CPSC 425: Computer Vision

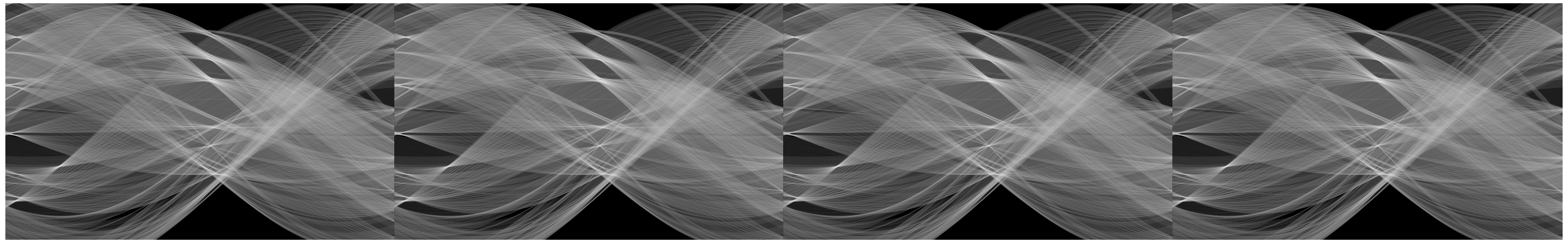


Image Credit: Ioannis (Yannis) Gkioulekas (CMU)

Lecture 16: Stereo

Menu for Today (**November 4, 2024**)

Topics:

— **Stereo** Vision, Epipolar Geometry

Readings:

— **Today's** Lecture: Szeliski 12.1, 12.3-12.4, 9.3

Reminders:

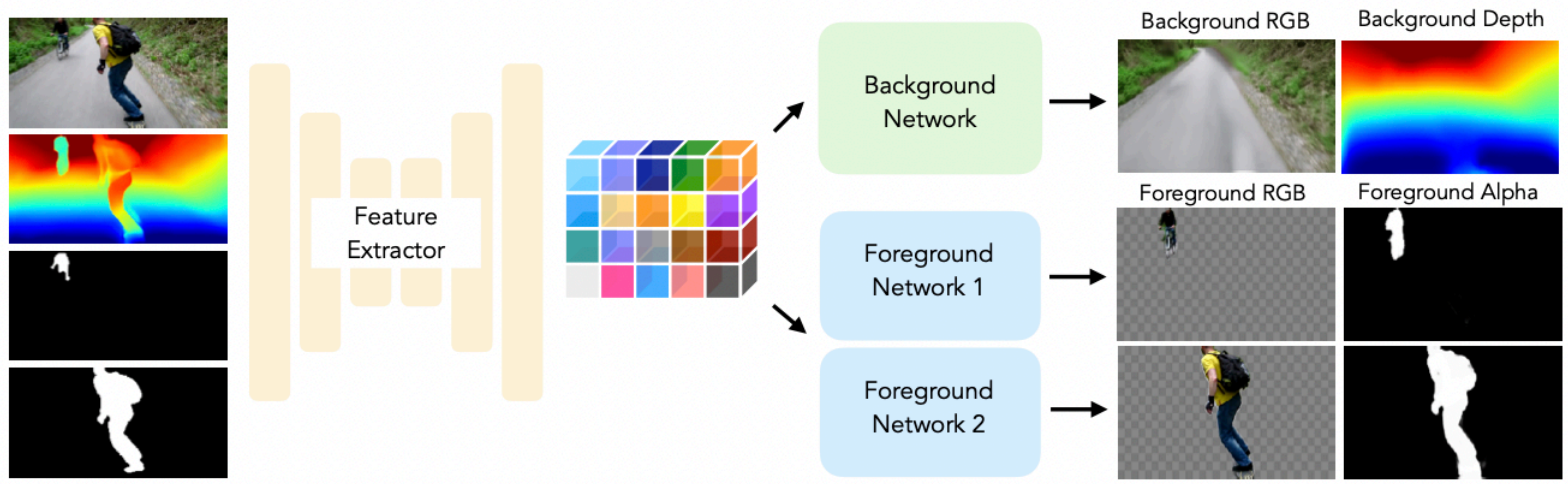
— **Midterms** are graded (Mean/Median: 72 (after scale); 67 (before scale))

38 A's (80+), 54 B's (>=68), 44 C's and below

— **Assignment 4**: RANSAC and Panoramas due **November 7th -> 8th**

— **Assignment 5: NEW —** Stereo and Optical Flow

Today's "**fun**" Example: **Omnimatte 360**




Inputs
RGB
Depth
Masks
Outputs
Layer 1
(RGBA)
Layer 2
(RGBA)
BG
(RGBD)


Inputs
RGB
Depth
Masks
Outputs
Layer 1
(RGBA)
Layer 2
(RGBA)
BG
(RGBD)

Summary of Hough Transform

Idea of **Hough transform**:

— For each token vote for all models to which the token could belong

— Return models that get many votes

e.g., For each point, vote for all lines that could pass through it; the true lines will pass through many points and so receive many votes

Advantages:

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

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

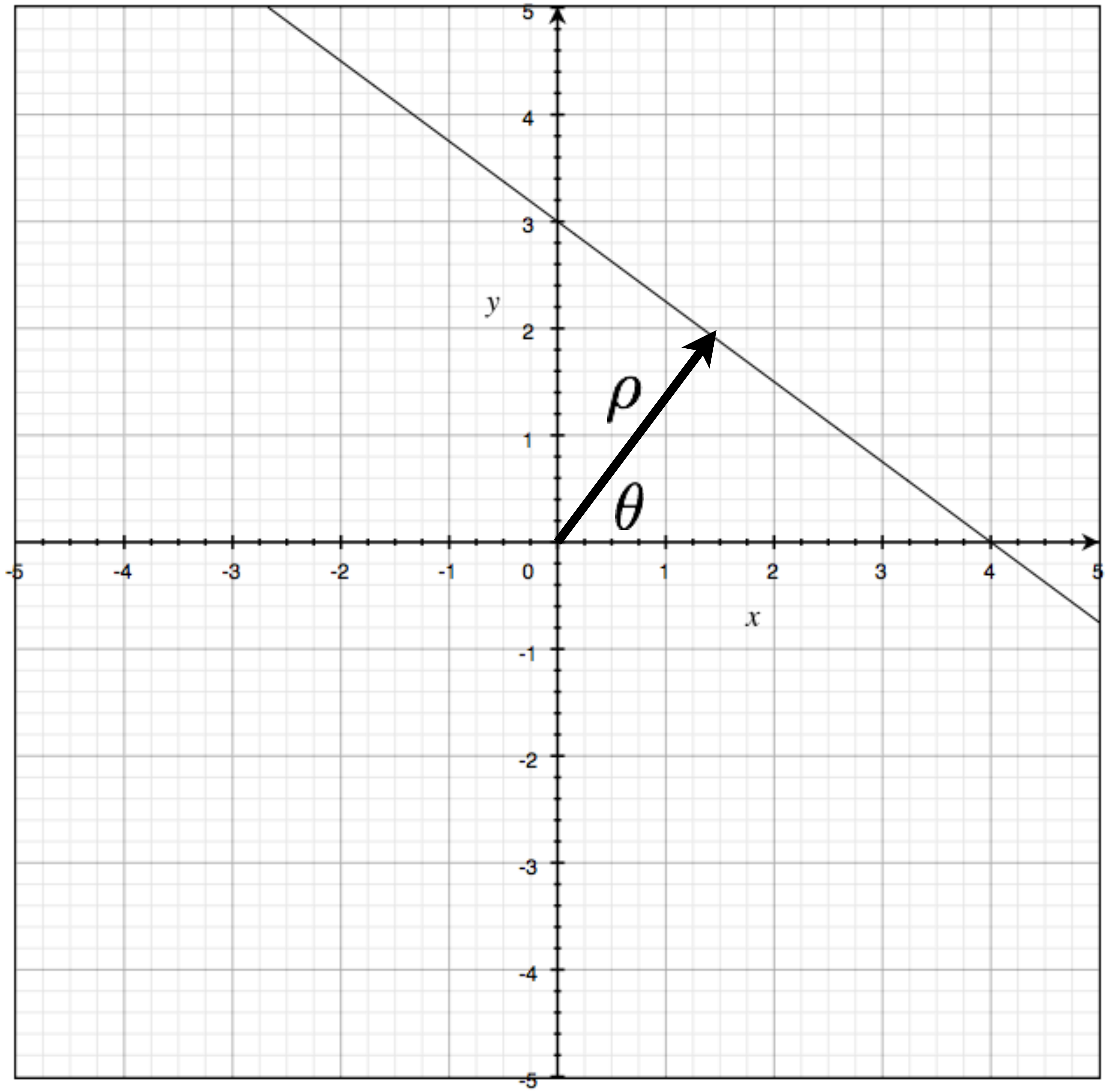
Disadvantages:

— Search time increases exponentially with the number of model parameters

— Can be tricky to pick a good bin size

Lines: Normal form

$$x\cos(\theta) + y\sin(\theta) = \rho$$



Forsyth/Ponce convention

$$x\cos(\theta) + y\sin(\theta) + r = 0$$

$$r \geq 0$$

$$0 \leq \theta \leq 2\pi$$

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Hough Transform: Lines

variables

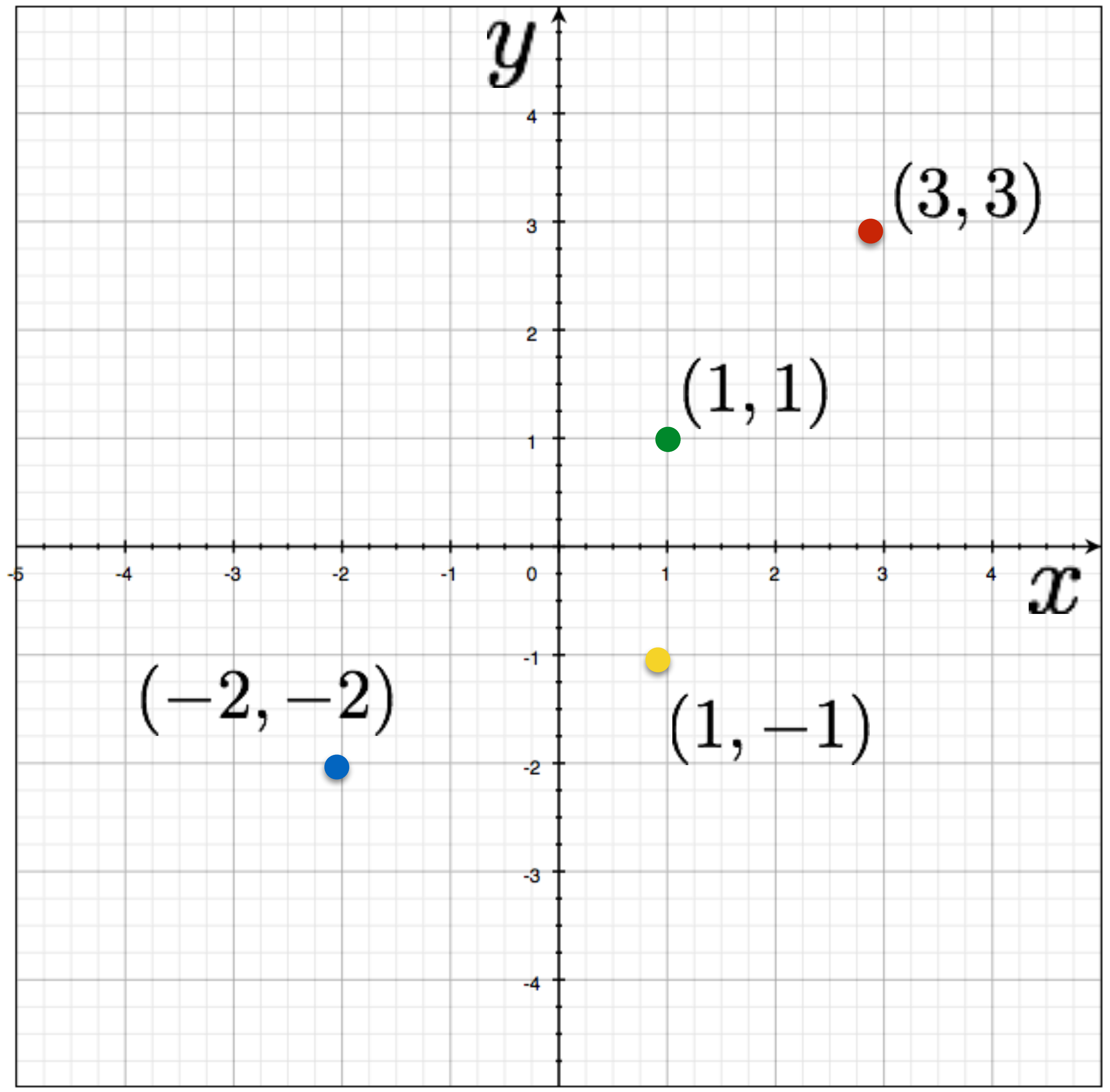
$$y = mx + b$$

parameters

parameters

$$x\cos(\theta) + y\sin(\theta) = \rho$$

variables



four points become?

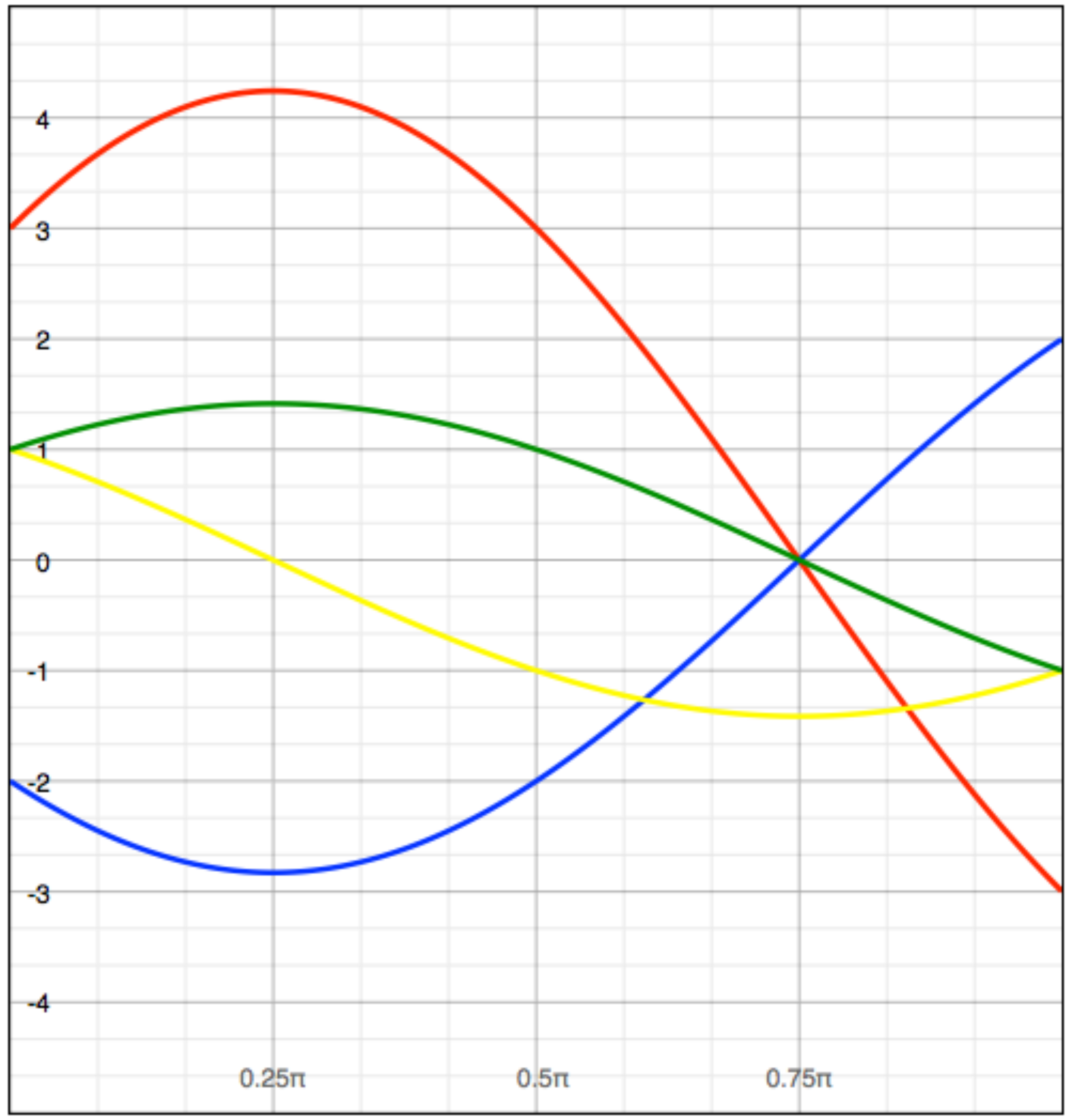
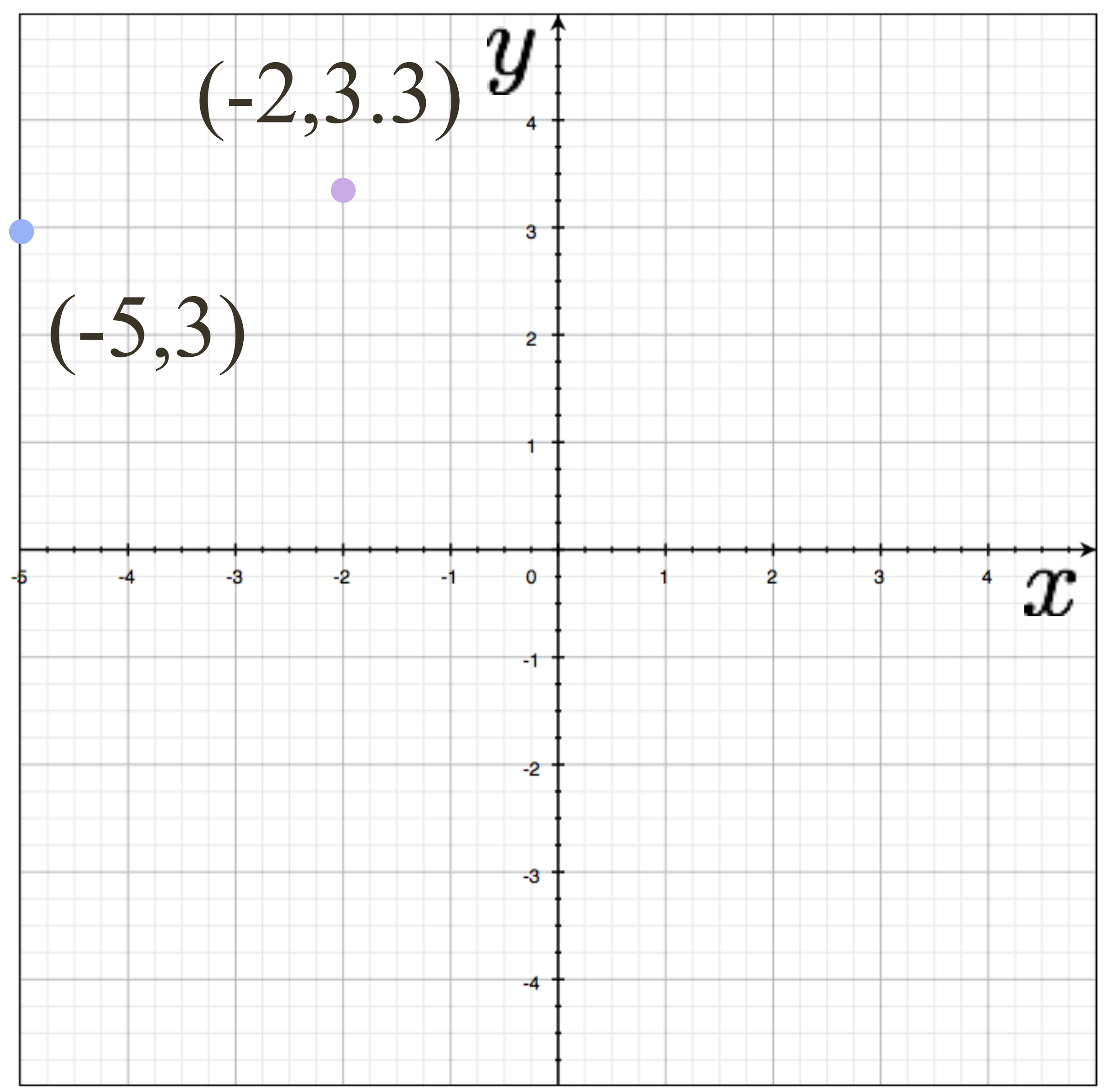


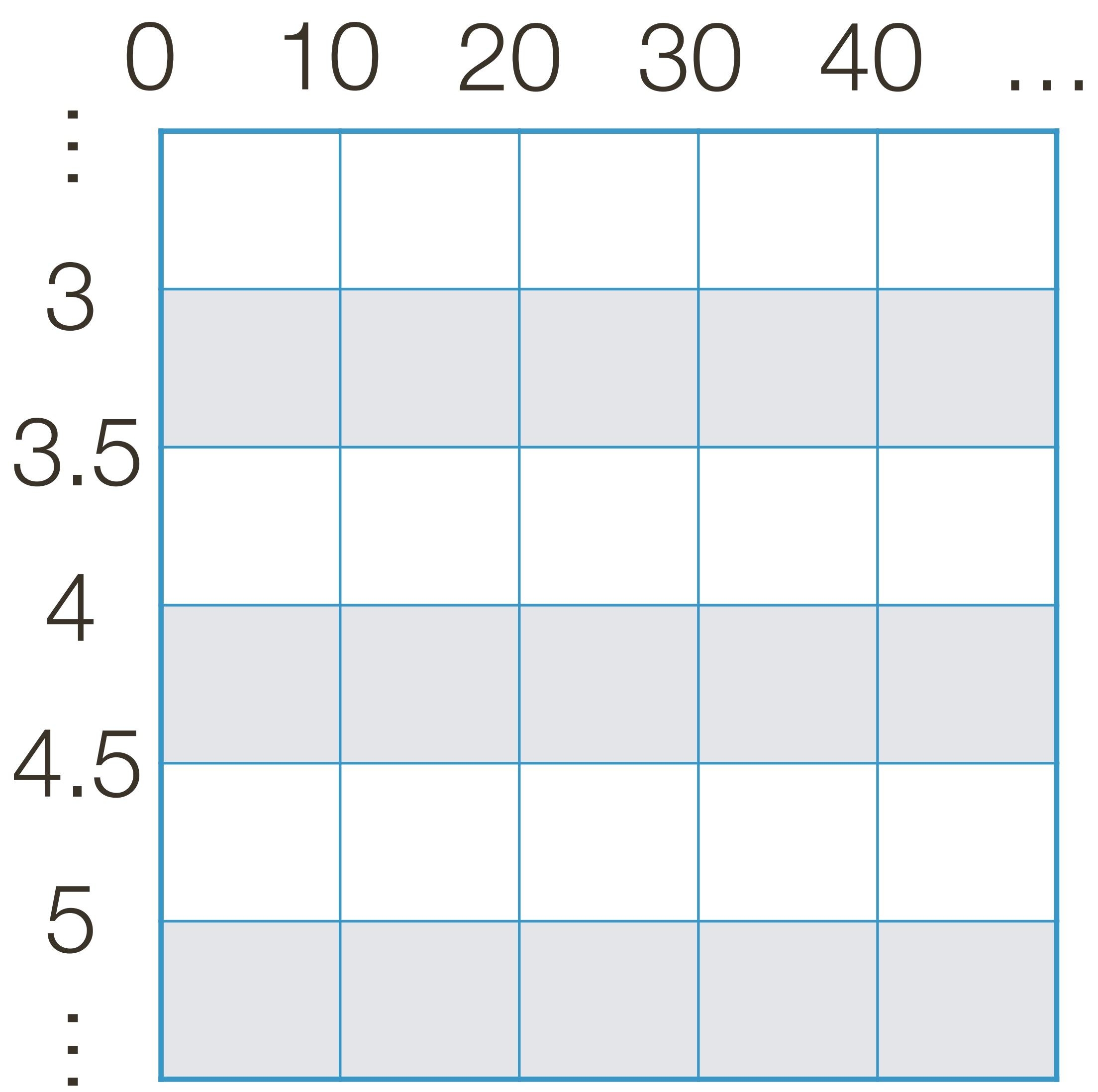
Image space

Parameter space

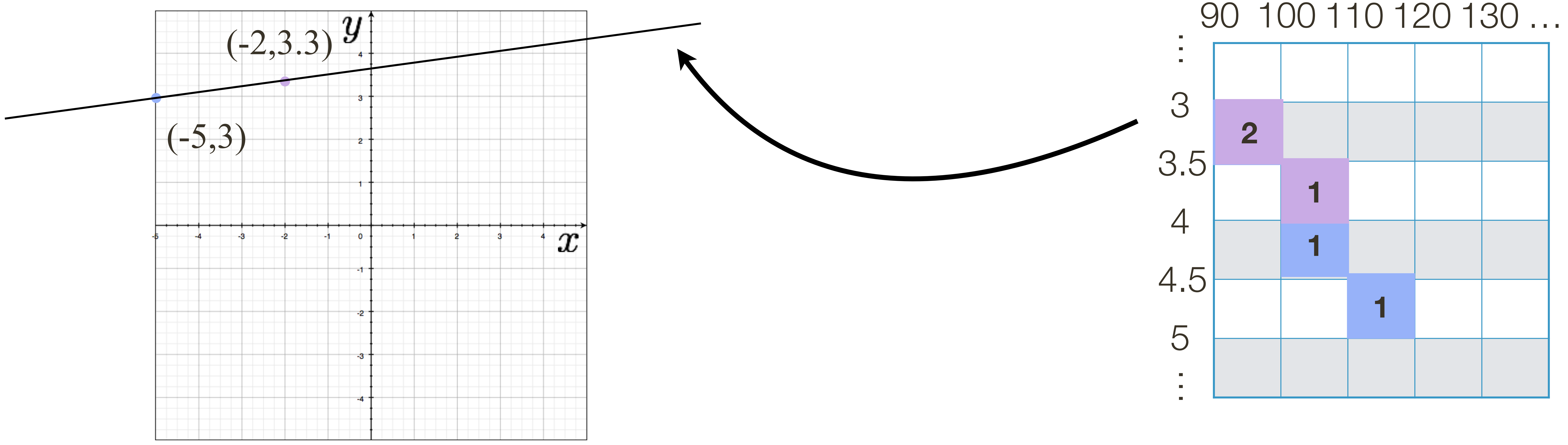
Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Example: Hough Transform for Lines





Example: Hough Transform for Lines



Example 1: Object Recognition — Implicit Shape Model

“**Training**” images of cows

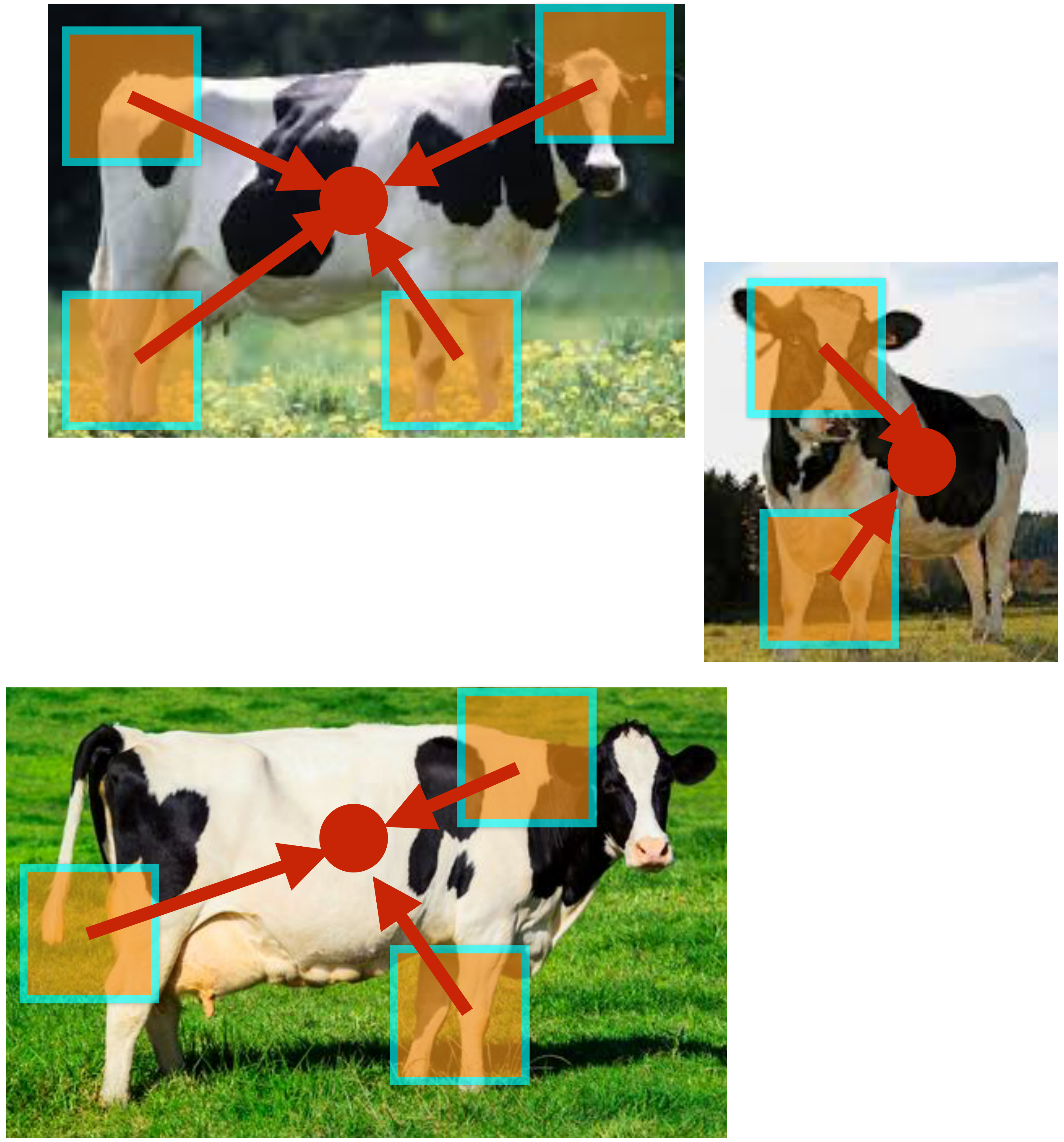
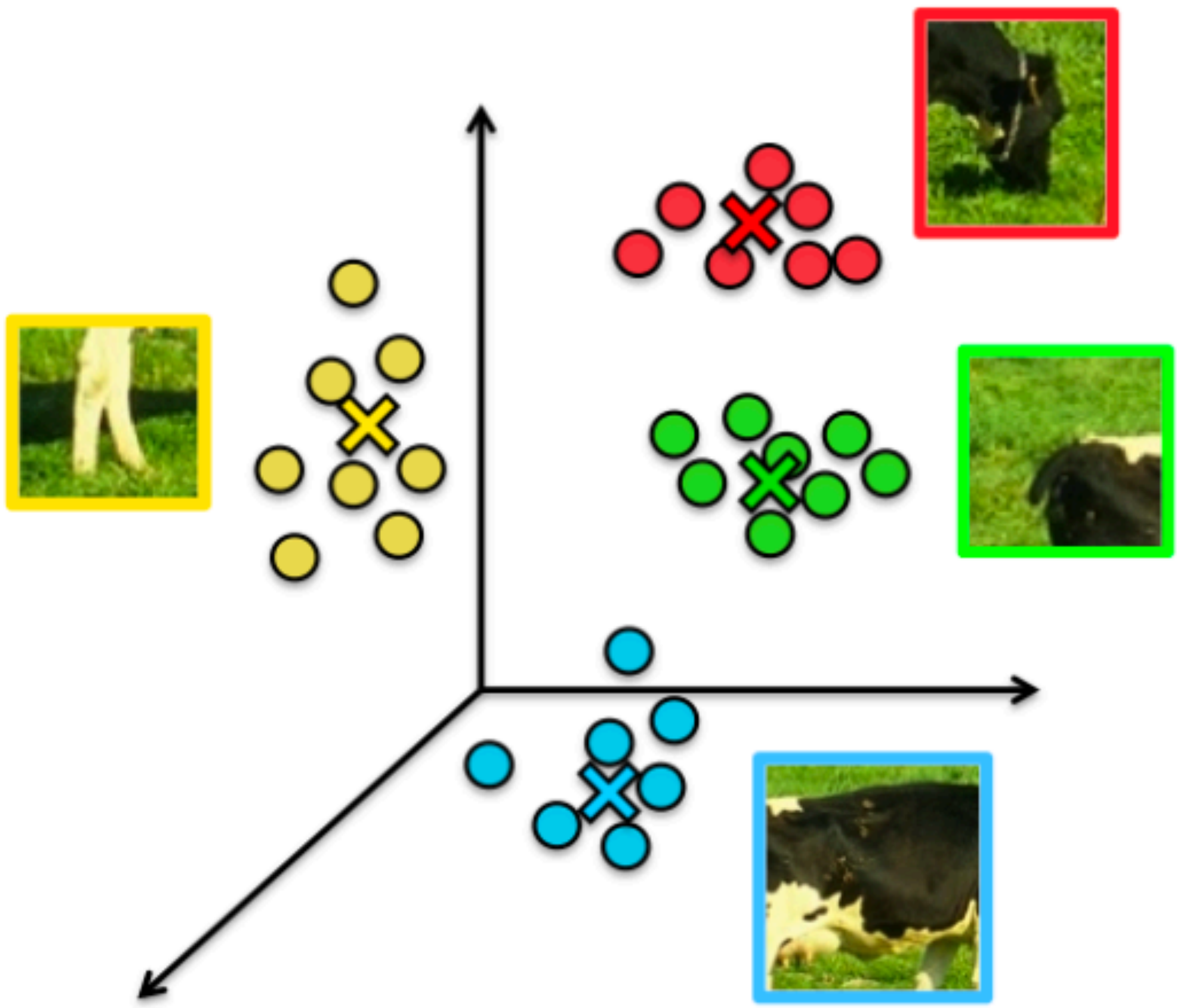


Image Index	Keypoint Index	Keypoint Detection (4D)	Keypoint Description (128D)	Offset to Centroid
Image 1	1	[x, y, s, Theta]	[…]	[x,y]
Image 1	2	[x, y, s, Theta]	[…]	[x,y]
….	….	….	…	…
Image 1	265	[x, y, s, Theta]	[…]	[x,y]
Image 2	1	[x, y, s, Theta]	[…]	[x,y]
Image 2	2	[x, y, s, Theta]	[…]	[x,y]
…	…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

Visual Words

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

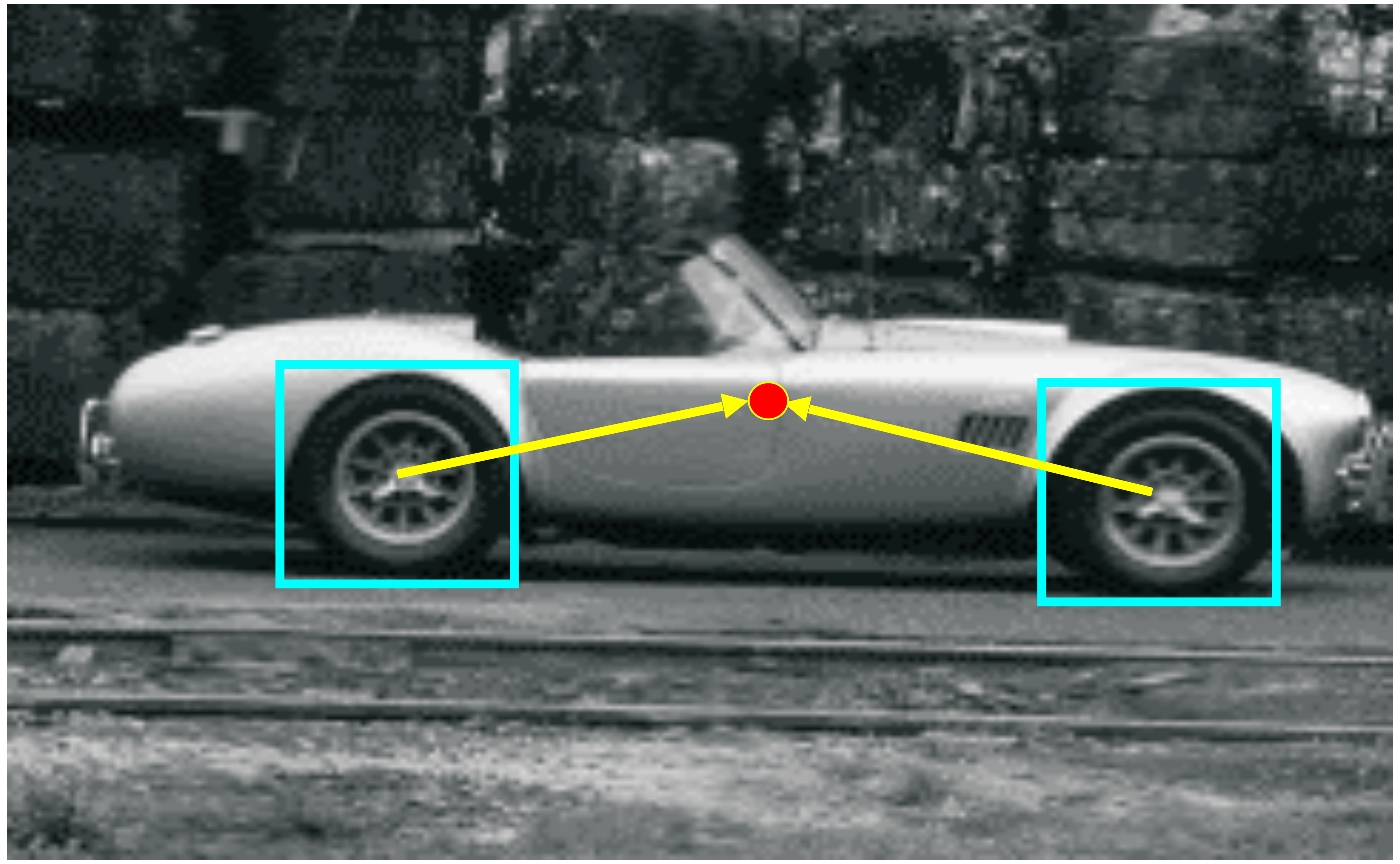


Visual words (visual codebook)!

