CPSC 425: Computer Vision

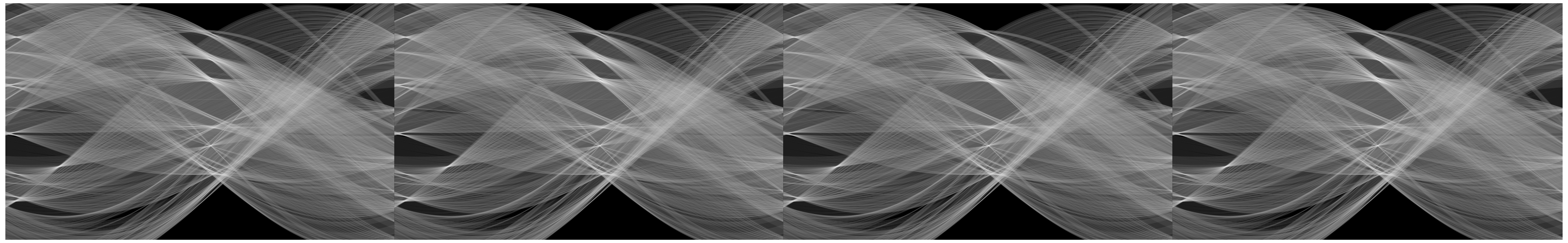


Image Credit: Ioannis (Yannis) Gkioulekas (CMU)

Lecture 22: Hough (cont)

Menu for Today (**November 2, 2020**)

Topics:

— Hough Transform for Object Detection

— Hough Transform for Segmentation, Depth estimation

— Stereo

Readings:

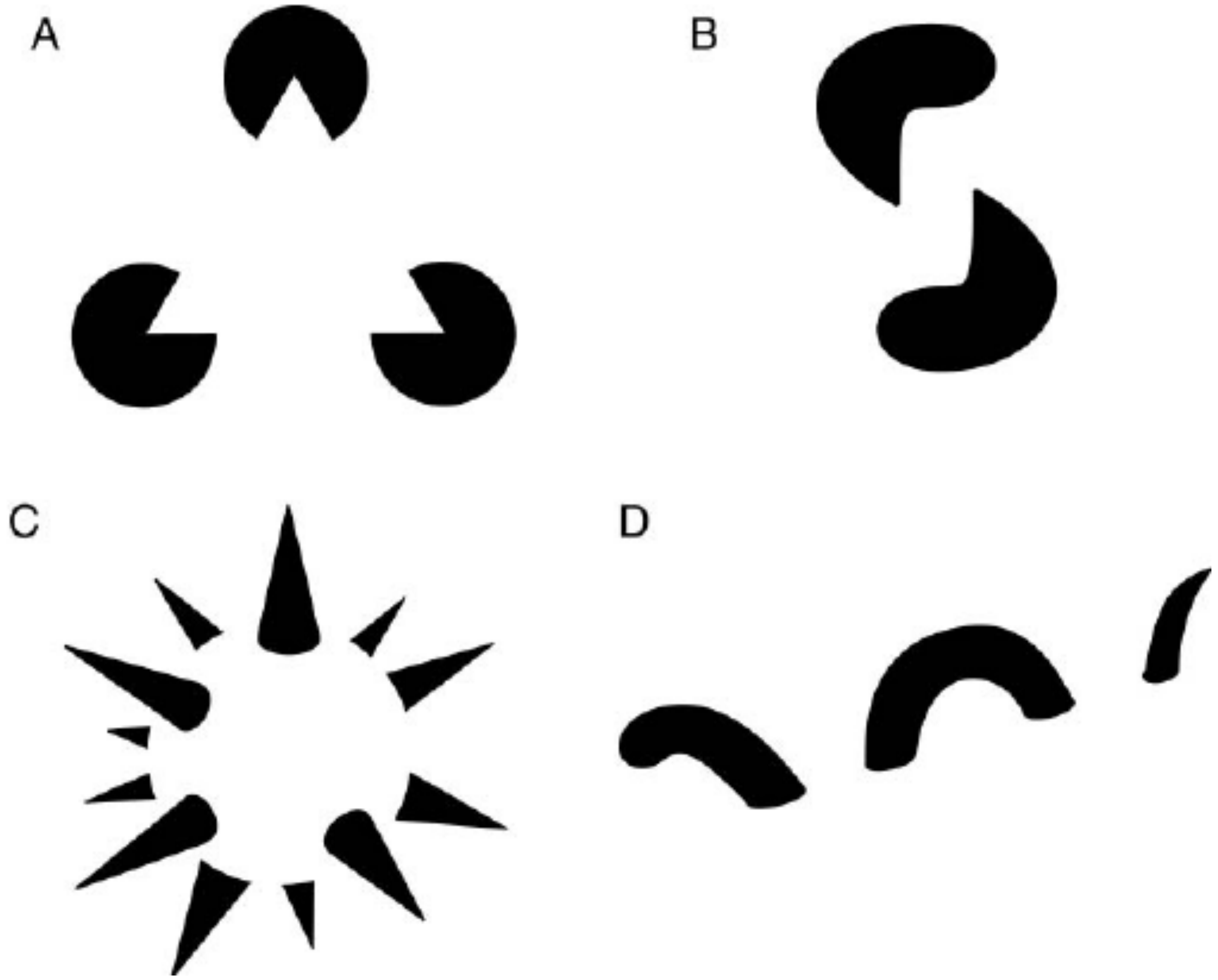
— **Today's** & **Next** Lecture: Forsyth & Ponce (2nd ed.) 7.1.1, 7.2.1, 7.4, 7.6

Reminders:

— **Assignment 4** is due on Friday

— Quiz next **Monday**

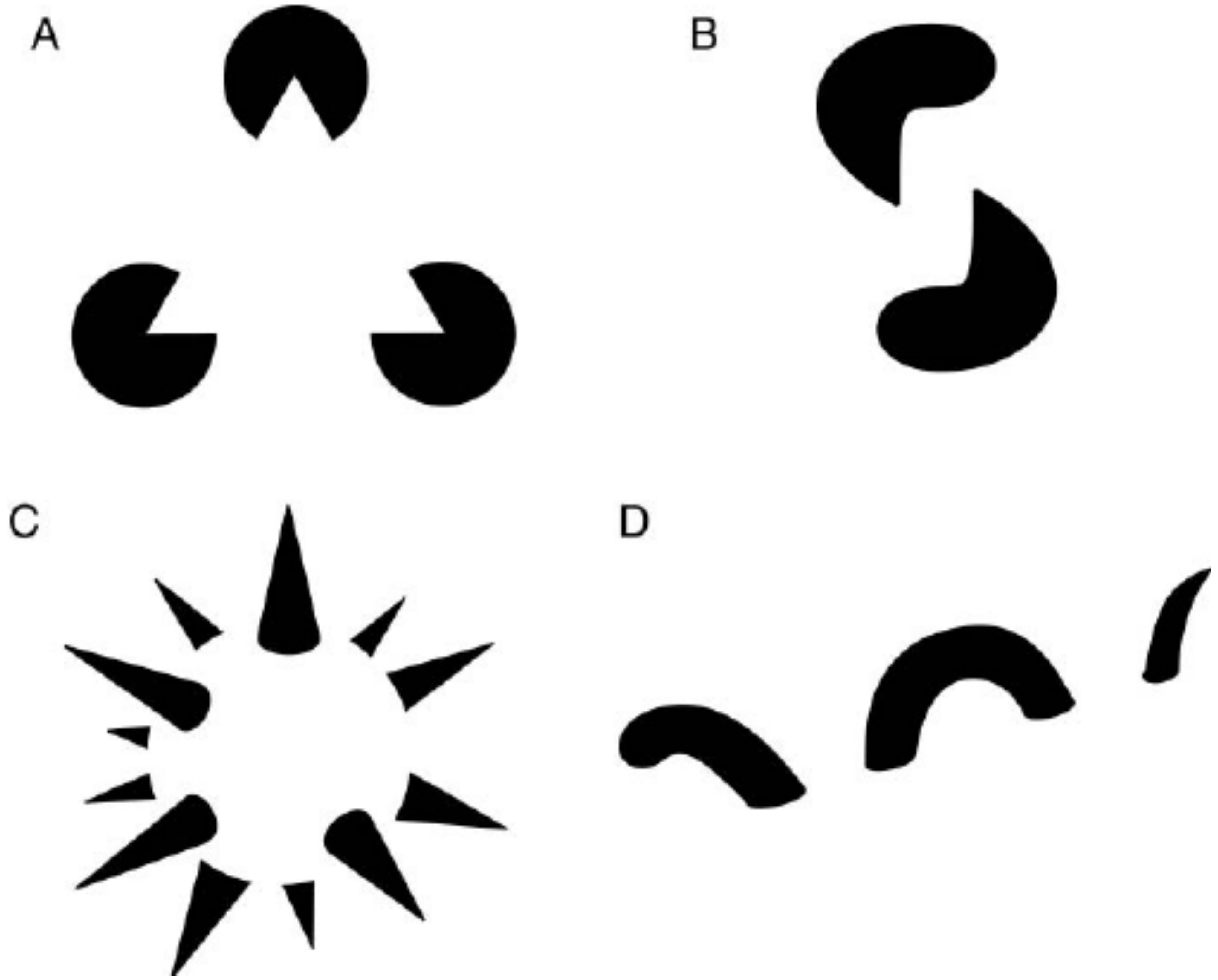
Today's "**fun**" Example: Tse's Volumetric Illusions



A. Kanizsa triangle
B. Tse's volumetric worm
C. Idesawa's spiky sphere
D. Tse's "sea monster"

Figure credit: Steve Lehar

Today's "**fun**" Example: Tse's Volumetric Illusions



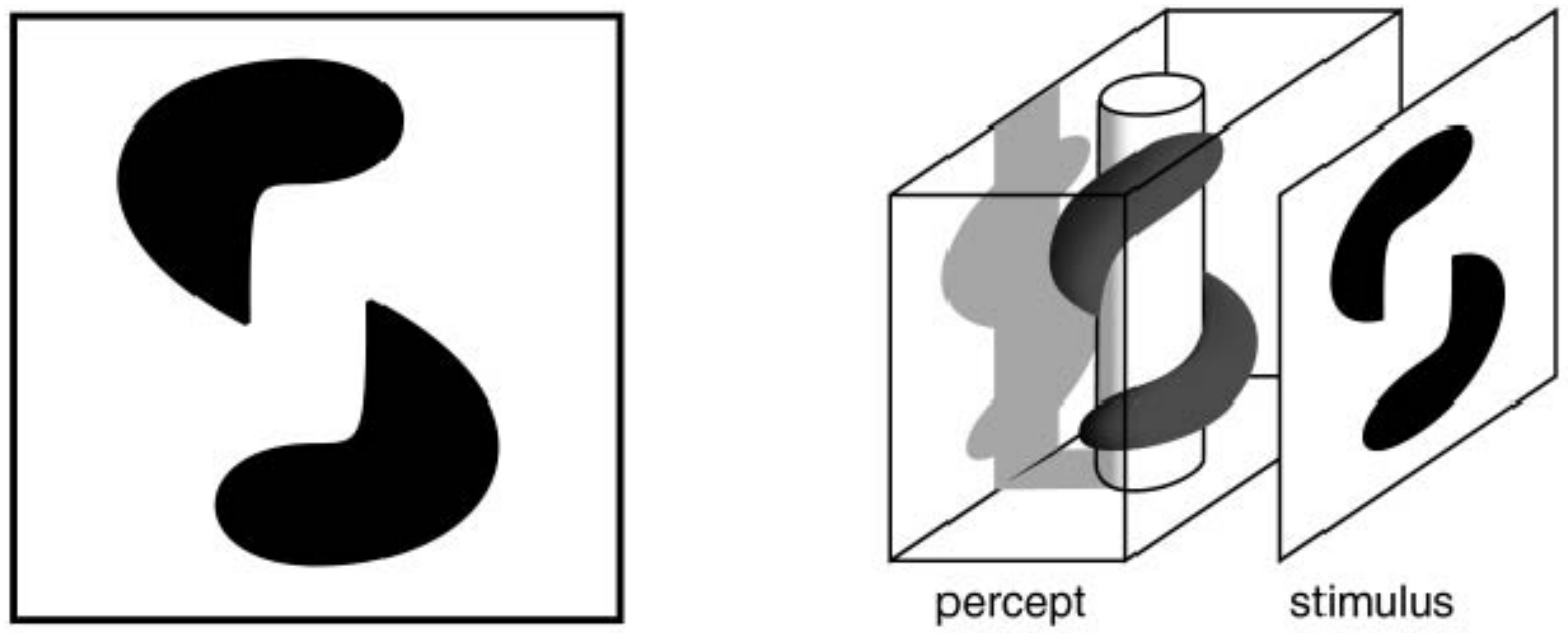


Figure credit: Steve Lehar

Today’s “**fun**” Example: FedEx

Lecture 21: Re-cap Hough Transform

The **Hough transform** is another technique for fitting data to a model

— a voting procedure

— possible model parameters define a quantized accumulator array

— data points “vote" for compatible entries in the accumulator array

A key is to have each data point (token) constrain model parameters as tightly as possible

Lecture 21: Re-cap Hough Transform

Advantages:

— Can handle high percentage of outliers: each point votes separately

— Can detect multiple instances of a model in a single pass

Disadvantages:

— Complexity of search time increases exponentially with the number of model parameters

— Can be tricky to pick a good bin size

Lecture 21: Re-cap Mechanics of Hough Transform

1. Construct a quantized array to represent θ and r

2. For each point, render curve (θ, r) into this array adding one vote at each cell

Difficulties:
— How big should the cells be? (too big, and we merge quite different lines; too small, and noise causes lines to be missed)

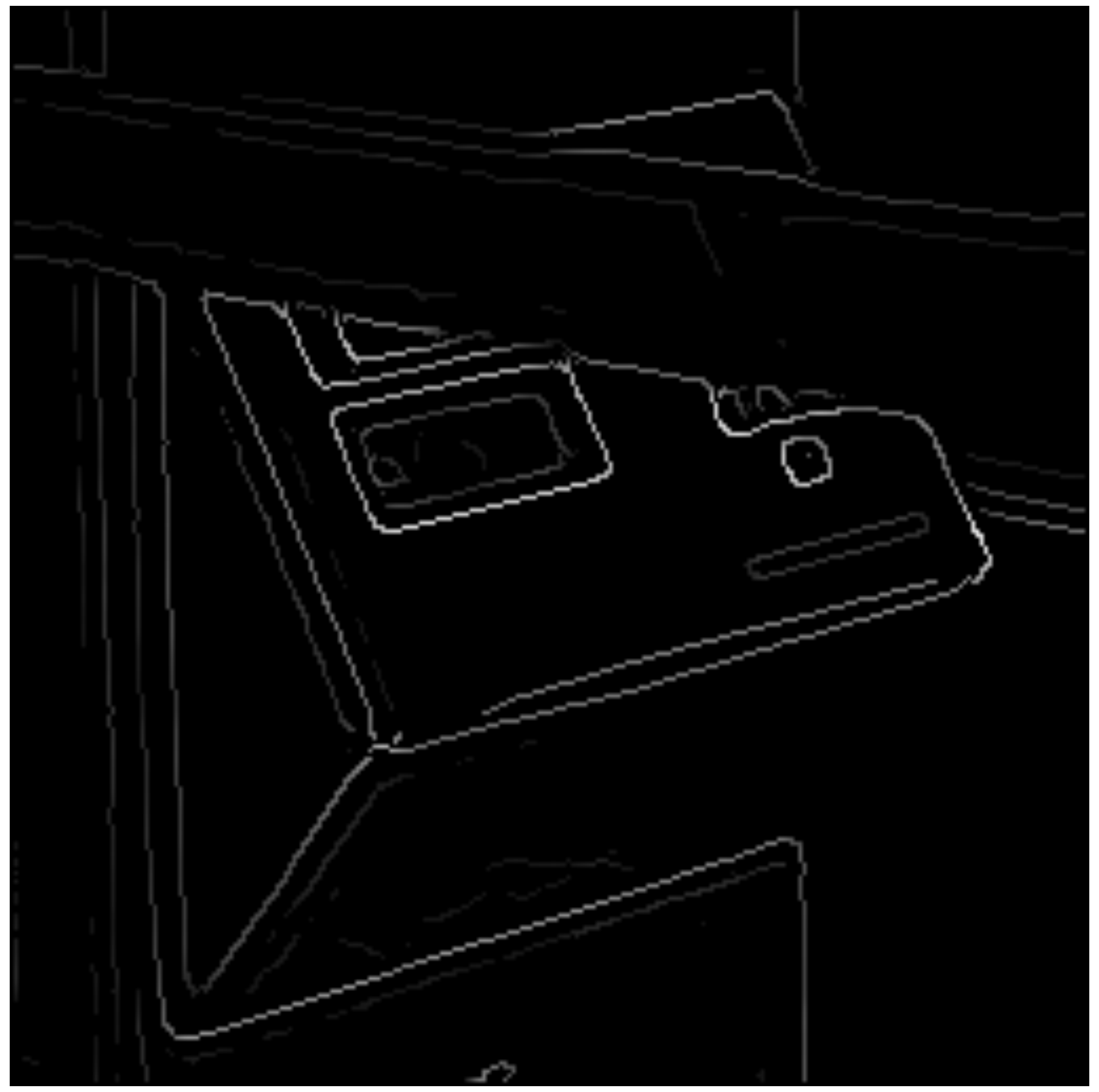
How many lines?

— Count the peaks in the Hough array

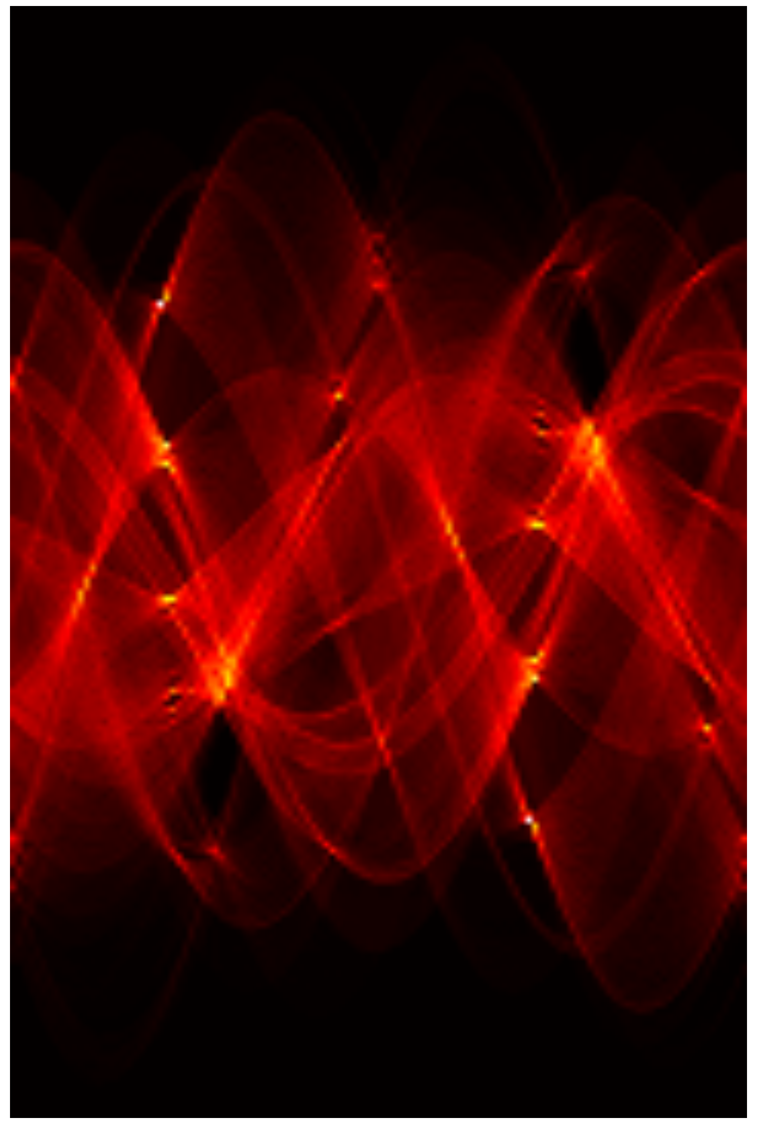
— Treat adjacent peaks as a single peak

Lecture 21: Re-cap Hough Transform

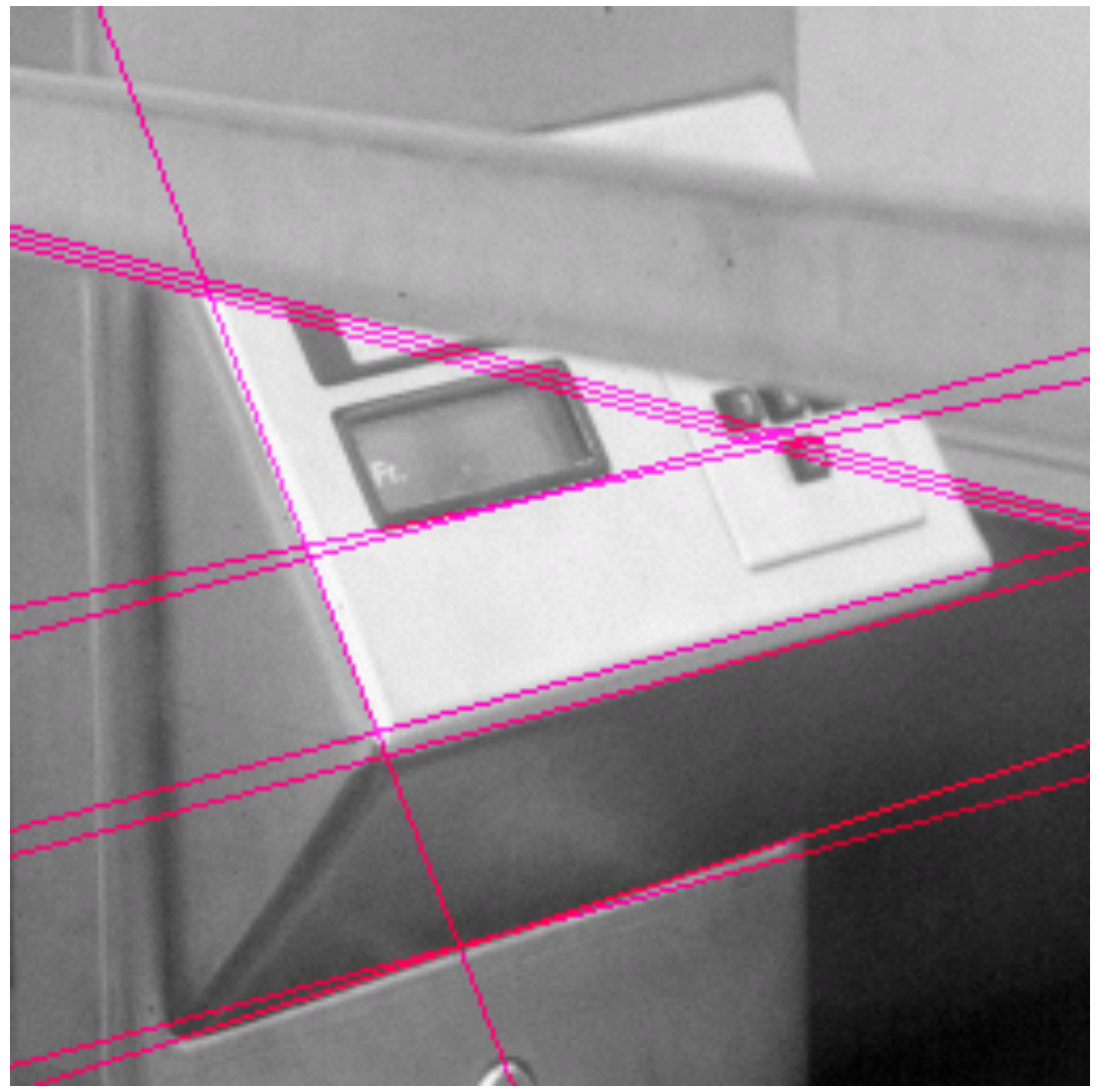
Original



Edges



Parameter space



Hough Lines

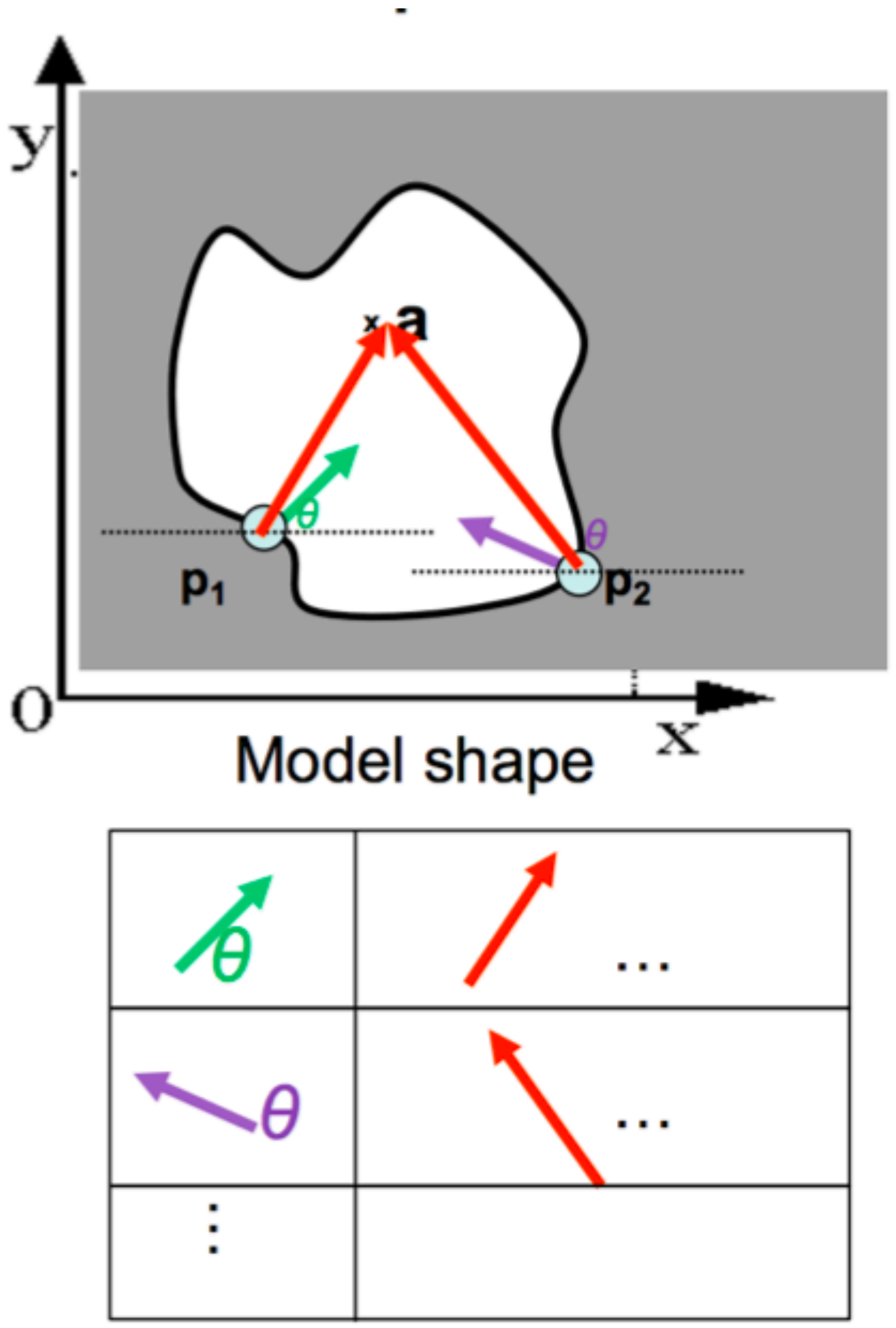
Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Generalized Hough Transform

What if we want to detect an **arbitrary** geometric shape?

Generalized Hough Transform

What if we want to detect an **arbitrary** geometric shape?



Offline procedure:

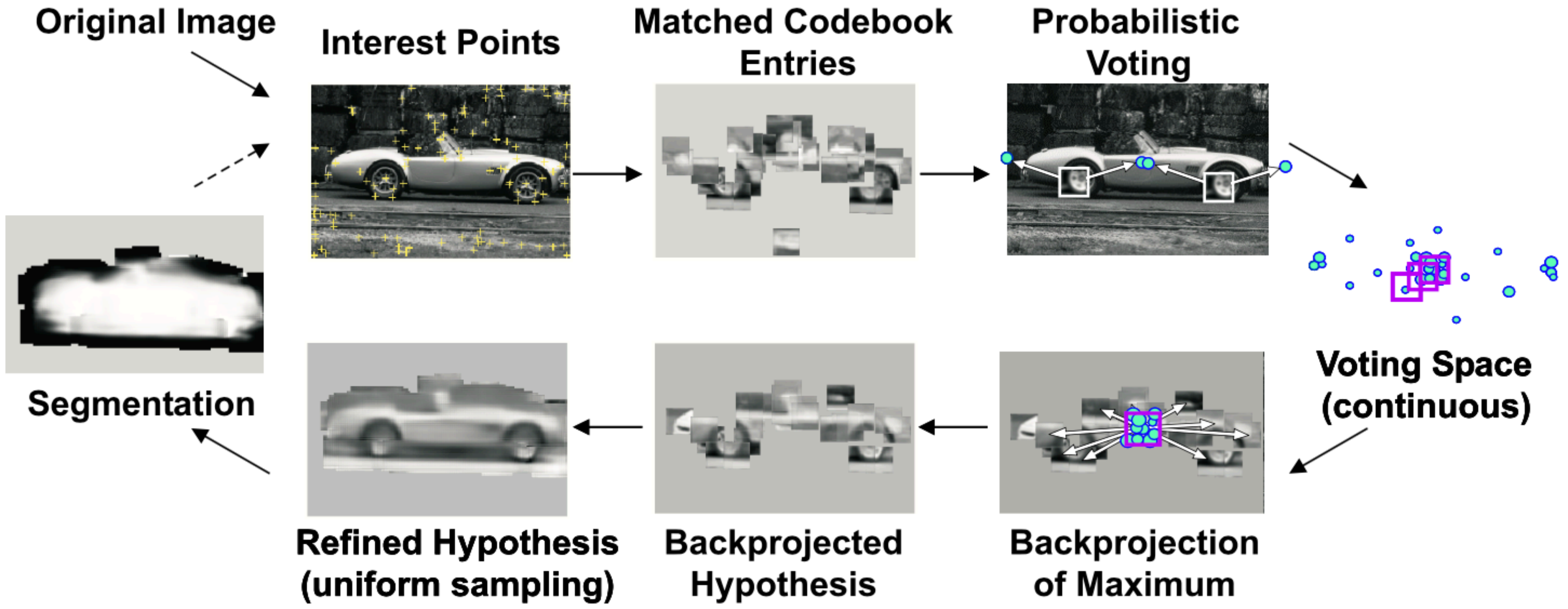
At each boundary point, compute displacement vector: **r** = **a** – $\mathbf{p}_i$.

Store these vectors in a table indexed by gradient orientation θ.

Dana H. Ballard, Generalizing the Hough Transform to Detect Arbitrary Shapes, 1980

Example 1: Object Recognition — Implicit Shape Model

Combined object detection and segmentation using an implicit shape model.
Image patches cast weighted votes for the object centroid.



B. Leibe, A. Leonardis, and B. Schiele, Combined Object Categorization and Segmentation with an Implicit Shape Model,
ECCV Workshop on Statistical Learning in Computer Vision 2004

Example 1: Object Recognition — Implicit Shape Model

Basic Idea:

— Find **interest points/keypoints** in an image (e.g., SIFT Keypoint detector or Corners)

— **Match patch** around each interest point to a training patch (e.g., SIFT Descriptor)

— **Vote** for object center given that training instances

— Find the patches that voted for the peaks (**back-project**)

Example 1: Object Recognition — Implicit Shape Model

"**Training**" images of cows

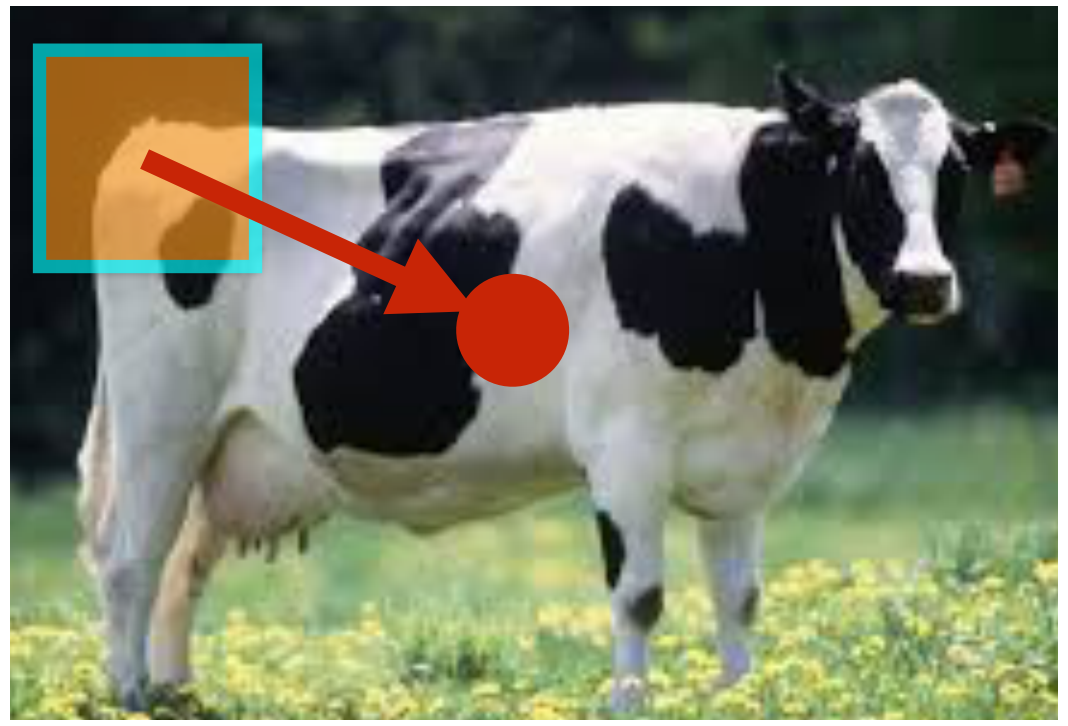


Image Index	**Keypoint Index**	**Keypoint Detection (4D)**	**Keypoint Description (128D)**	**Offset to Centroid**
Image 1	1	[x, y, s, Theta]	[…]	[x,y]

* Slide from Sanja Fidler

Example 1: Object Recognition — Implicit Shape Model

"**Training**" images of cows

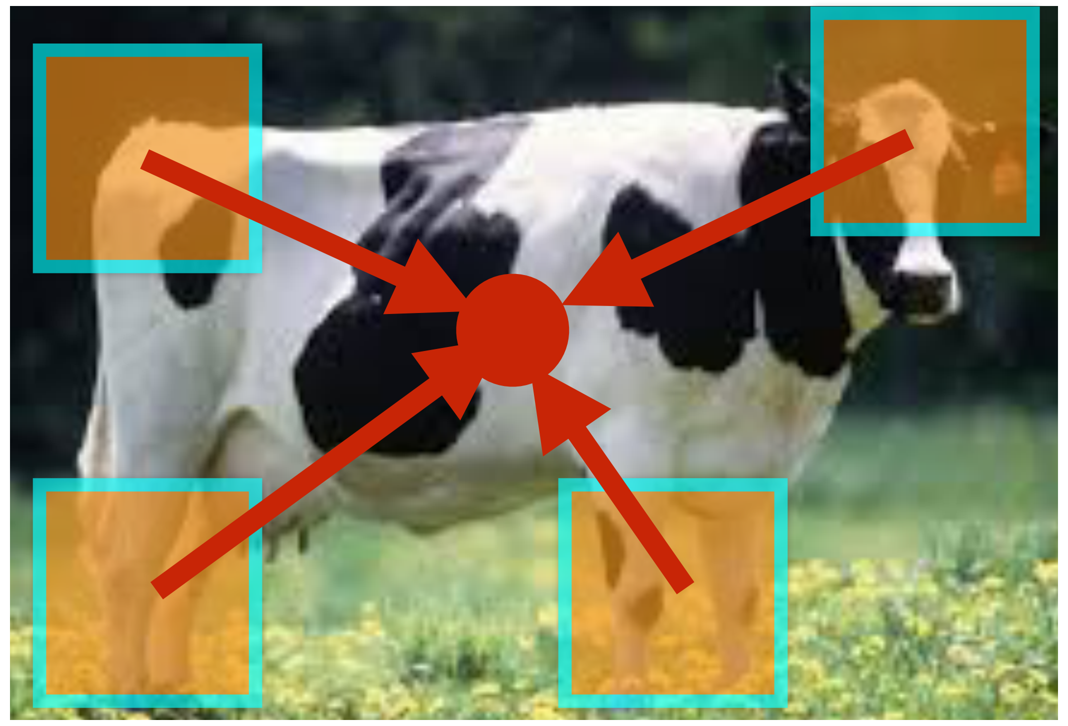
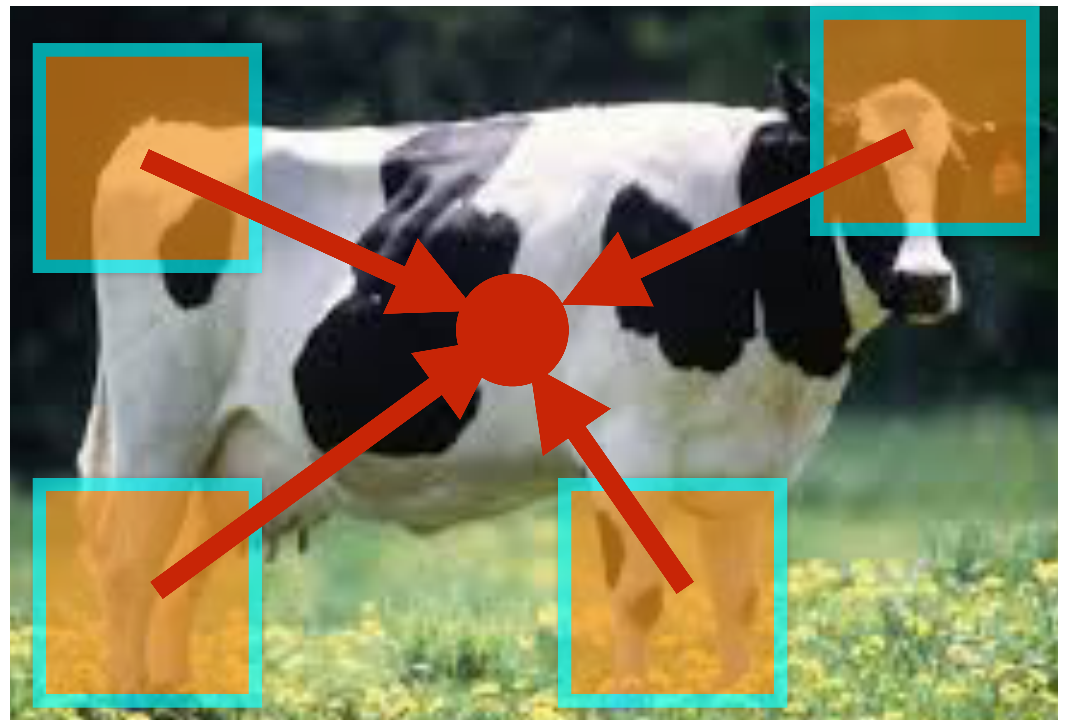


