Multiview stereo

Multi-view geometry

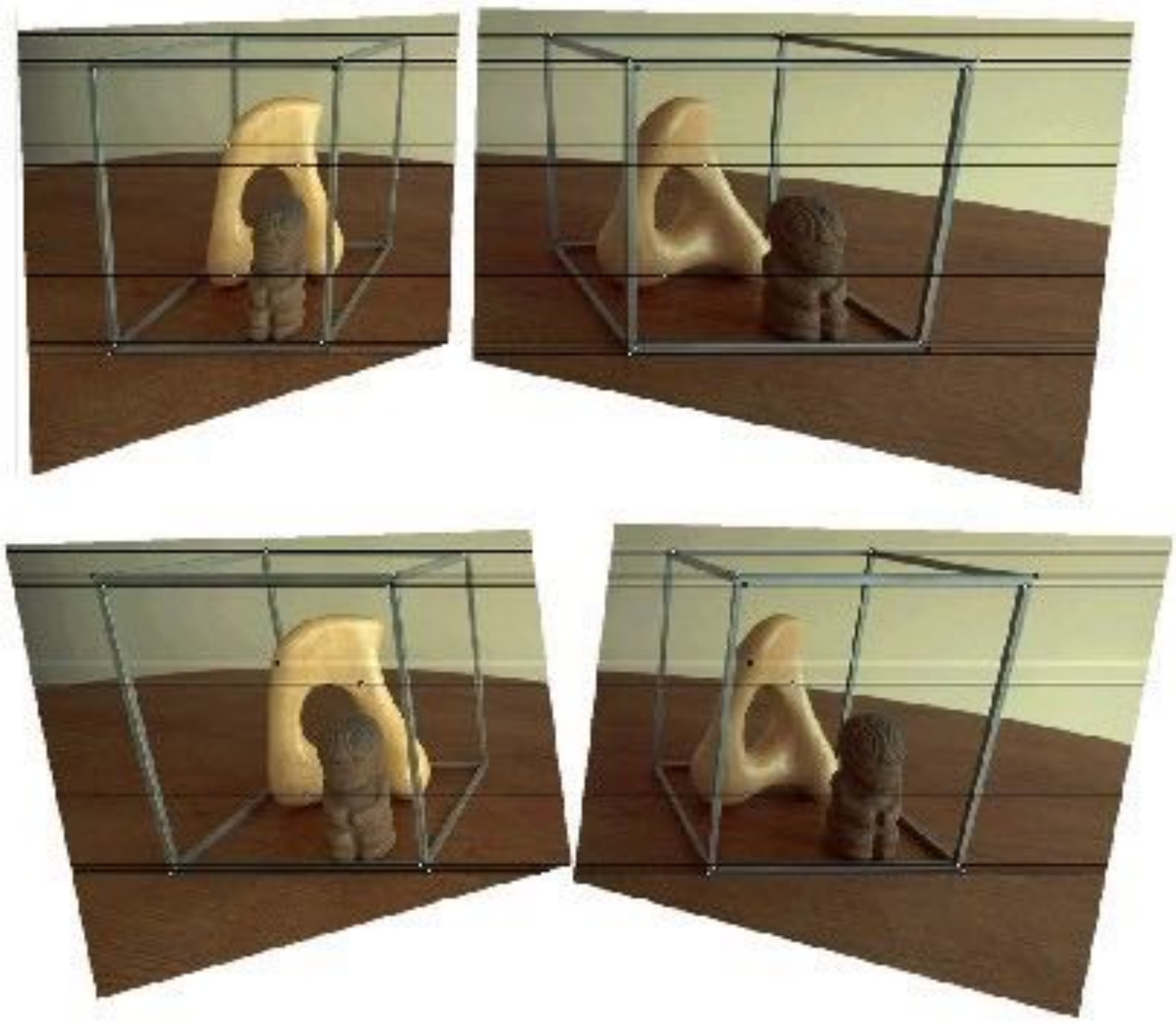
Three questions:

(i) **Correspondence geometry:** Given an image point x in the first view, how does this constrain the position of the corresponding point x' in the second image?

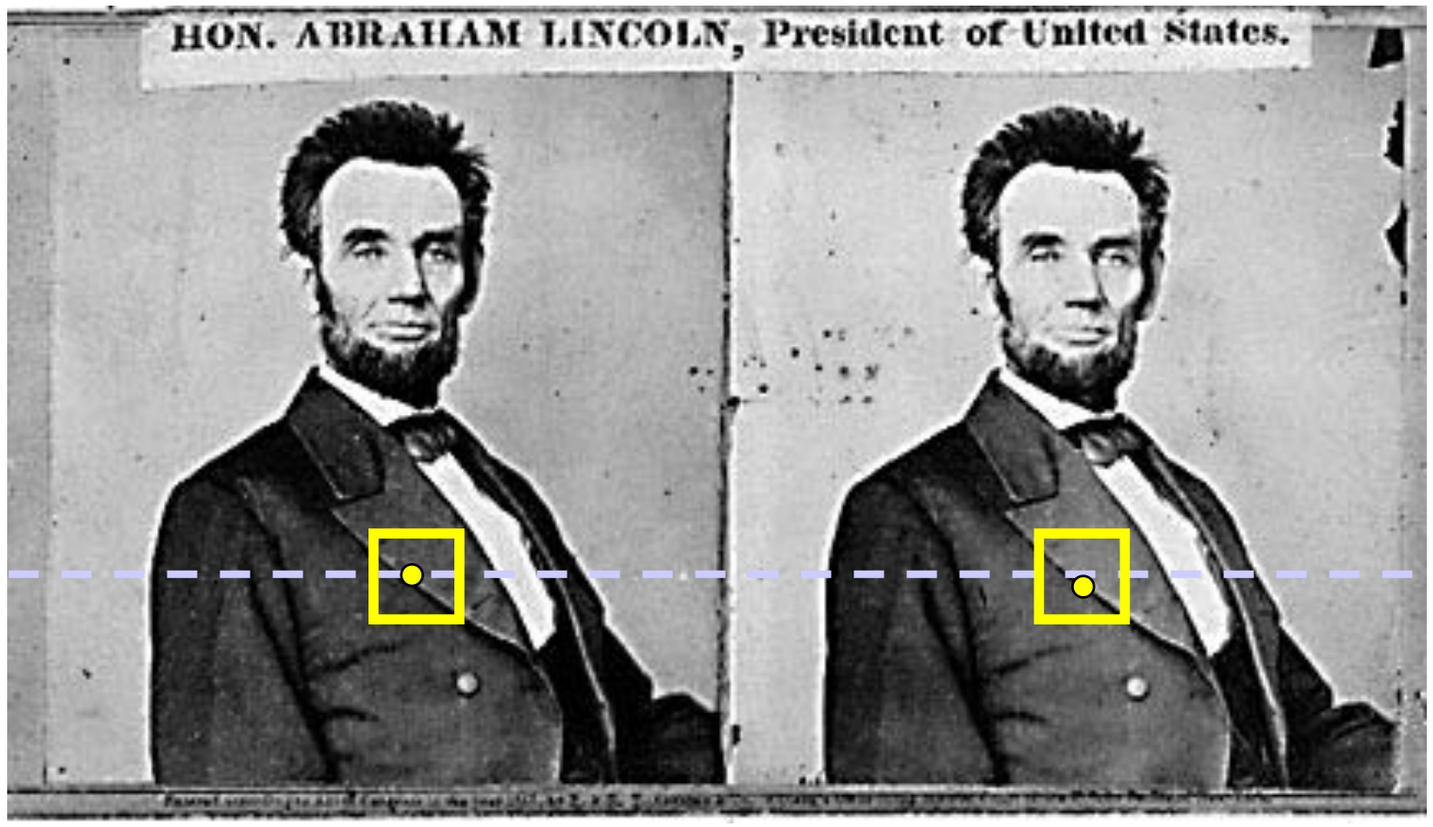
(ii) **Camera geometry (motion):** Given a set of corresponding image points $\{x_i \leftrightarrow x'_i\}$, i=1,…,n, what are the cameras P and P' for the two views?

(iii) **Scene geometry (structure):** Given corresponding image points $x_i \leftrightarrow x'_i$ and cameras P, P', what is the position of (their pre-image) X in space?

Stereo



Basic Stereo Algorithm



For each epipolar line

For each pixel in the left image

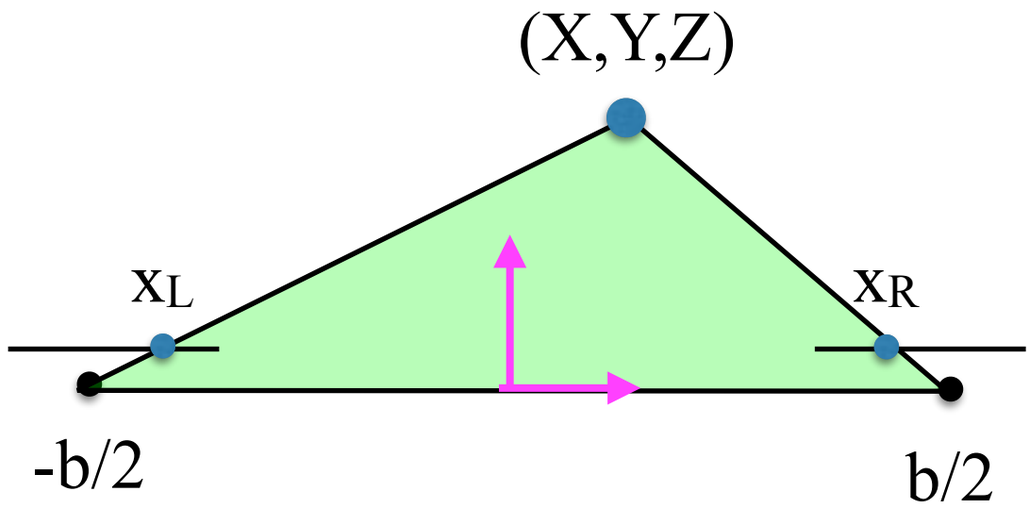
- compare with every pixel on same epipolar line in right image
- pick pixel with minimum match cost

Improvement: match ***windows***

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

Triangulation for Rectified Stereo Pairs

Top-down view where world coordinates are centered between cameras



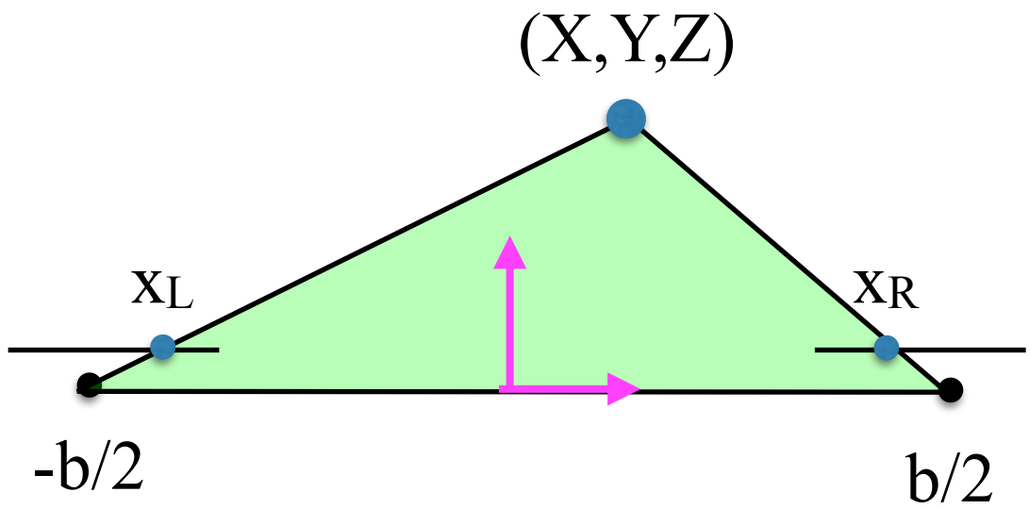
$$(X_L, Y, Z) \rightarrow (x_L, y_L) \qquad (X_R, Y, Z) \rightarrow (x_R, y_R)$$

$$X_L = X + \frac{b}{2} \qquad X_R = X - \frac{b}{2}$$

1. Write down perspective projection equations
2. Solve for (X,Y,Z) in terms of image coordinates, focal length, baseline

Triangulation for Rectified Stereo Pairs

Top-down view where world coordinates are centered between cameras

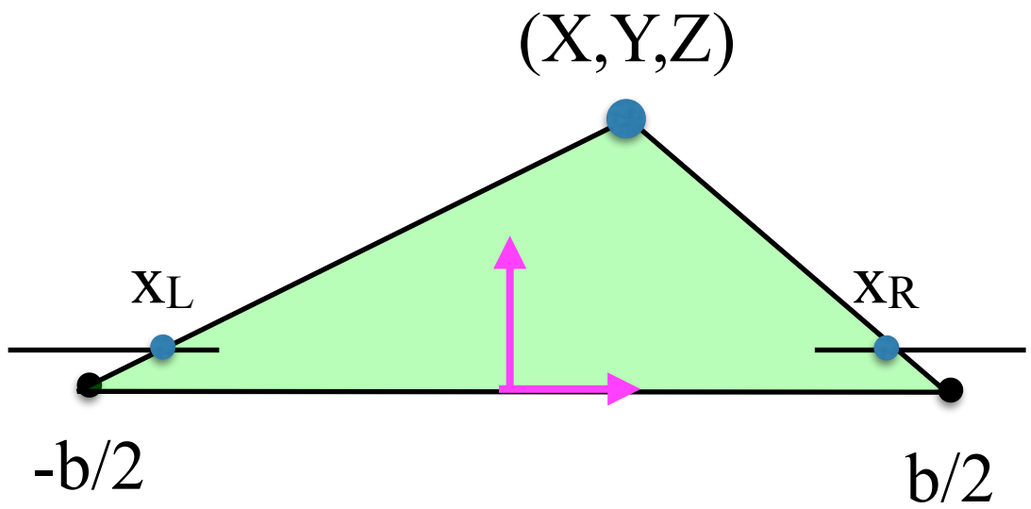


$$\frac{x_L}{f} = \frac{X + b/2}{Z} \qquad \frac{x_R}{f} = \frac{X - b/2}{Z} \qquad \frac{y_L}{f} = \frac{y_R}{f} = \frac{Y}{Z}$$

disparity $= x_L - x_R =$

Triangulation for Rectified Stereo Pairs

Top-down view where world coordinates are centered between cameras



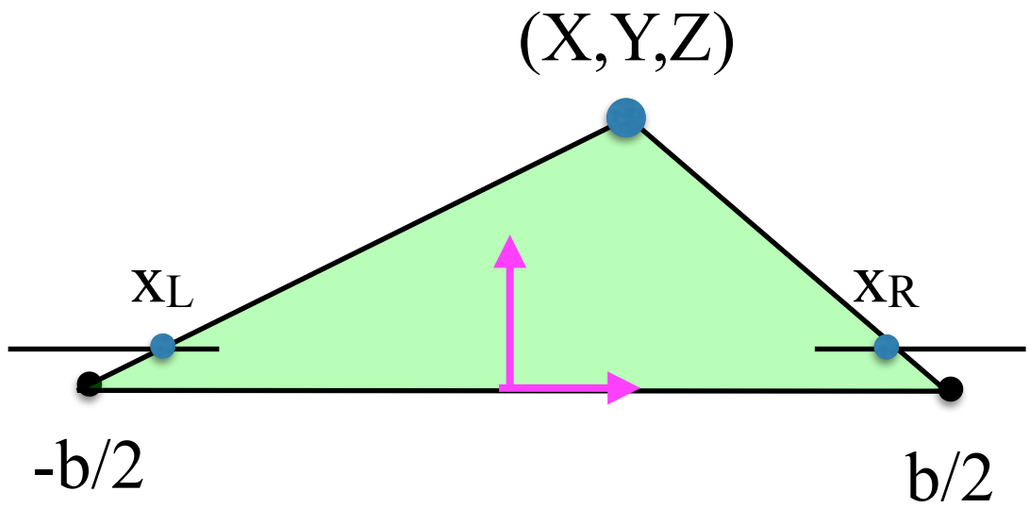
$$\frac{x_L}{f} = \frac{X + b/2}{Z} \qquad \frac{x_R}{f} = \frac{X - b/2}{Z} \qquad \frac{y_L}{f} = \frac{y_R}{f} = \frac{Y}{Z}$$

$$\text{disparity} = x_L - x_R = (X + \frac{b}{2} - X + \frac{b}{2})\frac{f}{Z}$$

$$= \frac{bf}{Z}$$

Triangulation for Rectified Stereo Pairs

Top-down view where world coordinates are centered between cameras



$$\frac{x_L}{f} = \frac{X + b/2}{Z} \qquad \frac{x_R}{f} = \frac{X - b/2}{Z} \qquad \frac{y_L}{f} = \frac{y_R}{f} = \frac{Y}{Z}$$

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

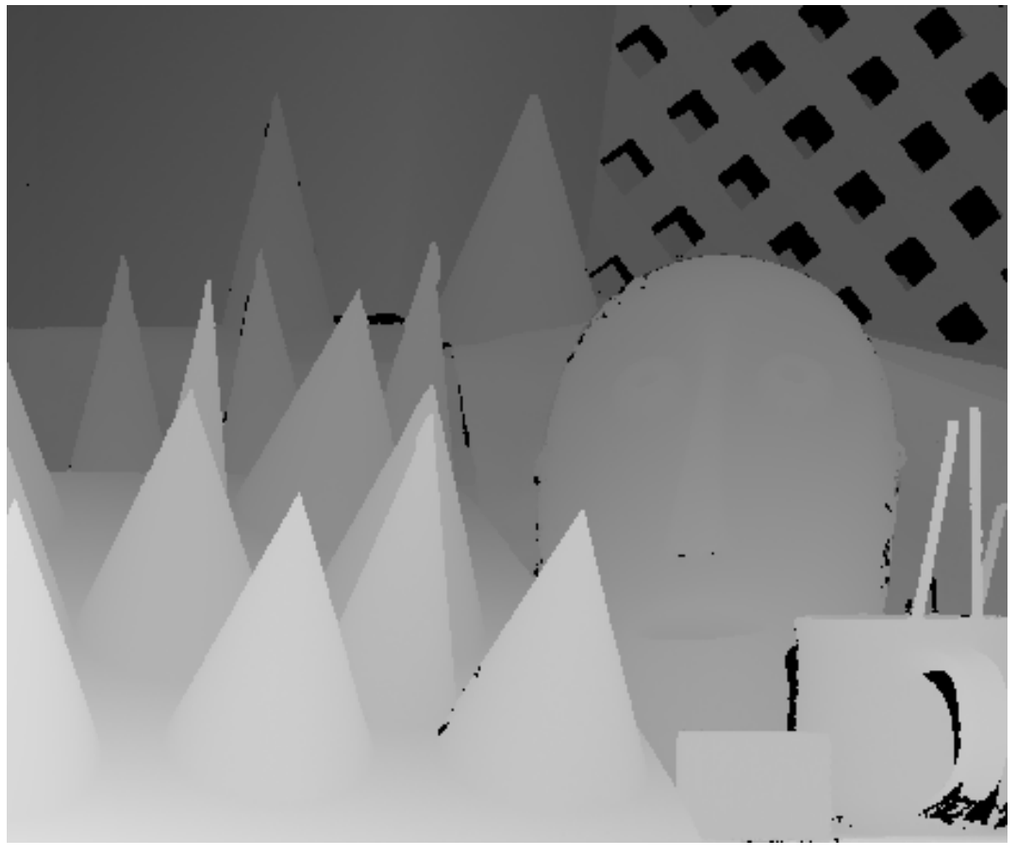
$d = x_L - x_R = \frac{bf}{Z}$ is the **disparity** between corresponding left and right image points

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

Disparity Maps

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

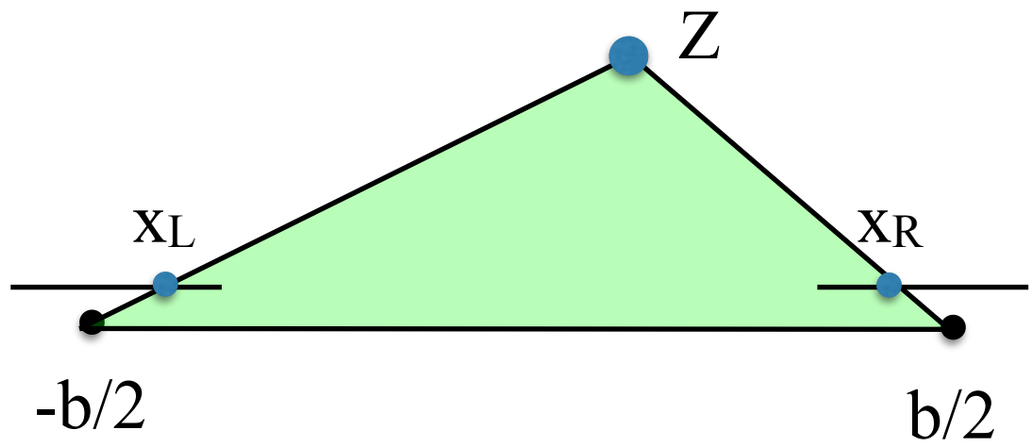
Disparity values (0-64)



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

If we double the size of scene geometry and baseline, what happens to disparity?