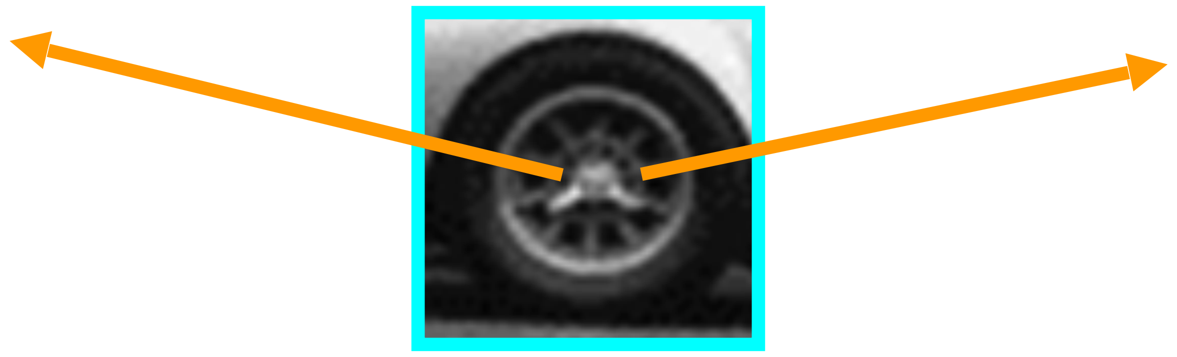
* Slide from Sanja Fidler

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

Example 1: Object Recognition — Implicit Shape Model

"**Training**" images of cows

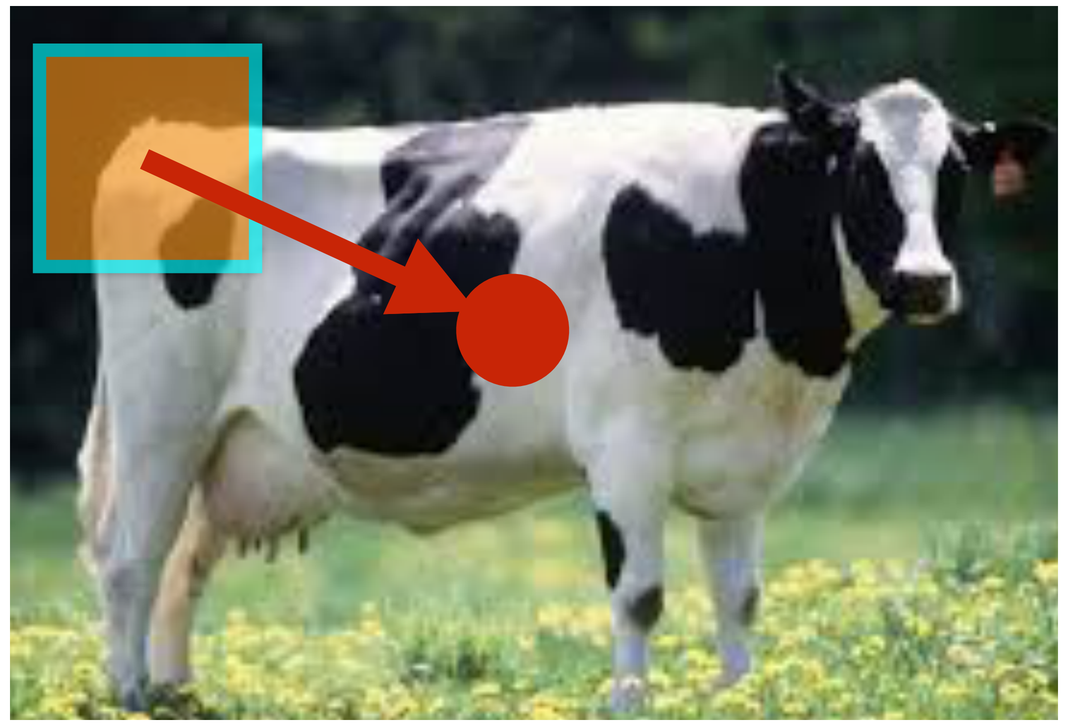
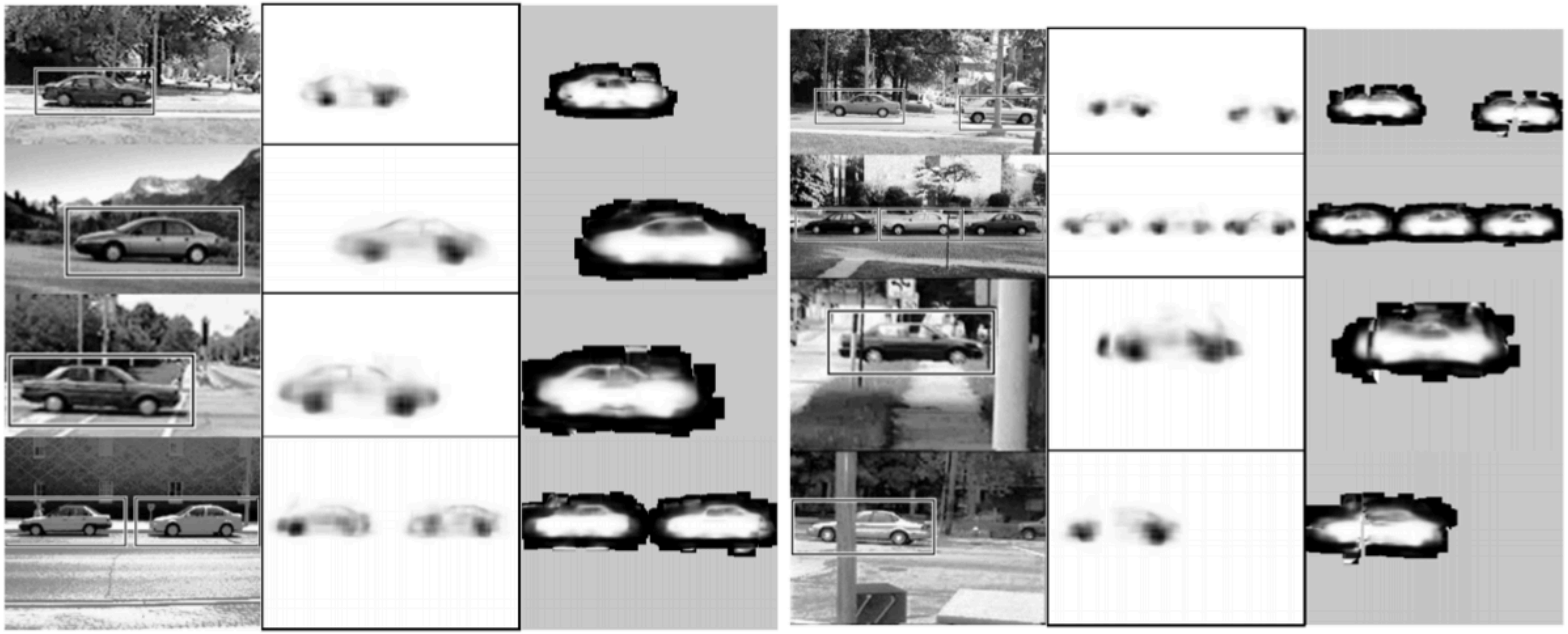


Image Index	Keypoint Index	Keypoint Detection (4D)	Keypoint Description (128D)	Offset to Centroid	Segment
Image 1	1	[x, y, s, Theta]	[…]	[x,y]	
Image 1	2	[x, y, s, Theta]	[…]	[x,y]	
….	….	….	…	…	
Image 1	265	[x, y, s, Theta]	[…]	[x,y]	
Image 2	1	[x, y, s, Theta]	[…]	[x,y]	
Image 2	2	[x, y, s, Theta]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

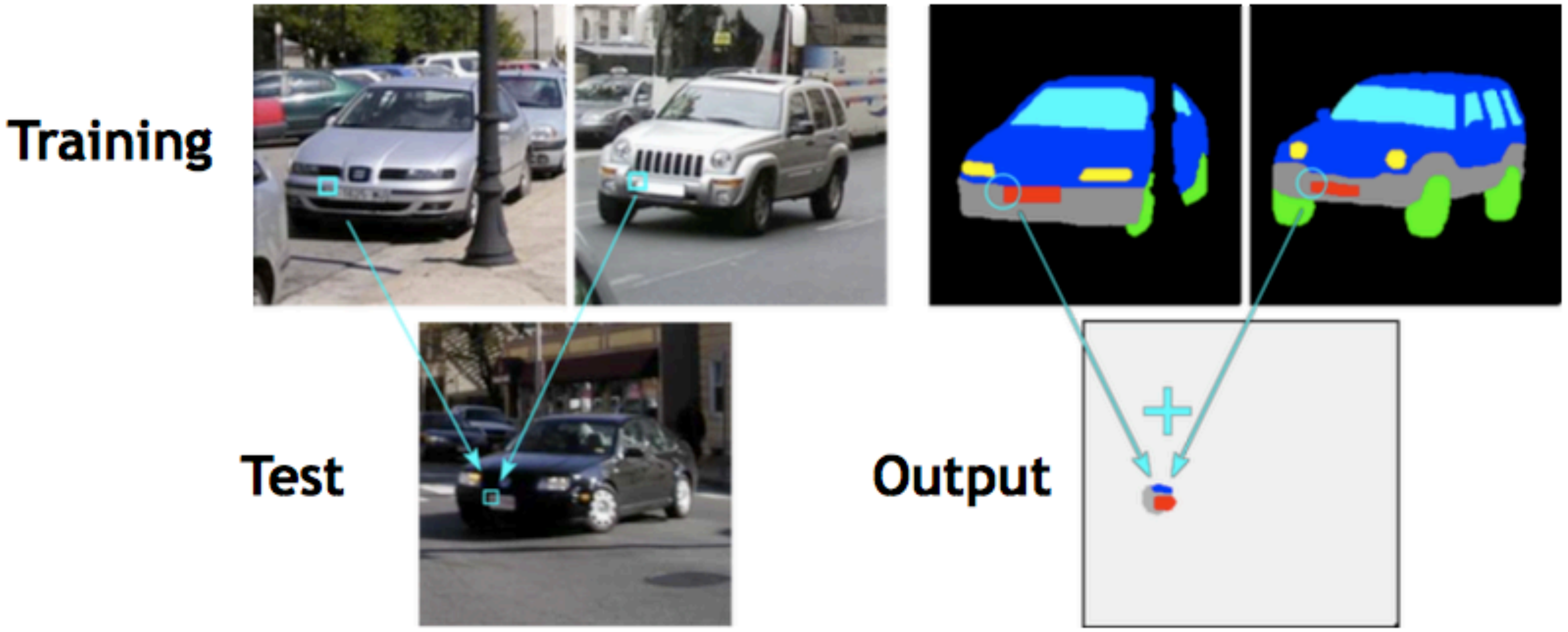
* Slide from Sanja Fidler

Inferring Other Information: **Segmentation**

[Source: B. Leibe]

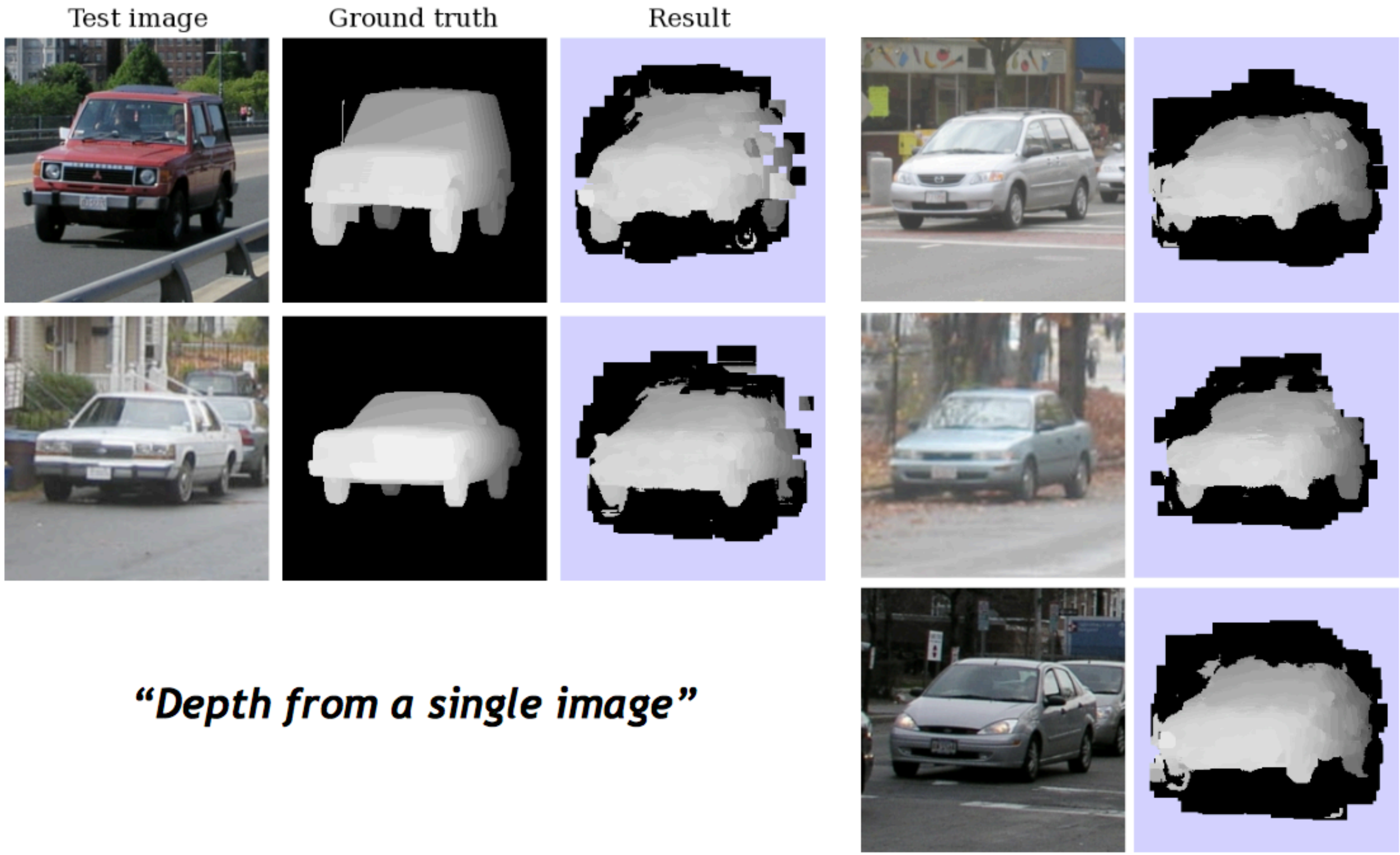
* Slide from Sanja Fidler

Inferring Other Information: **Part Labels**



* Slide from Sanja Fidler

Inferring Other Information: **Depth**



"Depth from a single image"

* Slide from Sanja Fidler

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

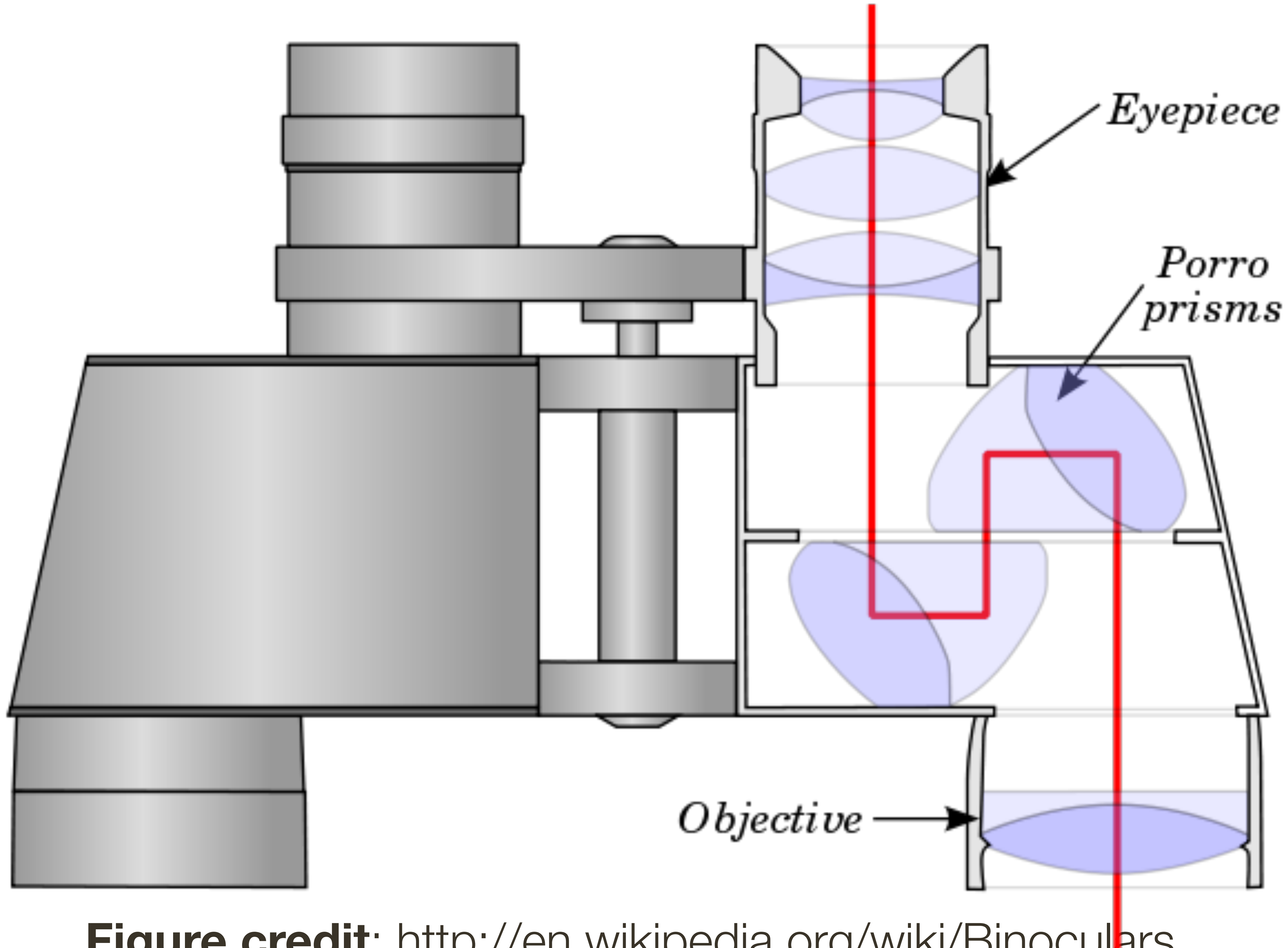


Figure credit: http://en.wikipedia.org/wiki/Binoculars

Stereo Vision

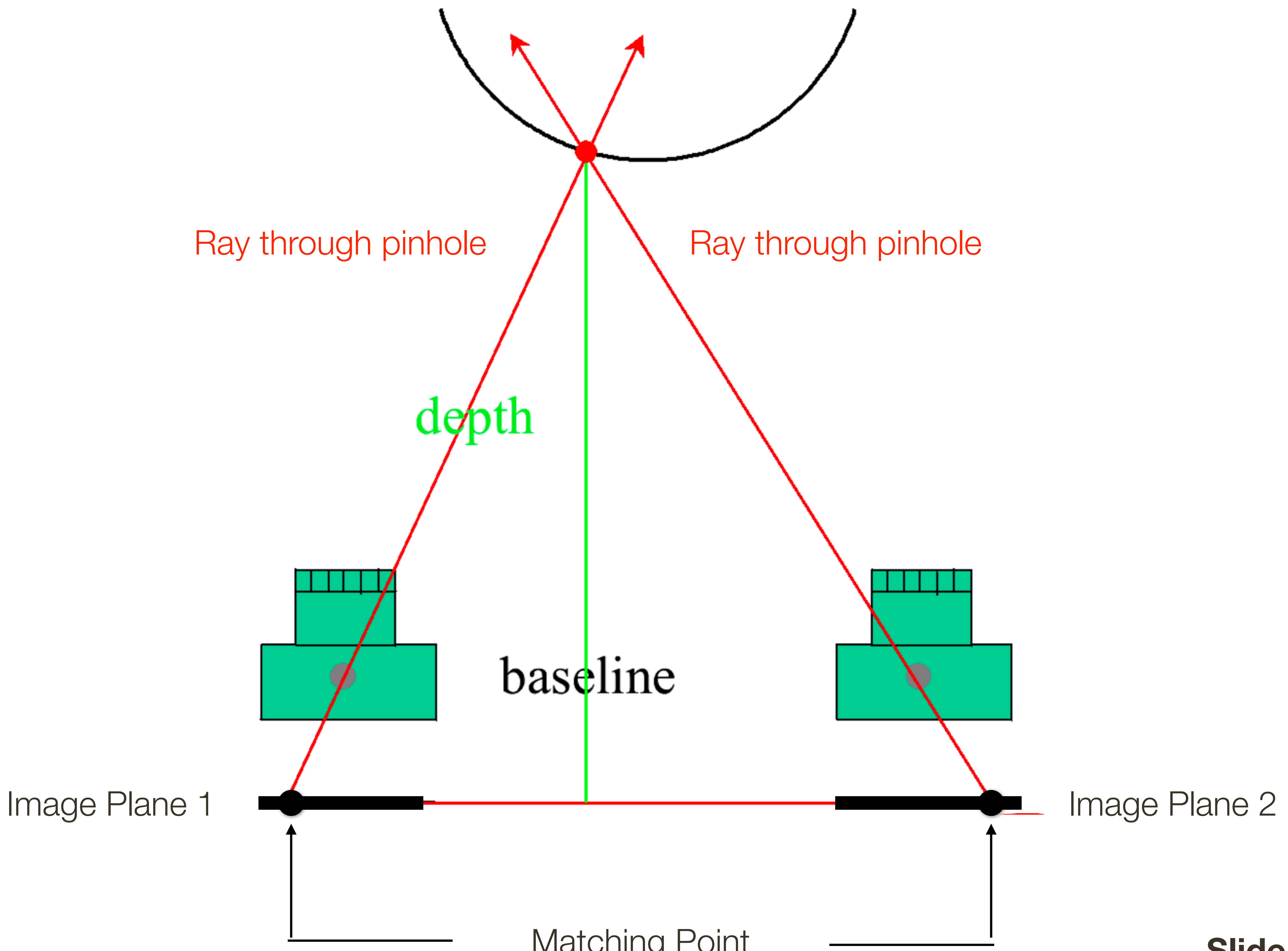
Task: Compute depth from two images acquired from (slightly) different viewpoints

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

Point Grey Research **Digiclops**

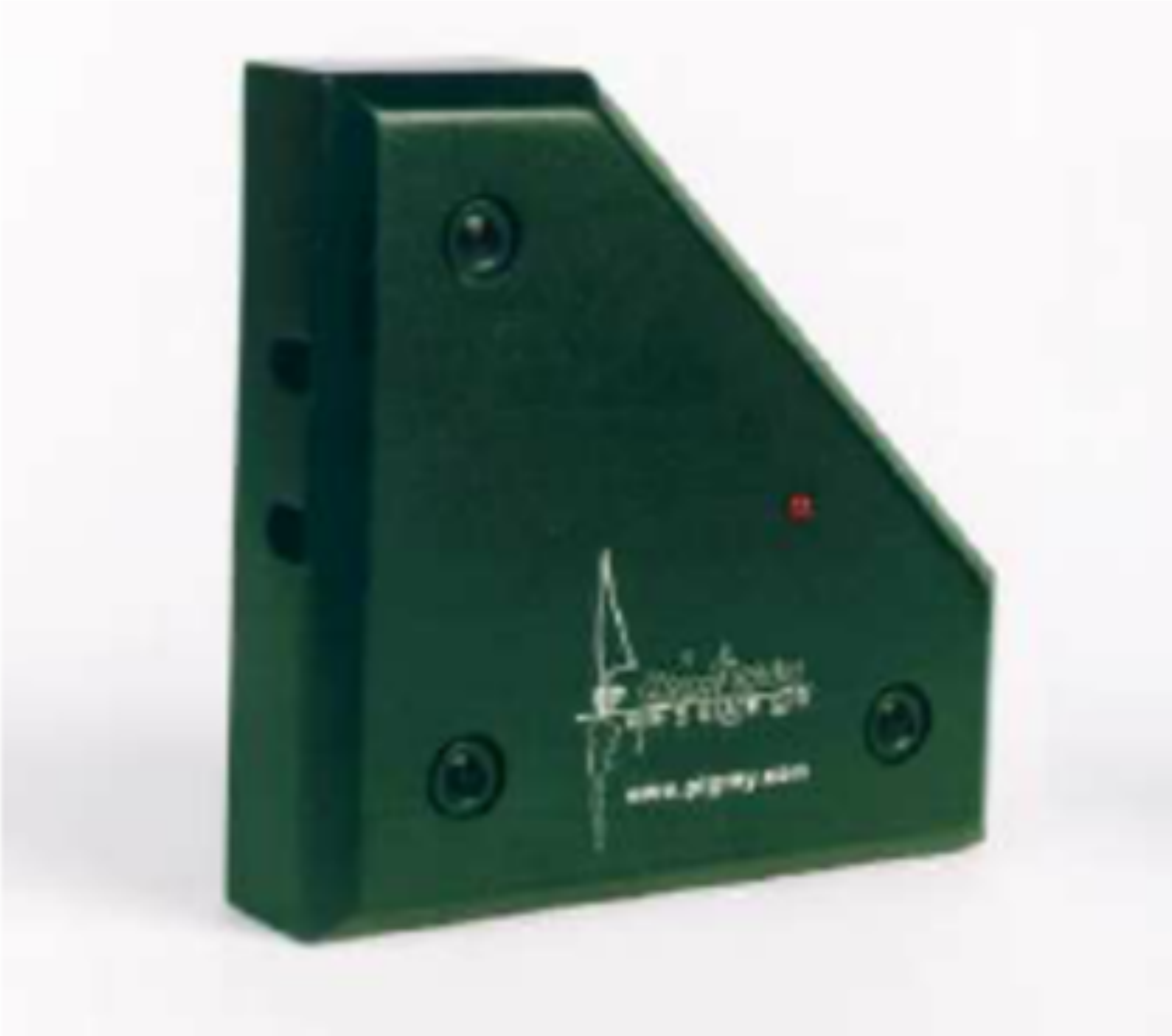
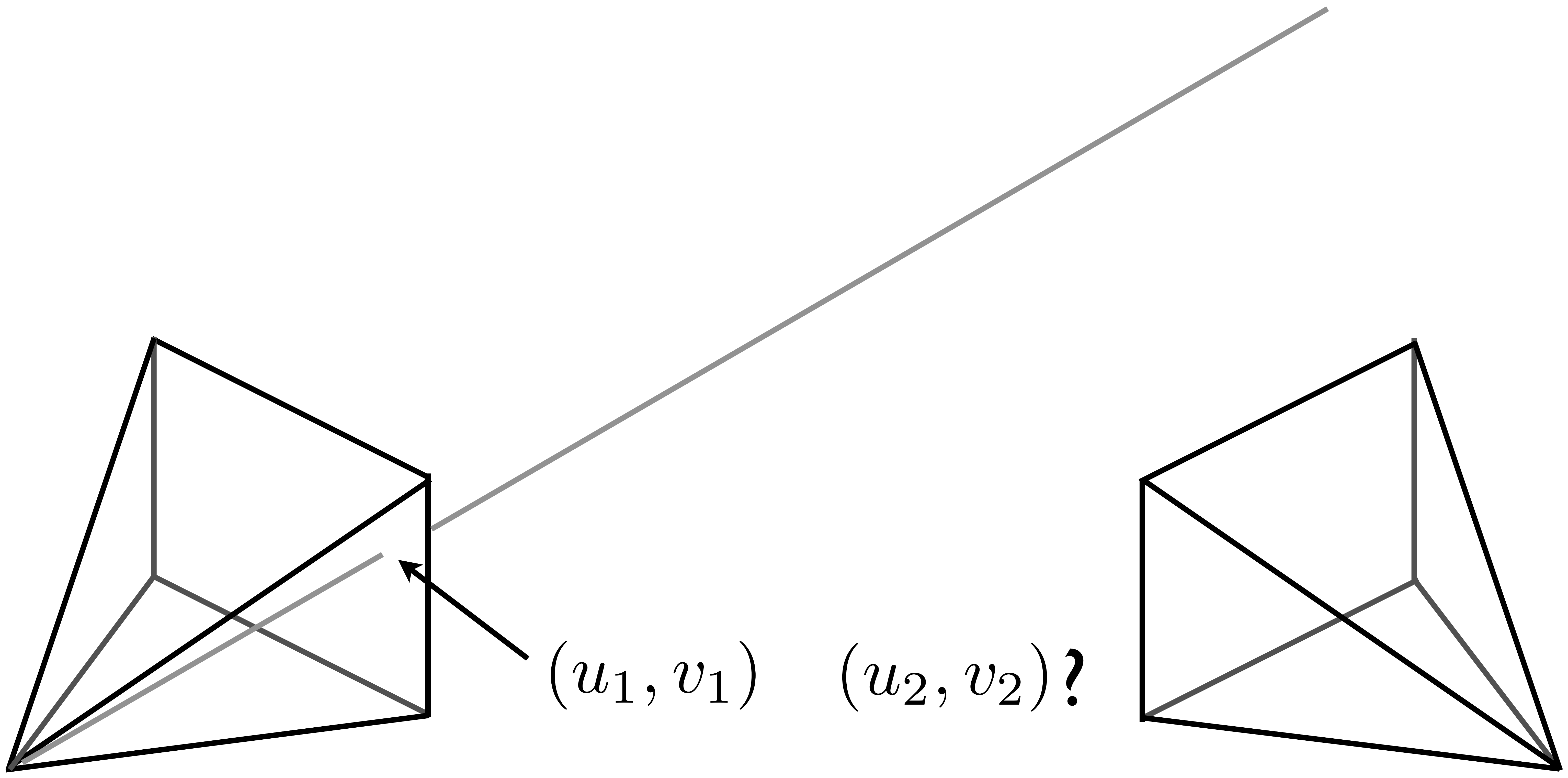


Image credit: Point Grey Research

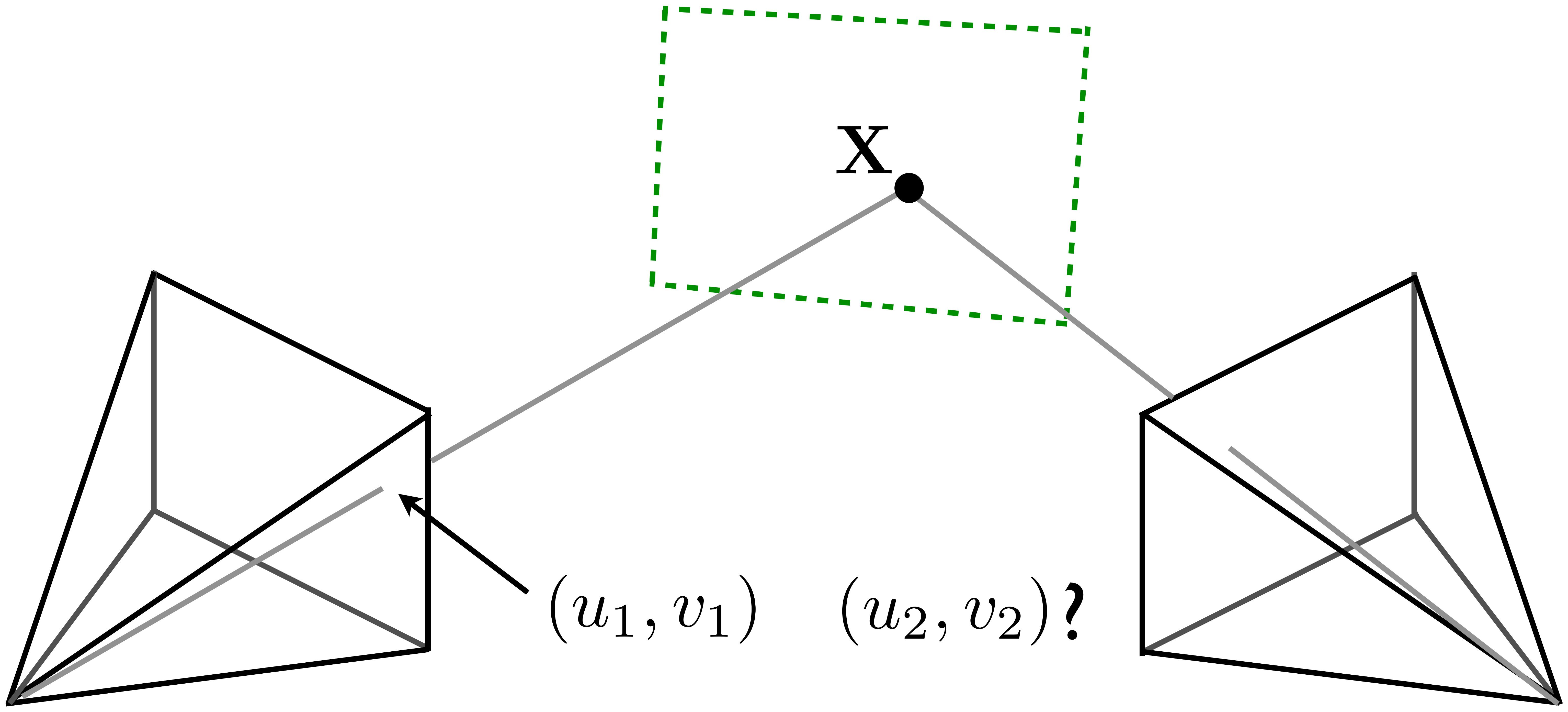
2-view **Geometry**

How do we find dense correspondences between two views?



2-view **Geometry**

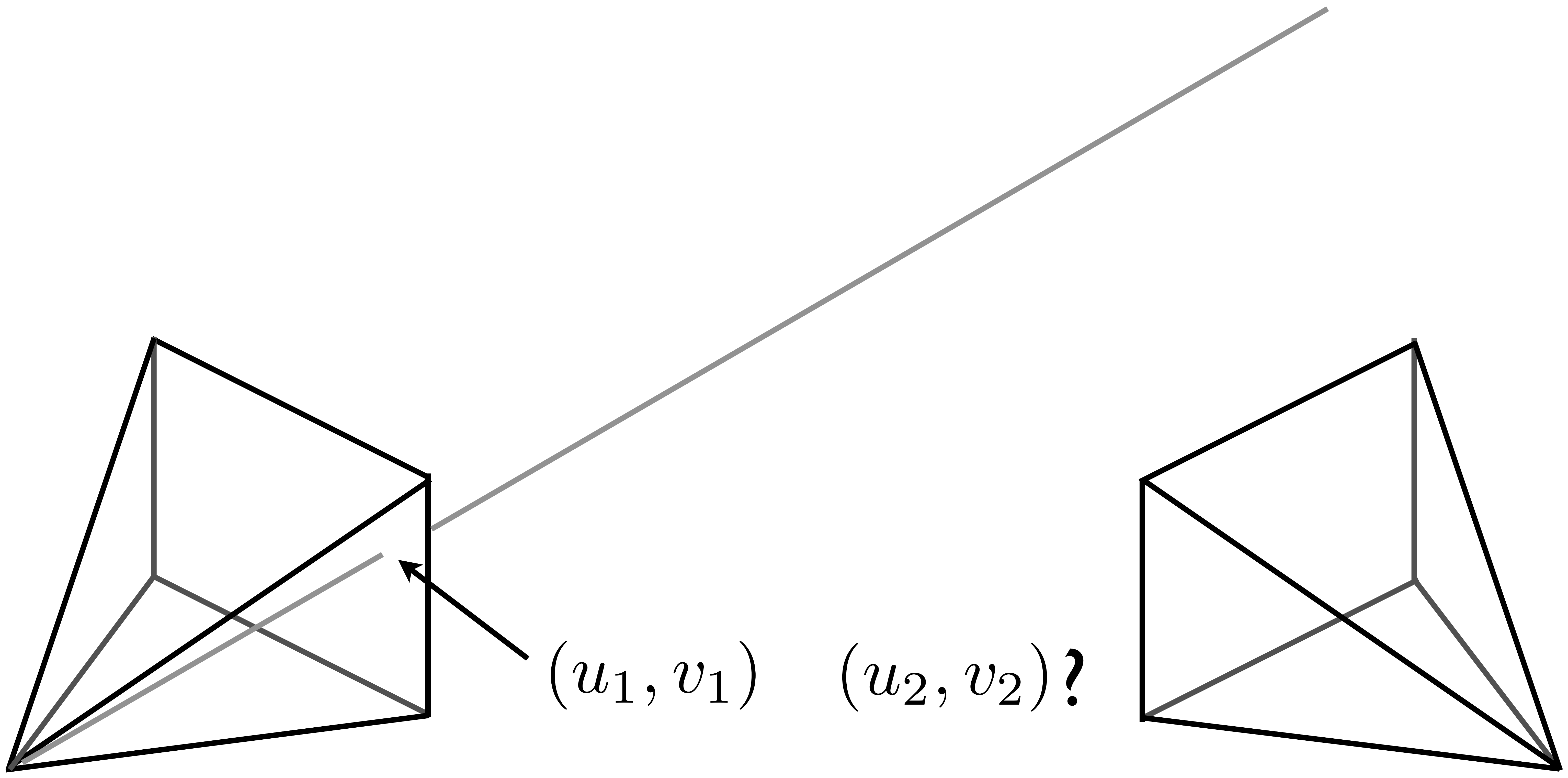
How do we find dense correspondences between two views?



Planar case: the mapping can be obtained by a homography

2-view **Geometry**

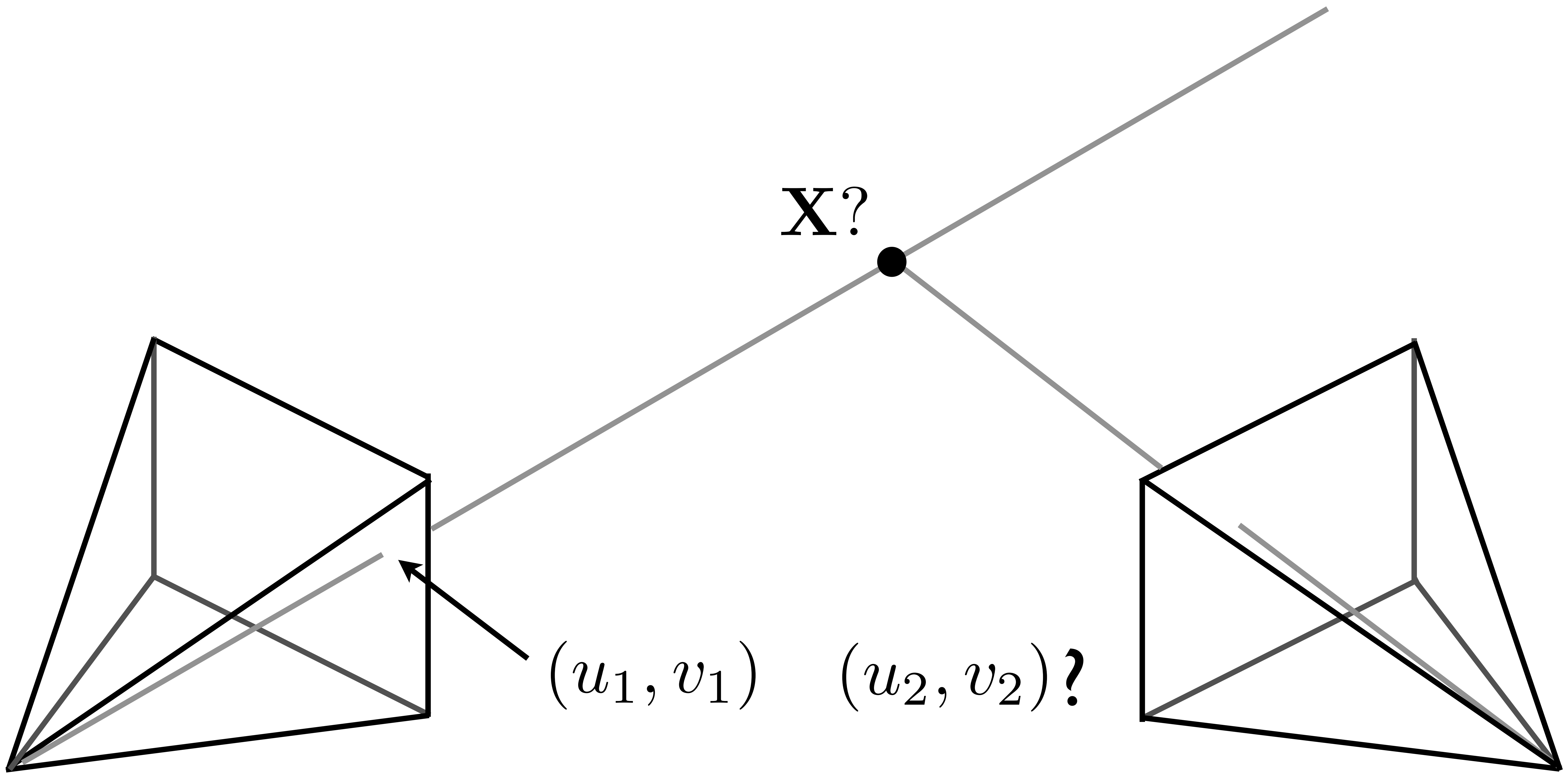
How do we find dense correspondences between two views?