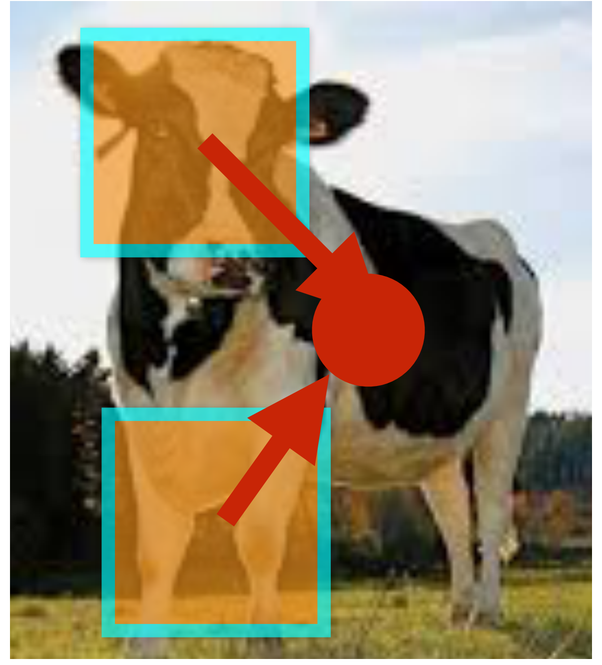
Image Index	Keypoint Index	Keypoint Detection (4D)	Keypoint Description (128D)	Offset to Centroid
Image 1	1	[x, y, s, Theta]	[...]	[x,y]
Image 1	2	[x, y, s, Theta]	[...]	[x,y]
....	....	....	...	...
Image 1	265	[x, y, s, Theta]	[...]	[x,y]

* Slide from Sanja Fidler

Example 1: Object Recognition — Implicit Shape Model

"**Training**" images of cows





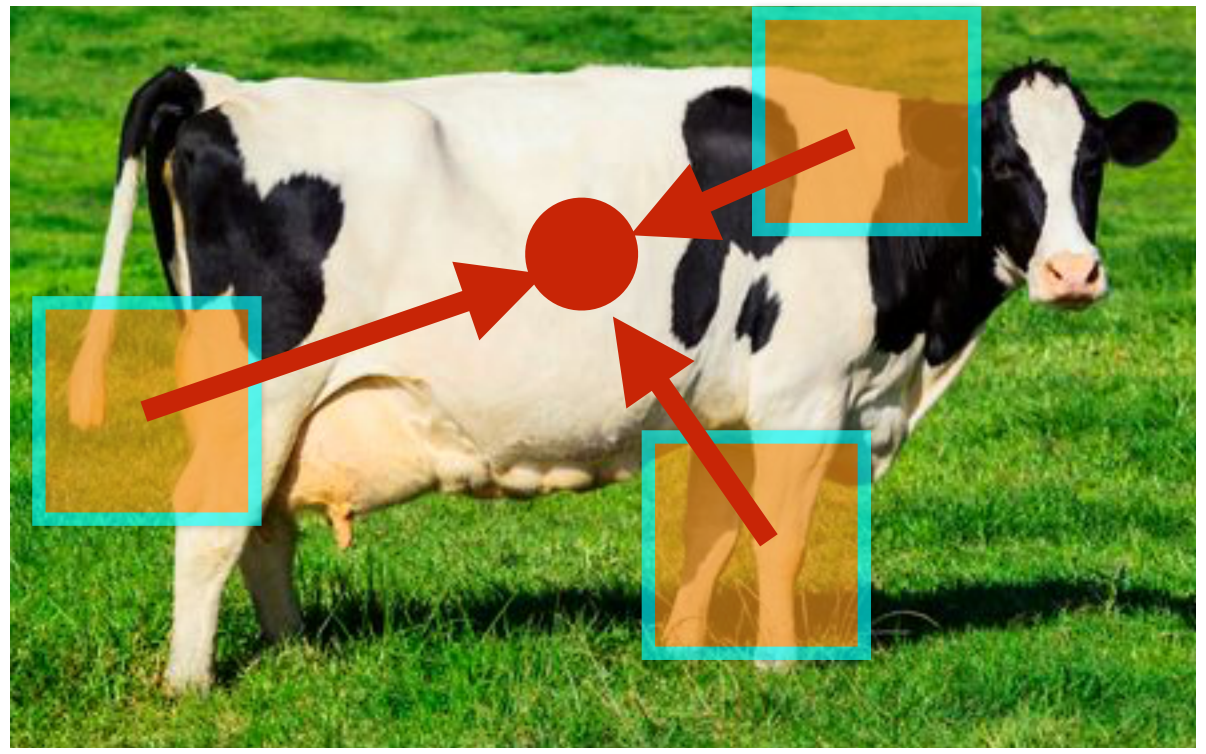


Image Index	**Keypoint Index**	**Keypoint Detection (4D)**	**Keypoint Description (128D)**	**Offset to Centroid**
Image 1	1	[x, y, s, Theta]	[…]	[x,y]
Image 1	2	[x, y, s, Theta]	[…]	[x,y]
….	….	….	…	…
Image 1	265	[x, y, s, Theta]	[…]	[x,y]
Image 2	1	[x, y, s, Theta]	[…]	[x,y]
Image 2	2	[x, y, s, Theta]	[…]	[x,y]
…	…	…	…	…
Image 2	645	[x, y, s, Theta]	[…]	[x,y]
Image K	1	[x, y, s, Theta]	[…]	[x,y]
Image K	2	[x, y, s, Theta]	[…]	[x,y]
…	…	…	…	…
Image K	134	[x, y, s, Theta]	[…]	[x,y]

* Slide from Sanja Fidler

Example 1: Object Recognition — Implicit Shape Model

“**Training**” images of cows

“**Testing**” image

* Slide from Sanja Fidler

Example 1: Object Recognition — Implicit Shape Model

“**Training**” images of cows

“**Testing**” image

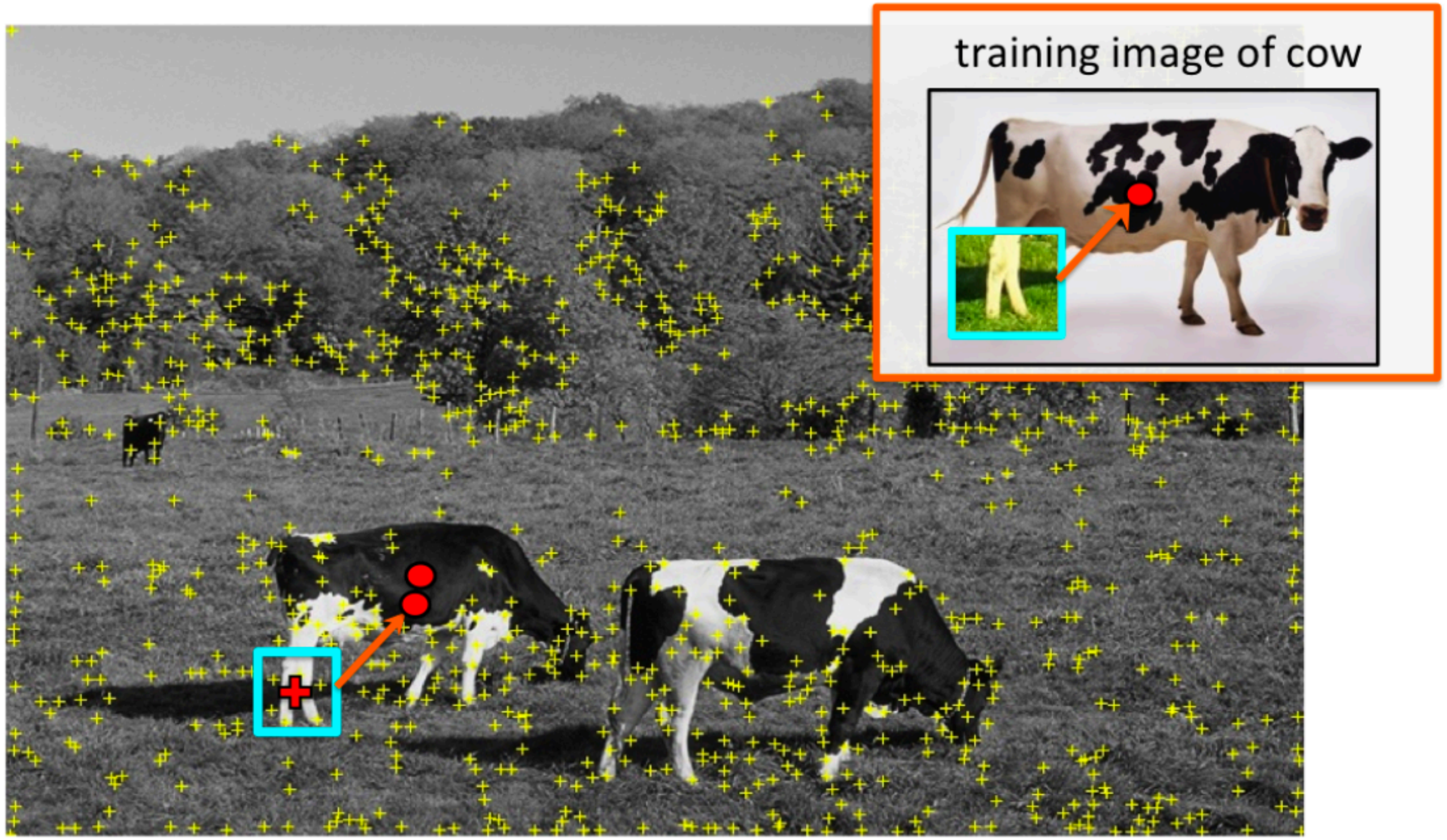
Vote for center of object

* Slide from Sanja Fidler

Example 1: Object Recognition — Implicit Shape Model

“**Training**” images of cows

“**Testing**” image



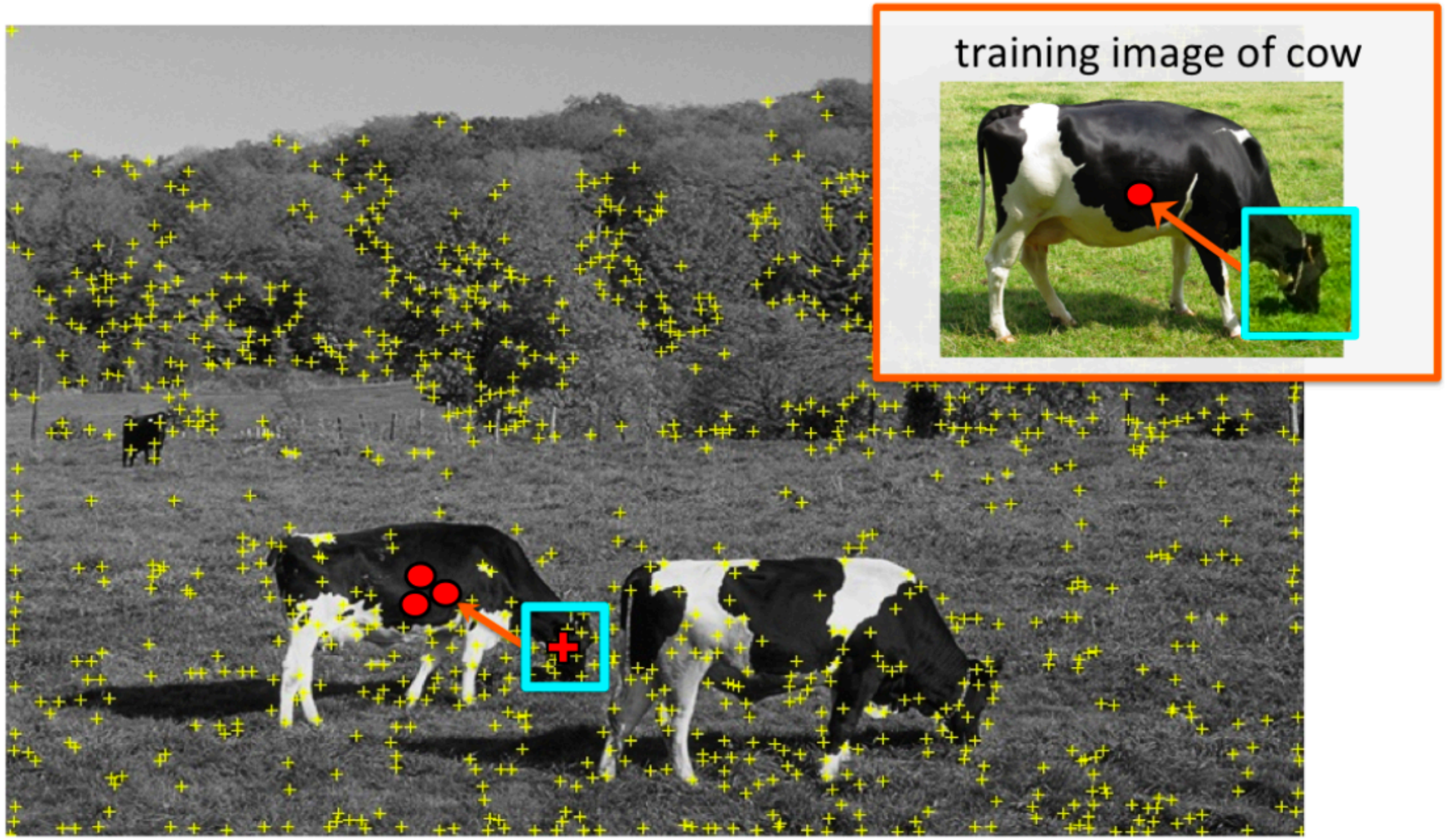
Vote for center of object

* Slide from Sanja Fidler

Example 1: Object Recognition — Implicit Shape Model

“**Training**” images of cows

“**Testing**” image



Vote for center of object

* Slide from Sanja Fidler

Example 1: Object Recognition — Implicit Shape Model

“**Training**” images of cows

“**Testing**” image

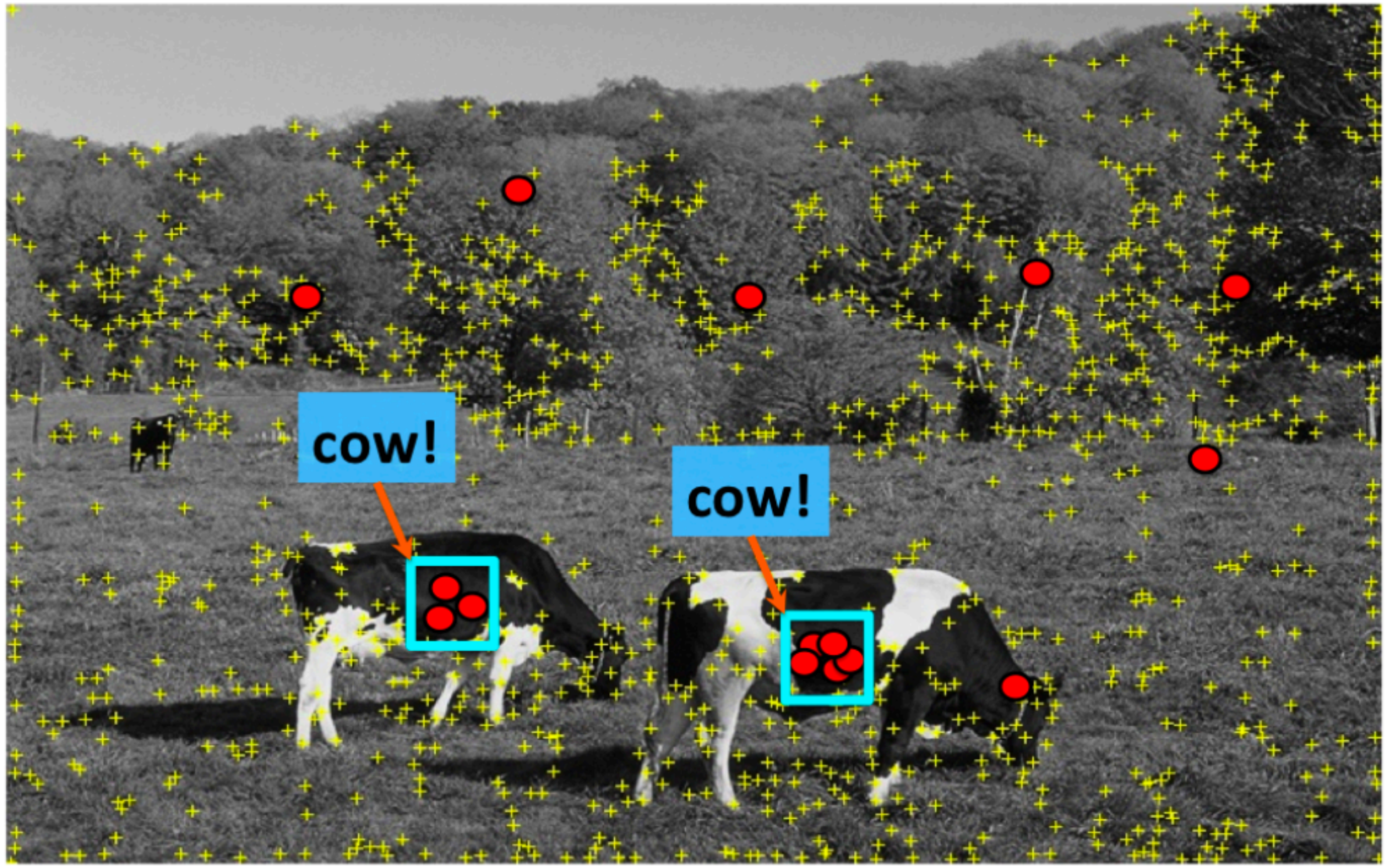
of course sometimes wrong votes are bound to happen

* Slide from Sanja Fidler

Example 1: Object Recognition — Implicit Shape Model

"**Training**" images of cows

"**Testing**" image



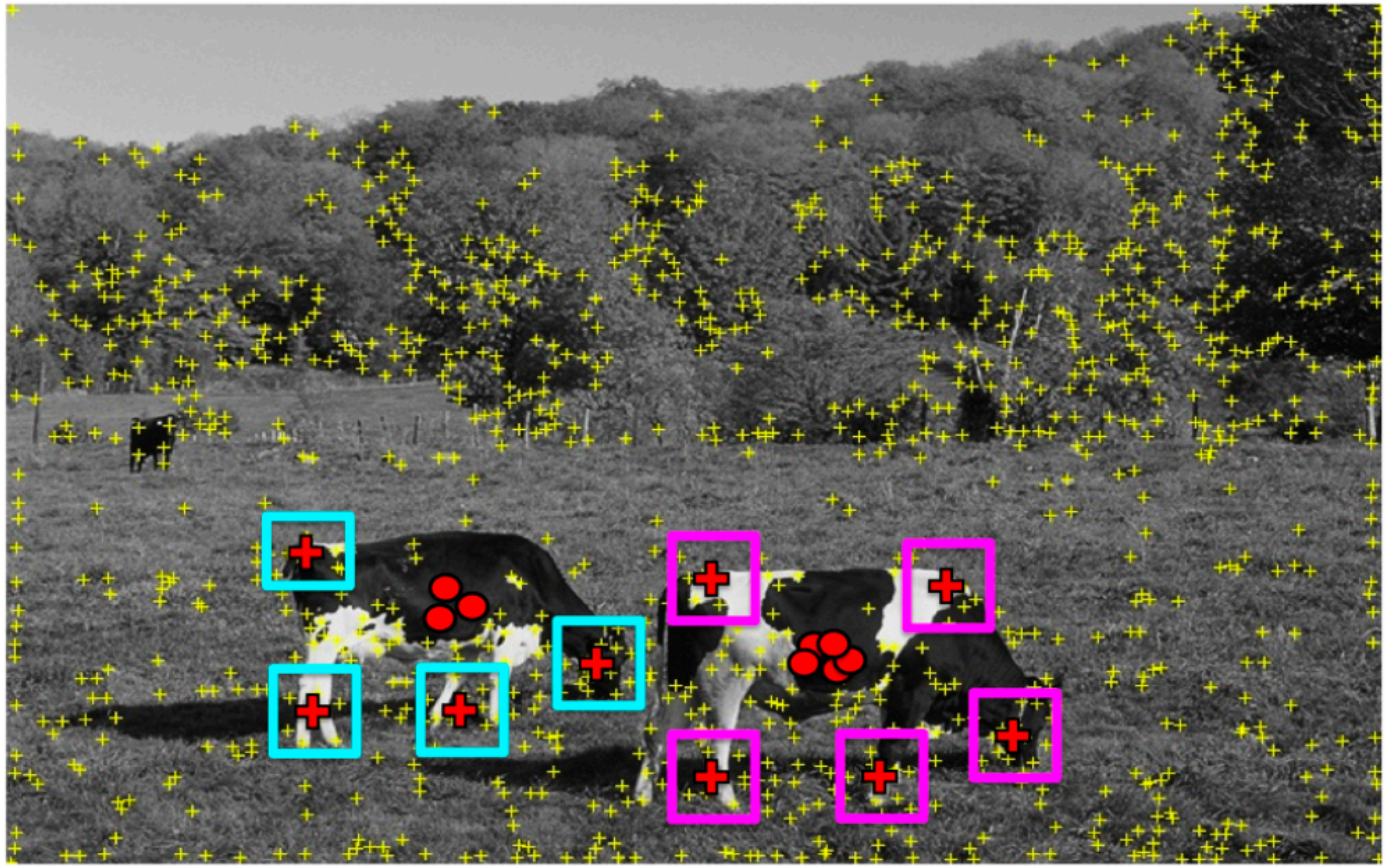
That's ok. We want only **peaks** in voting space.

* Slide from Sanja Fidler

Example 1: Object Recognition — Implicit Shape Model

“**Training**” images of cows

“**Testing**” image

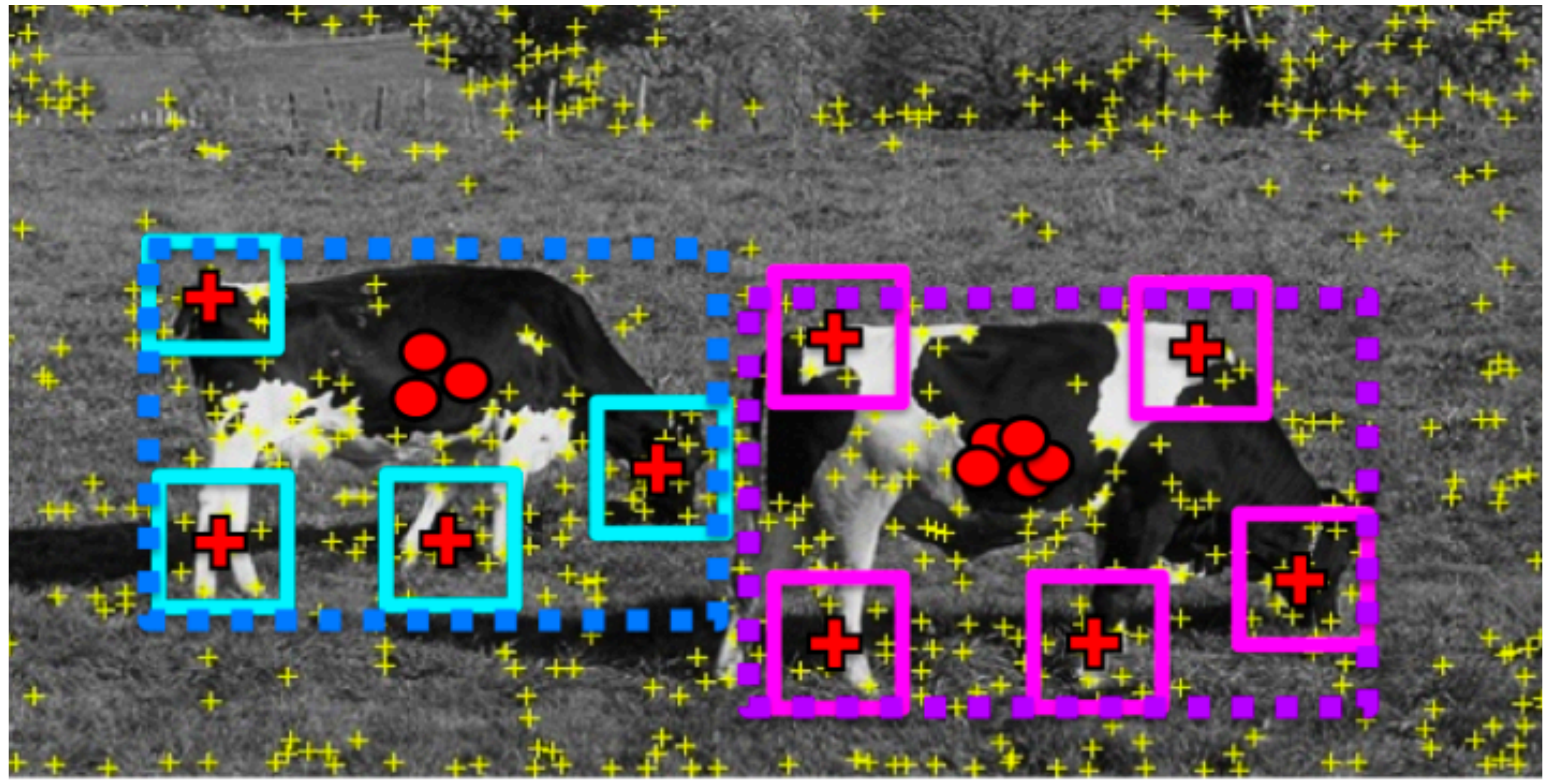


Find patches that voted for the peaks (back-project)

* Slide from Sanja Fidler

Example 1: Object Recognition — Implicit Shape Model

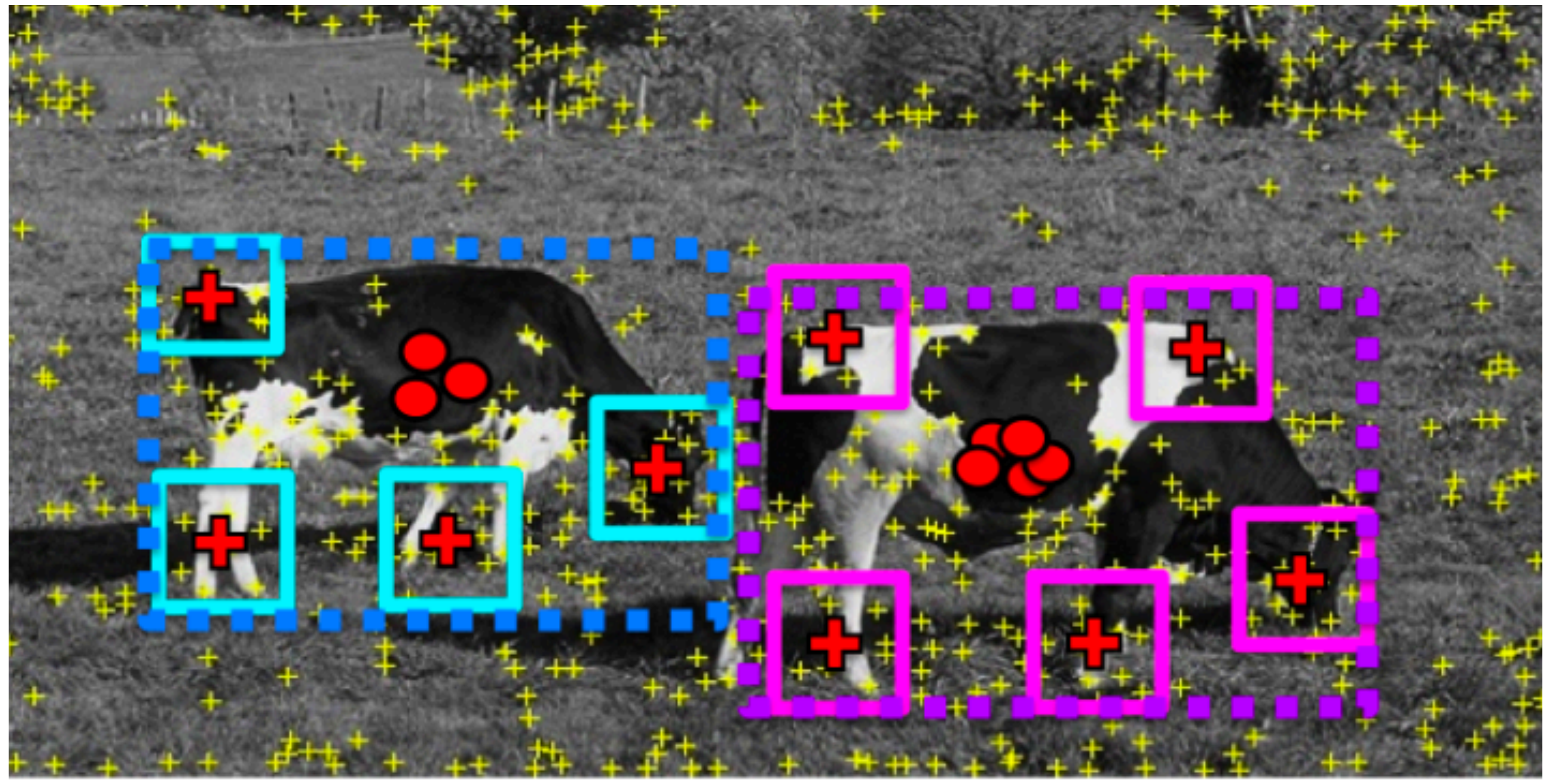
Image Index	Keypoint Index	Keypoint Detection (4D)	Keypoint Description (128D)	Offset to Centroid
Image 1	1	[x, y, s, Theta]	[…]	[x,y]
Image 1	2	[x, y, s, Theta]	[…]	[x,y]
….	….	….	…	…
Image 1	265	[x, y, s, Theta]	[…]	[x,y]
Image 2	1	[x, y, s, Theta]	[…]	[x,y]
Image 2	2	[x, y, s, Theta]	[…]	[x,y]
…	…	…	…	…
Image 2	645	[x, y, s, Theta]	[…]	[x,y]
Image K	1	[x, y, s, Theta]	[…]	[x,y]
Image K	2	[x, y, s, Theta]	[…]	[x,y]
…	…	…	…	…
Image K	134	[x, y, s, Theta]	[…]	[x,y]



* Slide from Sanja Fidler

Example 1: Object Recognition — Implicit Shape Model

Image Index	Keypoint Index	Keypoint Detection (4D)	Keypoint Description (128D)	Offset to Centroid
Image 1	1	[x, y, s, Theta]	[…]	[x,y]
Image 1	2	[x, y, s, Theta]	[…]	[x,y]
….	….	….	…	…
Image 1	265	[x, y, s, Theta]	[…]	[x,y]
Image 2	1	[x, y, s, Theta]	[…]	[x,y]
Image 2	2	[x, y, s, Theta]	[…]	[x,y]
…	…	…	…	…
Image 2	645	[x, y, s, Theta]	[…]	[x,y]
Image K	1	[x, y, s, Theta]	[…]	[x,y]
Image K	2	[x, y, s, Theta]	[…]	[x,y]
…	…	…	…	…
Image K	134	[x, y, s, Theta]	[…]	[x,y]



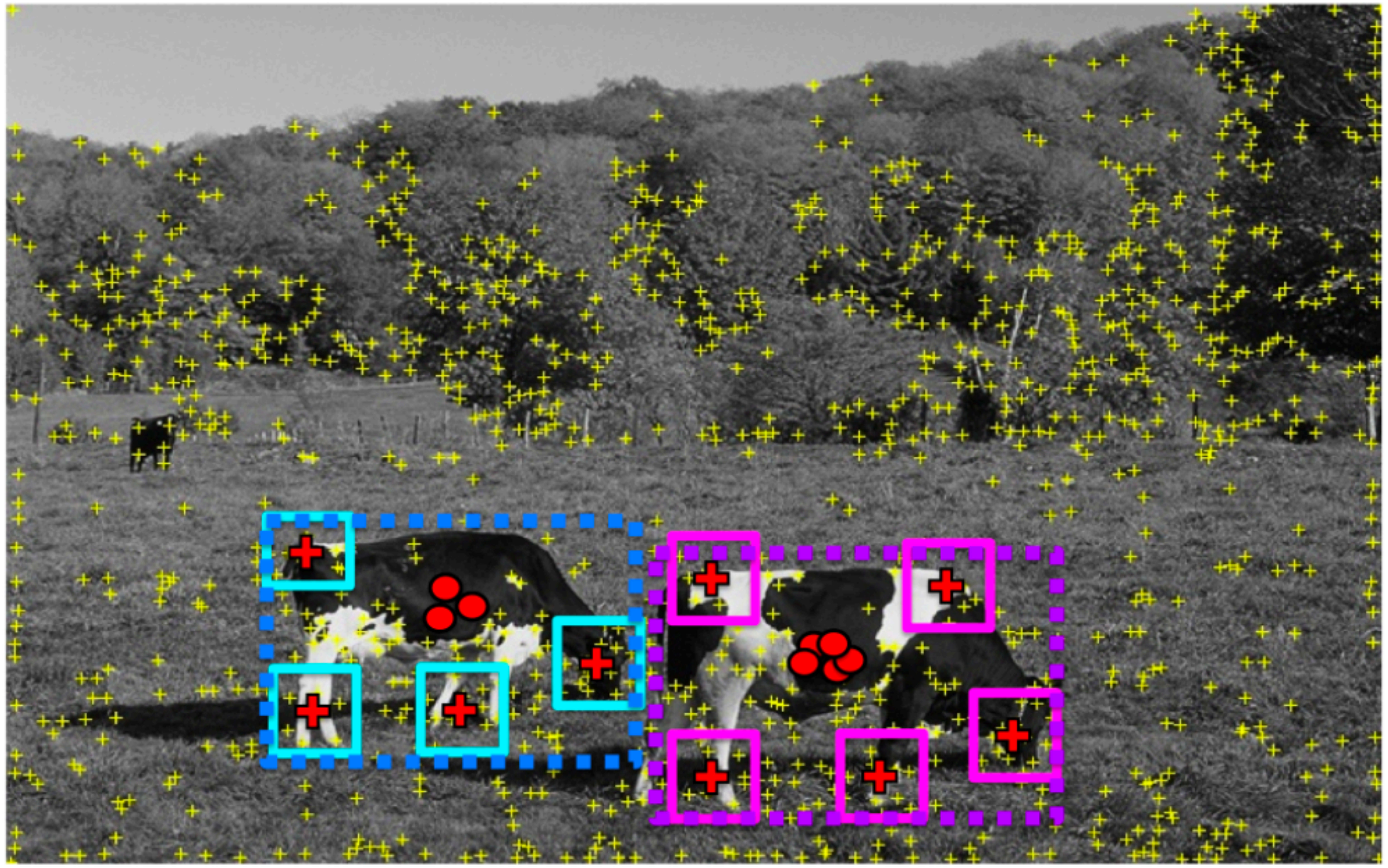
* Slide from Sanja Fidler

Example 1: Object Recognition — Implicit Shape Model

“**Training**” images of cows

“**Testing**” image

box around patches = object



Find objects based on the back projected patches

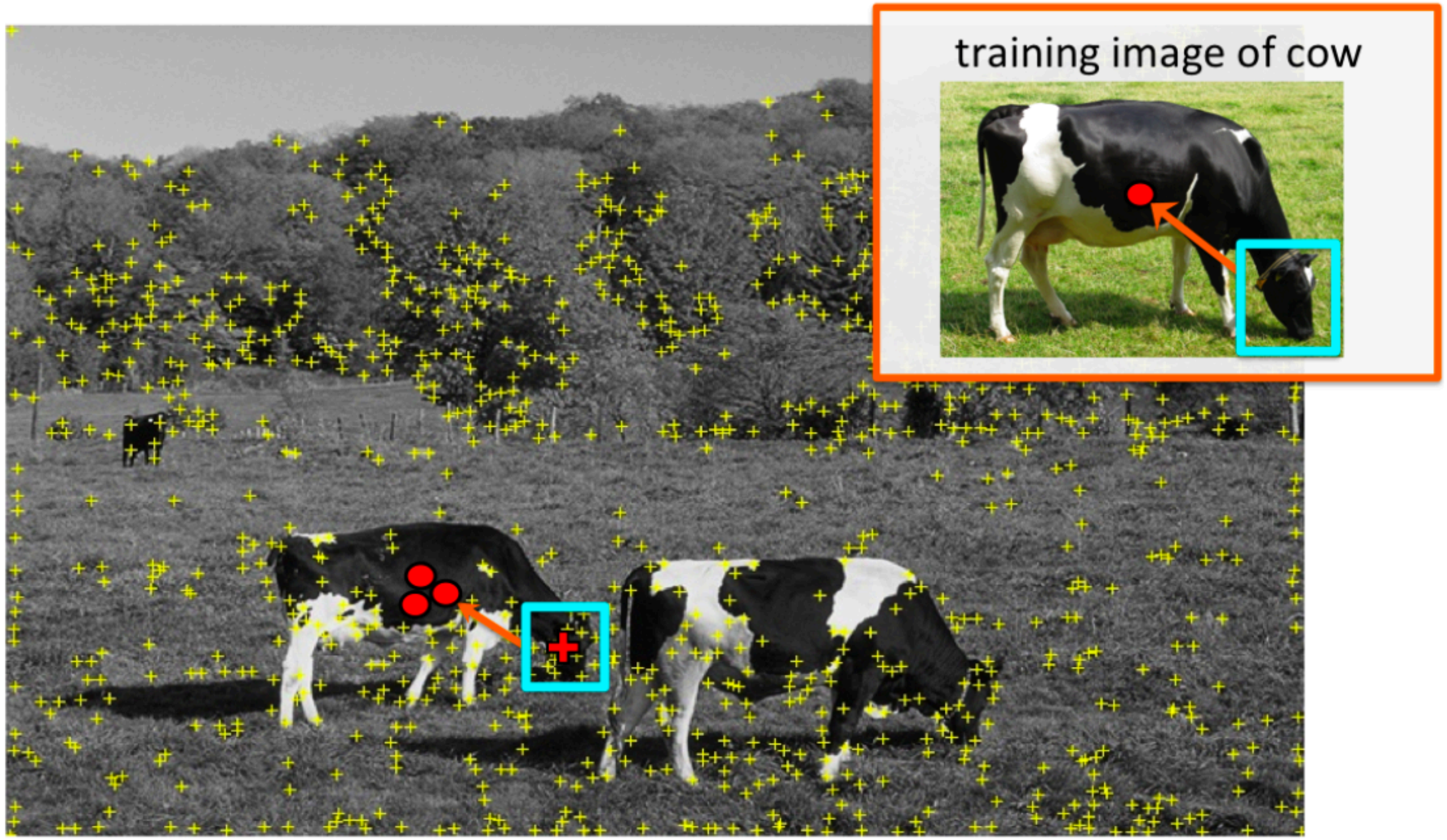
* Slide from Sanja Fidler

Example 1: Object Recognition — Implicit Shape Model

“**Training**” images of cows

“**Testing**” image

Really easy … but slow … how do we make it fast?