Numerical stability



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

Scene + camera variables: Z,f,b

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

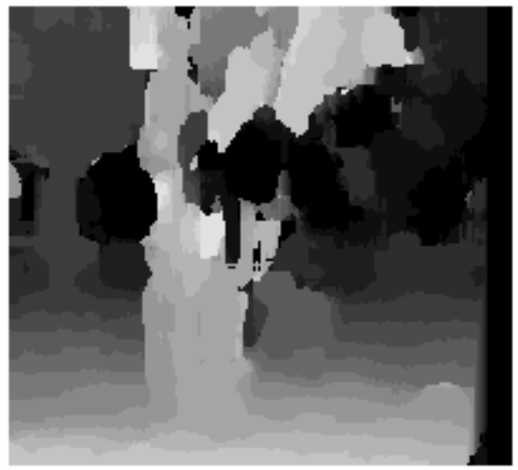
How do we characterize the error in depth Z given an error in disparity d, in terms of scene + camera?

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

1. Error inversely proportional to baseline (larger baselines increase numerical stability)
2. Error increases quadratically with depth (hard to reconstruct far away points

Disparity maps (in practice)

Small matching window
(better localization)



Large matching window
(better detection)

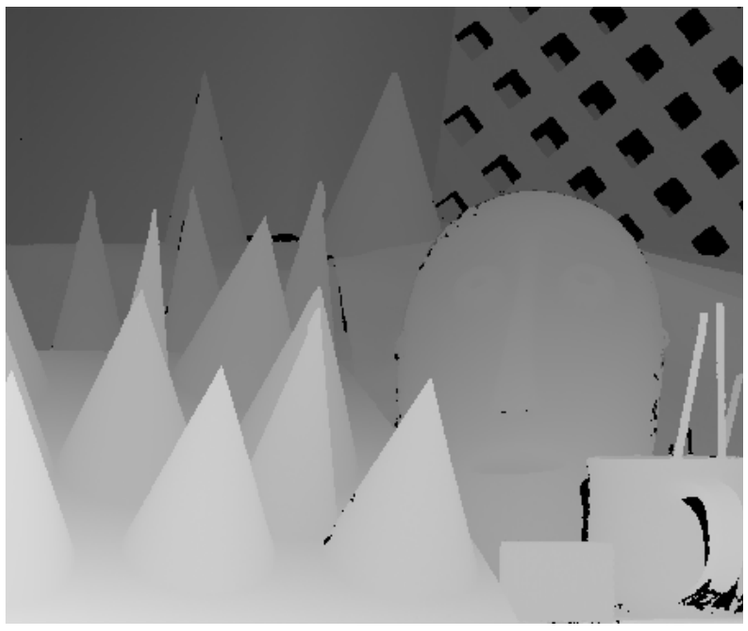
Variational stereo

Penalize differences in nearby disparities (a “1-d” flow problem!)

$$\min_{u,v} E_{intensity} + E_{smooth}$$

$$E_{intensity}(d) = \int\int (I_2(x + d(x,y), y) - I_1(x,y))^2 dxdy$$

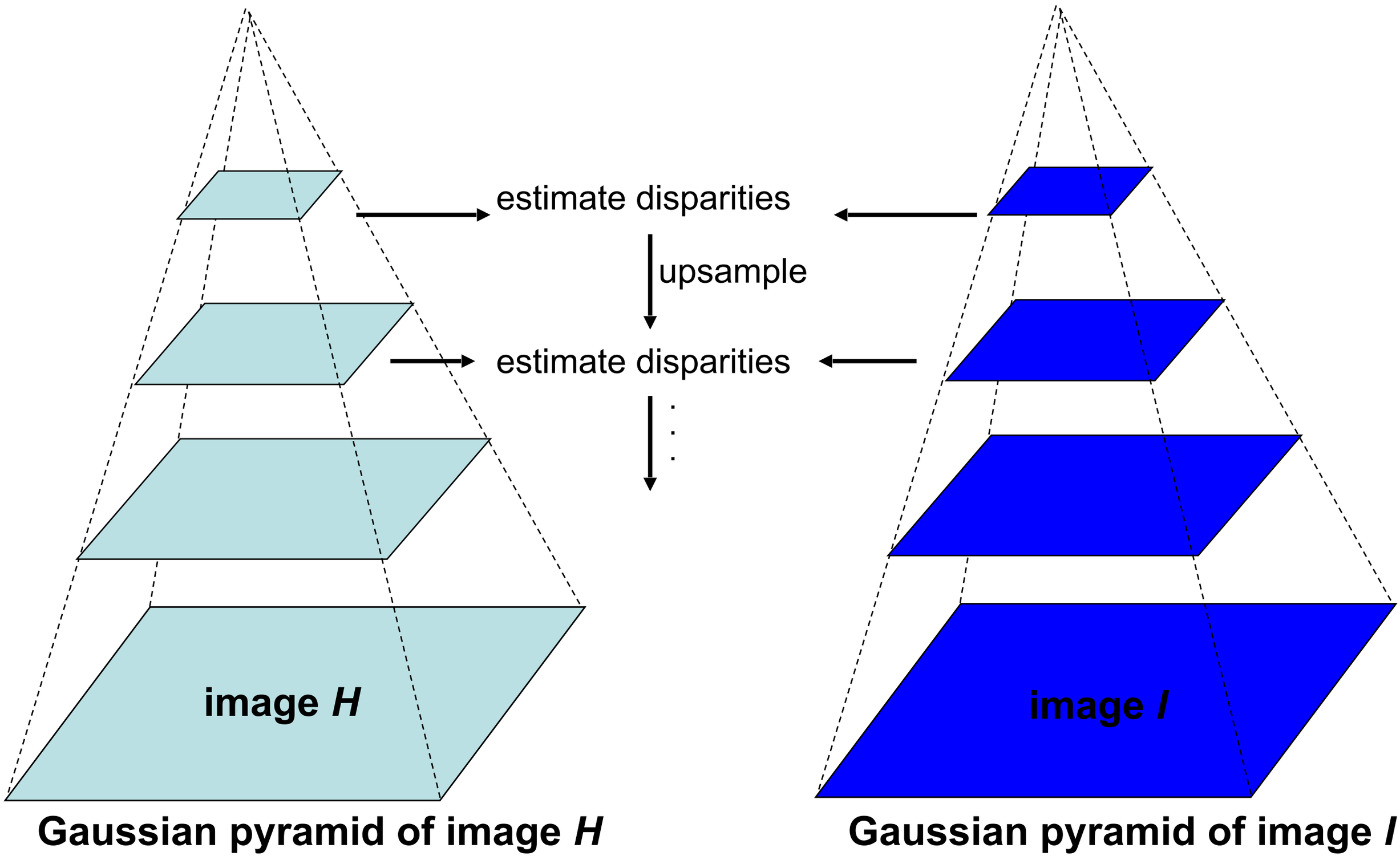
$$E_{smooth}(d) = \int\int ||\nabla d(x,y)||^2 dxdy$$



1. Linearlize $E_{intensity}$ term and solve with least squares

2. Add robust error terms $\rho(\cdot)$ to handle discontinuties

Coarse-to-fine stereo



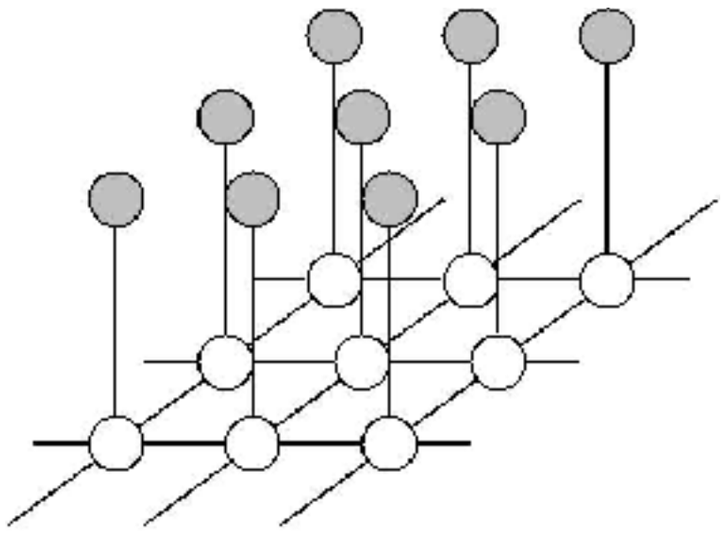
Discrete disparity estimation

$$z \in \{-5 \dots 5\}$$

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

$$\psi_{ij}(z_i, z_j) = \rho(z_i - z_j)$$

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

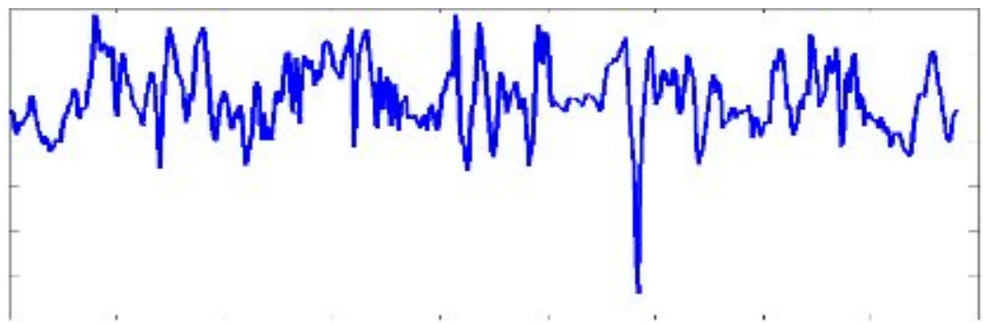


Solve with GraphCuts

Special case: single-scan-line consistency

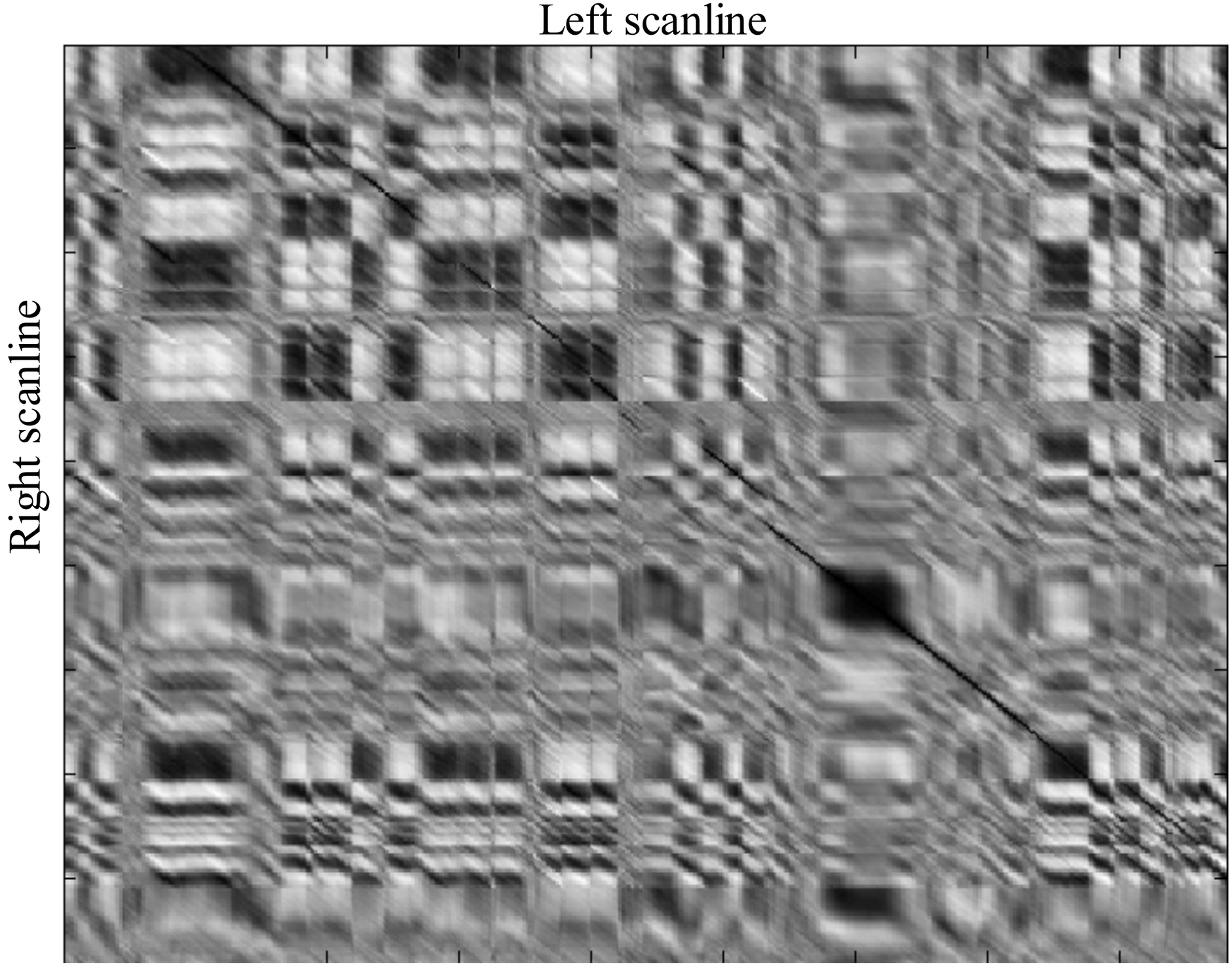
Left Image

Right Image

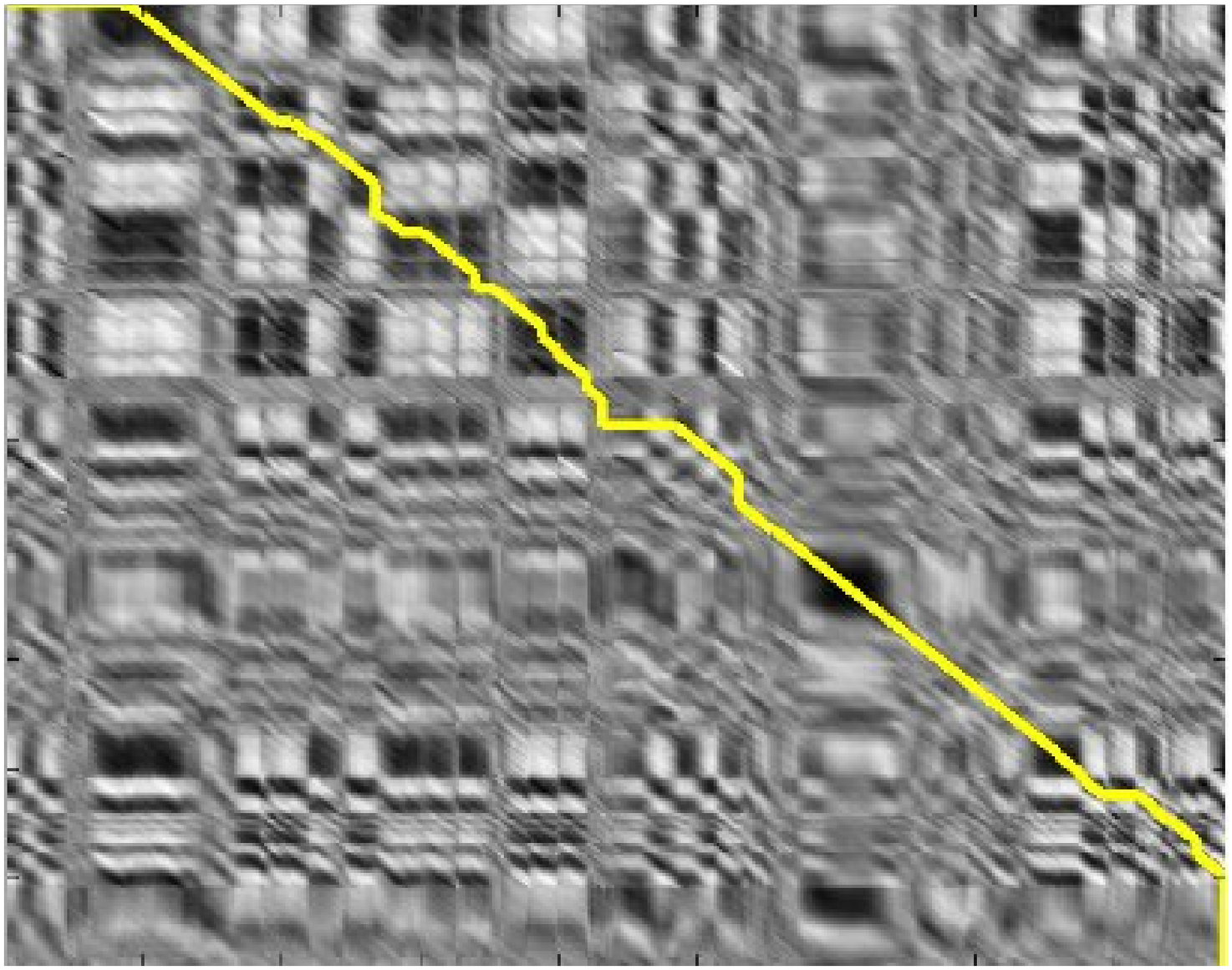


Dissimilarity Values
(1-NCC) or SSD

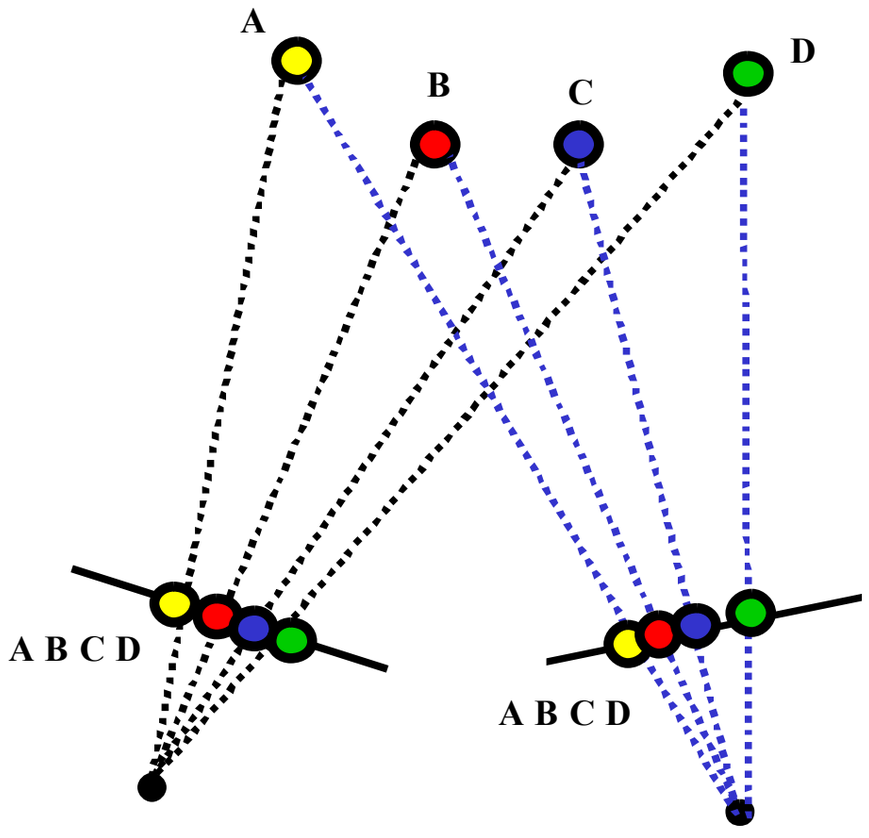
Disparity Space Image (DSI)



