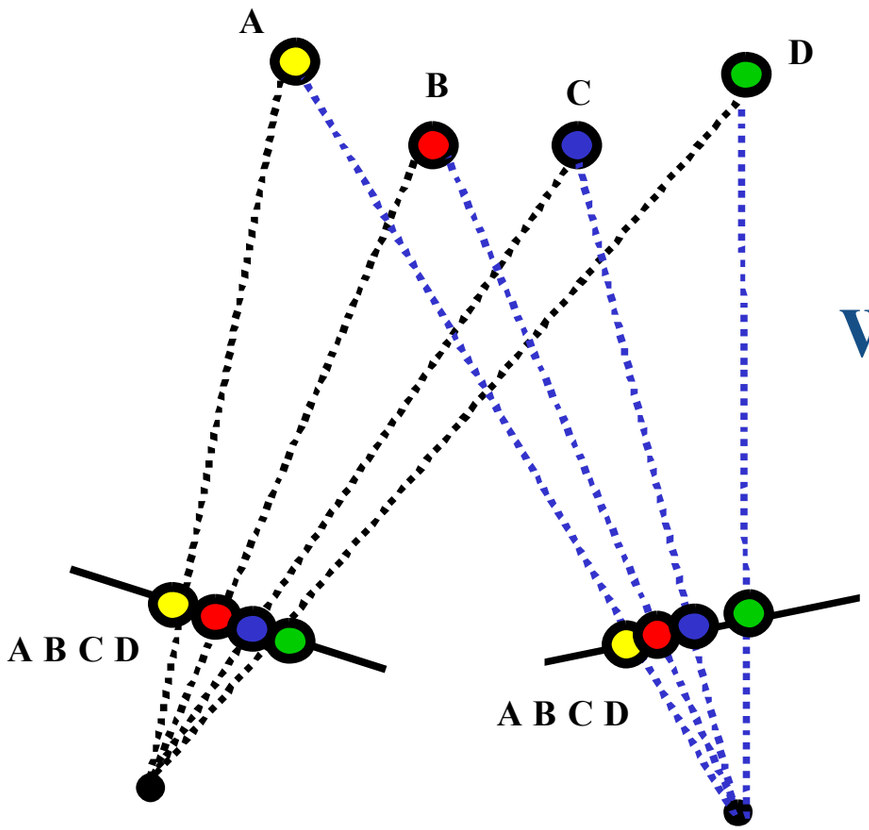
Often, we can enforce global scanline properties such as left-to-right ordering between two views!



1D ordering consistency

…can used to model occlusions (patches that appear in one view but are occluded in others)



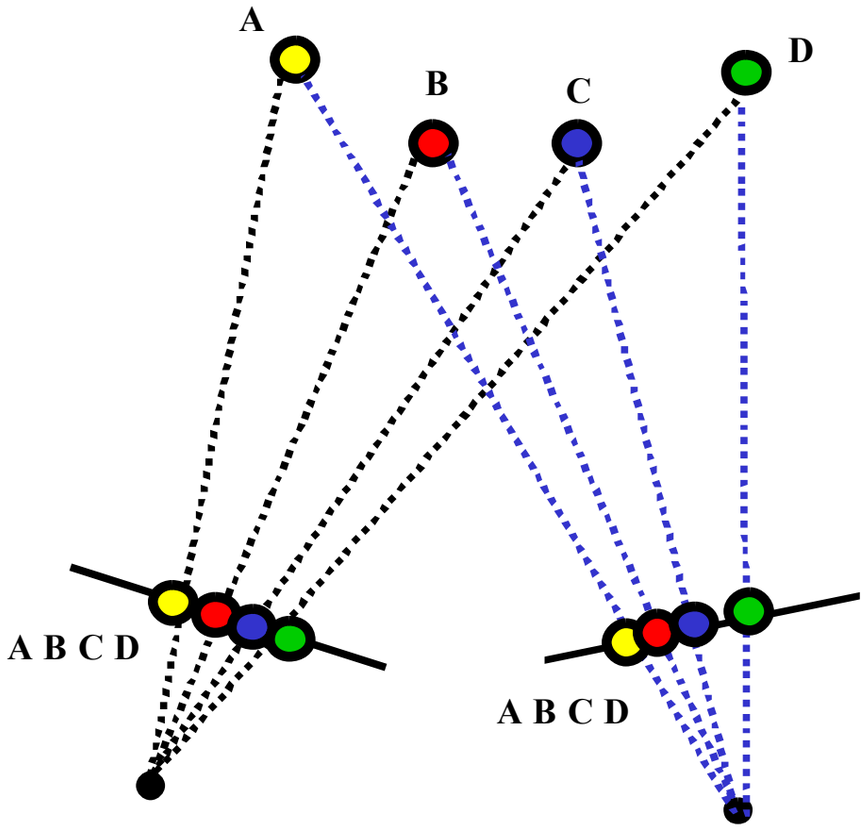


Ordering constraint…

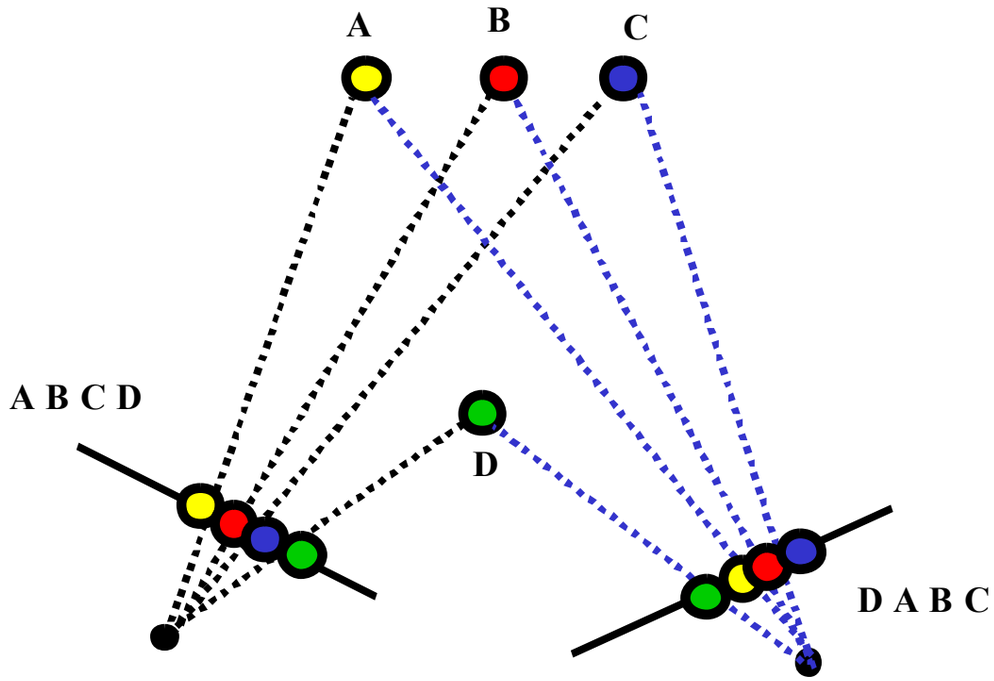
When will the ordering constraint not hold?

Imagine moving point D

Failures of ordering constraint



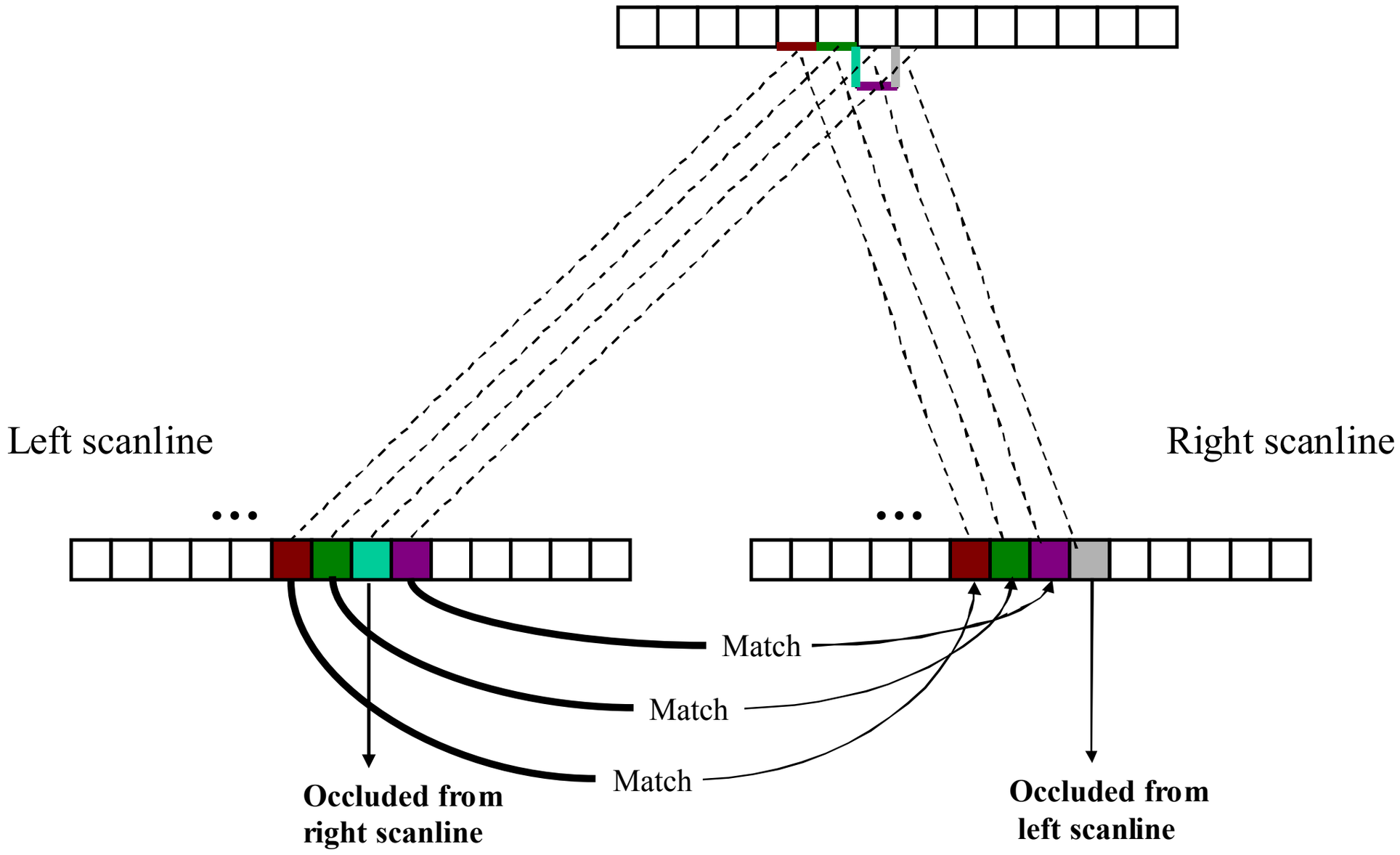
Ordering constraint…



…and its failure

1D ordering consistency

…can used to model occlusions (patches that appear in one view but are occluded in others)