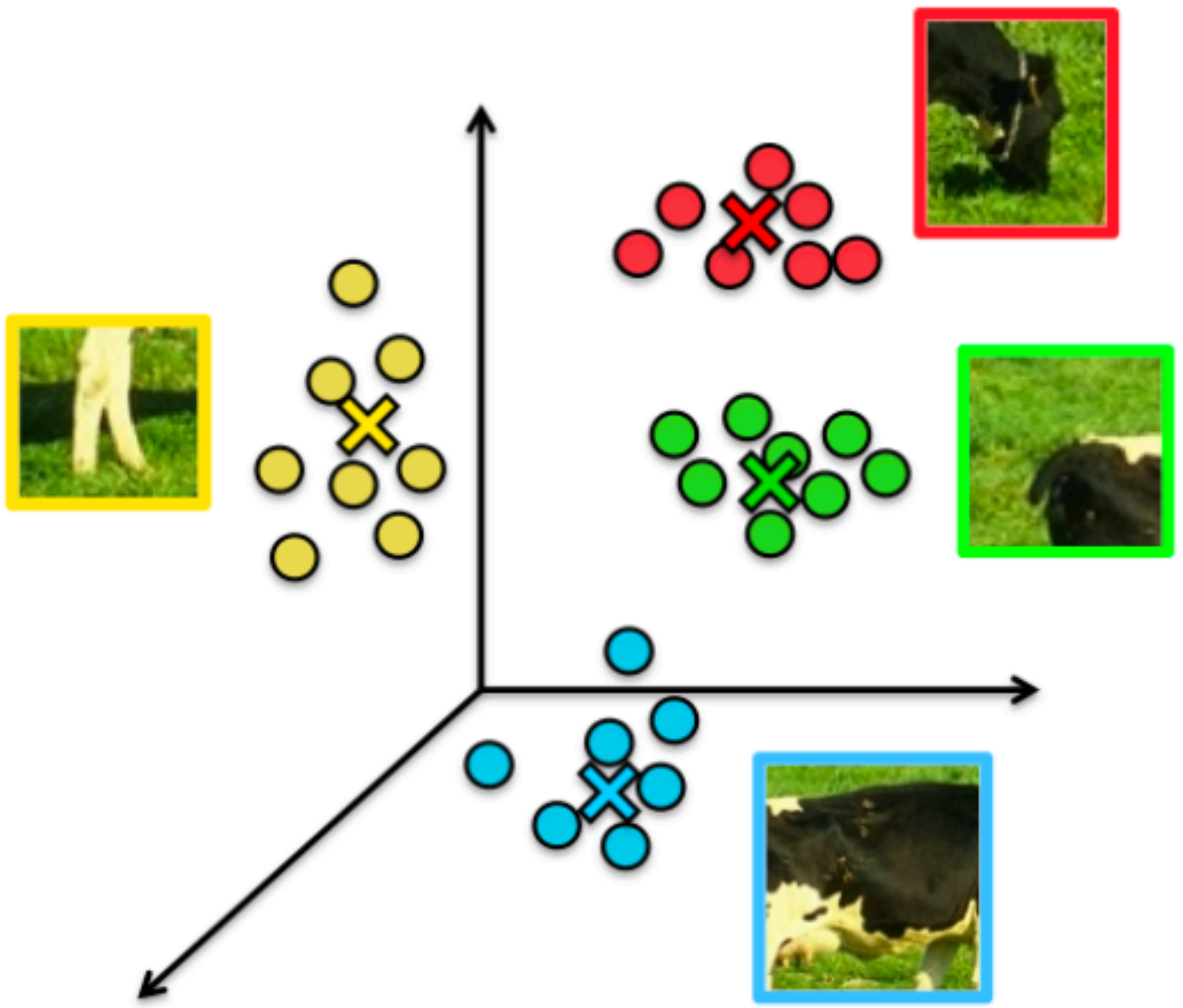


We need to match a patch around each yellow keypoint to all patches in all training images (**slow**)

* Slide from Sanja Fidler

Visual Words

- **Visual vocabulary** (we saw this for retrieval)
- Compare each patch to a small set of visual words (clusters)

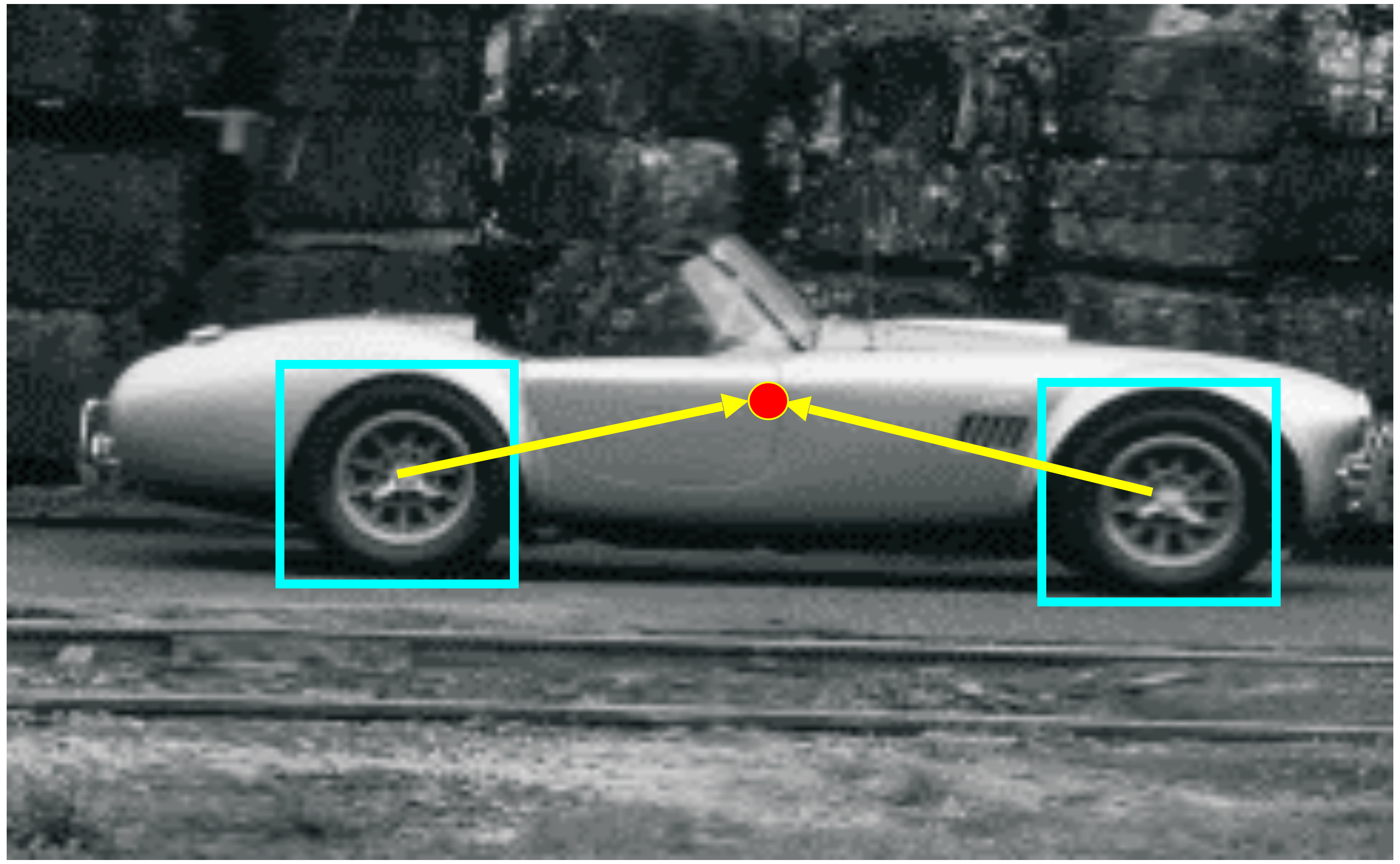


Visual words (visual codebook)!

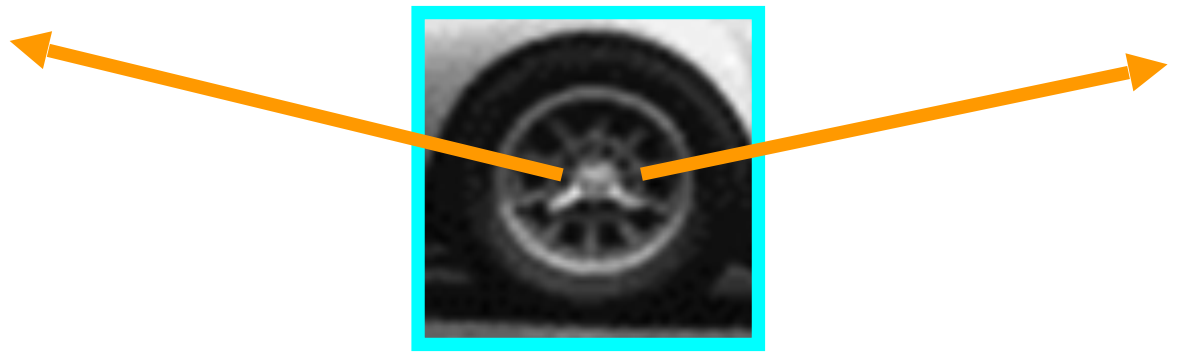
* Slide from Sanja Fidler

Example 1: Object Recognition — Implicit Shape Model

Index displacements by “visual codeword”



training image



visual codeword with
displacement vectors

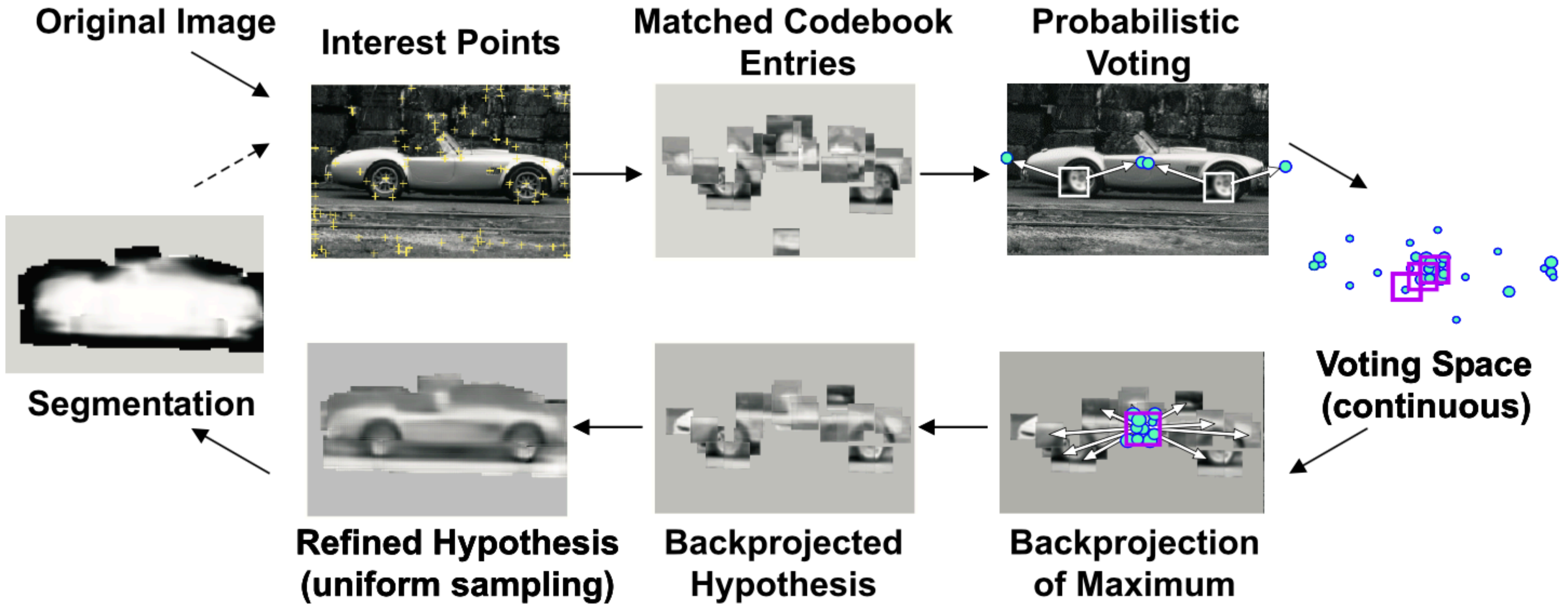
B. Leibe, A. Leonardis, and B. Schiele, Combined Object Categorization and Segmentation with an Implicit Shape Model, ECCV Workshop on Statistical Learning in Computer Vision 2004

Example 1: Object Recognition — Implicit Shape Model

B. Leibe, A. Leonardis, and B. Schiele, Combined Object Categorization and Segmentation with an Implicit Shape Model, ECCV Workshop on Statistical Learning in Computer Vision 2004

Inferring Other Information: **Segmentation**

Combined object detection and segmentation using an implicit shape model.
Image patches cast weighted votes for the object centroid.



B. Leibe, A. Leonardis, and B. Schiele, Combined Object Categorization and Segmentation with an Implicit Shape Model, ECCV Workshop on Statistical Learning in Computer Vision 2004

Example 1: Object Recognition — Implicit Shape Model

"**Training**" images of cows

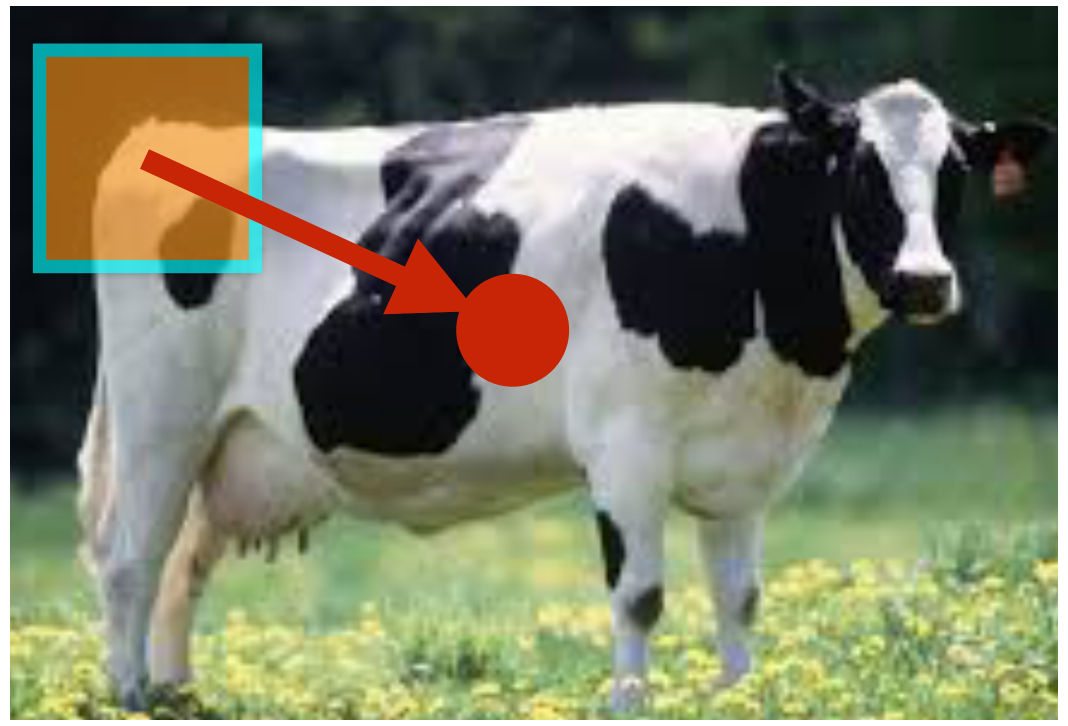
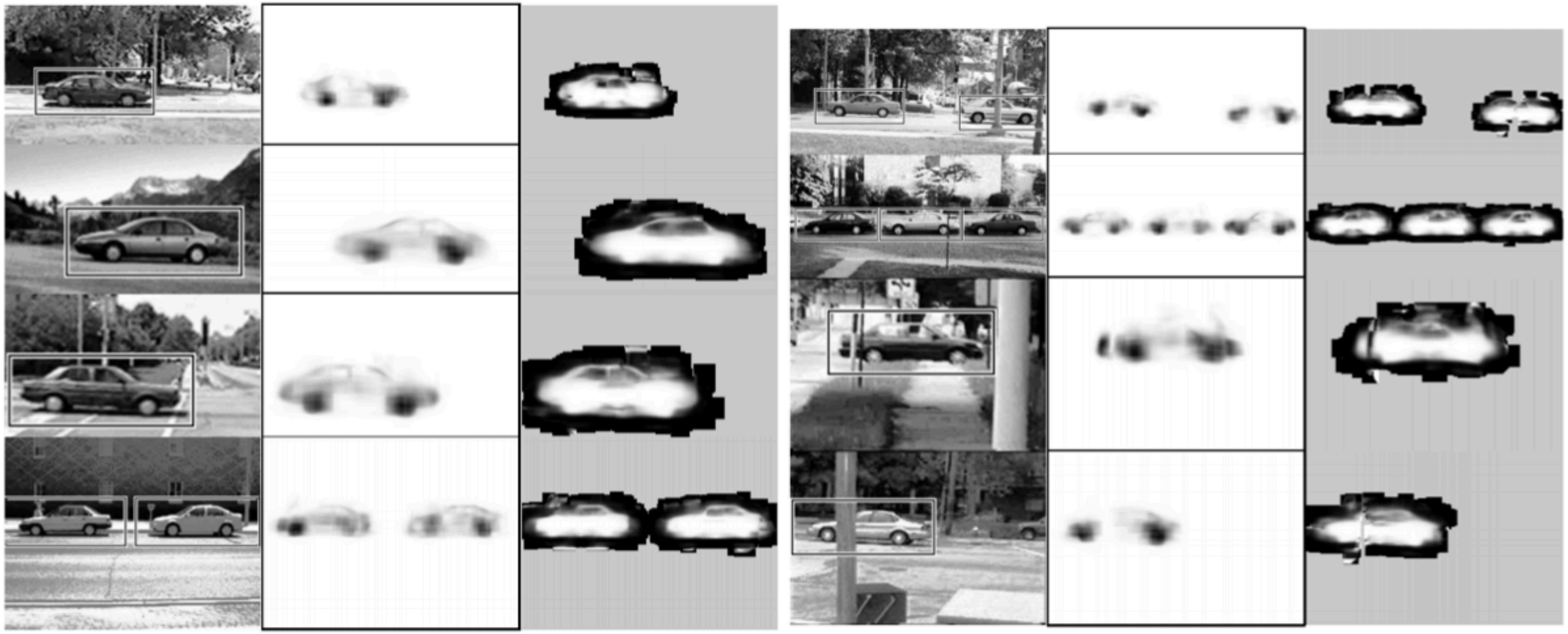


Image Index	Keypoint Index	Keypoint Detection (4D)	Keypoint Description (128D)	Offset to Centroid	Segment
Image 1	1	[x, y, s, Theta]	[…]	[x,y]	
Image 1	2	[x, y, s, Theta]	[…]	[x,y]	
….	….	….	…	…	
Image 1	265	[x, y, s, Theta]	[…]	[x,y]	
Image 2	1	[x, y, s, Theta]	[…]	[x,y]	
Image 2	2	[x, y, s, Theta]	[…]	[x,y]	
…	…	…	…	…	
Image 2	645	[x, y, s, Theta]	[…]	[x,y]	
Image K	1	[x, y, s, Theta]	[…]	[x,y]	
Image K	2	[x, y, s, Theta]	[…]	[x,y]	
…	…	…	…	…	
Image K	134	[x, y, s, Theta]	[…]	[x,y]	

* Slide from Sanja Fidler

Inferring Other Information: **Segmentation**

Idea: When back-projecting, back-project labeled segmentations per training patch



(a) detections (b) p(figure) (c) segmentation (a) detections (b) p(figure) (c) segmentation

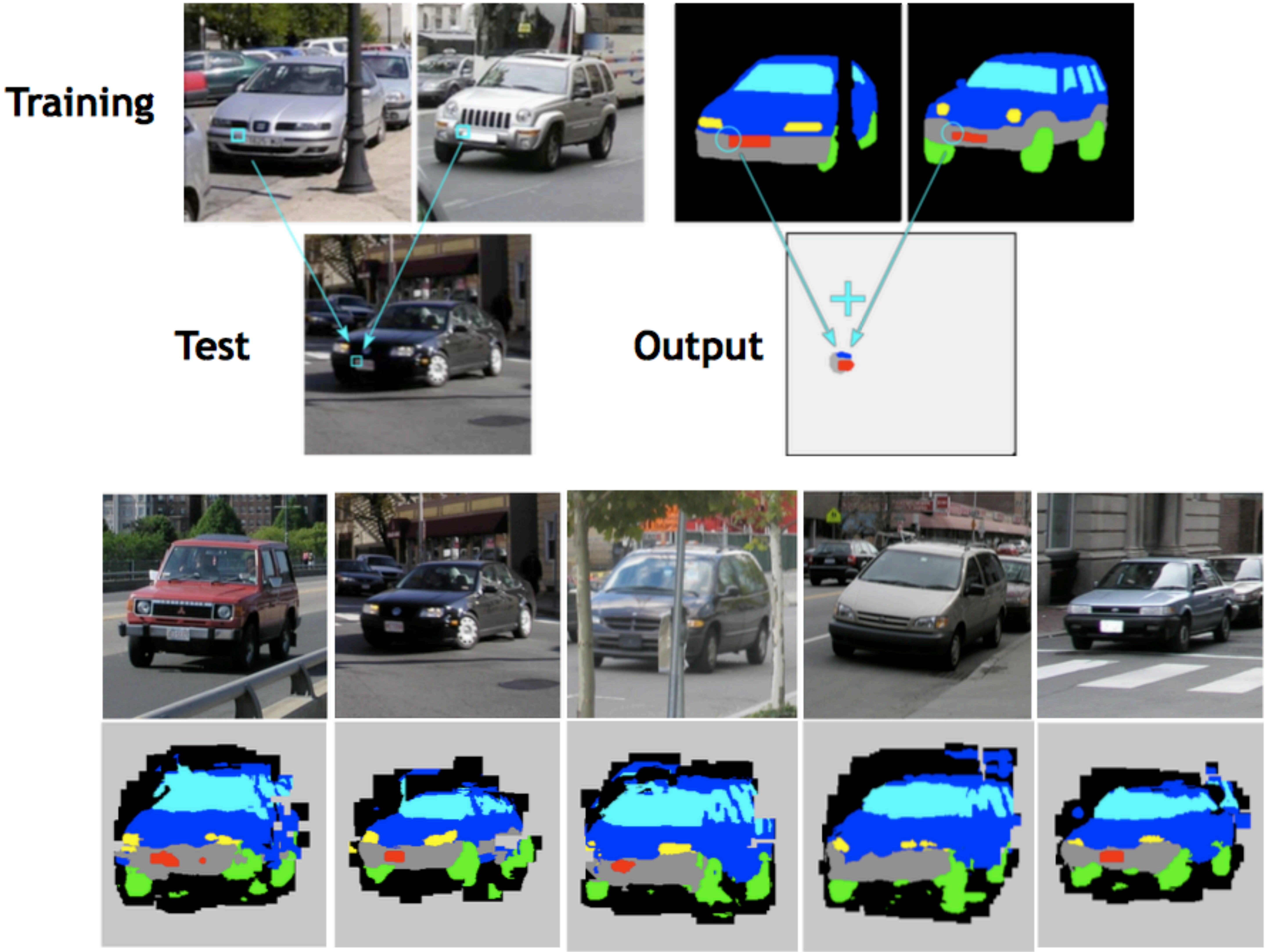
[Source: B. Leibe]

Inferring Other Information: **Segmentation**

[Source: B. Leibe]

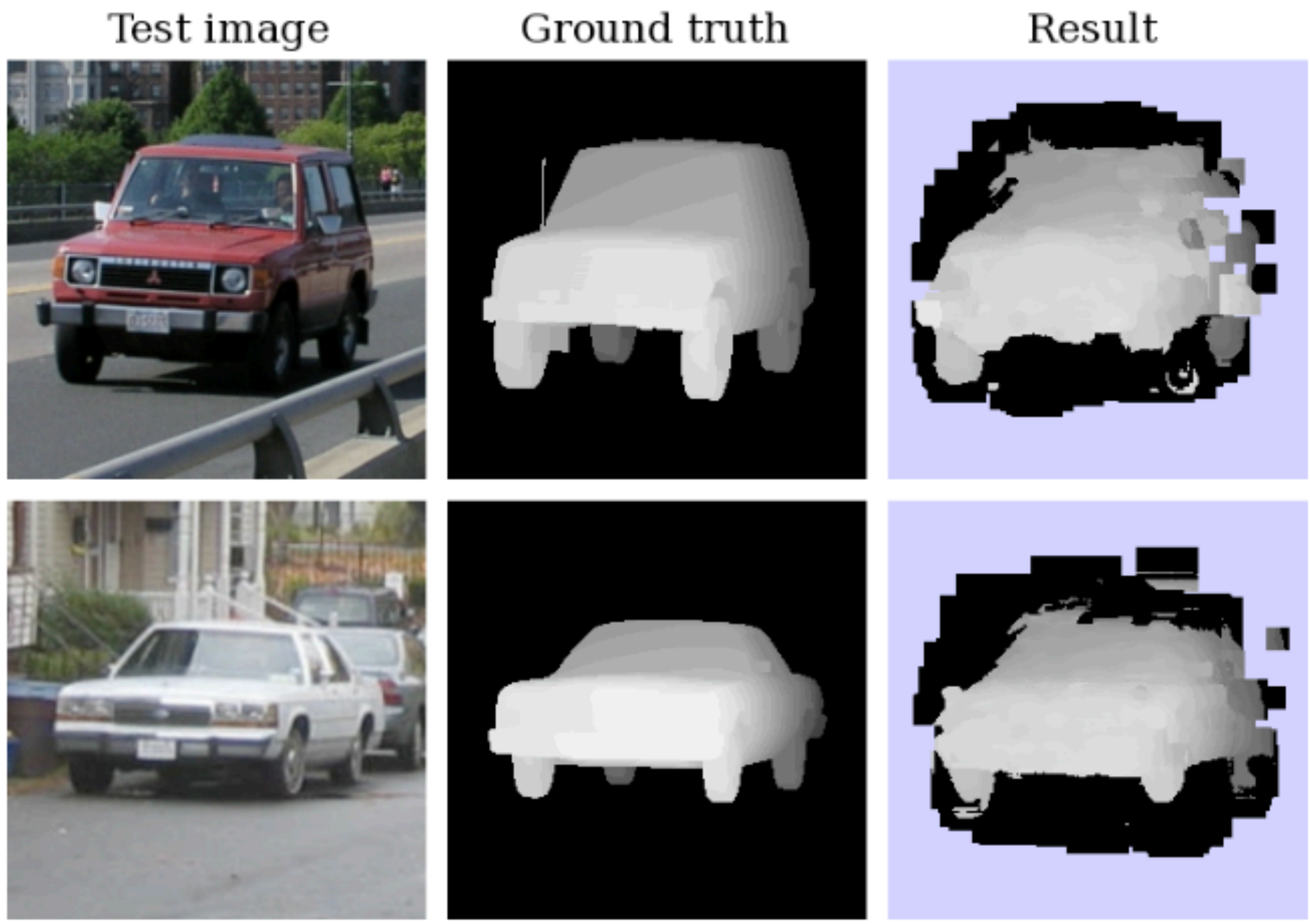
* Slide from Sanja Fidler

Inferring Other Information: **Part Labels**



* Slide from Sanja Fidler

Inferring Other Information: **Depth**



"Depth from a single image"

* Slide from Sanja Fidler

Example 2: Object Recognition — Boundary Fragments

Boundary fragments cast weighted votes for the object centroid. Also obtains an estimate of the object's contour.

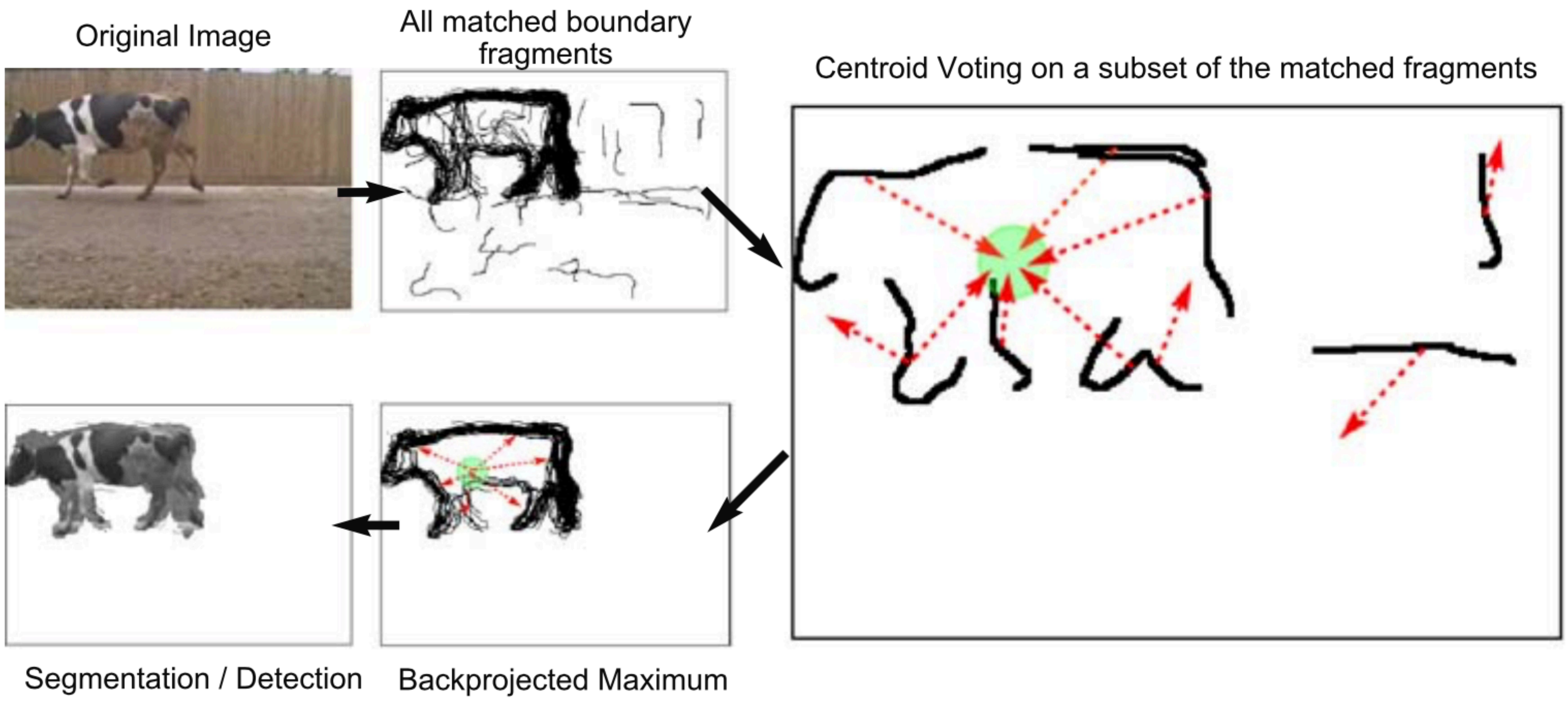


Image credit: Opelt et al., 2006

Example 2: Object Recognition — Boundary Fragments

Boundary fragments cast weighted votes for the object centroid. Also obtains an estimate of the object's contour.

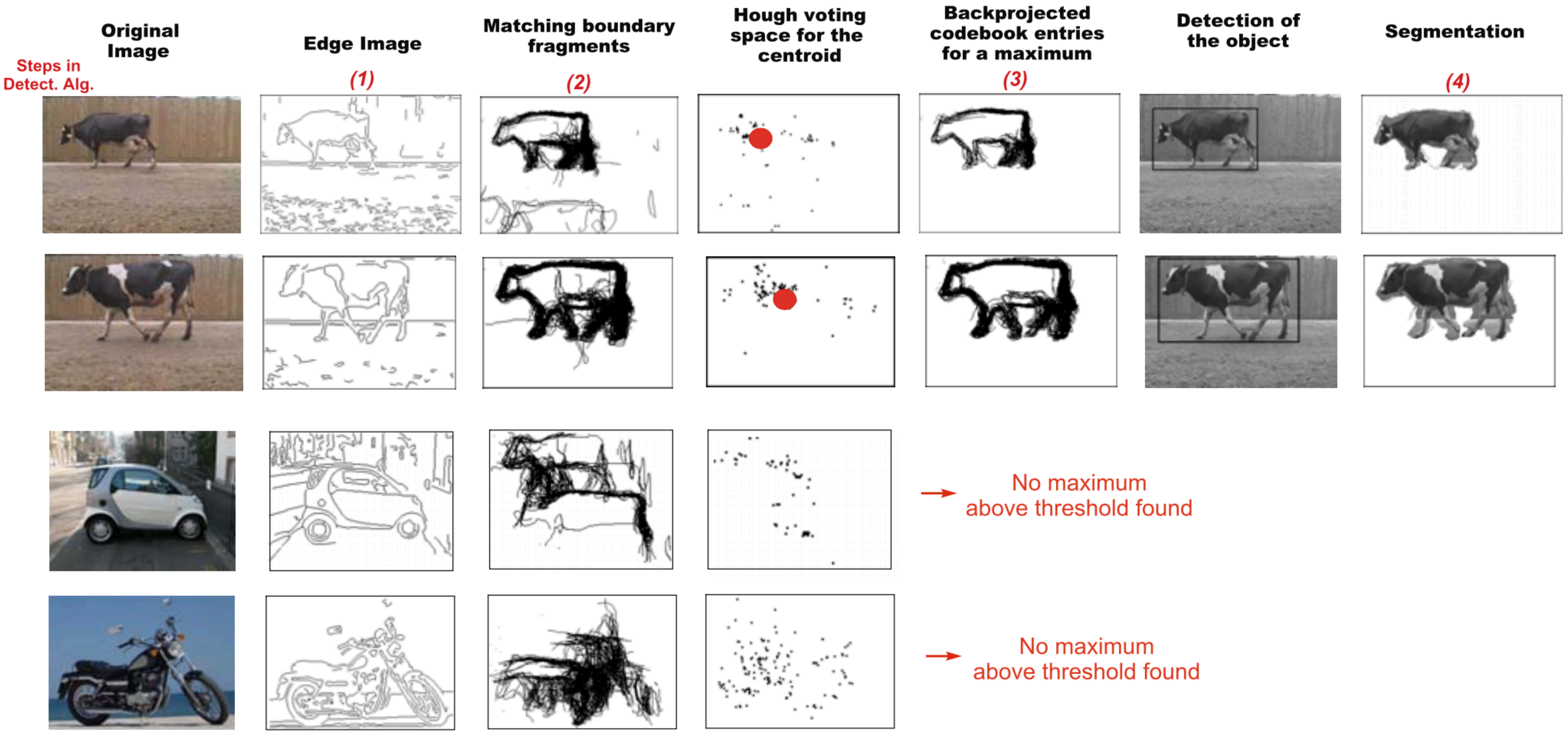


Image credit: Opelt et al., 2006

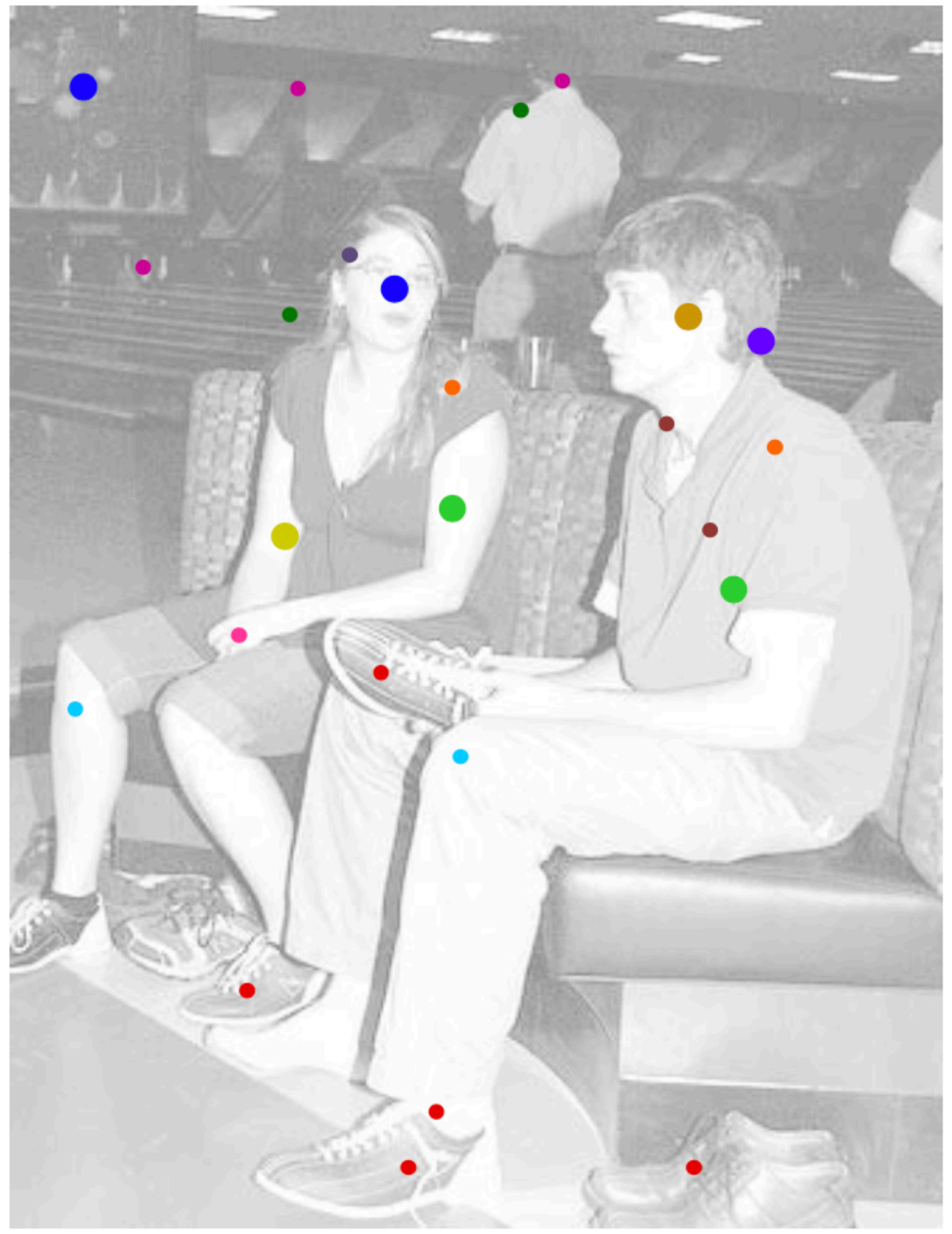
Example 3: Object Recognition — Poselets

Poselets are image patches that have distinctive appearance and can be used to infer some of the configuration of a parts-based object. Detected poselets vote for the object configuration.