Representing the cost of all scanline correspondences

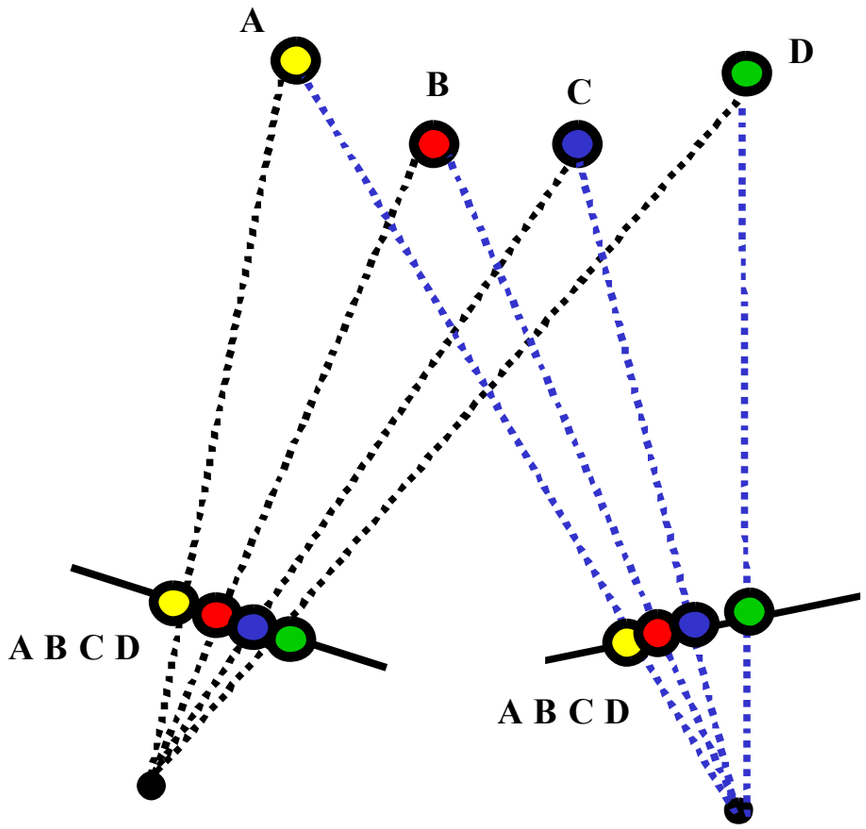


Ordering Constraint

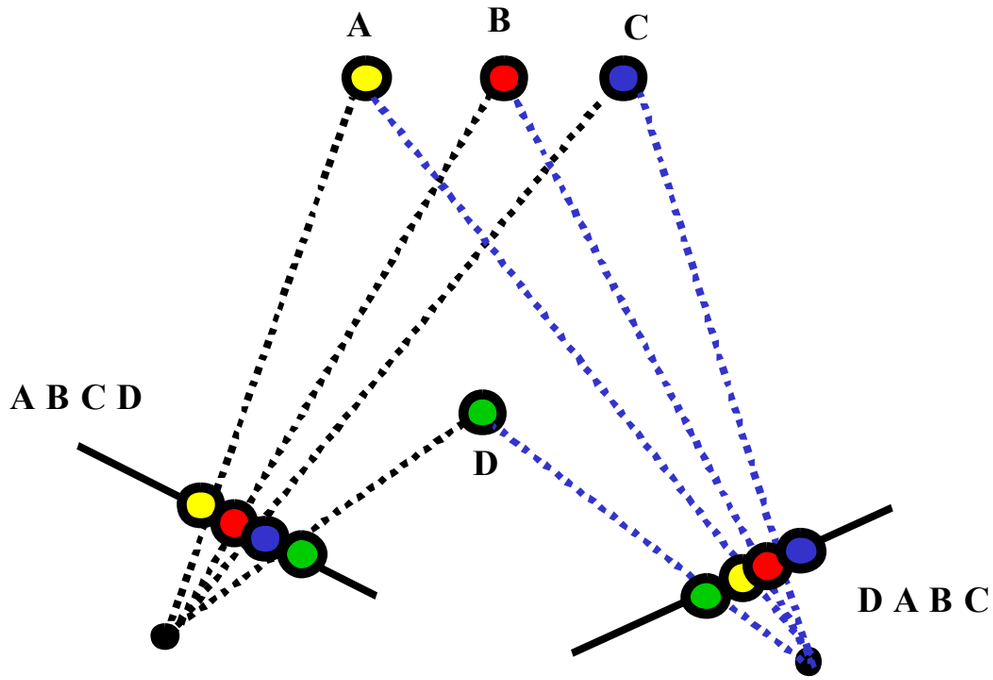


Correspondences should respect left to right order

…most of the time!

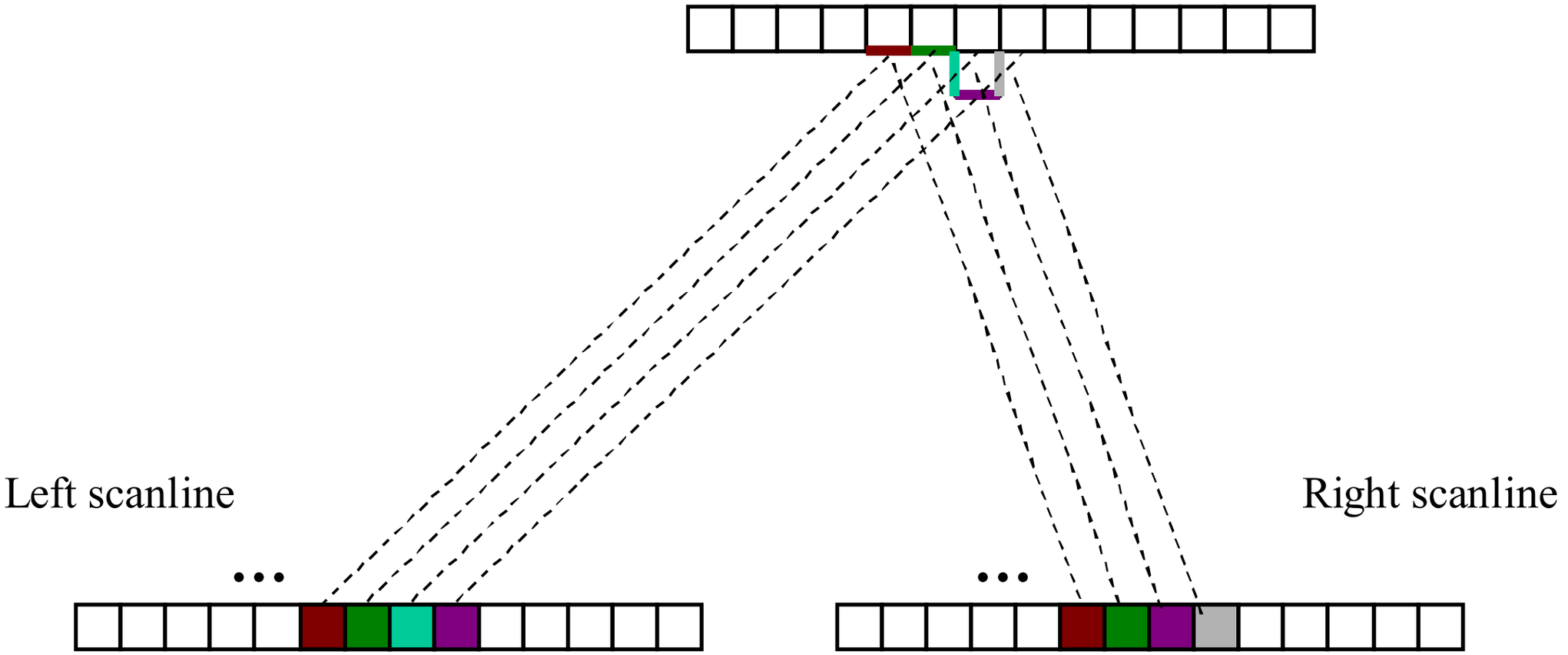


Ordering constraint…

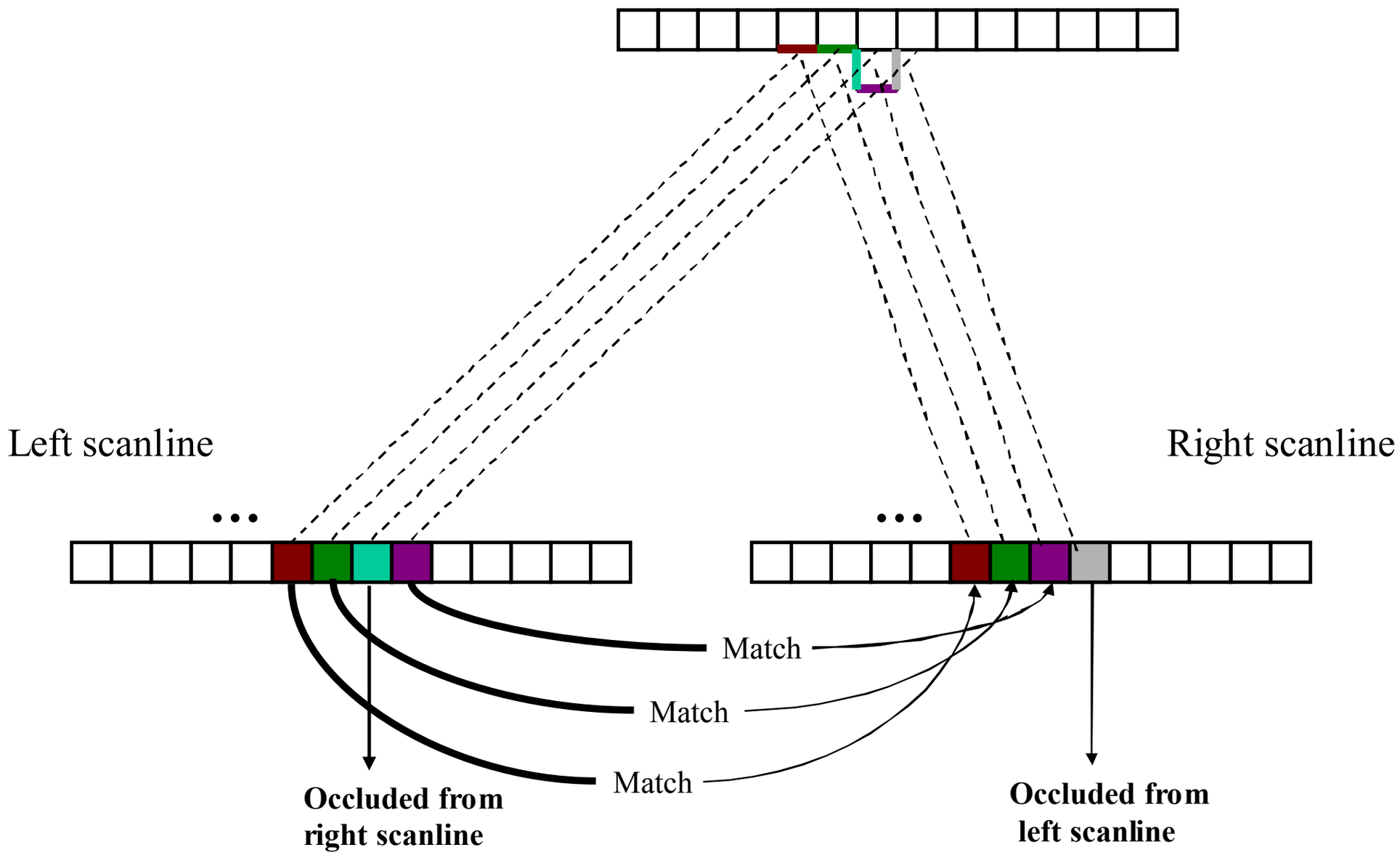


…and its failure

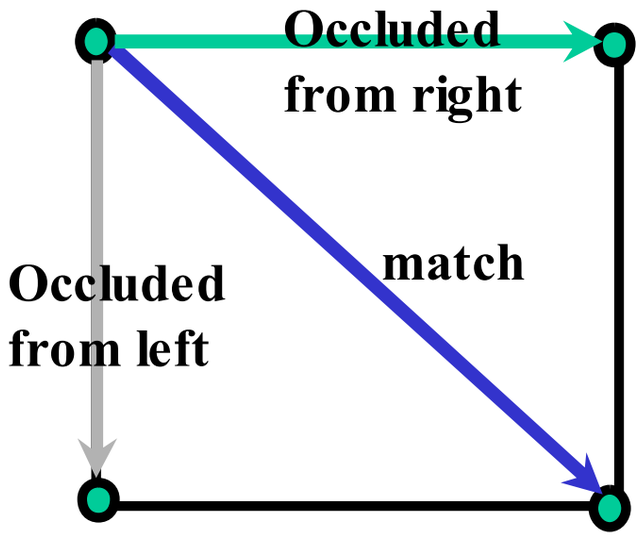
Occlusions

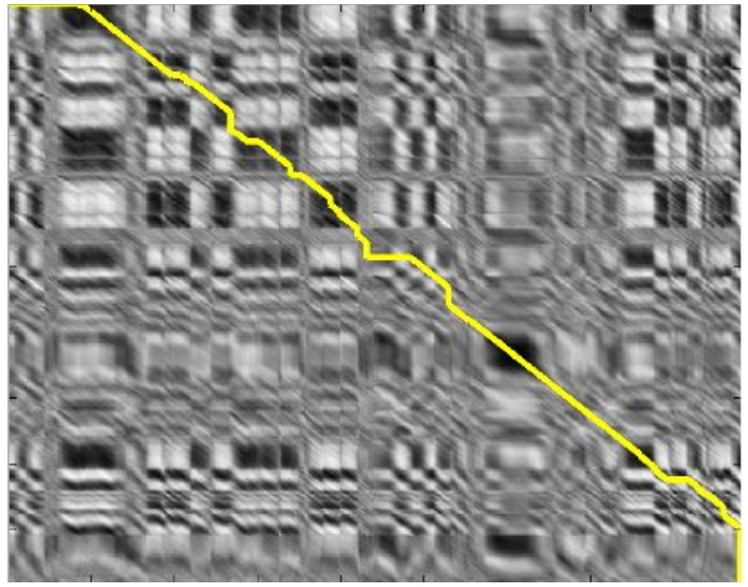


Occlusions


Left scanline
Right scanline
Match
Match
Match
Occluded from
right scanline
Occluded from
left scanline

Scanline correspondences



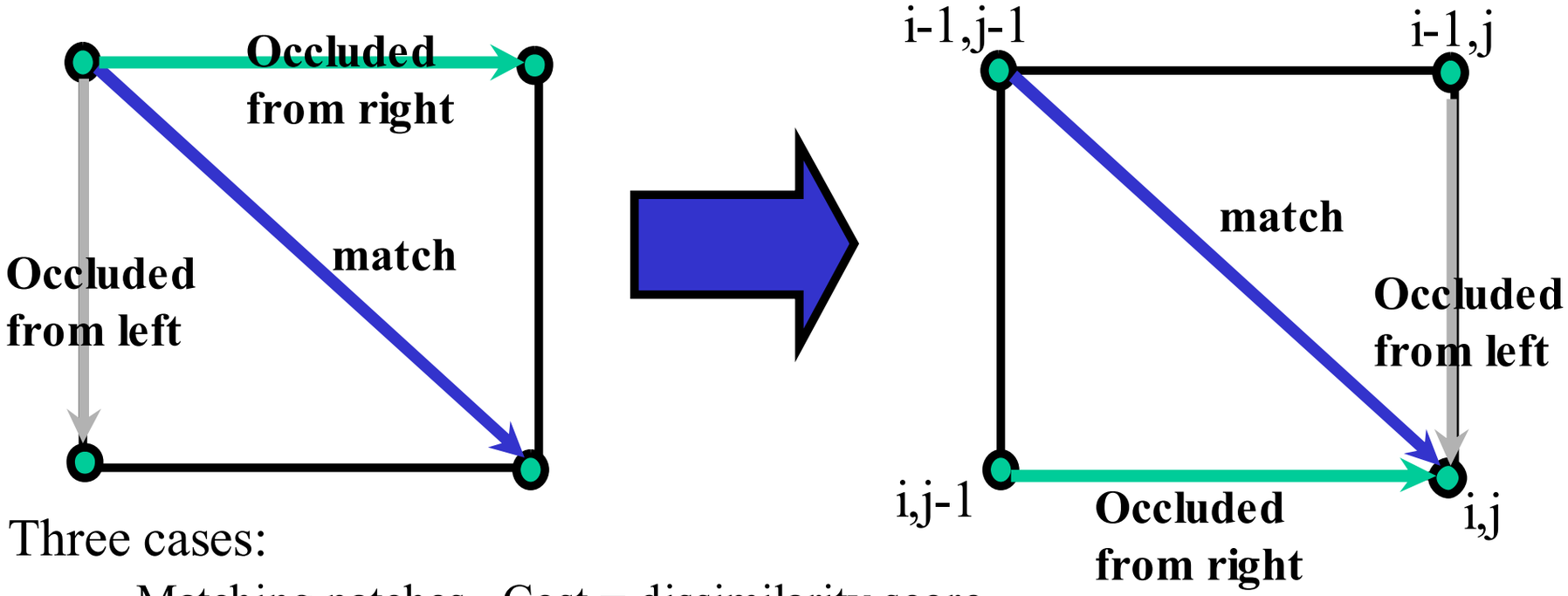


Claim: scanline correpondences with occlusions can be represented as *paths* in DSI

Let us score the *cost* of a path as sum of correspondence matching costs + occlusion costs

Best scanline correspondence = least-cost path

Compute partial scanline costs



Three cases:

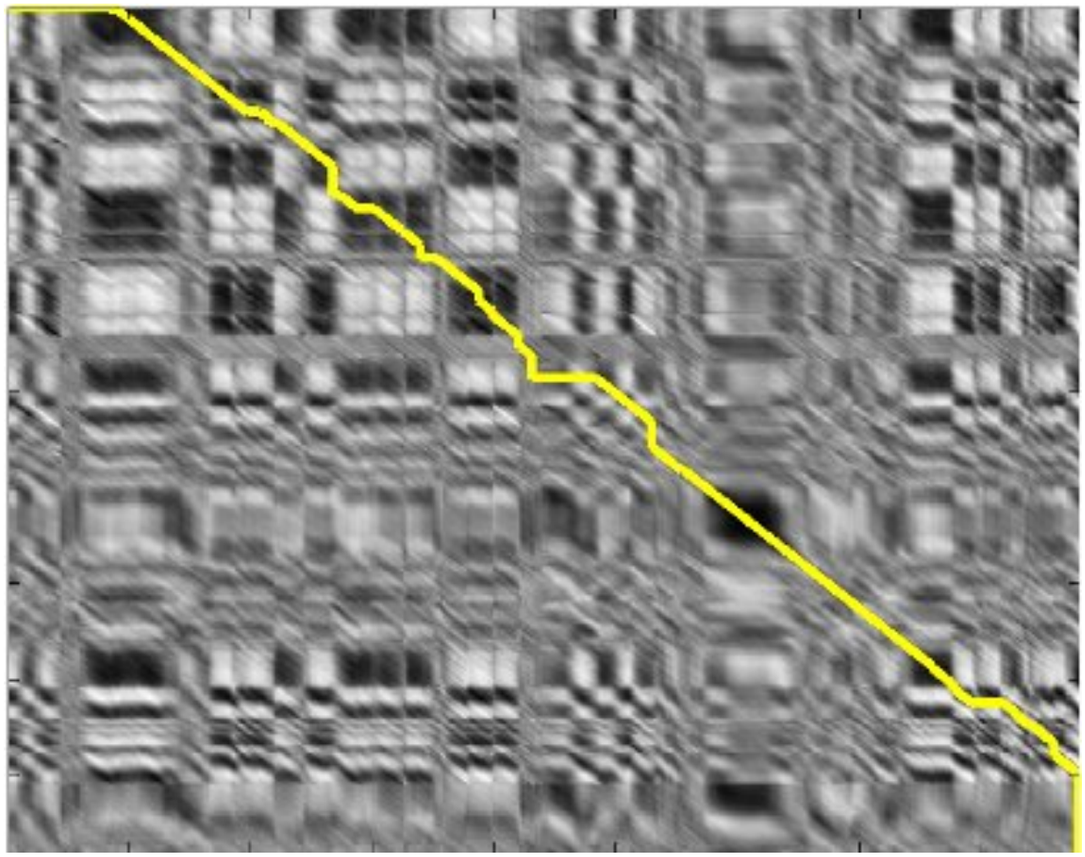
- Matching patches. Cost = dissimilarity score
- Occluded from right. Cost is some constant value.
- Occluded from left. Cost is some constant value.

$$C(i,j) = \min([\, C(i-1,j-1) + \text{dissimilarity}(i,j),\; C(i-1,j) + \text{occlusionConstant},\; C(i,j-1) + \text{occlusionConstant}]);$$

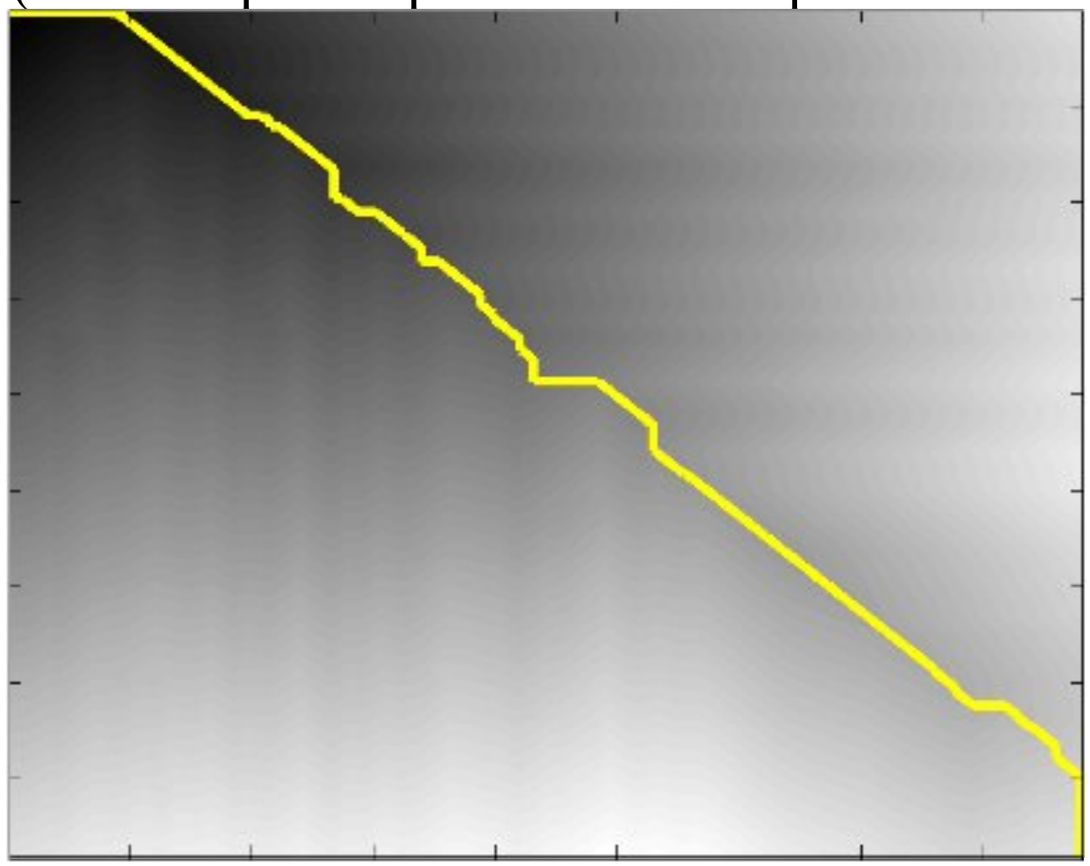
Cox, Hingorani, Rao, Maggs, "A Maximum Likelihood Stereo Algorithm," Computer Vision and Image Understanding, Vol 63(3), May 1996, pp.542-567.

Dynamic Programming

DSI



DP cost matrix
(cost of optimal path from each point to END)

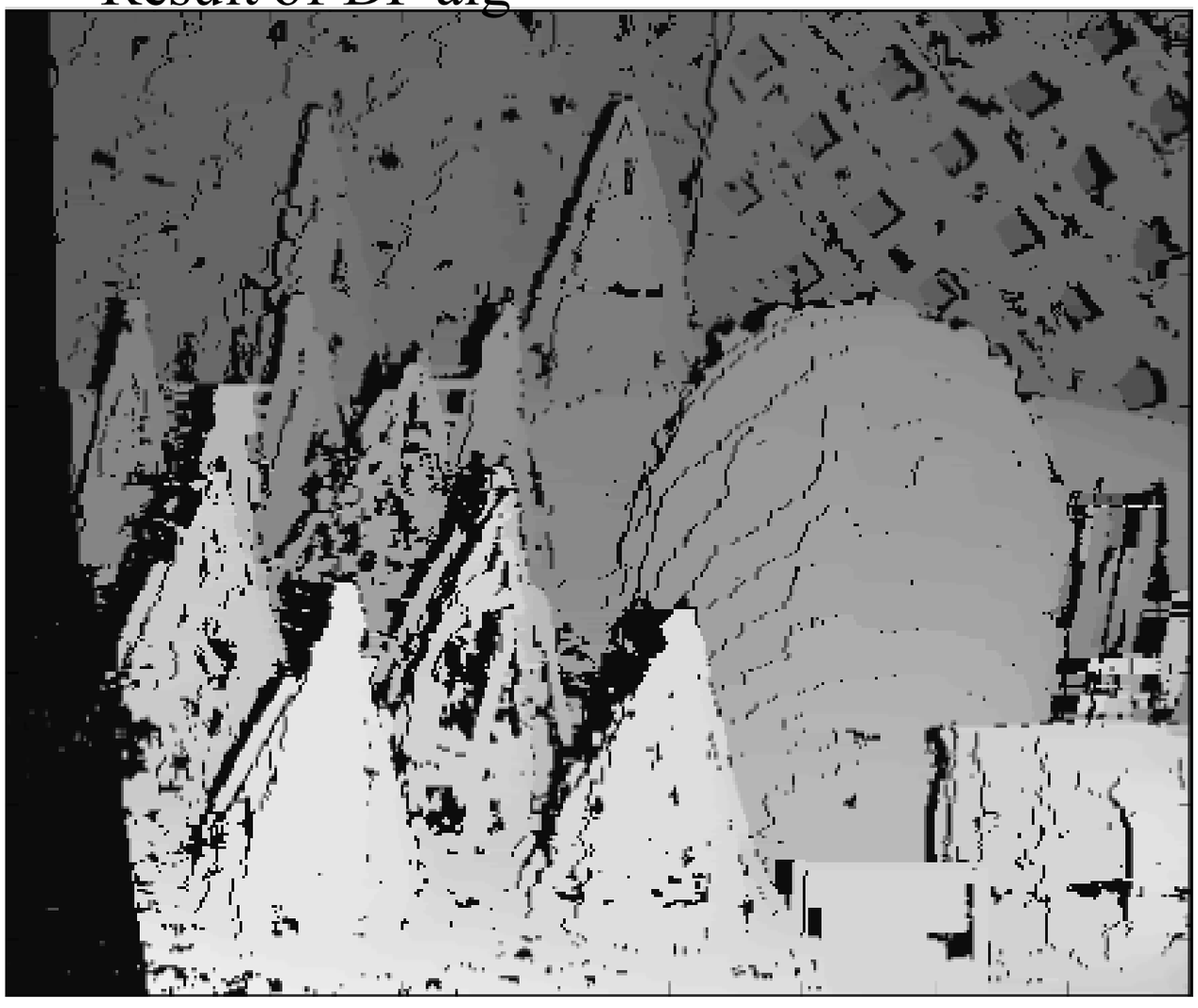


Each pixel in DSI is now marked with a disparity value or occlusion label

In practice, enforce upper bound on disparity by computing diagonal band of DSI

Results

Result of DP alg



Result without DP (independent pixels)

Result of DP alg. Black pixels = occluded.

Stereo evaluation: http://vision.middlebury.edu/stereo/

vision.middlebury.edu

stereo • mview • MRF • flow

Stereo Evaluation • Datasets • Code • Submit

Daniel Scharstein • Richard Szeliski

Welcome to the Middlebury Stereo Vision Page, formerly located at www.middlebury.edu/stereo. This website accompanies our taxonomy and comparison of two-frame stereo correspondence algorithms [1]. It contains:

- An on-line evaluation of current algorithms
- Many stereo datasets with ground-truth disparities
- Our stereo correspondence software
- An on-line submission script that allows you to evaluate your stereo algorithm in our framework

How to cite the materials on this website:

We grant permission to use and publish all images and numerical results on this website. If you report performance results, we request that you cite our paper [1]. Instructions on how to cite our datasets are listed on the datasets page. If you want to cite this website, please use the URL "**vision.middlebury.edu/stereo/**".

References:

[1] D. Scharstein and R. Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International Journal of Computer Vision*, 47(1/2/3):7-42, April-June 2002.
Microsoft Research Technical Report MSR-TR-2001-81, November 2001.

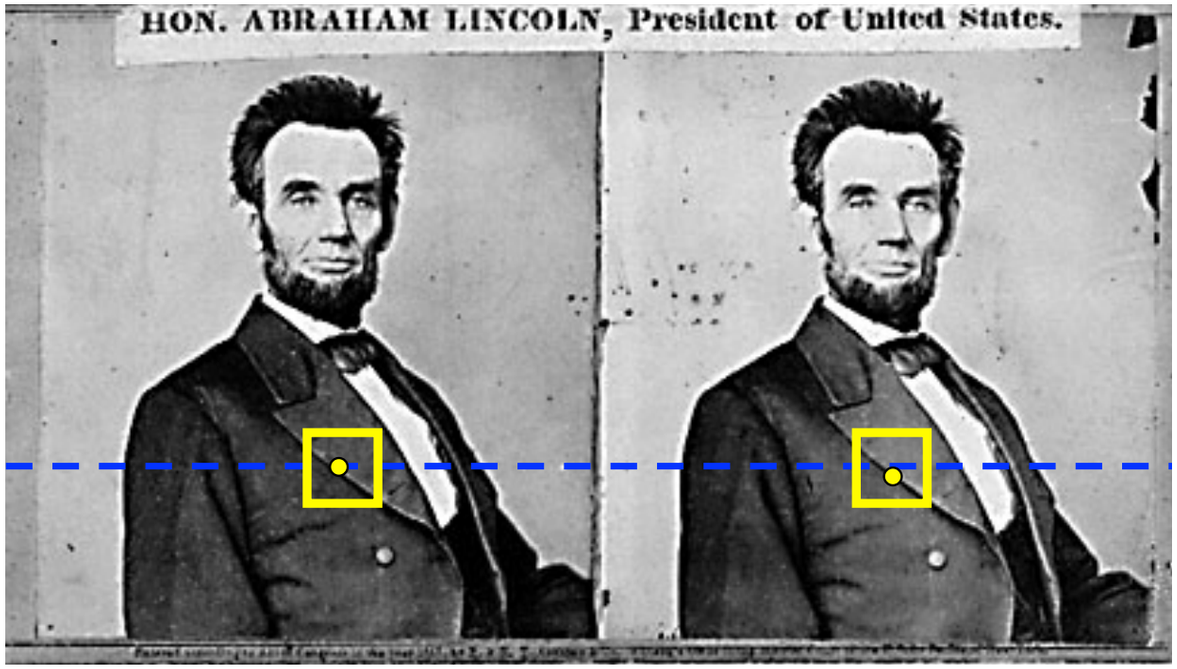
Stereo—best algorithms

Error Threshold = 1 (Error Threshold...)		Sort by nonocc			Sort by all			Sort by disc		
Algorithm	**Avg. Rank**	**Tsukuba** ground truth nonocc	all	disc	**Venus** ground truth nonocc	all	disc	**Teddy** ground truth nonocc	all	disc	**Cones** ground truth nonocc	all	disc
AdaptingBP [17]	2.8	1.11 6	1.37 3	5.79 7	**0.10** 1	0.21 2	**1.44** 1	4.22 4	7.06 2	11.8 4	**2.48** 1	7.92 2	**7.32** 1
DoubleBP2 [35]	2.9	**0.88** 1	**1.29** 1	**4.76** 1	0.13 3	0.45 5	1.87 5	3.53 2	8.30 3	**9.63** 1	2.90 3	8.78 8	7.79 2
DoubleBP [15]	4.9	0.88 2	1.29 2	4.76 2	0.14 5	0.60 13	2.00 7	3.55 3	8.71 5	9.70 2	2.90 4	9.24 11	7.80 3
SubPixDoubleBP [30]	5.6	1.24 10	1.76 13	5.98 8	0.12 2	0.46 6	1.74 4	**3.45** 1	8.38 4	10.0 3	2.93 5	8.73 7	7.91 4
AdaptOvrSegBP [33]	9.9	1.69 22	2.04 21	5.64 6	0.14 4	**0.20** 1	1.47 2	7.04 14	11.1 7	16.4 11	3.60 11	8.96 10	8.84 10
SymBP+occ [7]	10.8	0.97 4	1.75 12	5.09 4	0.16 6	0.33 3	2.19 8	6.47 8	10.7 6	17.0 14	4.79 24	10.7 21	10.9 20
PlaneFitBP [32]	10.8	0.97 5	1.83 14	5.26 5	0.17 7	0.51 8	1.71 3	6.65 9	12.1 13	14.7 7	4.17 20	10.7 20	10.6 19
AdaptDispCalib [36]	11.8	1.19 8	1.42 4	6.15 9	0.23 9	0.34 4	2.50 11	7.80 19	13.6 21	17.3 17	3.62 12	9.33 12	9.72 15
Segm+visib [4]	12.2	1.30 15	1.57 5	6.92 18	0.79 21	1.06 18	6.76 22	5.00 5	**6.54** 1	12.3 5	3.72 13	8.62 6	10.2 17
C-SemiGlob [19]	12.3	2.61 29	3.29 24	9.89 27	0.25 12	0.57 10	3.24 15	5.14 6	11.8 8	13.0 6	2.77 2	8.35 4	8.20 5
SO+borders [29]	12.8	1.29 14	1.71 9	6.83 15	0.25 13	0.53 9	2.26 9	7.02 13	12.2 14	16.3 9	3.90 15	9.85 16	10.2 18
DistinctSM [27]	14.1	1.21 9	1.75 11	6.39 11	0.35 14	0.69 16	2.63 13	7.45 18	13.0 17	18.1 19	3.91 16	9.91 18	8.32 7
CostAggr+occ [39]	14.3	1.38 17	1.96 17	7.14 19	0.44 16	1.13 19	4.87 19	6.80 11	11.9 10	17.3 16	3.60 10	8.57 5	9.36 13
OverSegmBP [26]	14.5	1.69 23	1.97 18	8.47 24	0.51 18	0.68 15	4.69 18	6.74 10	11.9 12	15.8 8	3.19 8	8.81 9	8.89 11
SegmentSupport [28]	15.1	1.25 11	1.62 7	6.68 13	0.25 11	0.64 14	2.59 12	8.43 24	14.2 22	18.2 20	3.77 14	9.87 17	9.77 16
RegionTreeDP [18]	15.7	1.39 19	1.64 8	6.85 16	0.22 8	0.57 10	1.93 6	7.42 17	11.9 11	16.8 13	6.31 30	11.9 27	11.8 23
EnhancedBP [24]	16.6	0.94 3	1.74 10	5.05 3	0.35 15	0.86 17	4.34 17	8.11 22	13.3 19	18.5 22	5.09 27	11.1 23	11.0 21

Multiview stereo

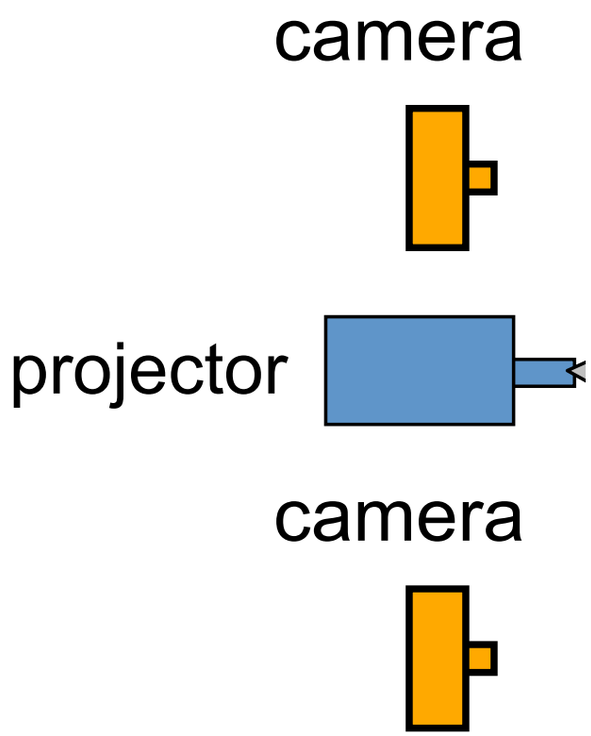
- stereo / dynamic programming
- active illumination
- volumetric models / visiblity reasoning
- patch-based methods