Ordering consistency

$$\phi_i(z_i) = \rho(||I_2(x_i + z_i, y_i) - I(x_i, y_i)||)$$

$$E(z) = \sum \phi_i(z_i) + \psi(z_1, z_2, \ldots)$$

where ψ assigns a high cost to orderings with swaps

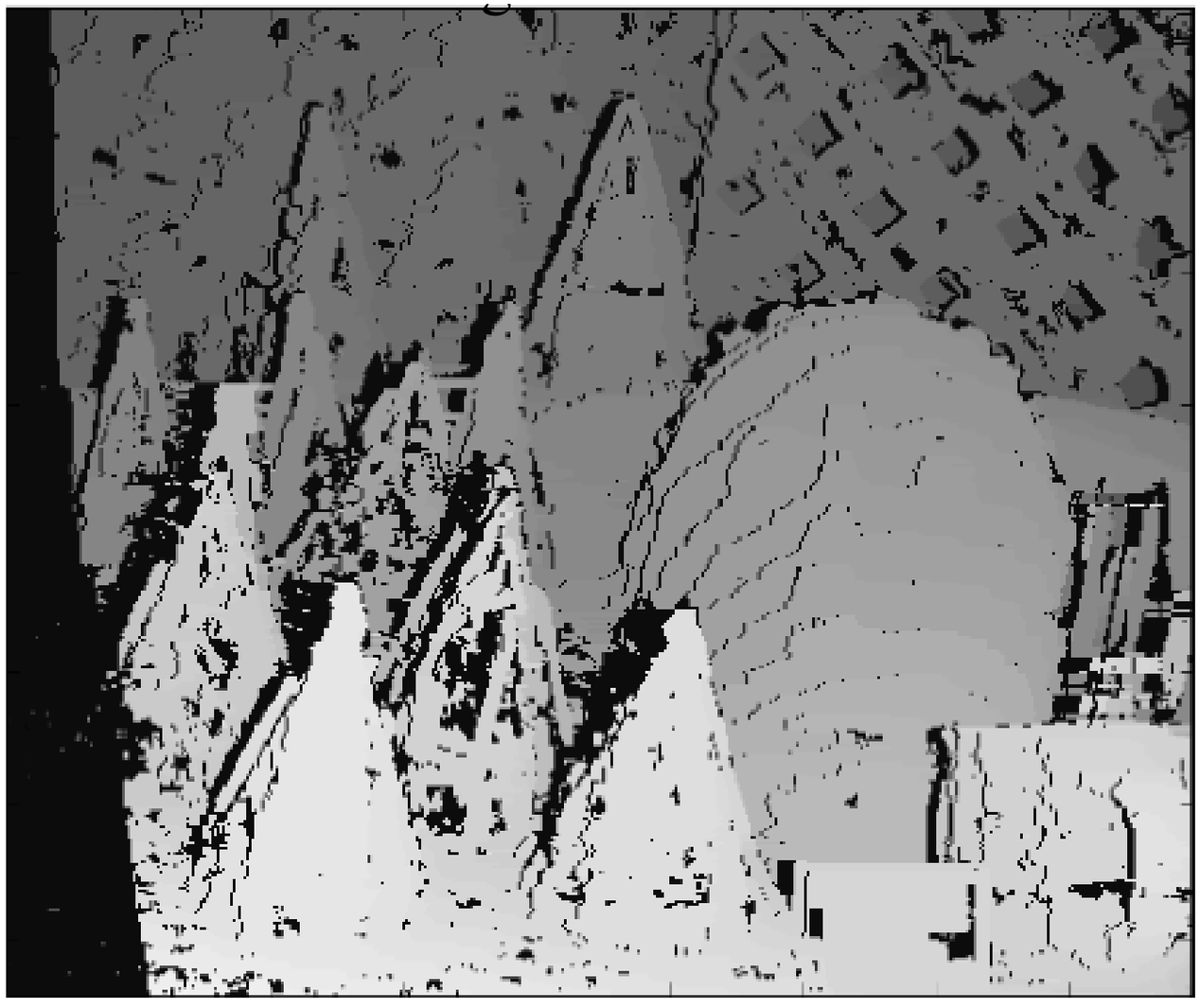
Left Image

Right Image

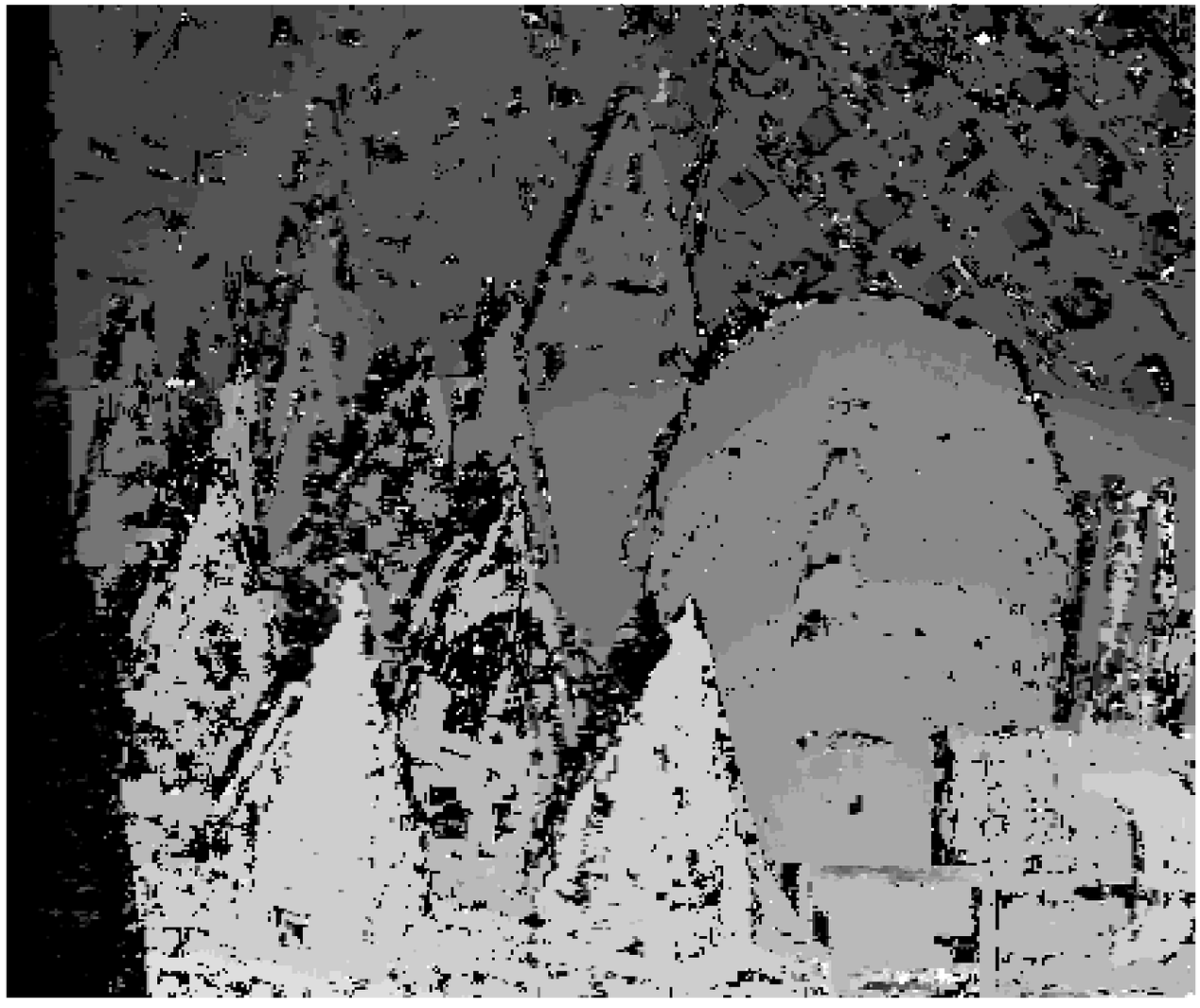
Dissimilarity Values
(1-NCC) or SSD

Results

Reasoning about ordering and neighboring pixels



Independent pixels

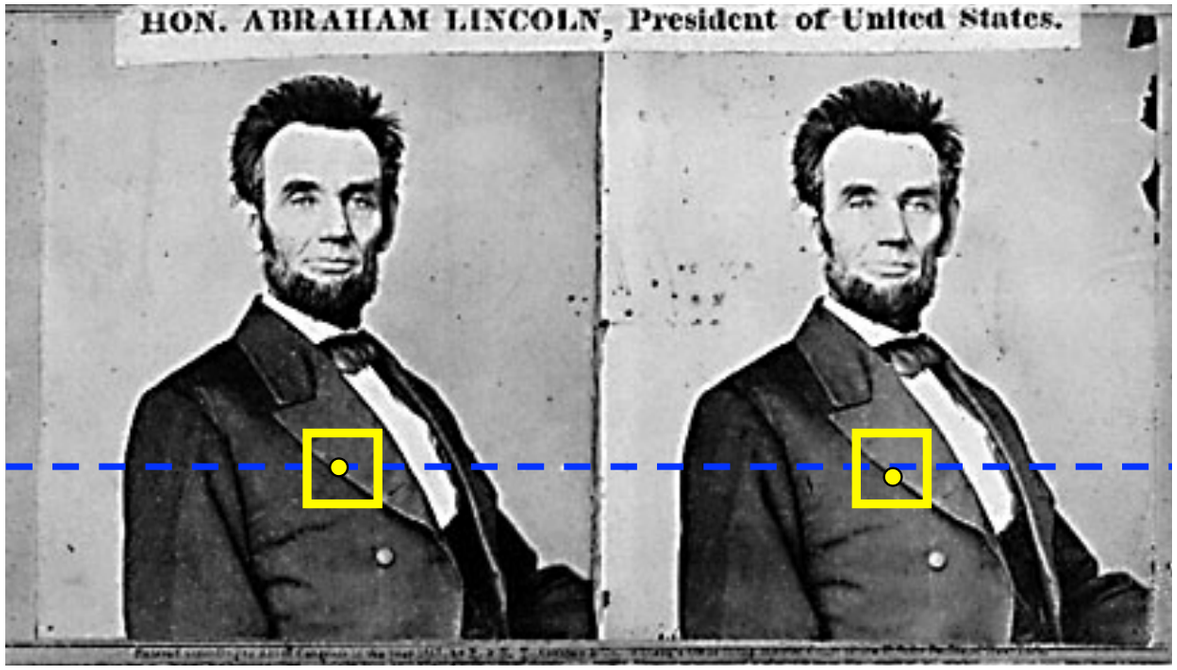


Black pixels = cannot find a match

Multiview stereo

- triangulation
- scanline correspondence (dynamic programming)
- **active illumination**
- volumetric models / visiblity reasoning
- patch-based methods

Hard part: find matching points



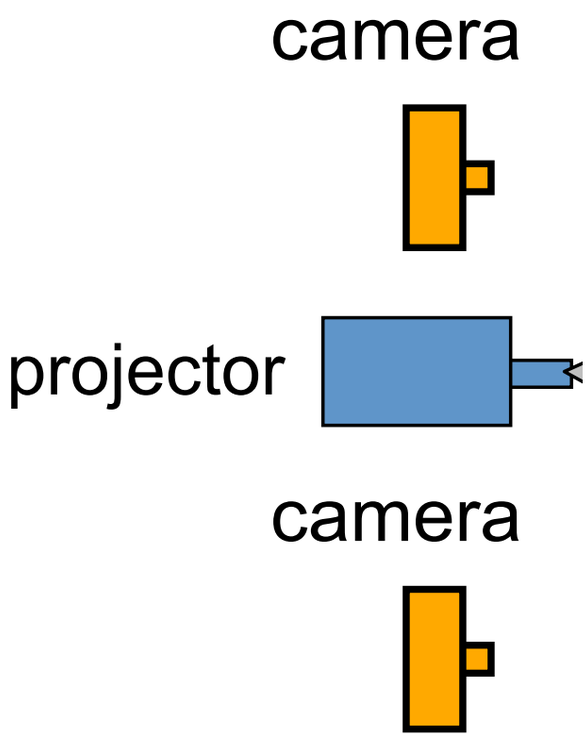
If patches look distinctive, they'll be easy to match
But lots of patches are not distinctive

Stereo matching

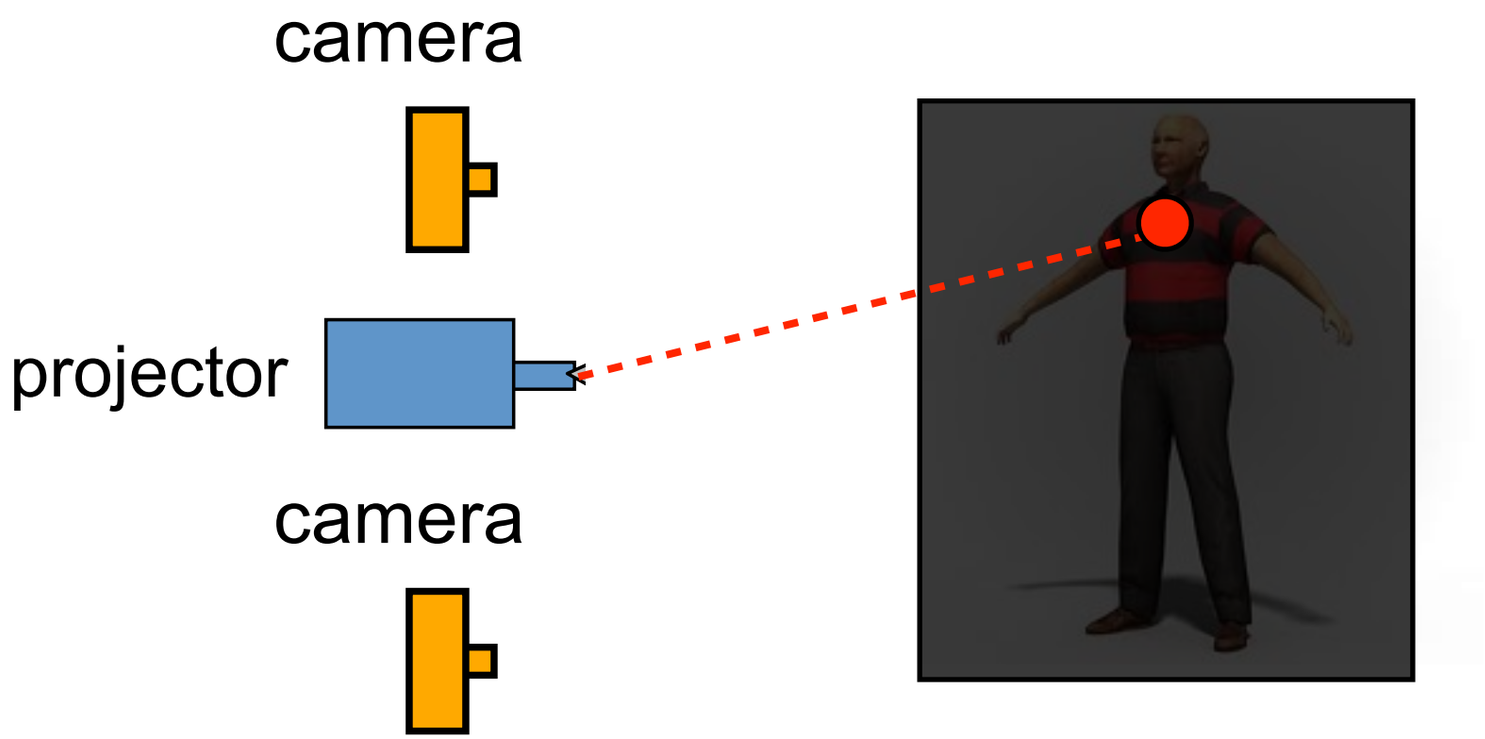
camera

camera

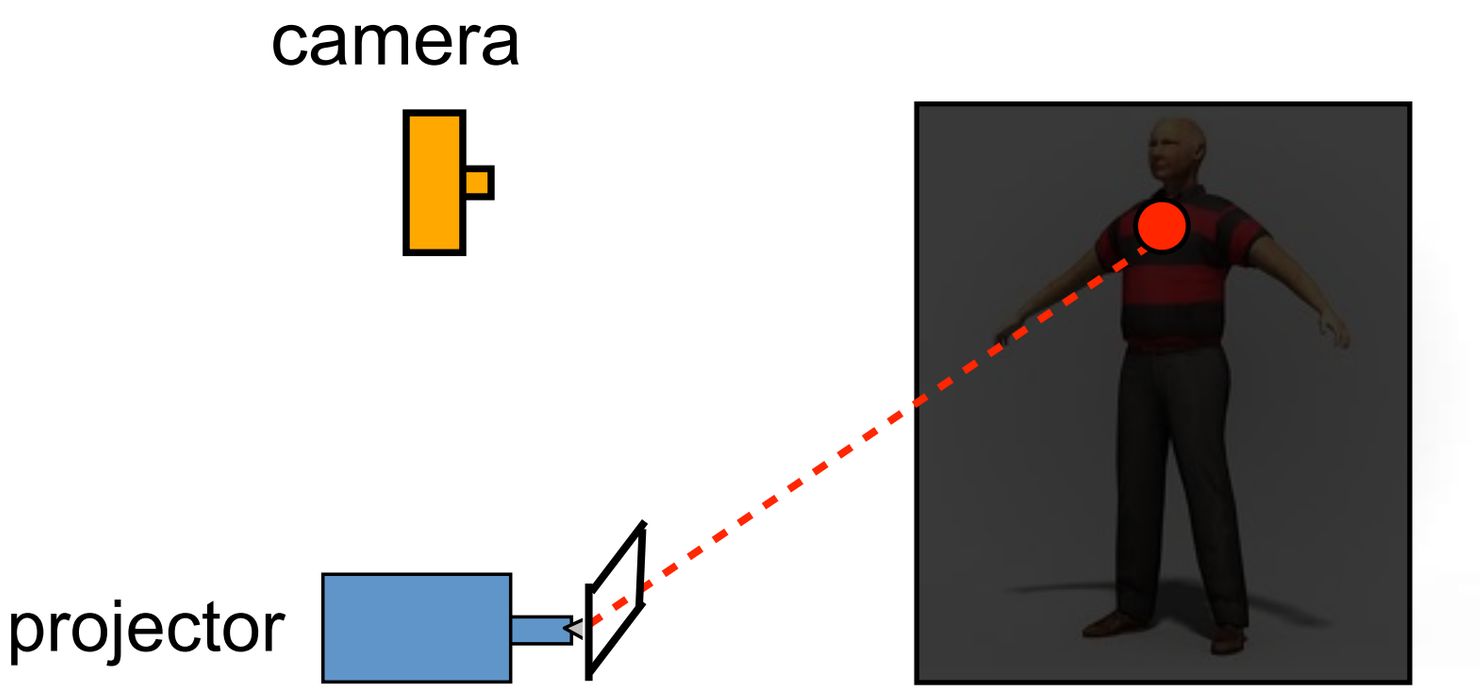
Trick: use a projector to project a distinctive appearance