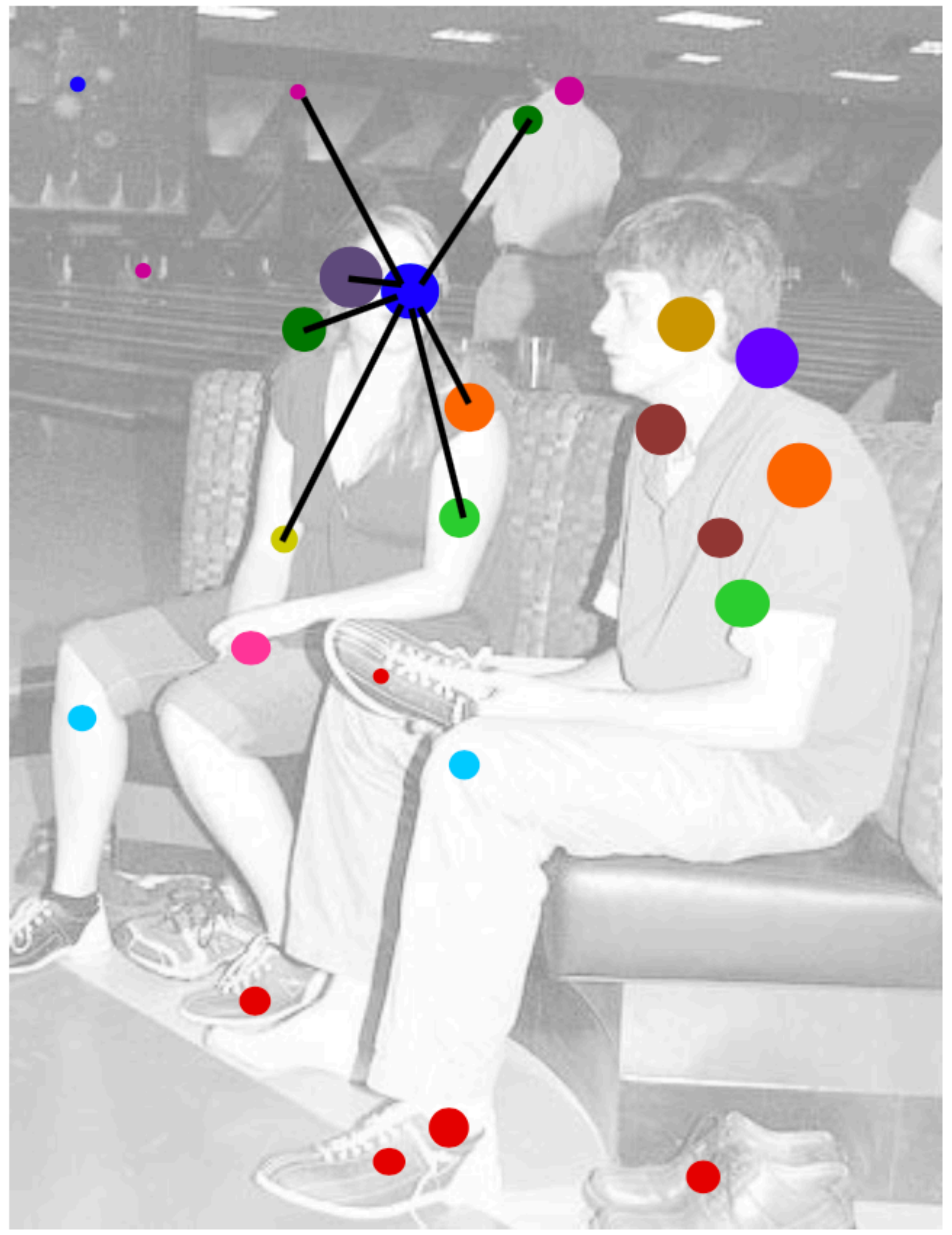
Image credit: Bourdev and Malik, 2009

Example 3: Object Recognition — Poselets

Poselets are image patches that have distinctive appearance and can be used to infer some of the configuration of a parts-based object. Detected poselets vote for the object configuration.

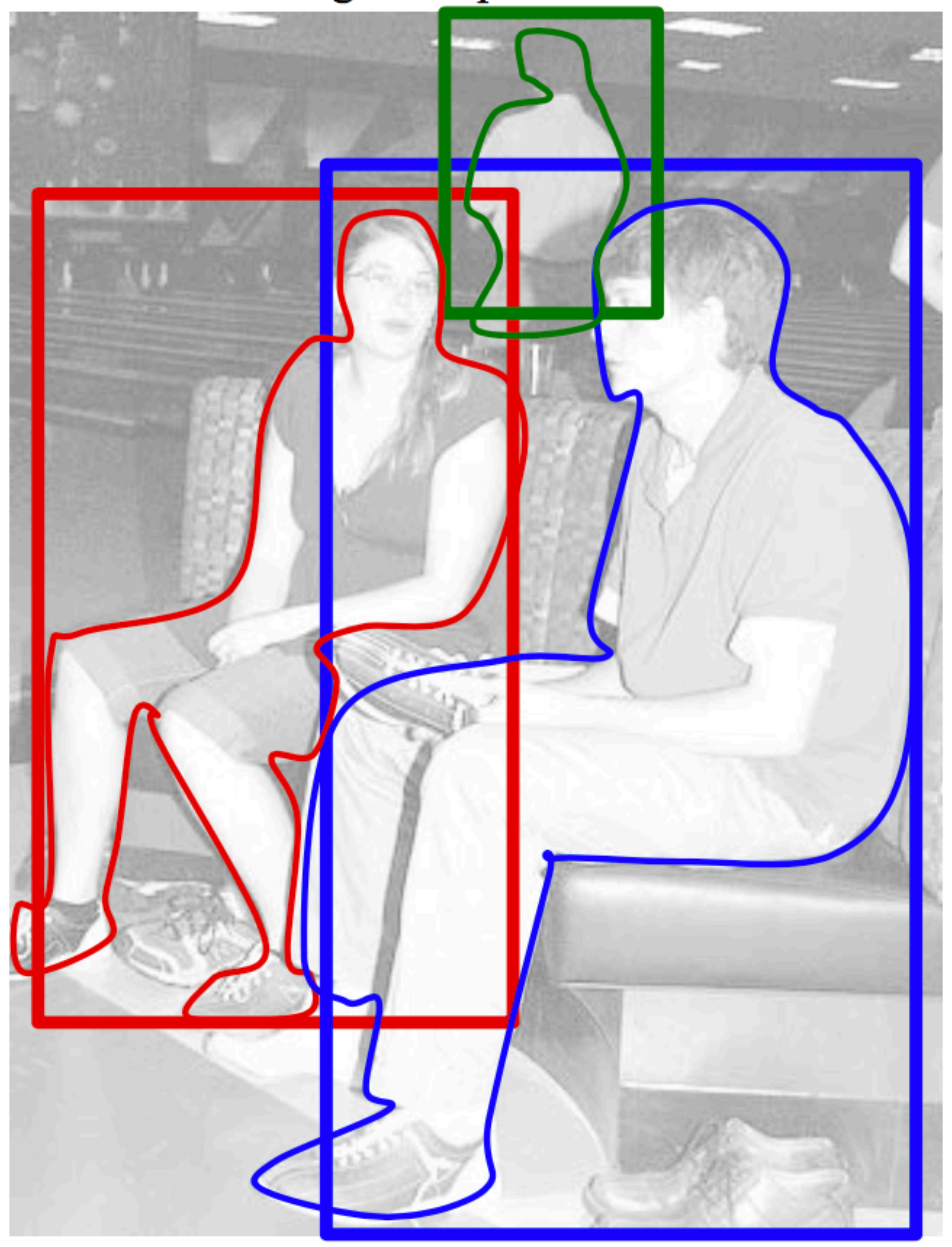


1. q-scores. Different colors illustrate different poselet detectors firing in the image. The blob size illustrates the score of the independent poselet classifier.



2. Q-scores (Section 4). Evidence from consistent poselet activations leads to a reranking based on mutual activation (Q-scores). Weaker activations consistent with others gain importance, whereas inconsistent ones get damped.

3. Clustering (Section 5). Activations are merged in a greedy manner starting with the strongest activation. Merging is based on pairwise consistency.



4. Bounding boxes (Section 6) and segmentations (Section 7). We predict the visible bounds and the contour of the person using the poselets within the cluster.

Image credit: Bourdev and Malik, 2009

Discussion of Hough Transform

Advantages:

— Can handle high percentage of outliers: each point votes separately

— Can detect multiple instances of a model in a single pass

Disadvantages:

— Complexity of search time increases exponentially with the number of model parameters

— Can be tricky to pick a good bin size

Summary of Hough Transform

The **Hough transform** is another technique for fitting data to a model

— a voting procedure

— possible model parameters define a quantized accumulator array

— data points “vote" for compatible entries in the accumulator array

A key is to have each data point (token) constrain model parameters as tightly as possible

CPSC 425: Computer Vision

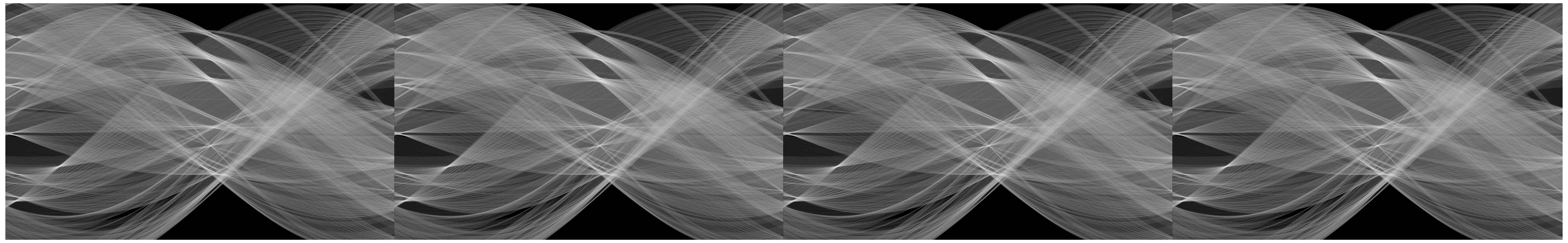


Image Credit: Ioannis (Yannis) Gkioulekas (CMU)

Lecture 22: Stereo

Stereo Vision

Problem Formulation:

Determine depth using two images acquired from (slightly) different viewpoints

Key Idea(s):

The 3D coordinates of each point imaged are constrained to lie along a ray. This is true also for a second image obtained from a (slightly) different viewpoint. Rays for the same point in the world intersect at the actual 3D location of that point

Stereo Vision

With two eyes, we acquire images of the world from slightly different viewpoints

We perceive **depth** based on **differences in the relative position of points** in the left image and in the right image

Binoculars

Binoculars enhance binocular depth perception in two distinct ways:

1. magnification
2. longer baseline (i.e., distance between entering light paths) compared to the normal human inter-pupillary distance

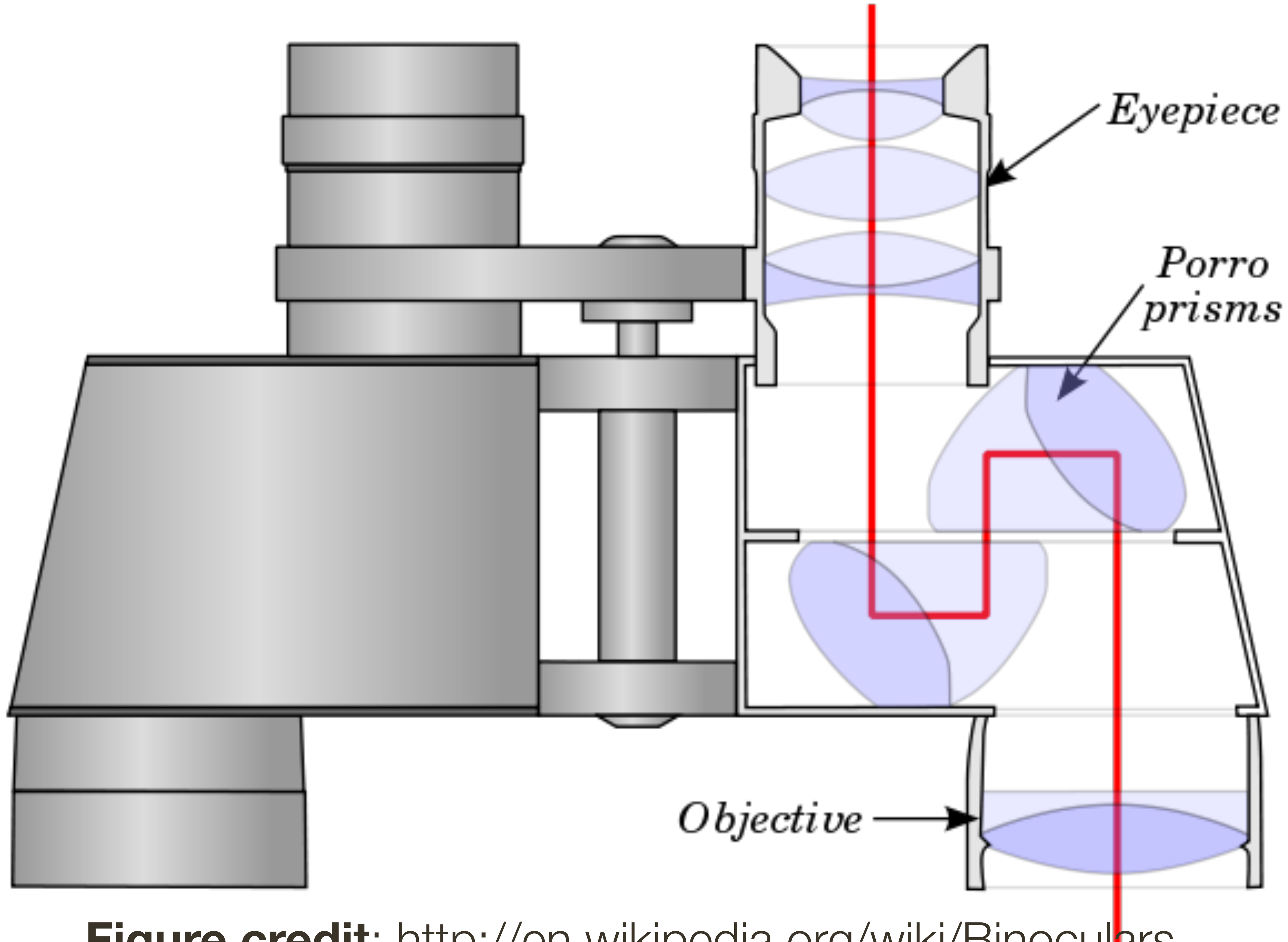


Figure credit: http://en.wikipedia.org/wiki/Binoculars

Stereo Vision

Task: Compute depth from two images acquired from (slightly) different viewpoints

Approach: “Match” locations in one image to those in another

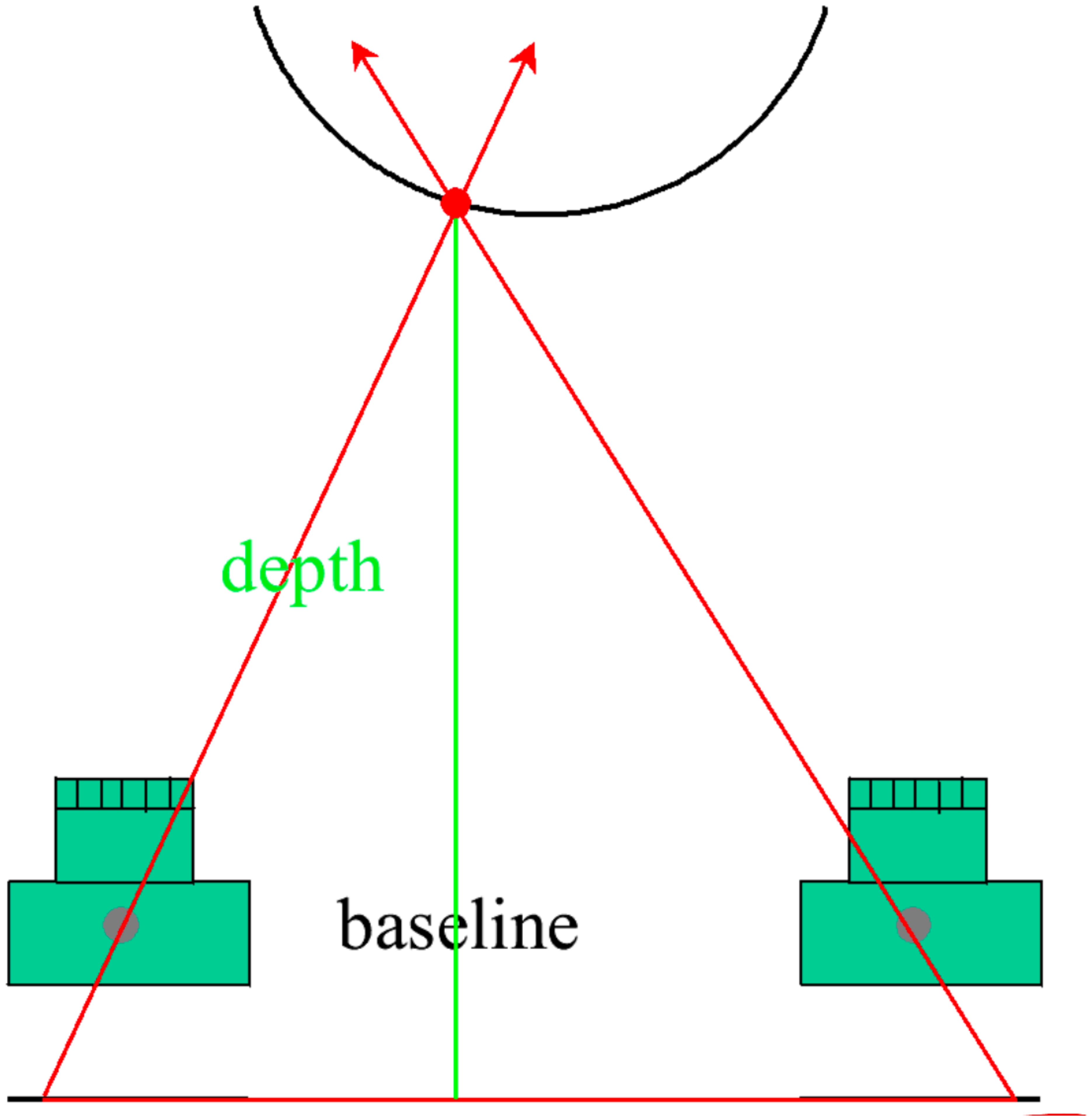
Sub-tasks:

— Calibrate cameras and camera positions

— Find all corresponding points (the hardest part)

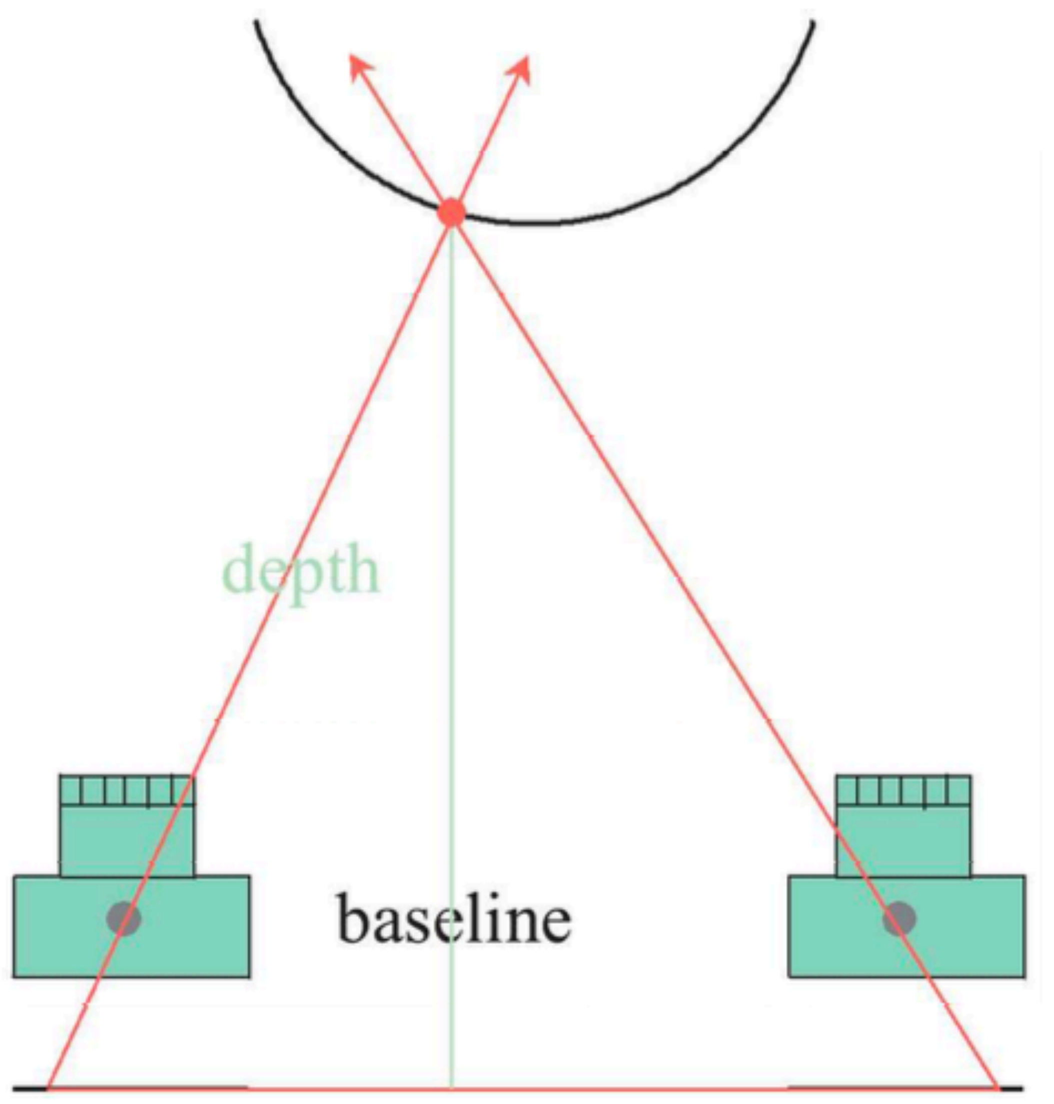
— Compute depth and surfaces

Stereo Vision

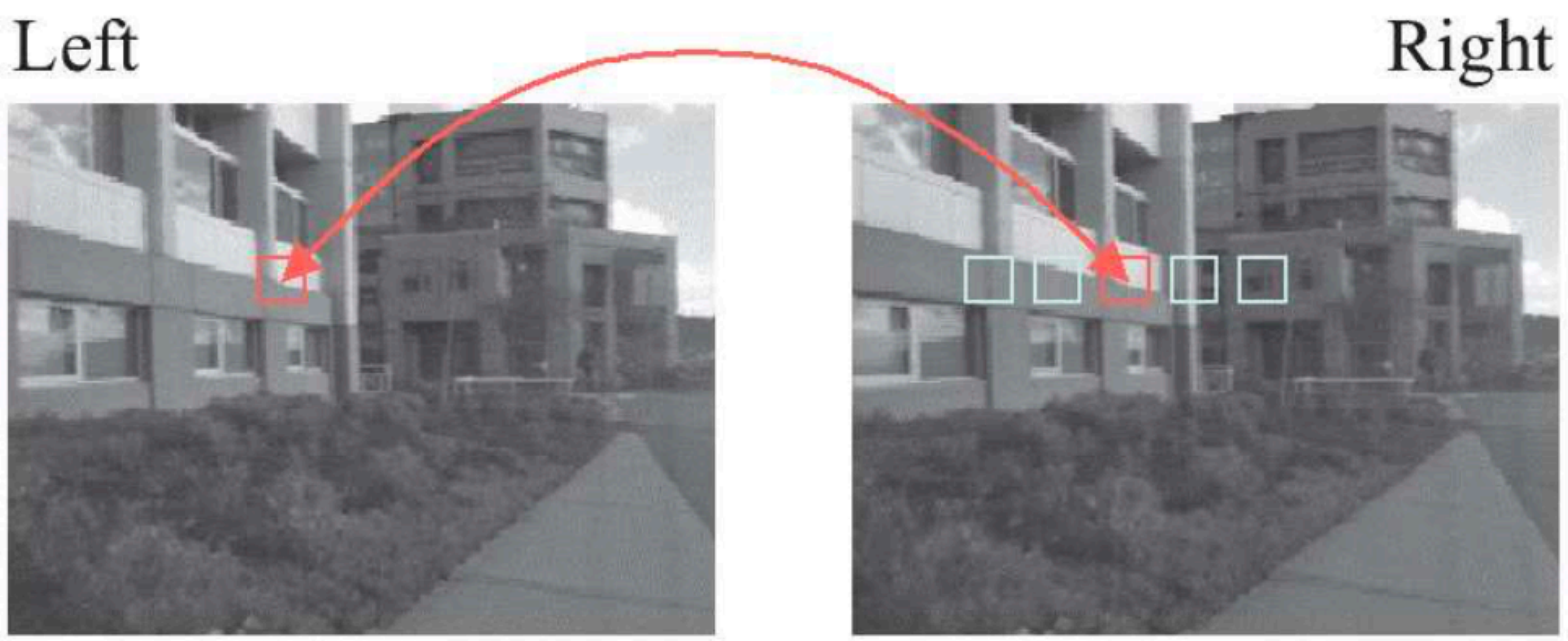


Slide credit: Trevor Darrell

Stereo Vision



Triangulate on two images of the same point



Match correlation windows across scan lines

Image credit: Point Grey Research
Slide credit: Trevor Darrell

Point Grey Research **Digiclops**

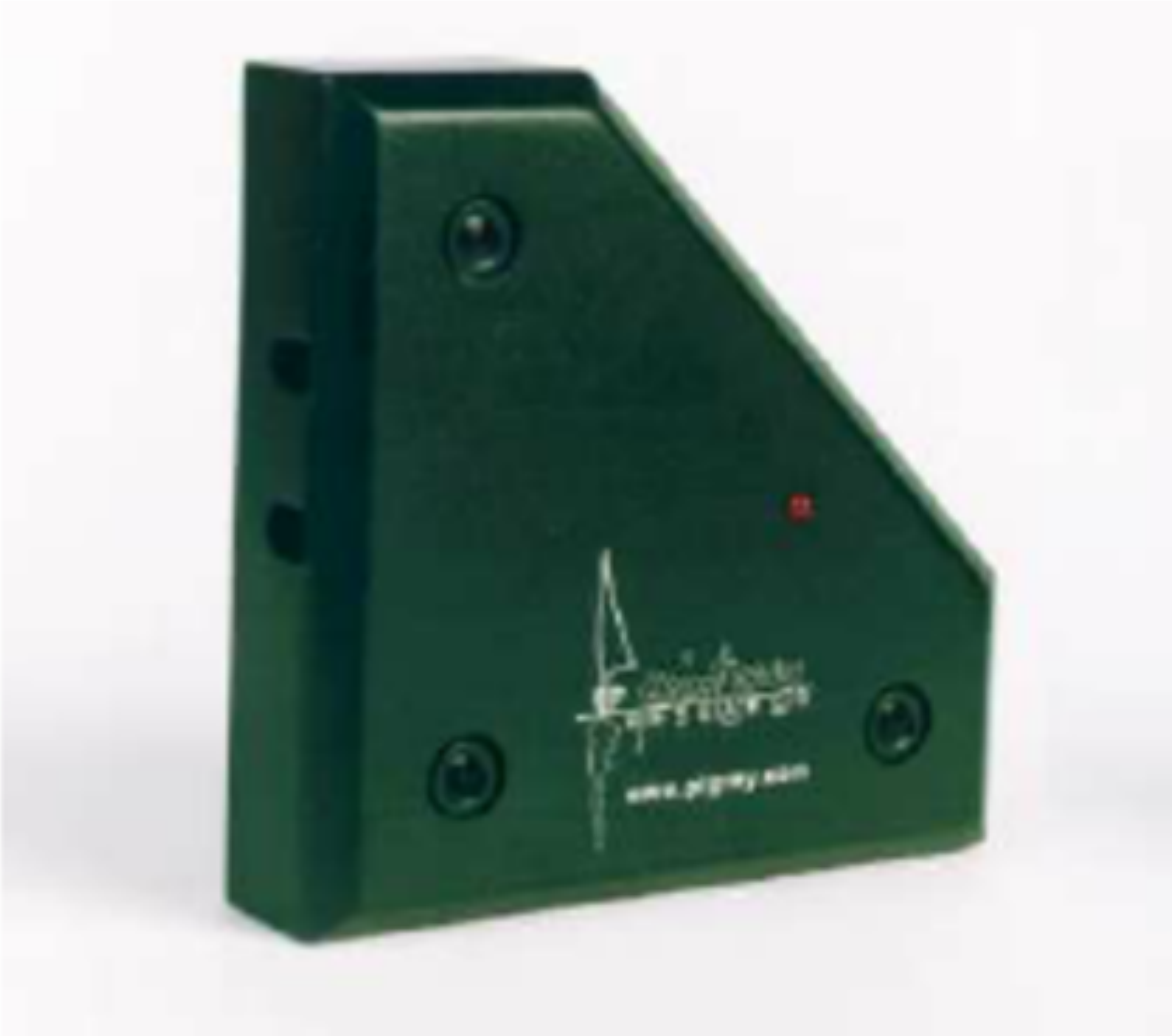
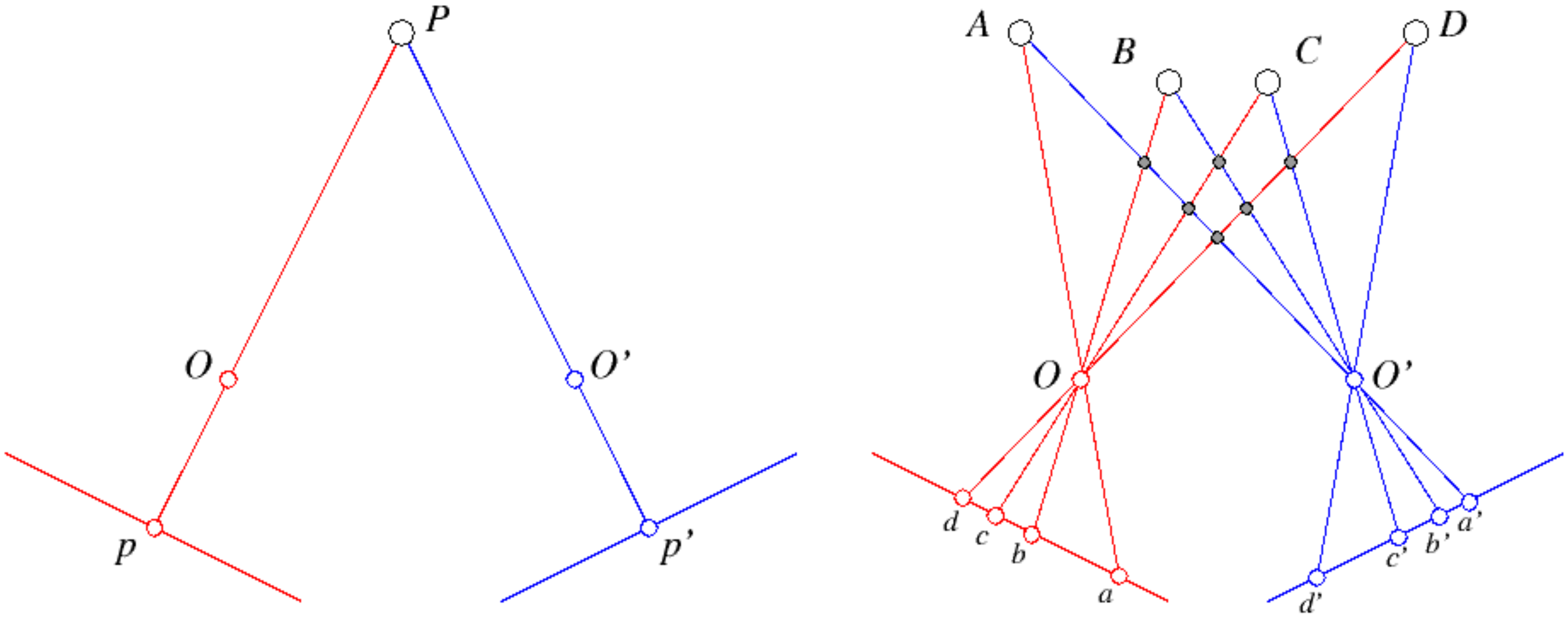


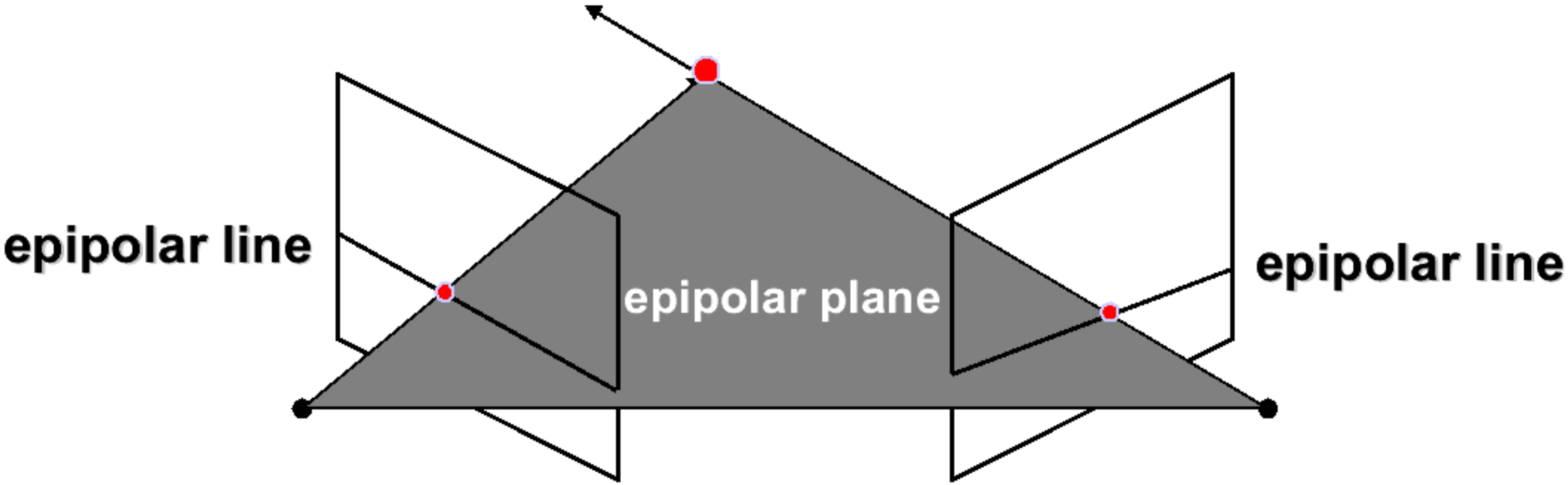
Image credit: Point Grey Research

Correspondence



Forsyth & Ponce (2nd ed.) Figure 7.2

The **Epipolar** Constraint



Matching points lie along corresponding epipolar lines

Reduces correspondence problem to 1D search along conjugate epipolar lines

Greatly reduces cost and ambiguity of matching

Slide credit: Steve Seitz

Simplest Case: **Rectified** Images

Image planes of cameras are **parallel**

Focal **points** are at same height

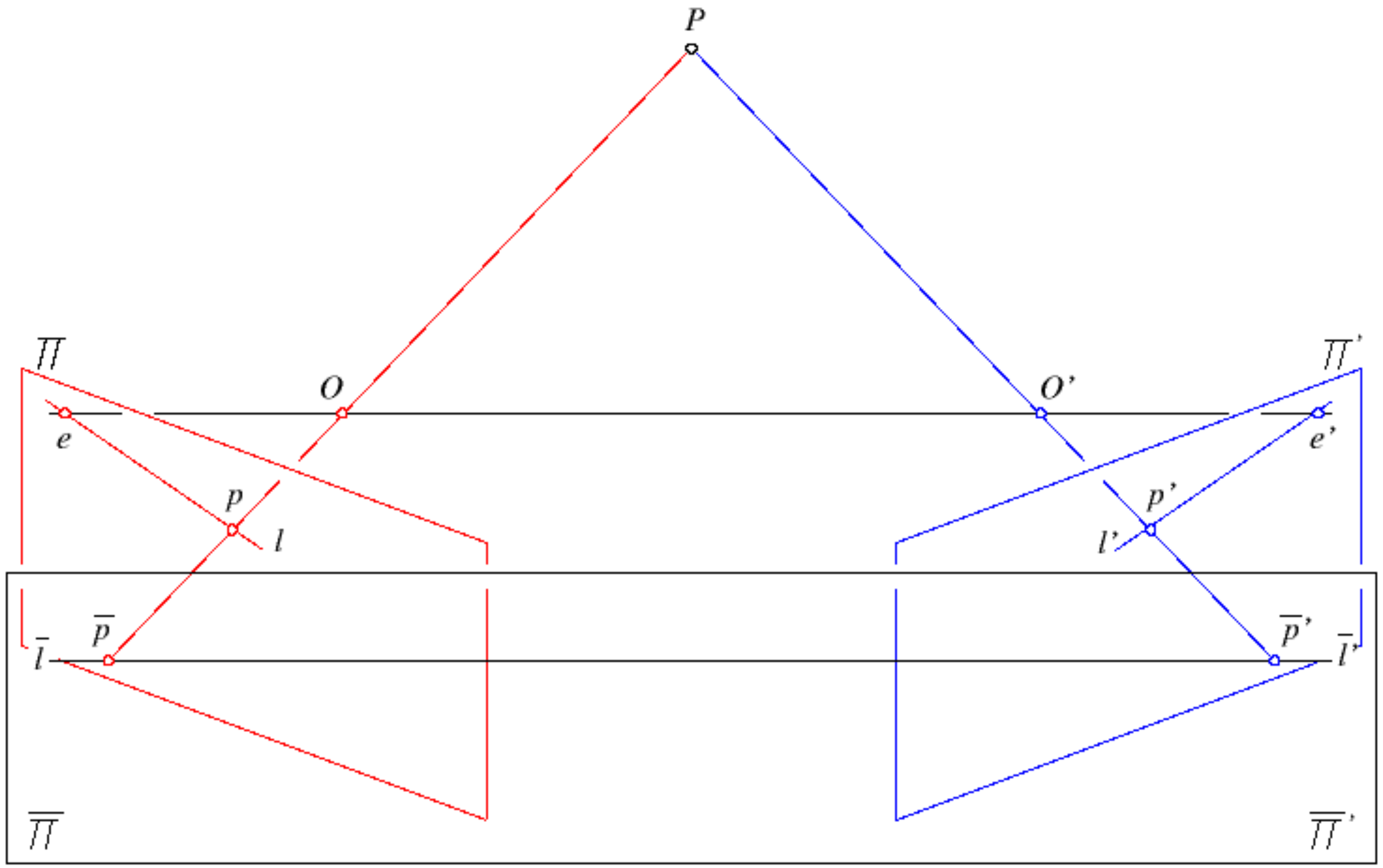
Focal **lengths** same

Then, **epipolar lines** fall along the **horizontal scan lines** of the images

We assume images have been **rectified** so that epipolar lines correspond to scan lines