Hard part: find matching points

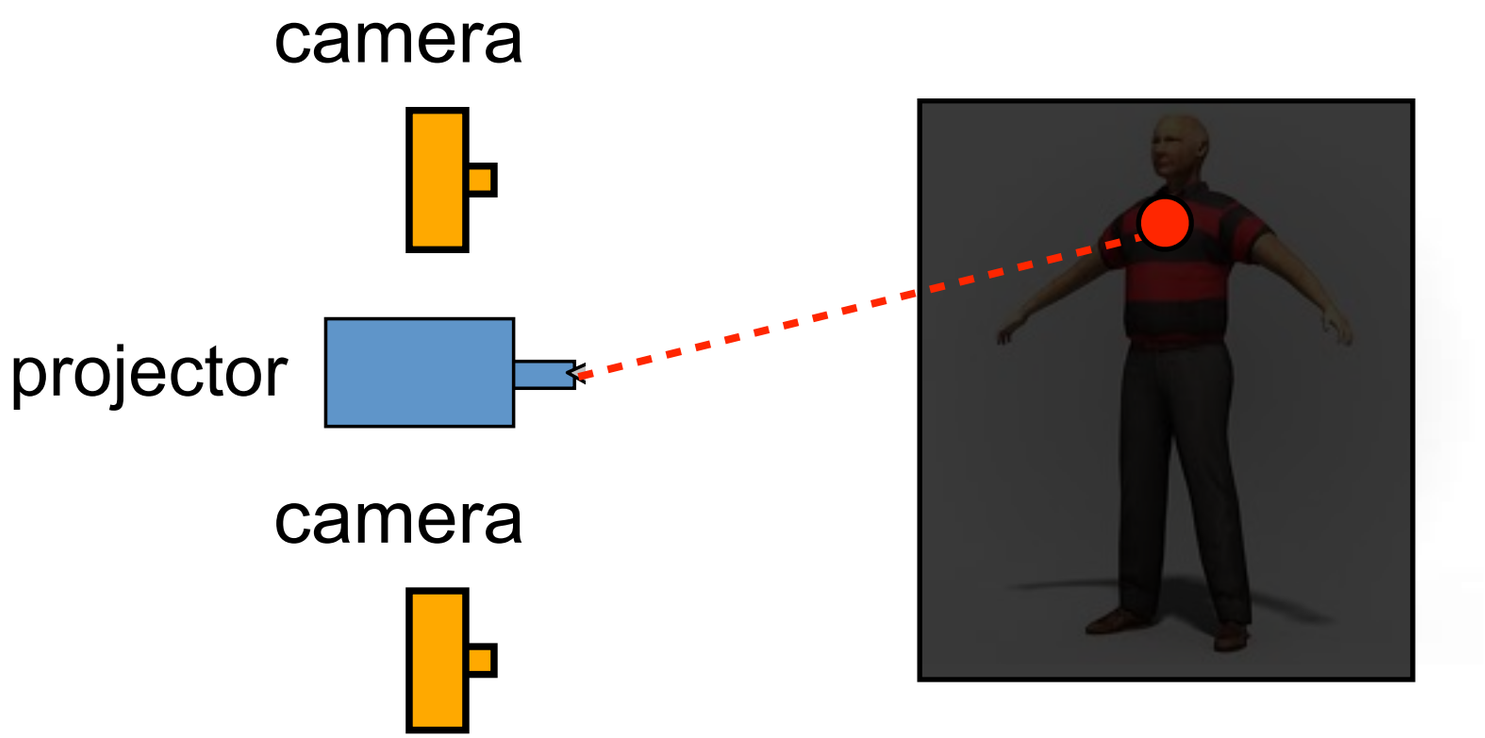


If patches look distinctive, they'll be easy to match
But lots of patches are not distinctive

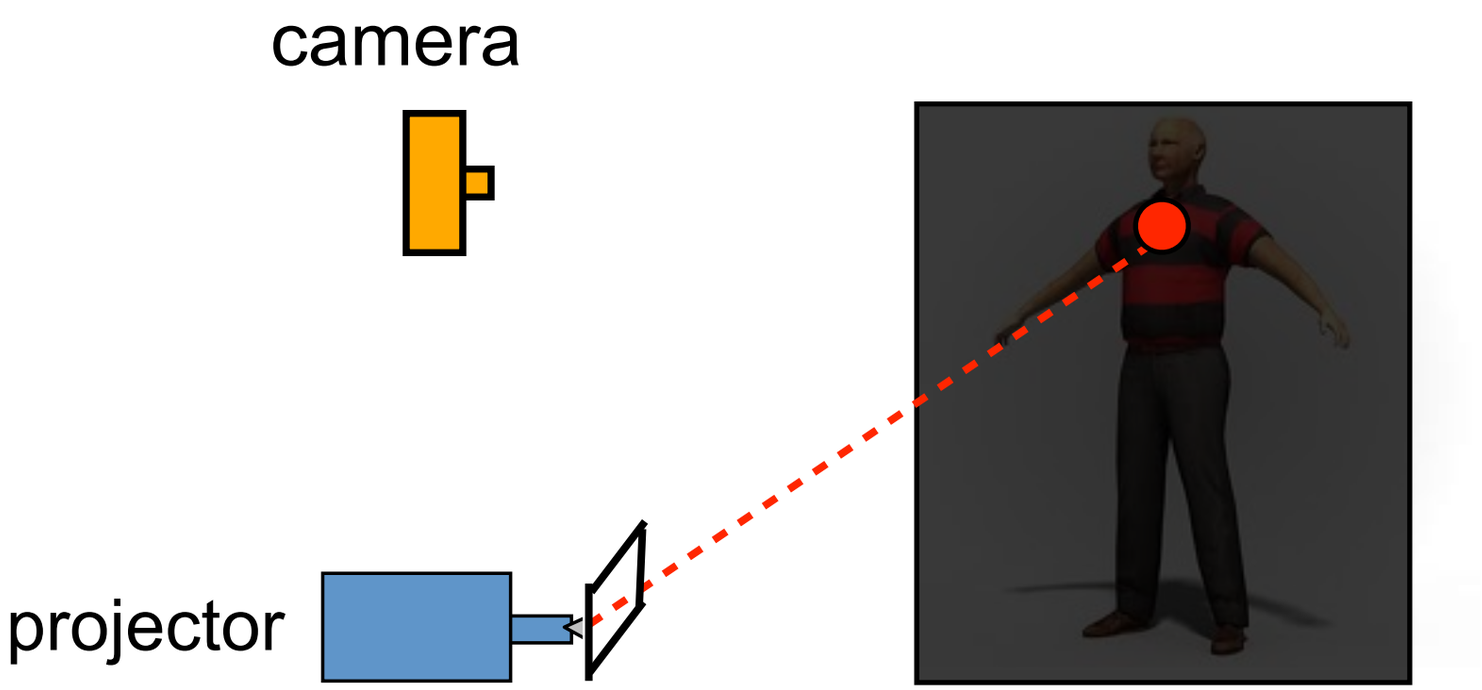
Trick: use a projector to project a distinctive appearance



Trick: use a projector to project a distinctive appearance



Trick: use a projector to project a distinctive appearance



Use projector as camera

Kinect

projector

camera

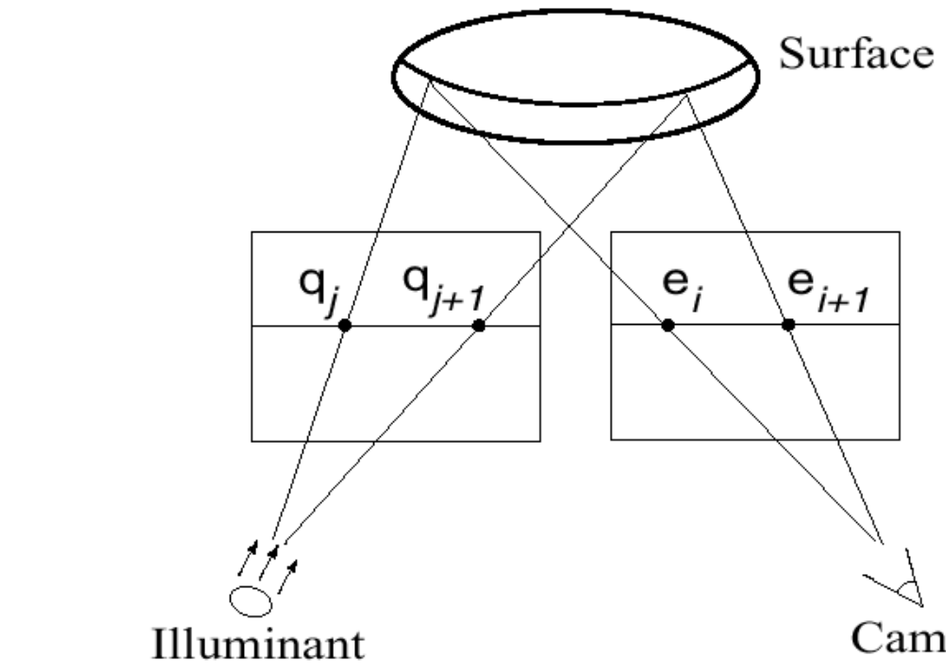
Infrared filter

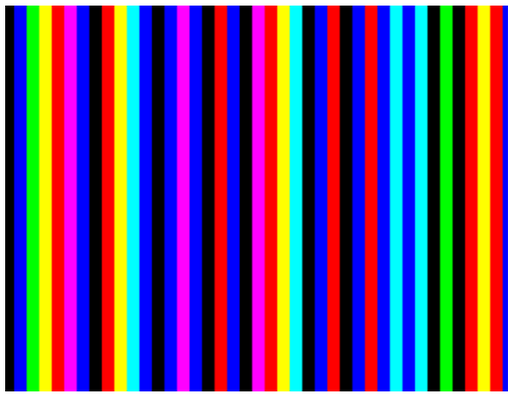
Spatially-varying speckle patterns

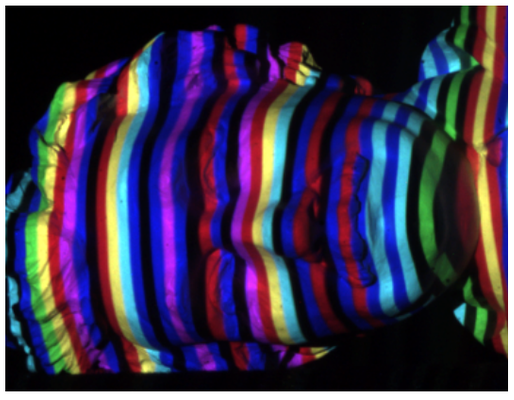
QR code

General approach: active illumination

Project an instantaneous pattern for which its easy to find correspondences





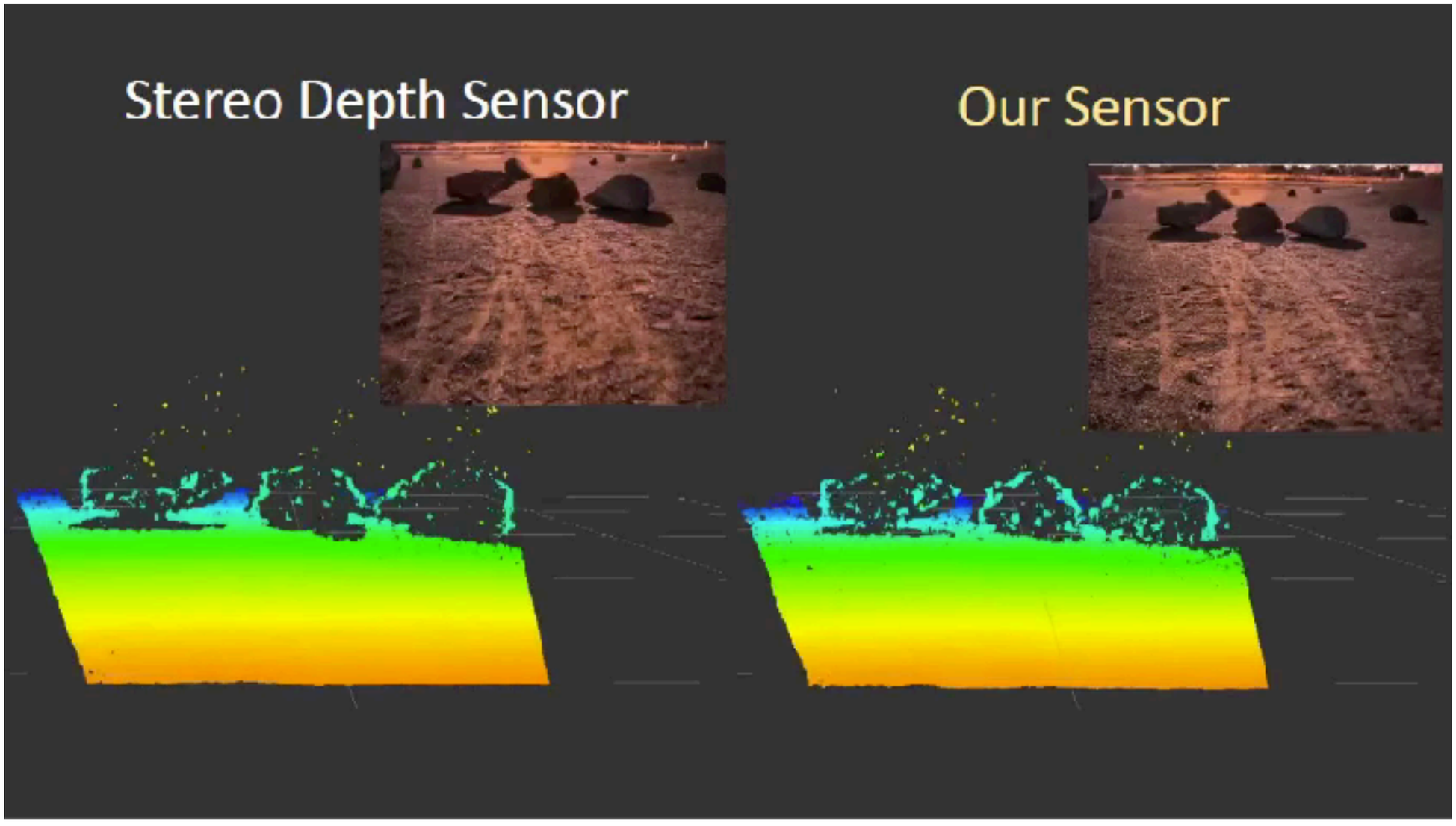


Search along epipolar line to find right color

Kinect

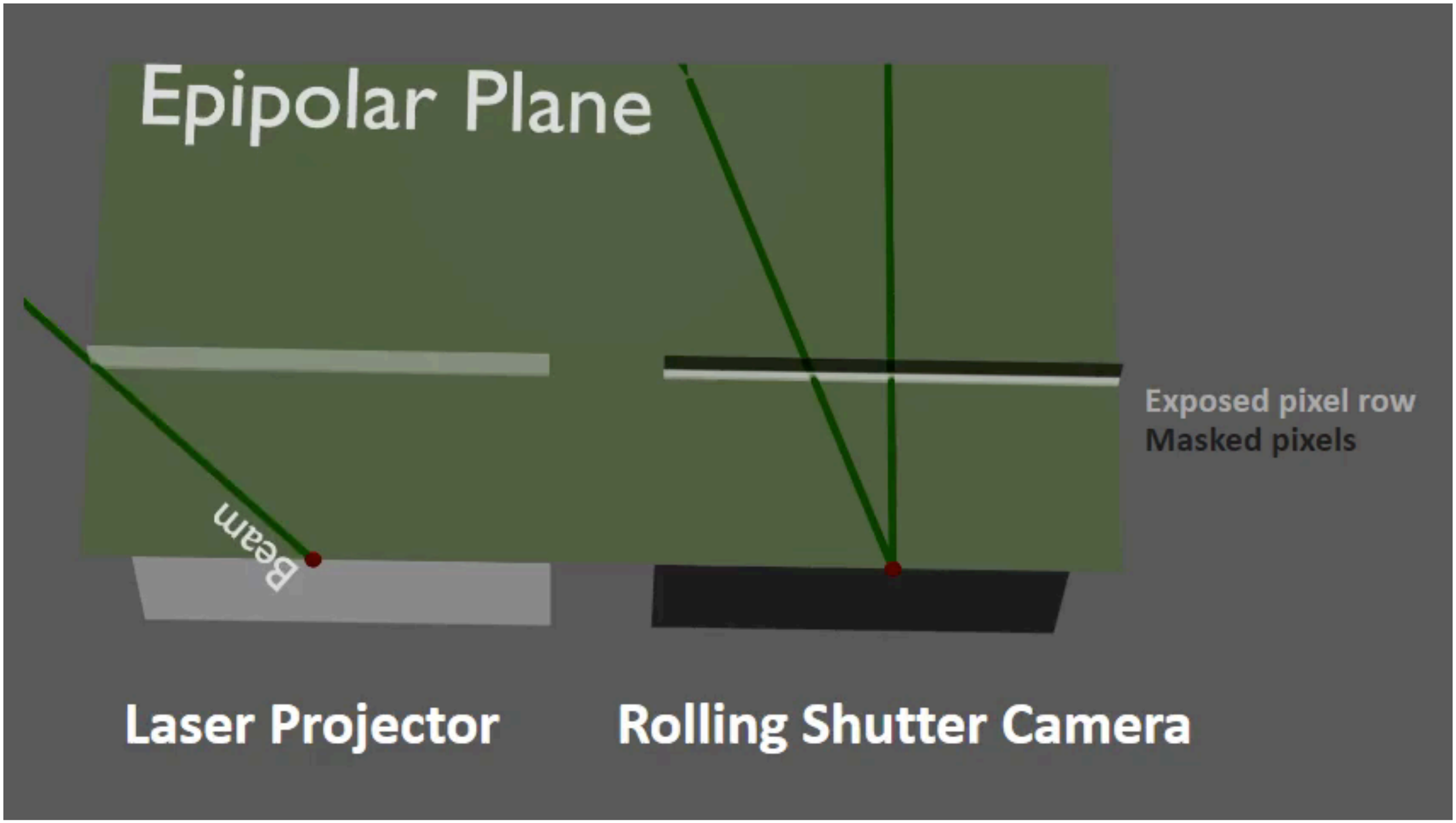
Why do we see “holes” around object borders?
Why can’t you use kinect outdoors?
Why can’t you use two kinects in the same room?