Trick: use a projector to project a distinctive appearance

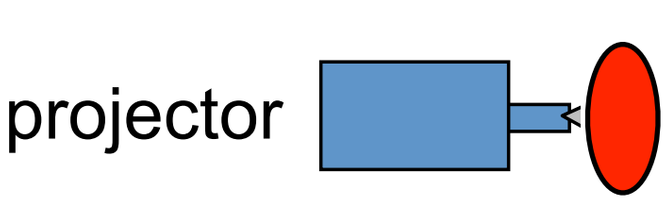


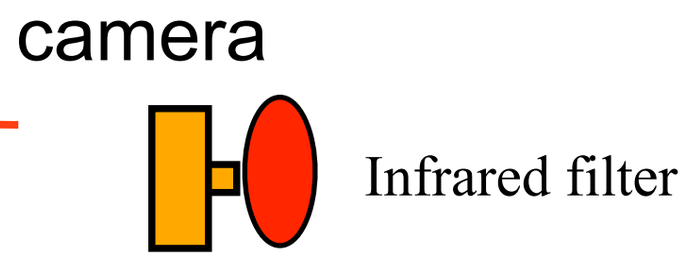
Trick: use a projector to project a distinctive appearance



Use just a projector and 1 camera

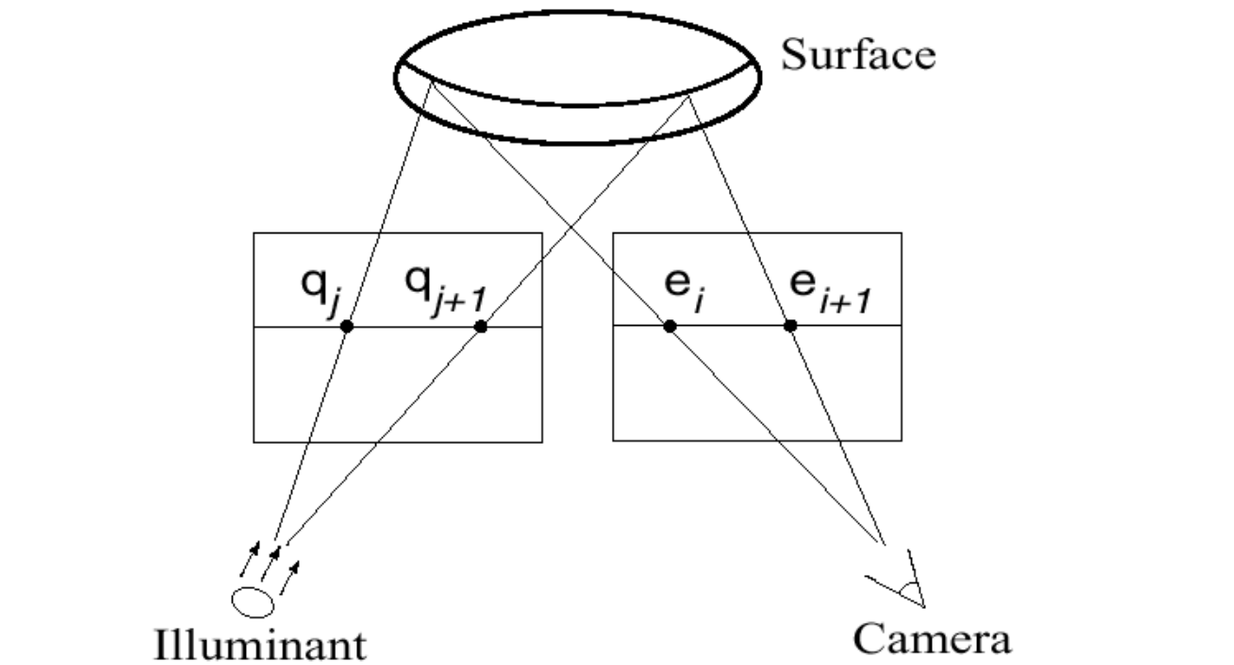
Kinect

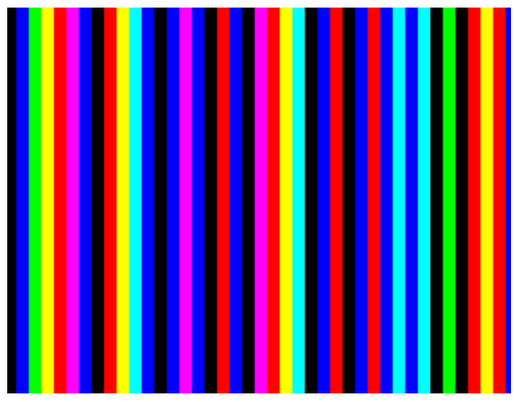


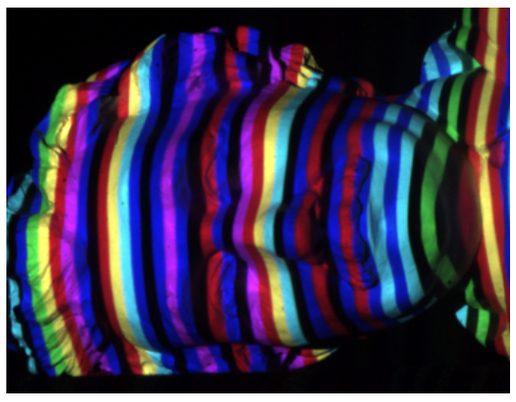


General approach: active illumination

Project an instantaneous pattern for which its easy to find correspondences







Search along epipolar line to find right color

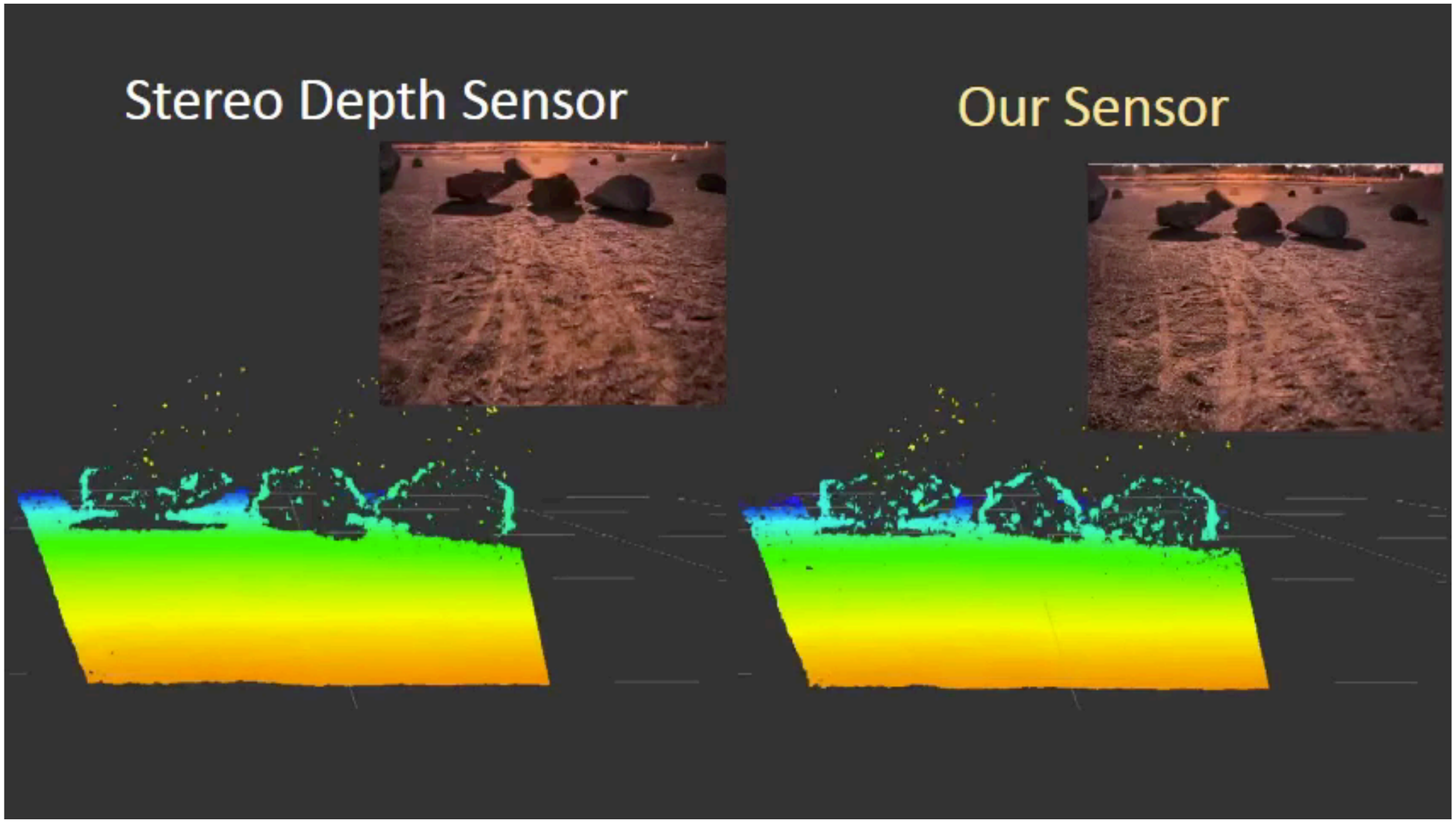
Kinect

Why do we see “holes” around object borders?

Why can’t you use kinect outdoors?

Another approach: Time of Flight

Outdoor depth estimation

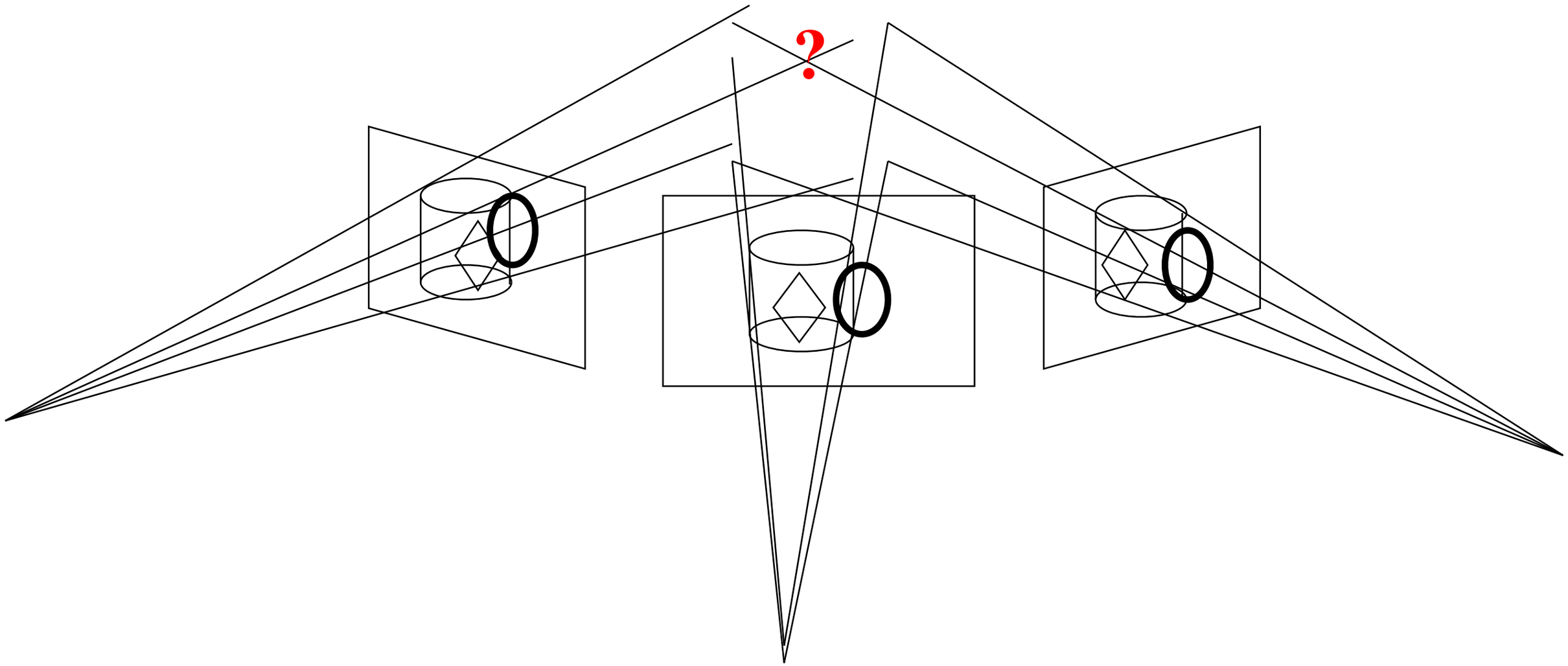


Episcan sensor, CMU

Multiview stereo

- triangulation
- scanline correspondence (dynamic programming)
- active illumination
- **volumetric multiview models / visibility reasoning**
- patch-based methods