- Simplifies algorithms
- Improves efficiency

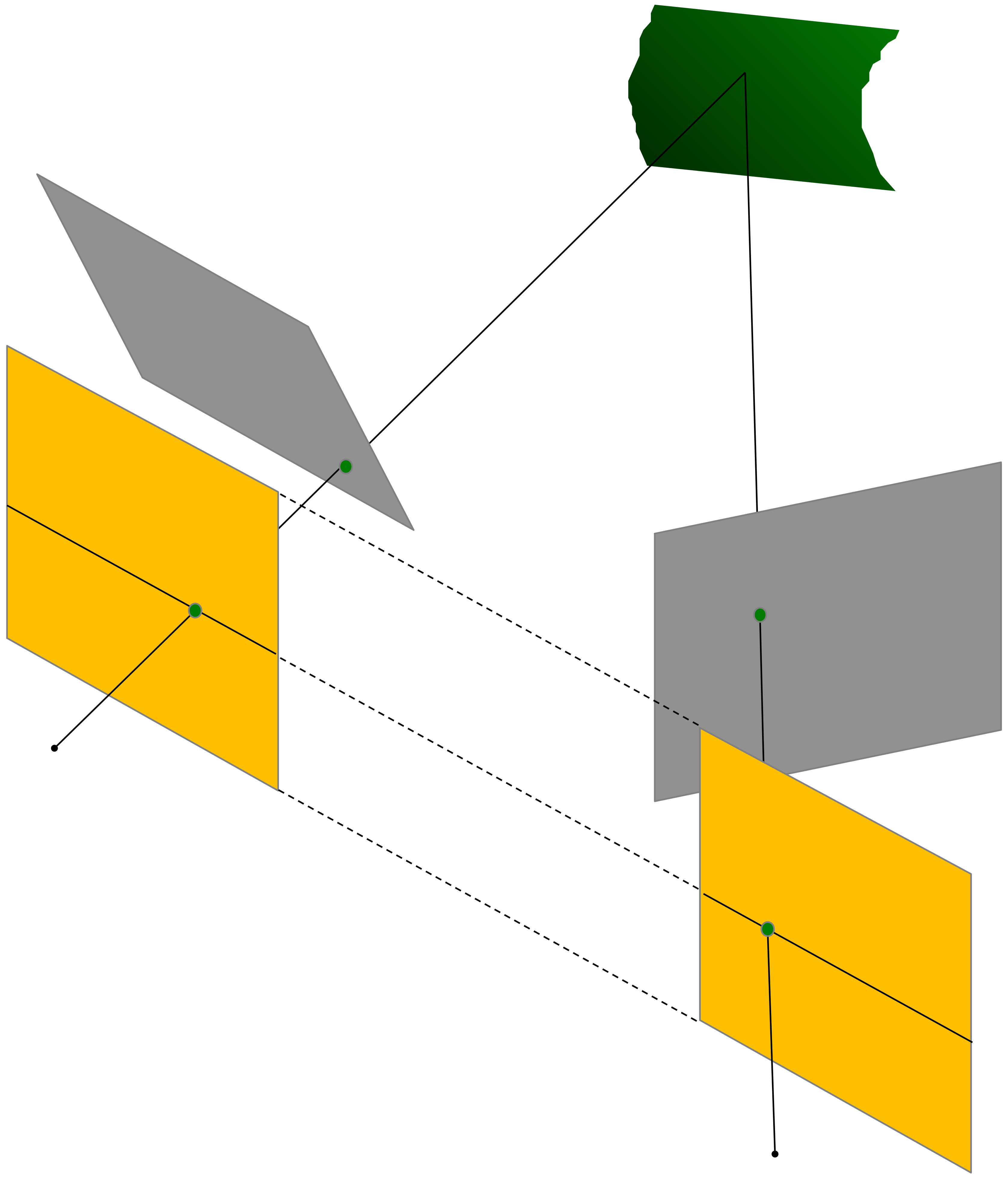
Rectified Stereo Pair



Rectified Stereo Pair

Reproject image planes onto a common plane parallel to the line between camera centers

Need two homographies (3x3 transform), one for each input image reprojection

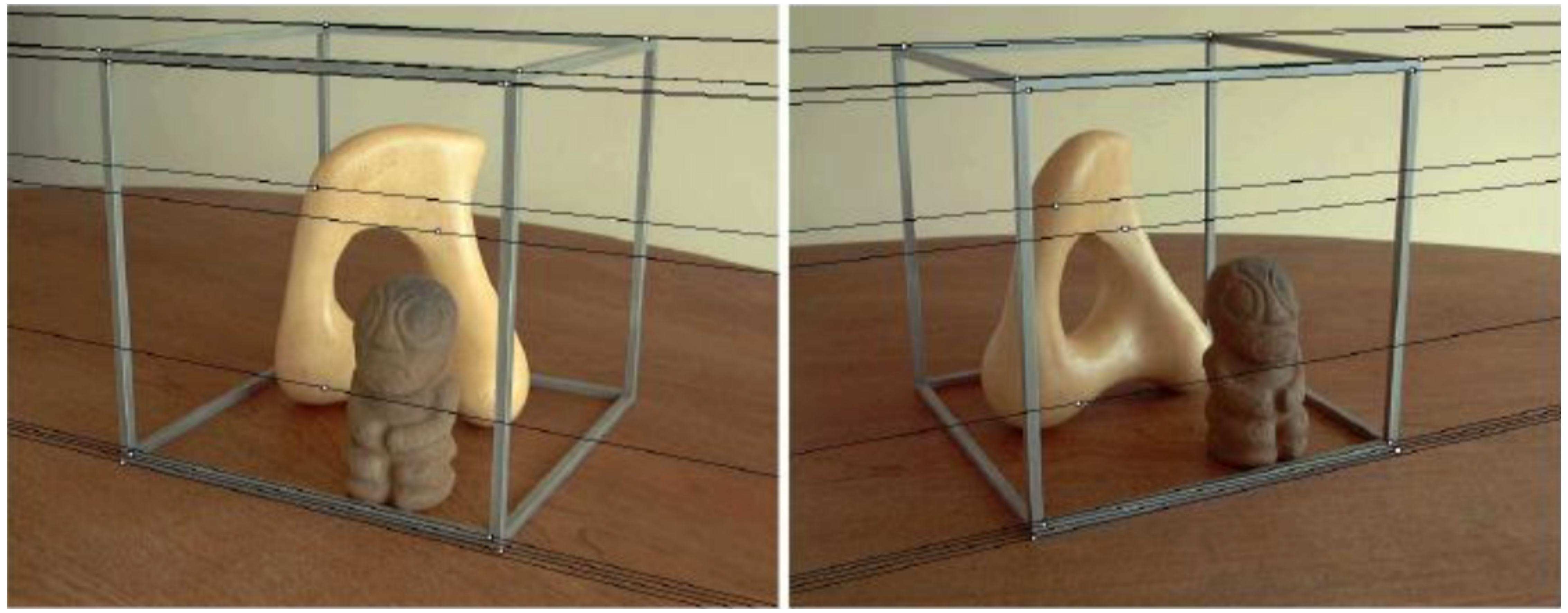


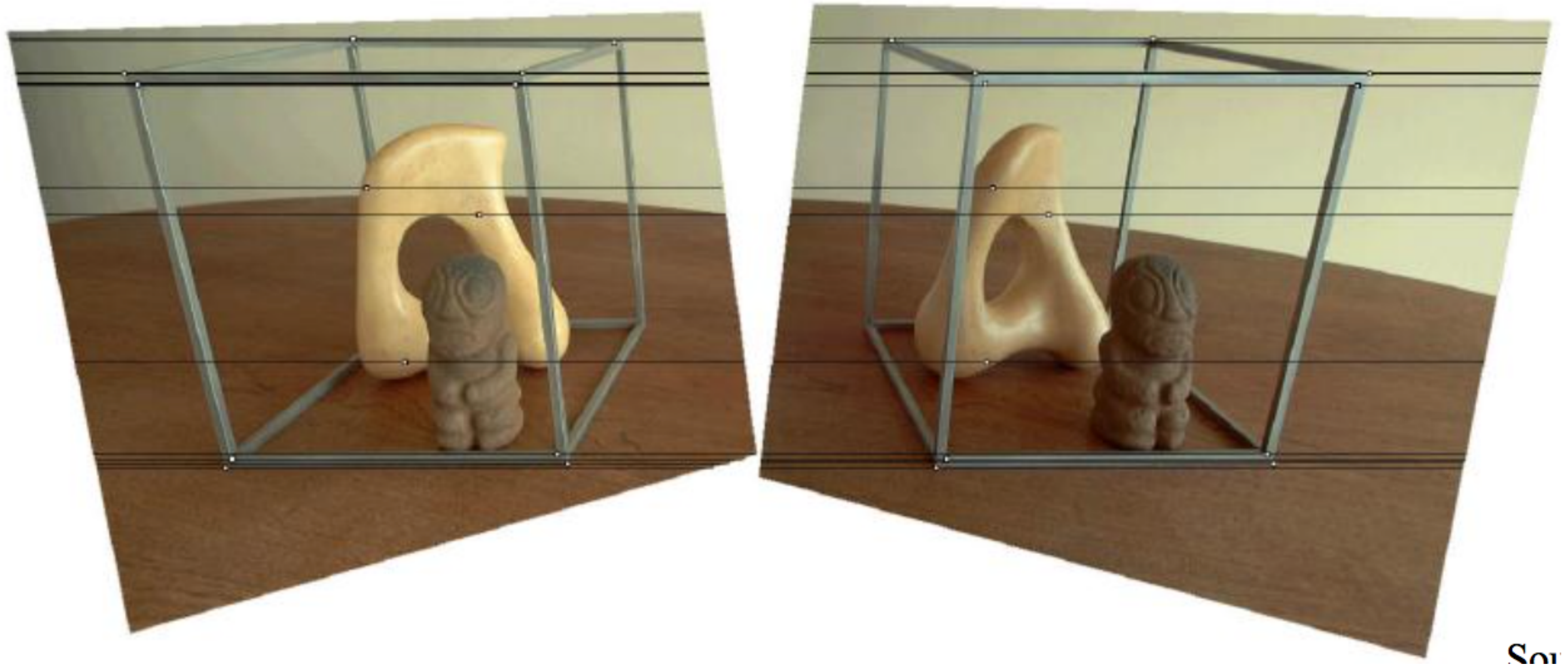
C. Loop and Z. Zhang. Computing Rectifying Homographies for Stereo Vision.Computer Vision and Pattern Recognition, 1999.

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Rectified Stereo Pair: Example

Before Rectification

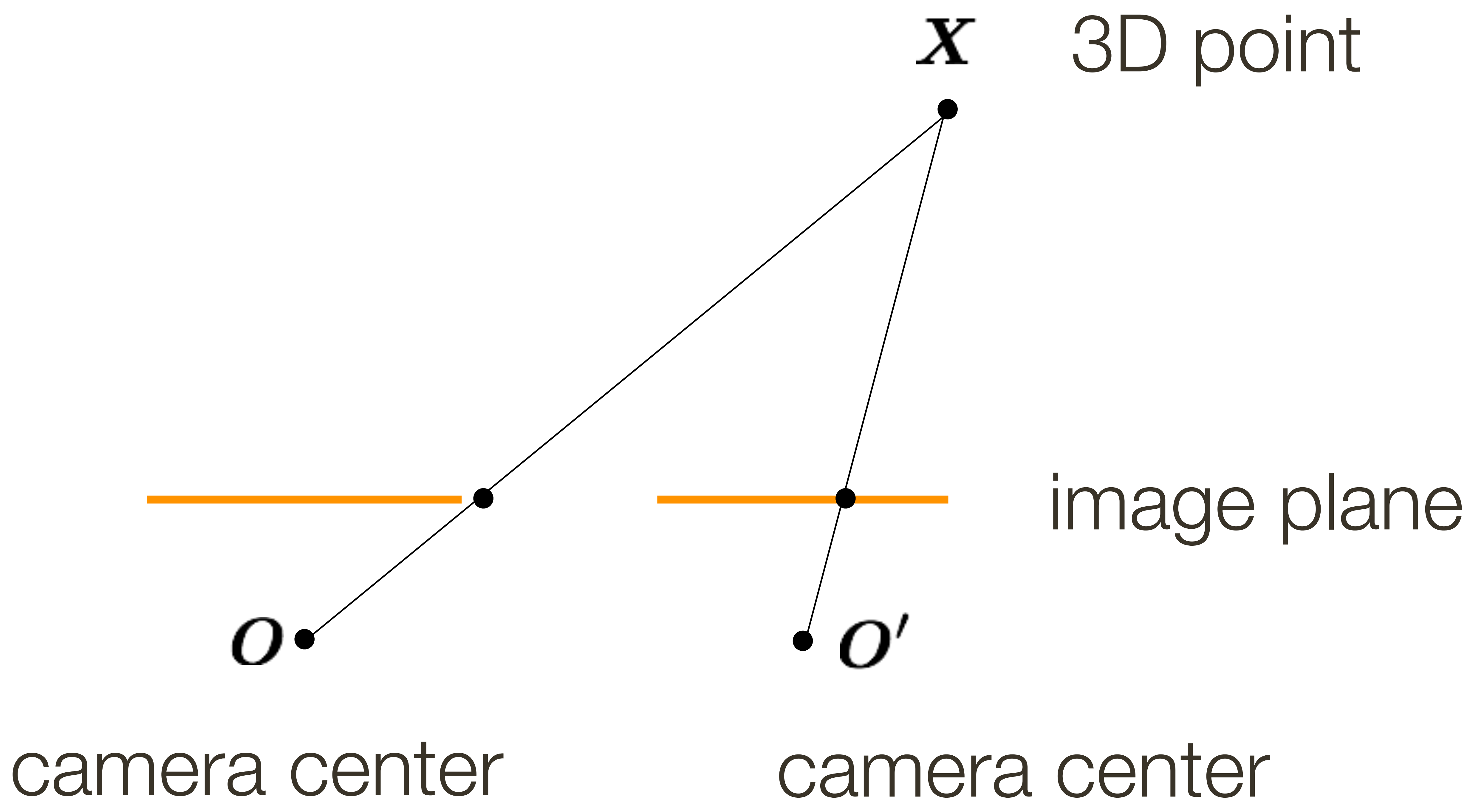




After Rectification

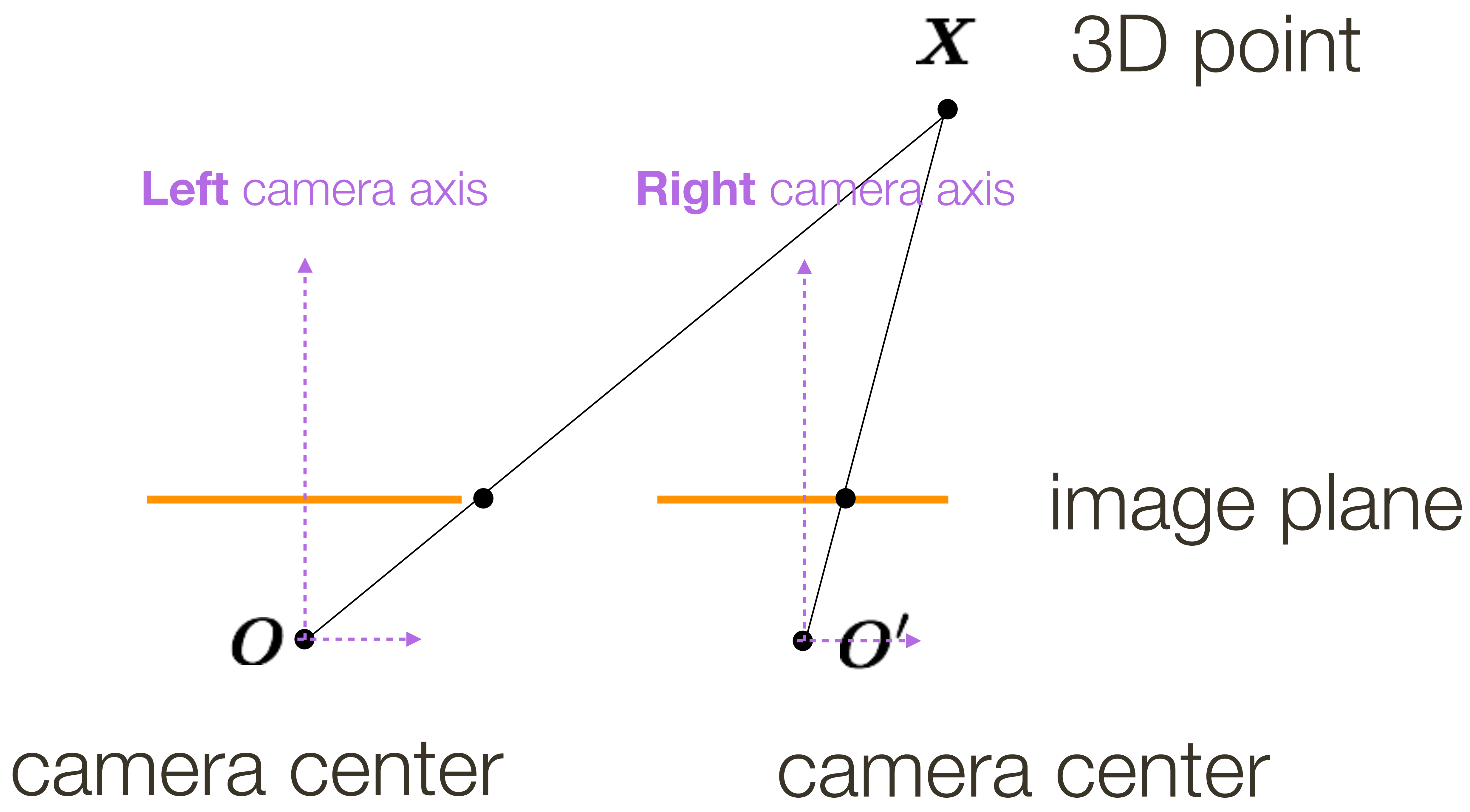
Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Rectified Stereo Pair: Depth Estimate



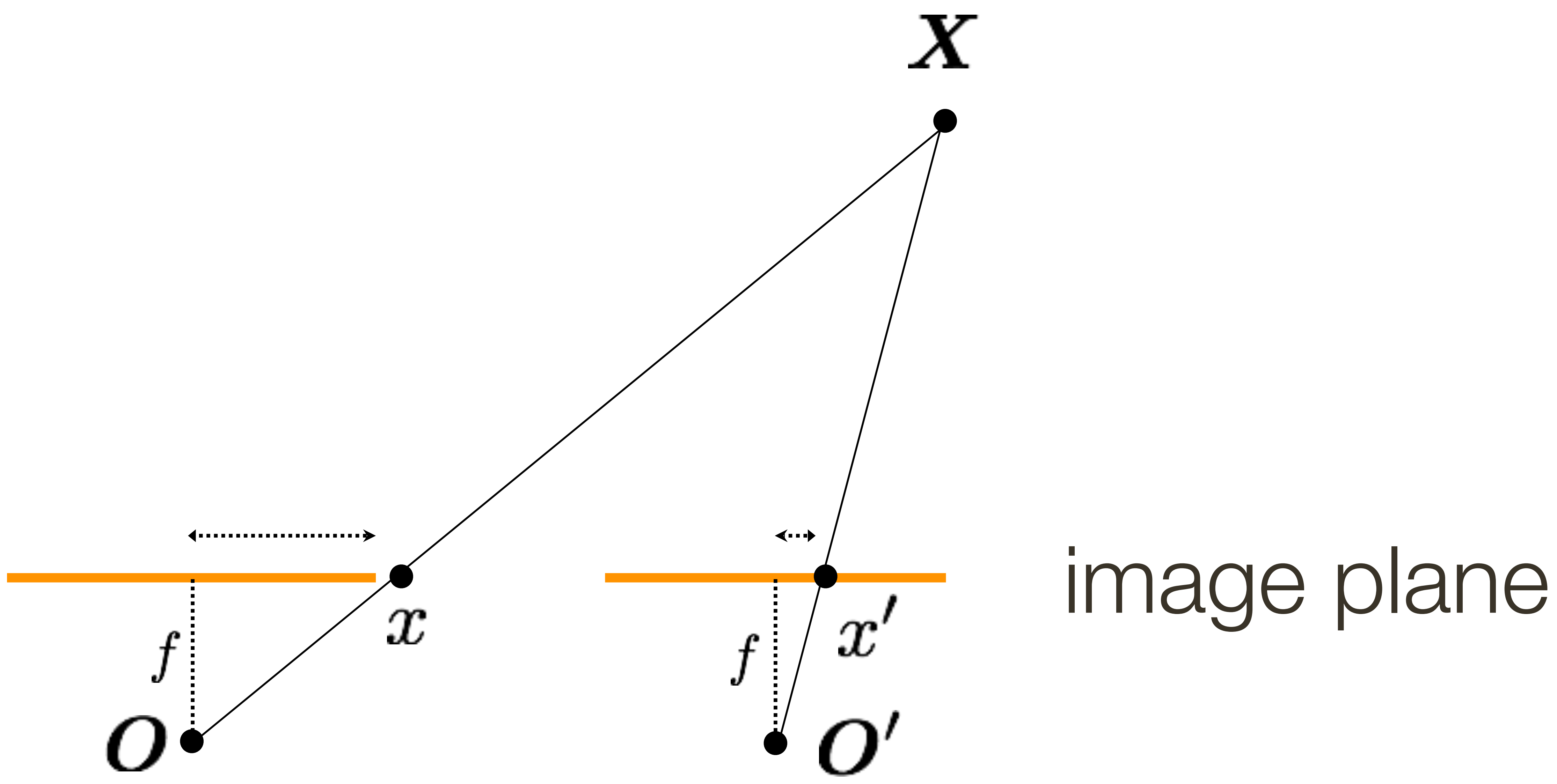
Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Rectified Stereo Pair: Depth Estimate



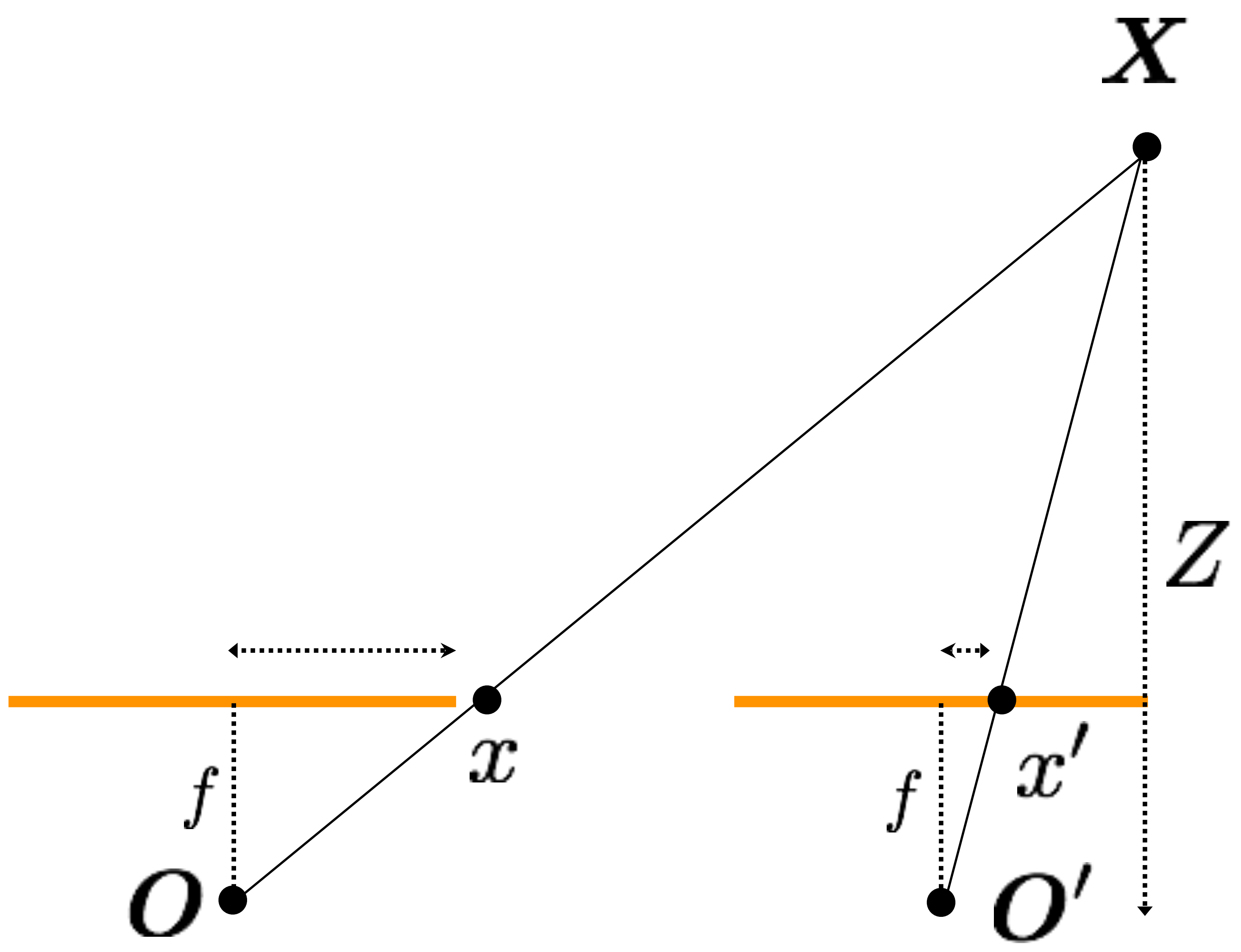
Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Rectified Stereo Pair: Depth Estimate



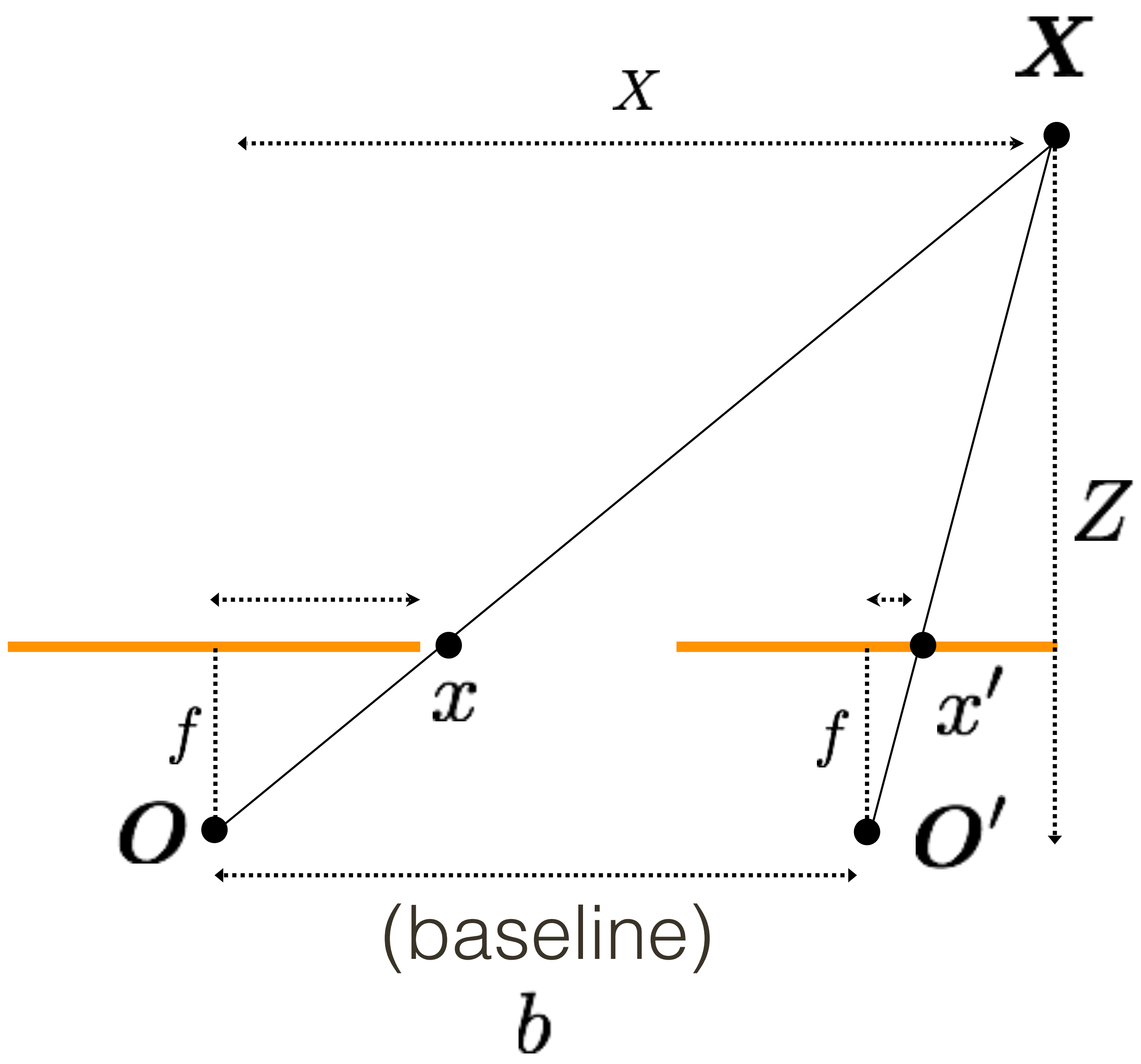
Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Rectified Stereo Pair: Depth Estimate



Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

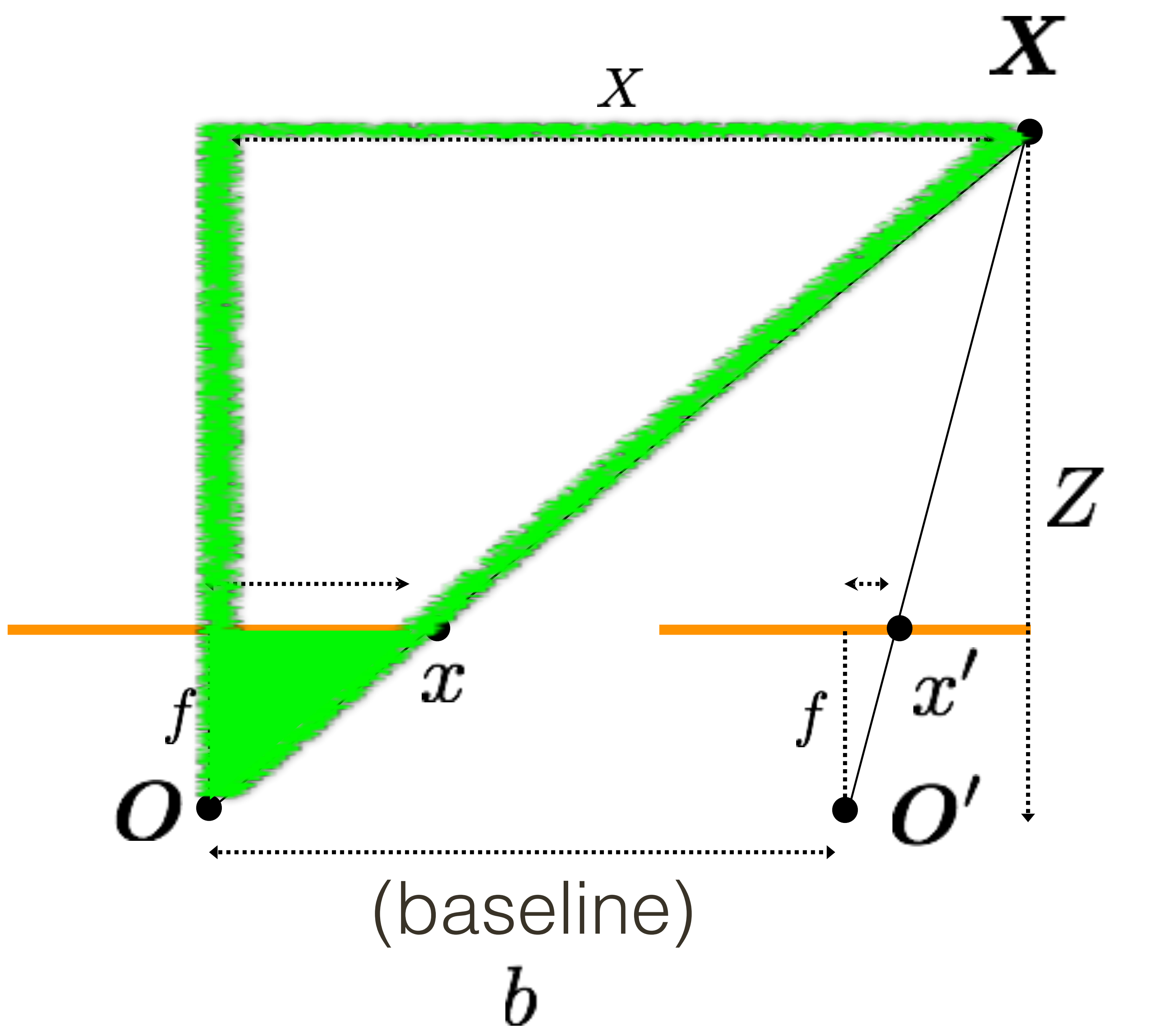
Rectified Stereo Pair: Depth Estimate



Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Rectified Stereo Pair: Depth Estimate

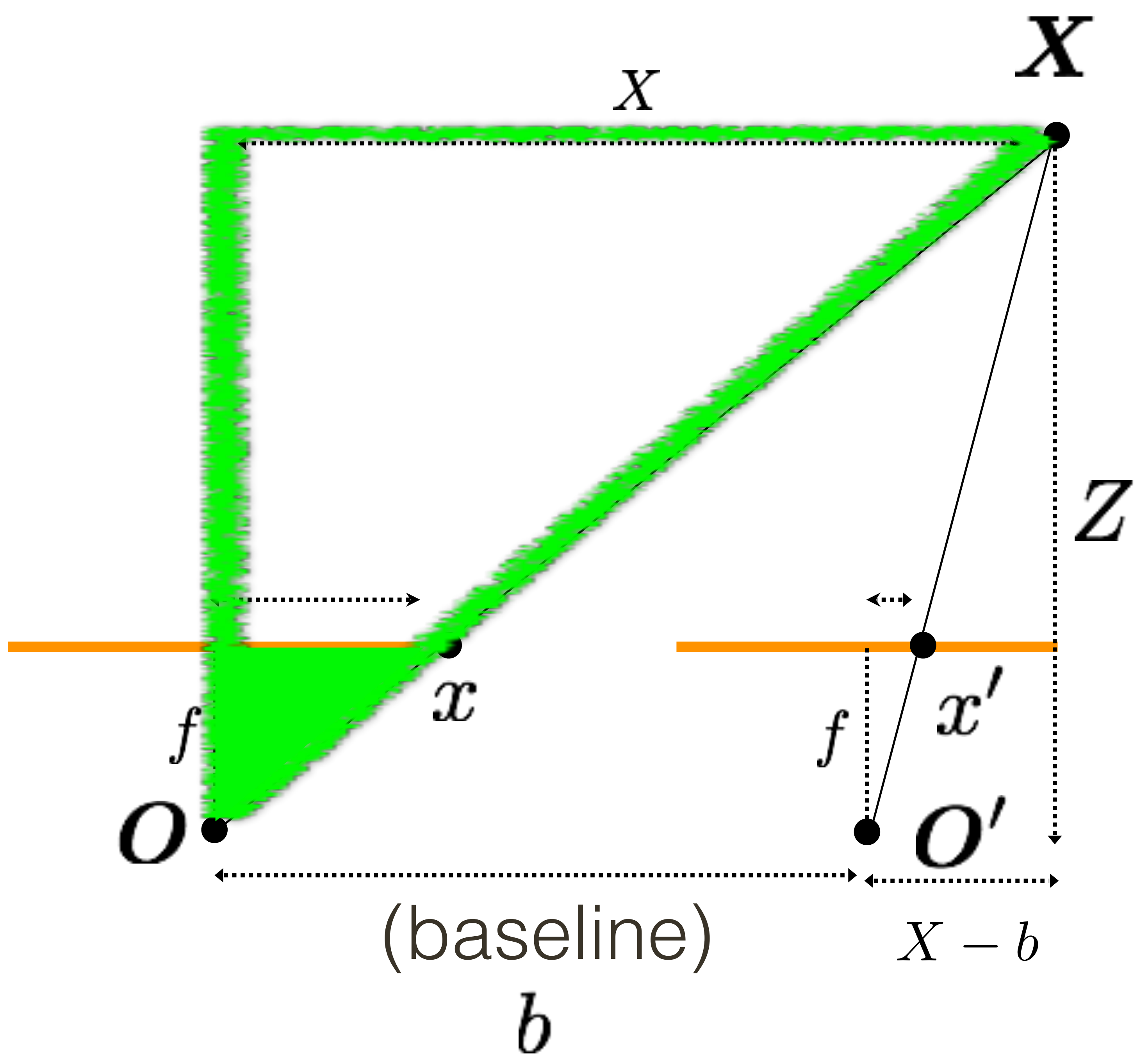
$$\frac{X}{Z} = \frac{x}{f}$$



Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Rectified Stereo Pair: Depth Estimate