Outdoor kinect



Episcan sensor, CMU

Episcan scanner



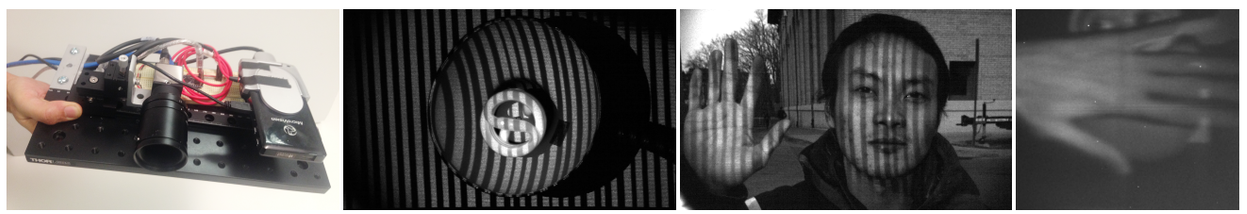
Homogeneous Codes for Energy-Efficient Illumination and Imaging

Matthew O'Toole
University of Toronto

Supreeth Achar
Carnegie Mellon University

Srinivasa G. Narasimhan
Carnegie Mellon University

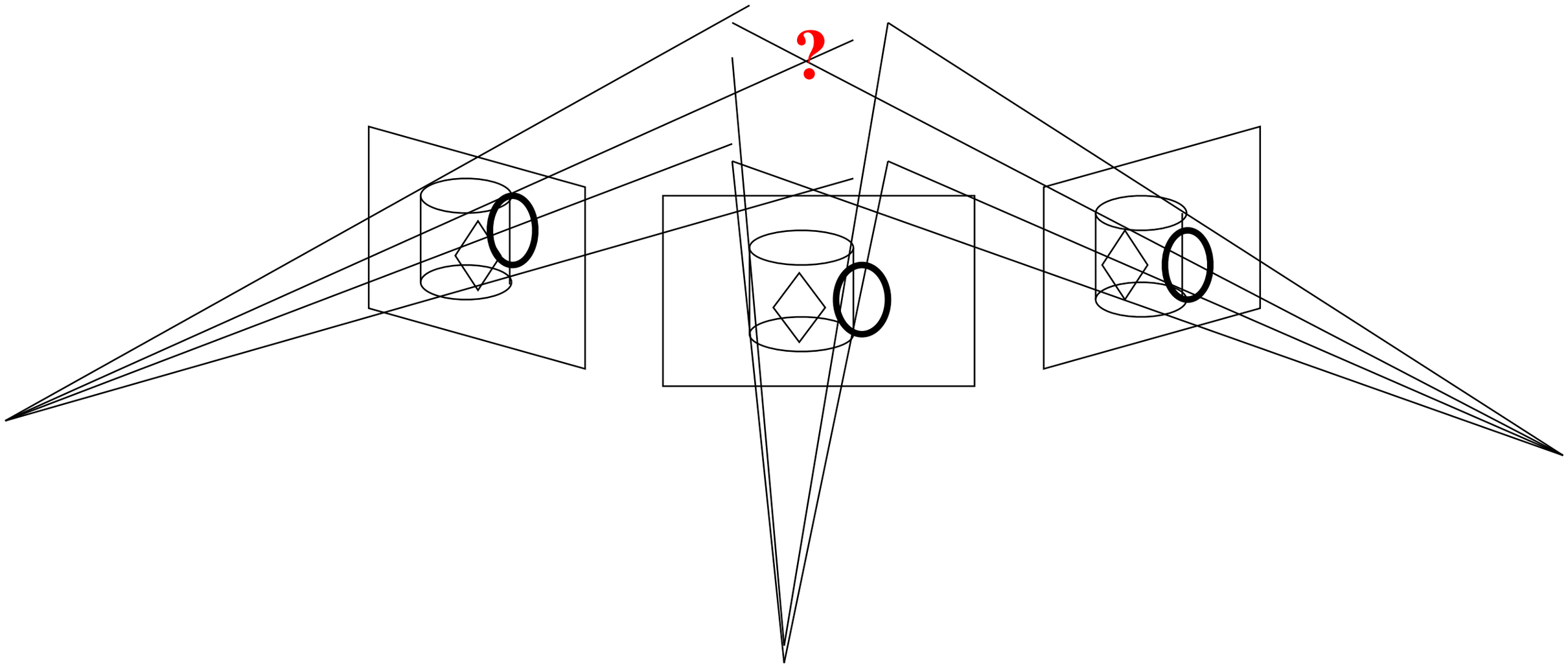
Kiriakos N. Kutulakos
University of Toronto



Multiview stereo

- stereo / dynamic programming
- active illumination
- volumetric models / visiblity reasoning
- patch-based methods

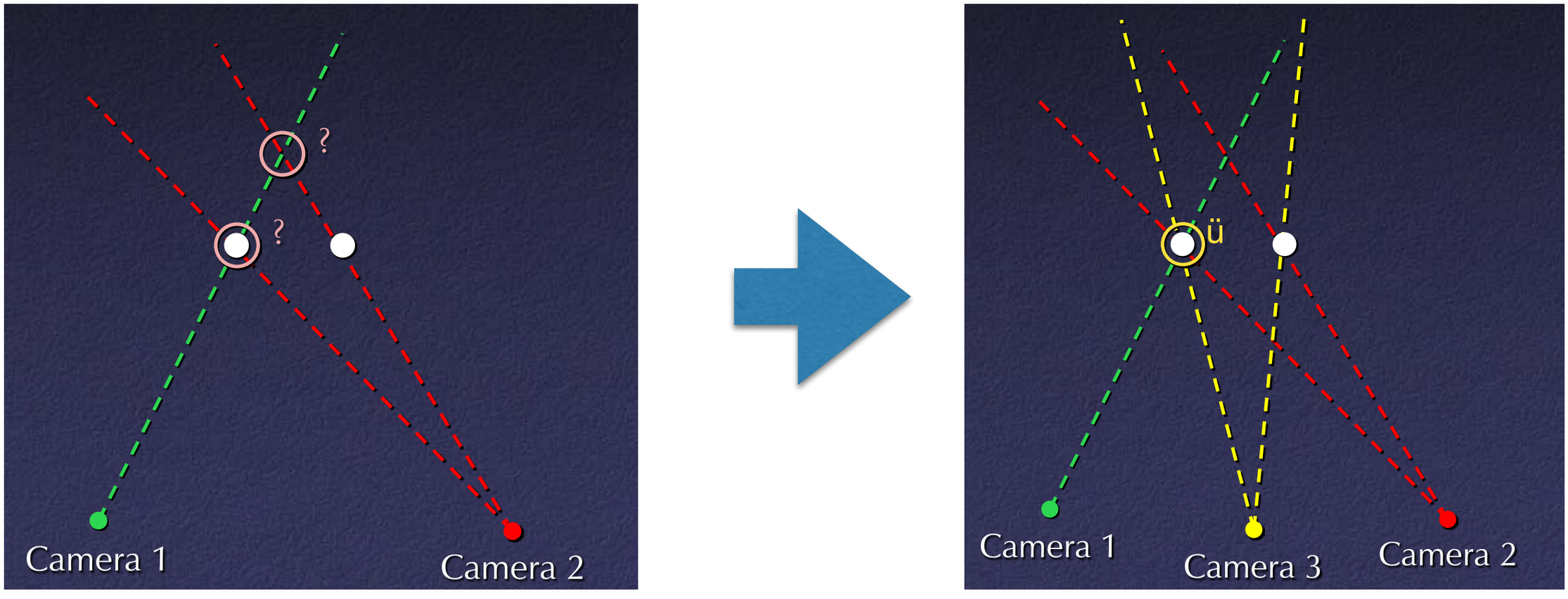
Dense multi view stereo



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

Trinocular stereo (version 0)

1. Pick 2 views, find correspondences
2. For each matching pair, reconstruct 3D point
3. If can't find correspondence near projected location, reject



Version 1: generalize 3x3 fundmamental matrix to a 3x3x3 trifocal tensor

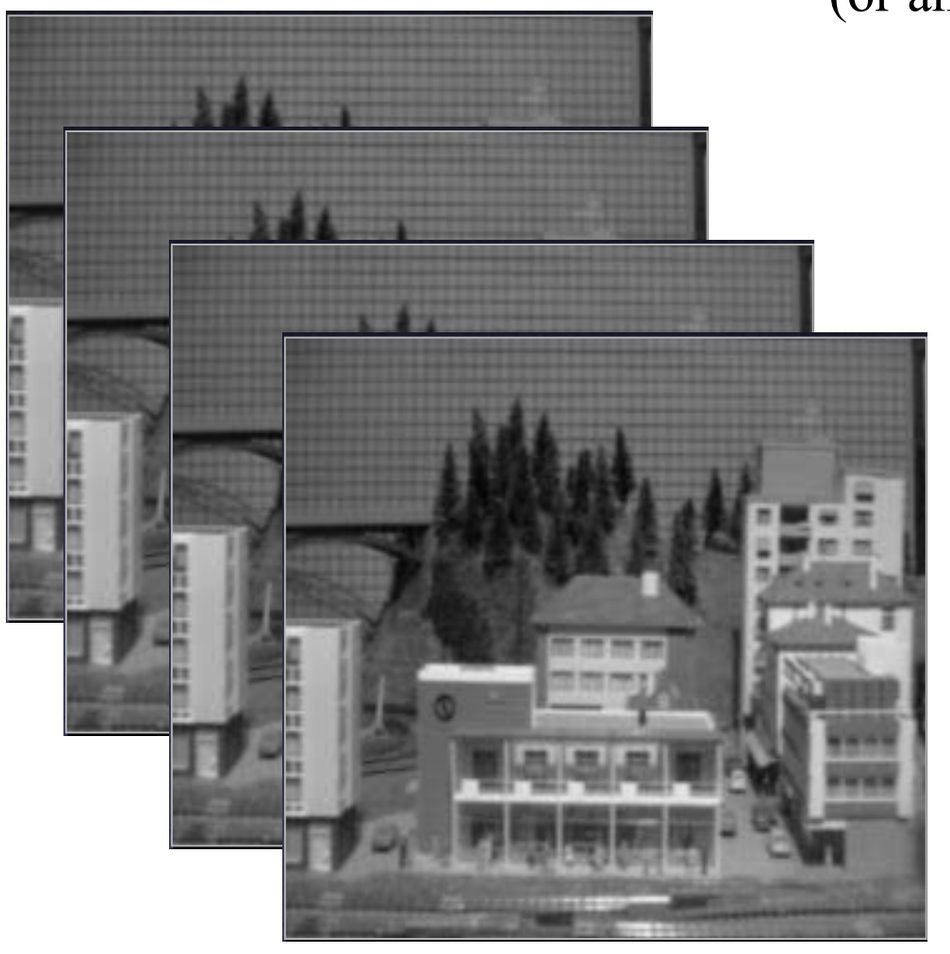
(constraints points and lines across 3 images)

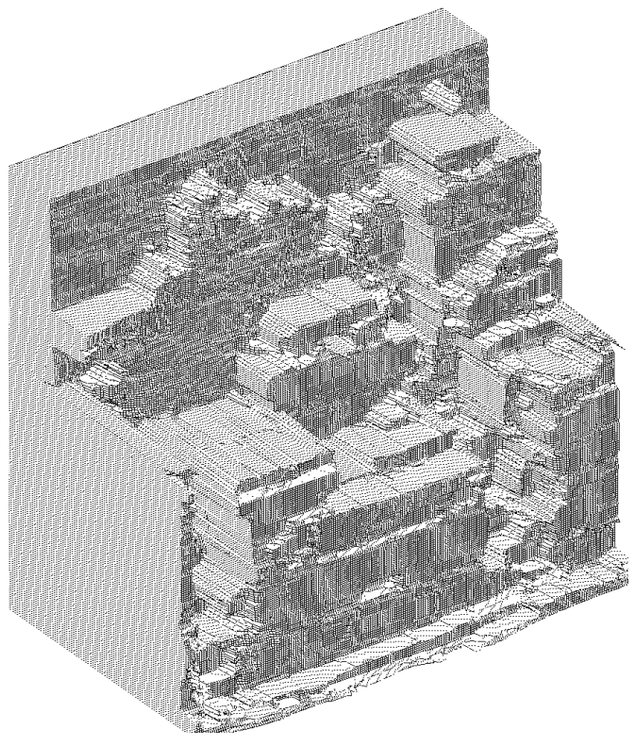
Multiview stereo (version 0)

-Pick one reference view
-For each point and for each candidate depth

- keep depths with low SSD error in all other views

(or any *photoconsistency* measure)



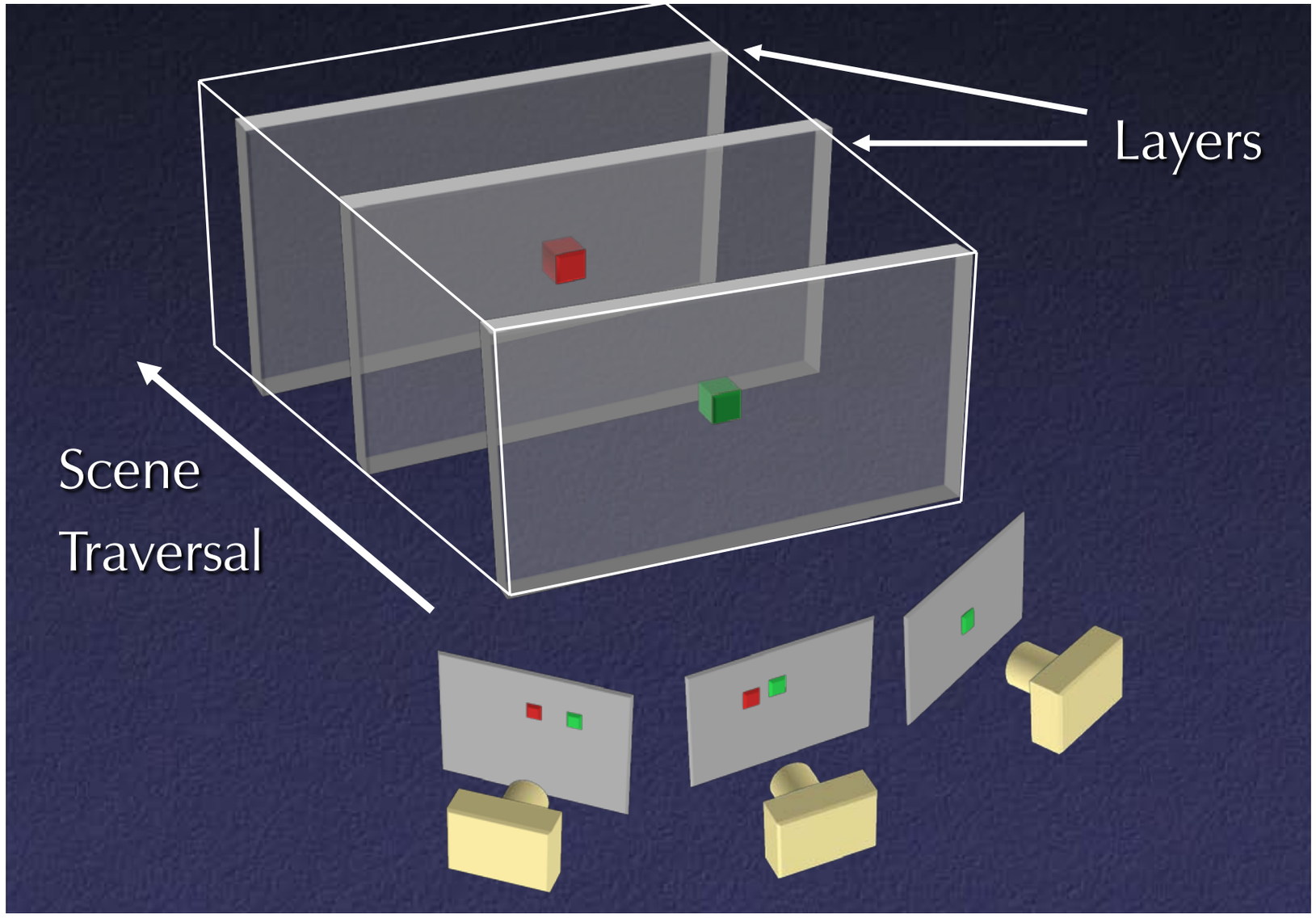


Problem: not all points are visible in all other views (occlusion and visibility major nuisance!)

Multiview stereo (version 1)

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

Use knowledge of occluding points to smartly select view for photoconsistency check



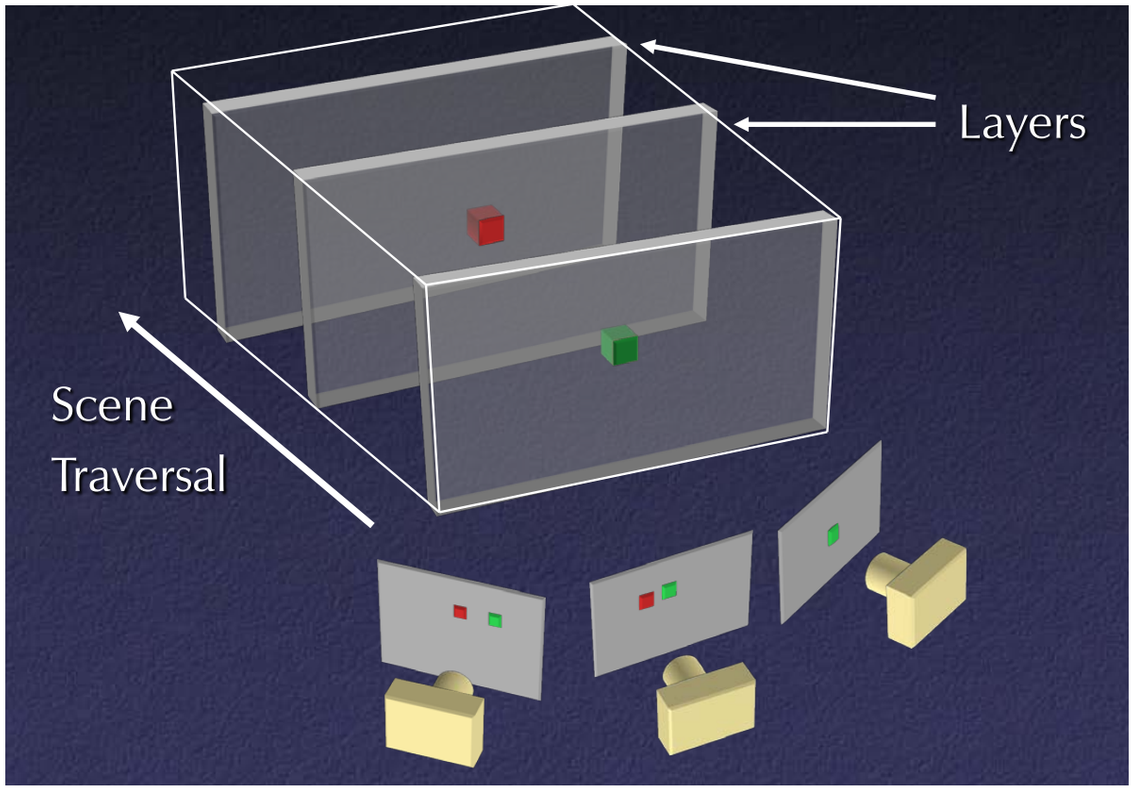
Store photoconsistent color in a 3D voxel grid (don’t need a reference image)

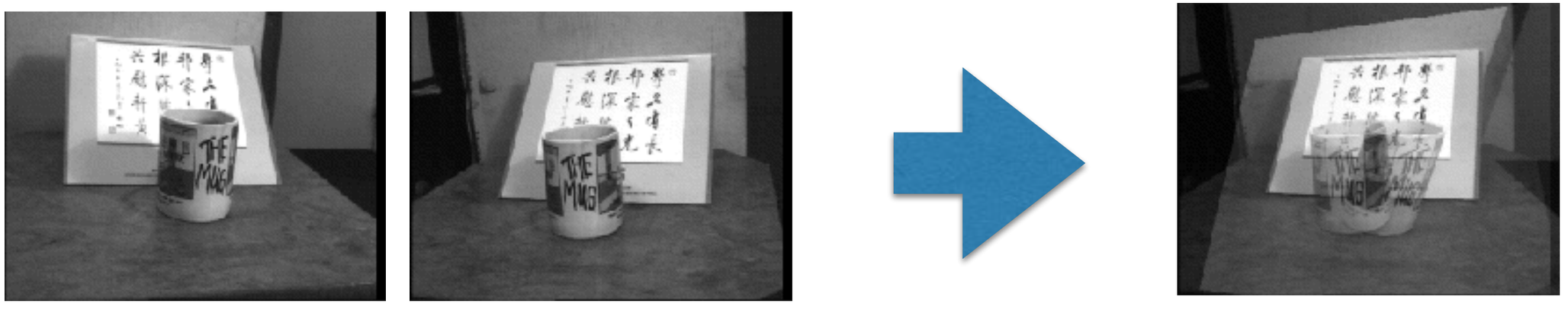
Reconstuct shape *and* appearance

Plane-sweep stereo (version 2)