Multi-view stereo

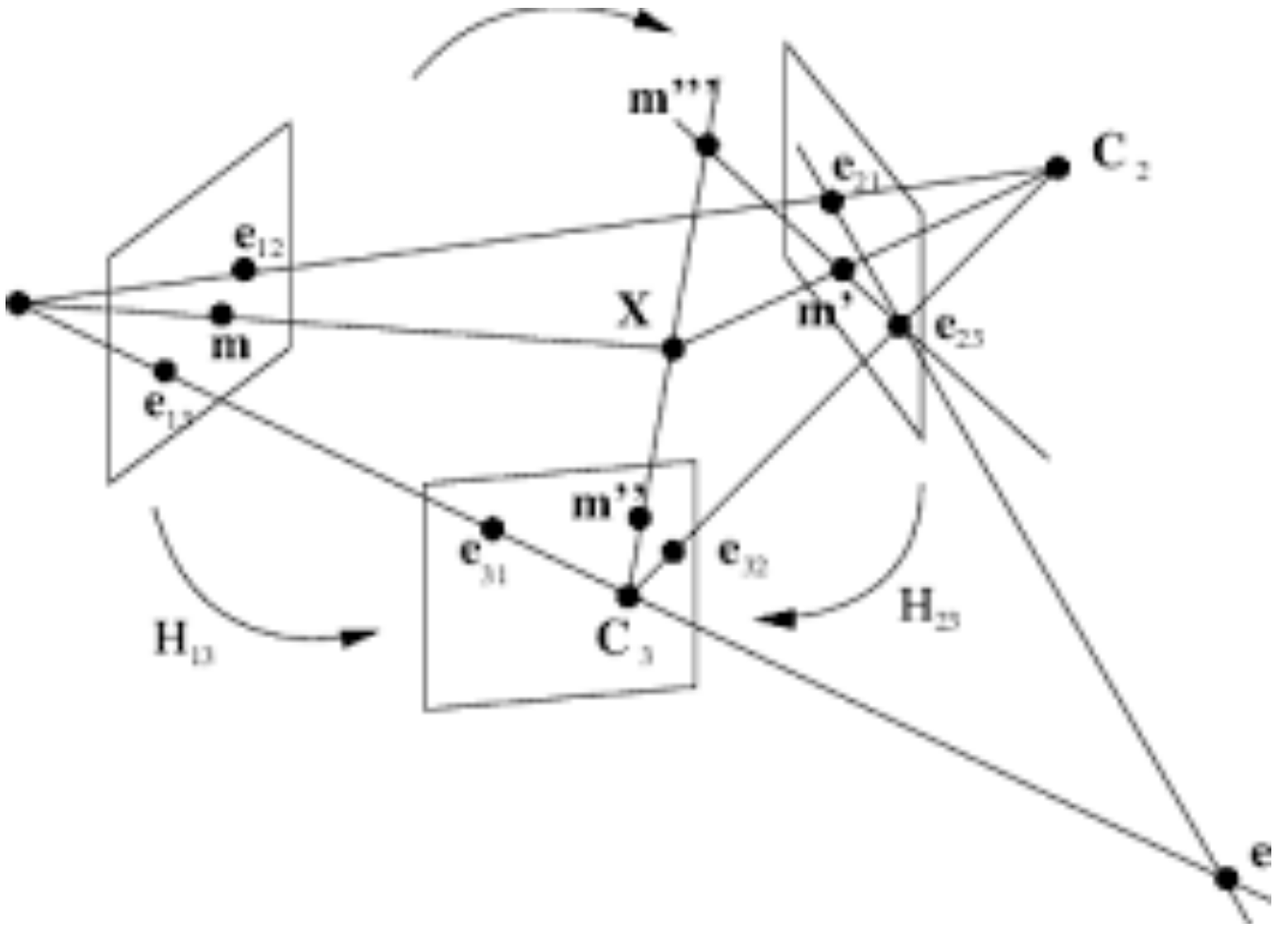


- Reconstruct the 3D position of the points corresponding to (all the) pixels in a set of images.
- Assumption: We know the relative position, orientation, and intrinsic camera matrix *K* of all the cameras
- **Number of cameras > 2**

Trinocular stereo (3 cameras)

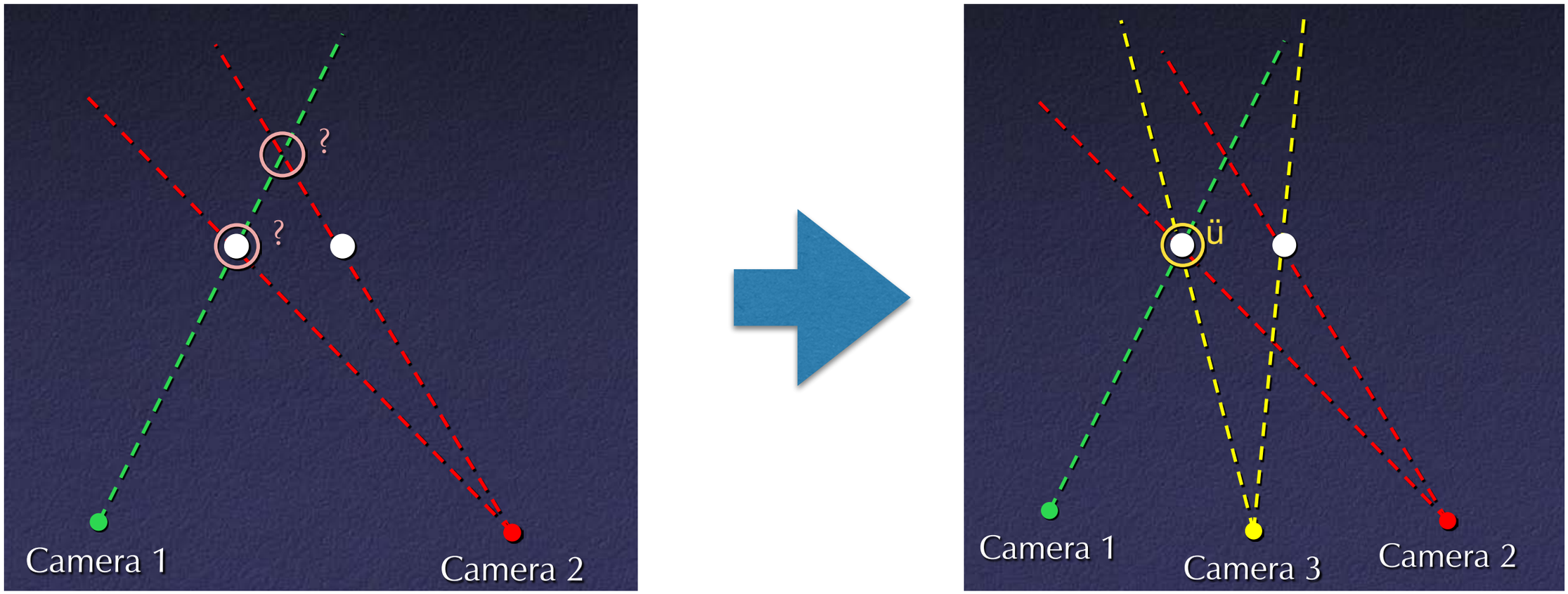
Generalize 3x3 fundamental matrix to a 3x3x3 trifocal tensor

(constrains points and lines across 3 images)



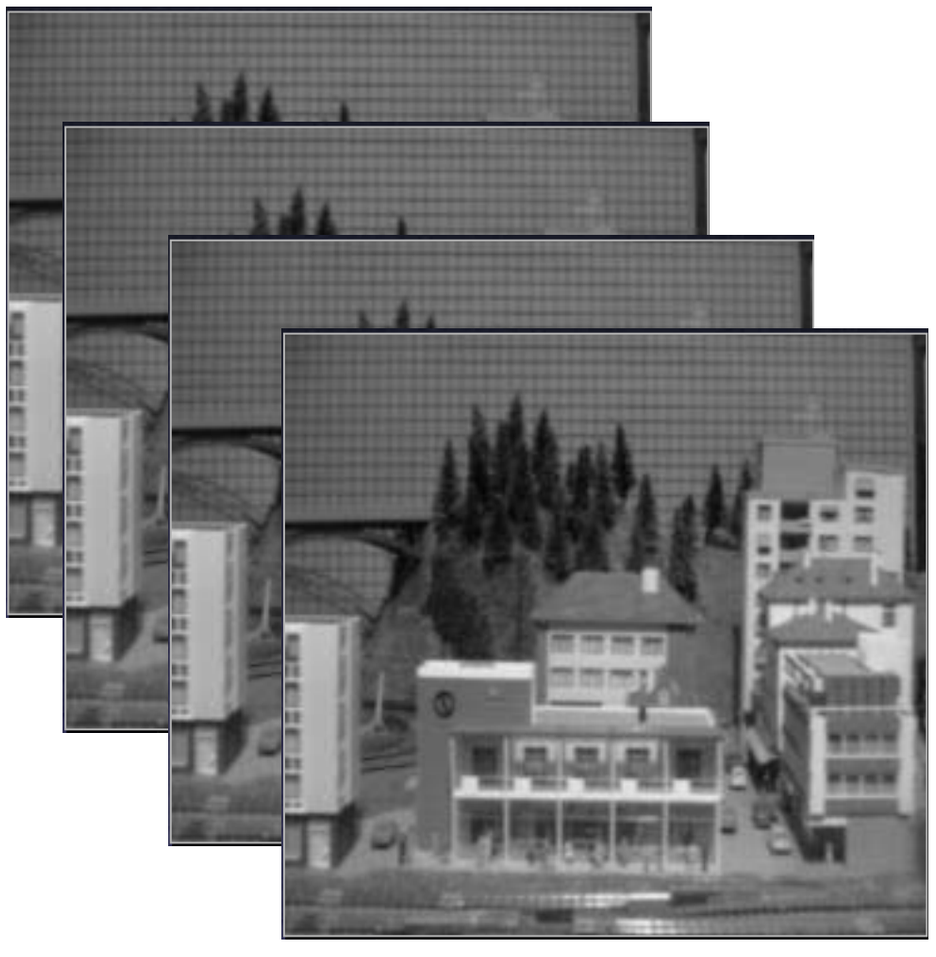
Multi-view stereo (>3 cameras) v0

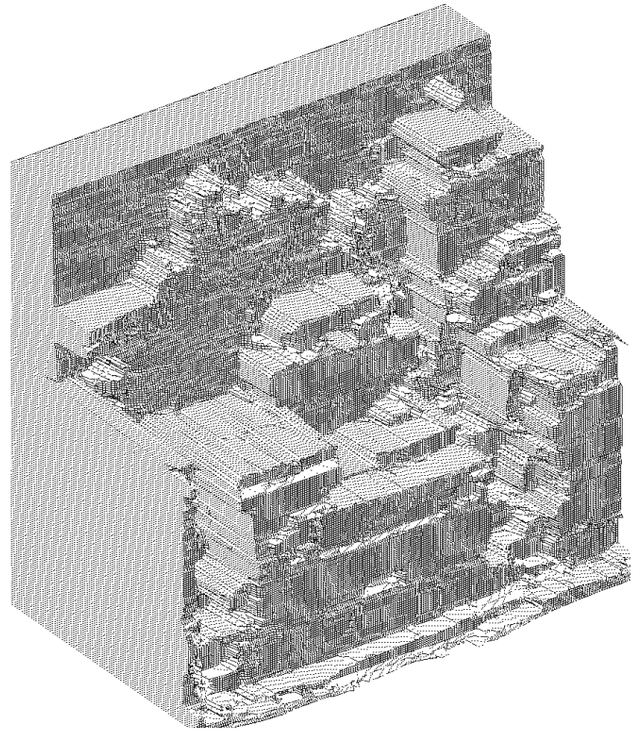
1. Pick 2 views, find correspondences, reconstruct 3D point
2. Project 3D point into other cameras and search for a nearby corresponding point
3. If can't find correspondence near projected location, reject



Multiview stereo (>3 cameras) v1

- Pick one reference view
- Iterate over a discrete set of “candidate depths”
- For each point and for each candidate depth
  - Project point into other views
  - Keep depths that are consistent with all other views





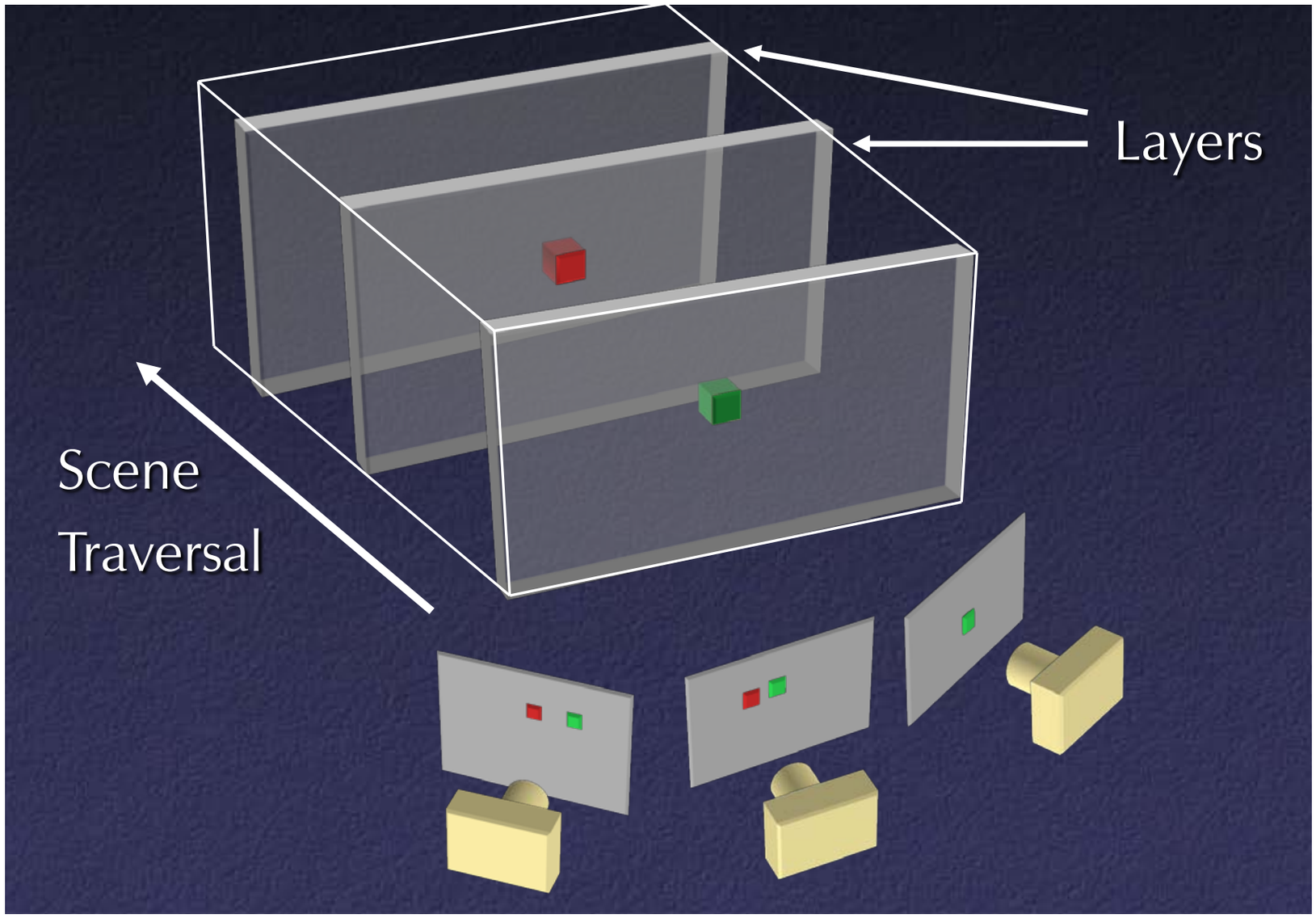
What order to do this?