Non-planar case: depends on the depth of the 3D point

2-view **Geometry**

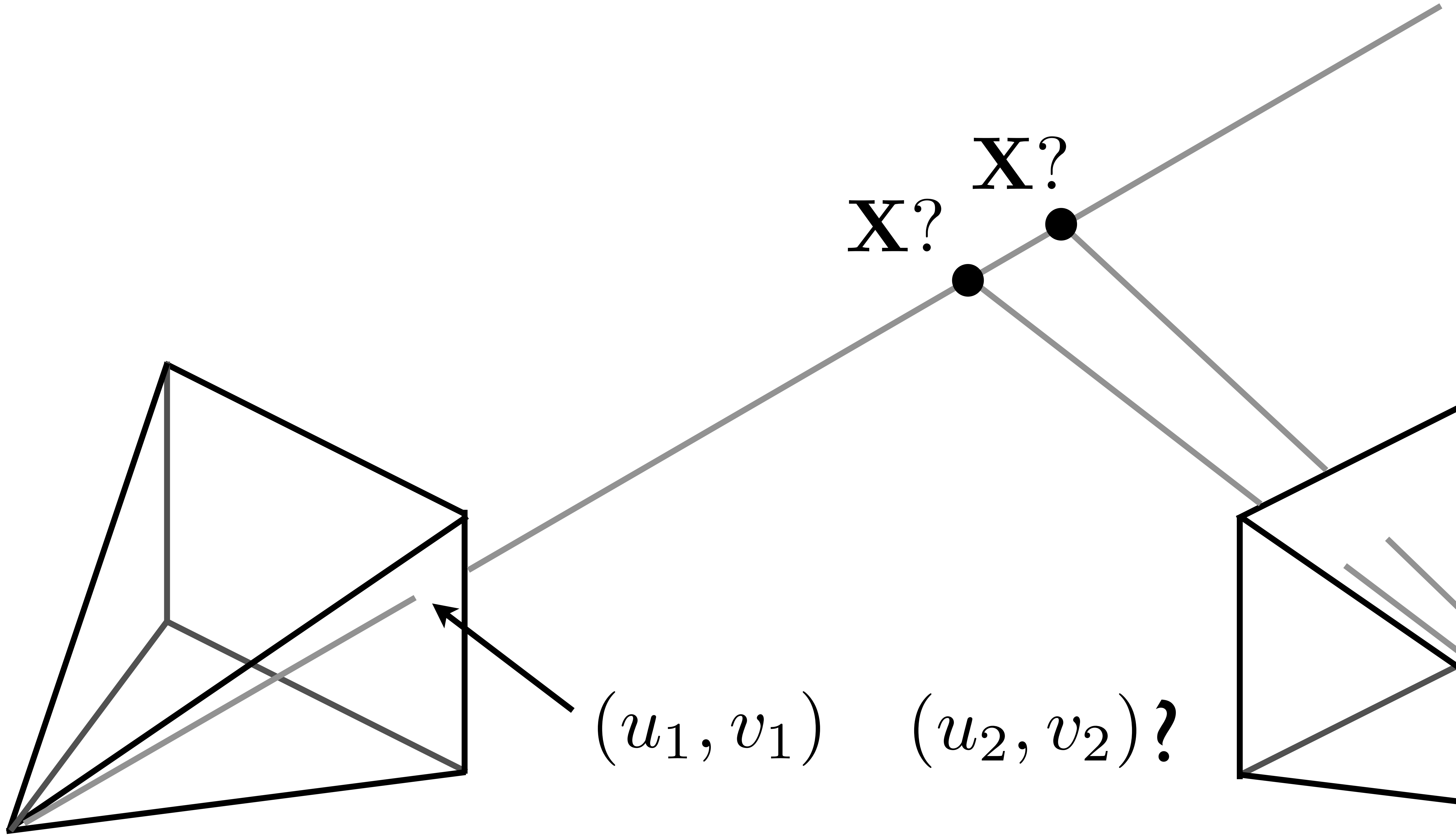
How do we find dense correspondences between two views?



Non-planar case: depends on the depth of the 3D point

2-view **Geometry**

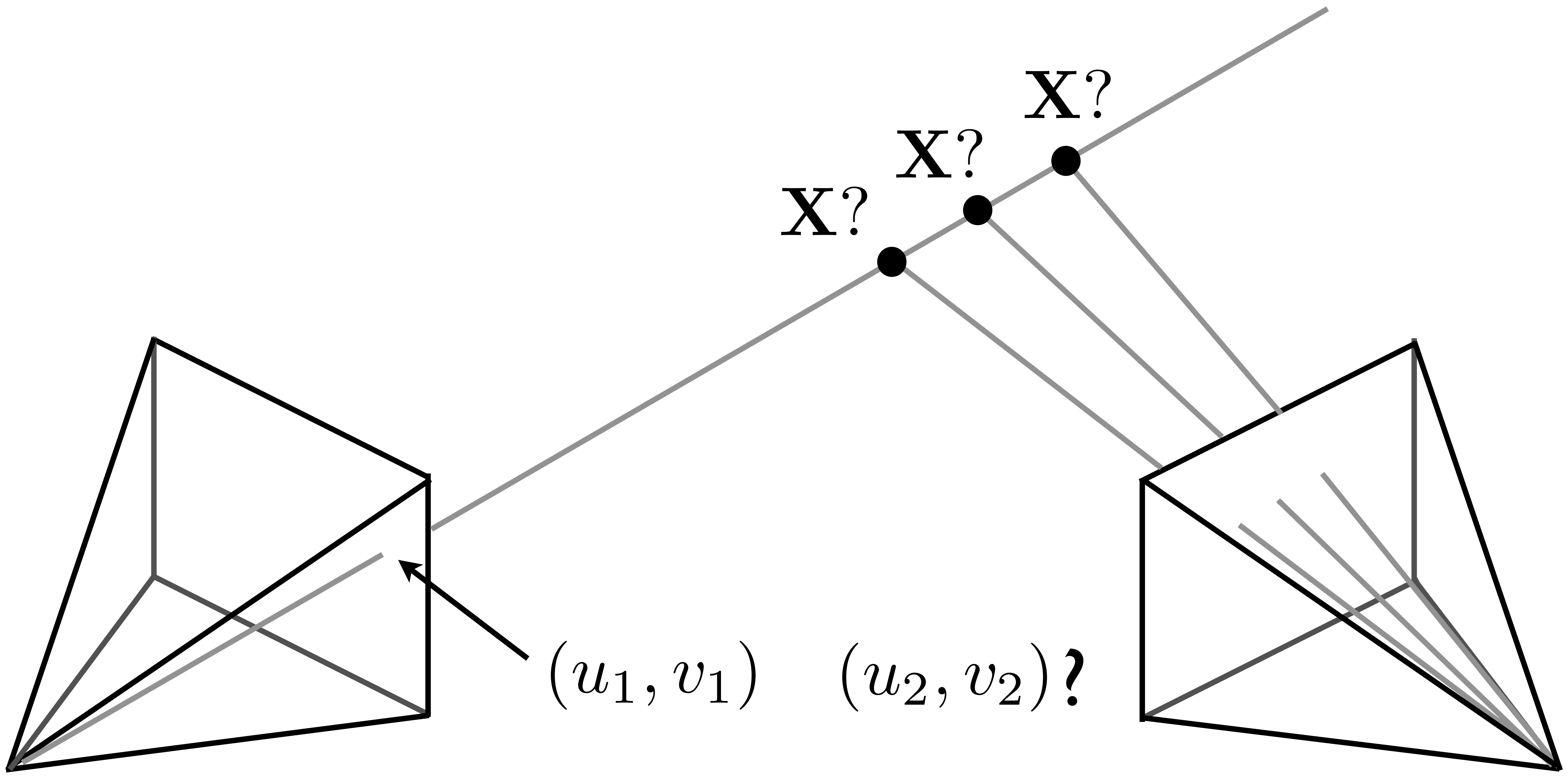
How do we find dense correspondences between two views?



Non-planar case: depends on the depth of the 3D point

2-view **Geometry**

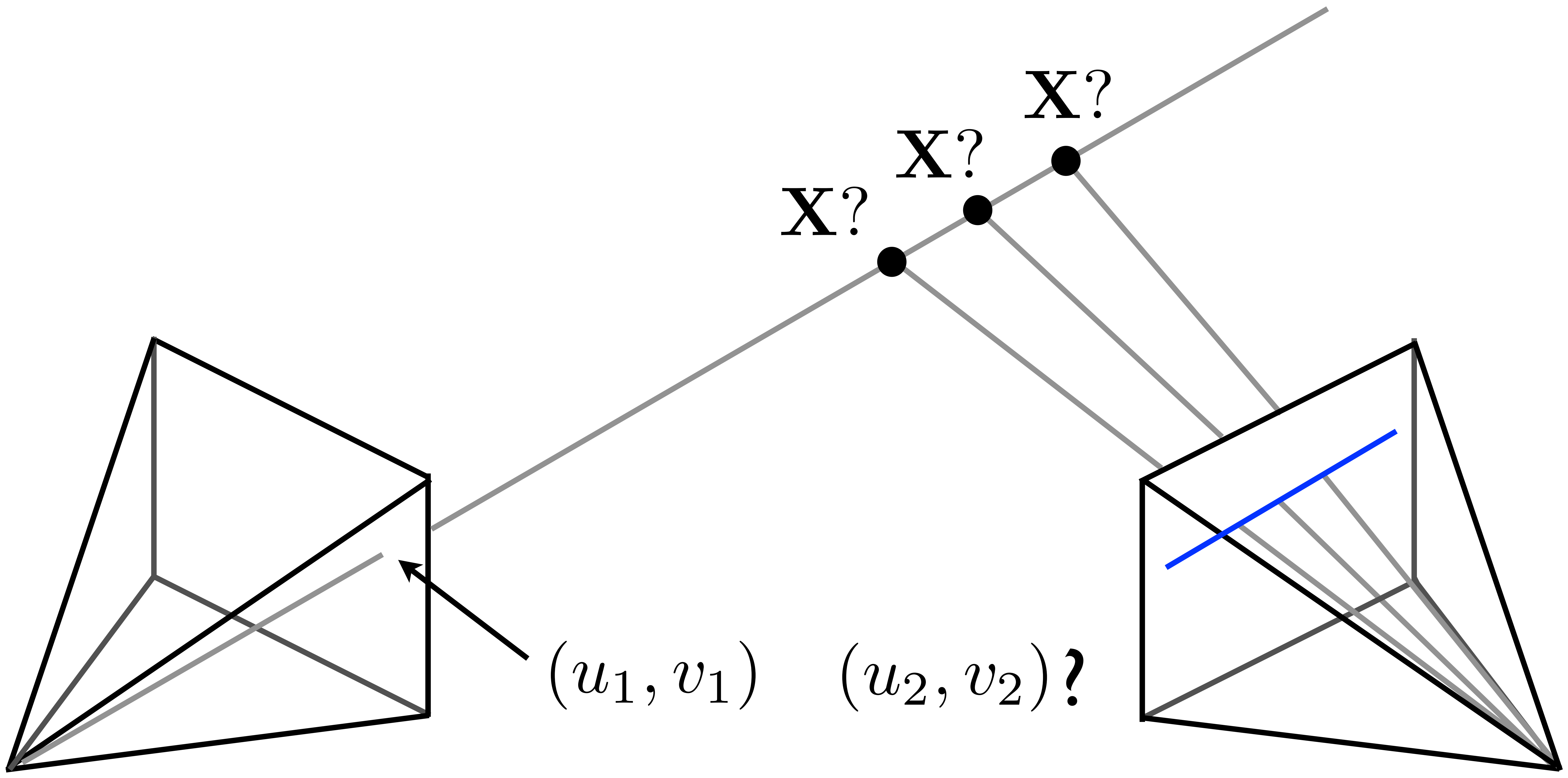
How do we find dense correspondences between two views?



Non-planar case: depends on the depth of the 3D point

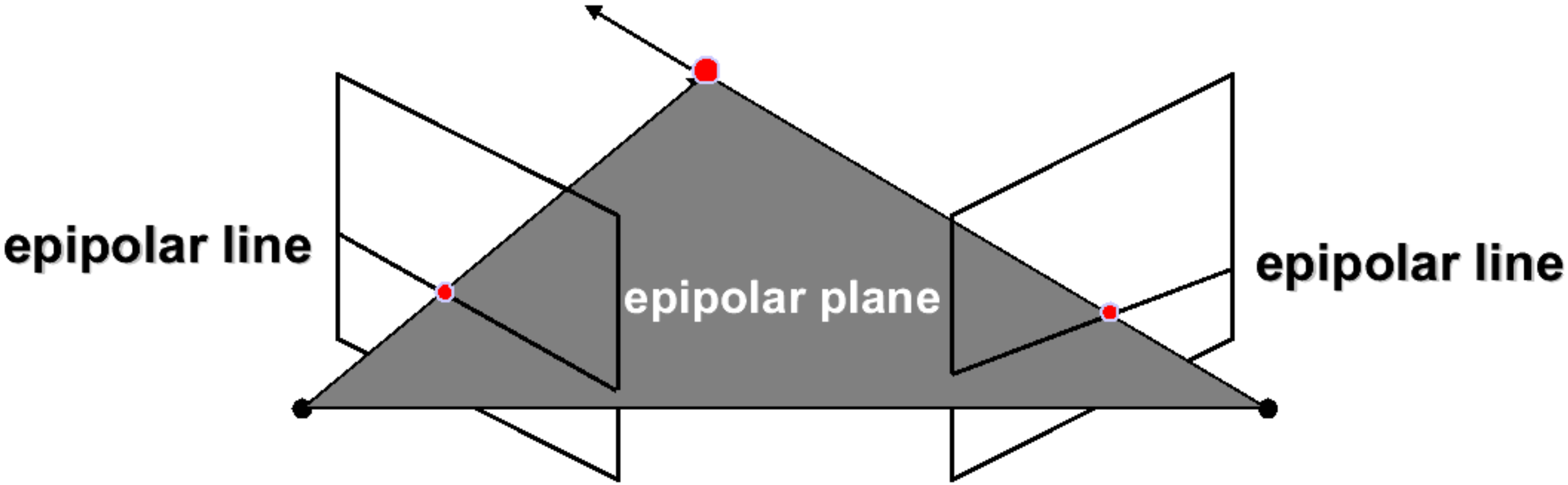
Epipolar Line

How do we find dense correspondences between two views?



A point in Image 1 must lie along the **line** in Image 2

The **Epipolar** Constraint



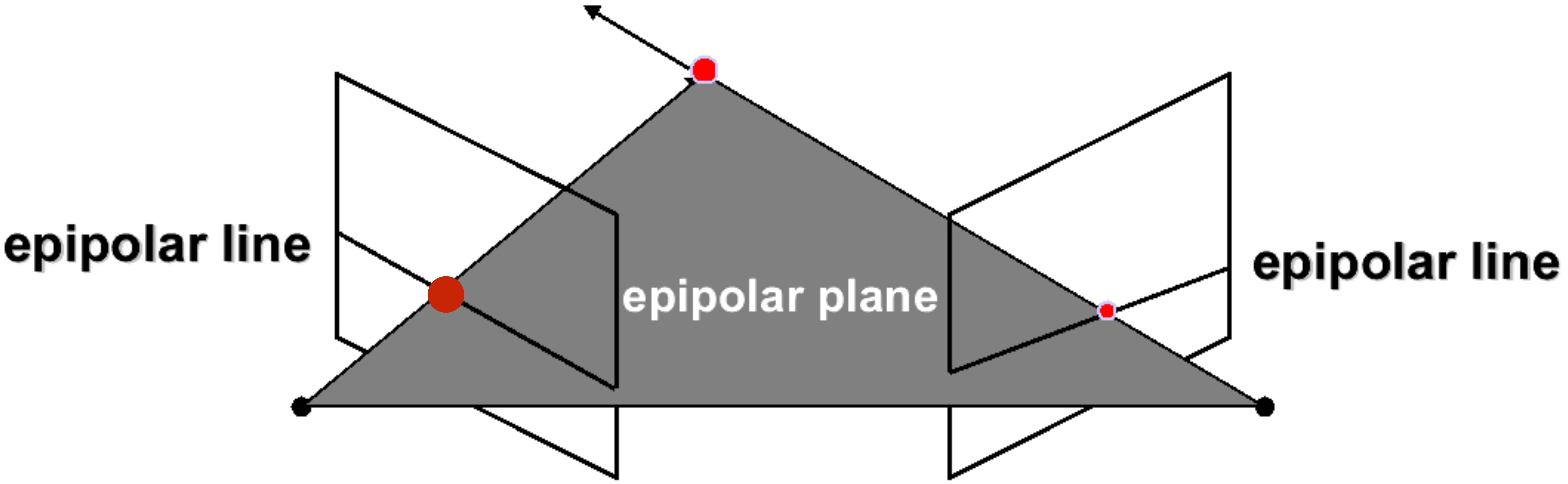
Matching points lie along corresponding epipolar lines

Reduces correspondence problem to 1D search along conjugate epipolar lines

Greatly reduces cost and ambiguity of matching

Slide credit: Steve Seitz

The **Epipolar** Constraint



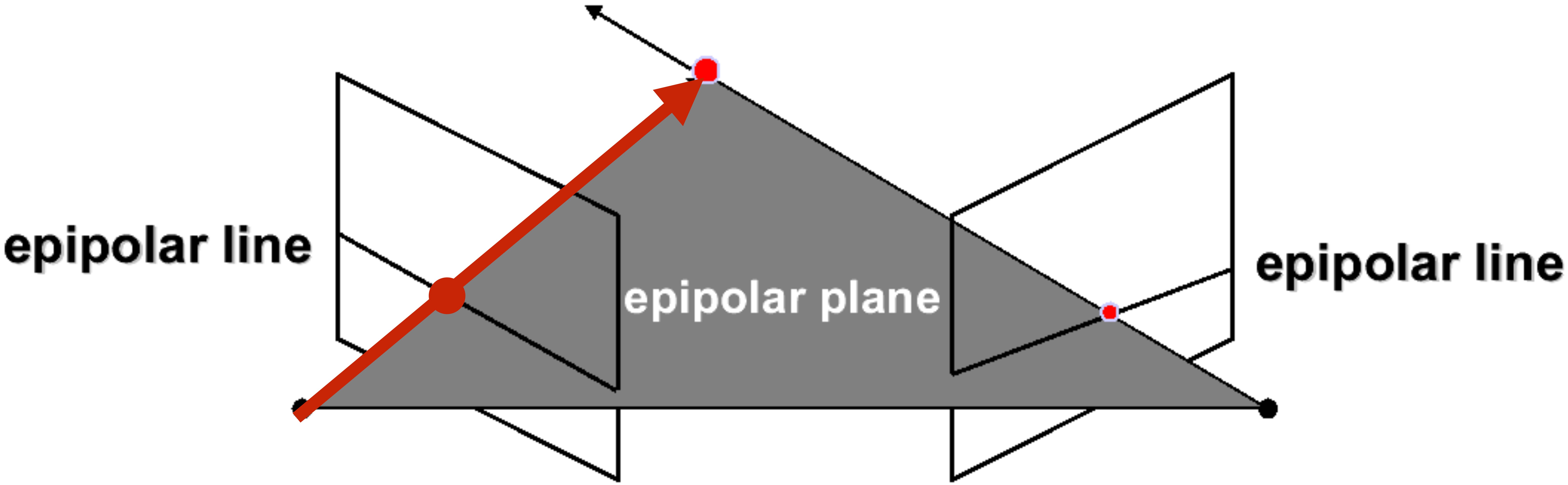
Matching points lie along corresponding epipolar lines

Reduces correspondence problem to 1D search along conjugate epipolar lines

Greatly reduces cost and ambiguity of matching

Slide credit: Steve Seitz

The **Epipolar** Constraint



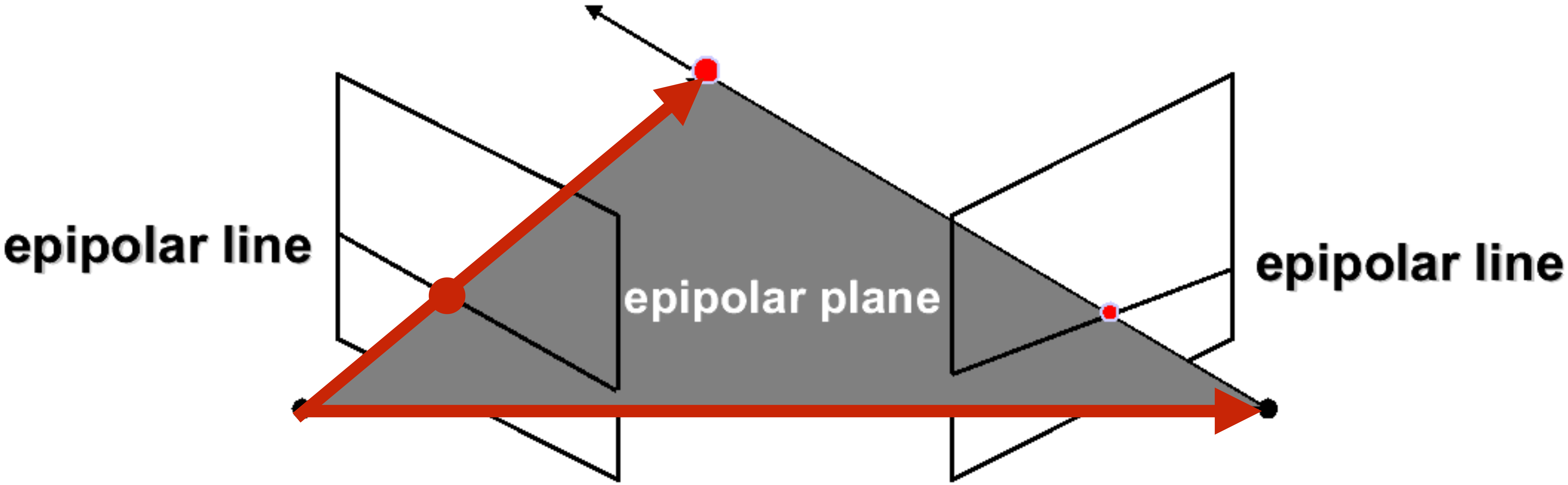
Matching points lie along corresponding epipolar lines

Reduces correspondence problem to 1D search along conjugate epipolar lines

Greatly reduces cost and ambiguity of matching

Slide credit: Steve Seitz

The **Epipolar** Constraint



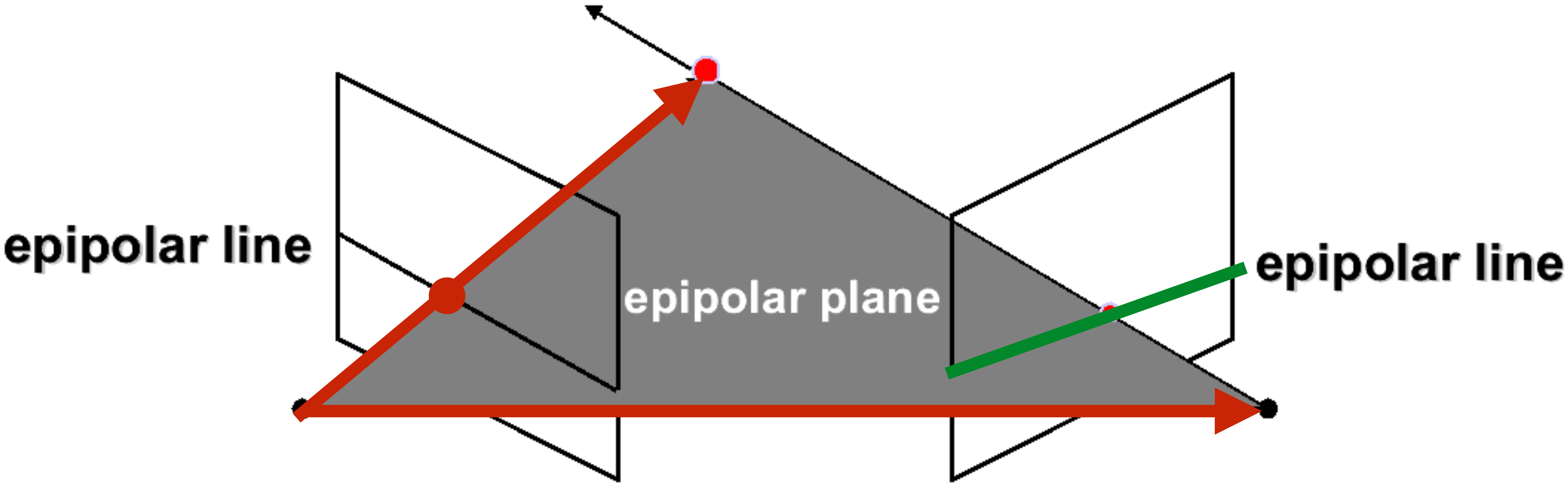
Matching points lie along corresponding epipolar lines

Reduces correspondence problem to 1D search along conjugate epipolar lines

Greatly reduces cost and ambiguity of matching

Slide credit: Steve Seitz

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

2-view **Stereo**

Search over matches constrained to (epipolar) line

(reduces to 1d search)

2-view **Stereo**

Search over matches constrained to (epipolar) line

(reduces to 1d search)

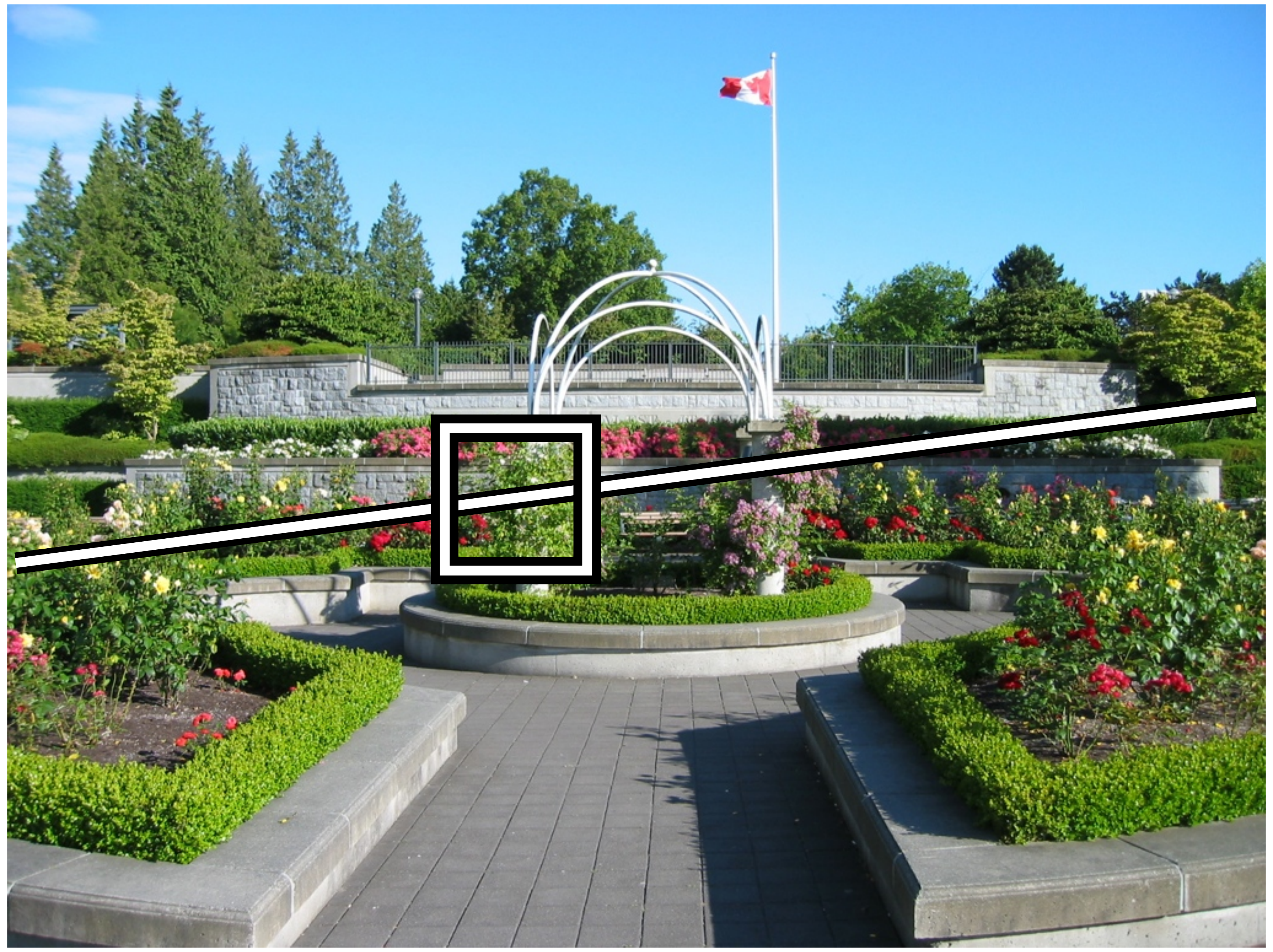
2-view **Stereo**

Search over matches constrained to (epipolar) line

(reduces to 1d search)

2-view **Stereo**

Search over matches constrained to (epipolar) line



(reduces to 1d search)

2-view **Stereo**

Search over matches constrained to (epipolar) line

(reduces to 1d search)

2-view **Stereo**

Search over matches constrained to (epipolar) line

(reduces to 1d search)

2-view **Stereo**

Search over matches constrained to (epipolar) line

(reduces to 1d search)

2-view **Stereo**

Search over matches constrained to (epipolar) line

(reduces to 1d search)

2-view **Stereo**

Search over matches constrained to (epipolar) line

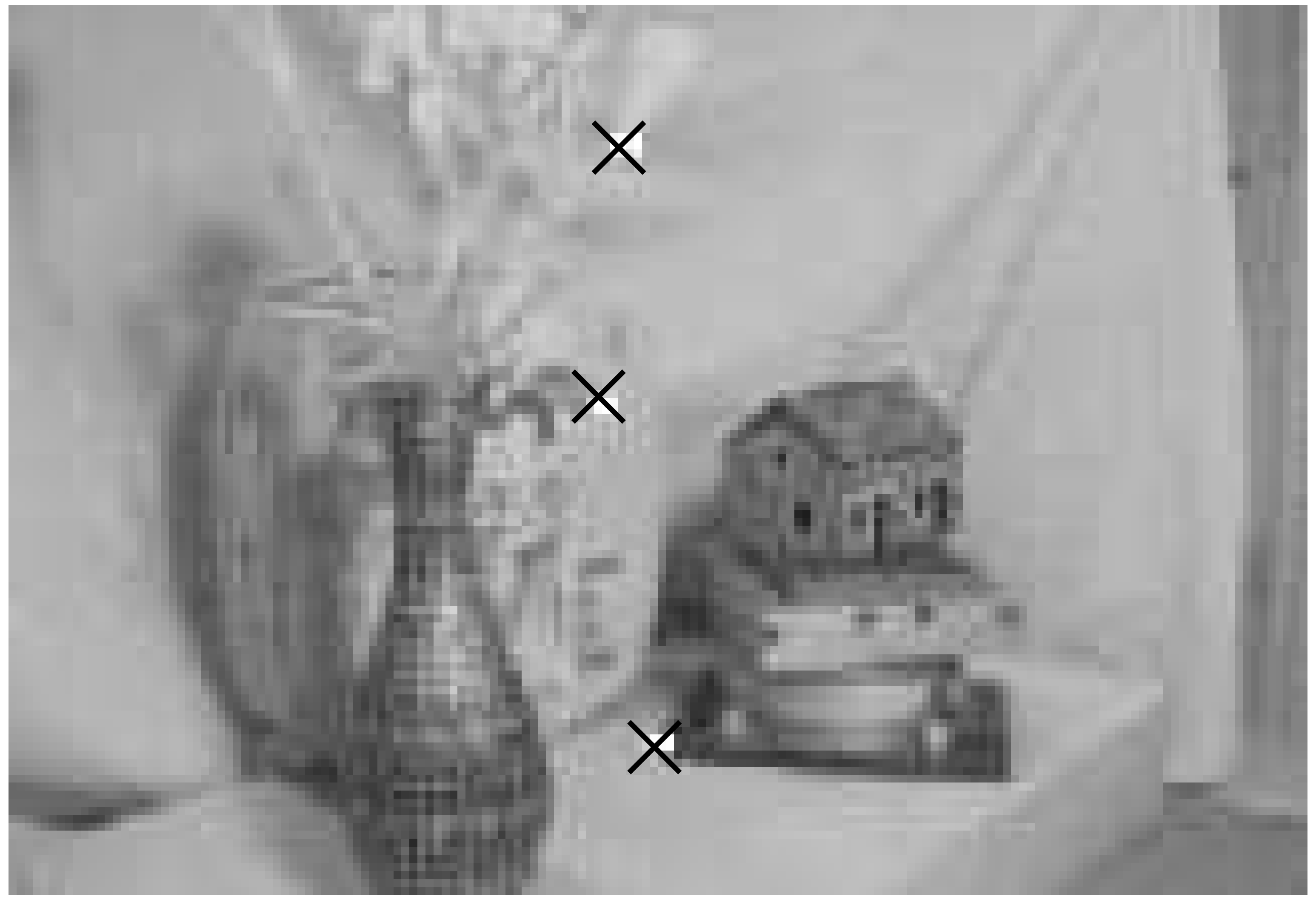
(reduces to 1d search)

2-view **Stereo**

Search over matches constrained to (epipolar) line

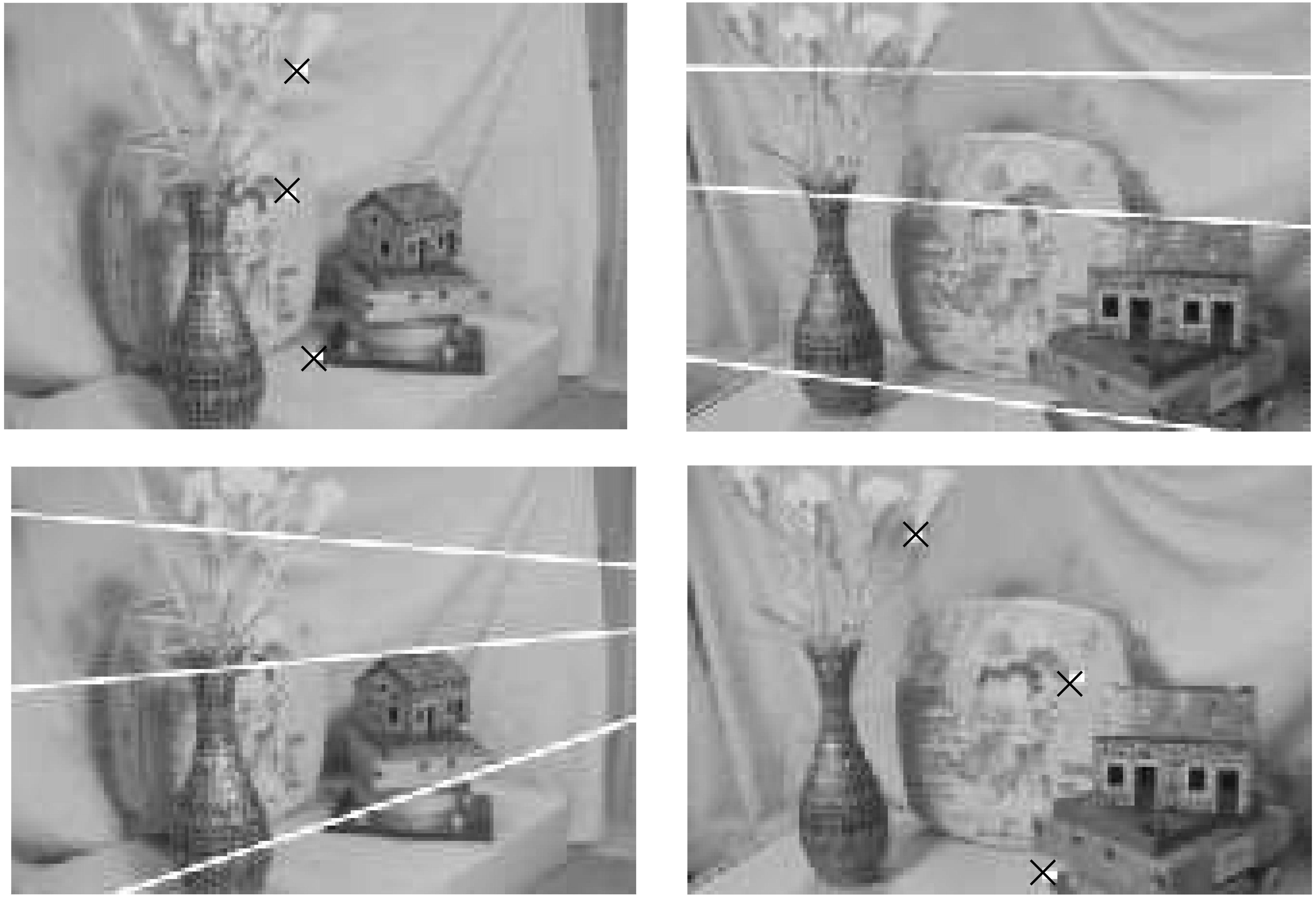
(reduces to 1d search)