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

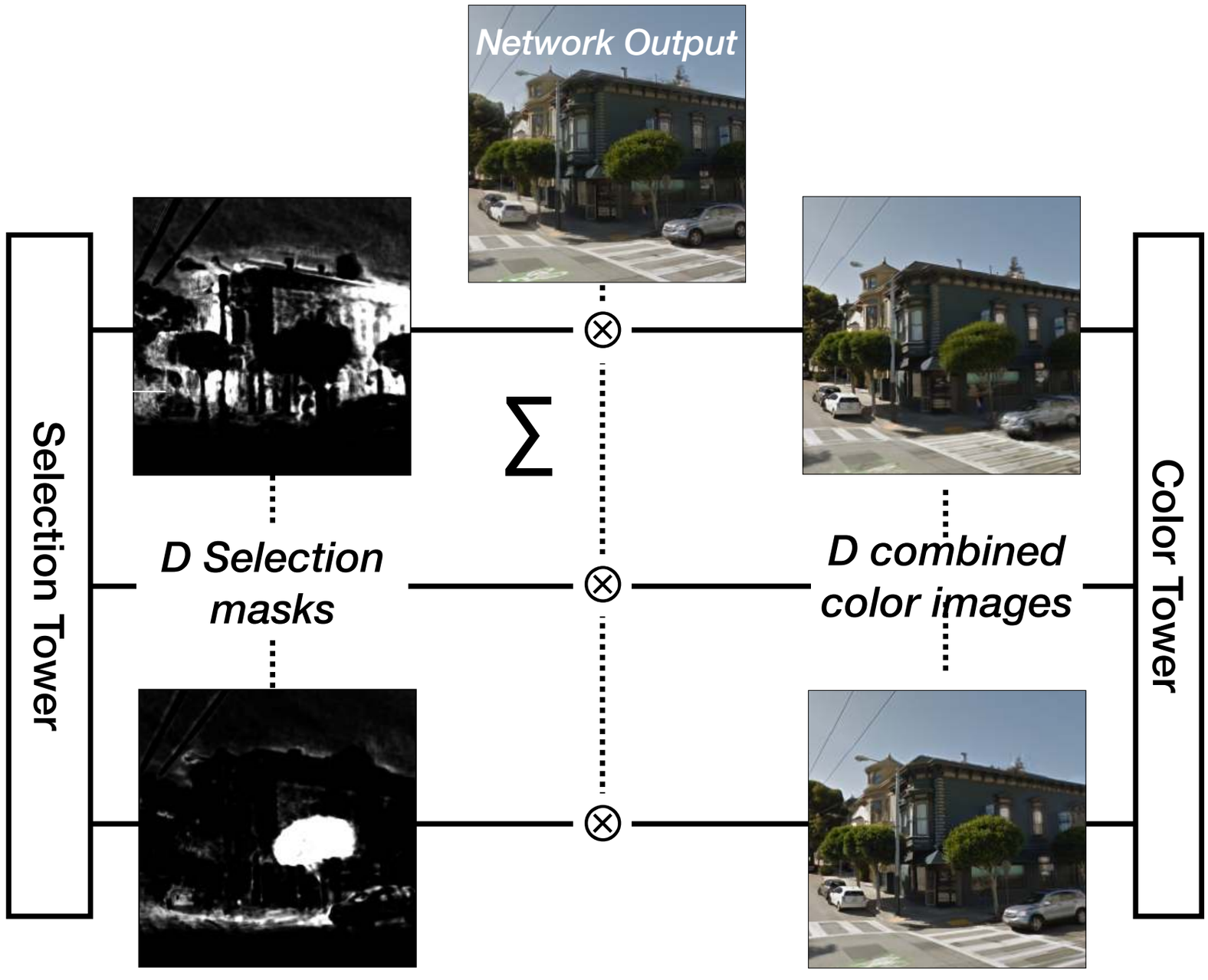
Quickly validate voxels in a plane by computing their appearance in a virtual view using all N cameras





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

Recent work: deep stereo



Train deep network select pixel from 1 of K depth planes

(At each pixel, output 1 of K classes)

Recent work: deep stereo

DeepStereo: Learning to Predict New Views from the World's Imagery

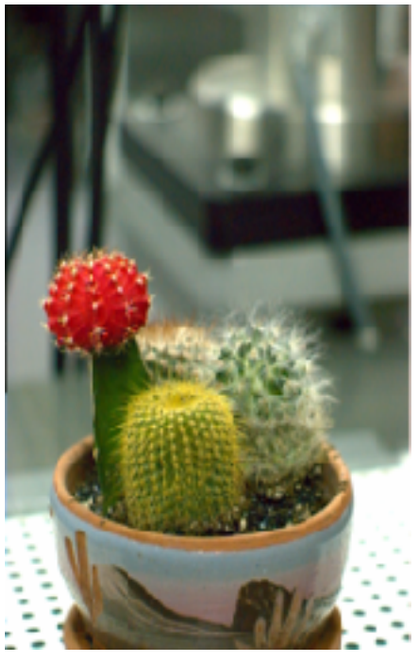
John Flynn
Zoox Inc.*
john.flynn@zoox.com

Ivan Neulander
Google Inc.
ineula@google.com

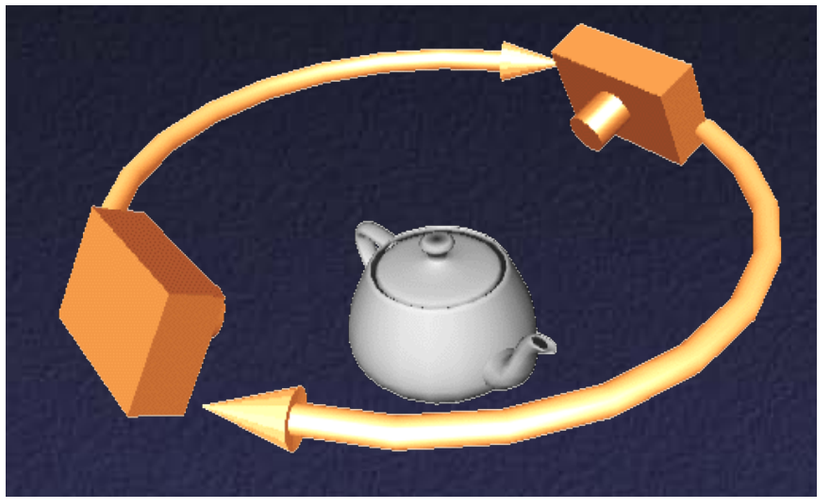
James Philbin
Zoox Inc.*
james@zoox.com

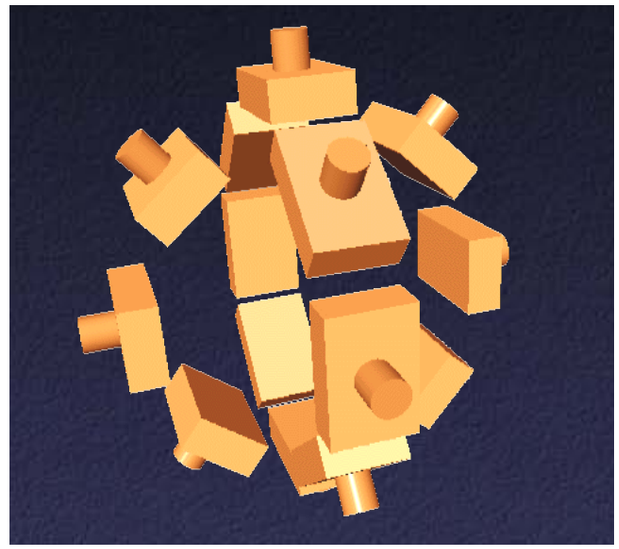
Noah Snavely
Google Inc.
snavely@google.com

Voxel coloring



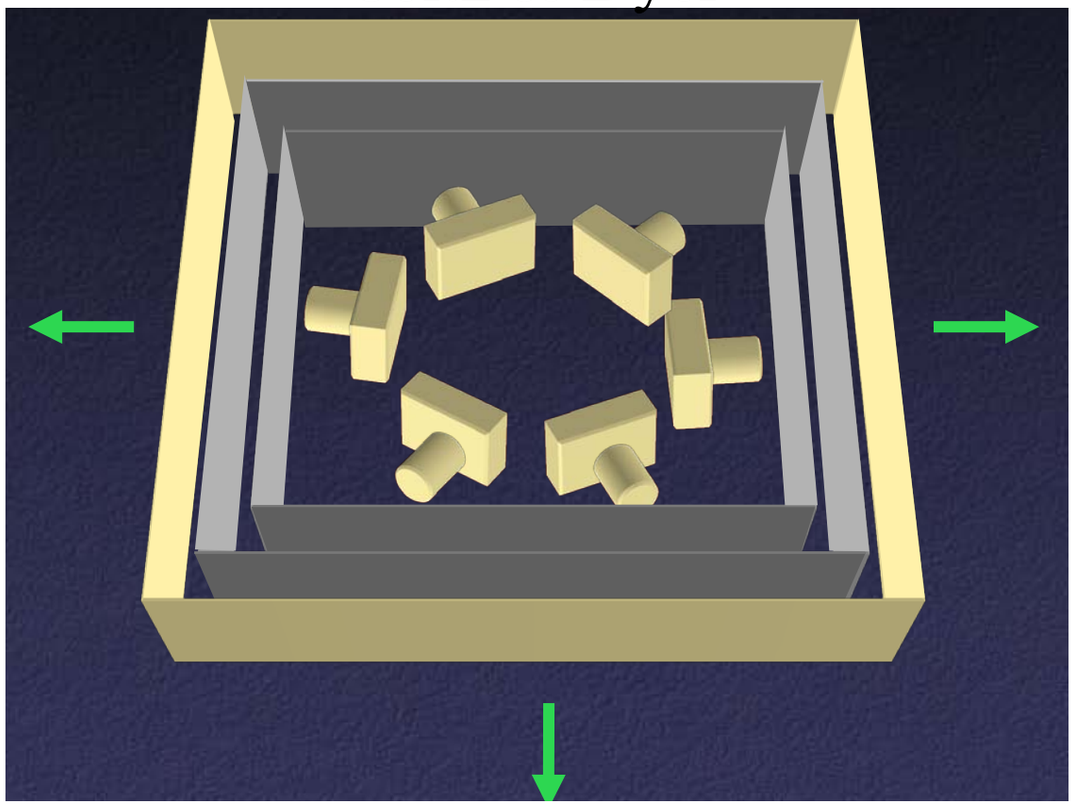
What about other camera steups?





Panoramic depth ordering

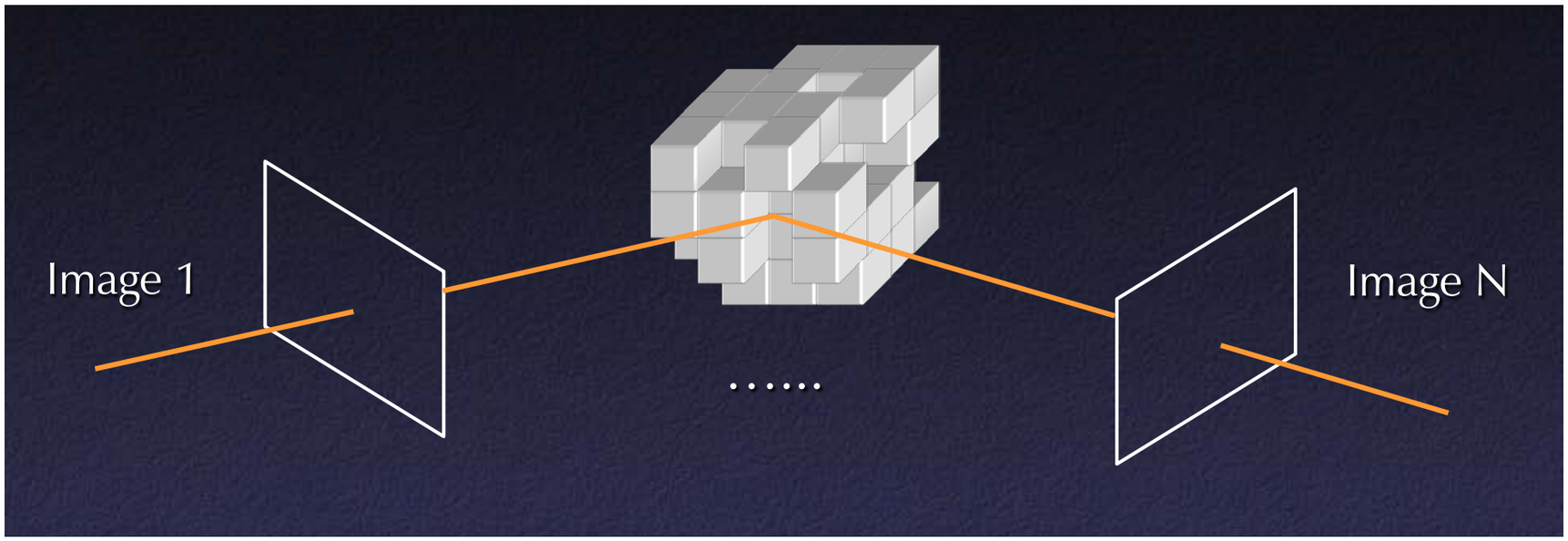
Seitz & Dyer



Layers radiate inwardly/outwardly

Space carving

Kutulakos & Seitz



Very simply 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

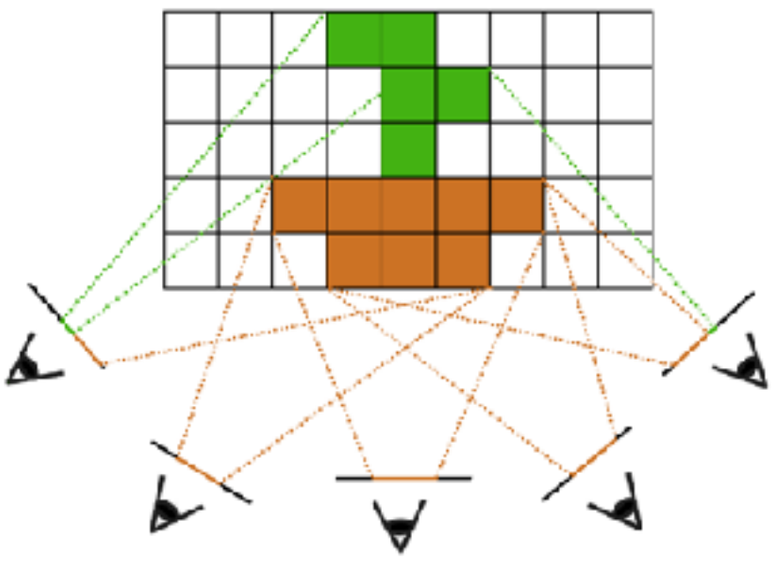
Convergence

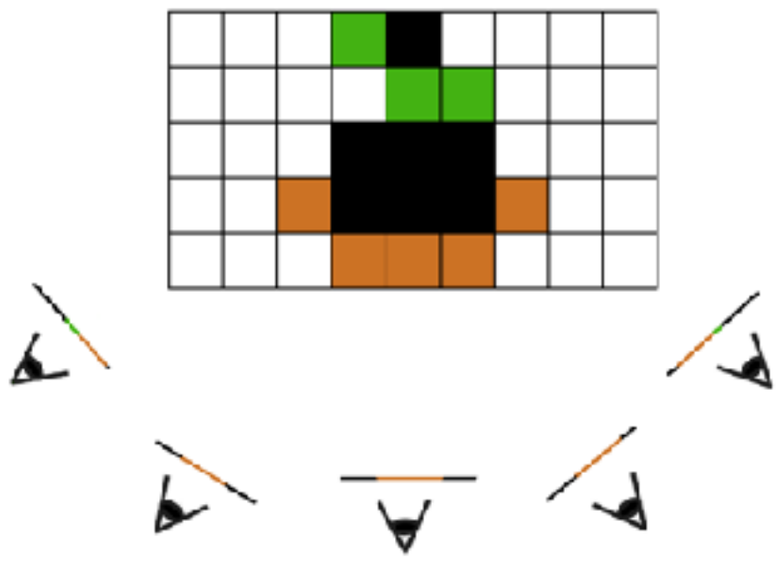
Consistency Property

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

Convergence Property

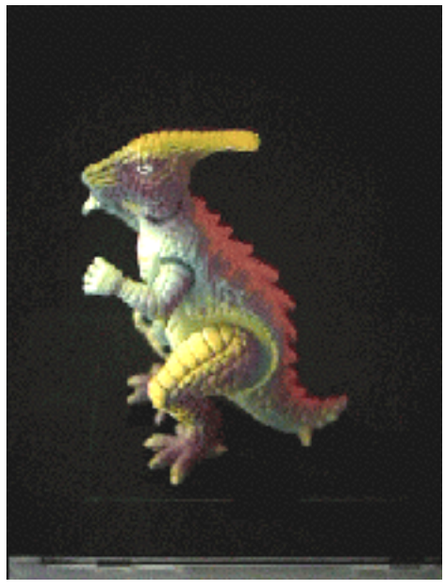
- Carving converges to a non-empty shape
  - > a point on the true scene is *never* removed