Key insight: not all points are visible in all other views (e.g. due to occlusions)

Multiview stereo (v1)

Hypothesize depths in a “smart” order where occluding points are found *first*

Use knowledge of occluding points to smartly select view for photo-consistency check



Store photo-consistent color in a 3D voxel grid

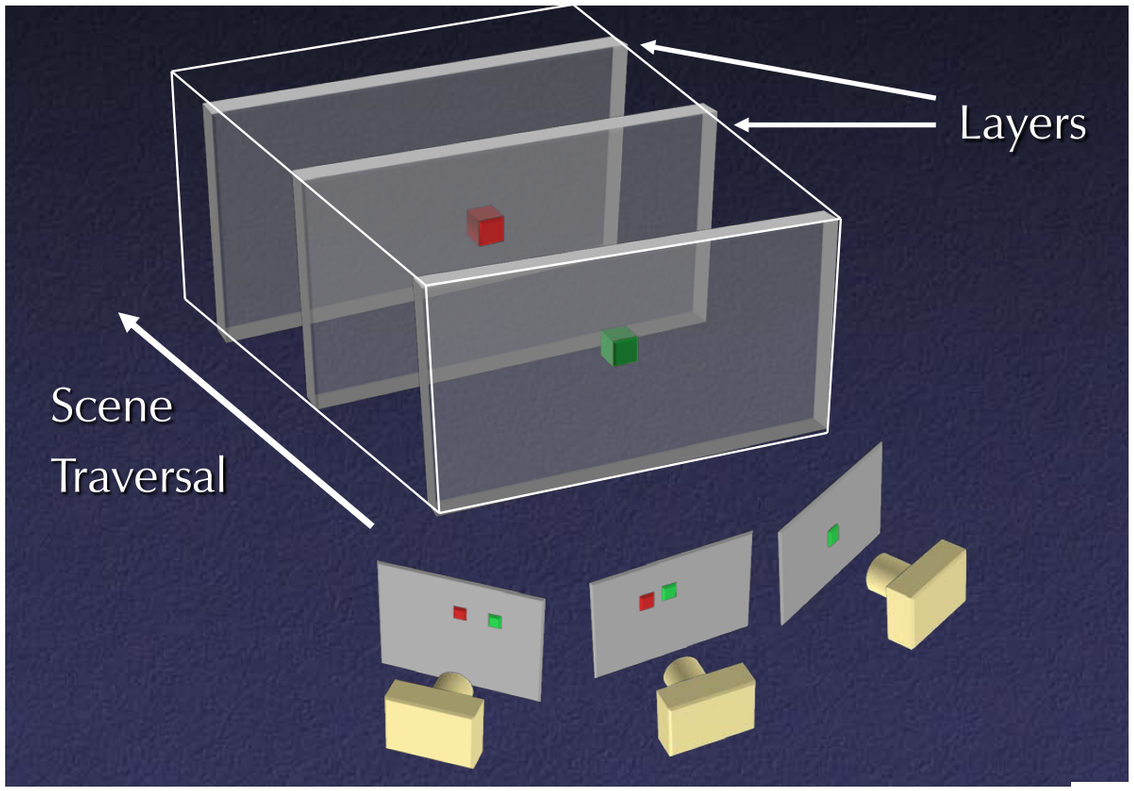
Reconstruct shape *and* appearance

Implementation: Plane-sweep stereo

Sweep over voxel plane-by-plane, starting closest-to-front

Imagine that all points in the world lie in this plane

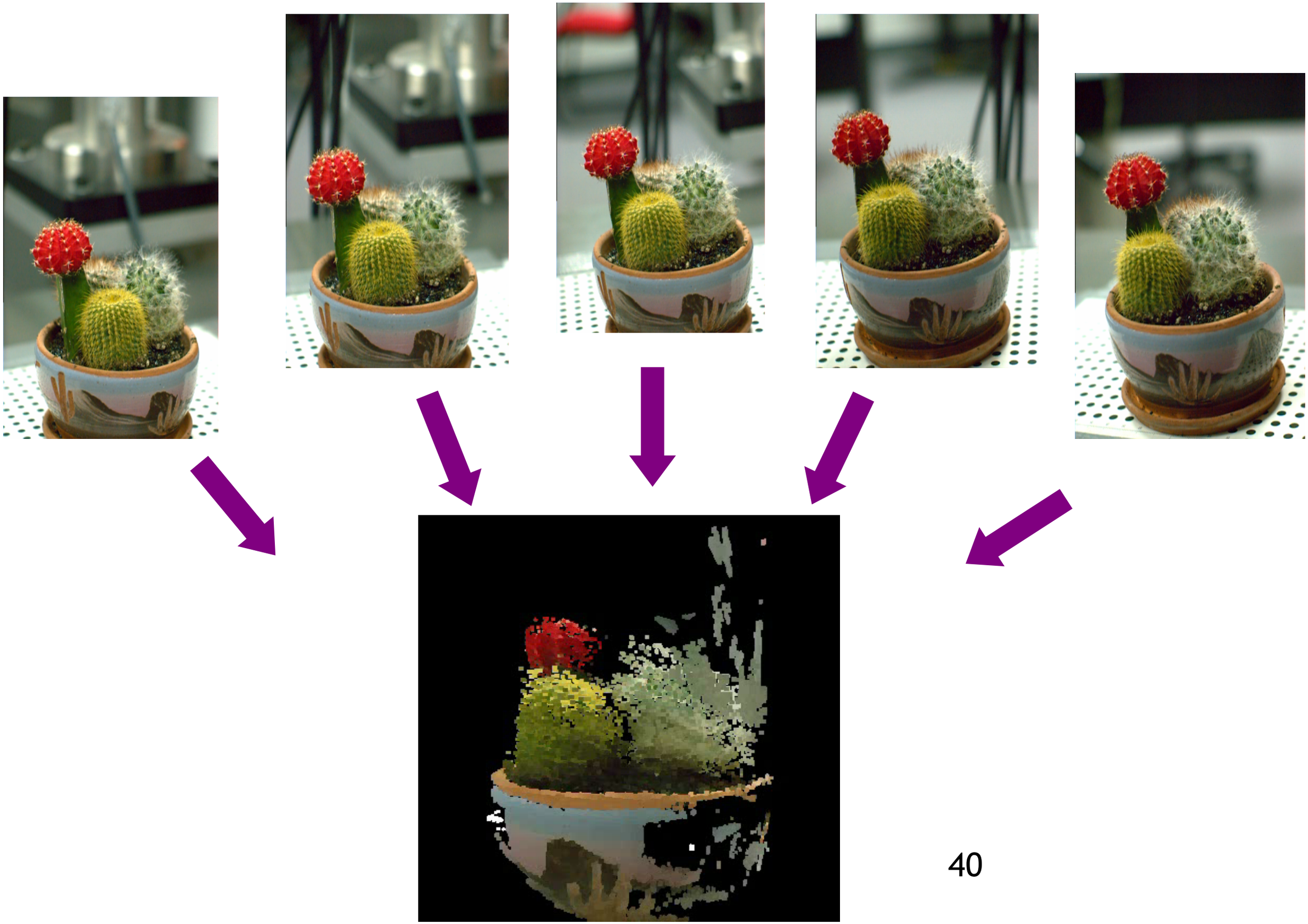
Validate all voxels in a plane by projecting their appearance into a virtual view for each of N cameras



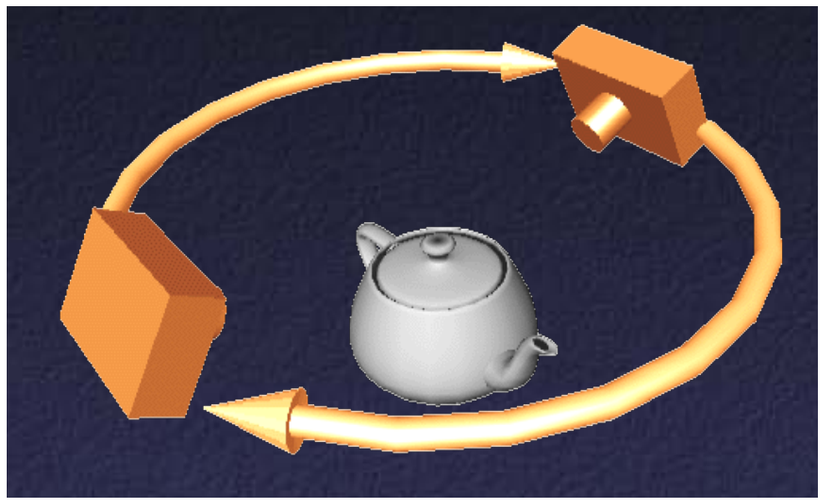
What is the transformation that warps image N to a virtual view?

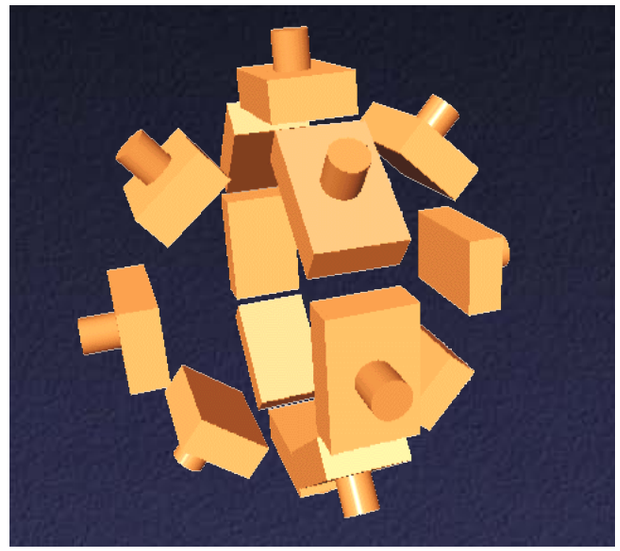
Homography!

Voxel coloring



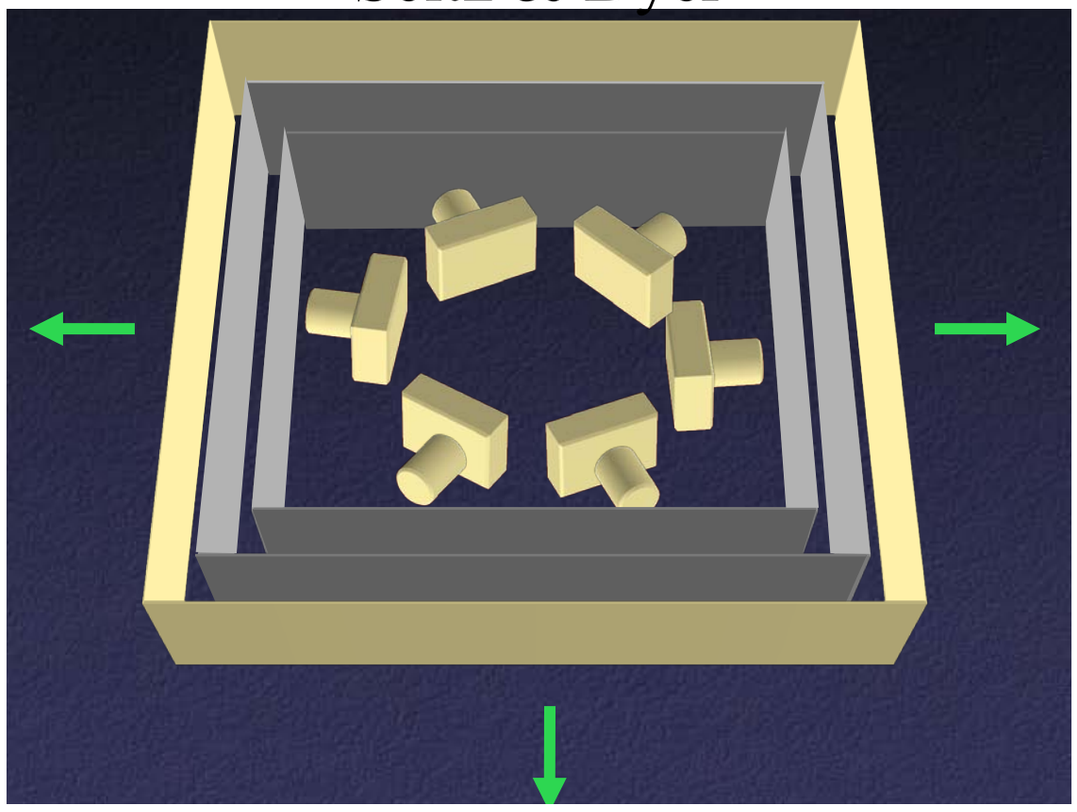
What about other camera setups?