Visualization of **Epipolar Lines**



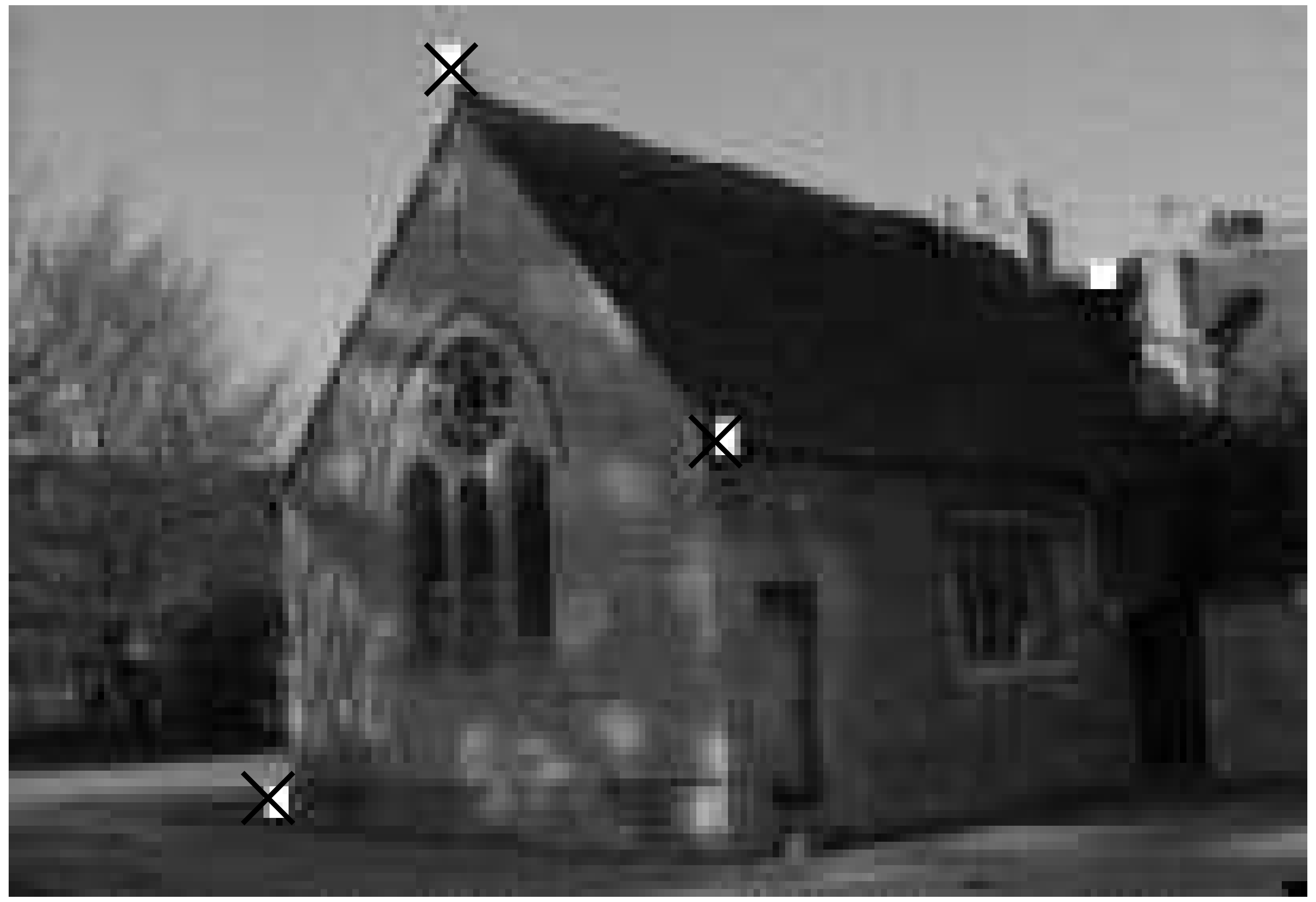
[R. Cipolla]

Visualization of **Epipolar Lines**



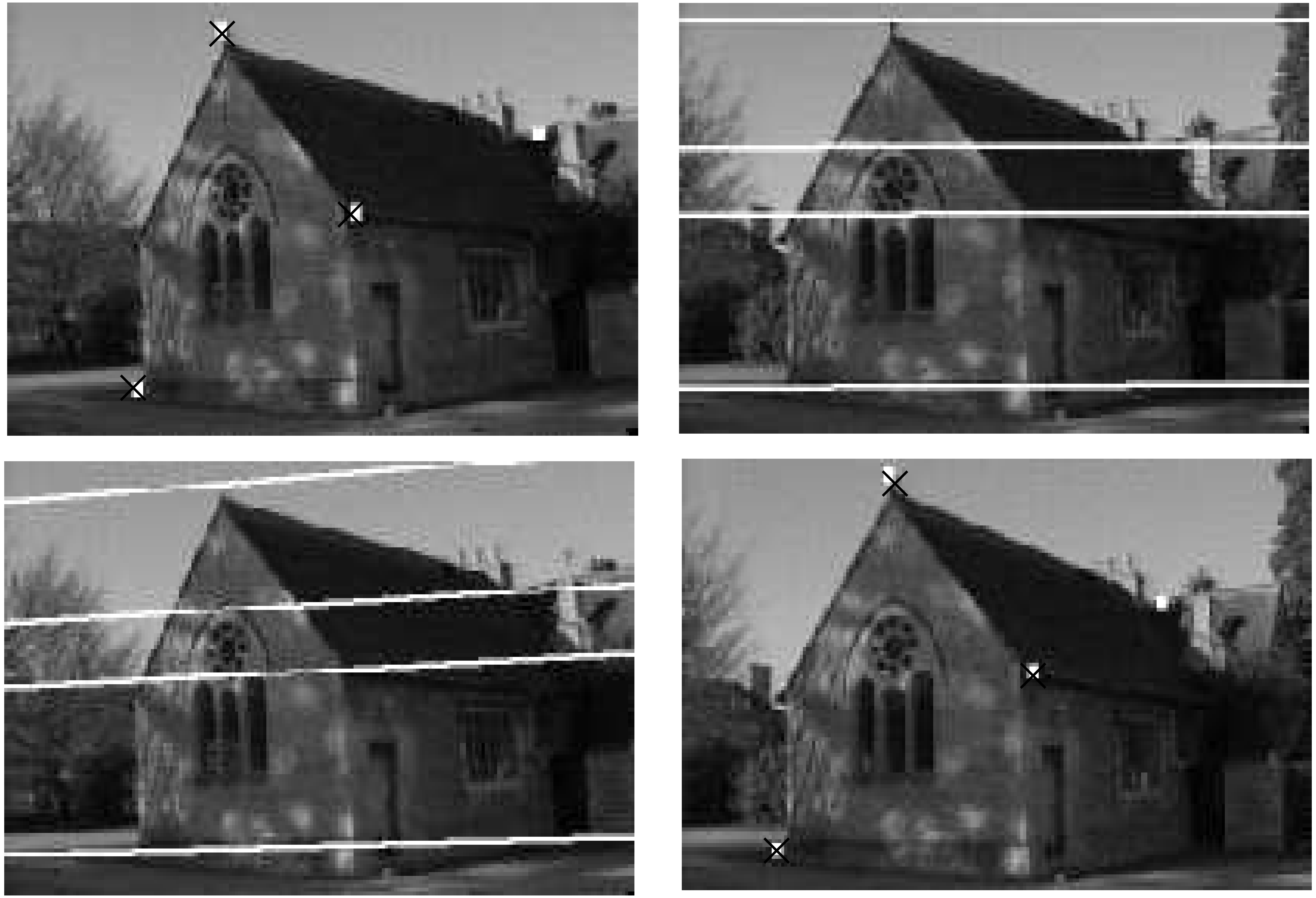
[R. Cipolla]

Visualization of **Epipolar Lines**



[R. Cipolla]

Visualization of **Epipolar Lines**

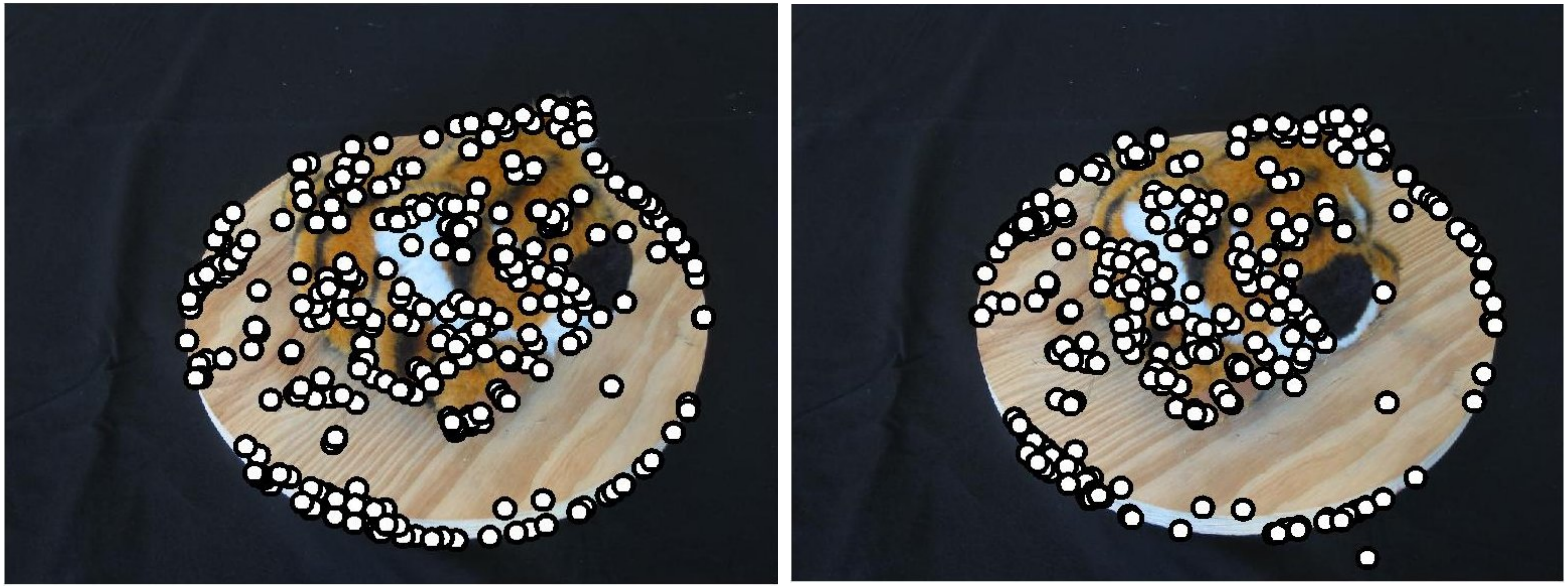


[R. Cipolla]

Improving RANSAC + Alignment with **Epipolar Geometry**

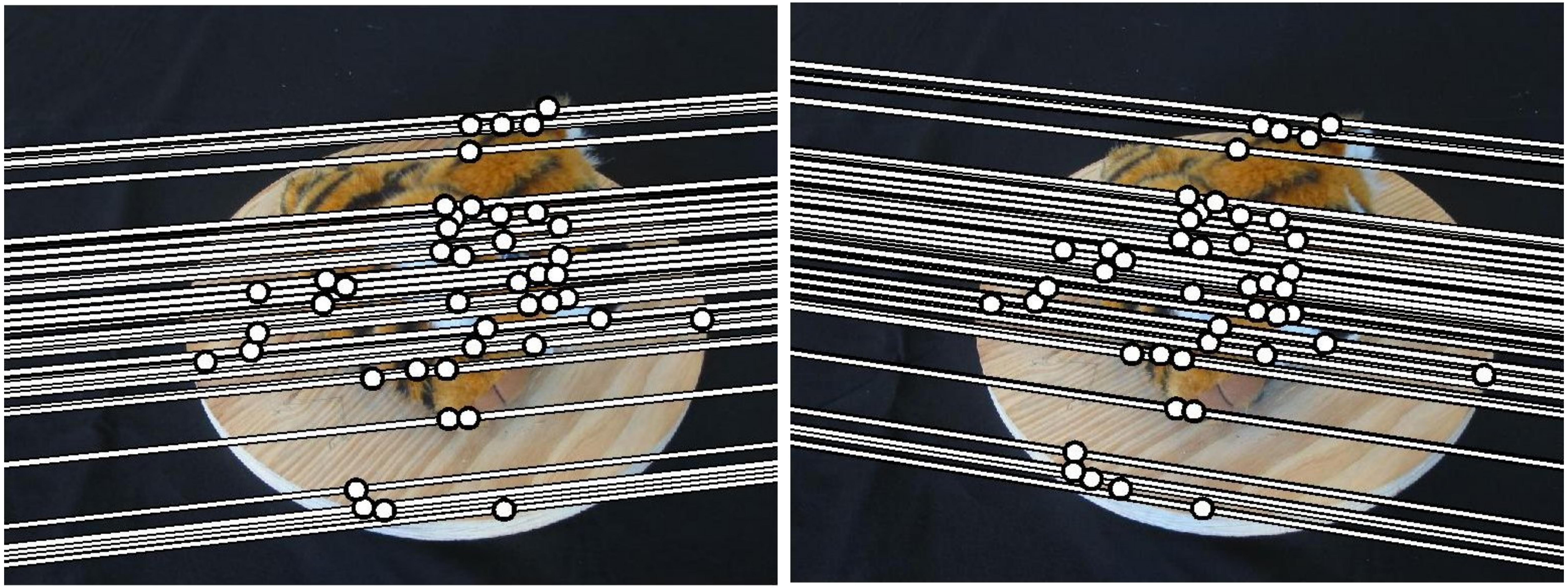
Improving RANSAC + Alignment with **Epipolar Geometry**

Raw SIFT features and their matches



Improving RANSAC + Alignment with **Epipolar Geometry**

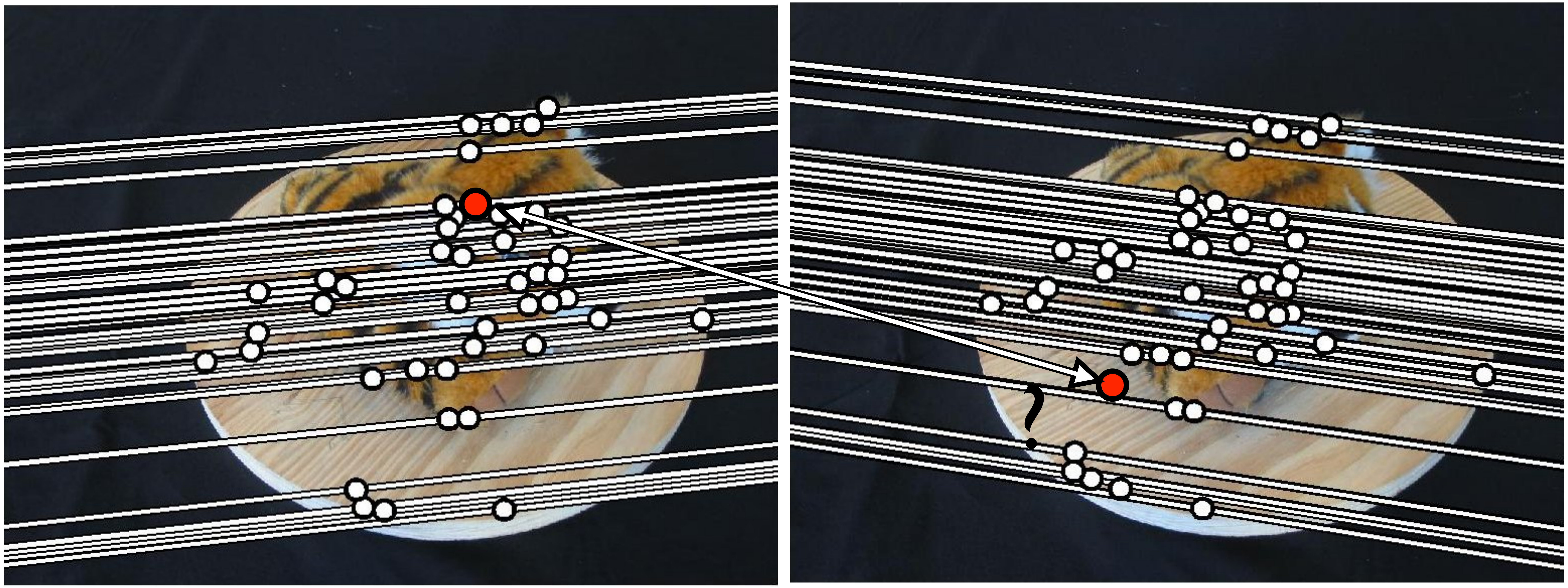
Instead of matching purely based on SIFT descriptor, leverage geometry to obtain matches close to epipolar lines



(gives more consistent geometrically valid matches)

Improving RANSAC + Alignment with **Epipolar Geometry**

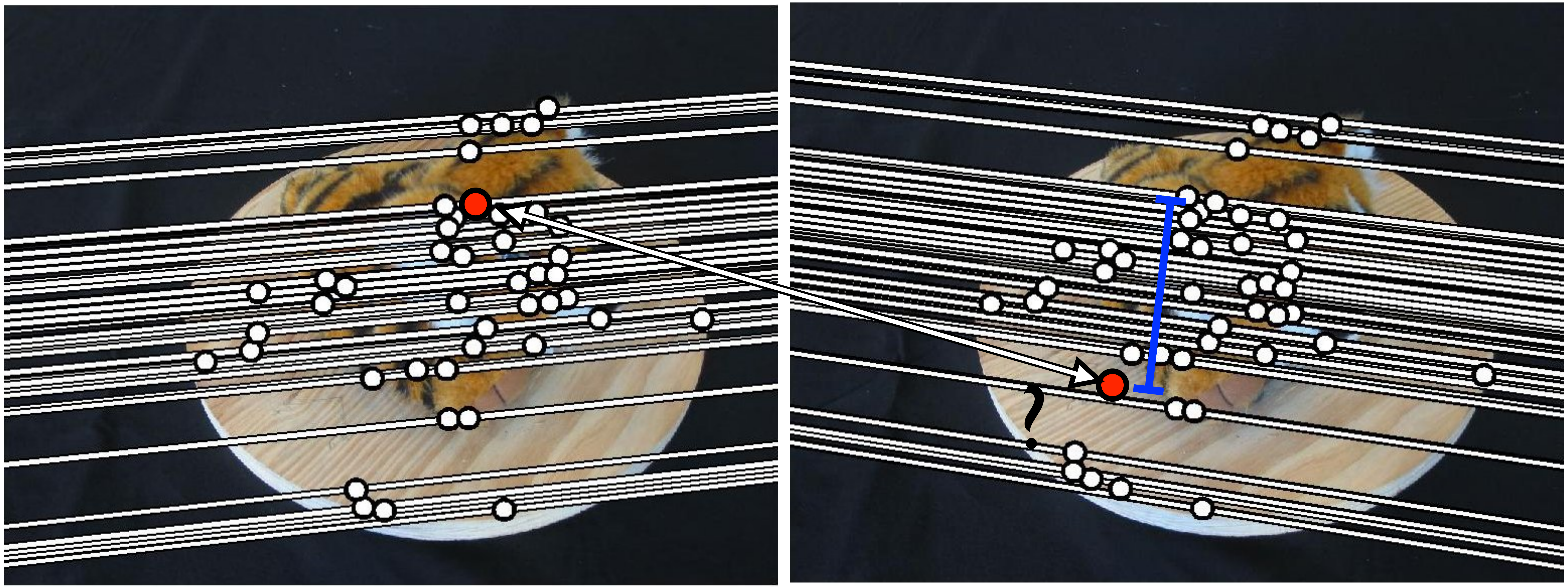
Instead of matching purely based on SIFT descriptor, leverage geometry to obtain matches close to epipolar lines



(gives more consistent geometrically valid matches)

Improving RANSAC + Alignment with **Epipolar Geometry**

Instead of matching purely based on SIFT descriptor, leverage geometry to obtain matches close to epipolar lines



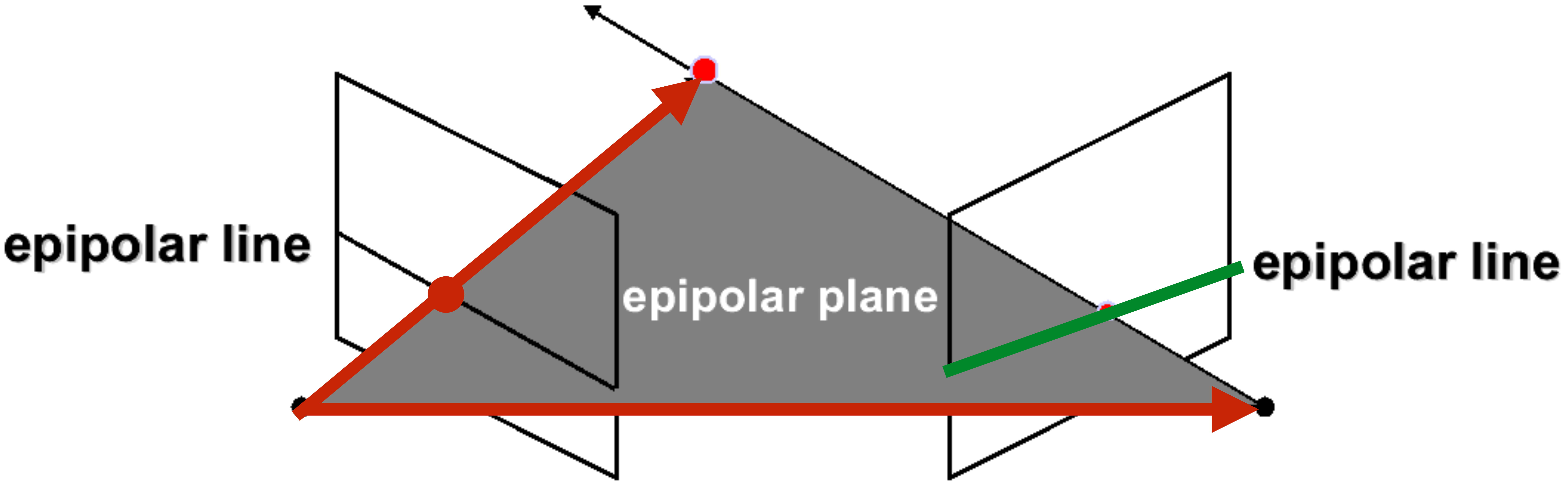
(gives more consistent geometrically valid matches)

Improving RANSAC + Alignment with **Epipolar Geometry**

Better matches lead to fewer iterations of RANSAC

(gives more consistent geometrically valid matches)

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

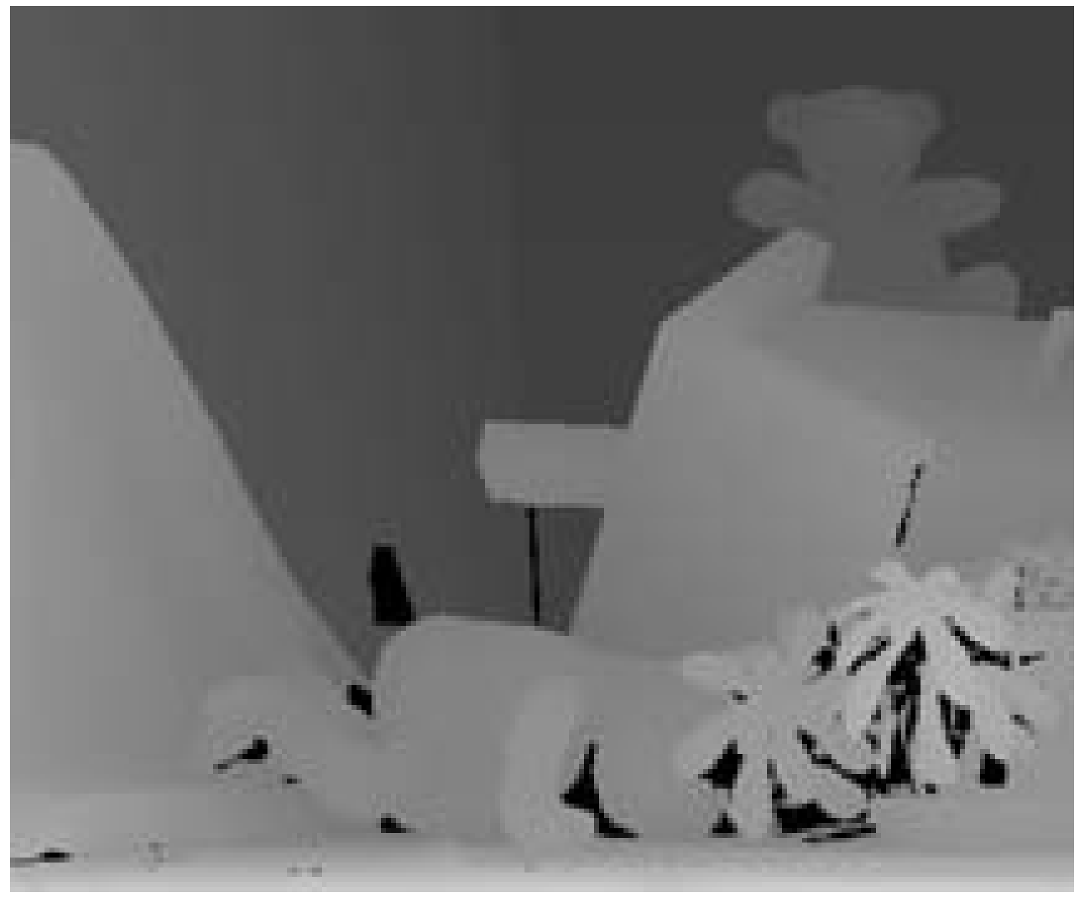
We assume images have been **rectified** so that epipolar lines correspond to scan lines

— Simplifies algorithms

— Improves efficiency

Stereo Matching in Rectified Images

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



— Stereo algorithms search along scanlines for match

— Distance along the scanline (difference in x coordinate) for a corresponding feature is called **disparity**

Stereo Matching in Rectified Images

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

— Stereo algorithms search along scanlines for match

— Distance along the scanline (difference in x coordinate) for a corresponding feature is called **disparity**

Stereo Matching in Rectified Images

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

— Stereo algorithms search along scanlines for match

— Distance along the scanline (difference in x coordinate) for a corresponding feature is called **disparity**

Stereo Matching in Rectified Images

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

— Stereo algorithms search along scanlines for match

— Distance along the scanline (difference in x coordinate) for a corresponding feature is called **disparity**

Stereo Matching in Rectified Images

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

— Stereo algorithms search along scanlines for match

— Distance along the scanline (difference in x coordinate) for a corresponding feature is called **disparity**

Stereo Matching in Rectified Images

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

— Stereo algorithms search along scanlines for match

— Distance along the scanline (difference in x coordinate) for a corresponding feature is called **disparity**

Stereo Matching in Rectified Images

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

— Stereo algorithms search along scanlines for match

— Distance along the scanline (difference in x coordinate) for a corresponding feature is called **disparity**

Stereo Matching in Rectified Images

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

— Stereo algorithms search along scanlines for match

— Distance along the scanline (difference in x coordinate) for a corresponding feature is called **disparity**

Stereo Matching in Rectified Images

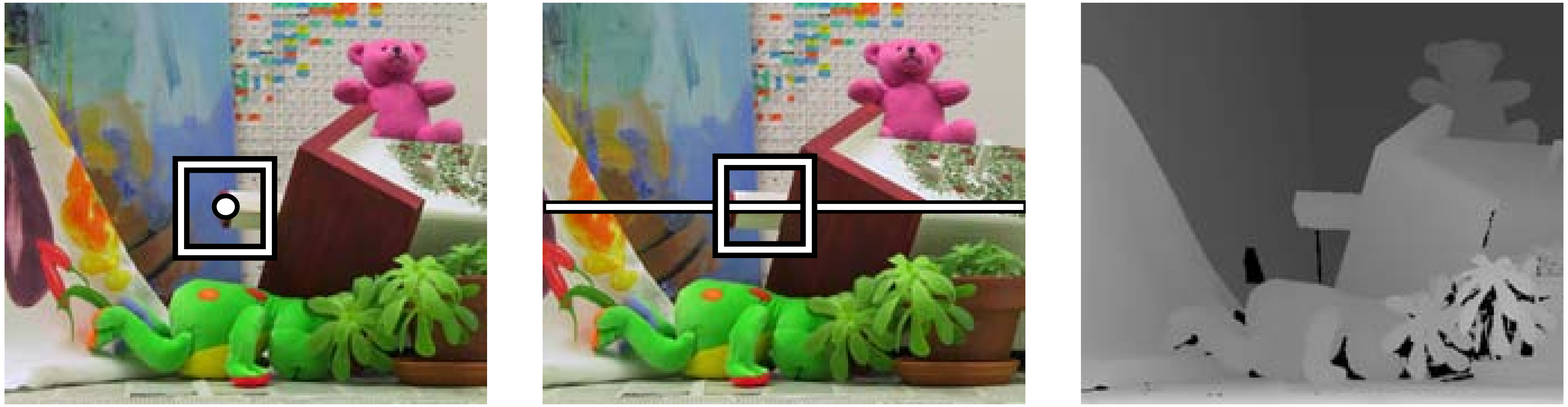
— In a standard stereo setup, where cameras are related by translation in the x direction, epipolar lines are horizontal

— Stereo algorithms search along scanlines for match

— Distance along the scanline (difference in x coordinate) for a corresponding feature is called **disparity**

Stereo Matching in Rectified Images

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



— Stereo algorithms search along scanlines for match

— Distance along the scanline (difference in x coordinate) for a corresponding feature is called **disparity**

Stereo Matching in Rectified Images

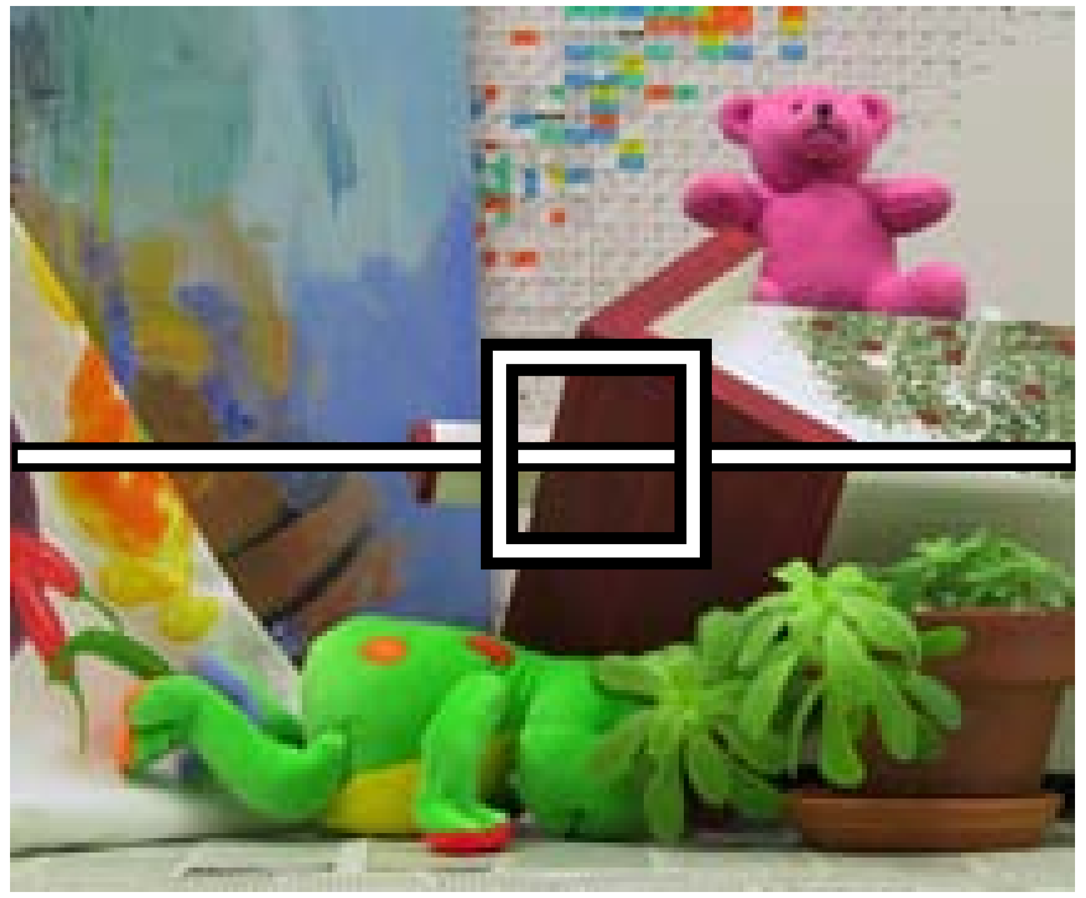
— In a standard stereo setup, where cameras are related by translation in the x direction, epipolar lines are horizontal

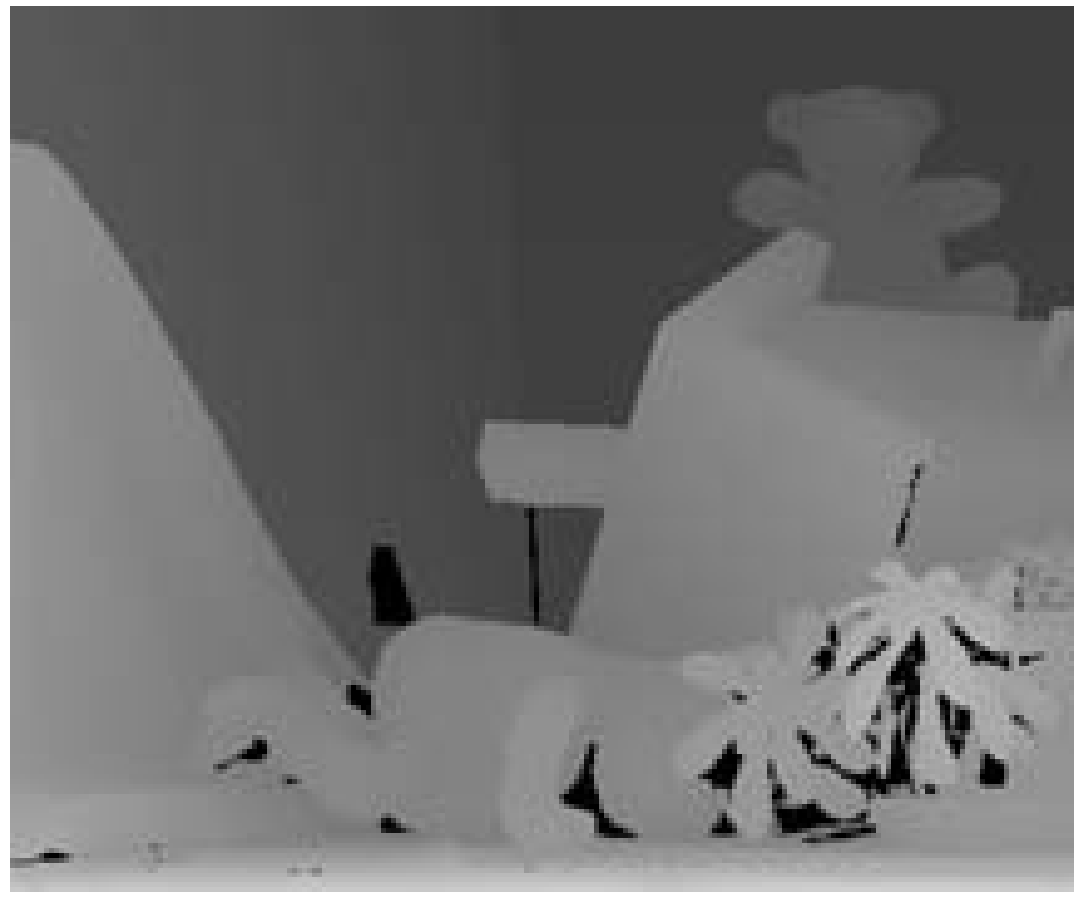
— Stereo algorithms search along scanlines for match

— Distance along the scanline (difference in x coordinate) for a corresponding feature is called **disparity**

Stereo Matching in Rectified Images

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





— Stereo algorithms search along scanlines for match

— Distance along the scanline (difference in x coordinate) for a corresponding feature is called **disparity**

Stereo Matching in Rectified Images (Left)

[D. Scharstein]

Stereo Matching in Rectified Images (Right)

[D. Scharstein]

Stereo Matching in Rectified Images (Right)

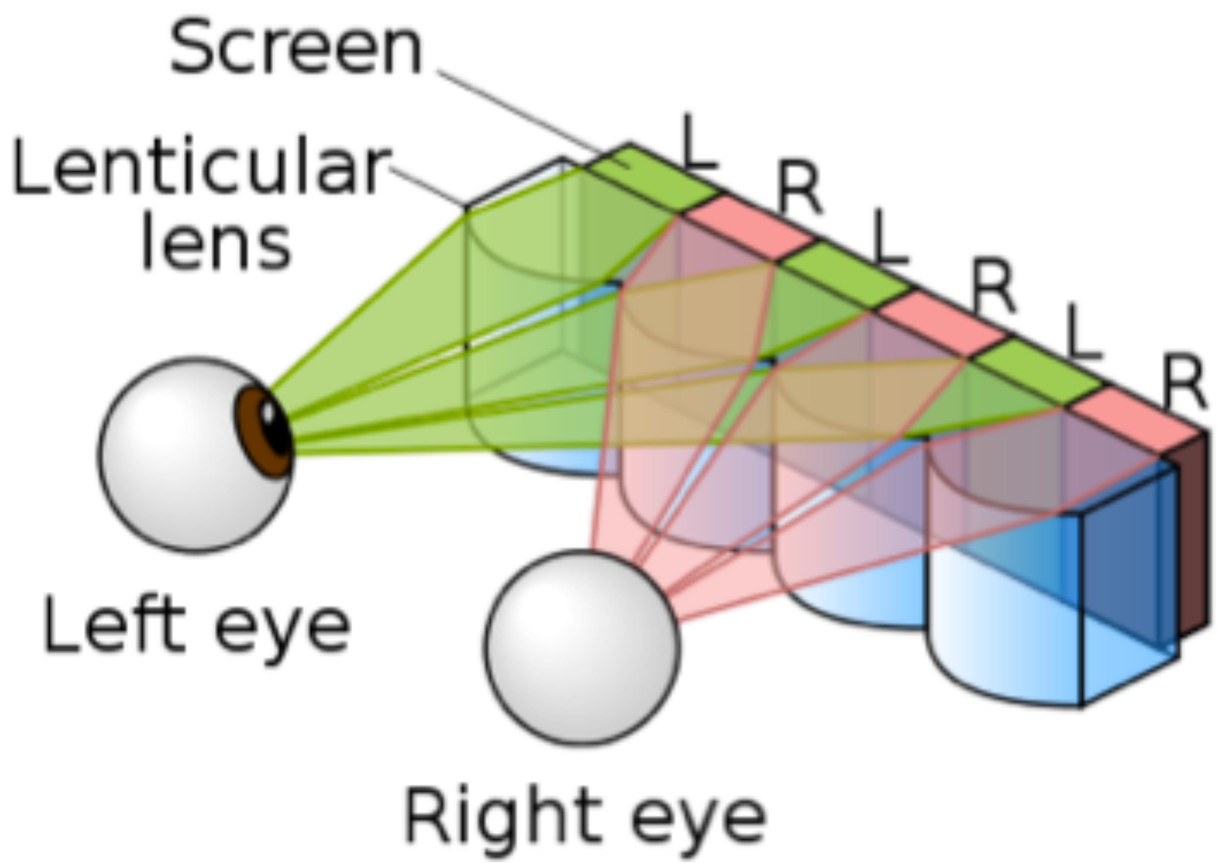
[D. Scharstein]