Calibrated Image Acquisition

Calibrated Turntable



Selected Dinosaur Images

Selected Flower Images

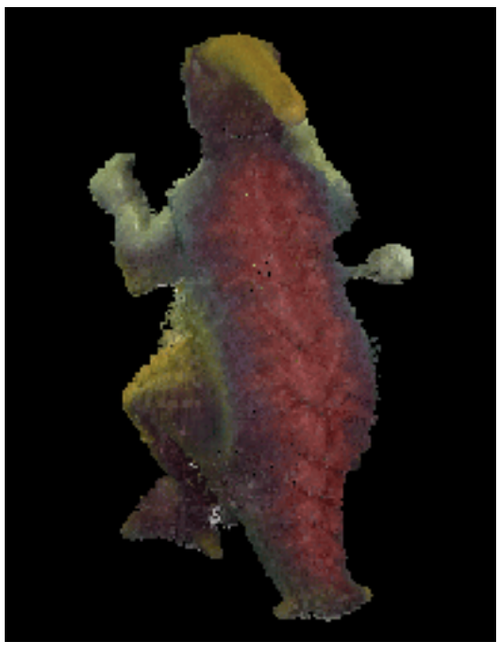
Voxel Coloring Results

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

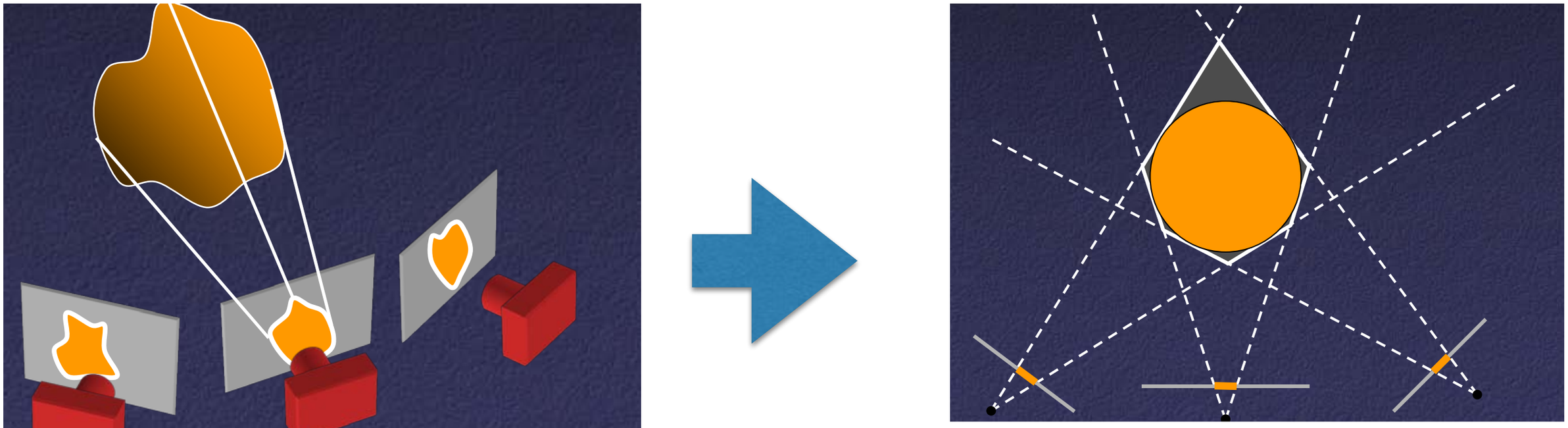
21 images



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

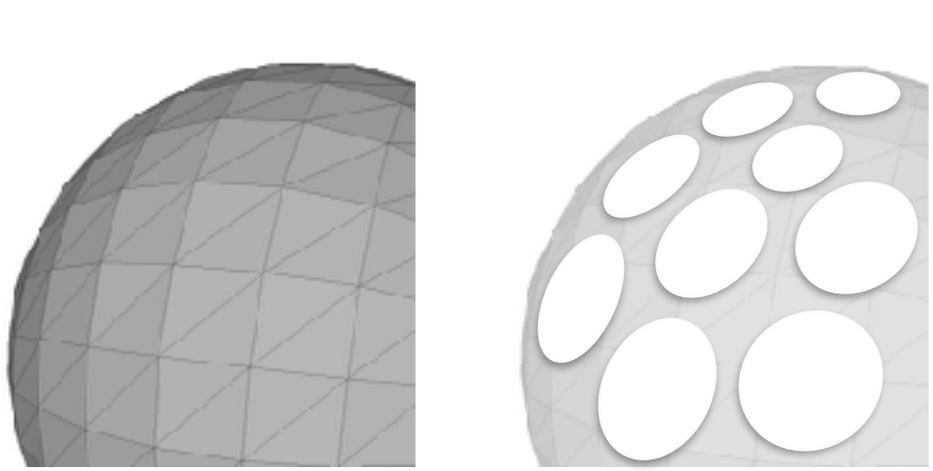
Multiview stereo

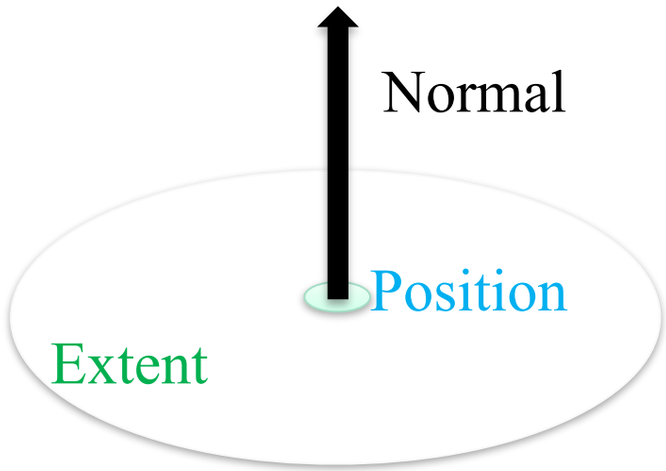
- stereo / dynamic programming
- active illumination
- volumetric models / visiblity reasoning
- patch-based methods

Long-standing leader

Accurate, Dense, and Robust Multi-View Stereopsis

Yasutaka Furukawa and Jean Ponce, *Fellow, IEEE*



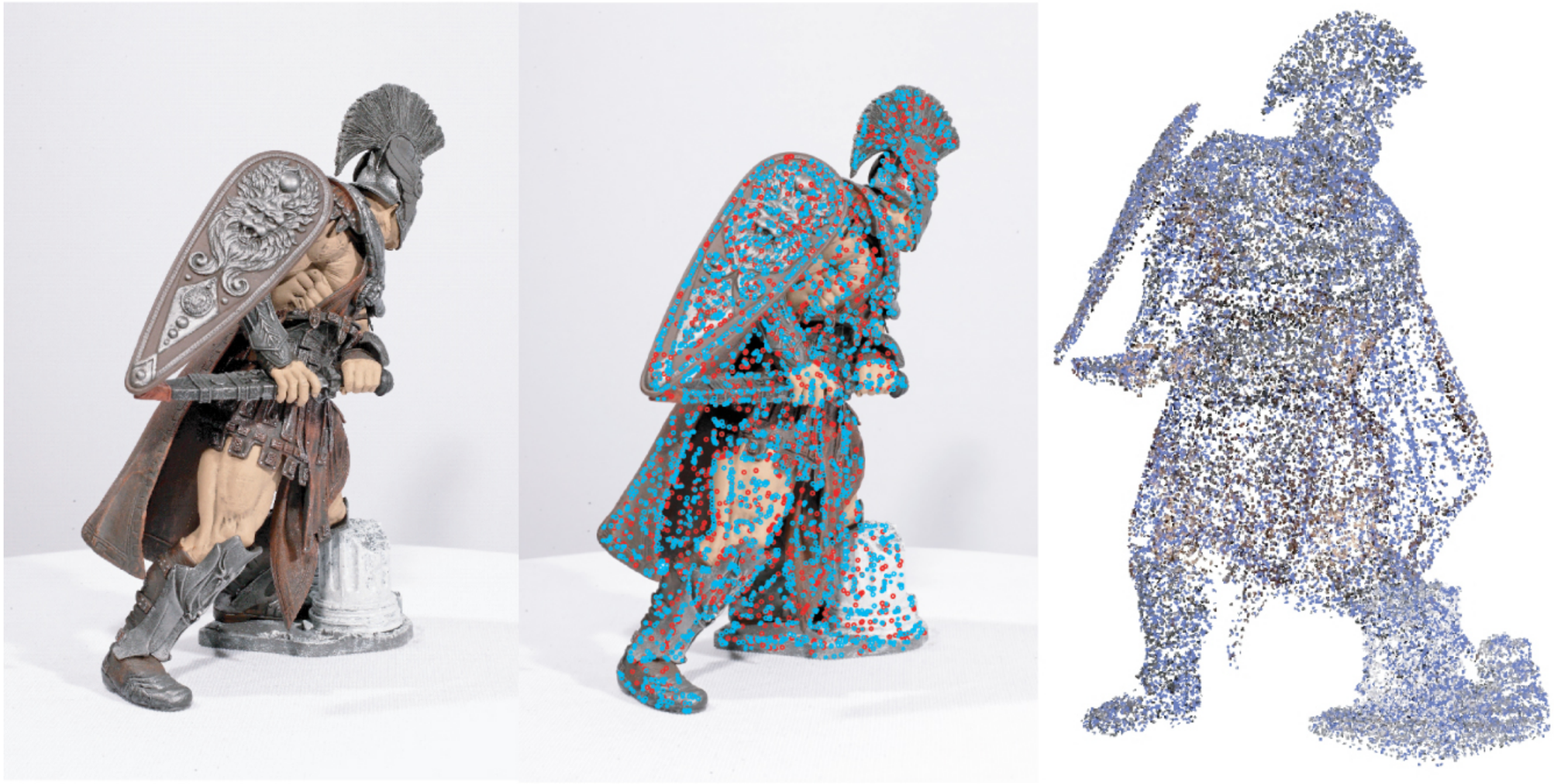


Patch-based Multiview Stereo (PMVS)

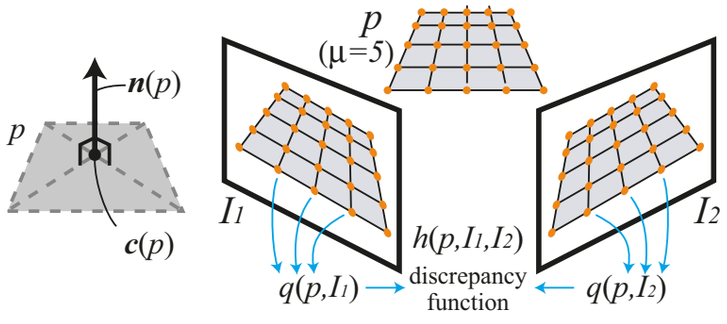
Pipeline: feature detection

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)



Pipeline: patch expansion

Expand set of points {p} by looking for hypothesizing 2D neighbors in visible images, backprojecting, and verifying photoconsistency

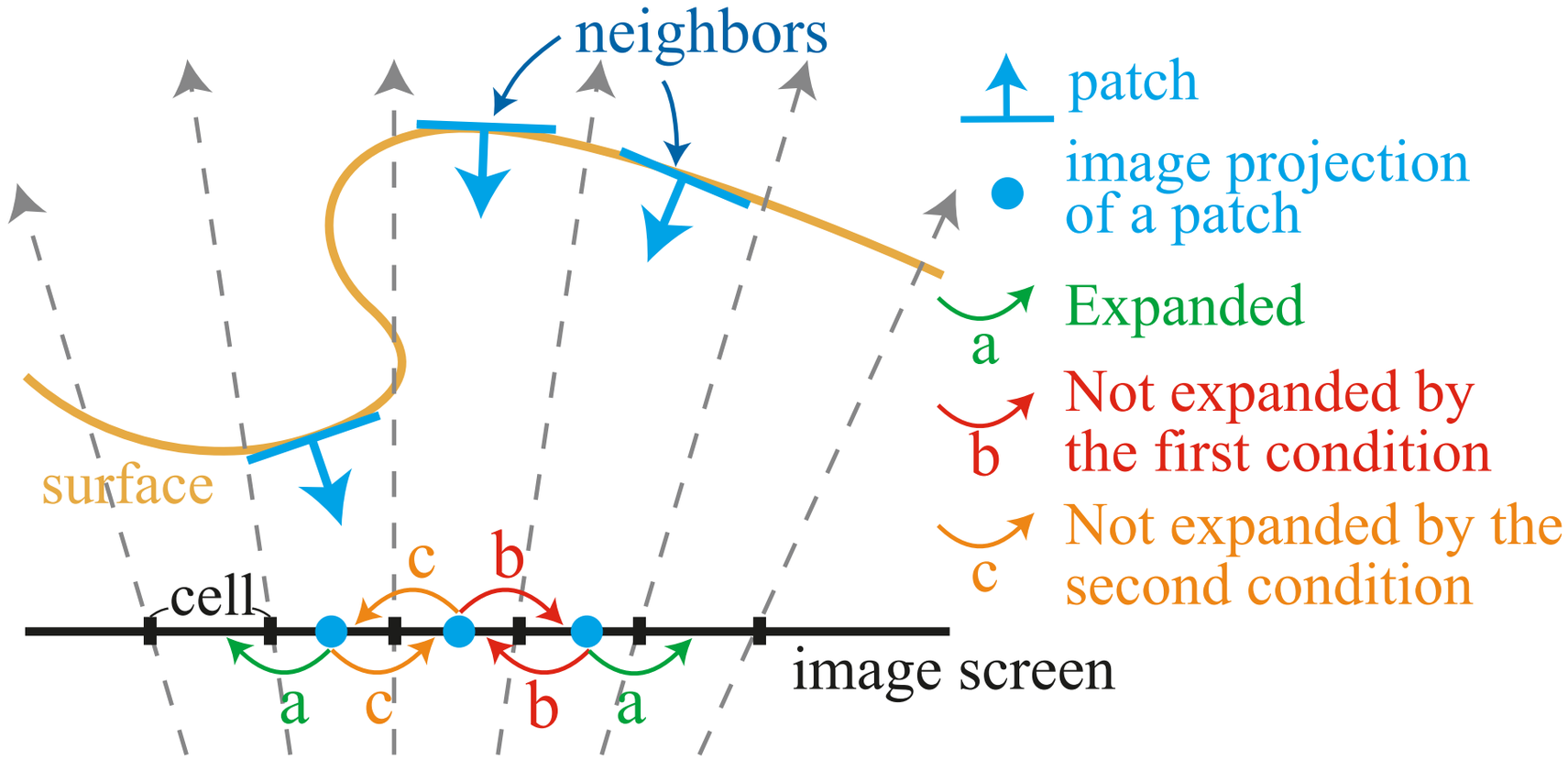


Fig. 5. (a) Given an existing patch, an expansion procedure is performed to generate new ones for the neighboring empty image cells in its visible images. The expansion procedure is not performed for an image cell (b) if there already exists a neighboring patch reconstructed there, or (c) if there is a depth discontinuity when viewed from the camera. See text for more details.

Pipeline: filter out outlier patches

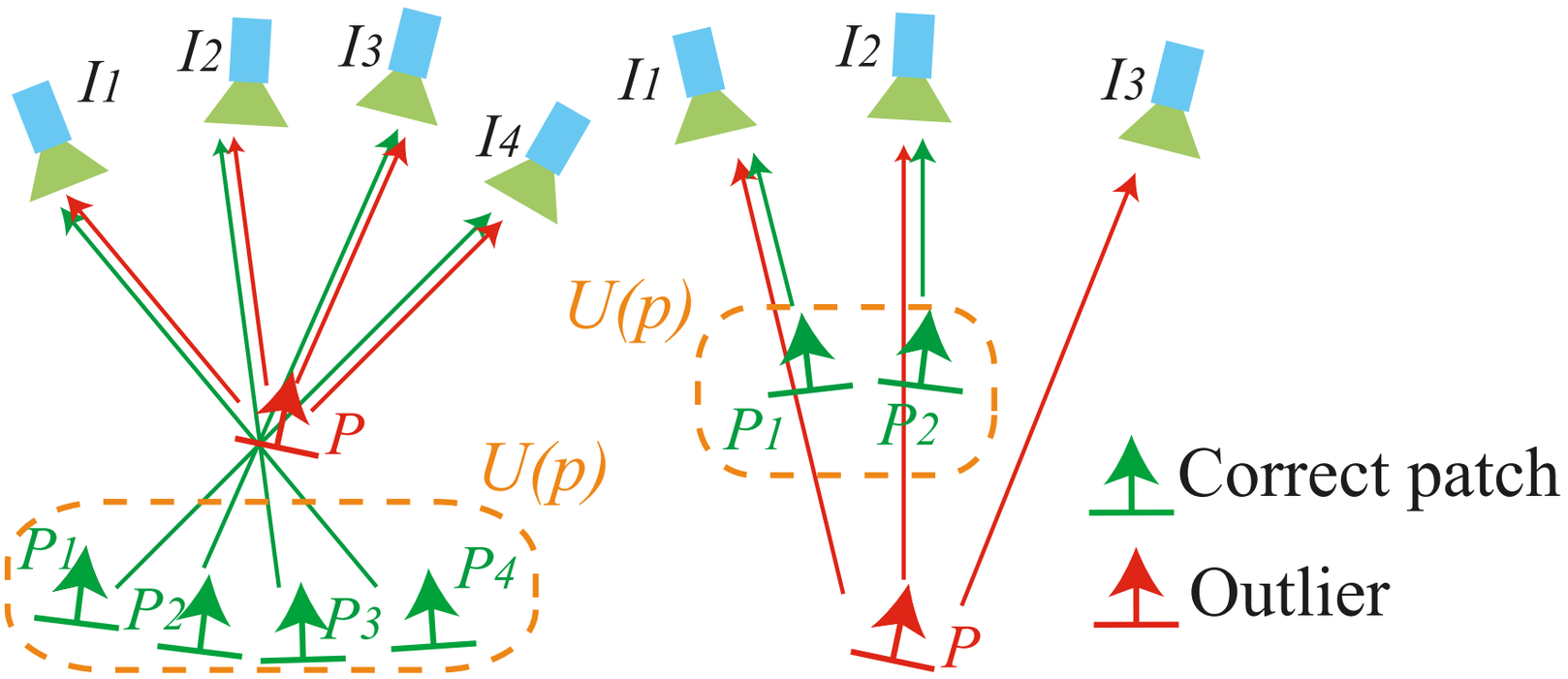
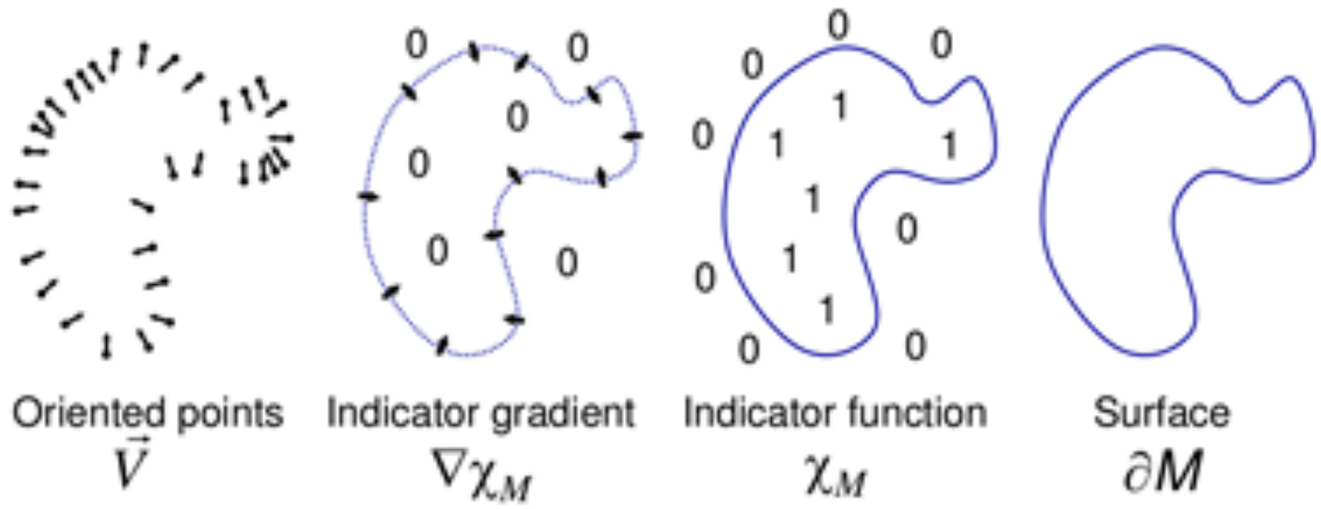


Fig. 7. The first filter enforces global visibility consistency to remove outliers (red patches). An arrow pointing from p_i to I_j represents a relationship $I_j \in V(p_i)$. In both cases (left and right), $U(p)$ denotes a set of patches that is inconsistent in visibility information with p.

Pipeline: construct mesh

Convert set of 3D patches (*surfel* model) into polygonal mesh



Represent surface implicitly using a volumetric signed distance function
Solve differential equation that equates gradients of function to normals

Results

Multiview stereo

- stereo / dynamic programming
- active illumination
- volumetric models / visiblity reasoning
- patch-based methods