Panoramic depth ordering

Seitz & Dyer

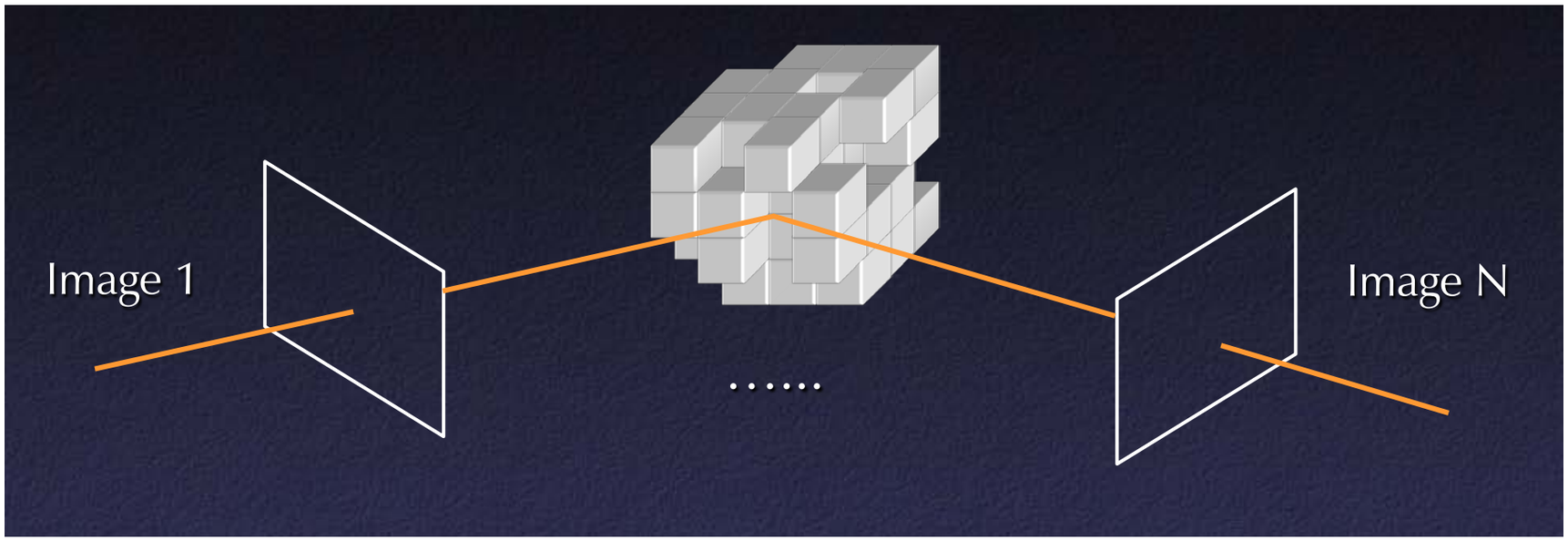


Layers radiate inwardly/outwardly

Space carving

Imagine all cameras facing inwards towards objects on a table

Kutulakos & Seitz



Very simple algorithm:

1. Initialize voxel grid to all ‘1’s
2. Repeatedly choose a voxel on current surface:

Project to visible images; carve out if not photoconsistent

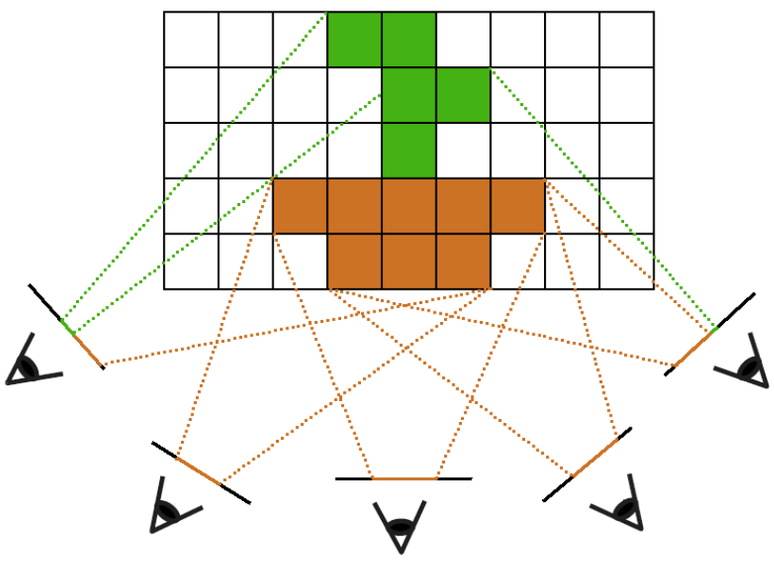
Convergence

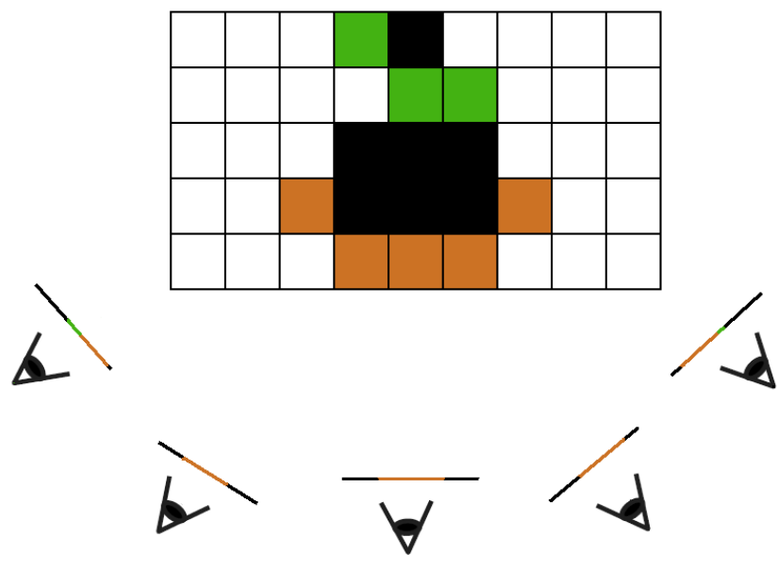
Consistency Property

- The resulting shape is photo-consistent
  - > all inconsistent points are removed

Convergence Property

- Carving converges to a non-empty shape
  - > a point on the true scene is *never* removed





Calibrated Image Acquisition

Calibrated Turntable

Selected Dinosaur Images

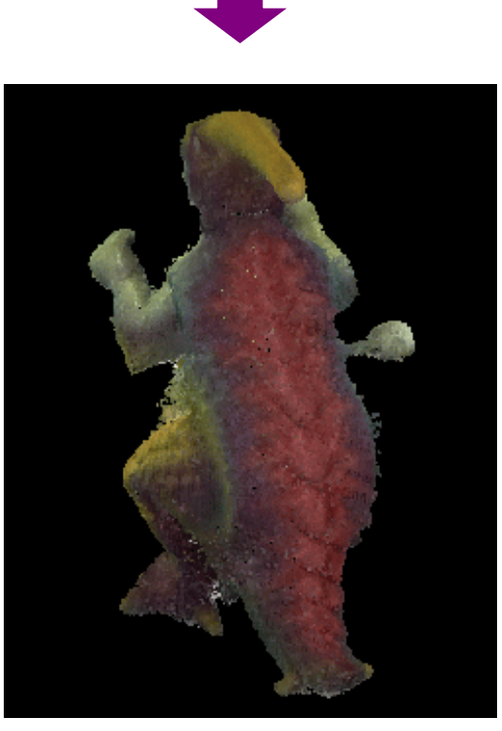
Selected Flower Images

Voxel Coloring Results

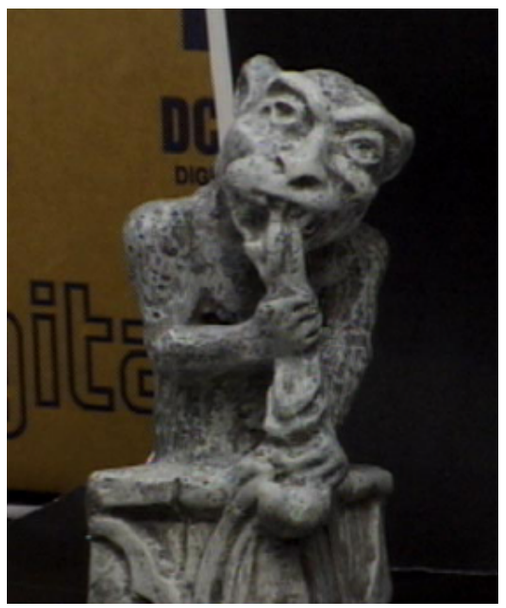
Dinosaur Reconstruction
72 K voxels colored
7.6 M voxels tested
7 min. to compute
on a 250MHz SGI

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

21 images



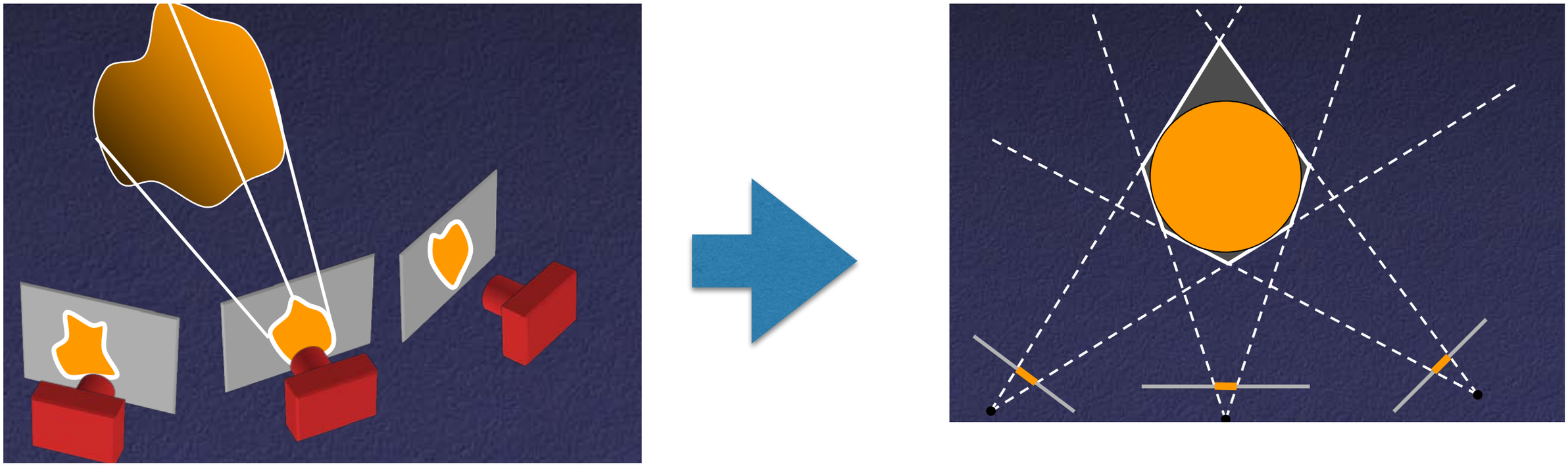
21 images



16 images

99 images

Silhoette carving



Backproject binary silhouettes and find intersection

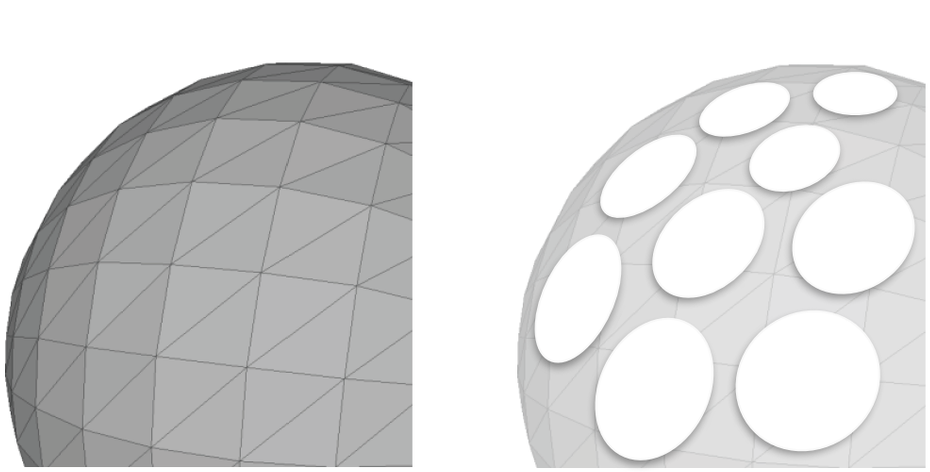
In limit of infinite cameras, this will produce convex hull reconstruction of object

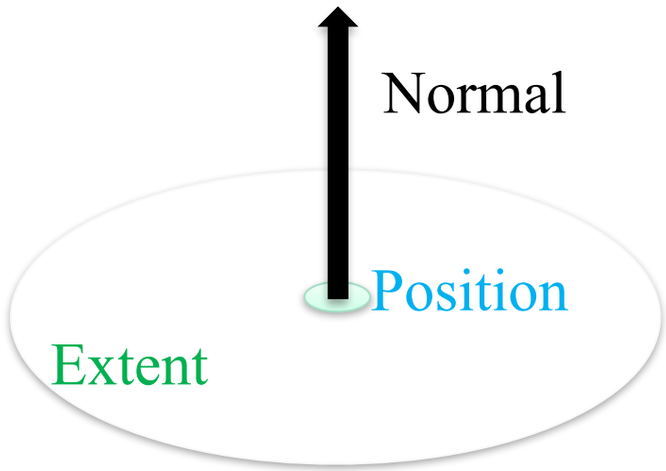
Multiview stereo

- triangulation
- scanline correspondence (dynamic programming)
- active illumination
- volumetric models / visiblity reasoning
- **patch-based methods**

Accurate, Dense, and Robust Multi-View Stereopsis

Yasutaka Furukawa and Jean Ponce, *Fellow, IEEE*





Easier to approximate surface by dense set of local planar patches

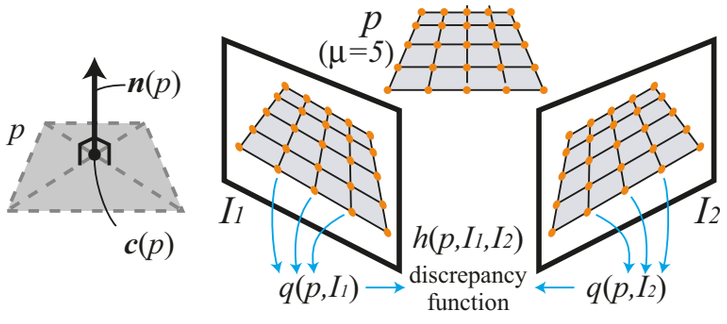
Patch-based Multiview Stereo (PMVS)

Pipeline: feature detection

Find sparse matches over pairs of images (using interest points + matching)
Triangulate to find sparse 3D points {p}