Anaglyph

Stereo pair with images encoded in different color channels

Stereo Displays

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



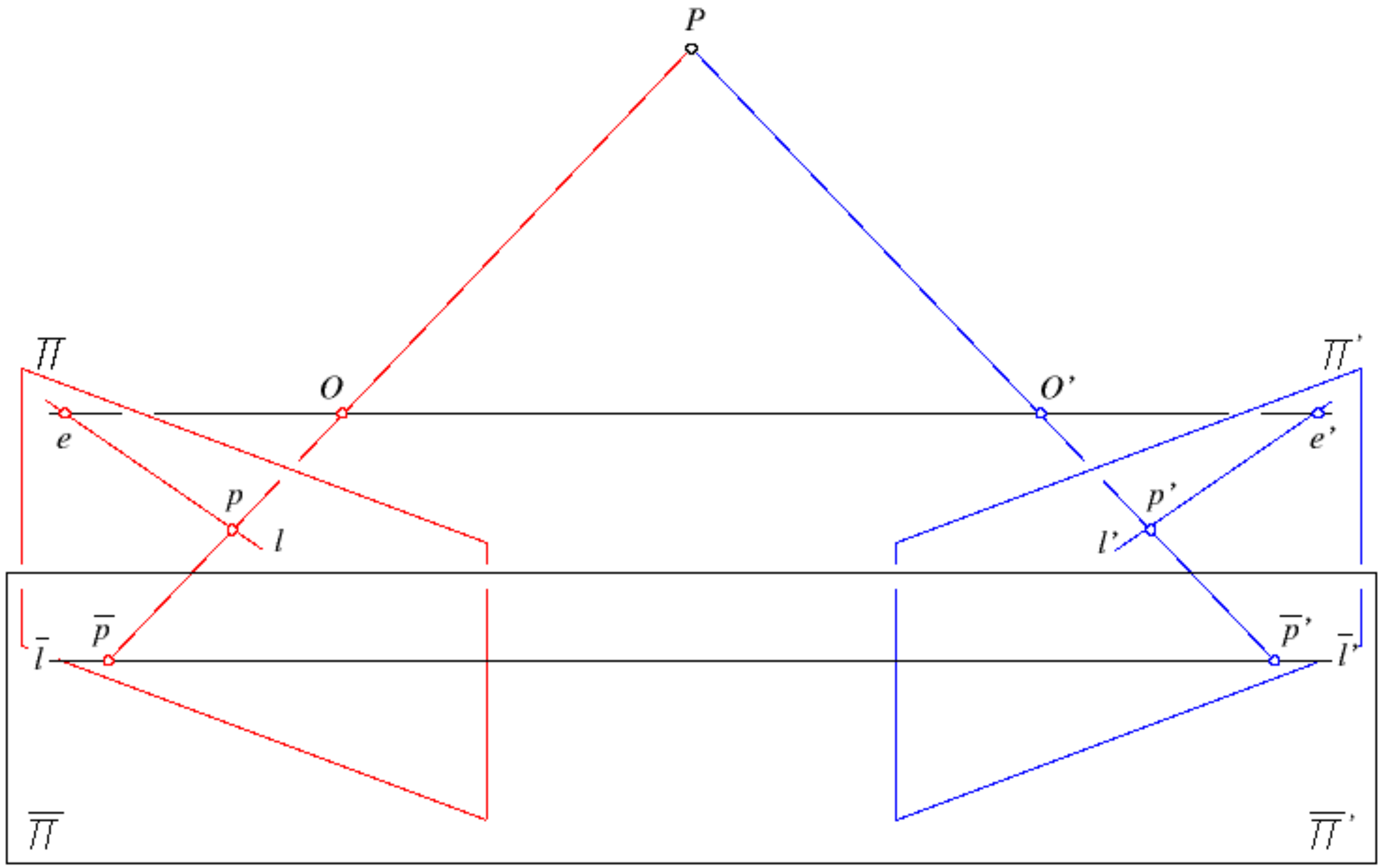
Lenticular lenses send different images directly to each eye, without the need for glasses

Stereo Displays

VR headsets send L/R images directly to each eye

[Google Cardboard]

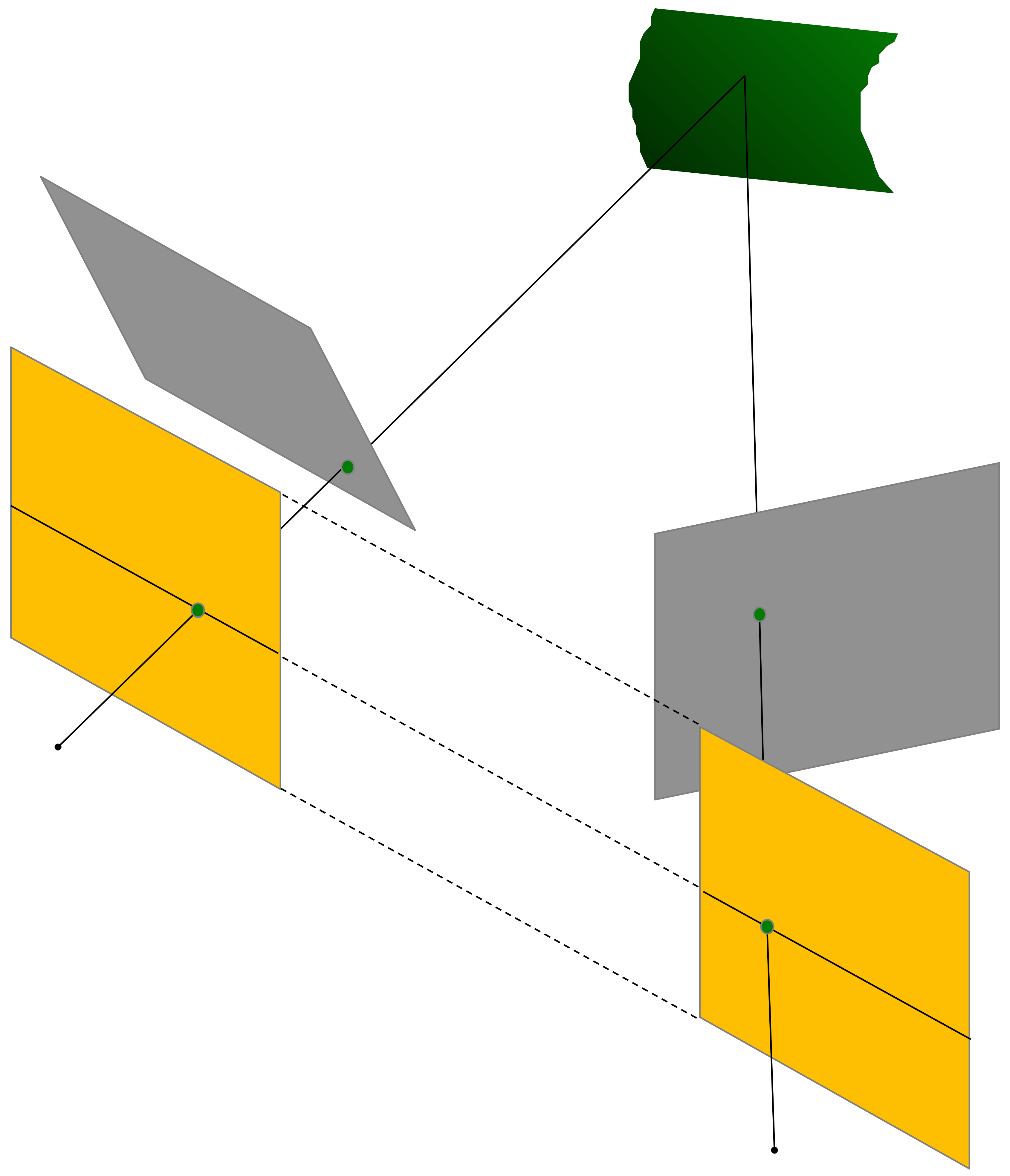
Rectified Stereo Pair



Any two camera views that overlap can be **rectified** so that epipolar lines correspond to scan lines (no special conditions must hold)

Rectified Stereo Pair

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



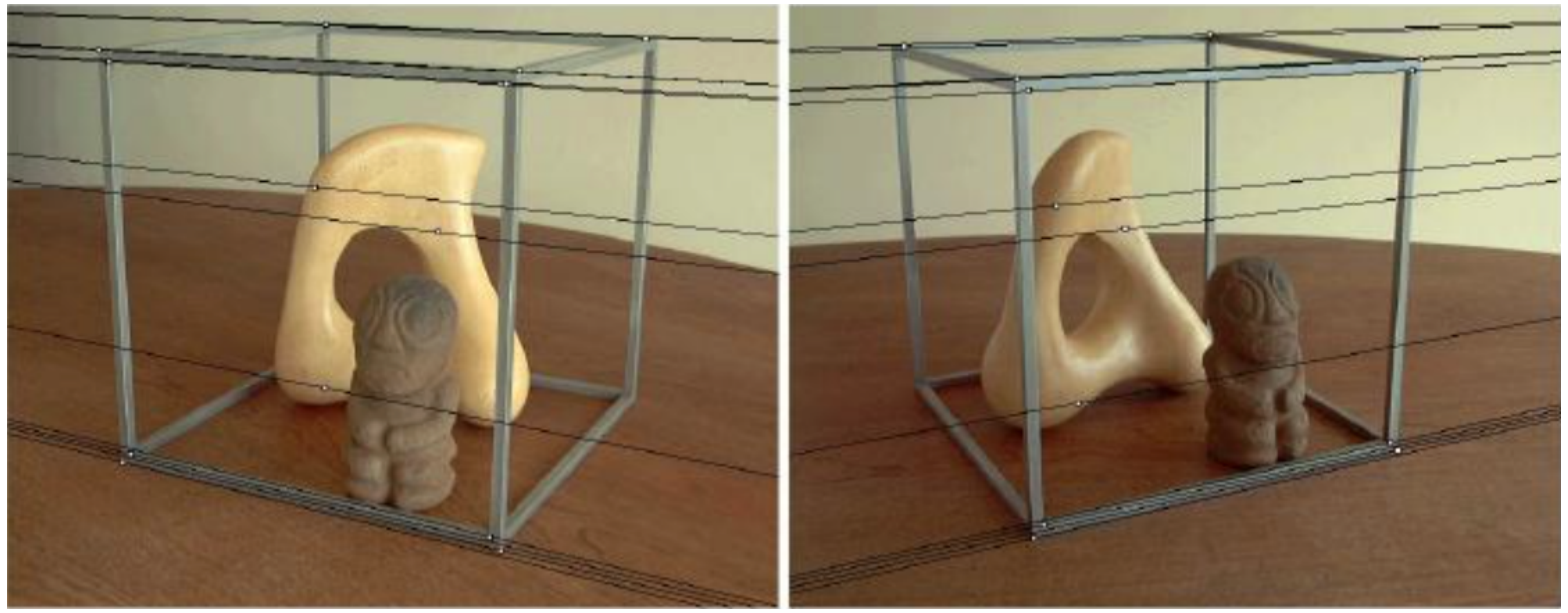
Need two homographies (3x3 transform), one for each input image reprojection

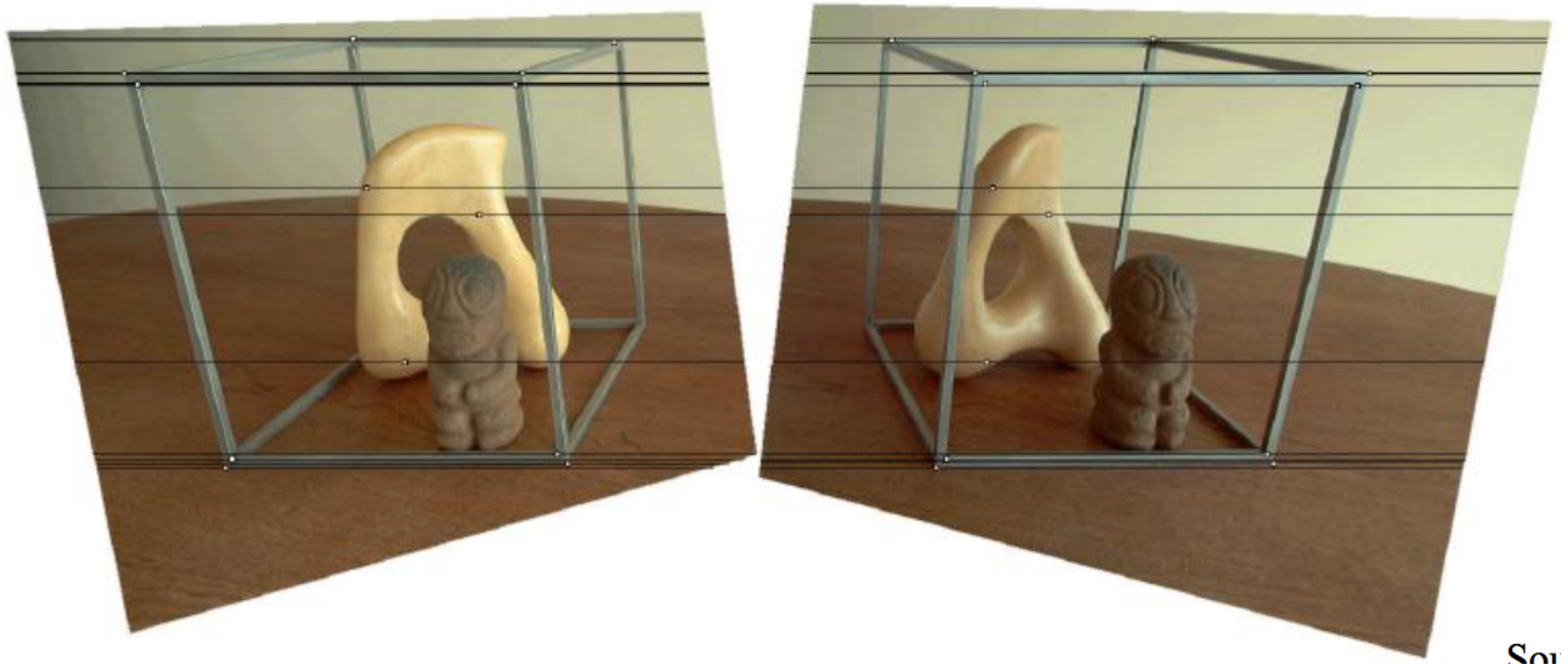
C. Loop and Z. Zhang. Computing Rectifying Homographies for Stereo Vision.Computer Vision and Pattern Recognition, 1999.

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Rectified Stereo Pair: Example

Before Rectification

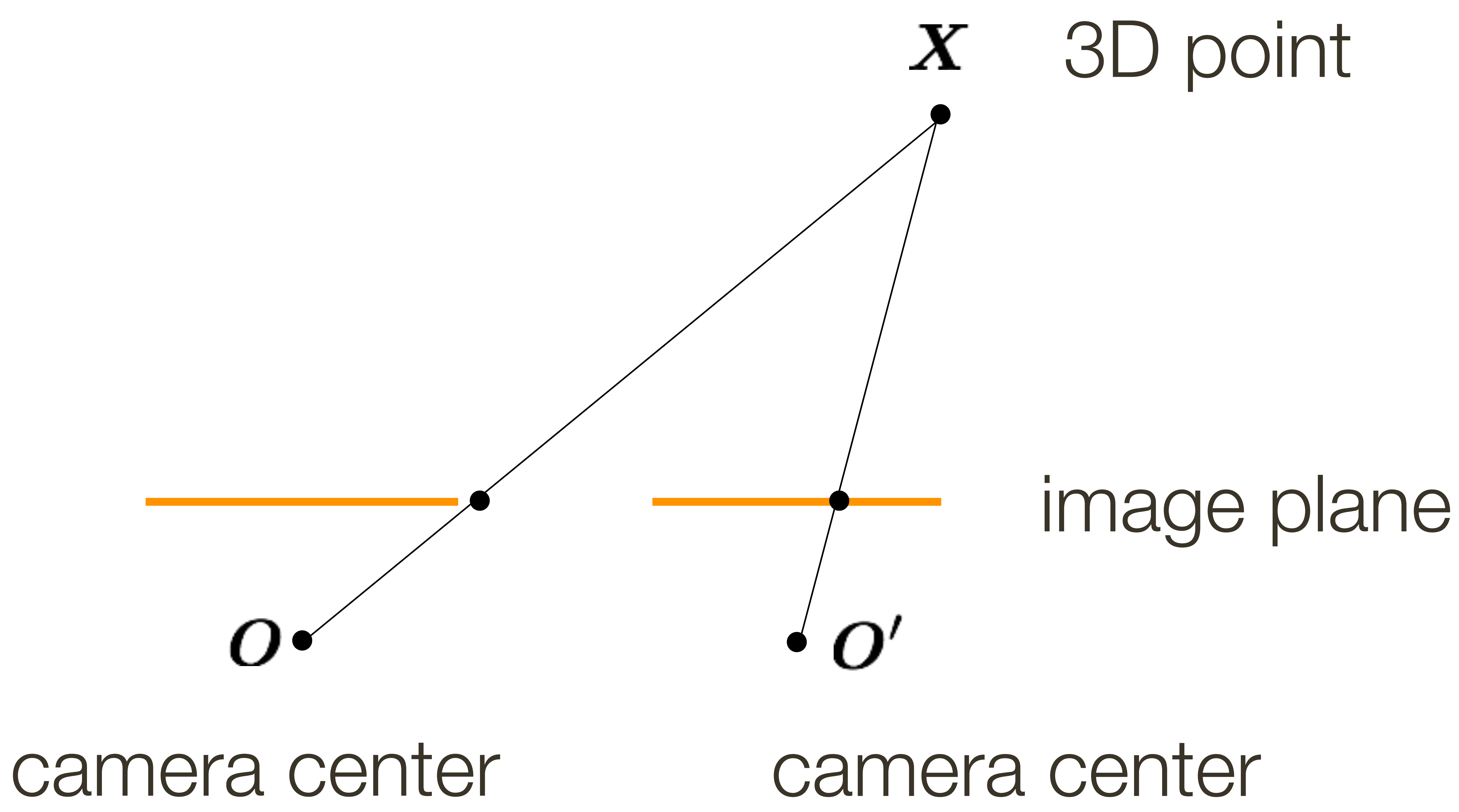




After Rectification

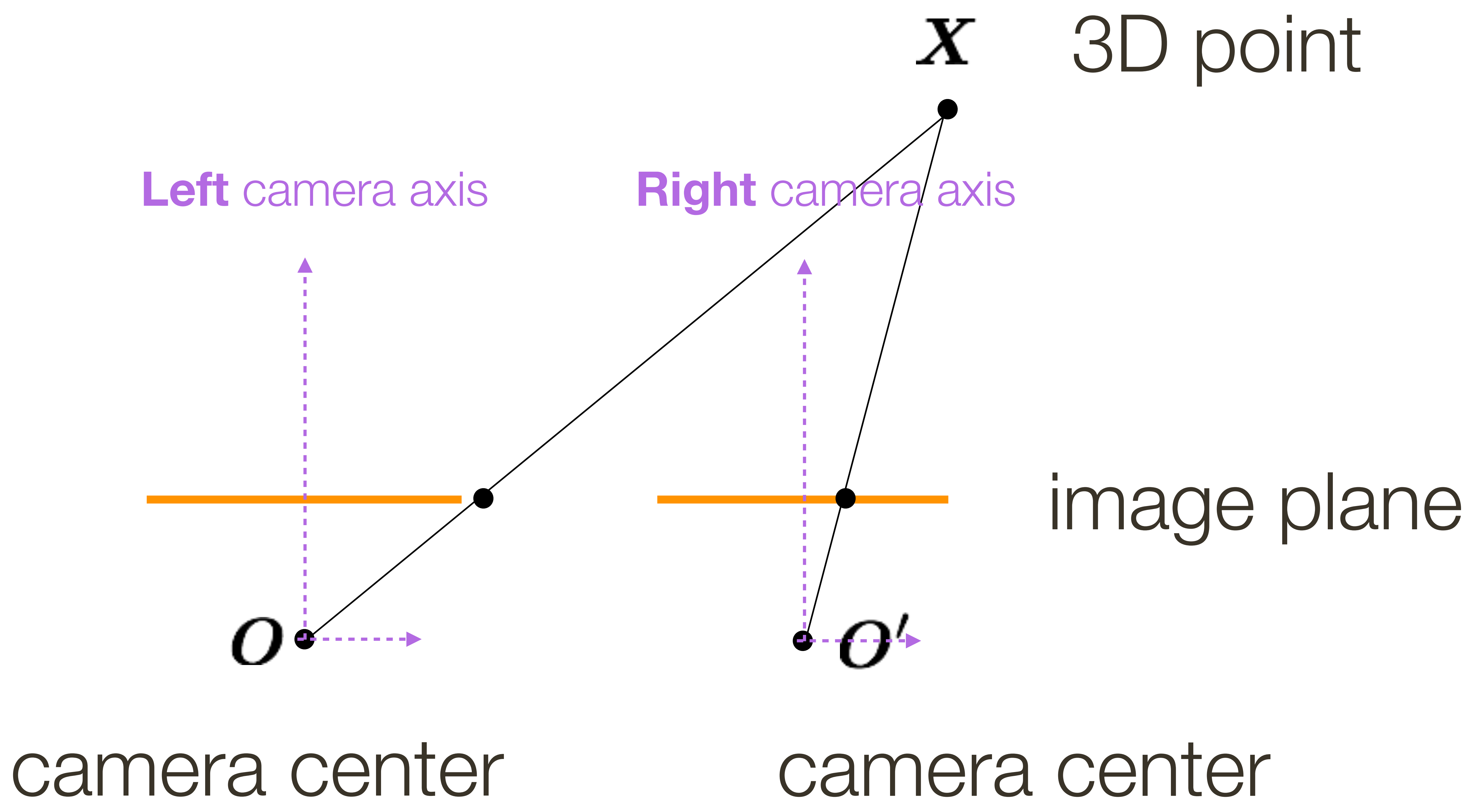
Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Rectified Stereo Pair: Depth Estimate



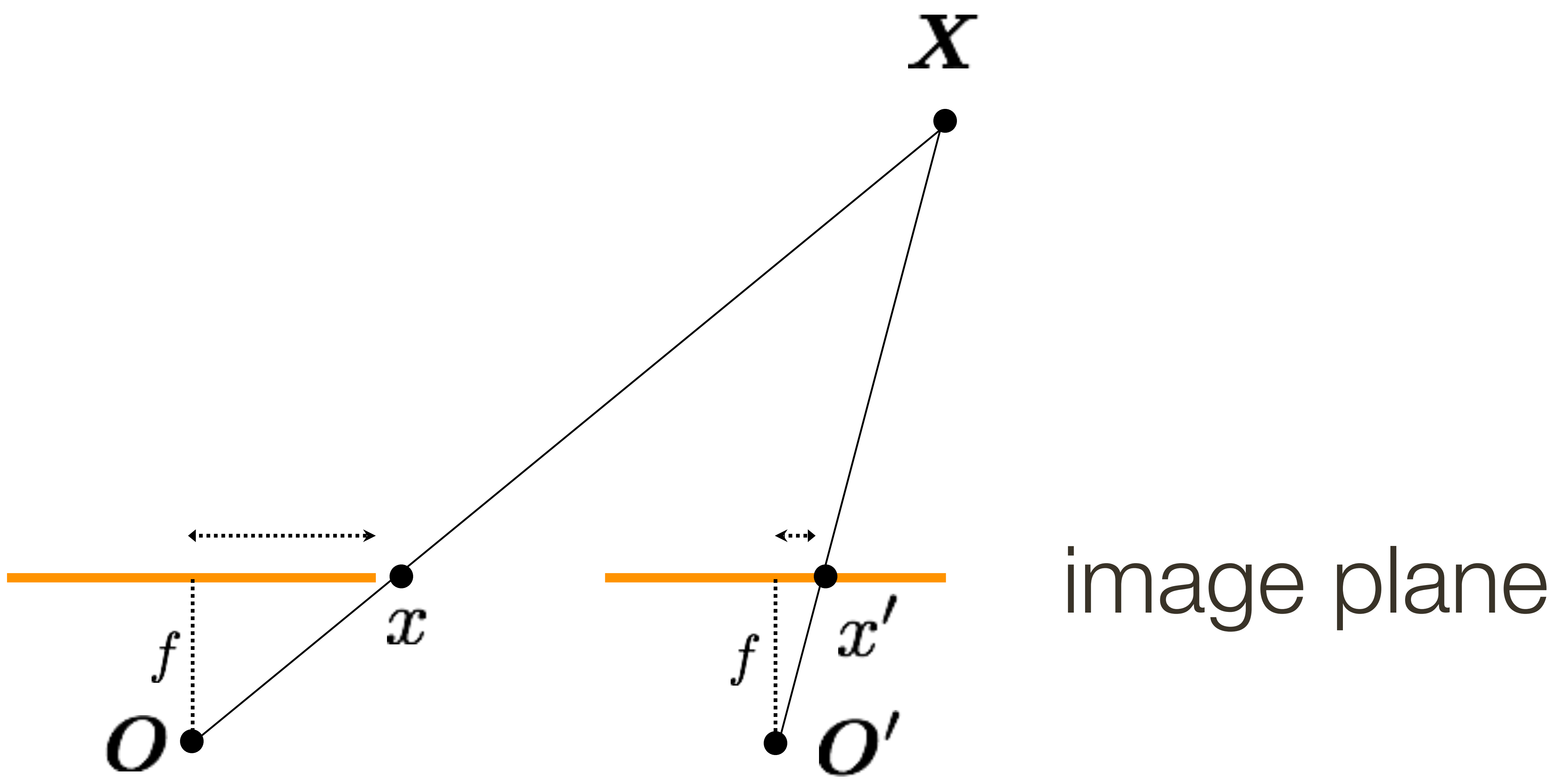
Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Rectified Stereo Pair: Depth Estimate



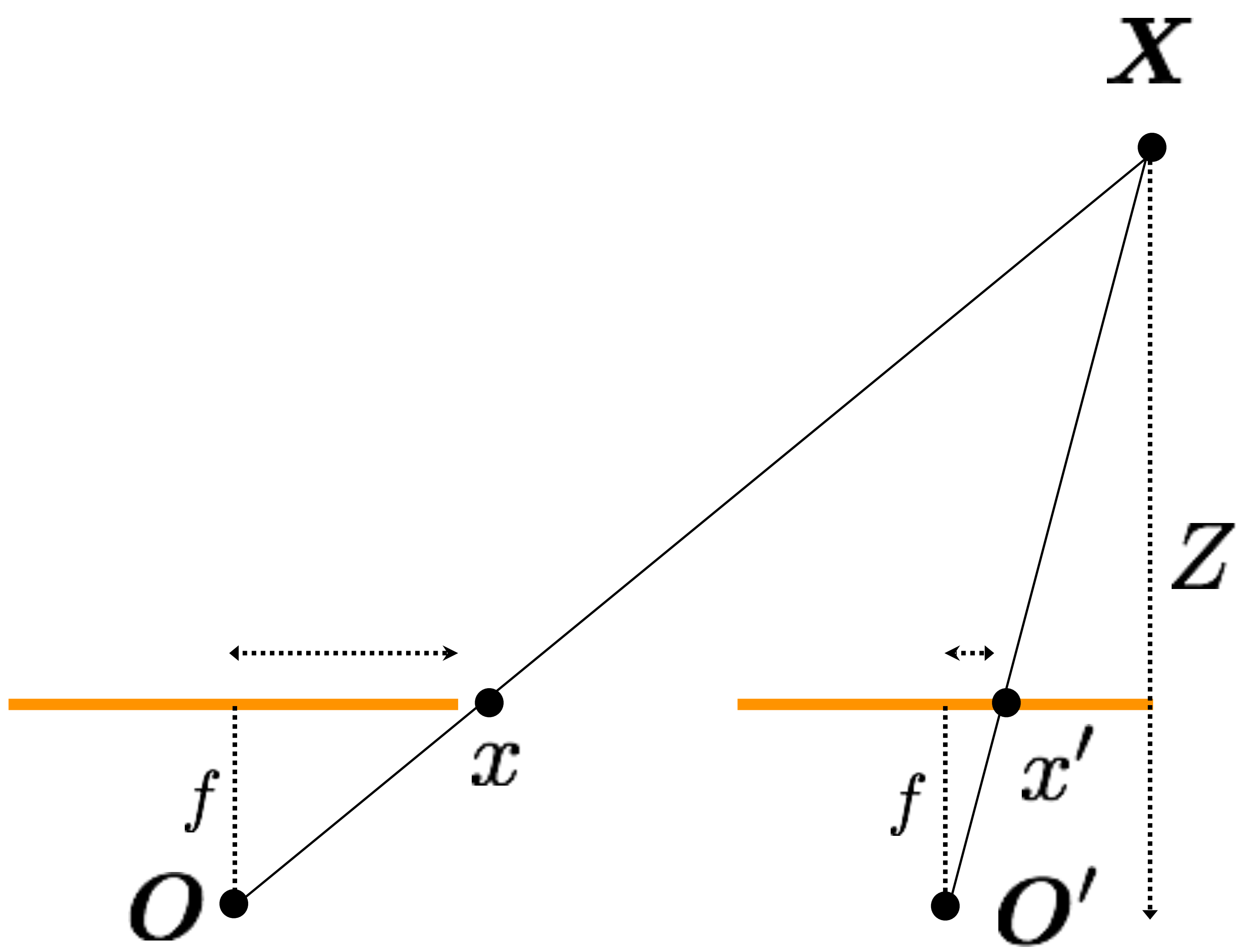
Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Rectified Stereo Pair: Depth Estimate



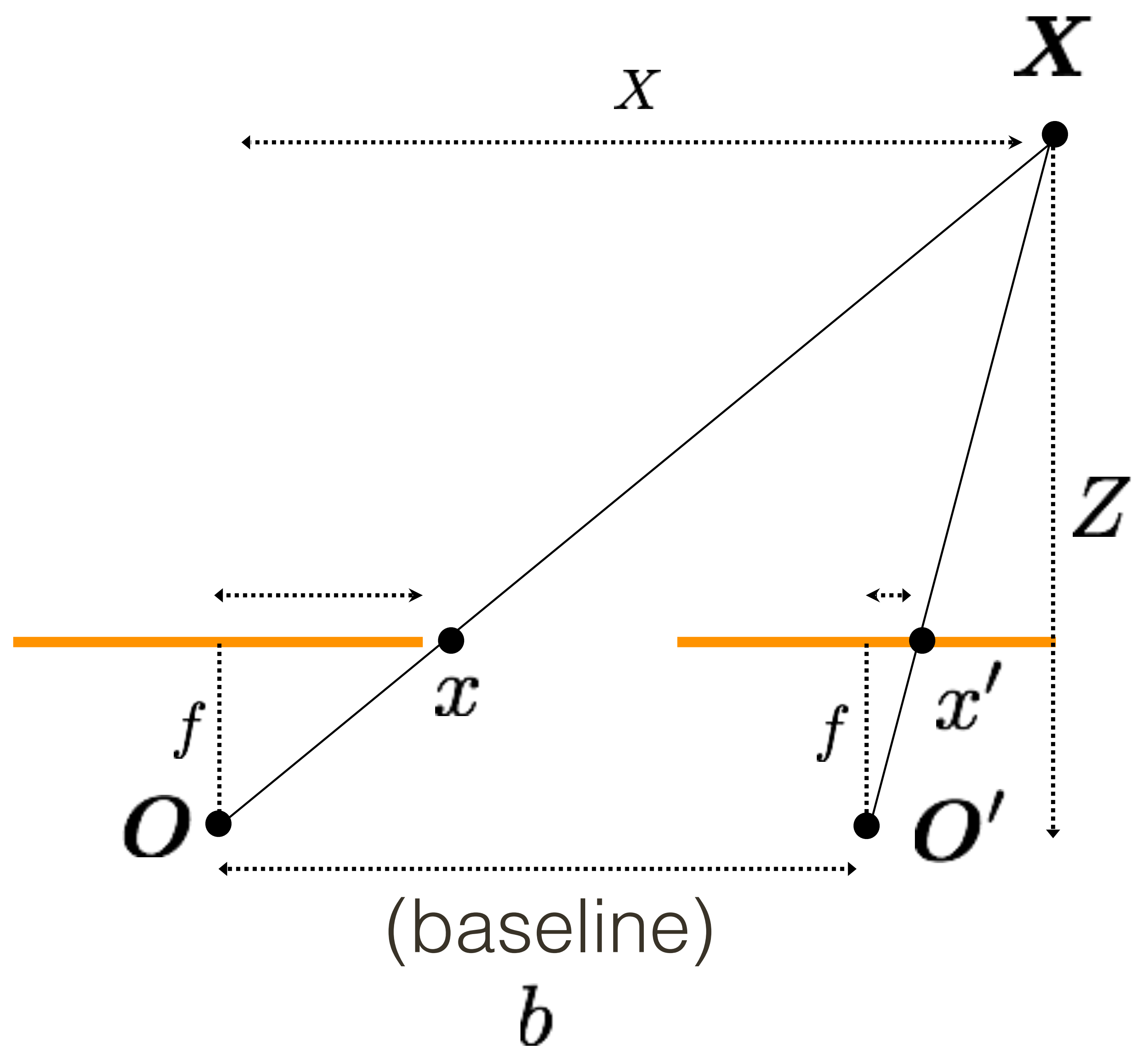
Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Rectified Stereo Pair: Depth Estimate



Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

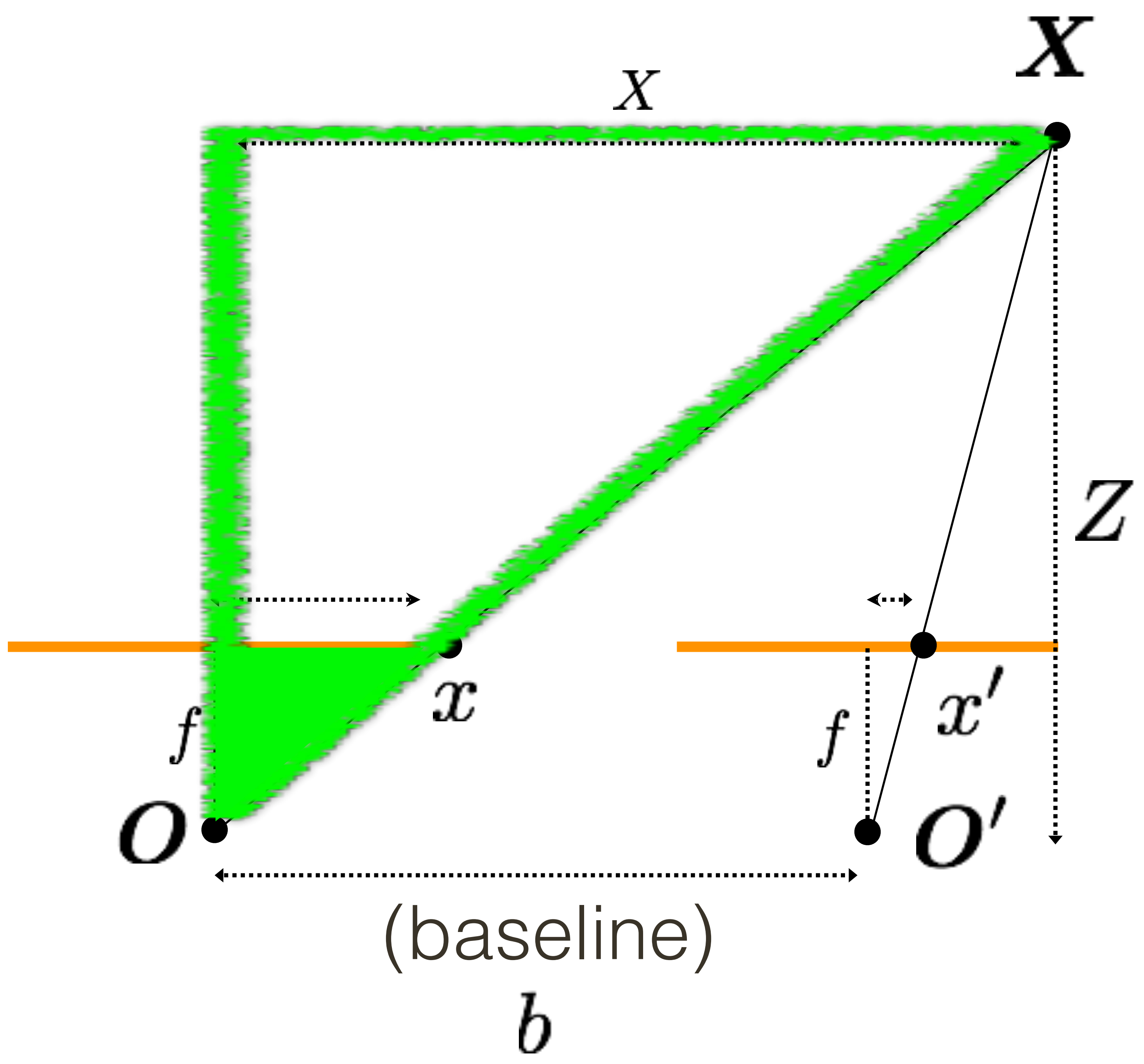
Rectified Stereo Pair: Depth Estimate



Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Rectified Stereo Pair: Depth Estimate