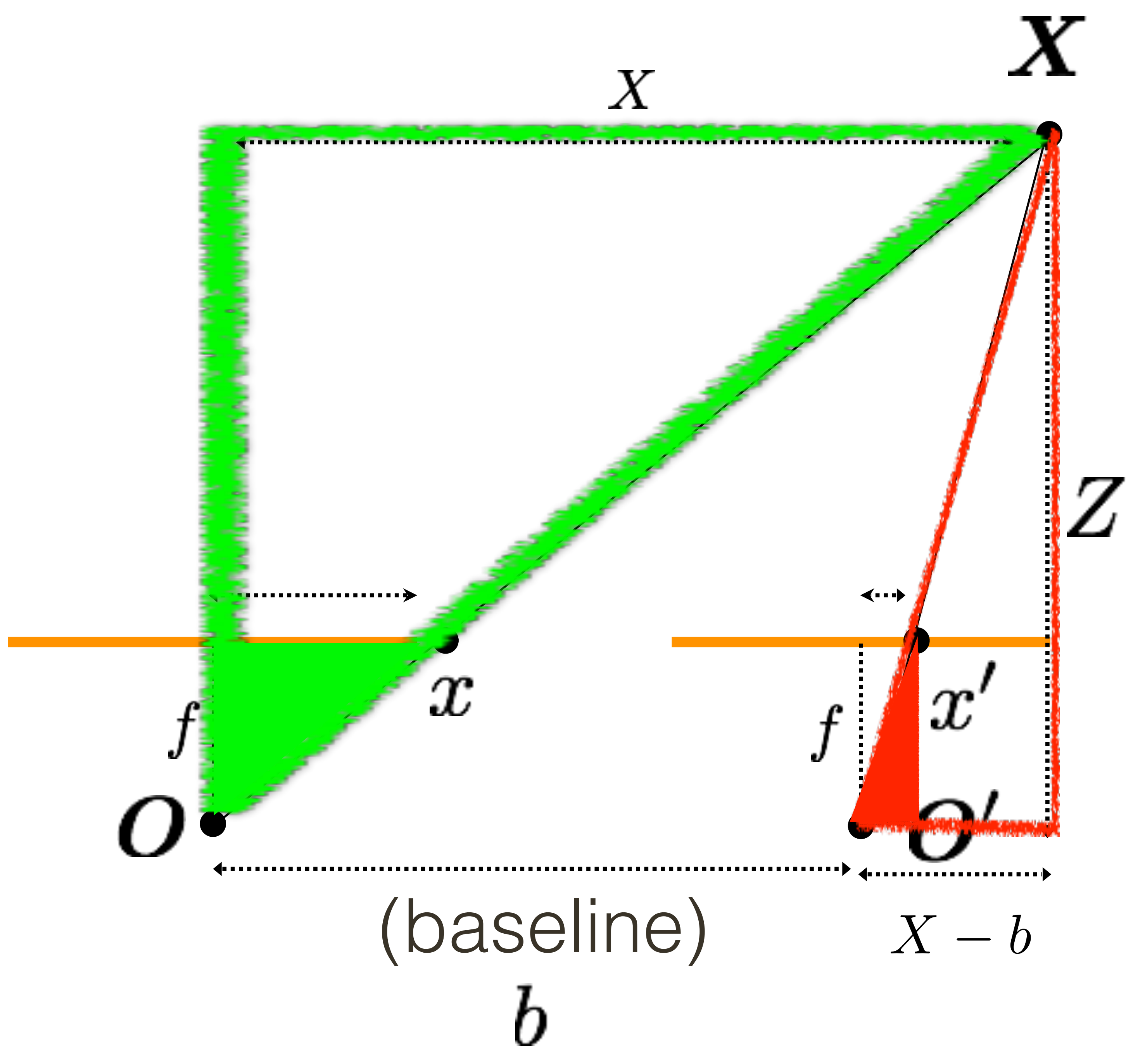
$$\frac{X}{Z} = \frac{x}{f}$$



Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Rectified Stereo Pair: Depth Estimate

$$\frac{X}{Z} = \frac{x}{f}$$

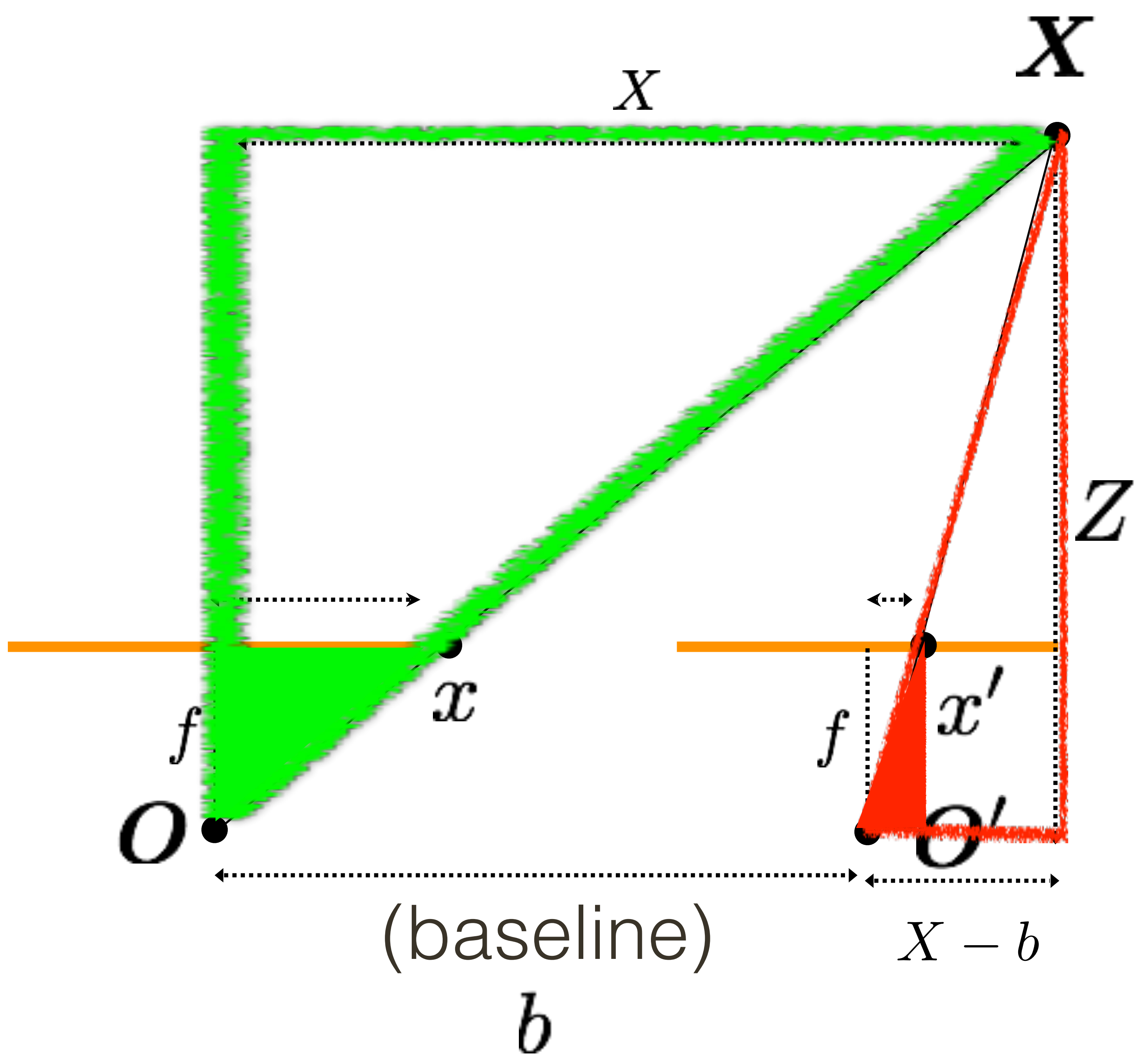


$$\frac{X - b}{Z} = \frac{x'}{f}$$

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Rectified Stereo Pair: Depth Estimate

$$\frac{X}{Z} = \frac{x}{f}$$



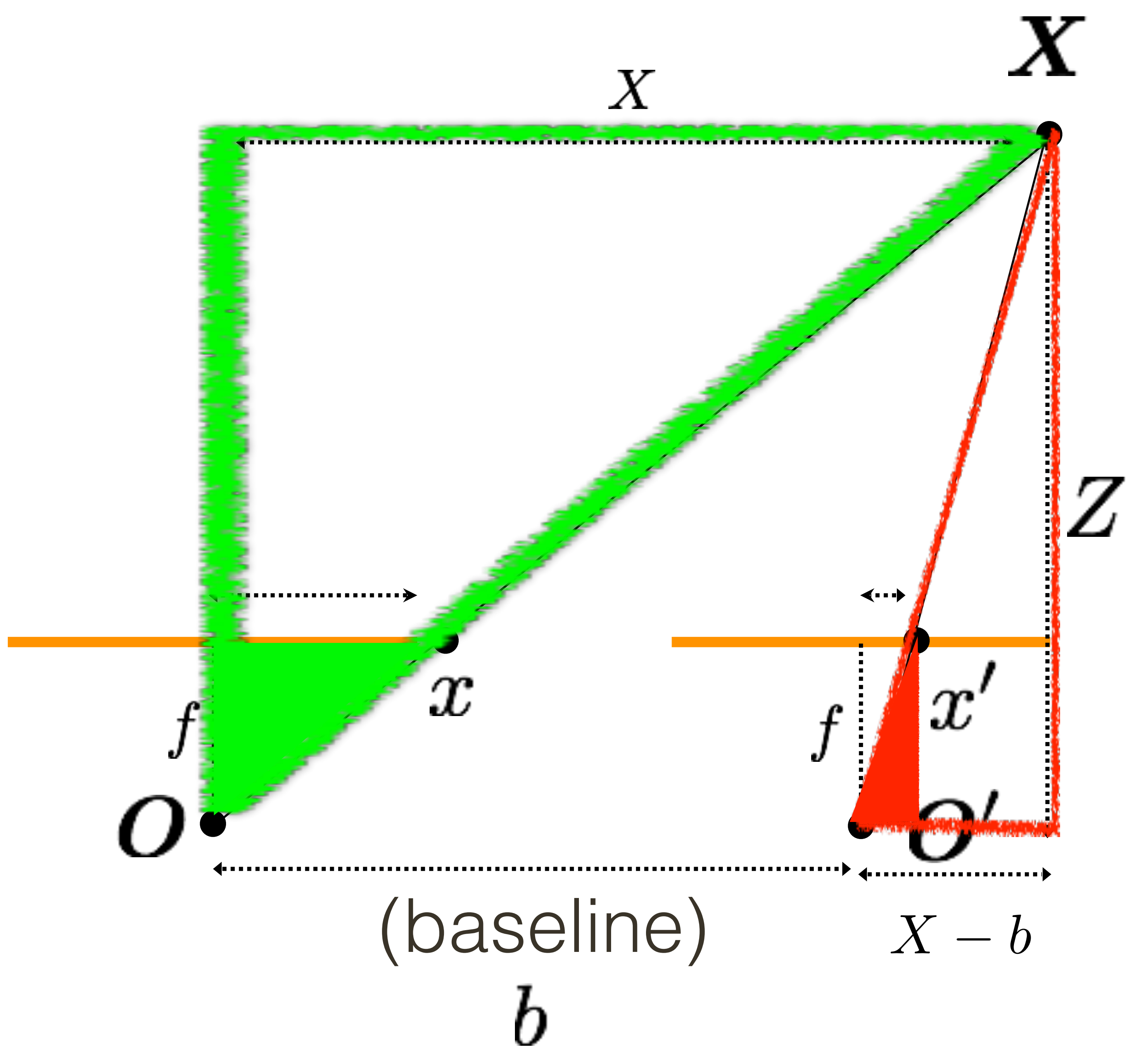
$$\frac{X - b}{Z} = \frac{x'}{f}$$

$$\frac{X}{Z} - \frac{b}{Z} = \frac{x'}{f}$$

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Rectified Stereo Pair: Depth Estimate

$$\frac{X}{Z} = \frac{x}{f}$$



$$\frac{X - b}{Z} = \frac{x'}{f}$$

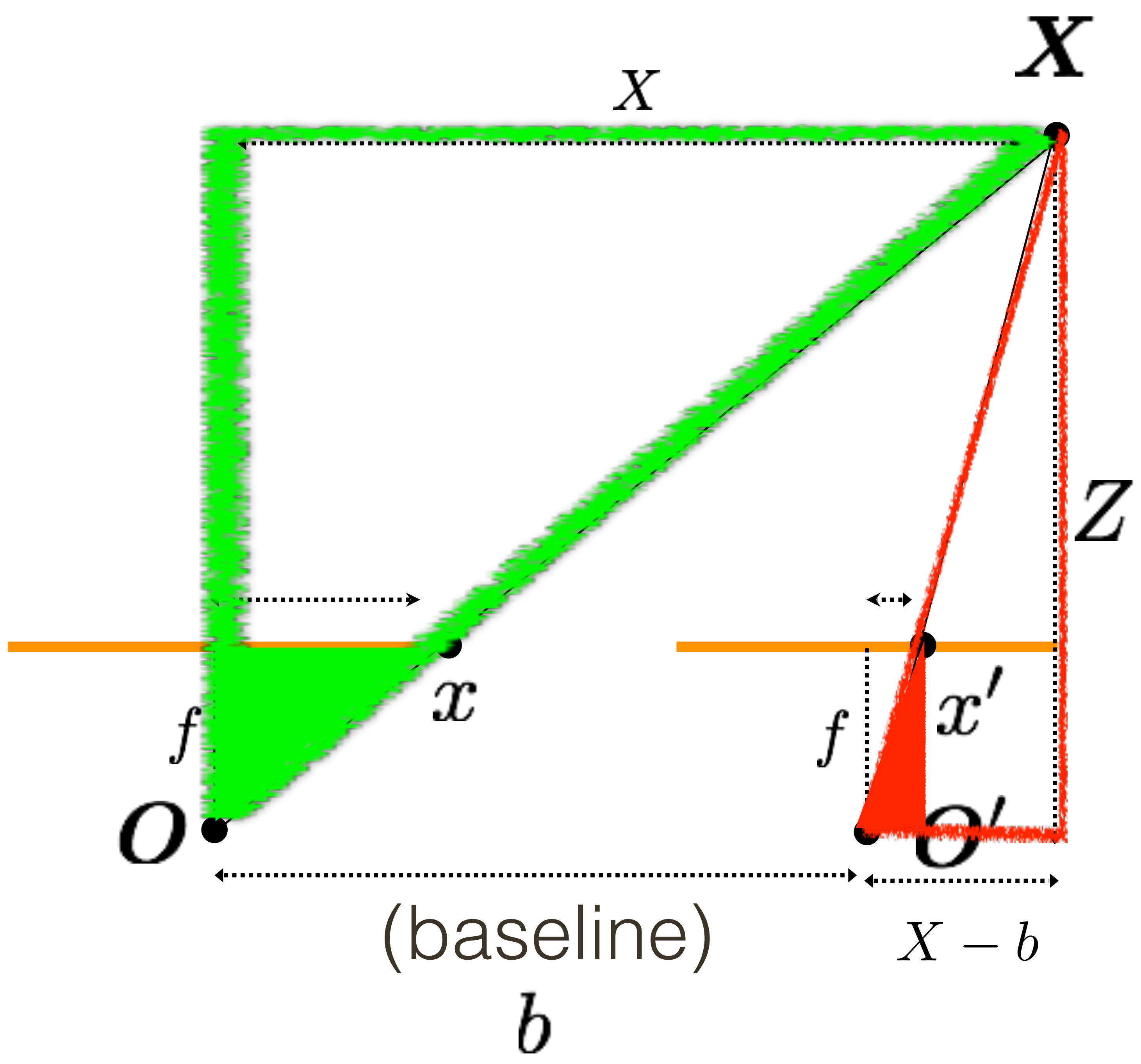
$$\frac{X}{Z} - \frac{b}{Z} = \frac{x'}{f}$$

$$\frac{x}{f} - \frac{b}{Z} = \frac{x'}{f}$$

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Rectified Stereo Pair: Depth Estimate

$$\frac{X}{Z} = \frac{x}{f}$$



$$\frac{X-b}{Z} = \frac{x'}{f}$$

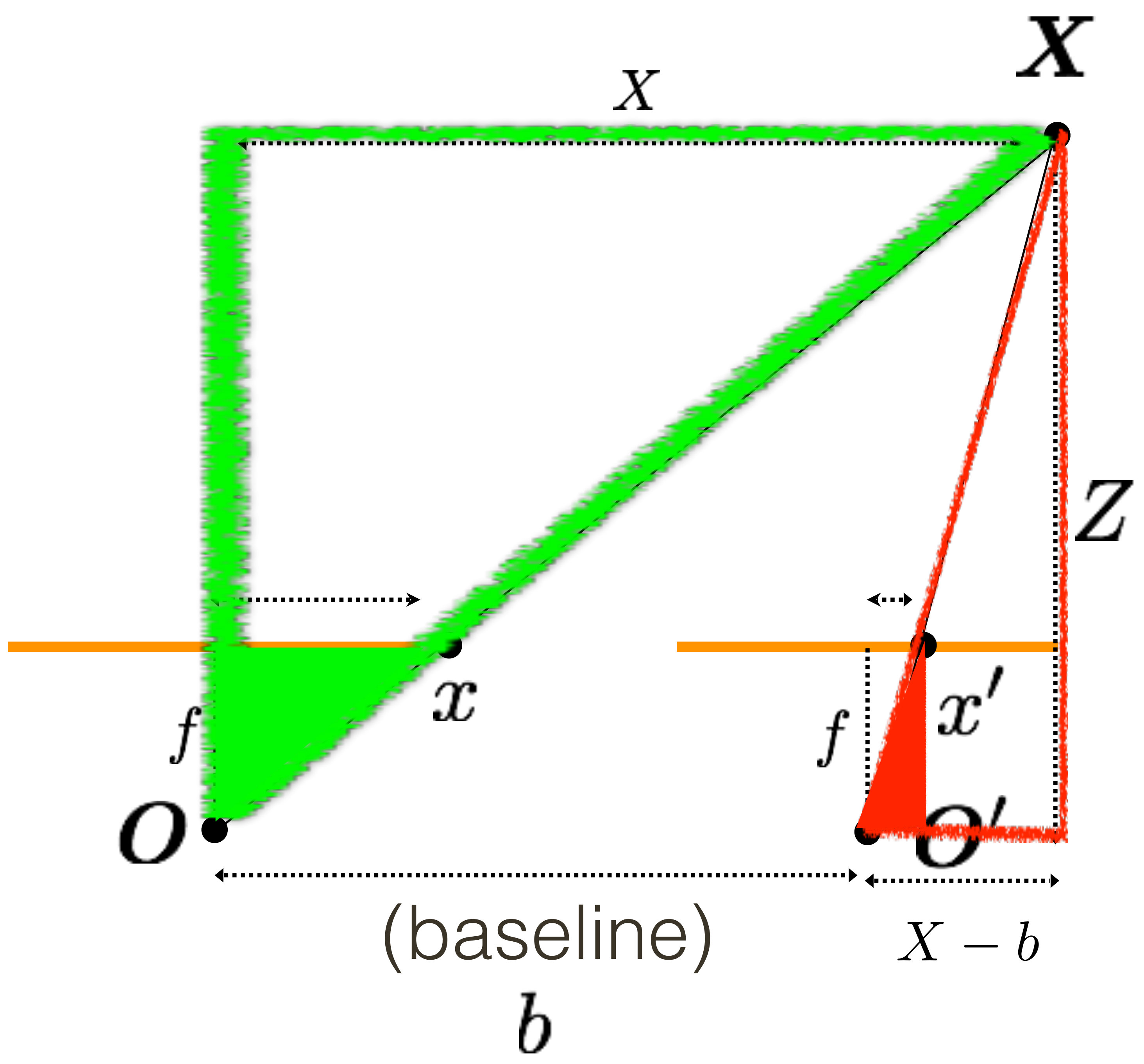
$$\frac{X}{Z} - \frac{b}{Z} = \frac{x'}{f}$$

$$\frac{x}{f} - \frac{b}{Z} = \frac{x'}{f}$$

$$\frac{x-x'}{f} = \frac{b}{Z}$$

Rectified Stereo Pair: Depth Estimate

$$\frac{X}{Z} = \frac{x}{f}$$



$$\frac{X - b}{Z} = \frac{x'}{f}$$

$$\frac{X}{Z} - \frac{b}{Z} = \frac{x'}{f}$$

$$\frac{x}{f} - \frac{b}{Z} = \frac{x'}{f}$$

$$\frac{x - x'}{f} = \frac{b}{Z}$$

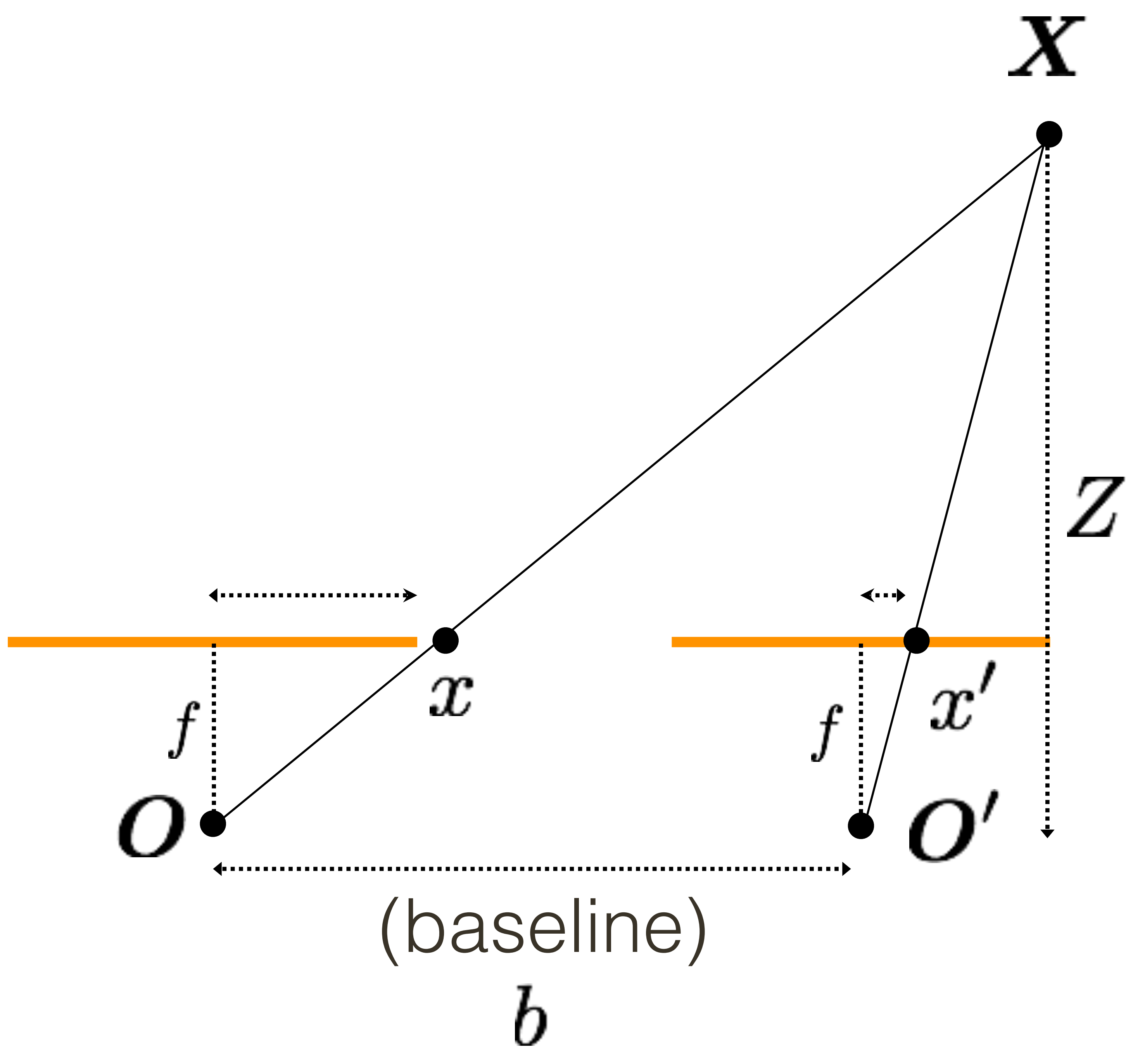
Disparity

(wrt to camera origin of image plane)

$$d = x - x'$$
$$= \frac{bf}{Z}$$

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Rectified Stereo Pair: Depth Estimate



Disparity

(wrt to camera origin of image plane)

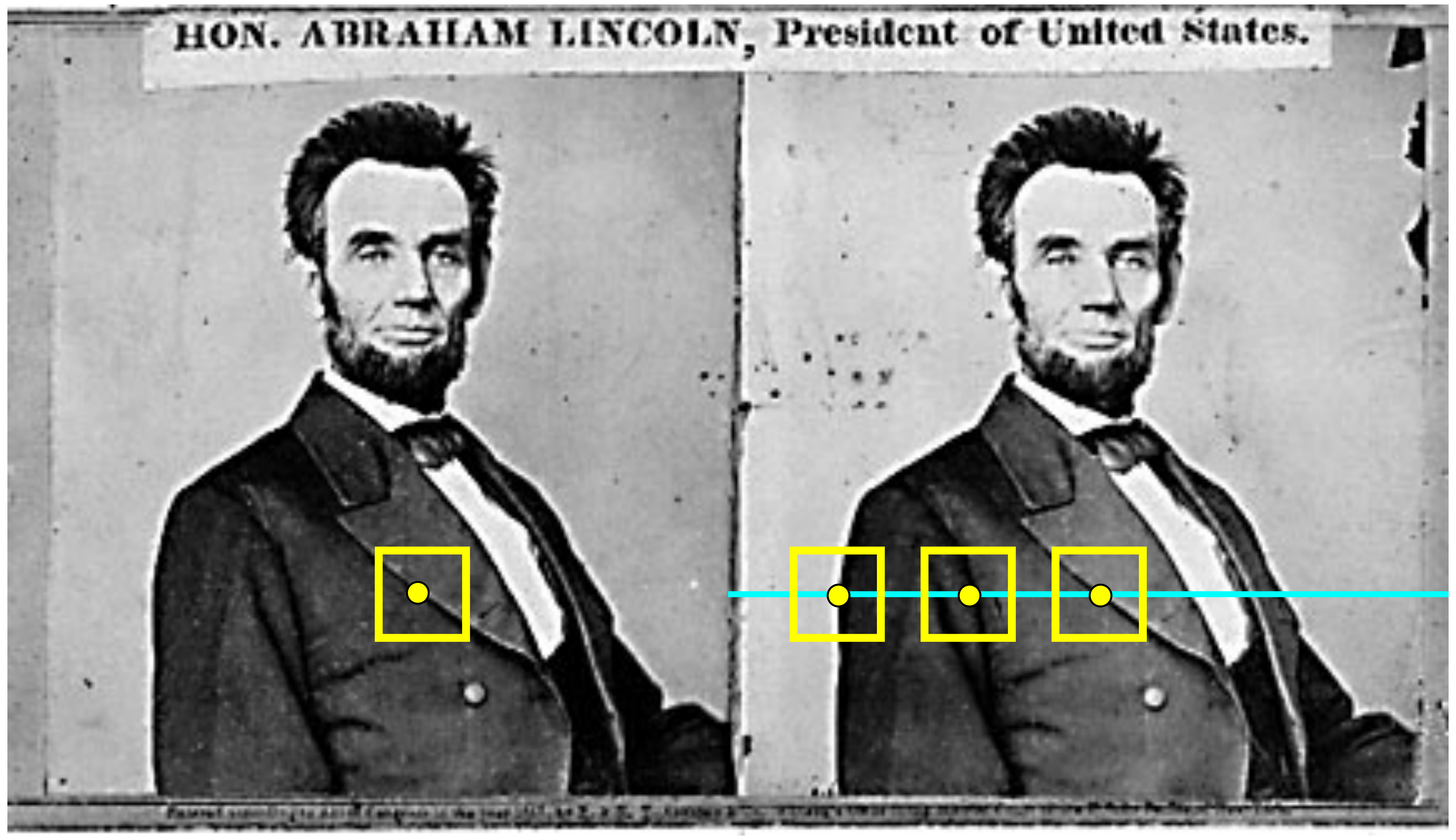
$$d = x - x'$$

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

inversely proportional to depth

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

(simple) **Stereo** Algorithm

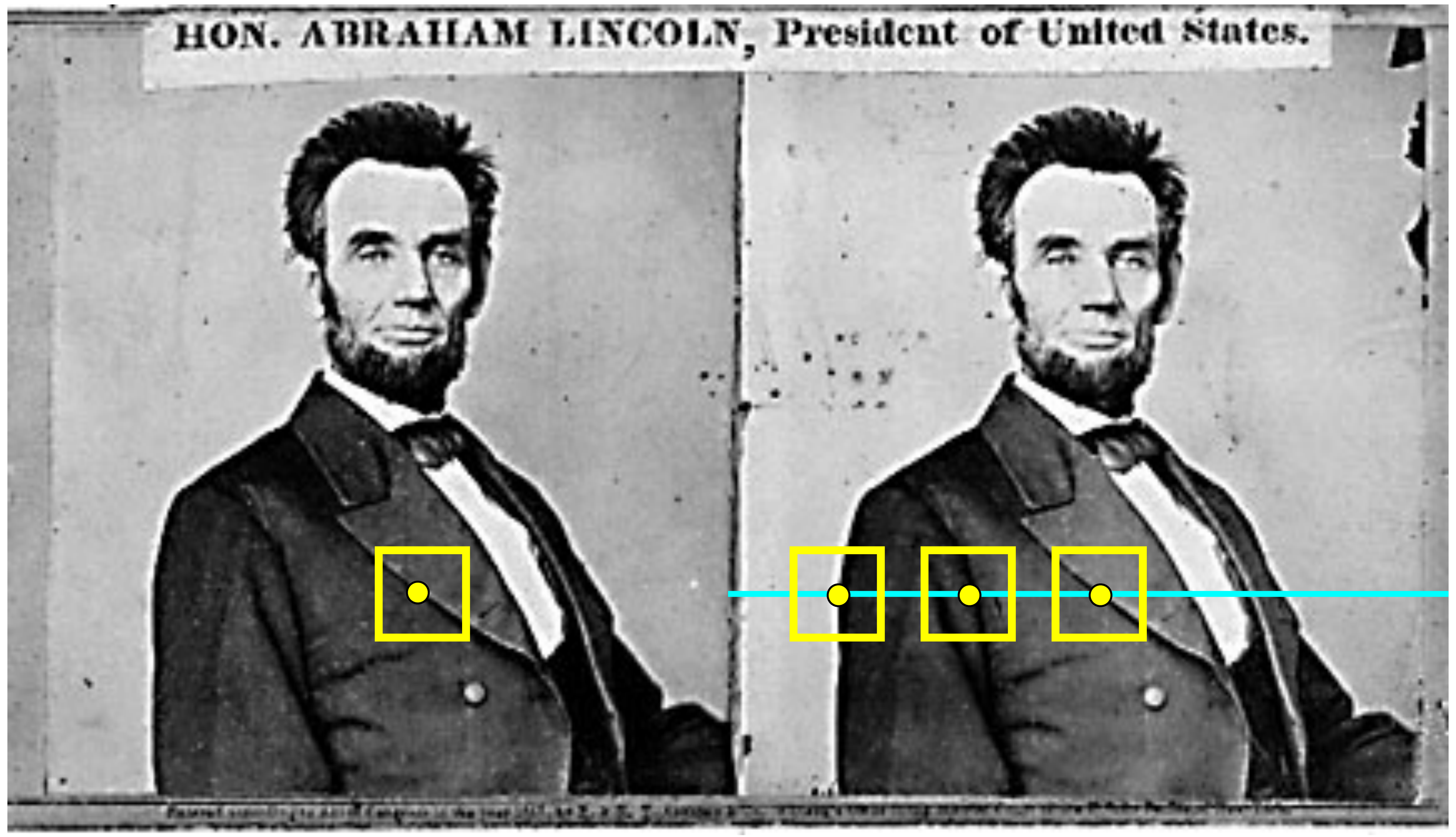



```
1.Rectify images
  (make epipolar lines horizontal)
2.For each pixel
  a.Find epipolar line
  b.Scan line for best match
  c.Compute depth from disparity
```

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

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

(simple) **Stereo** Algorithm



1. Rectify images
   (make epipolar lines horizontal)
2. For each pixel
   a. Find epipolar line
   b. Scan line for best match
   c. Compute depth from disparity $Z = \dfrac{bf}{d}$

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Correspondence: What should we match?

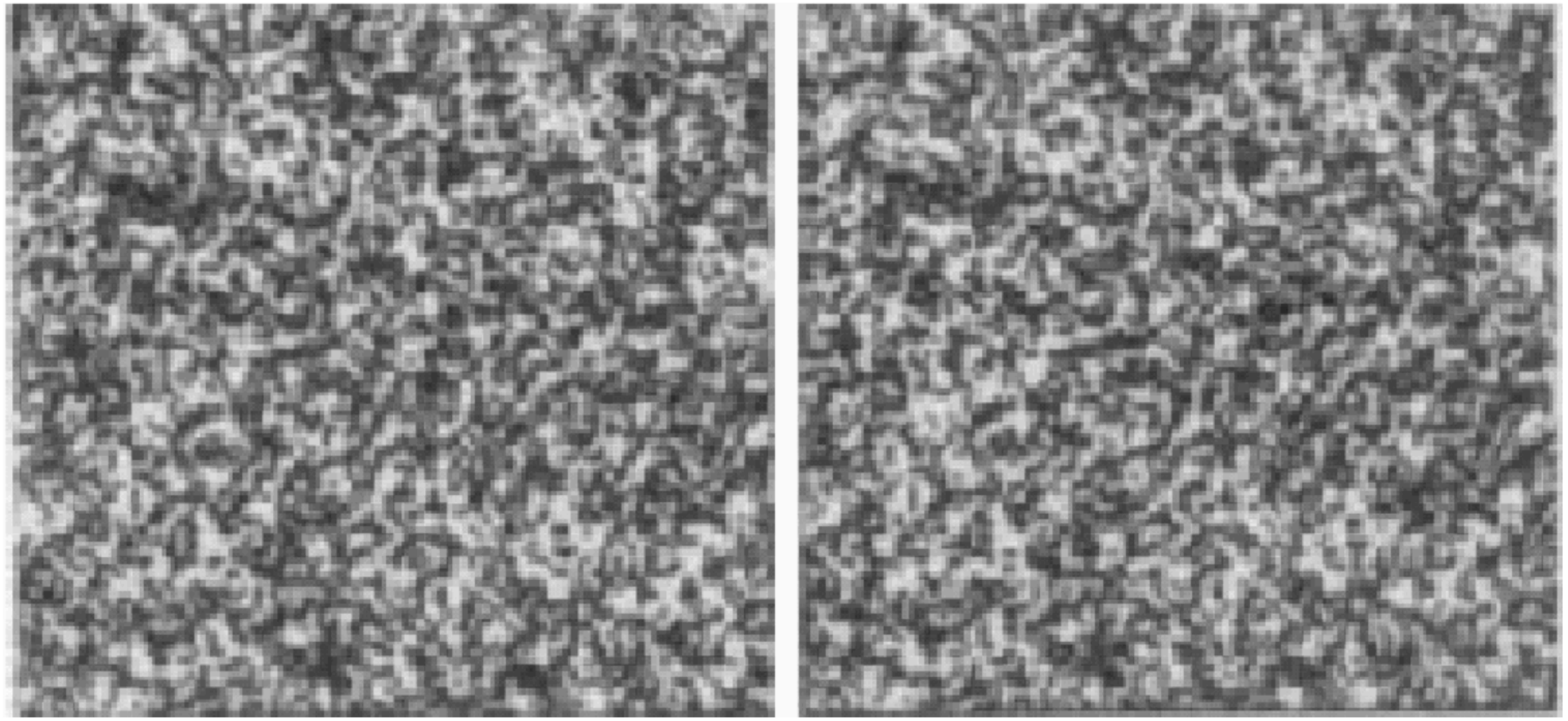
Objects?

Edges?

Pixels?

Collections of pixels?