Pipeline: patch optimization

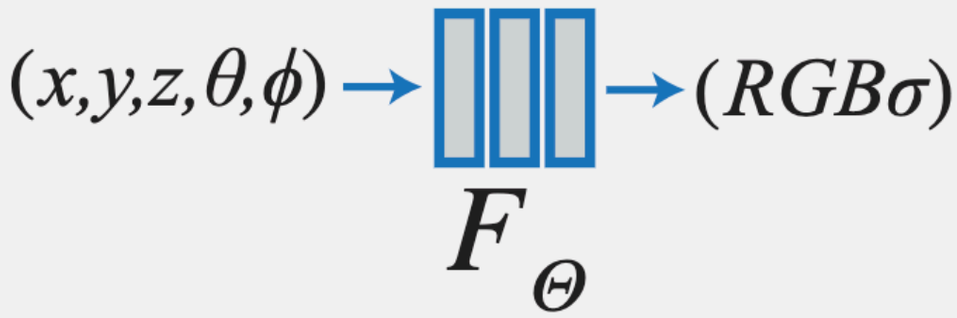
At each point p, estimate normal N(p) and visibility $V_i(p)$ in each image using photoconsistency check (NCC over ~9x9 pixels)



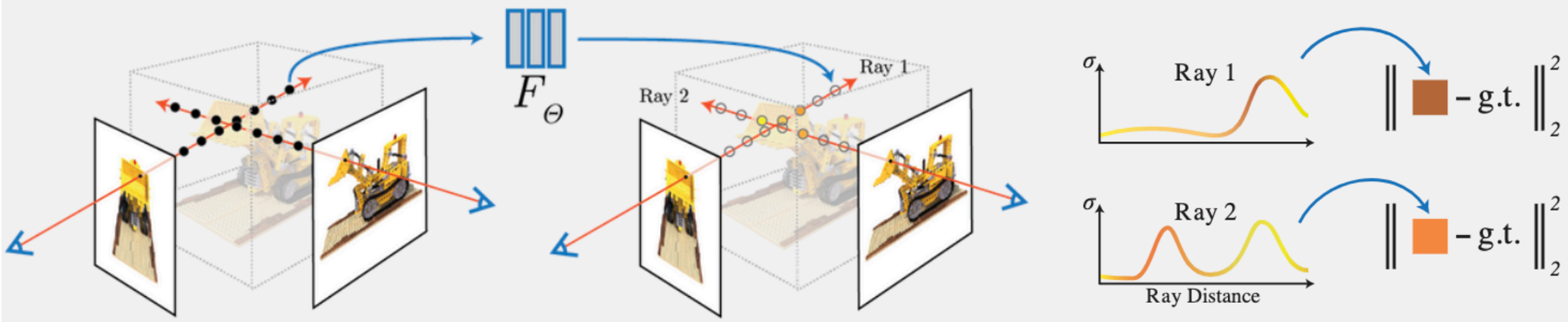
Results

Recent work: neural radiance fields (NERFs, ECCV 2020)

We present a method that achieves state-of-the-art results for synthesizing novel views of complex scenes by optimizing an underlying continuous volumetric scene function using a sparse set of input views.



Our algorithm represents a scene using a fully-connected (non-convolutional) deep network, whose input is a single continuous 5D coordinate (spatial location (x, y, z) and viewing direction (θ, φ)) and whose output is the volume density and view-dependent emitted radiance at that spatial location.

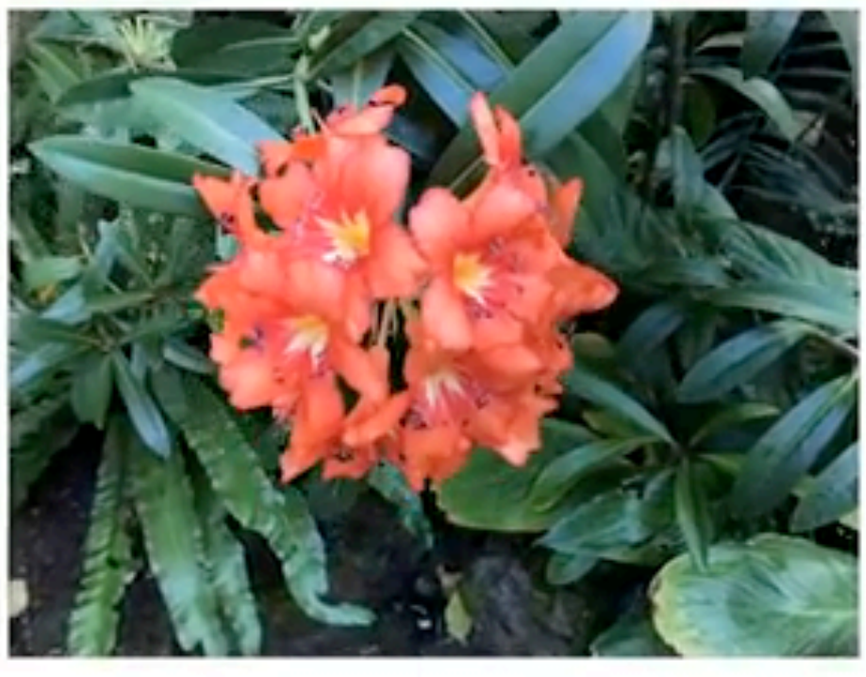


We synthesize views by querying 5D coordinates along camera rays and use classic volume rendering techniques to project the output colors and densities into an image. Because volume rendering is naturally differentiable, the only input required to optimize our representation is a set of images with known camera poses. We describe how to effectively optimize neural radiance fields to render photorealistic novel views of scenes with complicated geometry and appearance, and demonstrate results that outperform prior work on neural rendering and view synthesis.

NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis

Ben Mildenhall* UC Berkeley
Pratul P. Srinivasan* UC Berkeley
Matthew Tancik* UC Berkeley
Jonathan T. Barron Google Research
Ravi Ramamoorthi UC San Diego
Ren Ng UC Berkeley

* Denotes Equal Contribution



Structure from motion

Recover both 3D scene geometry and camera positions

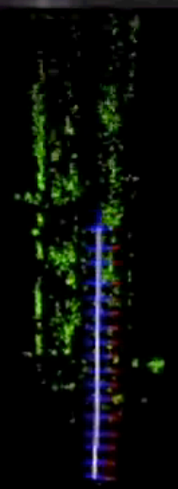
SfM vs Previous Approaches

- Previous: Estimate camera structure first, then estimate scene geometry
- New: Jointly estimate camera structure and scene geometry
  - In robotics: SLAM (very fast / real-time)
  - In computer vision: Structure from Motion (very slow but higher focus on accuracy)

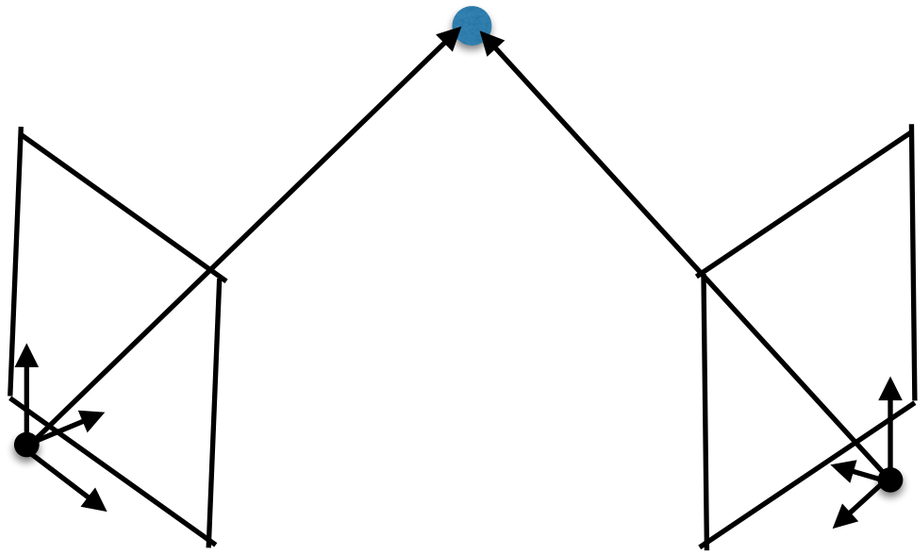
SLAM: Simultaneous Localization and Mapping

(iteratively build map of 3D points and camera poses as new images arrive)

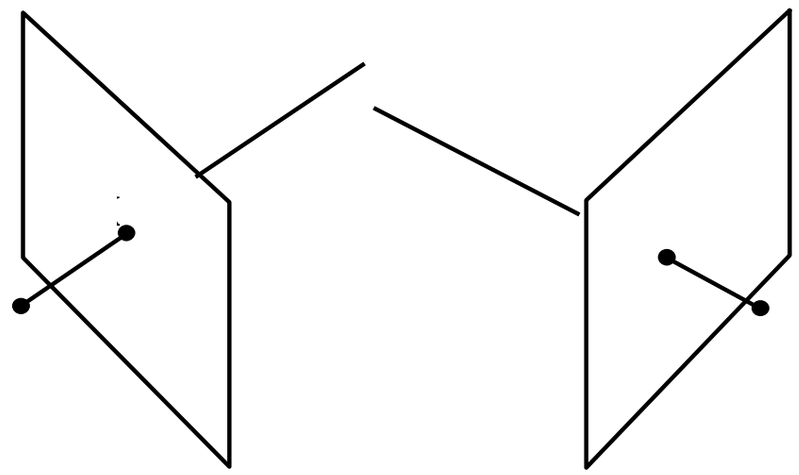
Sequence 00



Recall: 2-view stereo

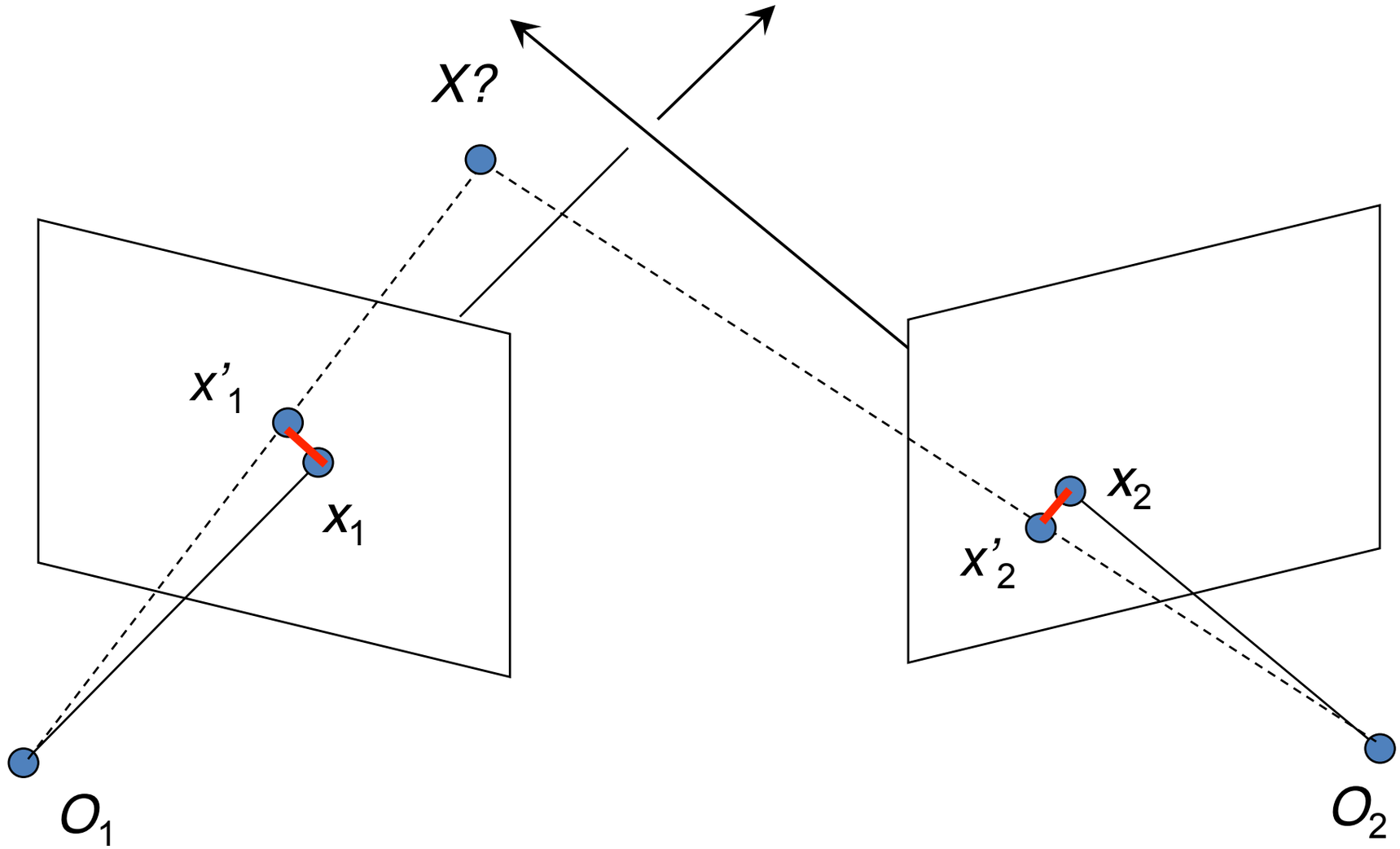


An annoying detail



What to do when ray’s don’t intersect?