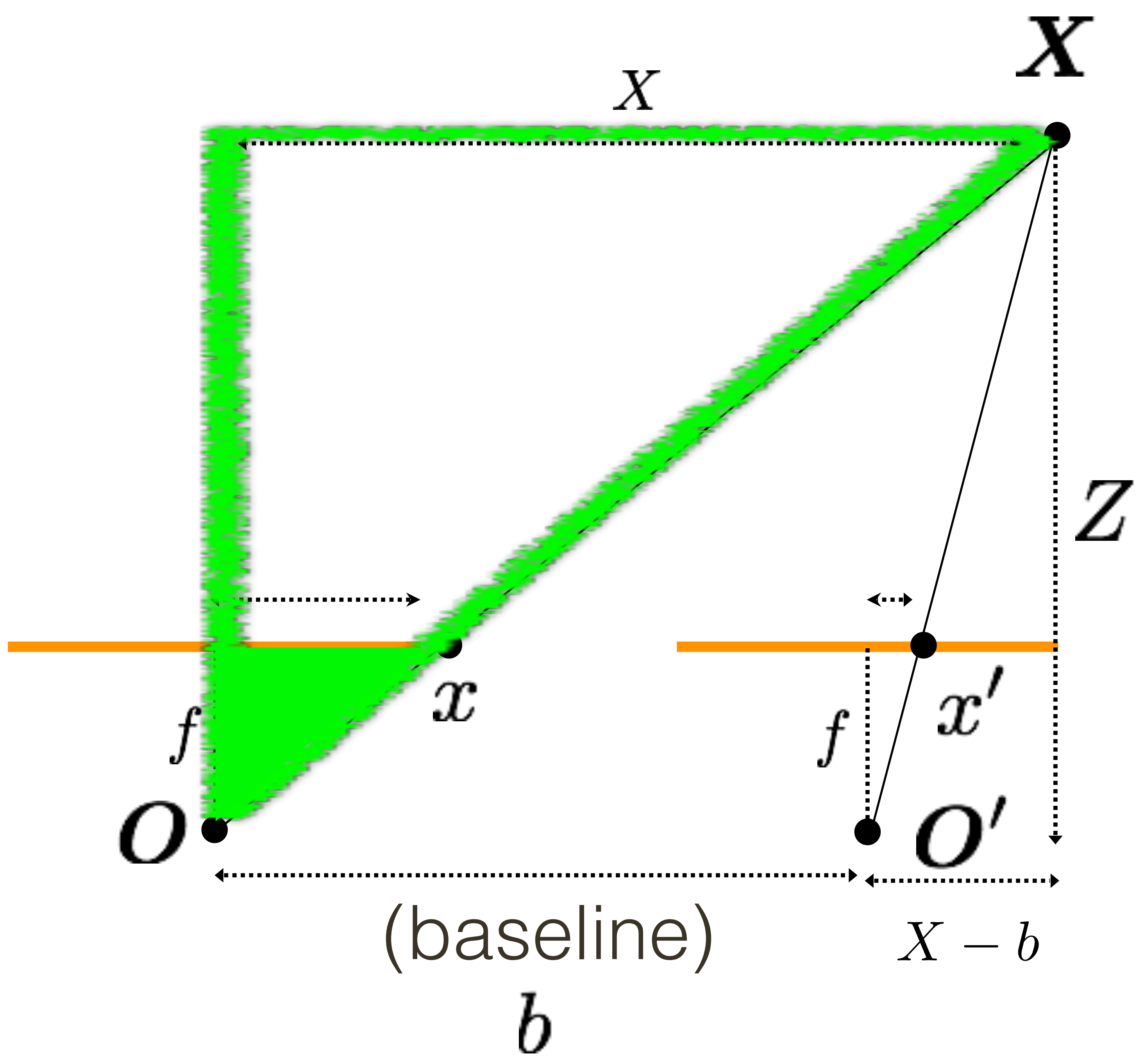
$$\frac{X}{Z} = \frac{x}{f}$$



Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

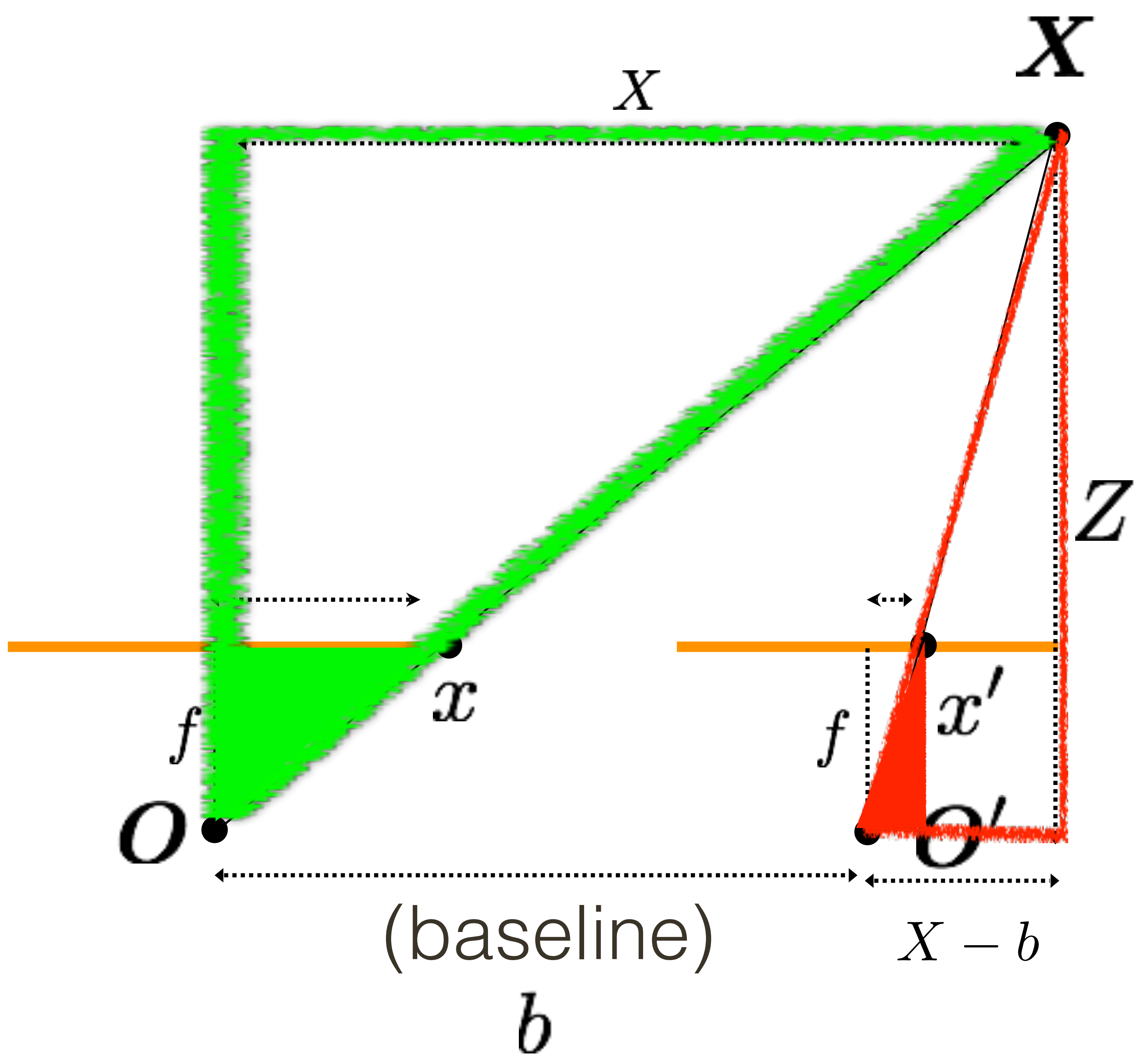
Rectified Stereo Pair: Depth Estimate

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



Rectified Stereo Pair: Depth Estimate

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

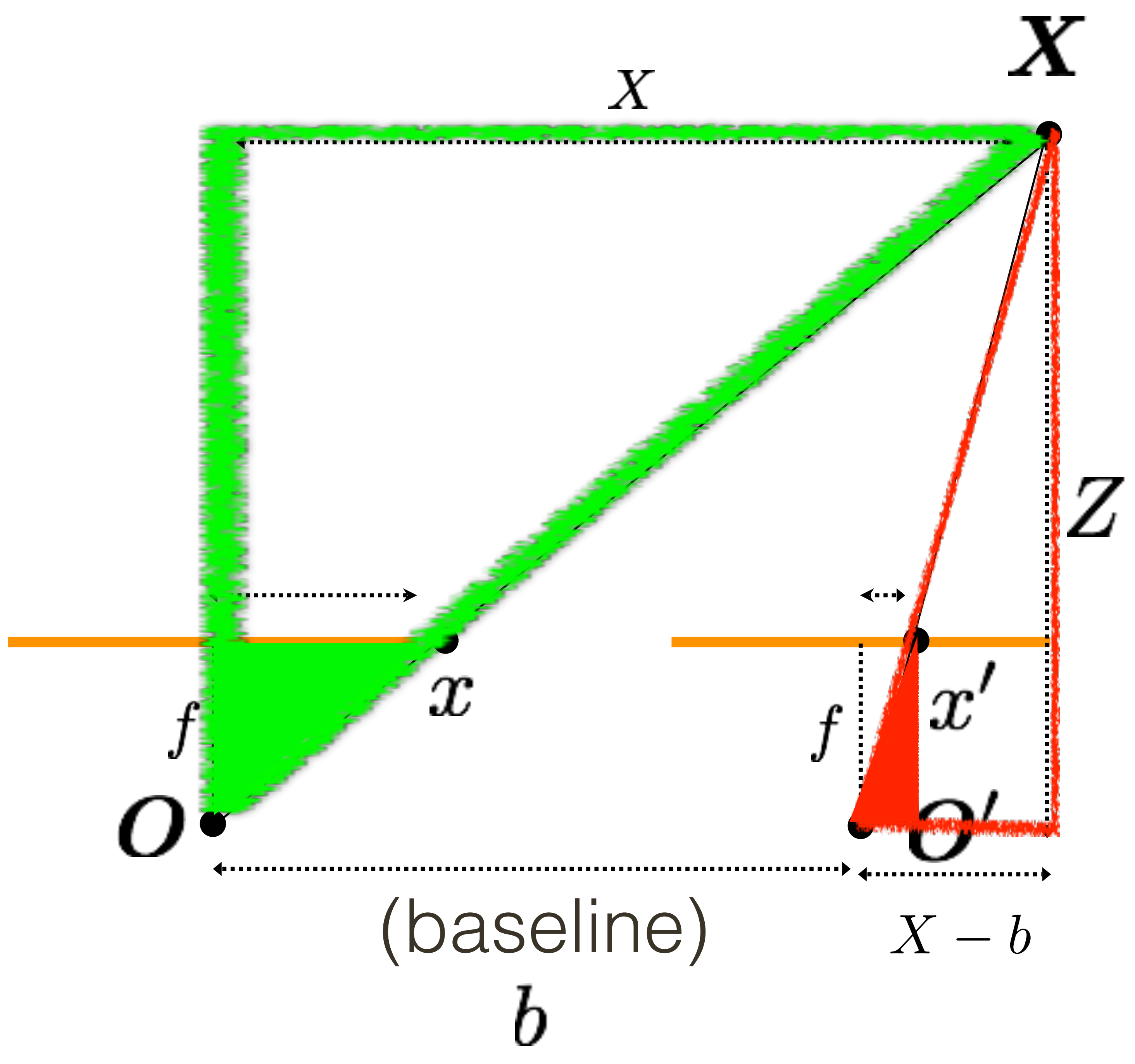


$$\frac{X-b}{Z} = \frac{x'}{f}$$

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Rectified Stereo Pair: Depth Estimate

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



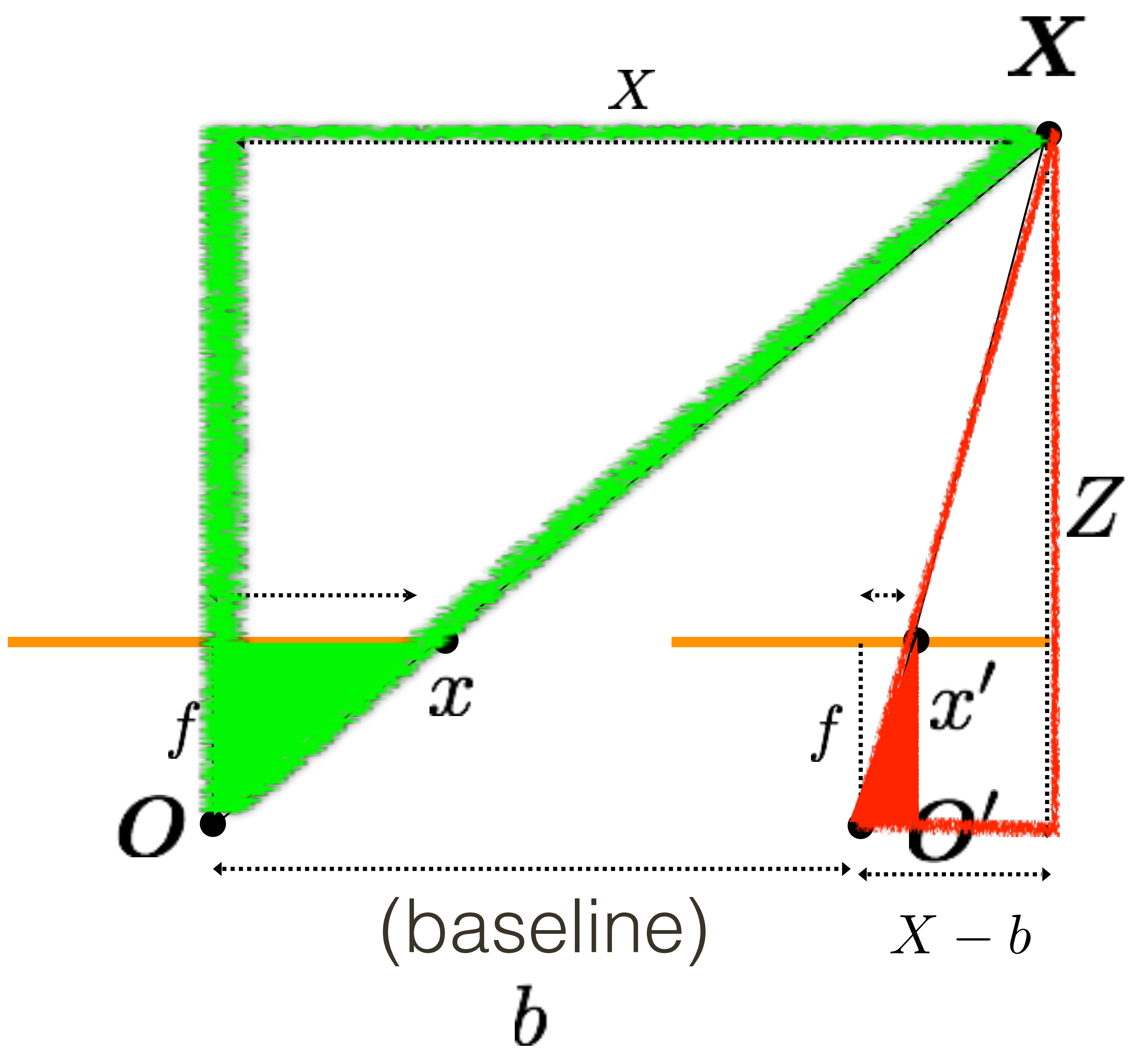
$$\frac{X - b}{Z} = \frac{x'}{f}$$

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

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Rectified Stereo Pair: Depth Estimate

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



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

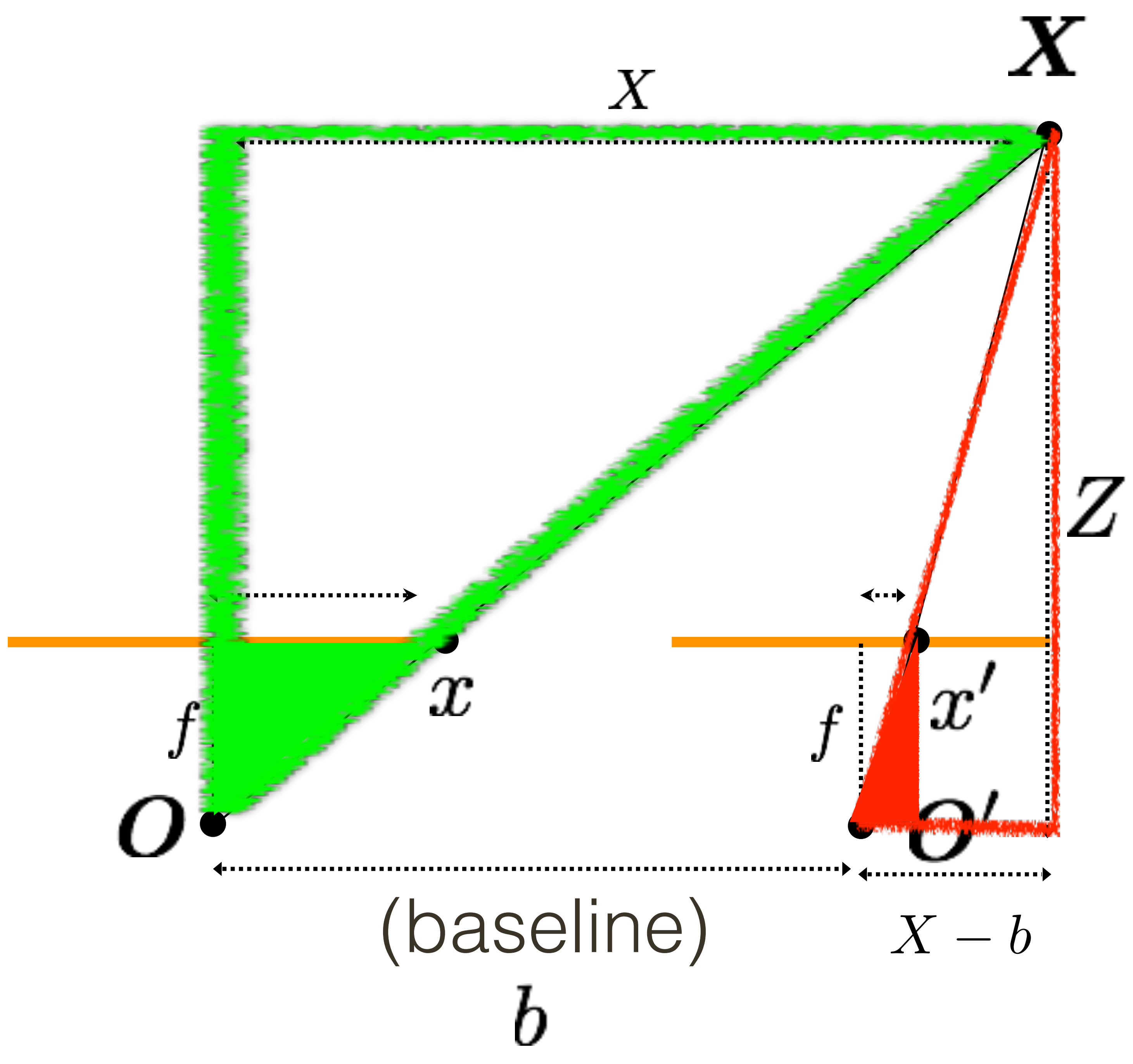
$$\frac{X}{Z} - \frac{b}{Z} = \frac{x'}{f}$$

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

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Rectified Stereo Pair: Depth Estimate

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



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

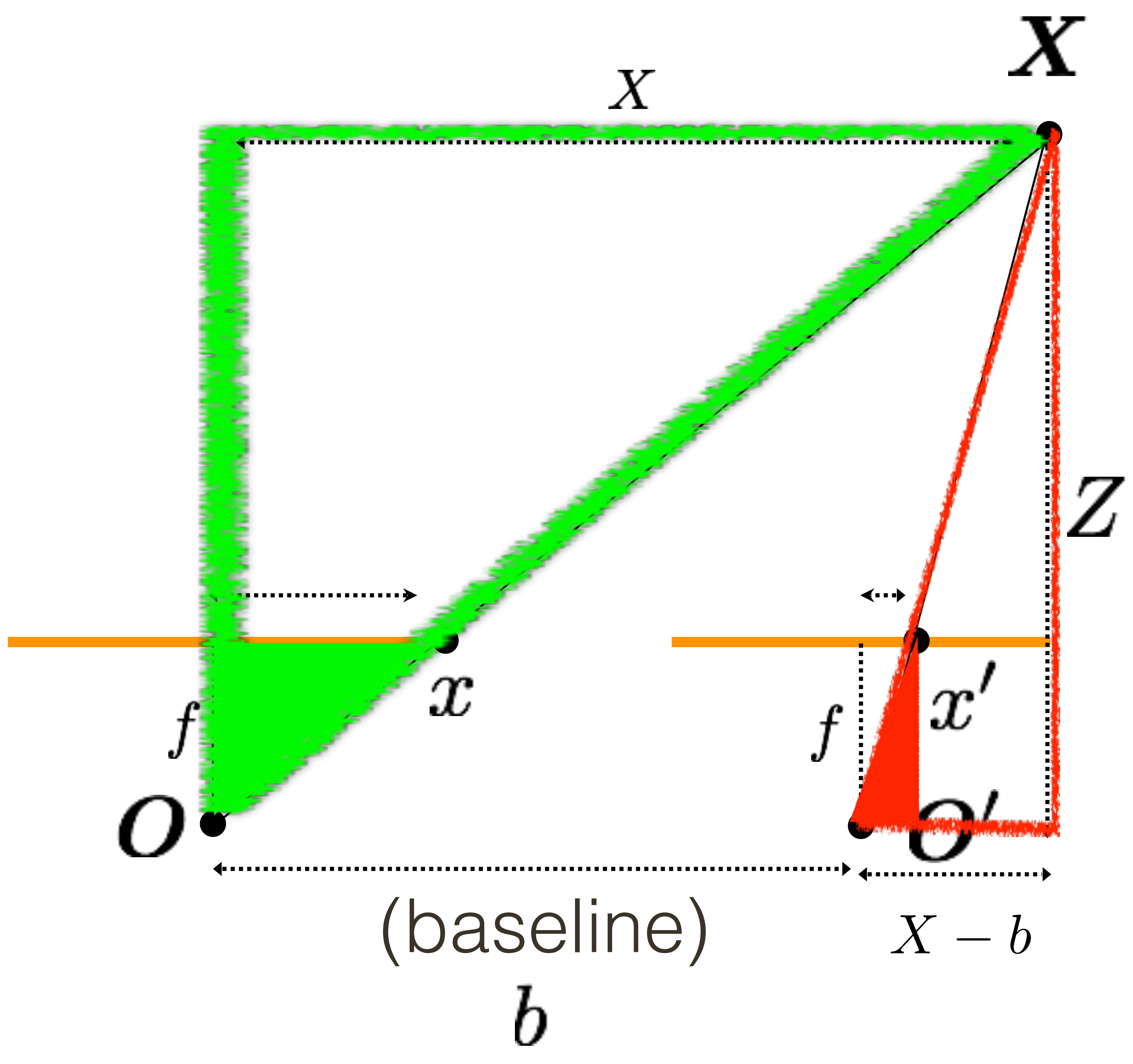
$$\frac{X}{Z} - \frac{b}{Z} = \frac{x'}{f}$$

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

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

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Rectified Stereo Pair: Depth Estimate



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

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

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

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

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

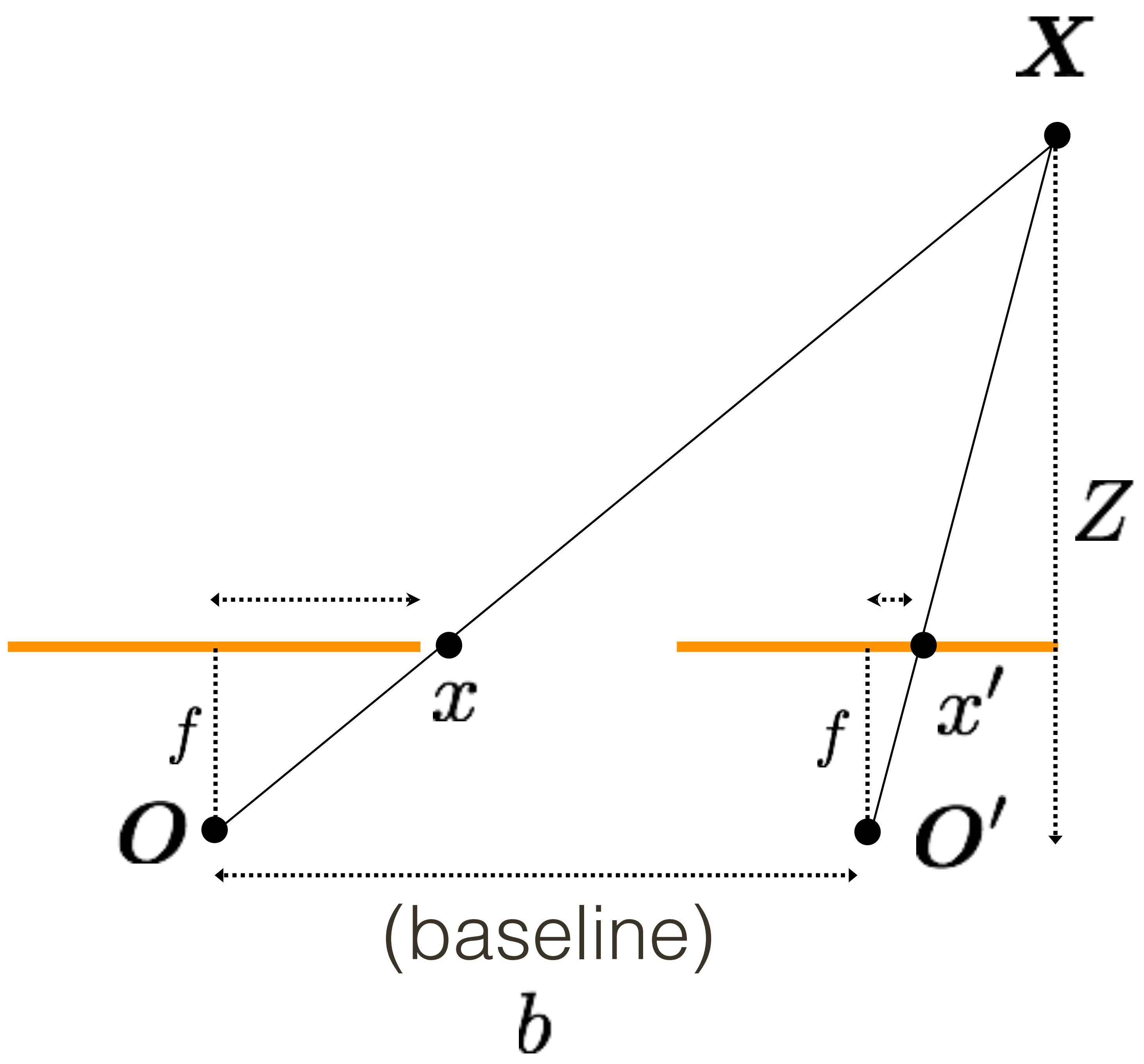
Disparity

(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)

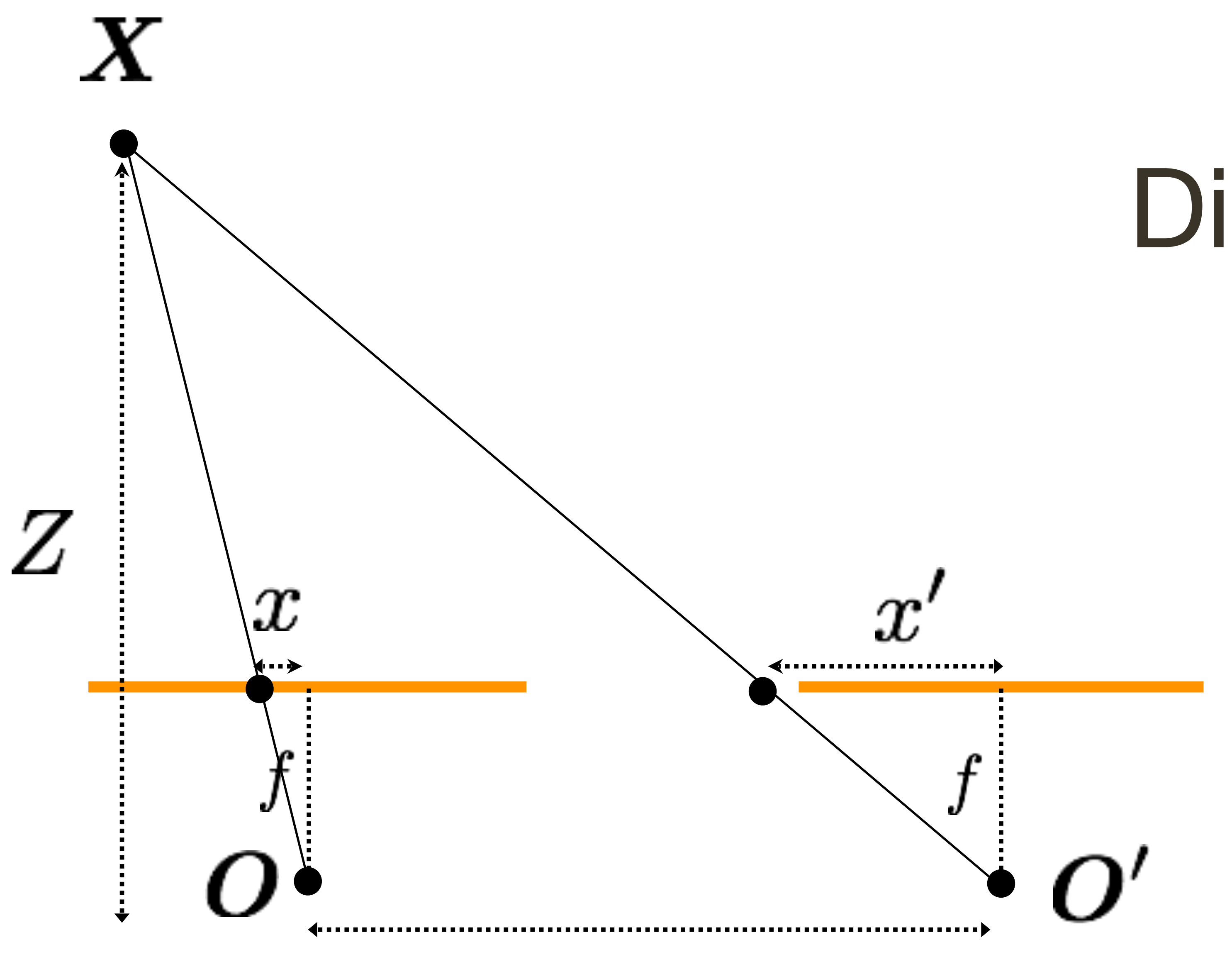
$$d = x - x'$$

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

inversely proportional to depth

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Rectified Stereo Pair: Depth Estimate



(baseline)
b

Disparity will always be **positive**

Disparity

(wrt to camera origin of image plane)