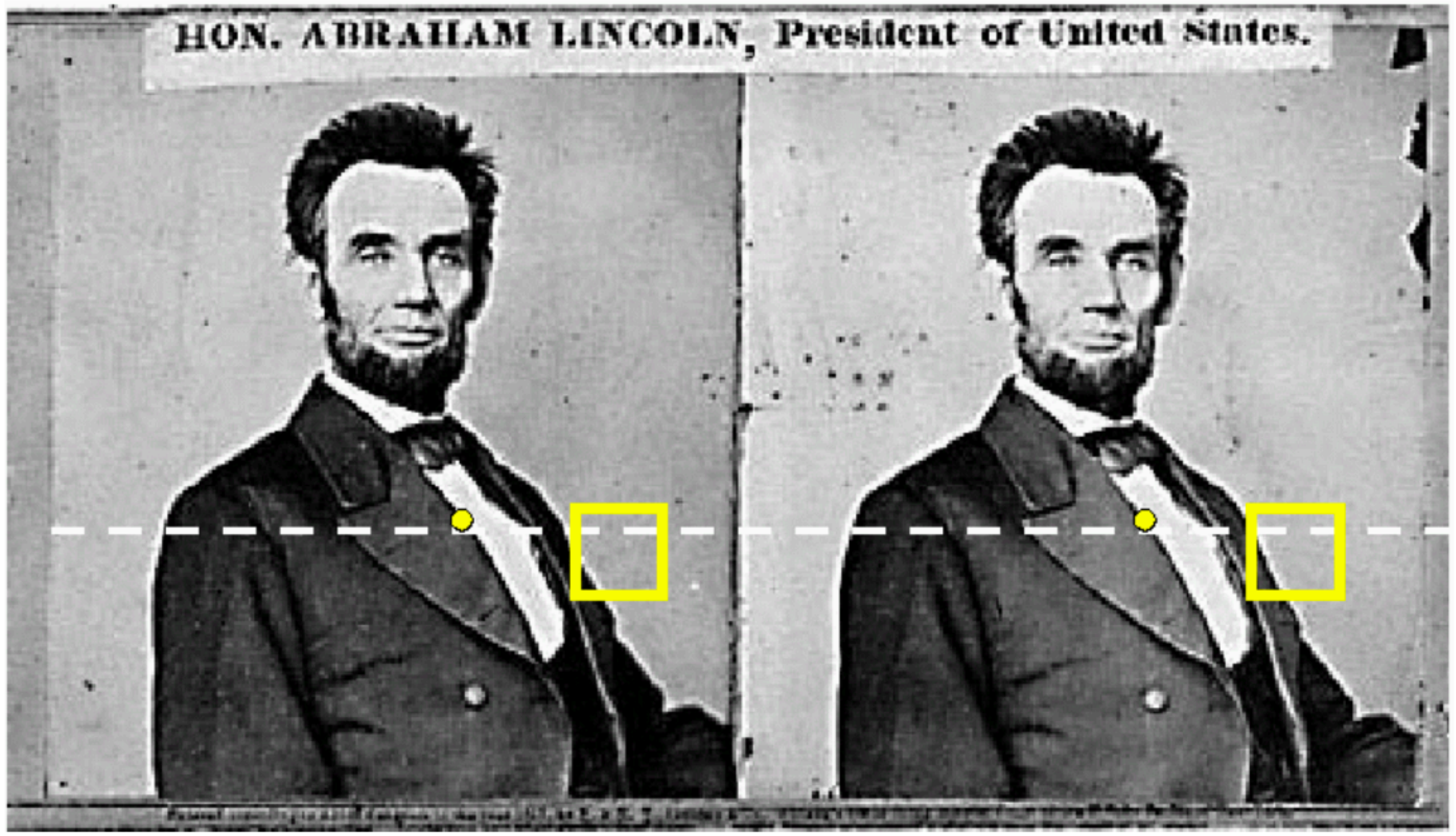
Random Dot Stereograms



Julesz (1960) showed that **recognition is not needed** for stereo

"When viewed monocularly, the images appear completely random. But when viewed stereoscopically, the image pair gives the impression of a square markedly in front of (or behind) the surround."

Method: Pixel Matching



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

This leaves too much ambiguity!

Slide credit: Steve Seitz

Sum of Squared (Pixel) Differences

$\mathbf{w}_L$ and $\mathbf{w}_R$ are corresponding $m \times m$ windows of pixels

Define the window function, $\mathbf{W}_m(x,y)$, by

$$\mathbf{W}_m(x,y) = \left\{(u,v) \mid x - \frac{m}{2} \le u \le x + \frac{m}{2}, y - \frac{m}{2} \le v \le y + \frac{m}{2}\right\}$$

SSD measures intensity difference as a function of disparity:

$$C_R(x,y,d) = \sum_{(u,v)\in\mathbf{W}_m(x,y)} [I_L(u,v) - I_R(u-d,v)]^2$$

Image Normalization

$$\bar{I} = \frac{1}{|\mathbf{W}_m(x,y)|} \sum_{(u,v)\in\mathbf{W}_m(x,y)} I(u,v)$$

Average Pixel

$$||I||_{\mathbf{W}_m(x,y)} = \sqrt{\sum_{(u,v)\in\mathbf{W}_m(x,y)} [I(u,v)]^2}$$

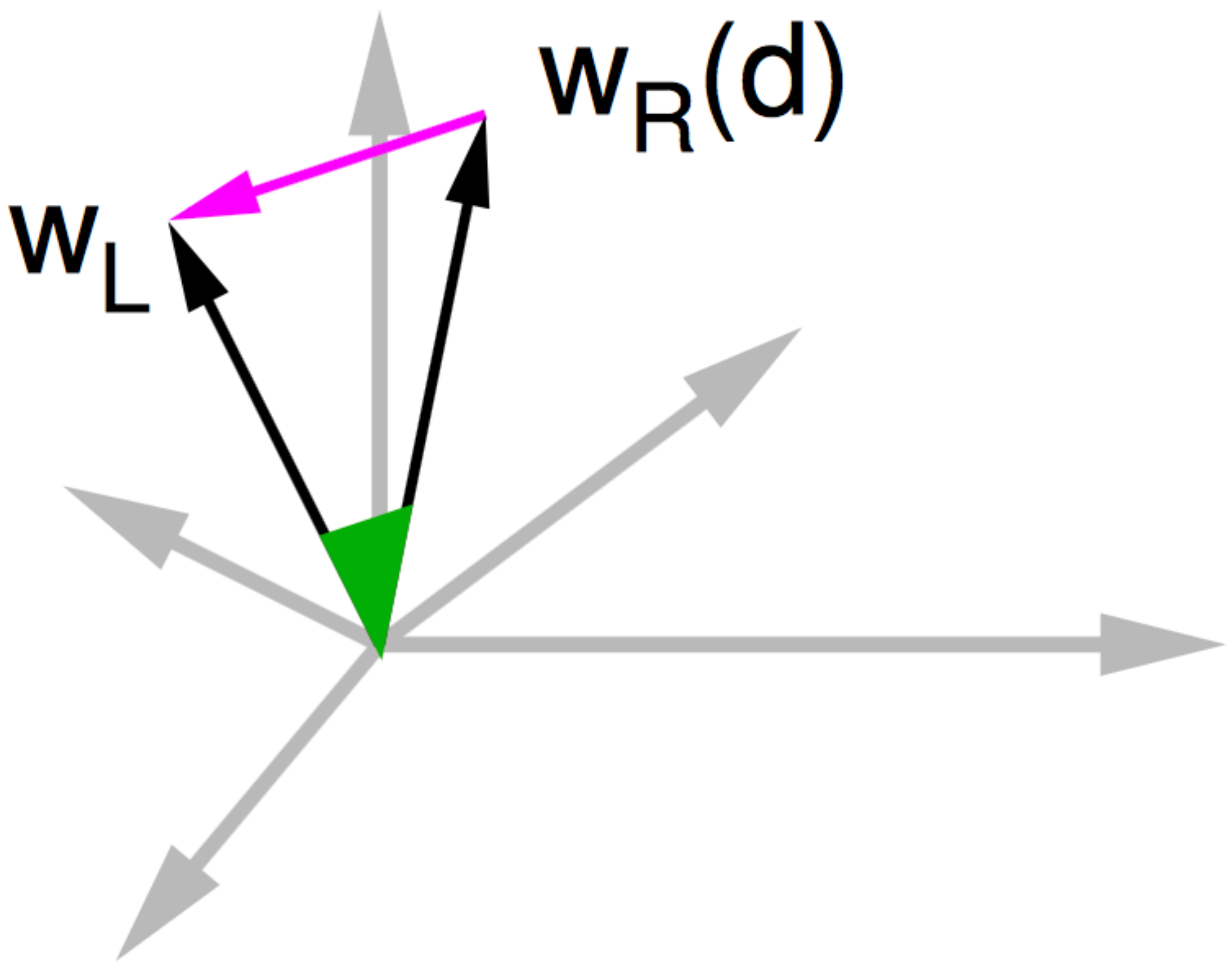
Window Magnitude

$$\hat{I}(x,y) = \frac{I(x,y) - \bar{I}}{||I - \bar{I}||_{\mathbf{W}_m(x,y)}}$$

Normalized Pixel: subtract the mean, normalize to unit length

Image **Metrics**

(Normalized) Sum of Squared Differences



(Normalized) Correlation

Image **Metrics**

Assume $\mathbf{w}_L$ and $\mathbf{w}_R(d)$ are normalized to unit length (Normalized)

Sum of Squared Differences:

$$C_{SSD}(d) = \sum_{(u,v)\in\mathbf{W}_m(x,y)} \left[\hat{I}_L(u,v) - \hat{I}_R(u-d,v)\right]^2$$

$$= ||\mathbf{w}_L - \mathbf{w}_R(d)||^2$$

(Normalized) **Correlation**:

$$C_{NC}(d) = \sum_{(u,v)\in\mathbf{W}_m(x,y)} \hat{I}_L(u,v)\hat{I}_R(u-d,v)$$

$$= \mathbf{w}_L \cdot \mathbf{w}_R(d) = \cos\theta$$

Image **Metrics**

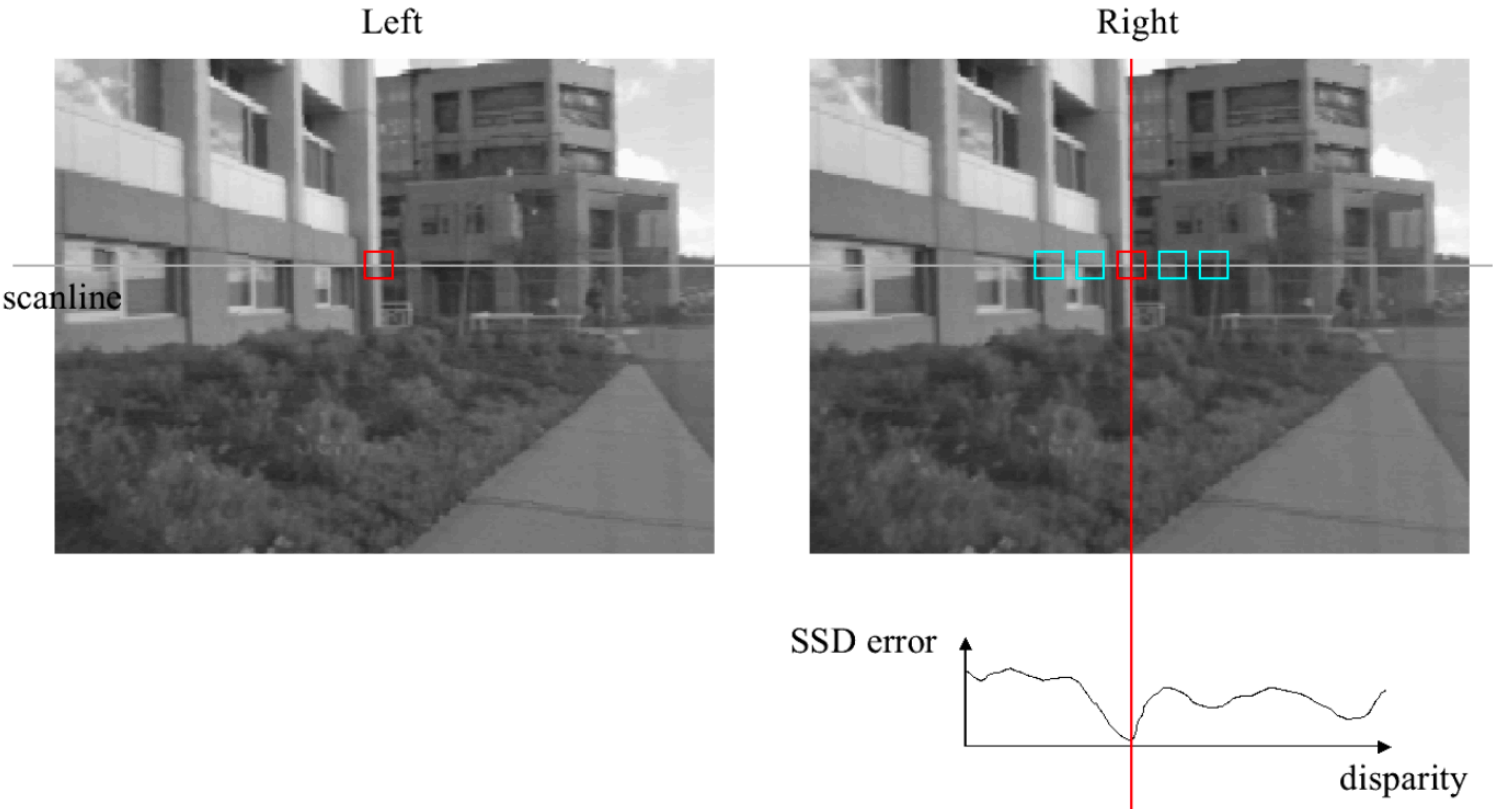
Let d^* be the value of d that minimizes C_{SSD}

Then d^* also is the value of d that minimizes C_{NC}

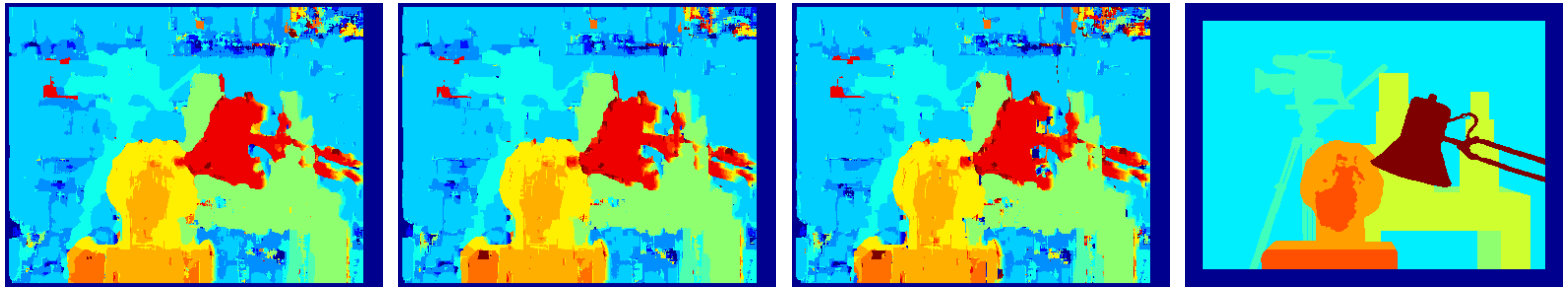
That is,

$$d^* = \arg\min_d ||\mathbf{w}_L - \mathbf{w}_R(d)||^2 = \arg\min_d \mathbf{w}_L \cdot \mathbf{w}_R(d)$$

Method: Correlation



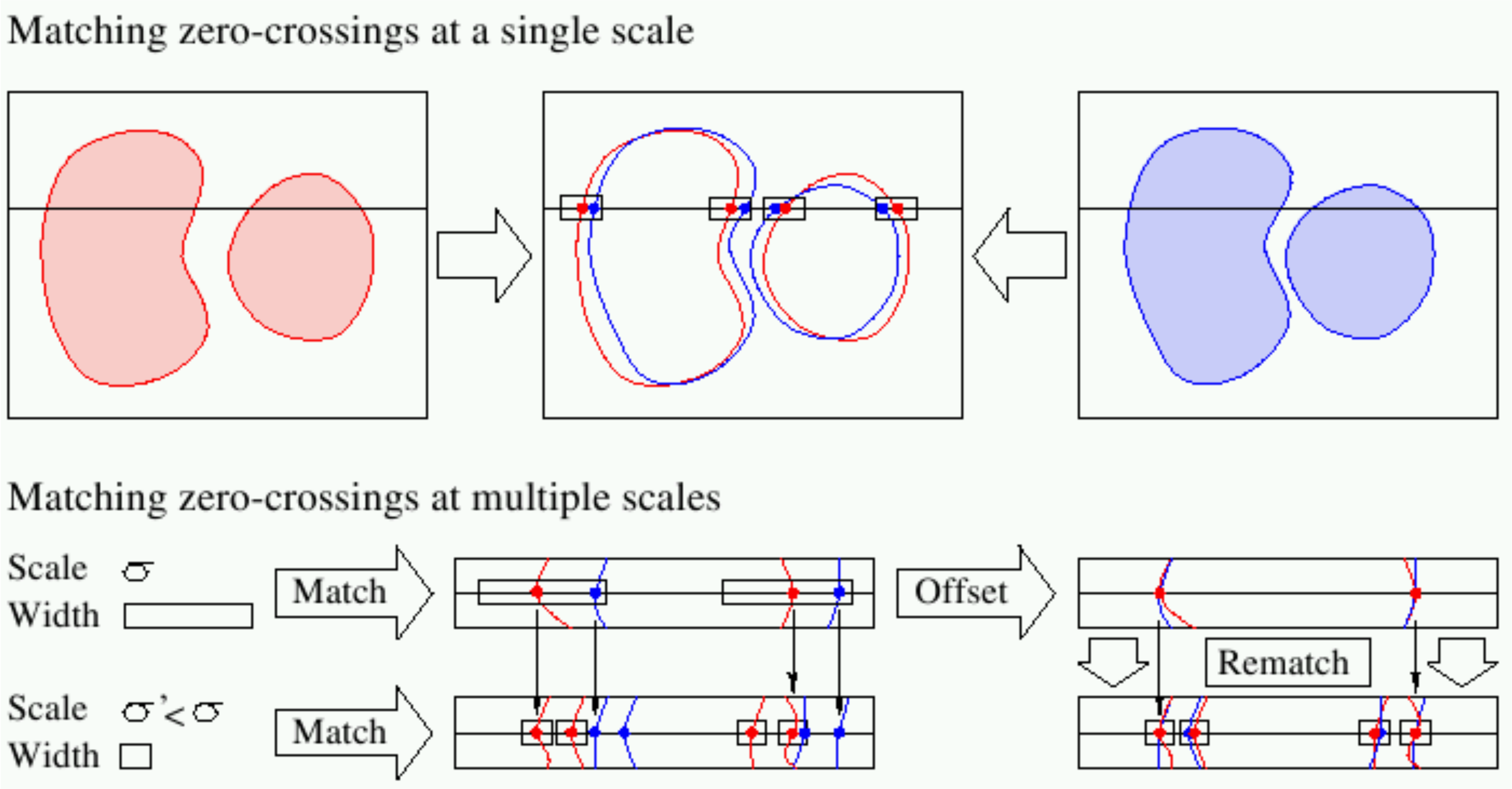
Similarity Measure	Formula
Sum of Absolute Differences (SAD)	$\sum_{(i,j)\in W} \lvert I_1(i,j) - I_2(x+i, y+j)\rvert$
Sum of Squared Differences (SSD)	$\sum_{(i,j)\in W} \left(I_1(i,j) - I_2(x+i, y+j)\right)^2$
Zero-mean SAD	$\sum_{(i,j)\in W} \lvert I_1(i,j) - \bar{I}_1(i,j) - I_2(x+i,y+j) + \bar{I}_2(x+i,y+j)\rvert$
Locally scaled SAD	$\sum_{(i,j)\in W} \lvert I_1(i,j) - \frac{\bar{I}_1(i,j)}{\bar{I}_2(x+i,y+j)} I_2(x+i,y+j)\rvert$
Normalized Cross Correlation (NCC)	$\frac{\sum_{(i,j)\in W} I_1(i,j).I_2(x+i,y+j)}{\sqrt[2]{\sum_{(i,j)\in W} I_1^2(i,j).\sum_{(i,j)\in W} I_2^2(x+i,y+j)}}$



SAD SSD NCC Ground truth

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Method: Edges



Forsyth & Ponce (2nd ed.) Figure 7.12 (Top & Middle)

Method: Edges (aside)

The **Marr/Poggio** (1979) multiscale stereo algorithm:

1. Convolve the two (rectified) images with $\nabla^2 G_\sigma$ filters of increasing $\sigma_1 < \sigma_2 < \sigma_3 < \sigma_4$

2. Find zero crossings along horizontal scanlines of the filtered images

3. For each filter scale σ, match zero crossings with the same parity and roughly equal orientations in a $[-\mathbf{w}_\sigma, +\mathbf{w}_\sigma]$ disparity range, with $\mathbf{w}_\sigma = 2\sqrt{2}\sigma$

4. Use the disparities found at larger scales to control eye vergence and cause unmatched regions at smaller scales to come into correspondence

Which Method is **Better**: Correlation or Edges?

Edges are more "meaningful" [Marr]. . . . . . but hard to find!

Edges tend to fail in dense texture (outdoors)

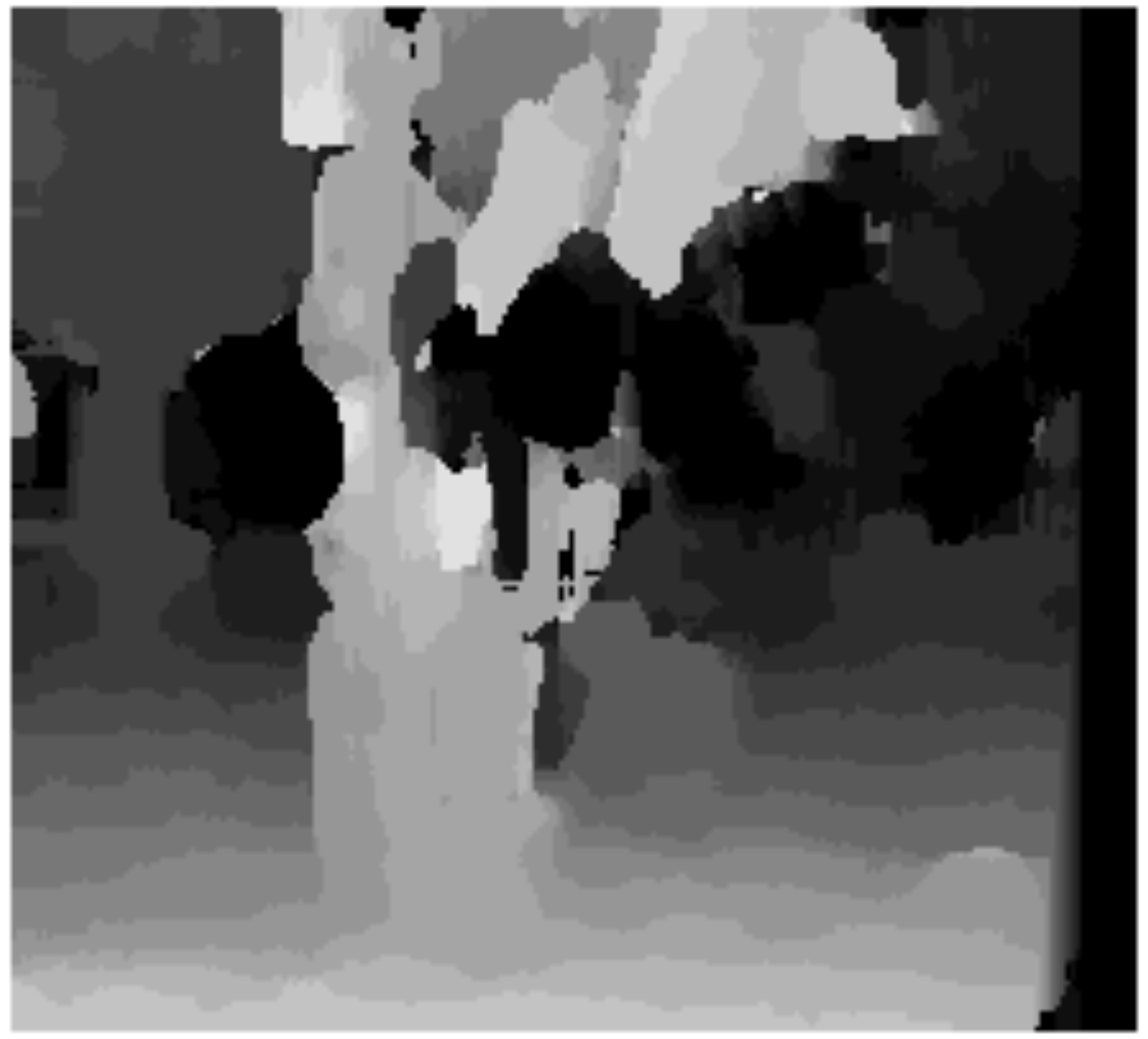
Correlation tends to fail in smooth, featureless regions

Note: Correlation-based methods are "dense." Edge-based methods are "relatively sparse"

Effect of **Window Size**

W = 3

Smaller window
- \+ More detail
- \- More noise



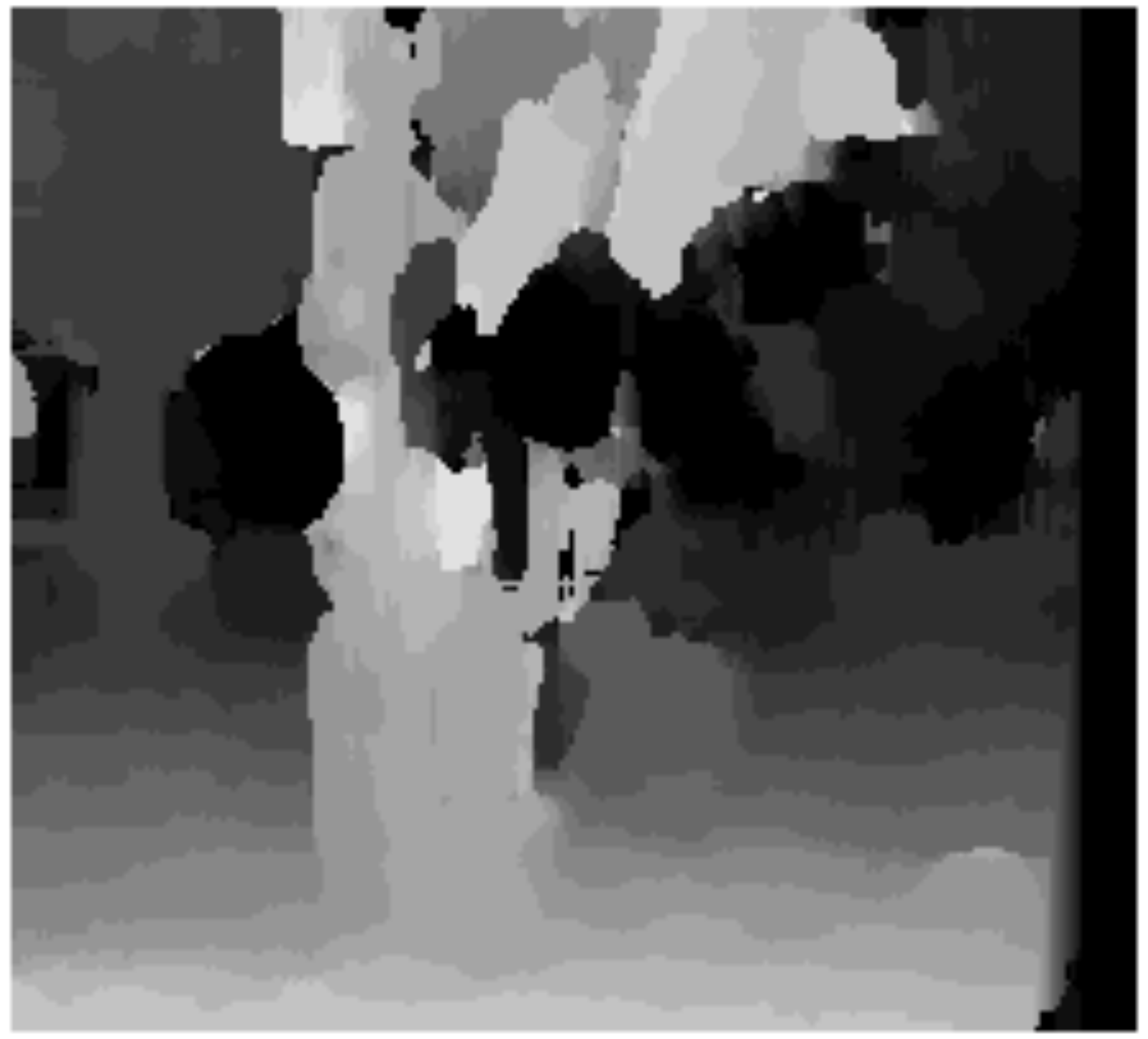
W = 20

Larger window
- \+ Smoother disparity maps
- \- Less detail
- \- Fails near boundaries

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Effect of **Window Size**

W = 3

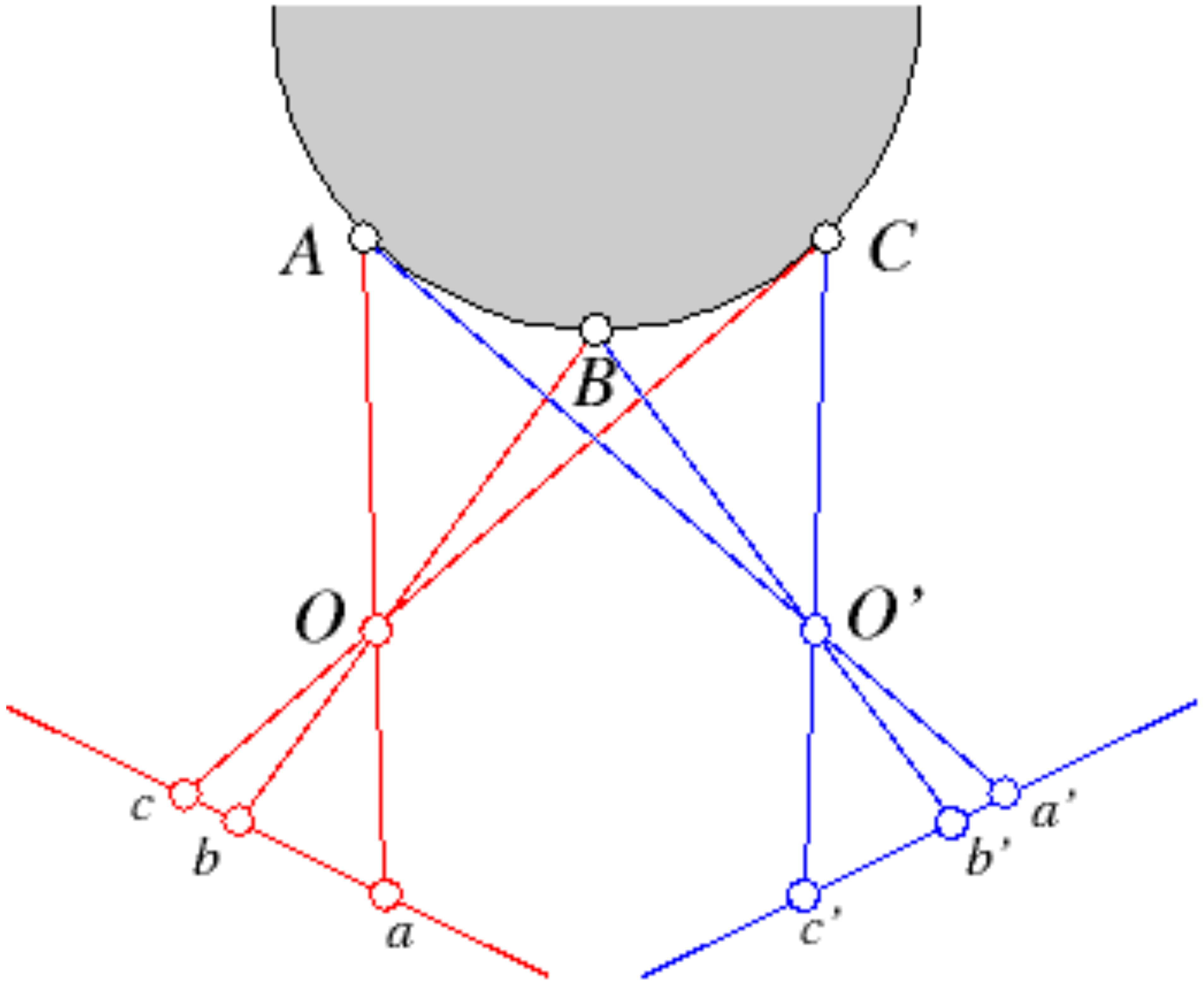


W = 20

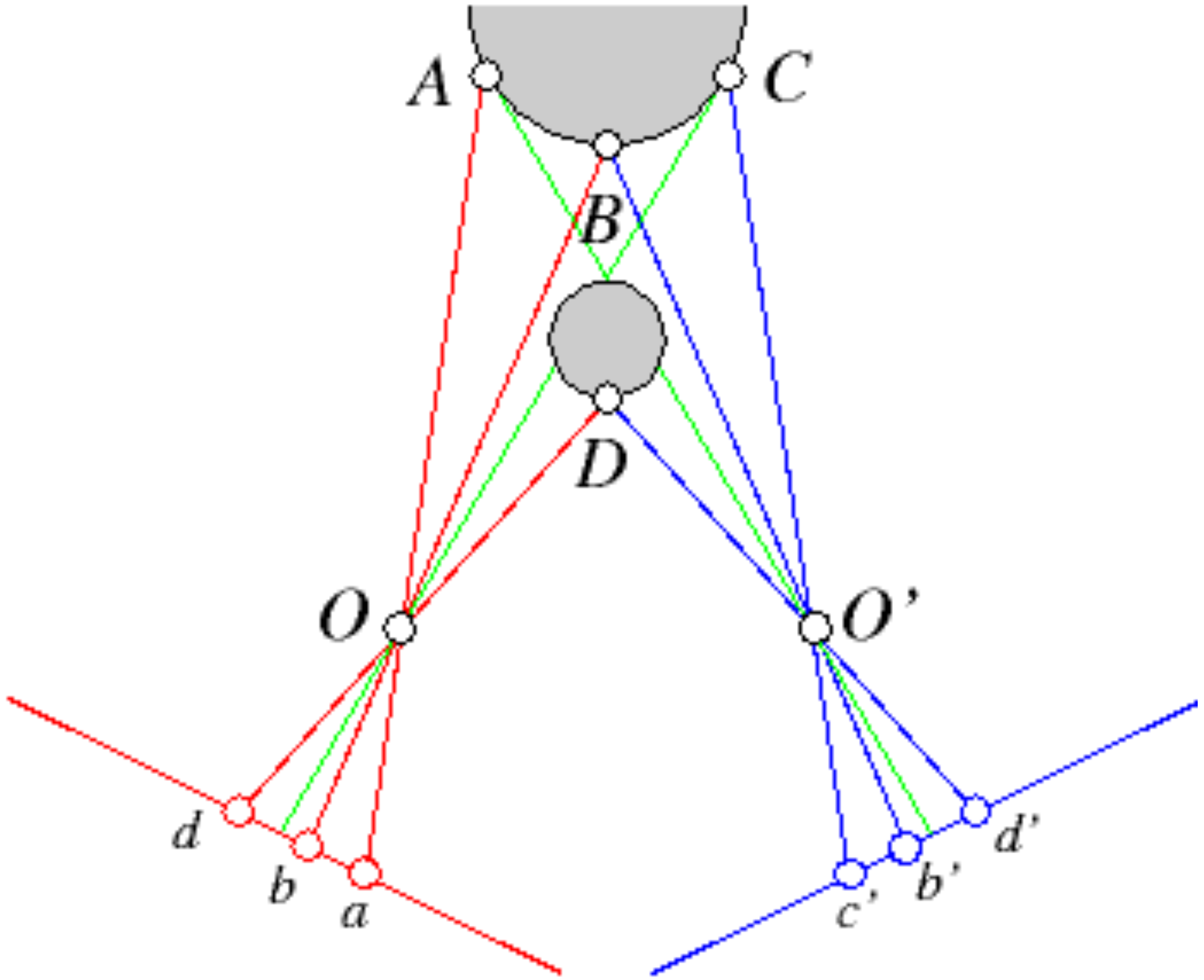
Note: Some approaches use an adaptive window size — try multiple sizes and select best match