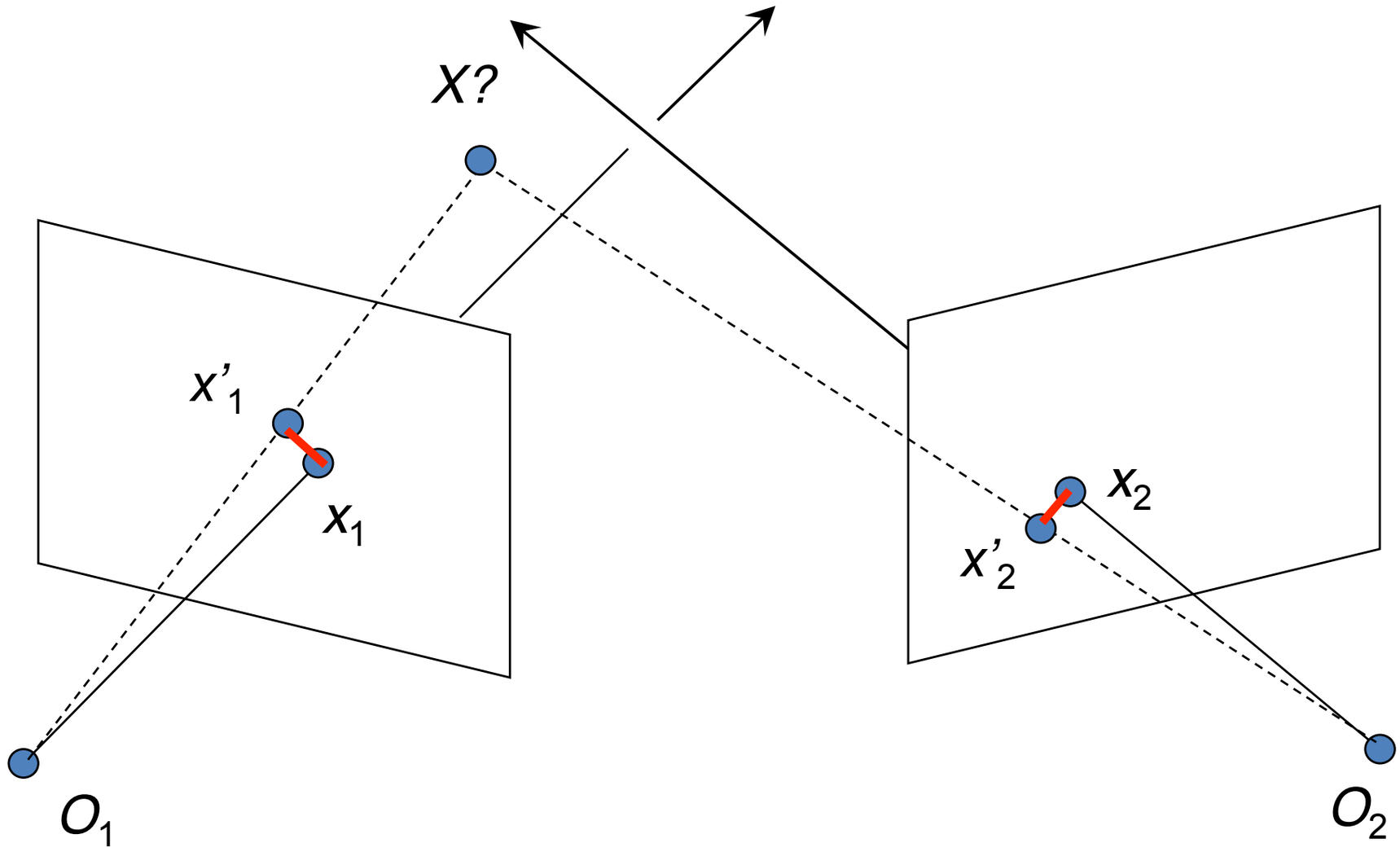
Minimize reprojection error



$$\mathbf{x}' = Proj(\mathbf{X}, M)$$

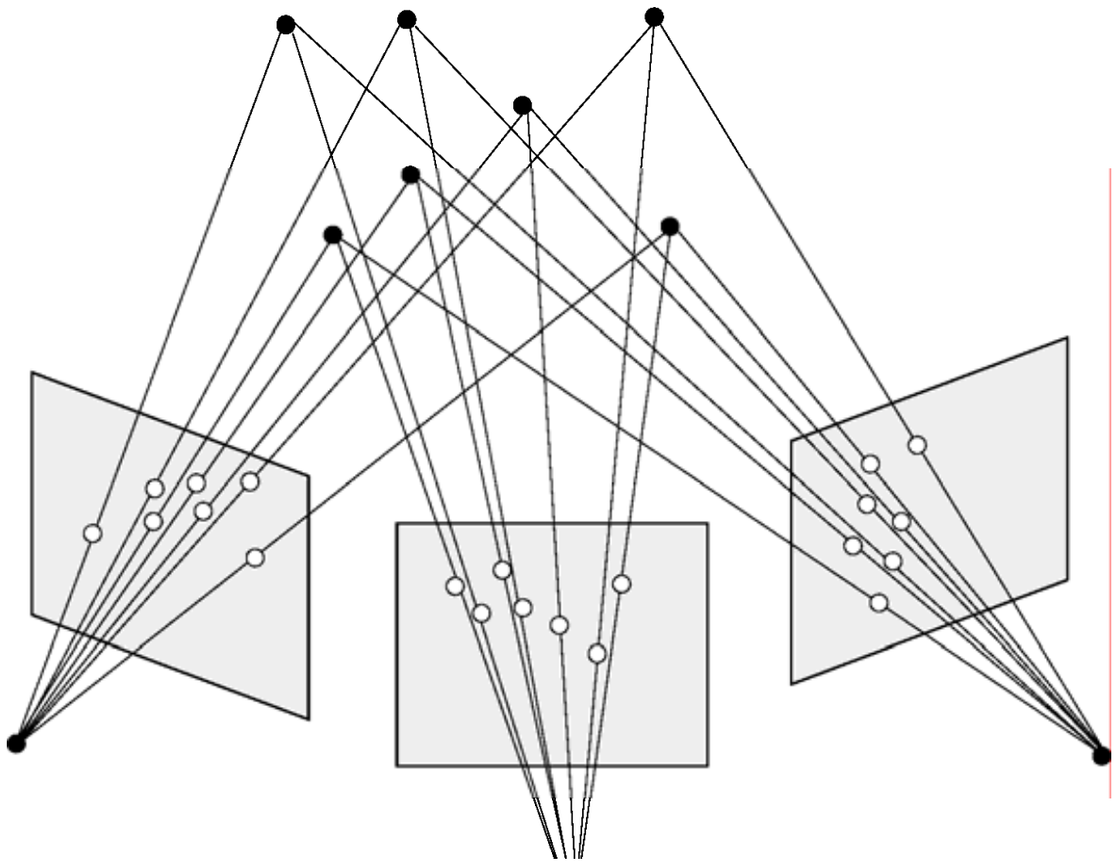
Perspective projection equations where M = 3x4 matrix of extrinsics (R,T) and intrinsics (f) of camera

Minimize reprojection error



$$\min_{\mathbf{X}} f(\mathbf{X}) = ||\mathbf{x}_1 - Proj(\mathbf{X}, M_1)||^2 + ||\mathbf{x}_2 - Proj(\mathbf{X}, M_2)||^2$$

Generalize triangulation to multiple points from multiple cameras



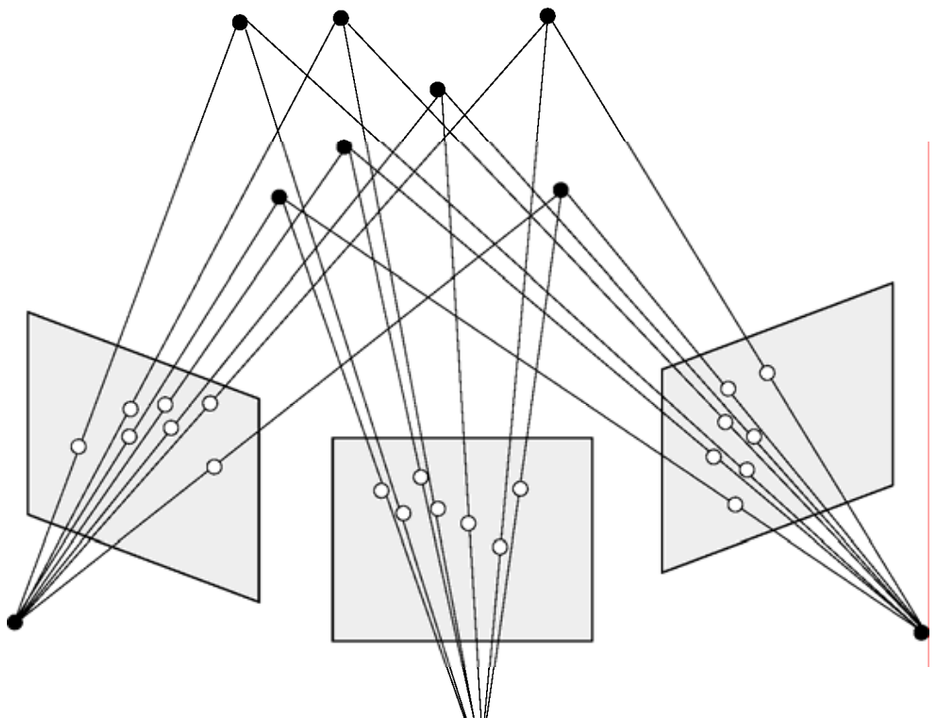
$$\min_{\mathbf{X}_1,\mathbf{X}_2,\ldots} \sum_{i=1}^{m}\sum_{j=1}^{n} ||\mathbf{x}_{ij} - Proj(\mathbf{X}_j, M_i)||^2$$

cameras points

As written, could this minimization be done independently for each 3D point?

Bundle adjustment

Minimize reprojection error over multiple 3D points **and cameras**



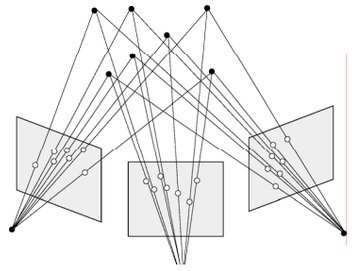
$$\min_{\mathbf{X}_1,\mathbf{X}_2,\ldots,M_1,M_2,\ldots} \sum_{i=1}^{m}\sum_{j=1}^{n} ||\mathbf{x}_{ij} - Proj(\mathbf{X}_j, M_i)||^2$$

(not all points will be visible in all views)

$$\min_{\mathbf{X}_1,\mathbf{X}_2,\ldots,M_1,M_2,\ldots} \sum_{i=1}^{m}\sum_{j=1}^{n} \text{vis}_{ij} ||\mathbf{x}_{ij} - Proj(\mathbf{X}_j, M_i)||^2$$

0 or 1

Basic SFM pipeline



1. Find candidate correspondences (interest points + descriptor matches)

2. Select a subset of correspondences that are consistent with epipolar constraints on pairs of images (RANSAC + fundamental matrix)

3. Solve for 3D points and camera that minimize reprojection error

$$\min_{\mathbf{X}_1,\mathbf{X}_2,\ldots,M_1,M_2,\ldots} \sum_{i=1}^{m}\sum_{j=1}^{n} ||\mathbf{x}_{ij} - Proj(\mathbf{X}_j, M_i)||^2$$

Is this a linear or nonlinear optimization?

$$\mathbf{x}' = Proj(\mathbf{X}, M) = \begin{bmatrix} \frac{\mathbf{m_1^T X}}{\mathbf{m_3^T X}} \\ \frac{\mathbf{m_2^T X}}{\mathbf{m_3^T X}} \end{bmatrix}$$

Non-linear least squares

Where did we previously see non-linear least squares?

How did we solve it?

Lukas Kanade -> Gauss-Newton Optimization

1. **Start with an initial guess**
2. **Linearize about the guess (with Taylor series)**
3. **Solve for a delta X and delta M**
4. **Go back to step 2**

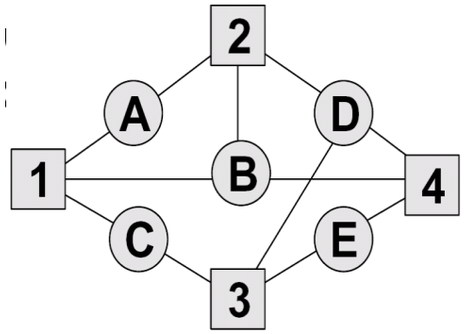
We can also use other nonlinear least-squares optimizers such as **Levenberg Marquardt**
(more often used in practice)

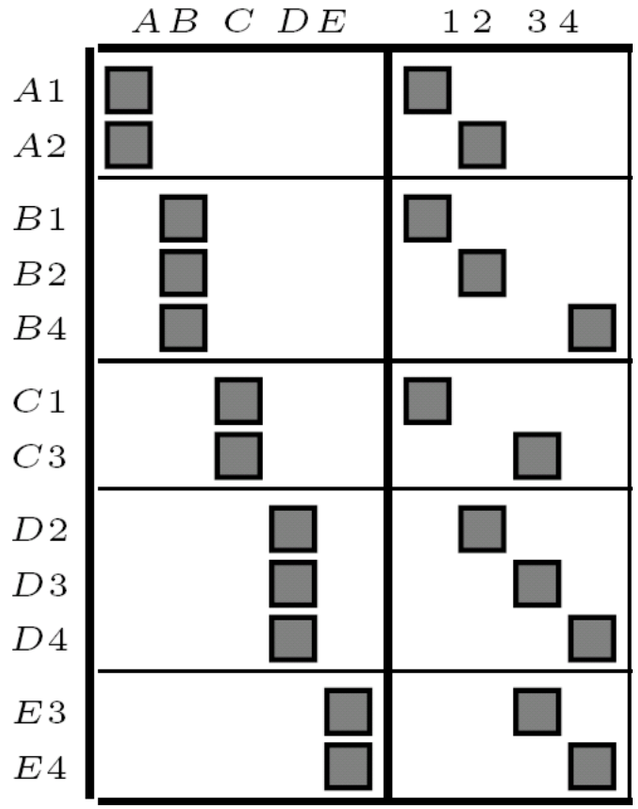
Bundle adjustment: nonlinear least-squares

$$\min_{\mathbf{x}_1,\mathbf{x}_2,\ldots,M_1,M_2,\ldots} \sum_{i=1}^{m}\sum_{j=1}^{n} \text{vis}_{ij} ||\mathbf{x}_{ij} - Proj(\mathbf{X}_j, M_i)||^2$$

Encode visibility graphs of what points are seen in what images, i.e.

- Points: A,B,C,D,E
- Images: 1,2,3





Visibility graph implies Jacobian is sparse

We can take advantage of the sparsity to reduce the speed and memory of the method

City-scale SFM

"Building Rome in a Day" CACM 2011

Important to exploit visibility graph

Summary: 3D geometric vision

- Single-view geometry
  - The pinhole camera model
    - Variation: orthographic projection
  - The perspective projection matrix
  - Intrinsic parameters
  - Extrinsic parameters
  - Calibration
- Multiple-view geometry
  - Triangulation
  - The epipolar constraint
    - Essential matrix and fundamental matrix
  - Stereo
    - Binocular, multi-view
  - Structure from motion