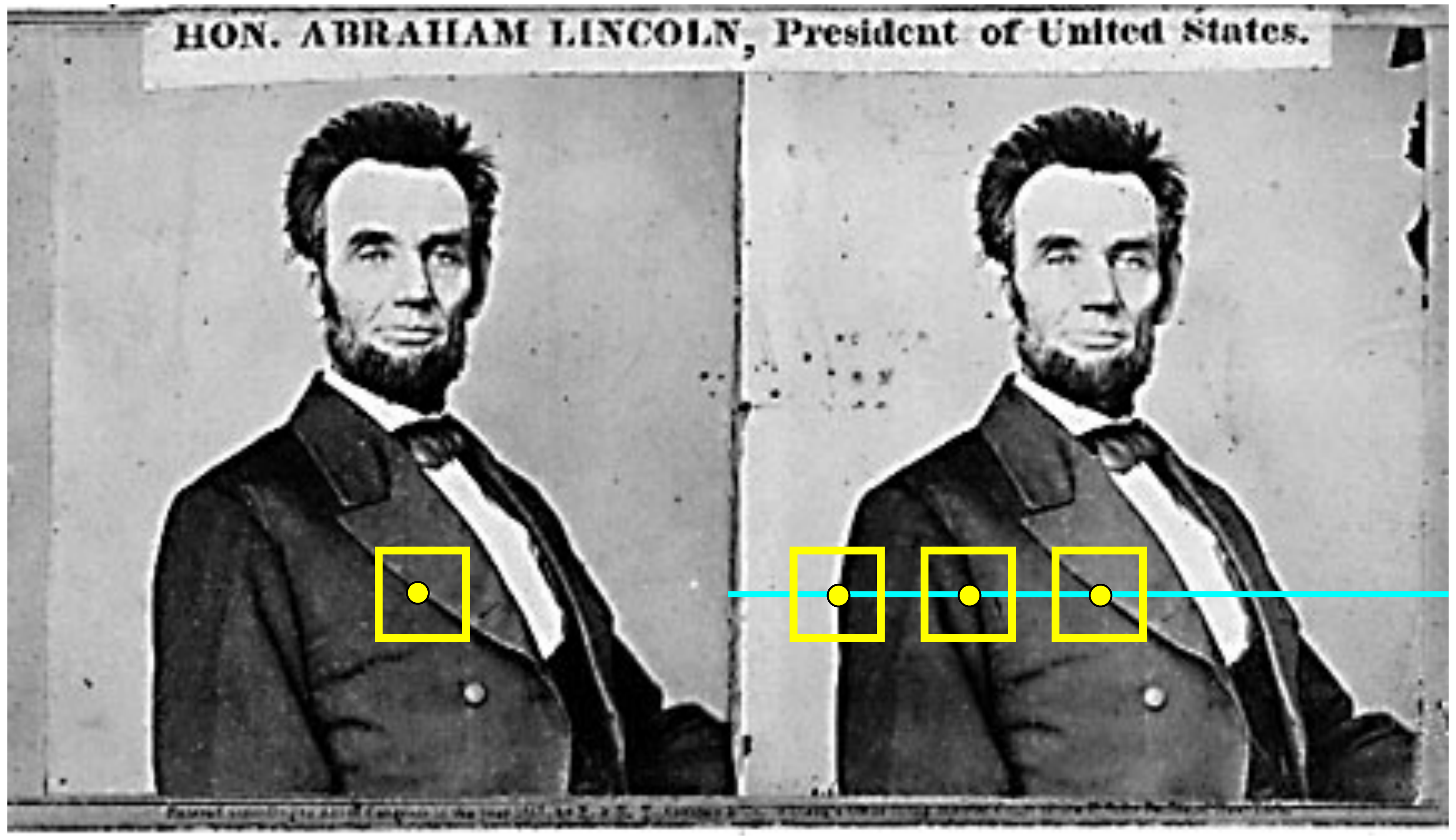
$$d = x - x'$$

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

inversely proportional to depth

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

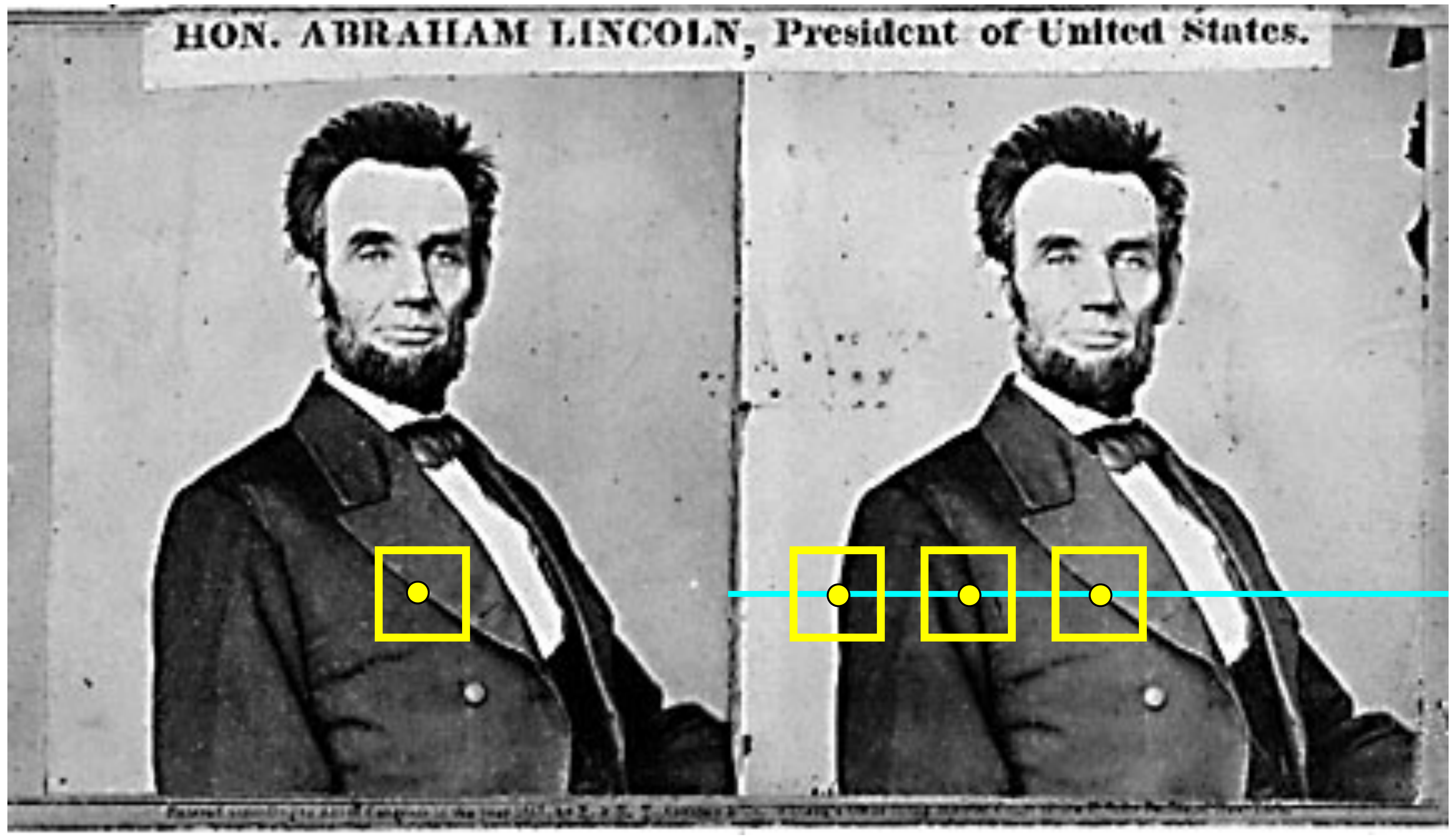
(simple) **Stereo** Algorithm



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

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

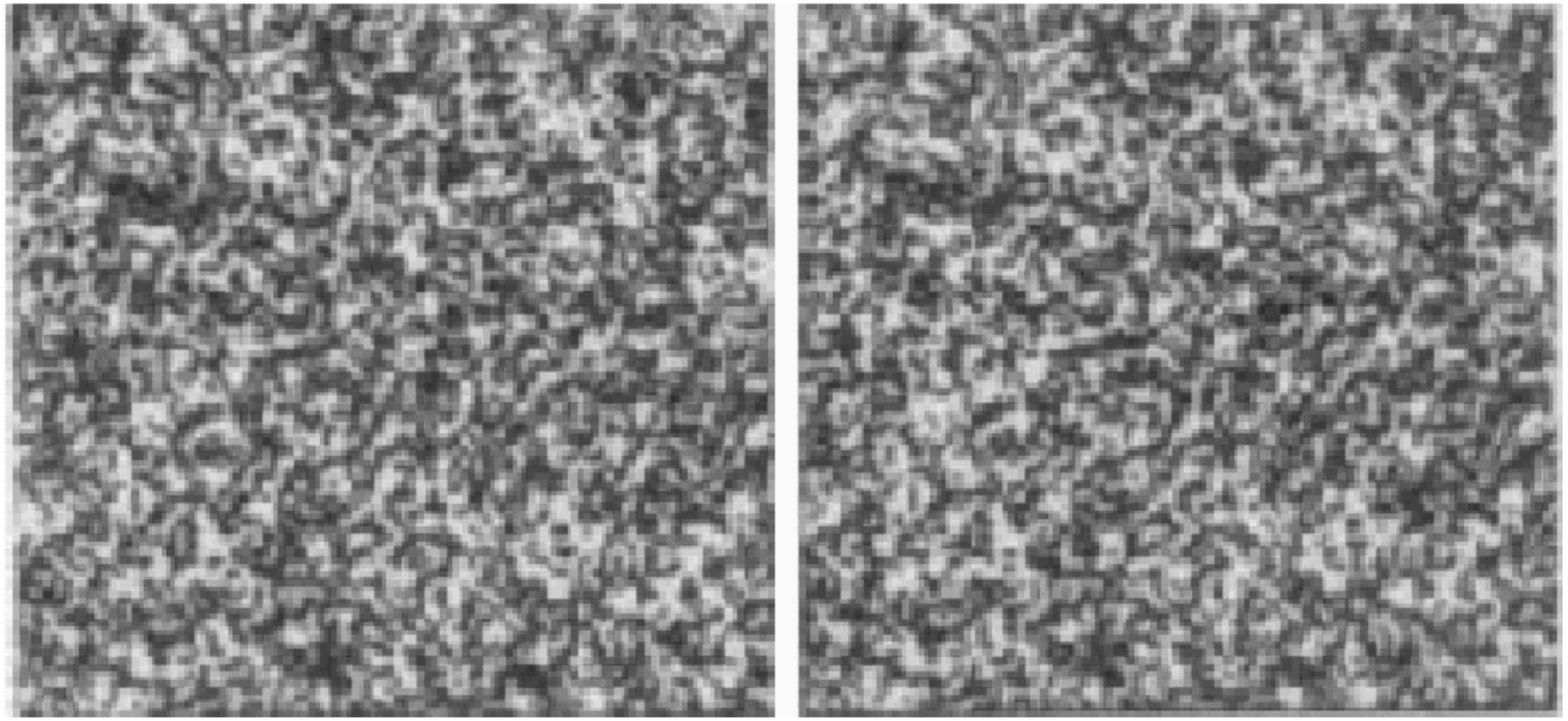
(simple) **Stereo** Algorithm



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

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

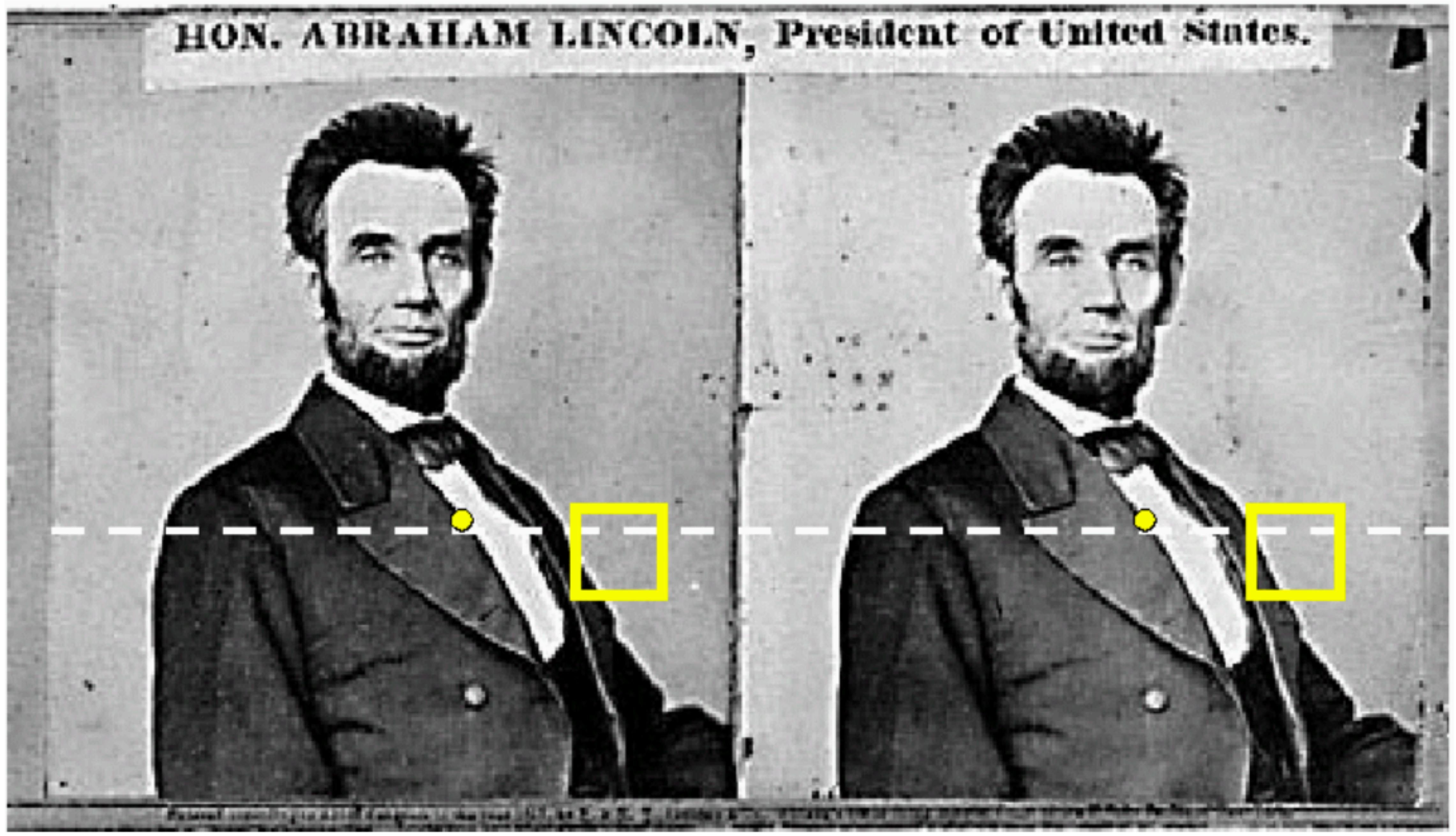
Random Dot Stereograms



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

Block Matching: 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

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

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**

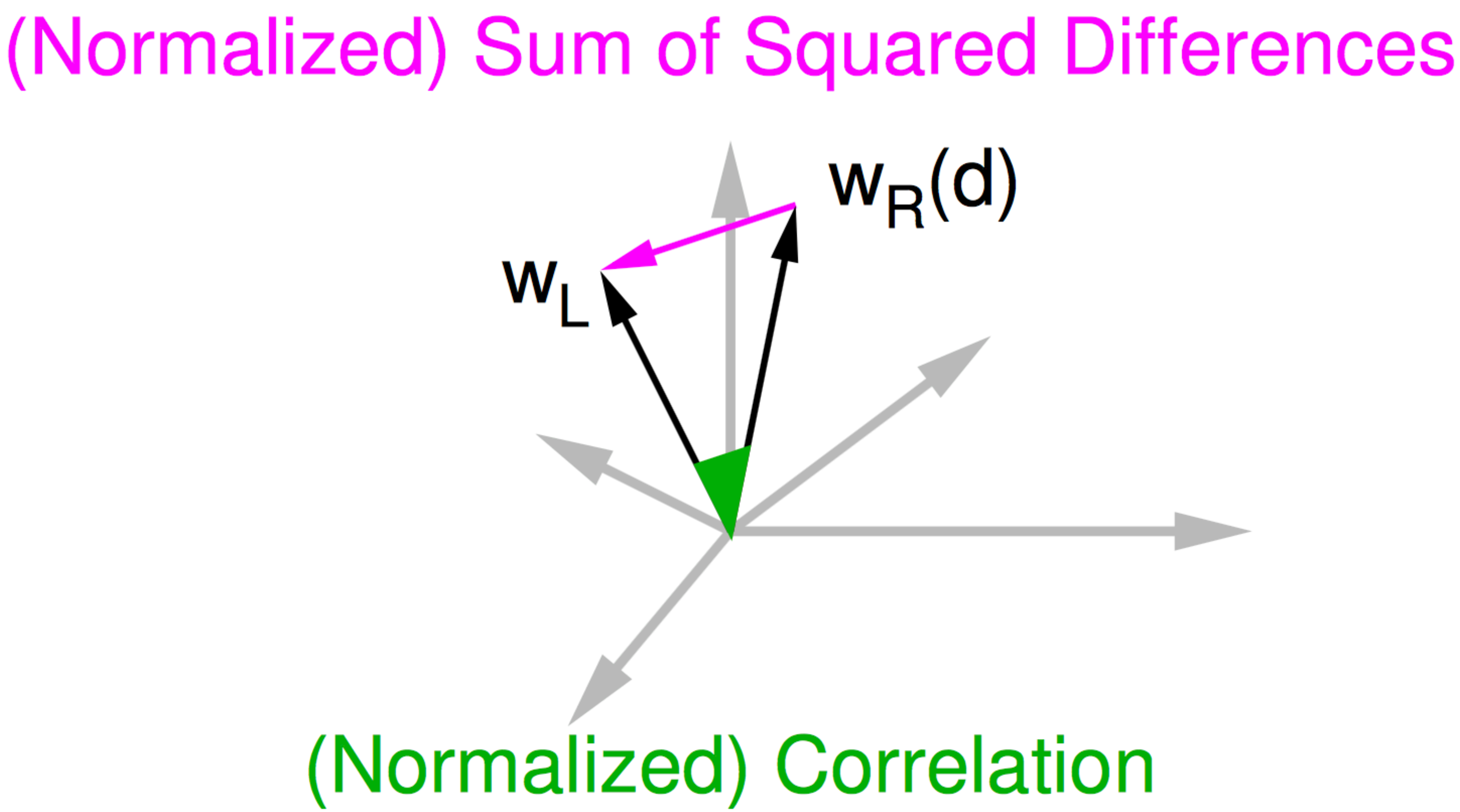


Image **Metrics**

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**

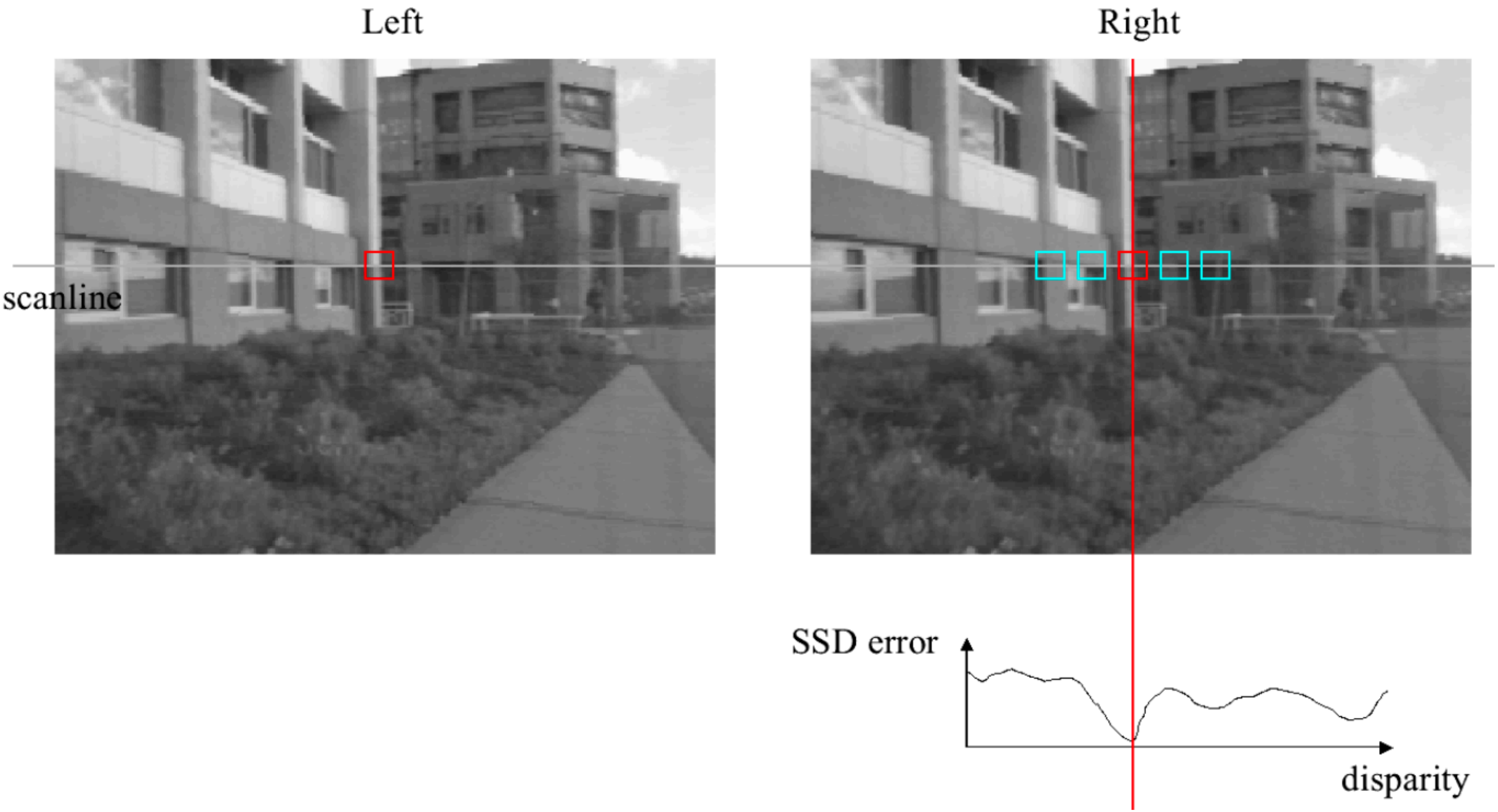
Let d^* be the value of d that minimizes C_{SSD}

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

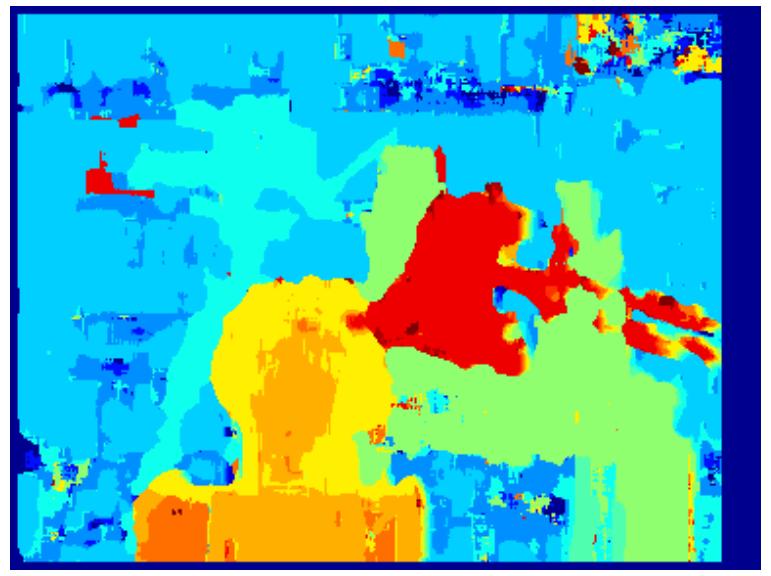
That is,

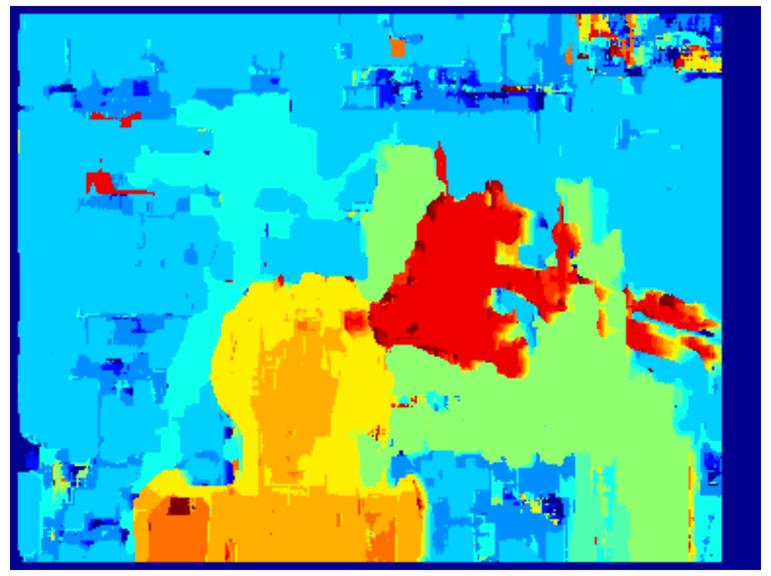
$$d^* = \arg\min_d ||\mathbf{w}_L - \mathbf{w}_R(d)||^2 = \arg\min_d \mathbf{w}_L \cdot \mathbf{w}_R(d)$$