Ordering Constraints

Ordering constraint …

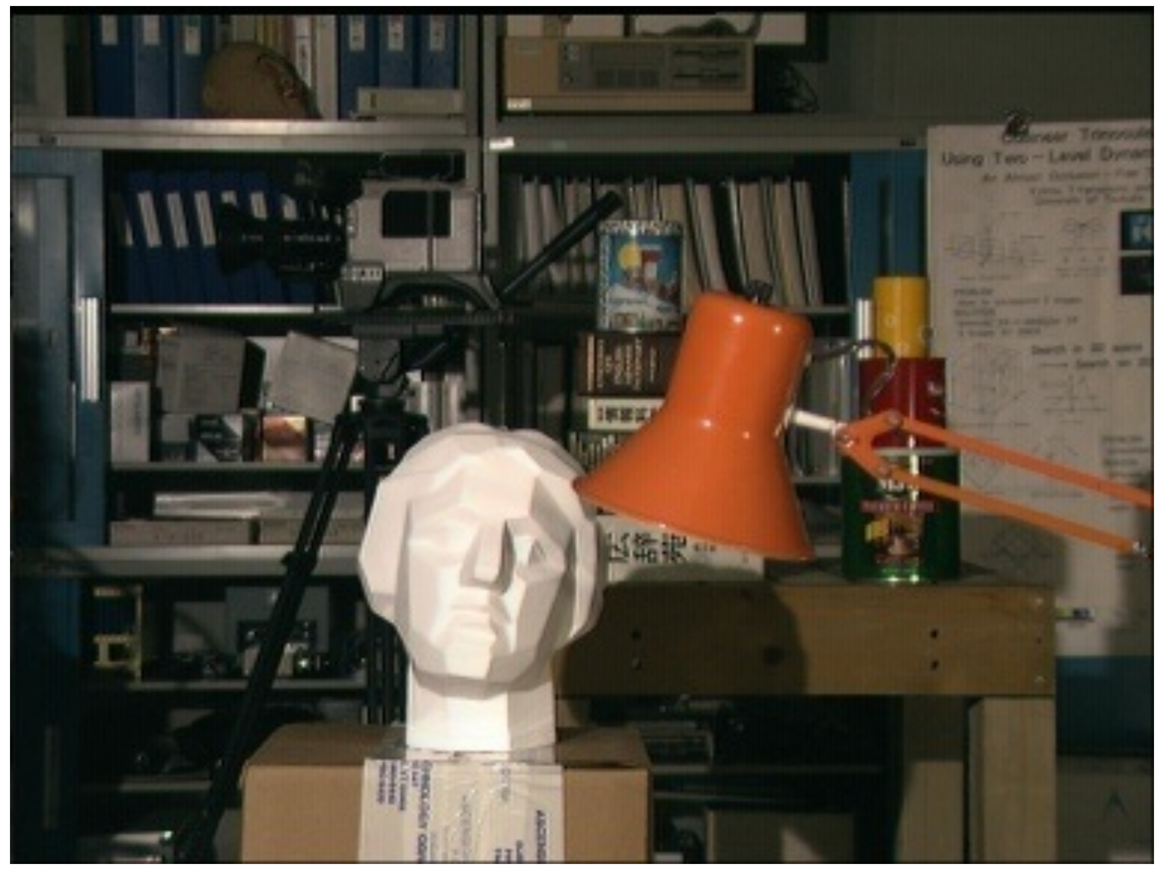


…. and a **failure** case

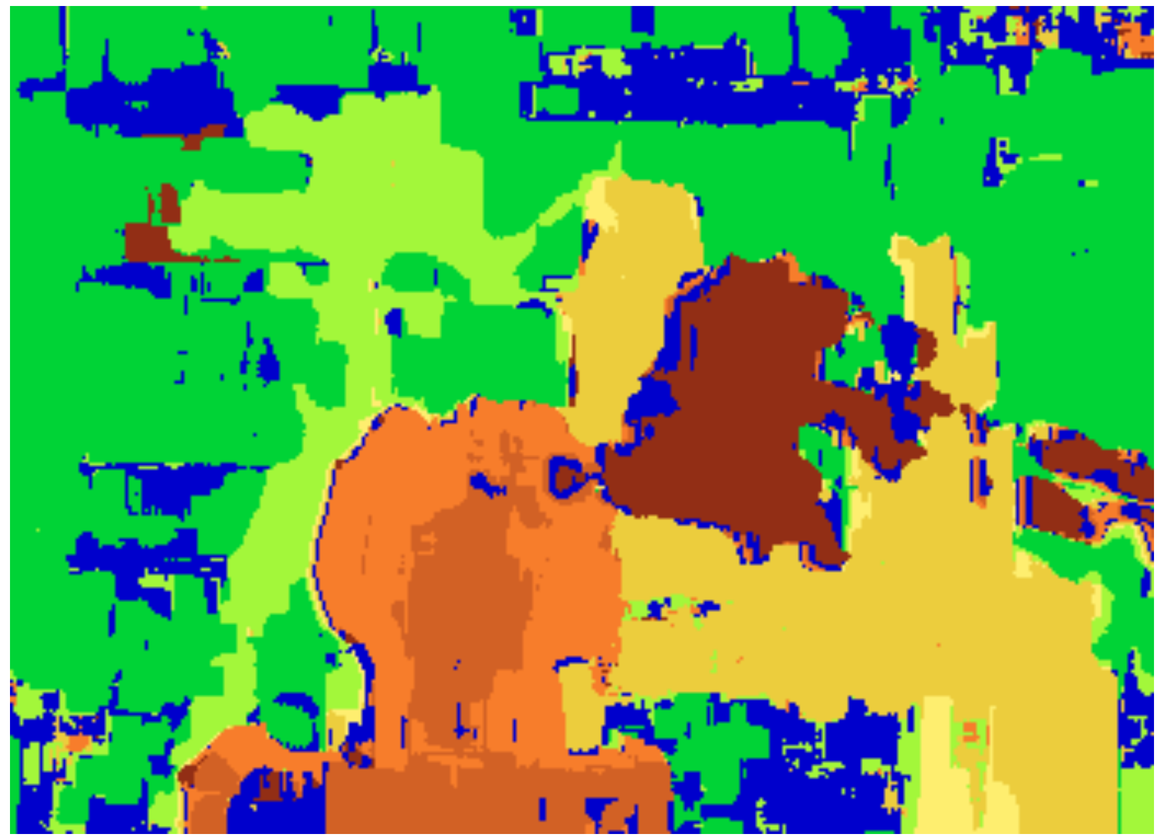


Forsyth & Ponce (2nd ed.) Figure 7.13

Block Matching Techniques: Result



Block matching



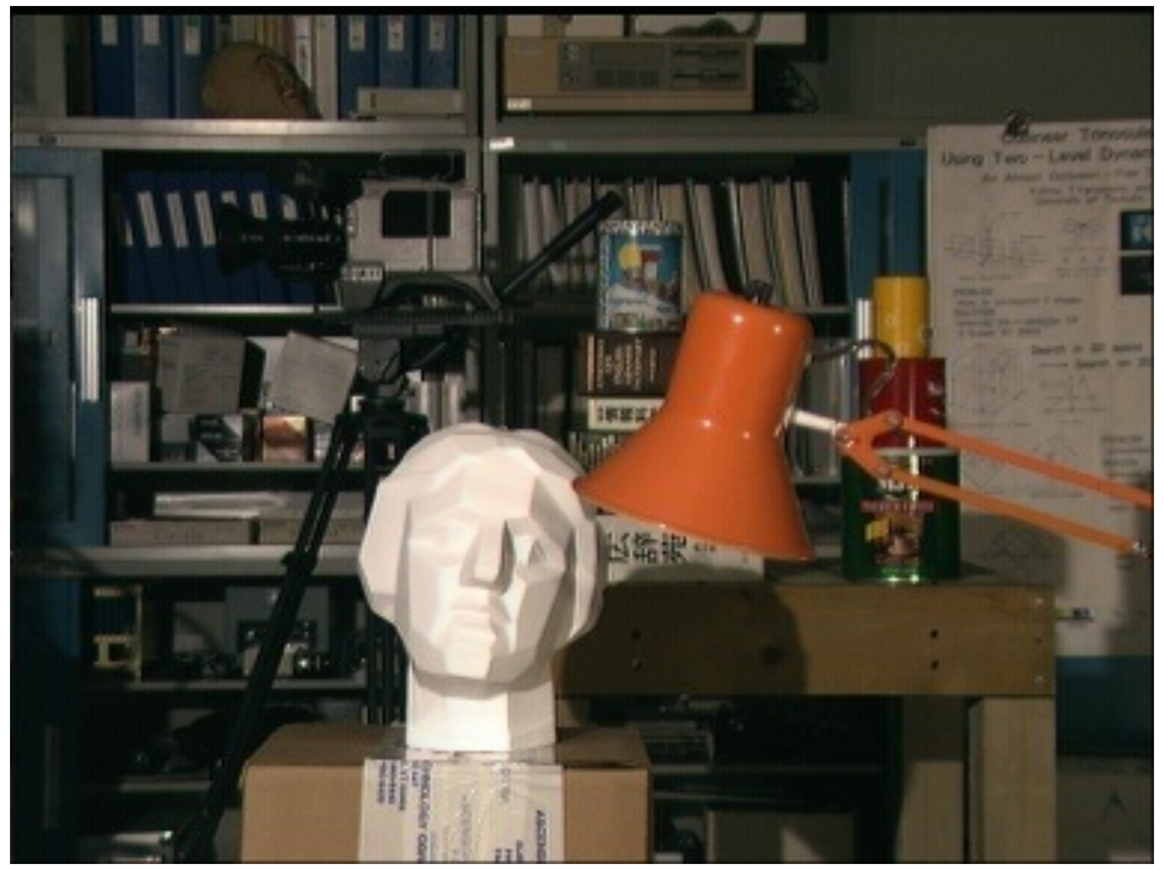
Ground truth

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

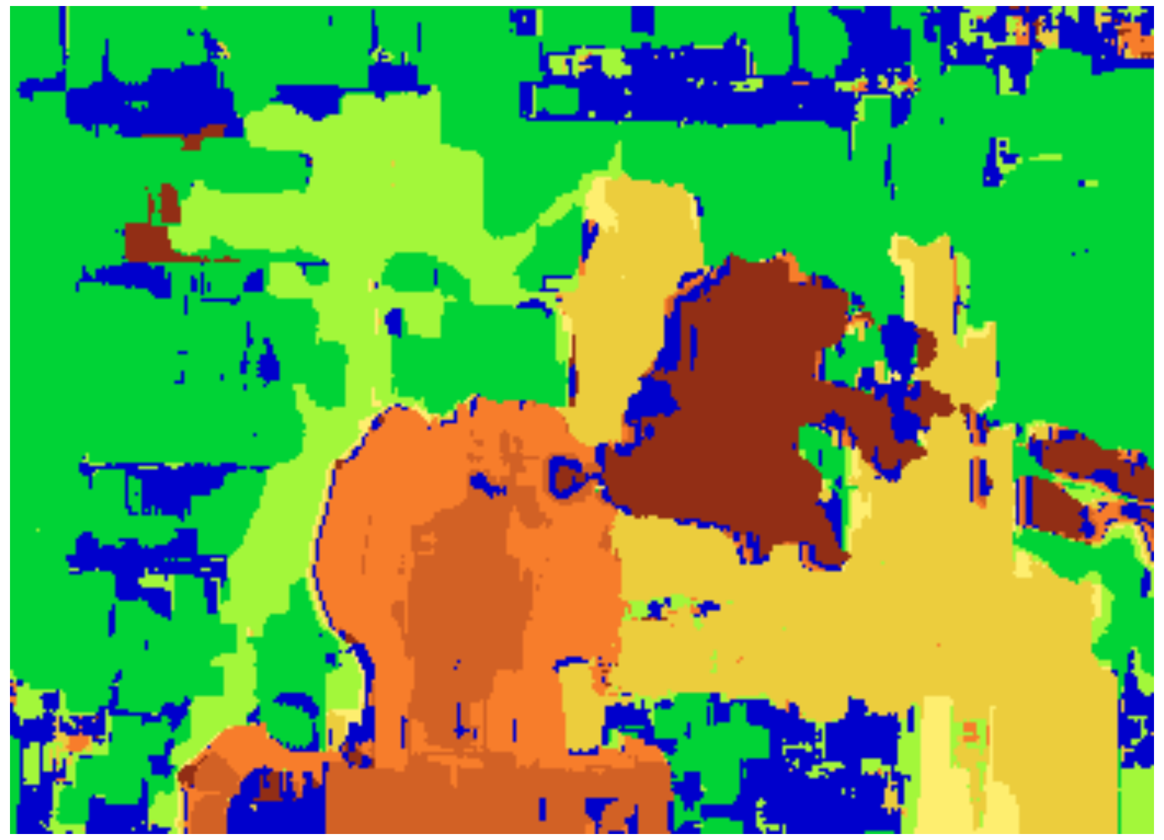
Block Matching Techniques: Result

Too many **discontinuities**.
We expect disparity values to change slowly.

Let's make an assumption:
depth should change smoothly



Block matching



Ground truth

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Stereo Matching as **Energy Minimization**

energy function
(for one pixel)

$$E(d) = \underbrace{E_d(d)}_{\text{data term}} + \lambda \underbrace{E_s(d)}_{\text{smoothness term}}$$

Want each pixel to find a good match in the other image

(block matching result)

Adjacent pixels should (usually) move about the same amount

(smoothness function)

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Stereo Matching as **Energy Minimization**

$$E(d) = E_d(d) + \lambda \boxed{E_s(d)}$$

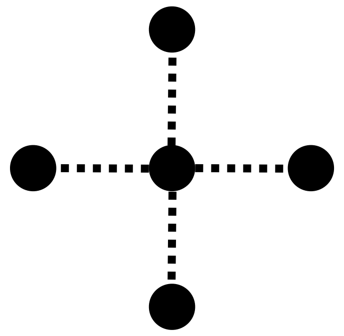
$$E_d(d) = \sum_{(x,y) \in I} C(x, y, d(x,y))$$

SSD distance between windows centered at I(x, y)
and J(x+ d(x,y), y)

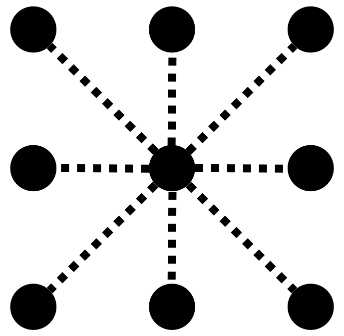
$$E_s(d) = \sum_{(p,q) \in \mathcal{E}} V(d_p, d_q)$$

smoothness term

$\mathcal{E}$: set of neighboring pixels



4-connected neighborhood



8-connected neighborhood

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

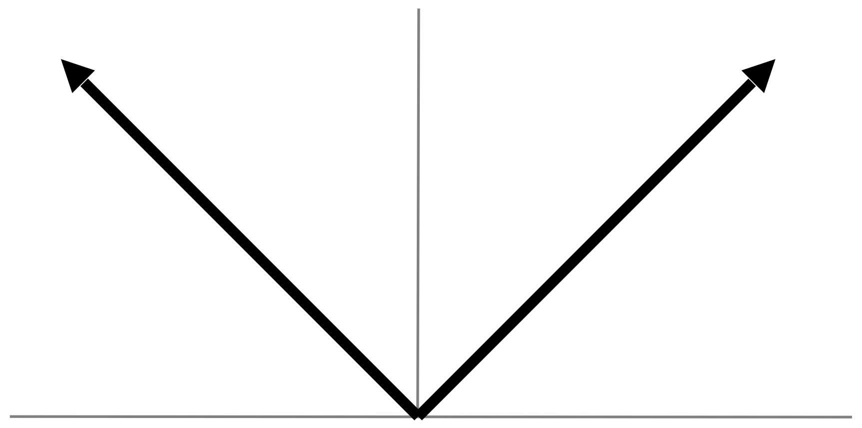
Stereo Matching as **Energy Minimization**

$$E_s(d) = \sum_{(p,q)\in\mathcal{E}} V(d_p, d_q)$$

smoothness term

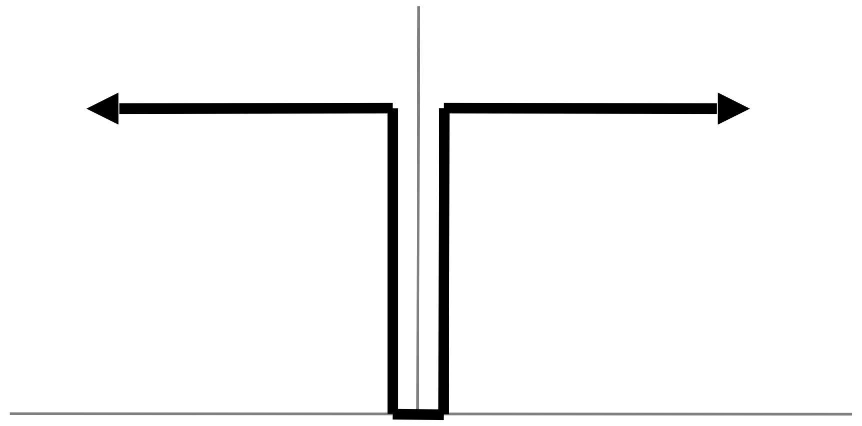
$$V(d_p, d_q) = |d_p - d_q|$$

L_1 distance



$$V(d_p, d_q) = \begin{cases} 0 & \text{if } d_p = d_q \\ 1 & \text{if } d_p \neq d_q \end{cases}$$

"Potts model"



Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

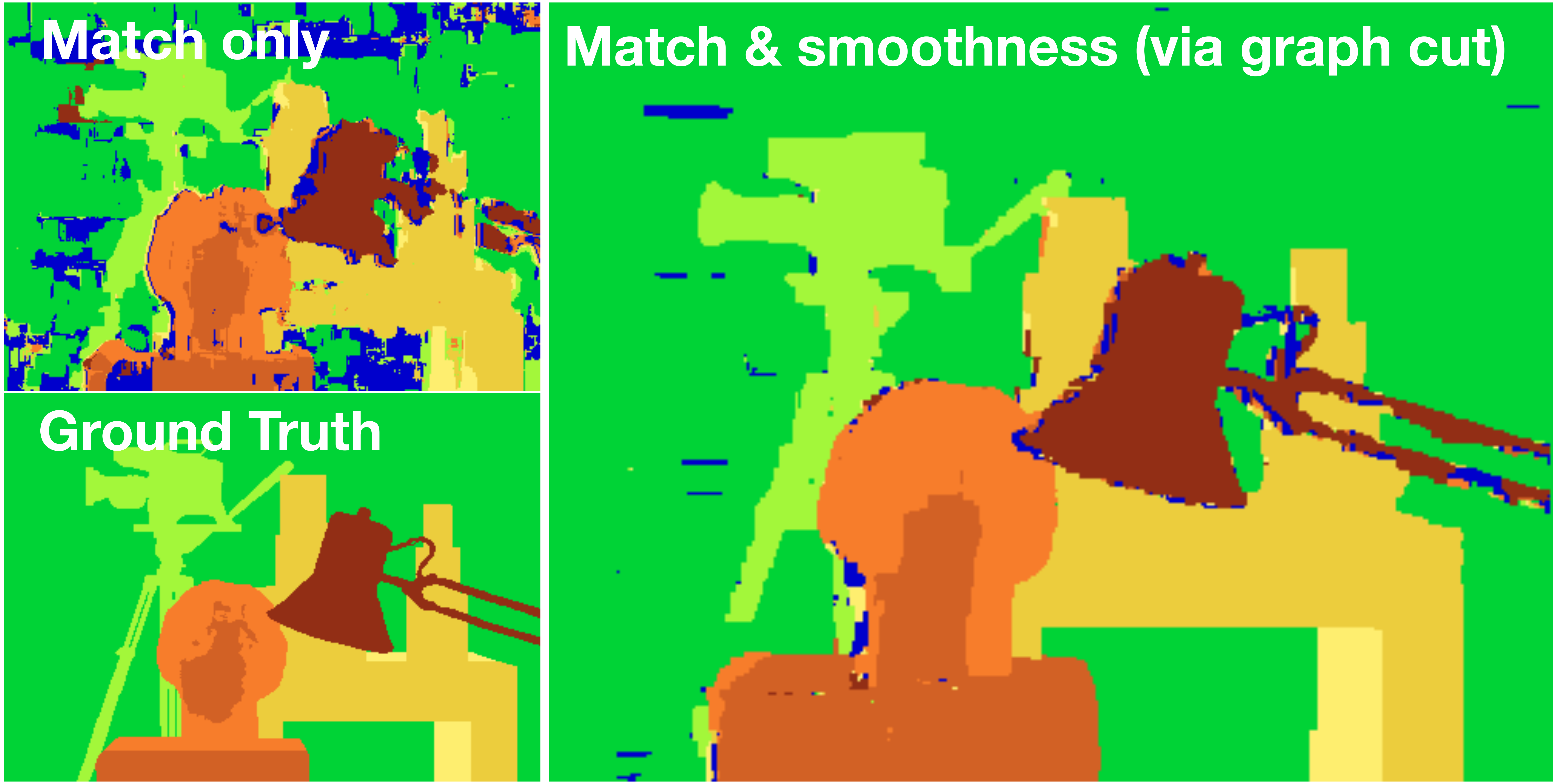
Stereo Matching as **Energy Minimization**: Solution

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

Can minimize this independently per scanline
using **dynamic programming** (DP)

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Stereo Matching as **Energy Minimization**

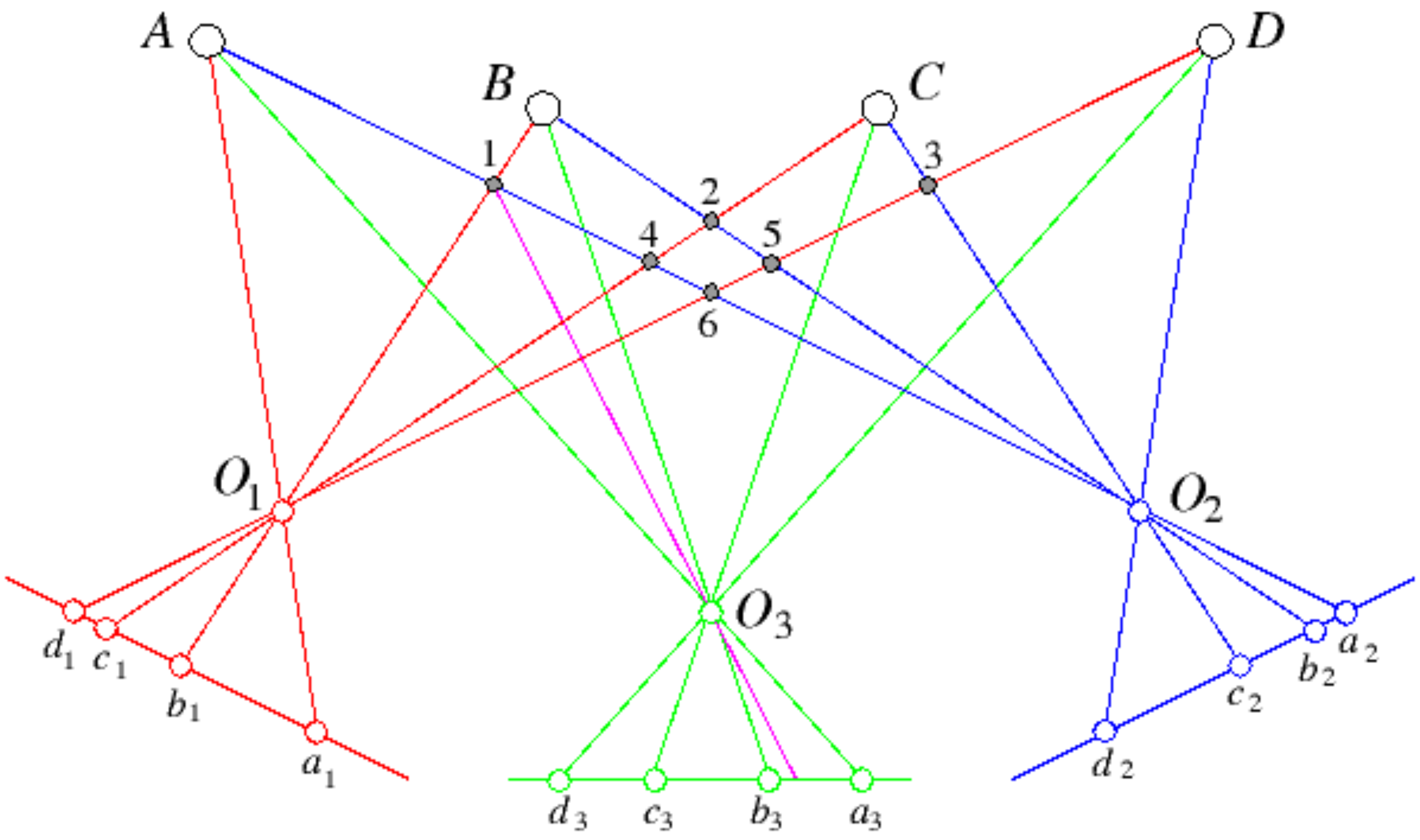


Y. Boykov, O. Veksler, and R. Zabih, Fast Approximate Energy Minimization via Graph Cuts, PAMI 2001

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Idea: Use More Cameras

Adding a third camera reduces ambiguity in stereo matching



Forsyth & Ponce (2nd ed.) Figure 7.17

Point Grey Research **Digiclops**

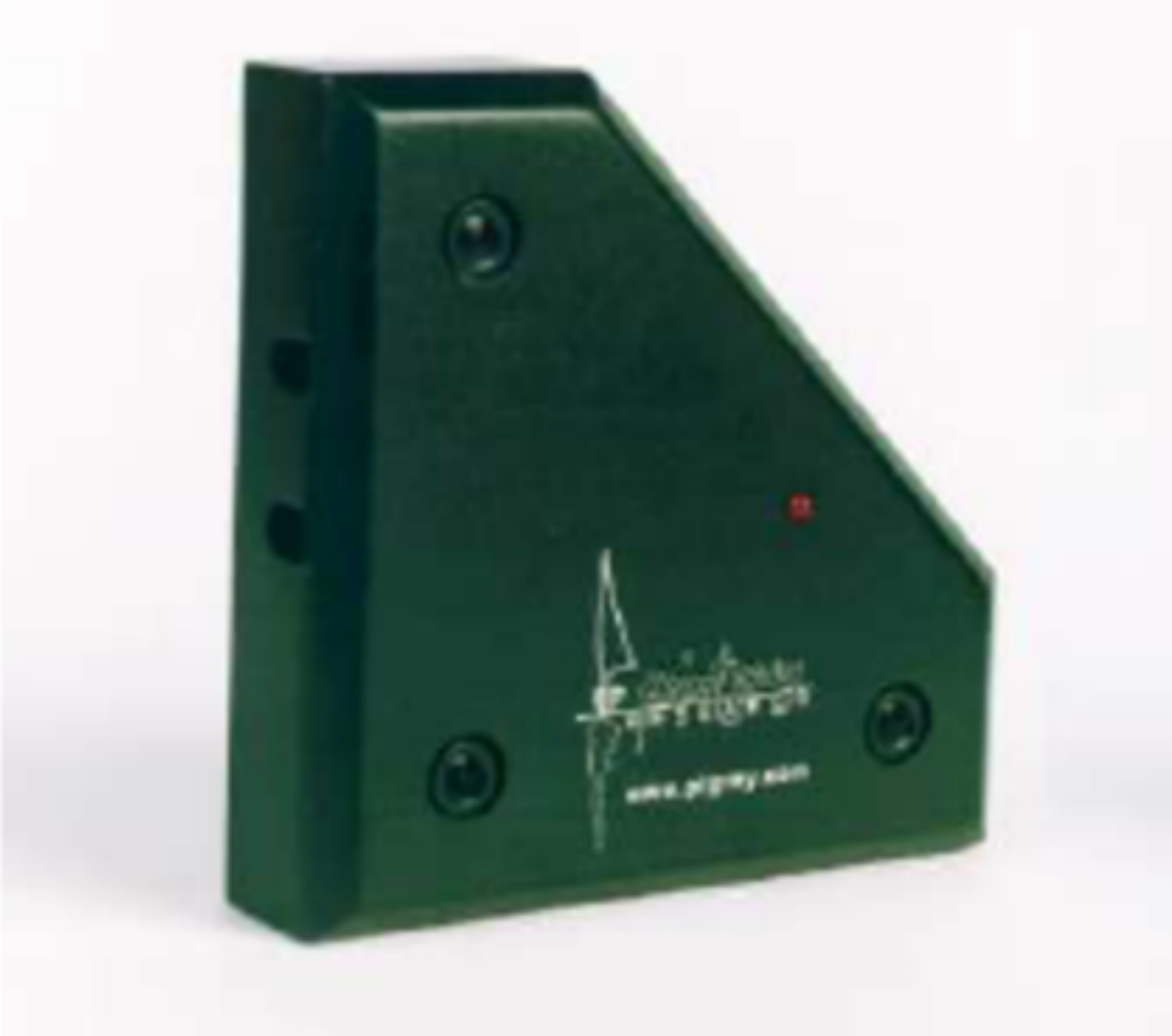
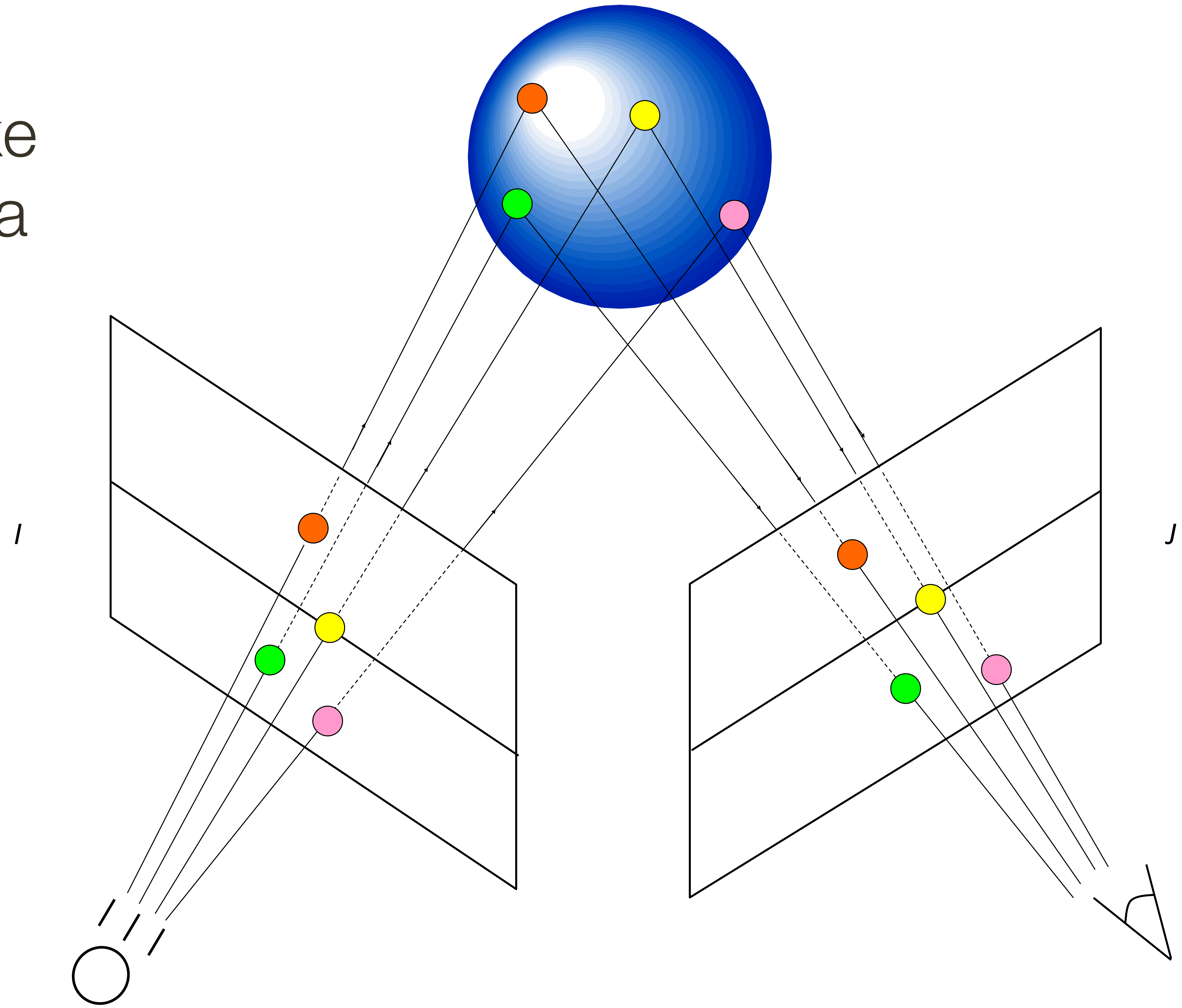


Image credit: Point Grey Research

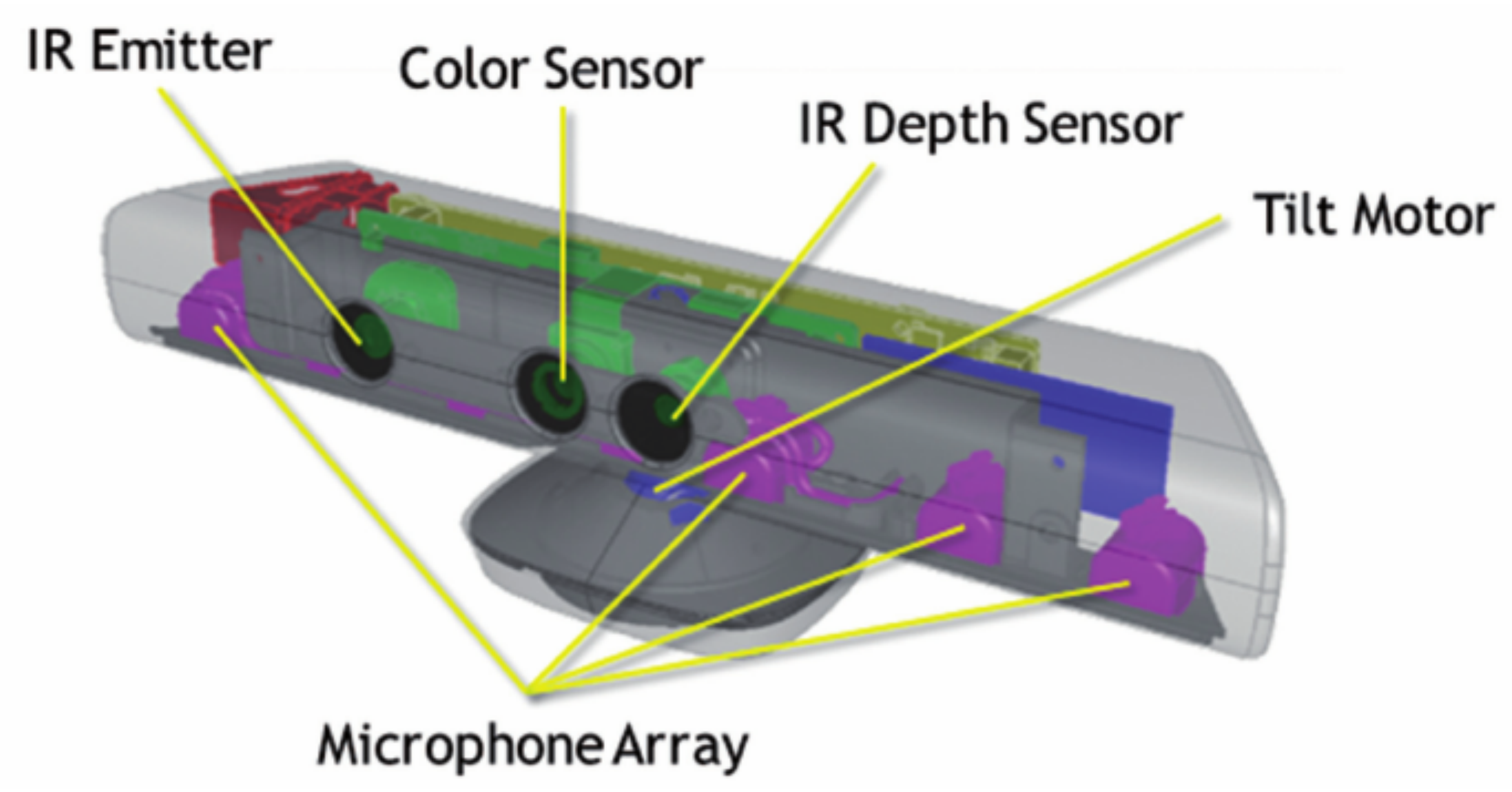
Structured Light Imaging: Structured Light and One Camera

Projector acts like
“reverse” camera

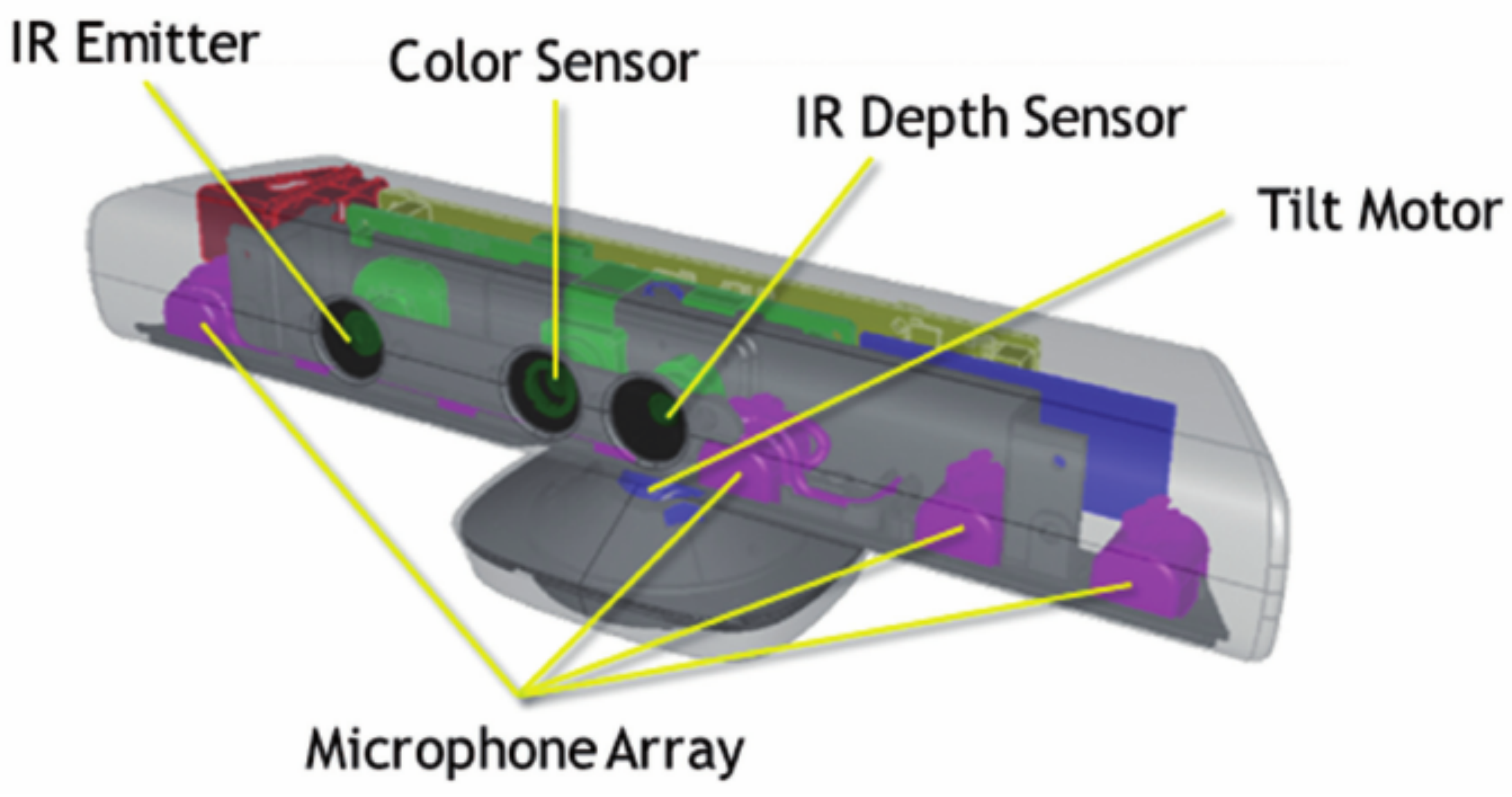


Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Microsoft **Kinect**



Microsoft **Kinect**



Summary

Stereo is formulated as a **correspondence** problem

— determine match between location of a scene point in one image and its location in another

If we assume calibrated cameras and image **rectification**, **epipolar lines** are horizontal scan lines

What do we match?

— Individual pixels?

— Patches?

— Edges?