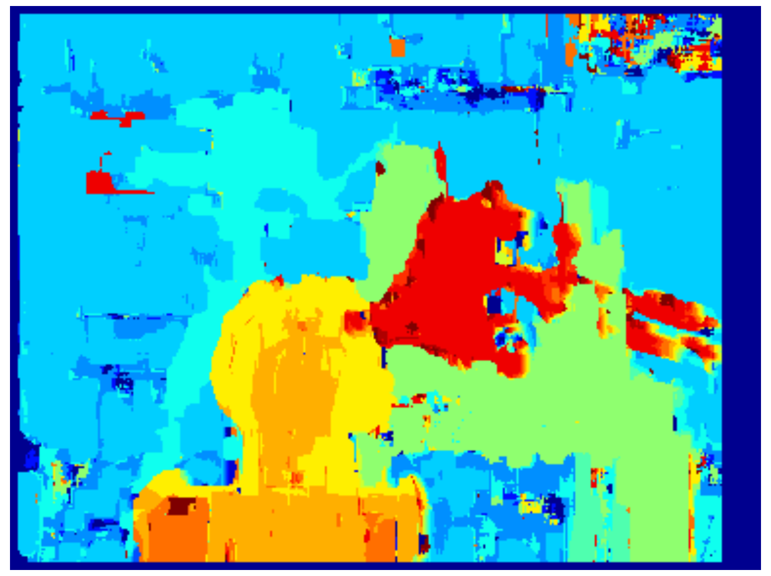
Method: Correlation



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

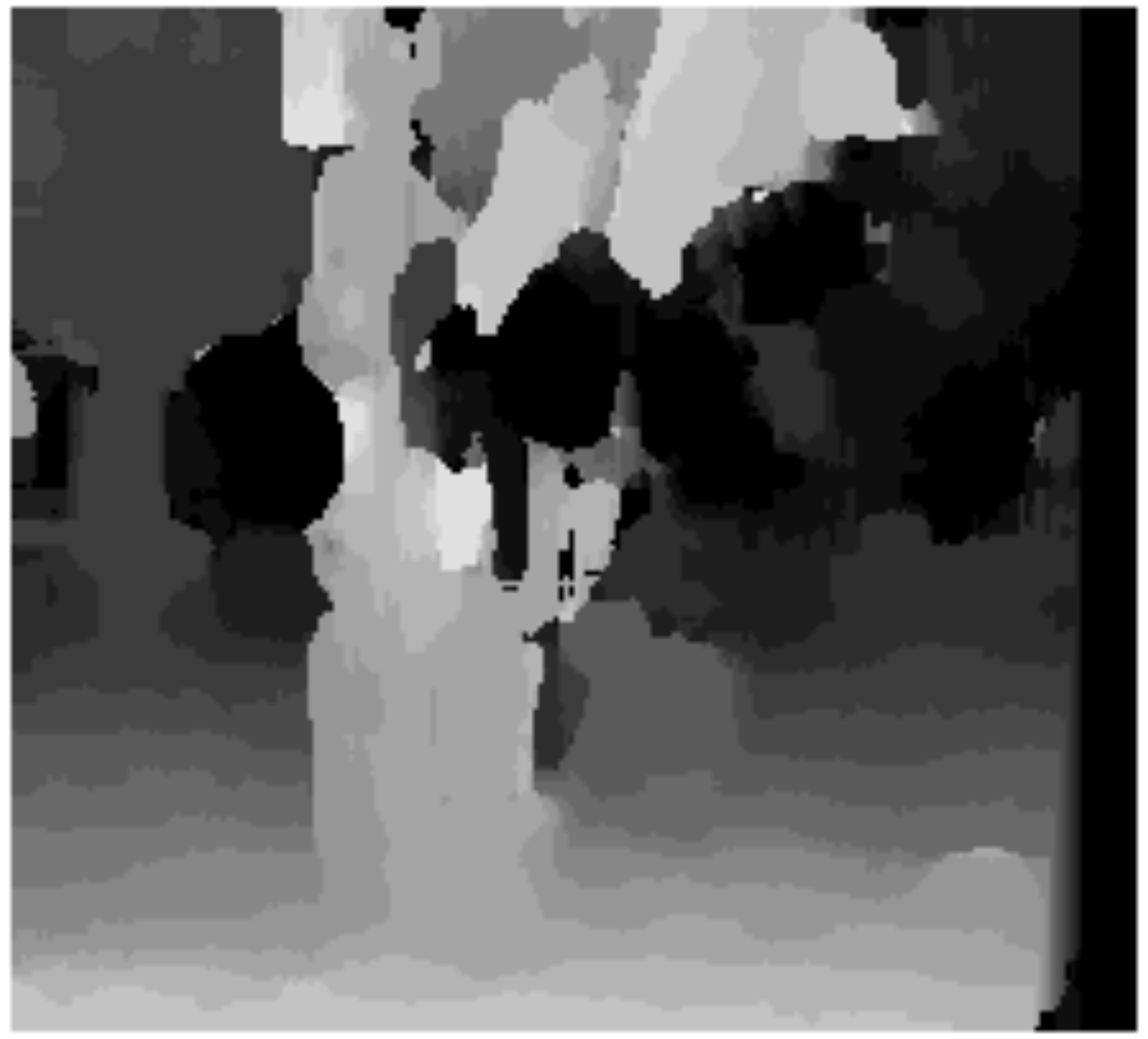

Ground truth

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Effect of **Window Size**

W = 3

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



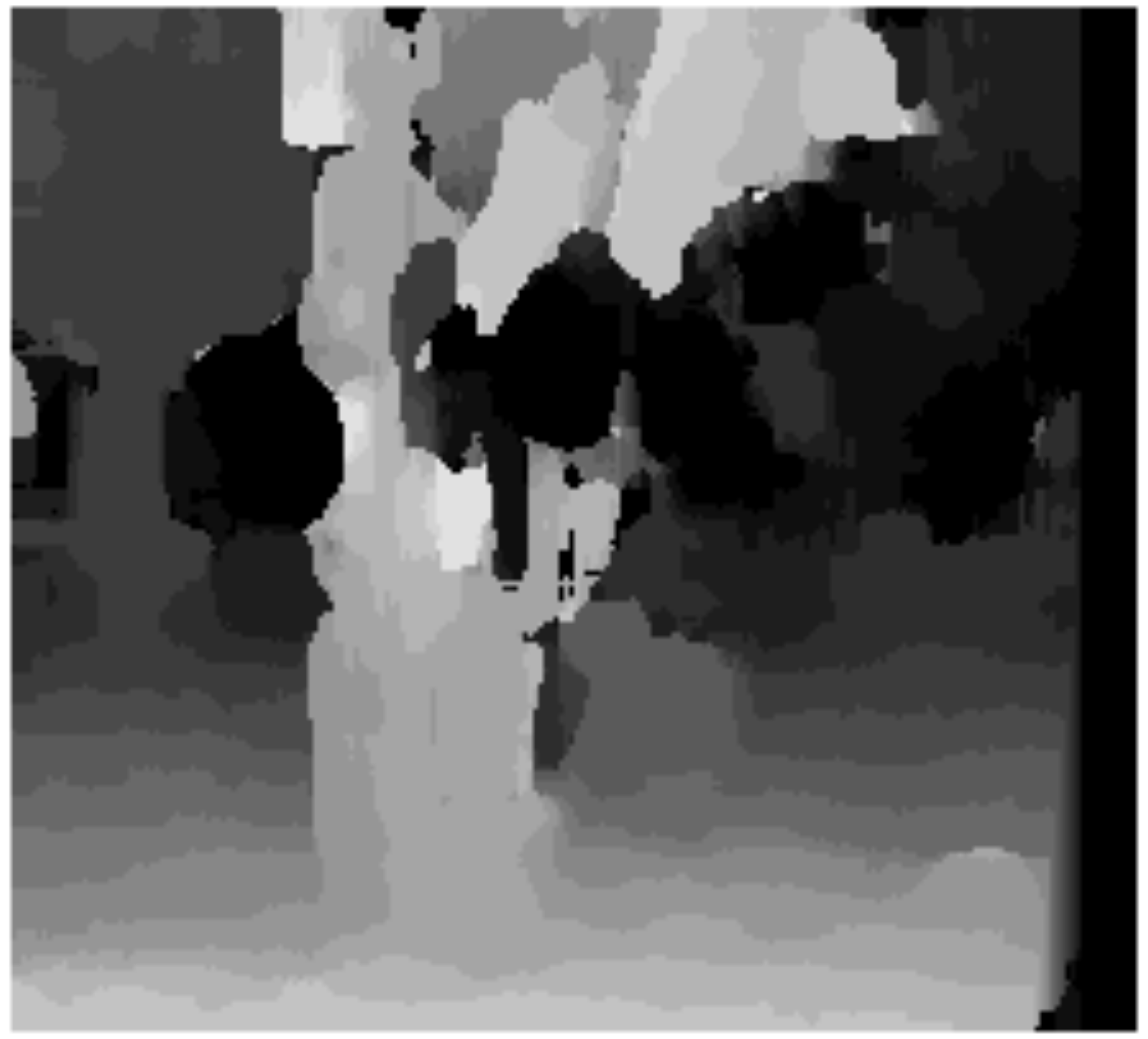
W = 20

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

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Effect of **Window Size**

$W = 3$

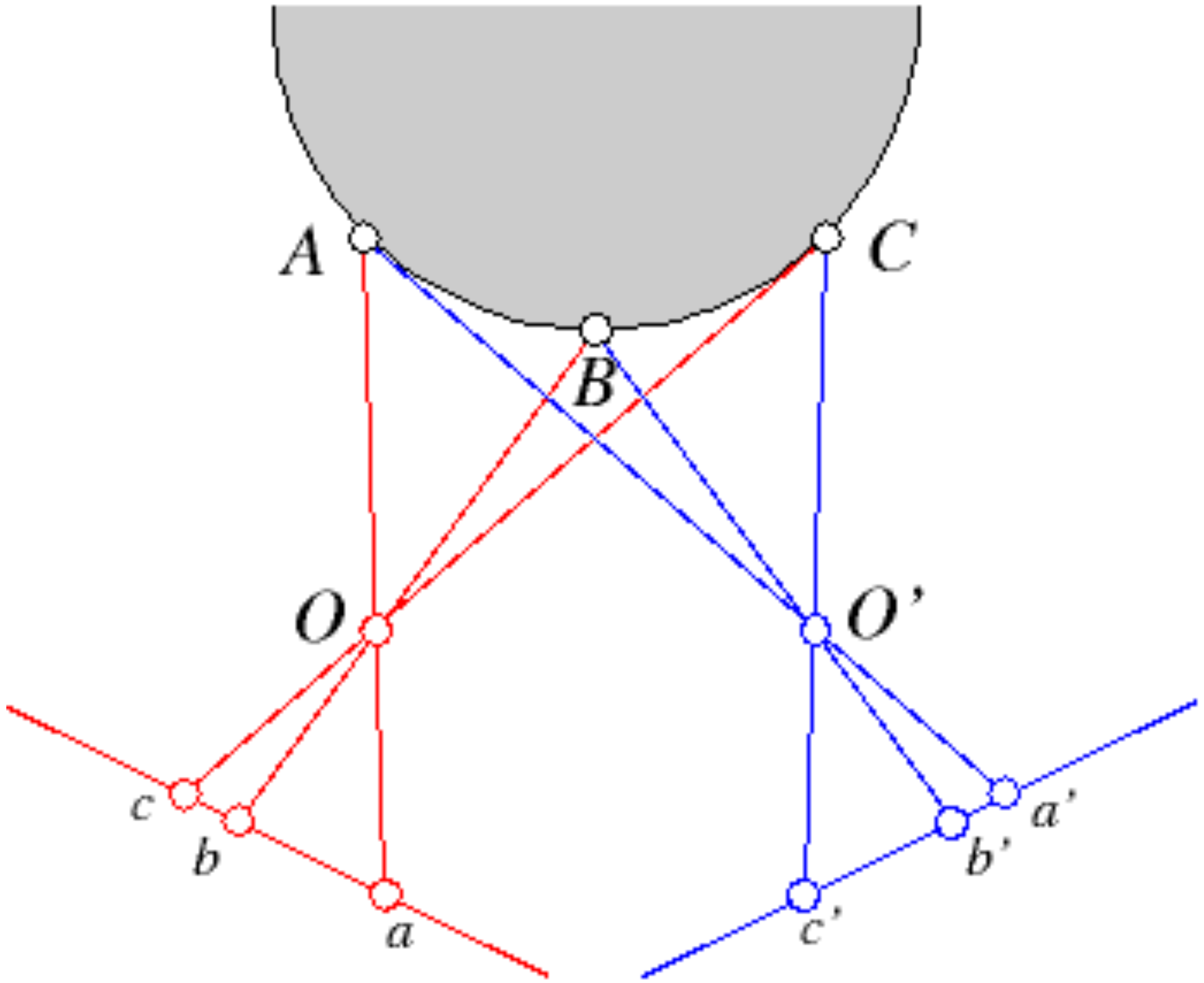


$W = 20$

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

Ordering Constraints

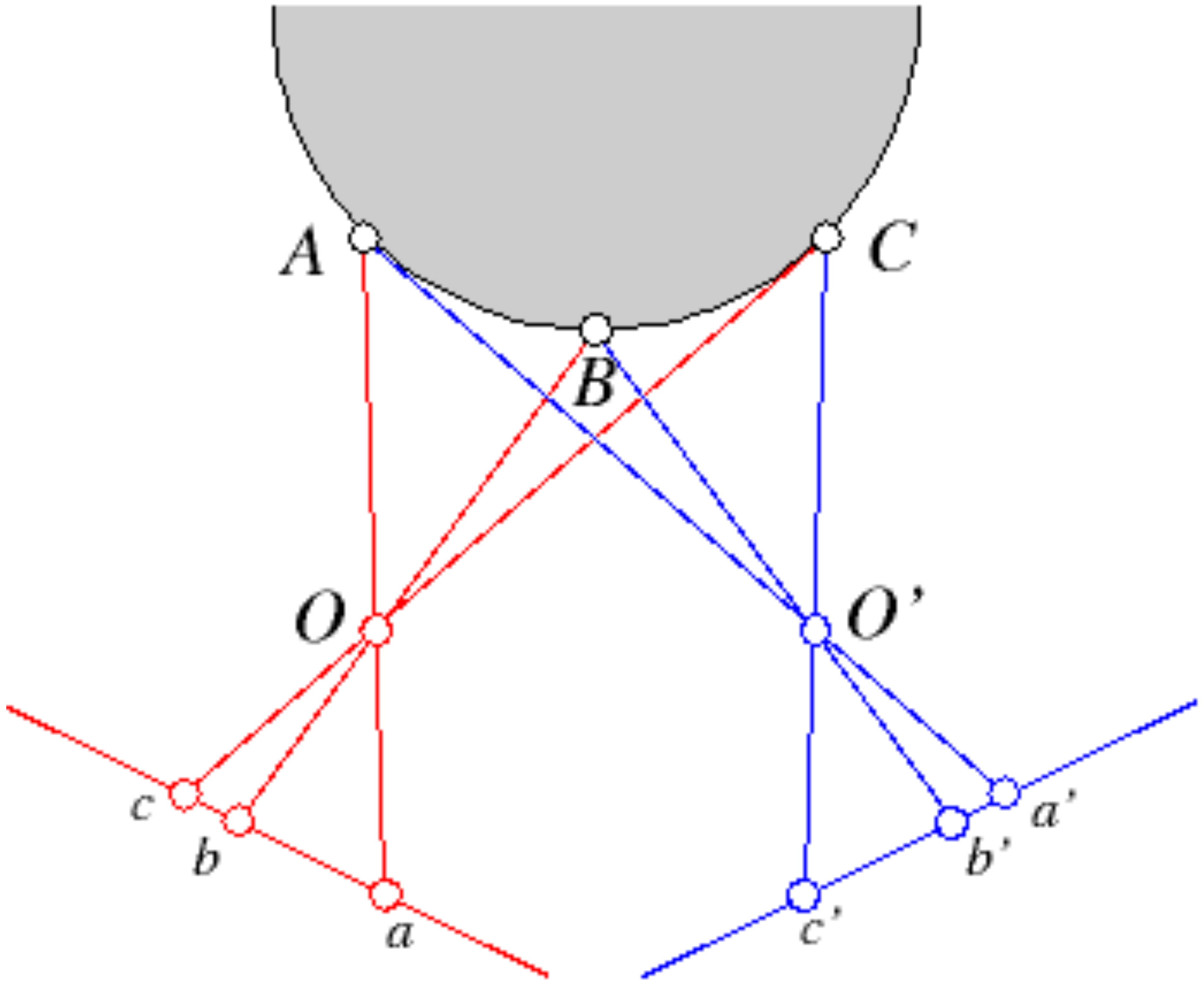
Ordering constraint …



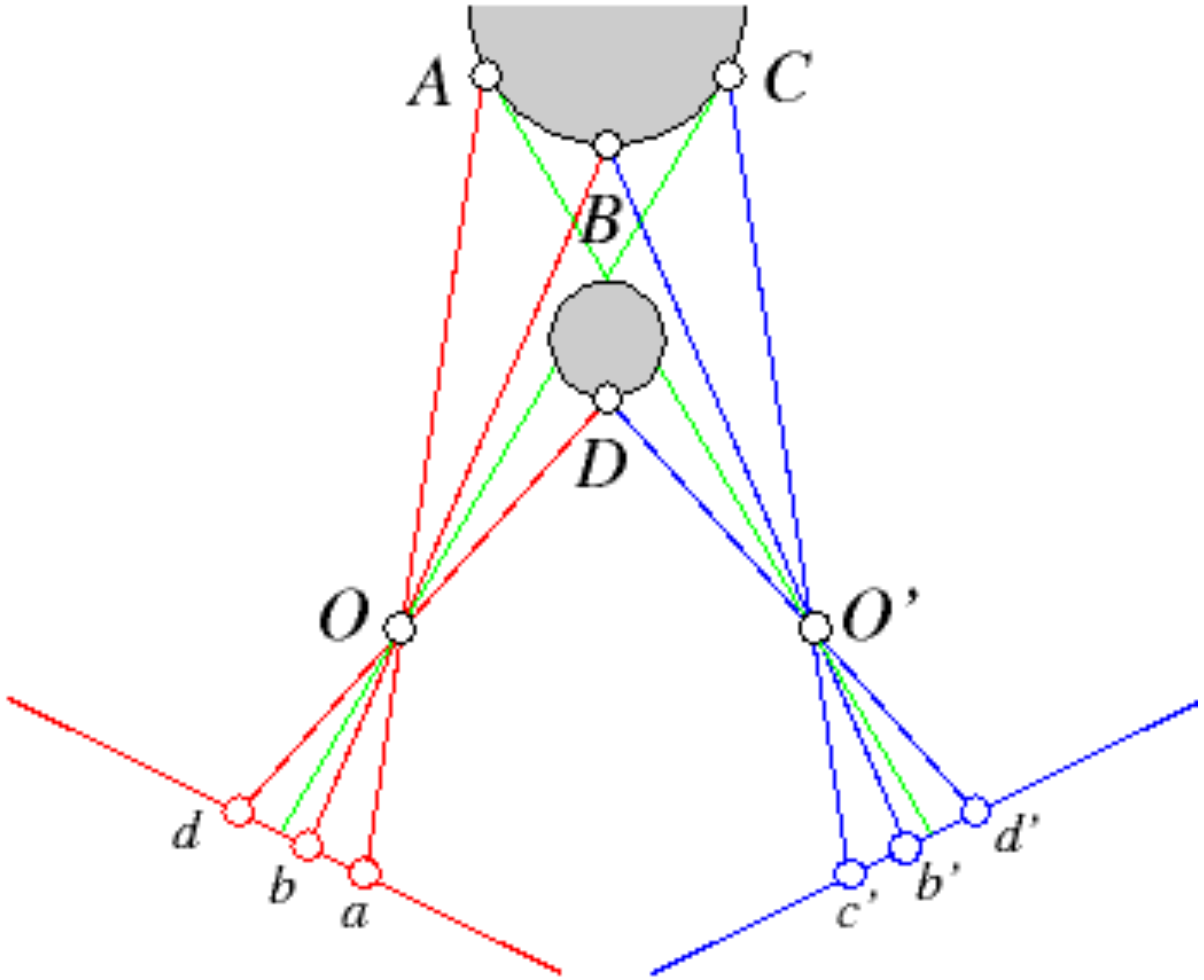
Forsyth & Ponce (2nd ed.) Figure 7.13

Ordering Constraints

Ordering constraint …

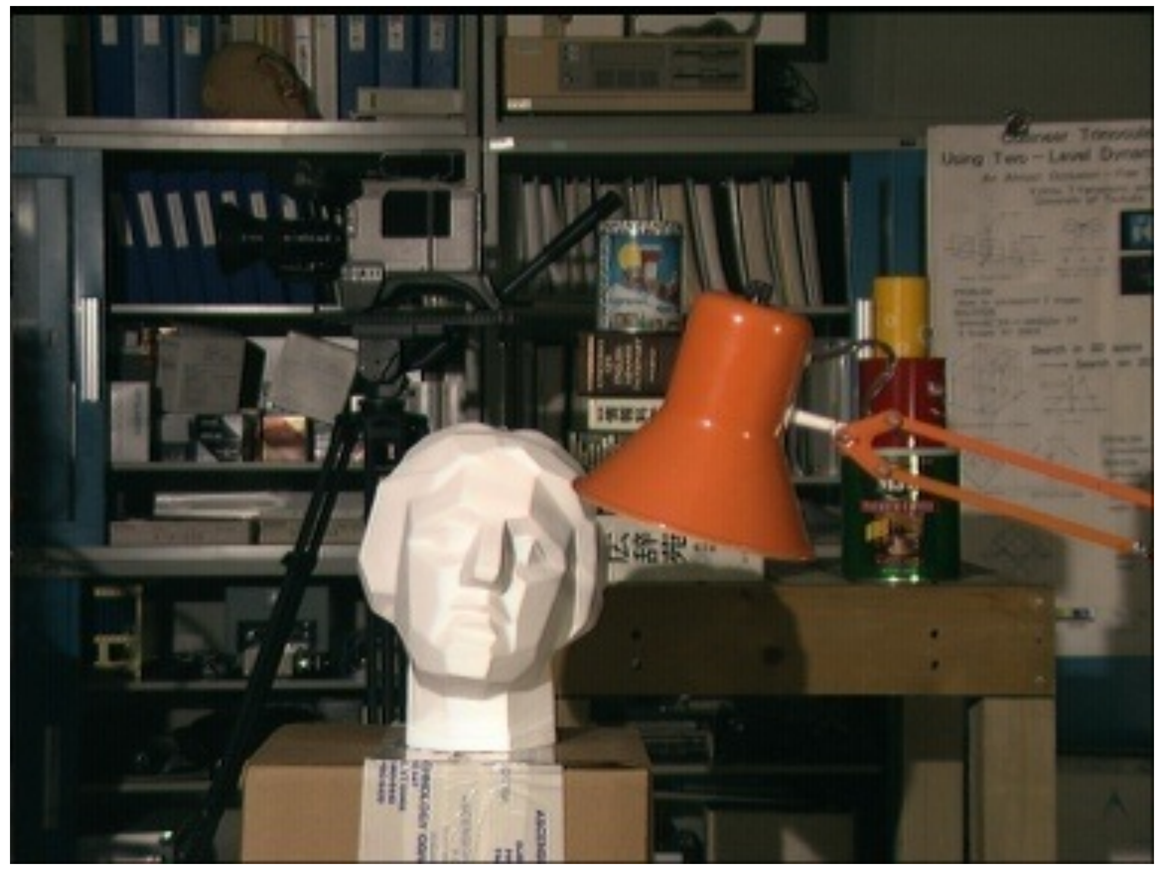


…. and a **failure** case

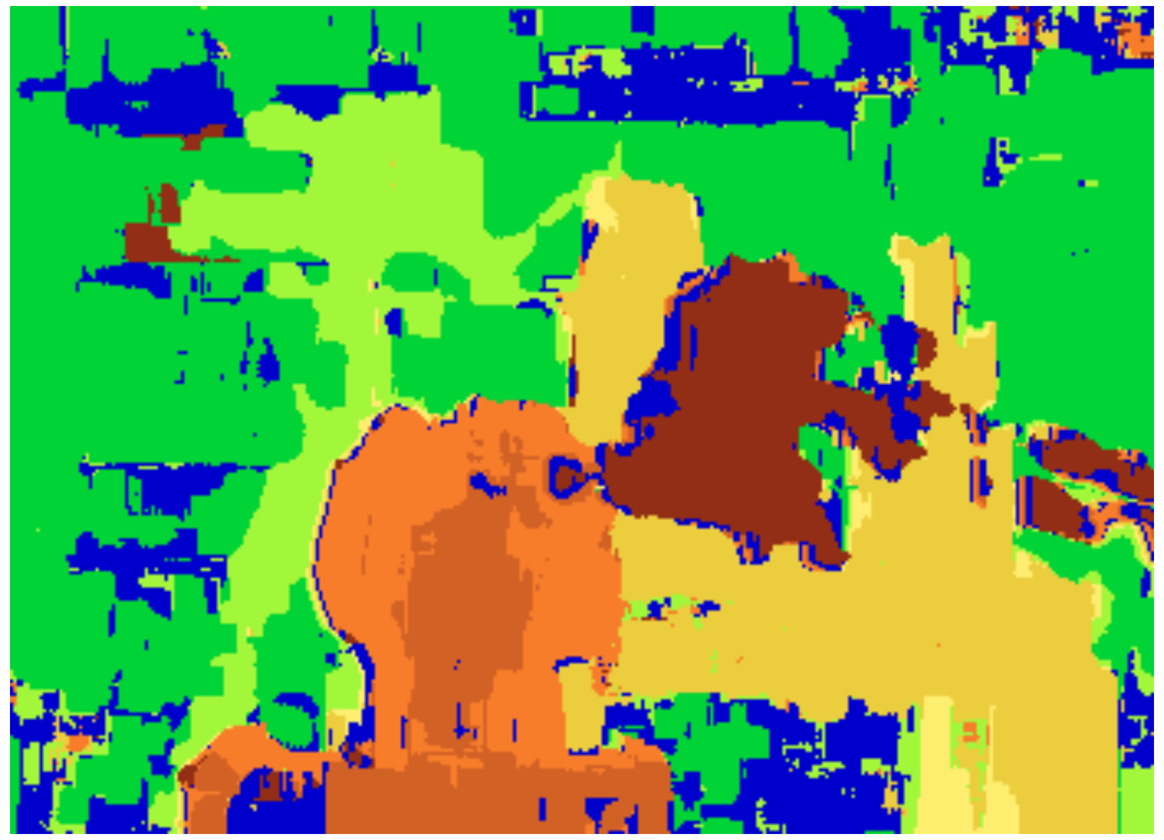


Forsyth & Ponce (2nd ed.) Figure 7.13

Block Matching Techniques: Result



Block matching



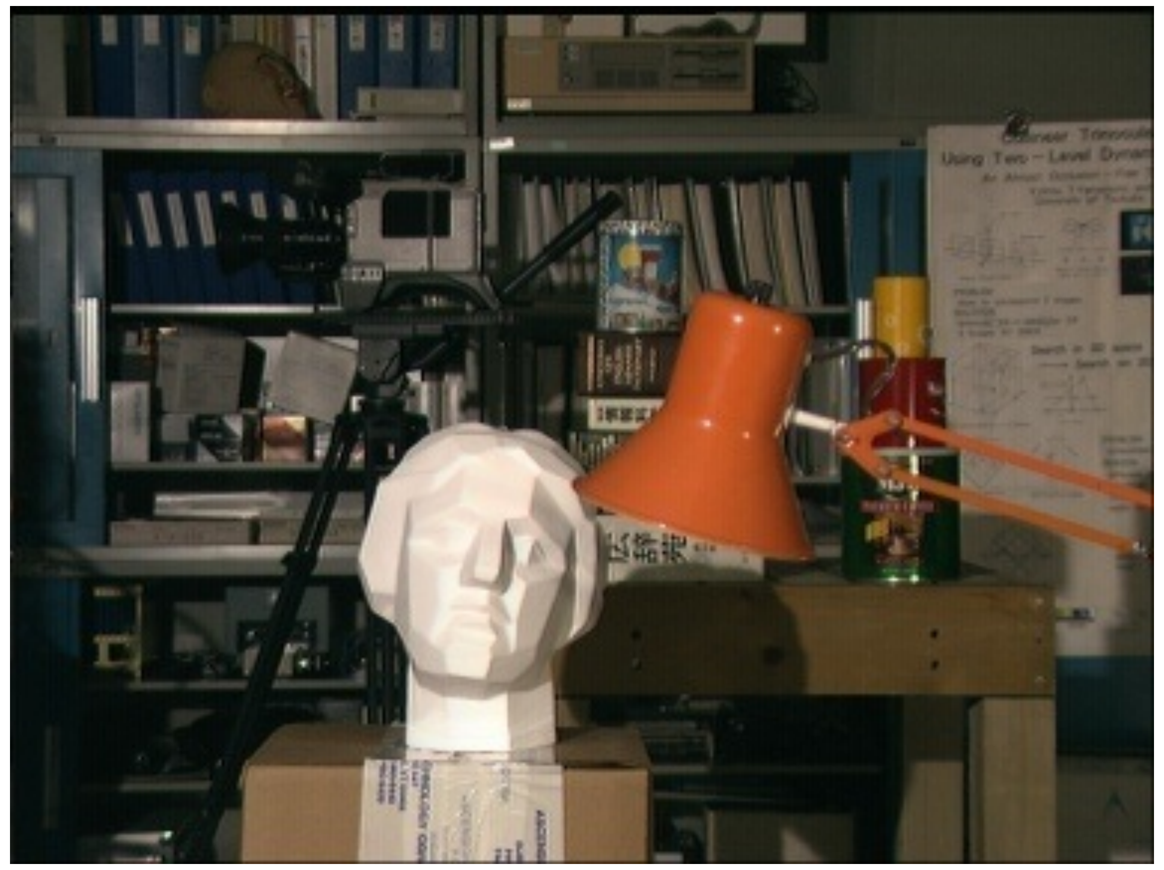
Ground truth

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

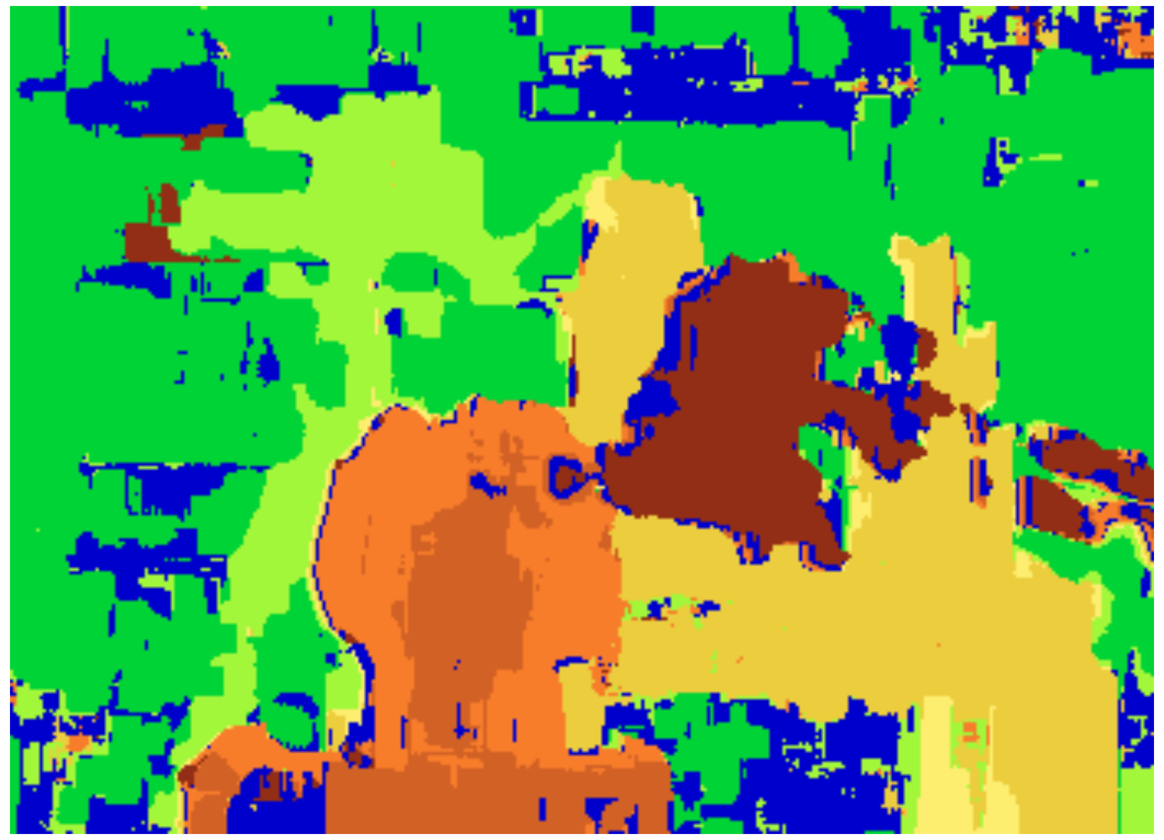
Block Matching Techniques: Result

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**

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

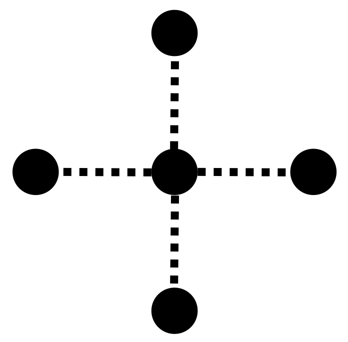
$$E_d(d) = \sum_{(x,y) \in I} C(x, y, d(x, y))$$

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

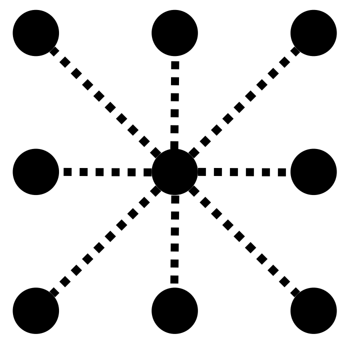
$$E_s(d) = \sum_{(p,q) \in \mathcal{E}} V(d_p, d_q)$$

smoothness term

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



4-connected neighborhood



8-connected neighborhood

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

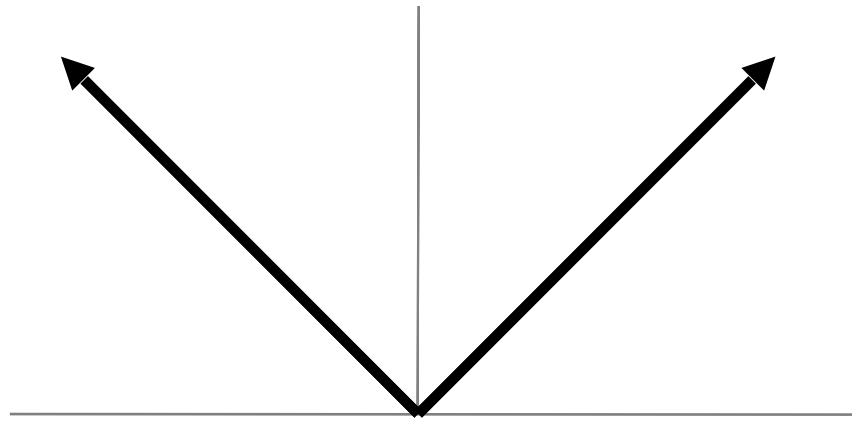
Stereo Matching as **Energy Minimization**

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

smoothness term

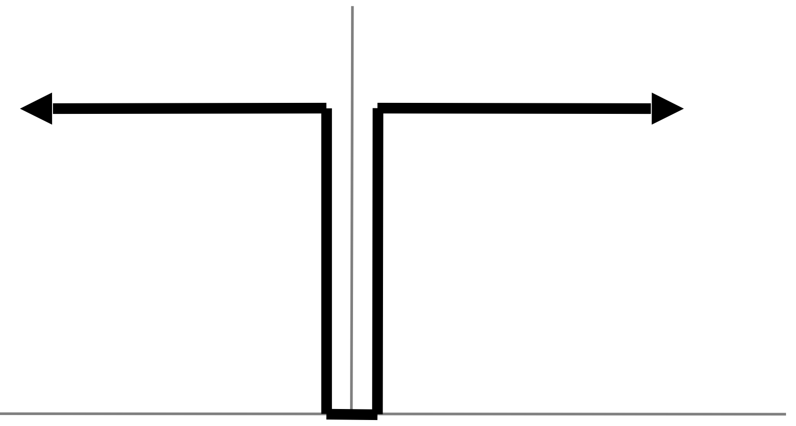
$$V(d_p, d_q) = |d_p - d_q|$$

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)

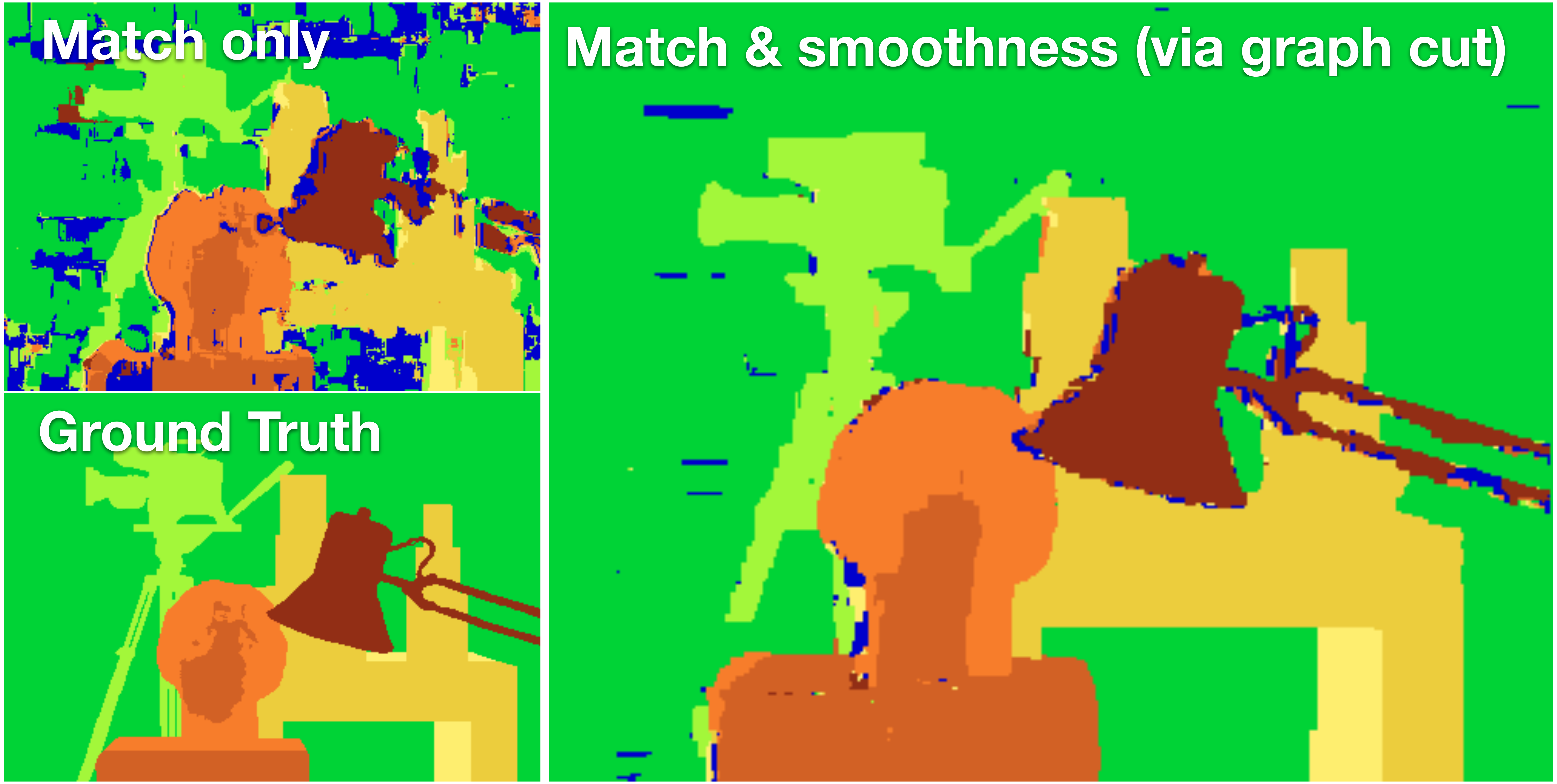
Stereo Matching as **Energy Minimization**: Solution

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

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

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

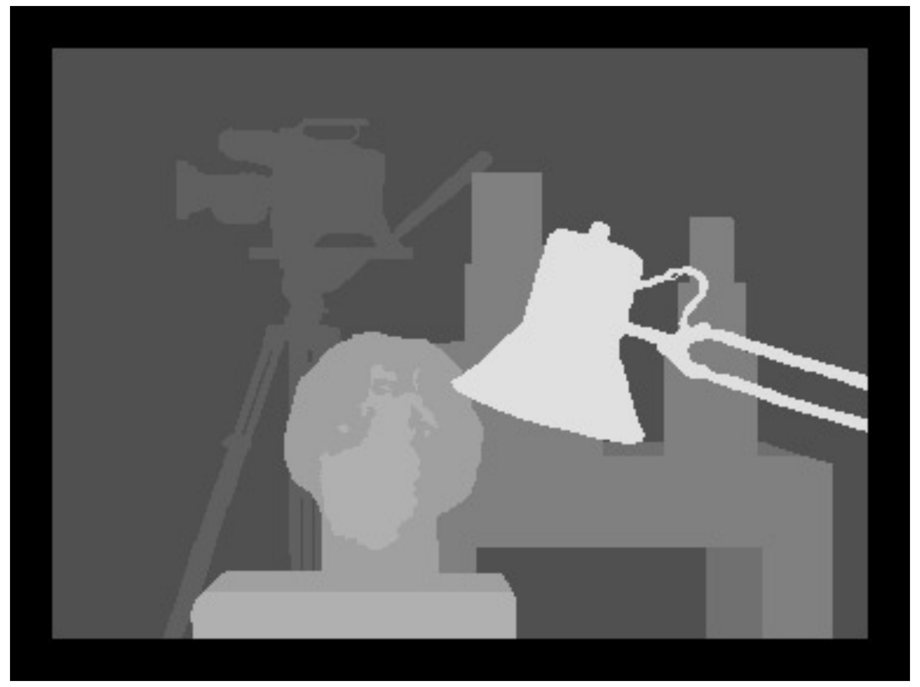
Stereo Matching as **Energy Minimization**



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

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Stereo Matching as **Energy Minimization**



Ground truth

Graph Cuts
[Kolmogorov Zabih 2001]

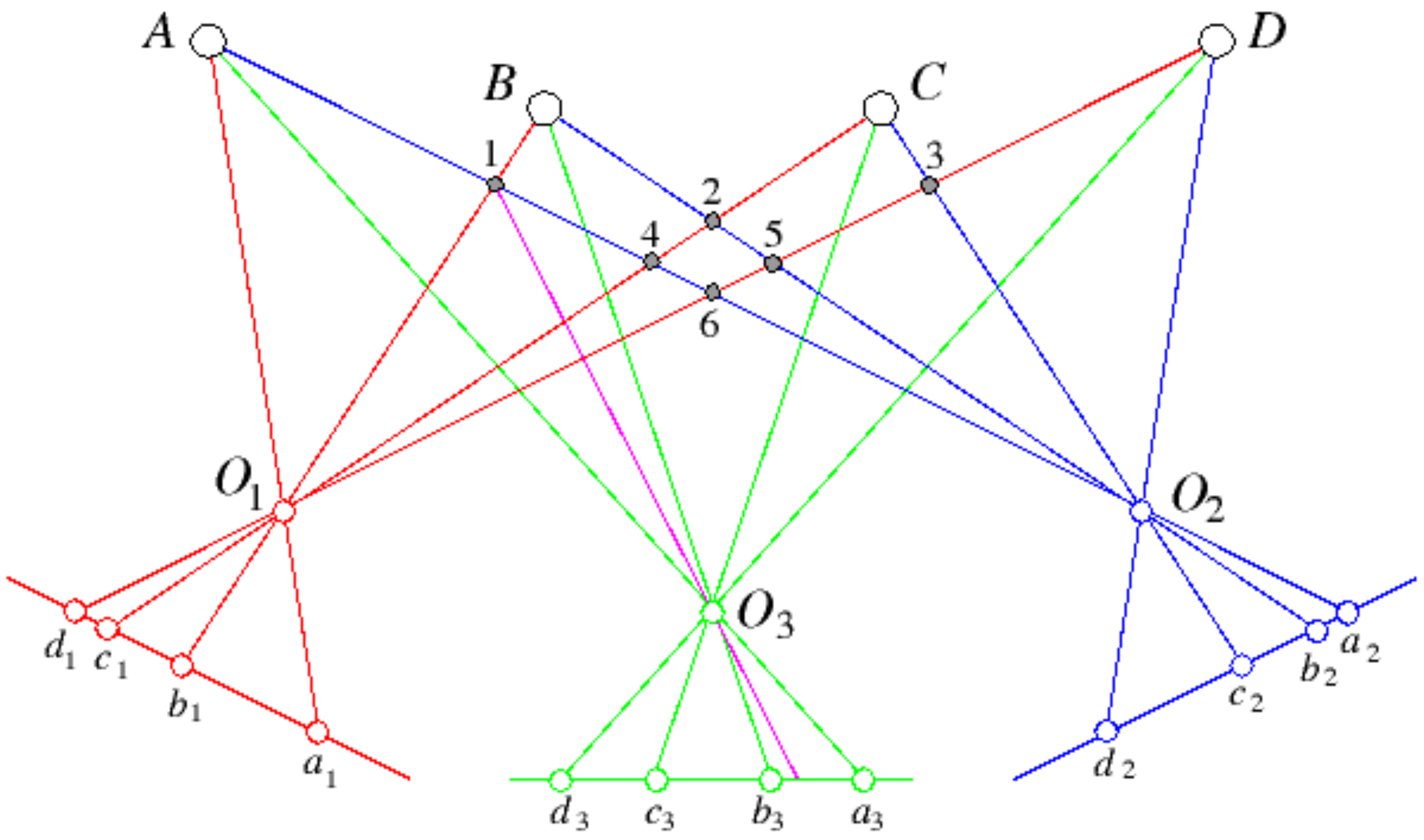
Dynamic Programming

SSD 21px aggregation

[Scharstein Szeliski 2002]

Idea: Use More Cameras

Adding a third camera reduces ambiguity in stereo matching



Forsyth & Ponce (2nd ed.) Figure 7.17

Point Grey Research **Digiclops**

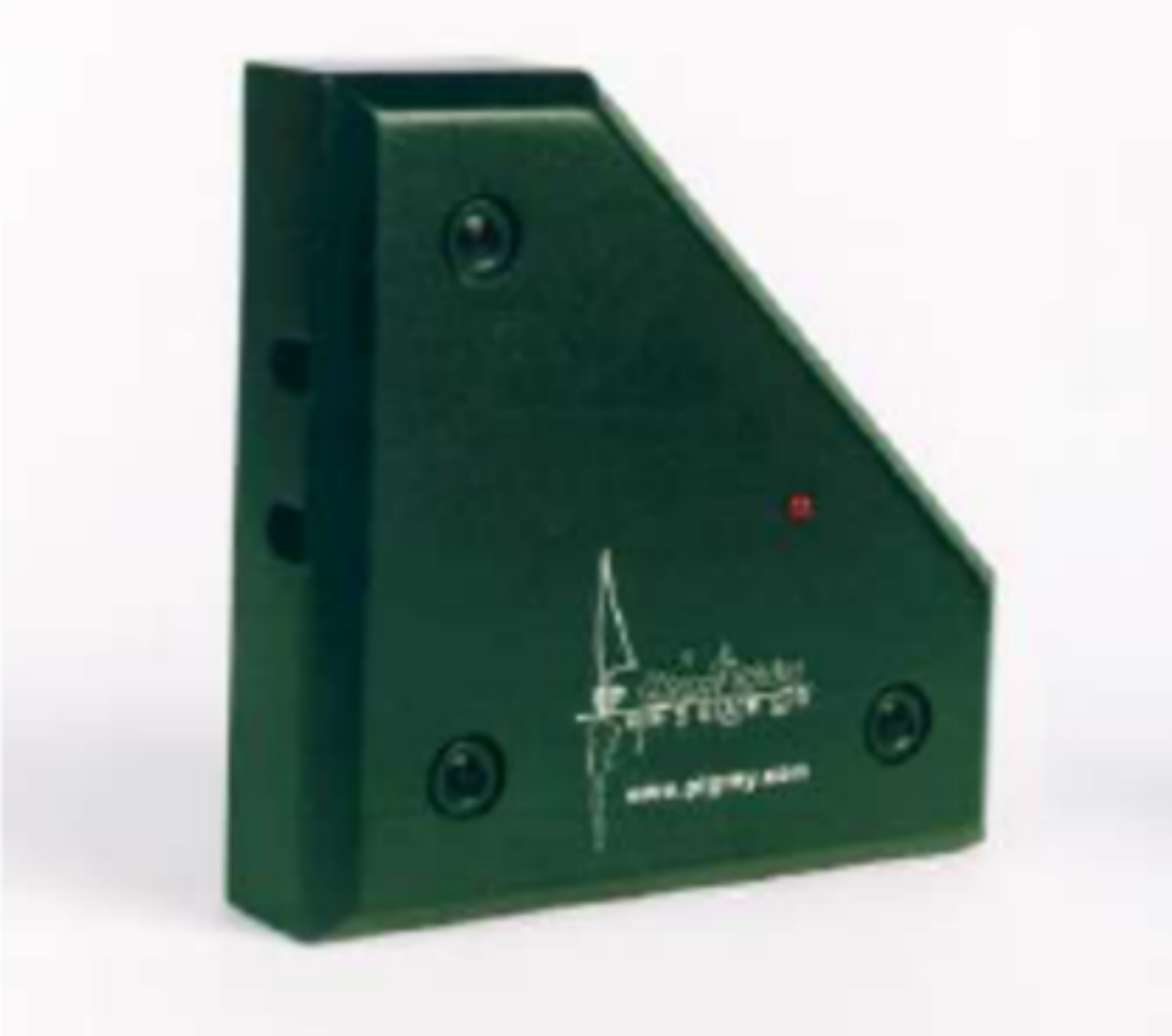
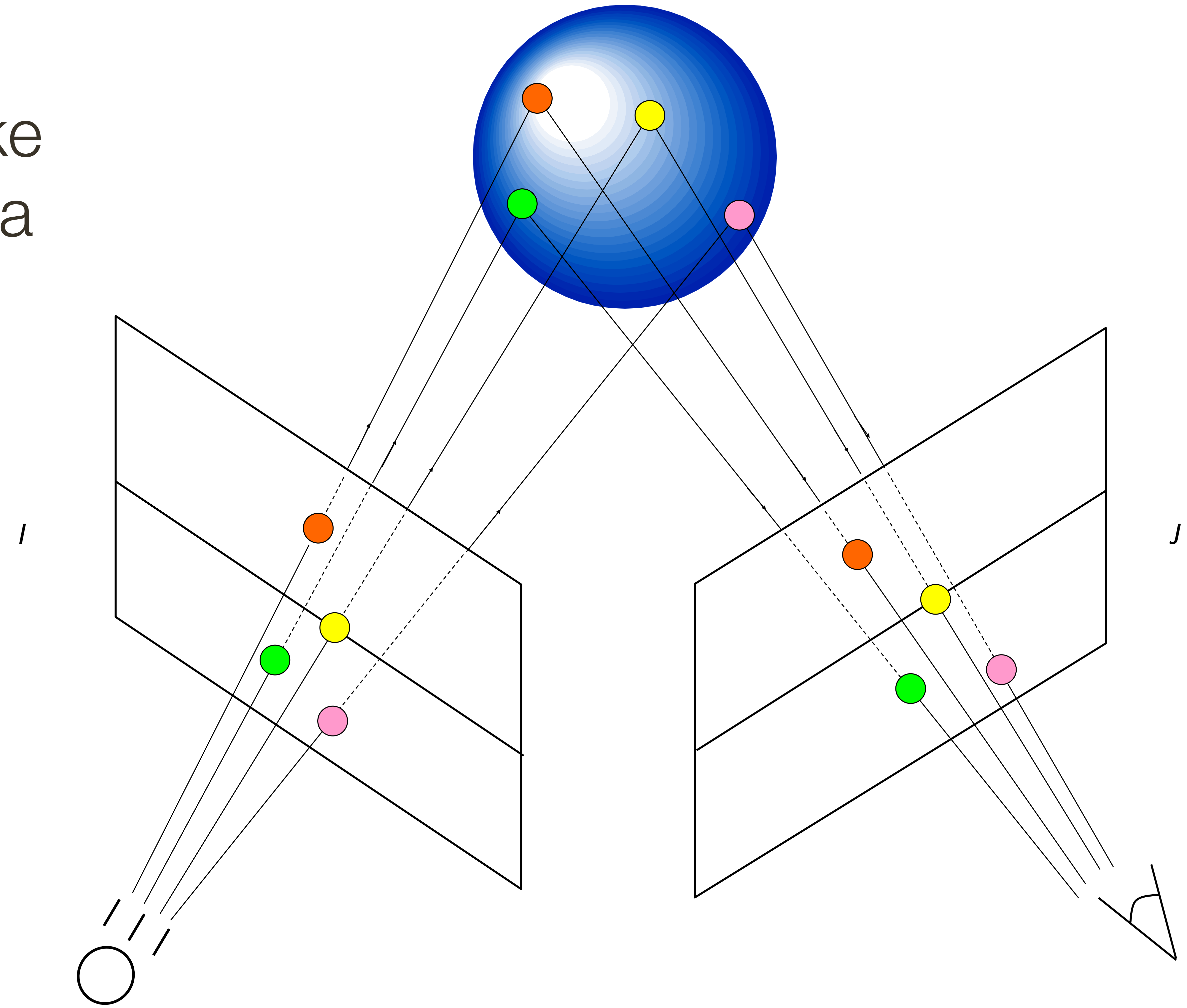


Image credit: Point Grey Research

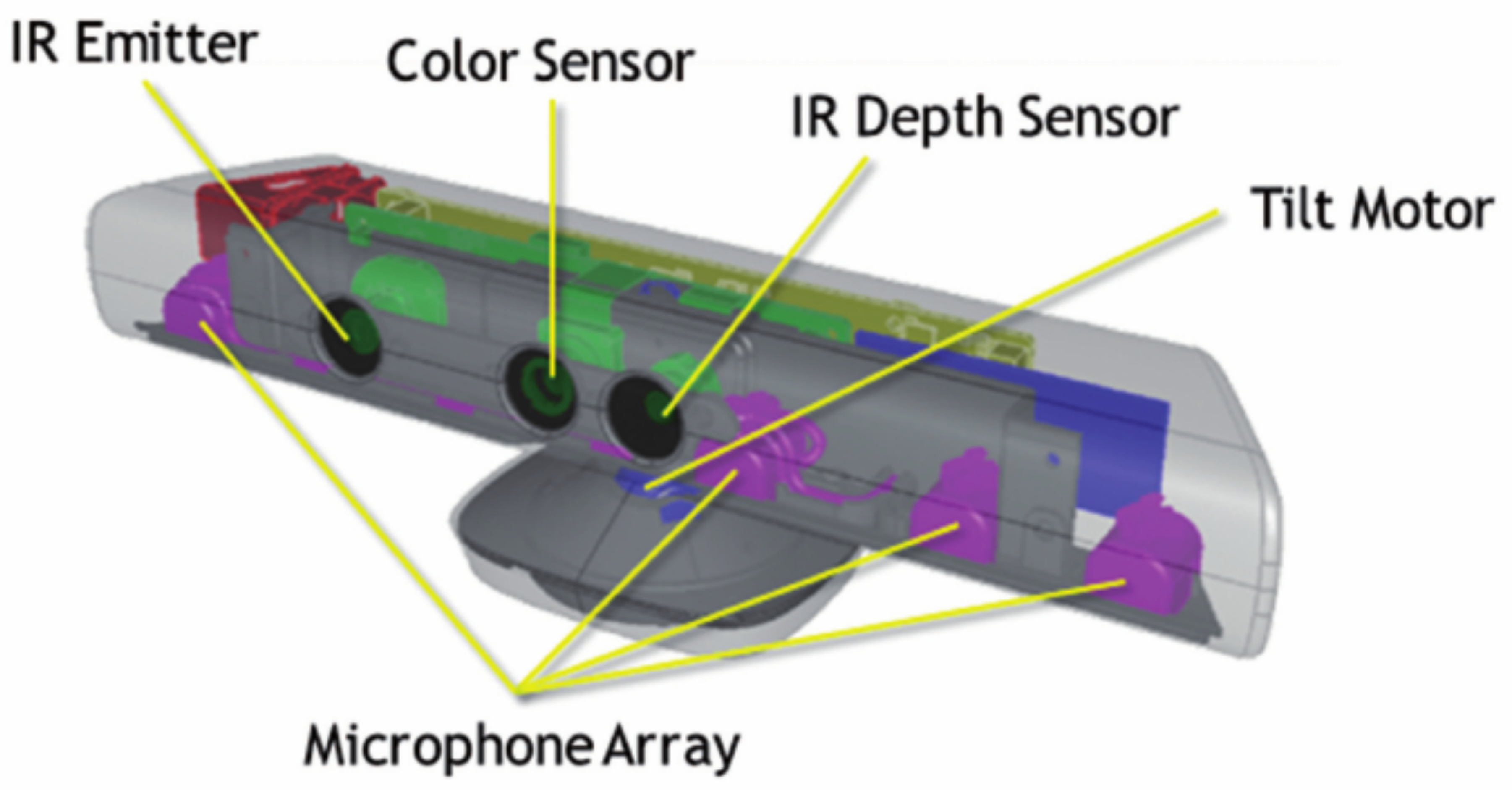
Structured Light Imaging: Structured Light and One Camera

Projector acts like
“reverse” camera

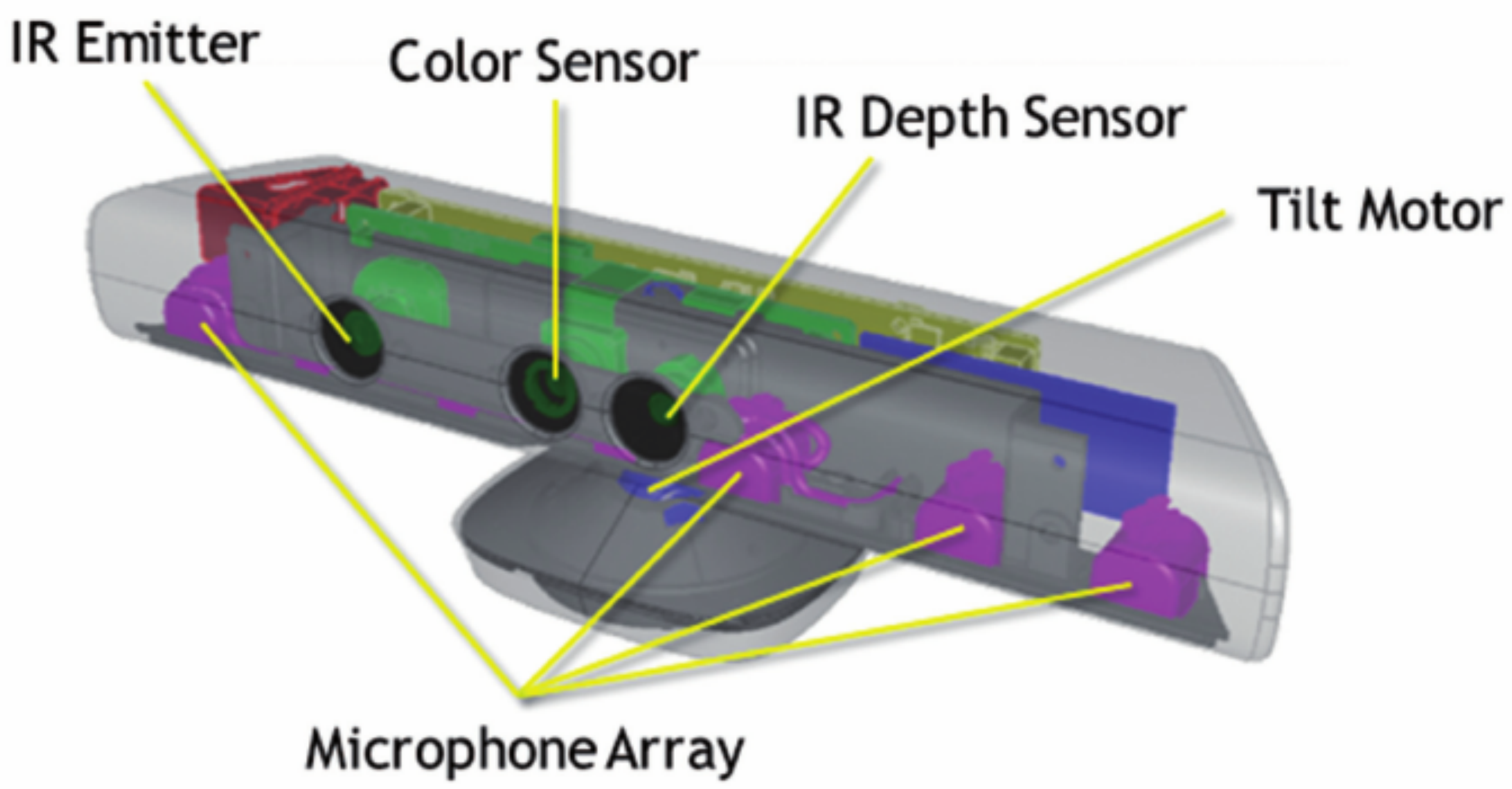


Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Microsoft **Kinect**



Microsoft **Kinect**



Stereo Vision Summary

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

Stereo algorithms work by finding **matches** between points along corresponding lines in a second image, known as epipolar lines.

A point in one image projects to an **epipolar line** in a second image

In an axis-aligned / rectified stereo setup, matches are found along horizontal scanlines