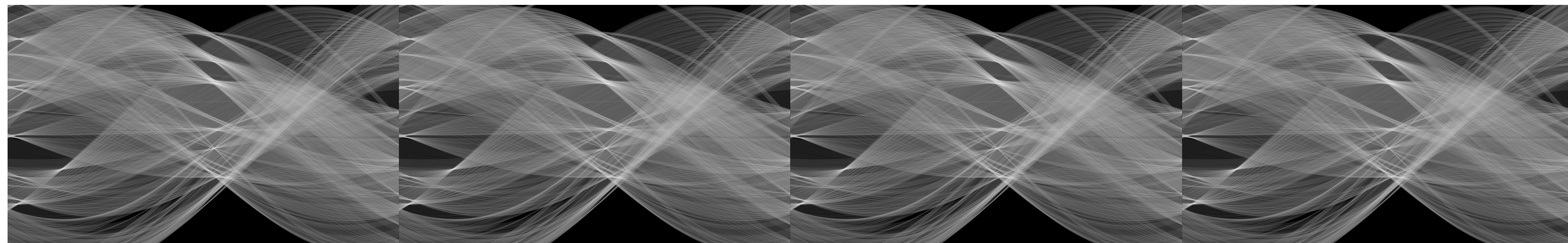




# CPSC 425: Computer Vision



**Image Credit:** Ioannis (Yannis) Gkioulekas (CMU)

## Lecture 16: Stereo

# Menu for Today (March 9, 2020)

## Topics:

- Stereo Vision
- More Than 2 Cameras
- Structured Light
- Optical Flow

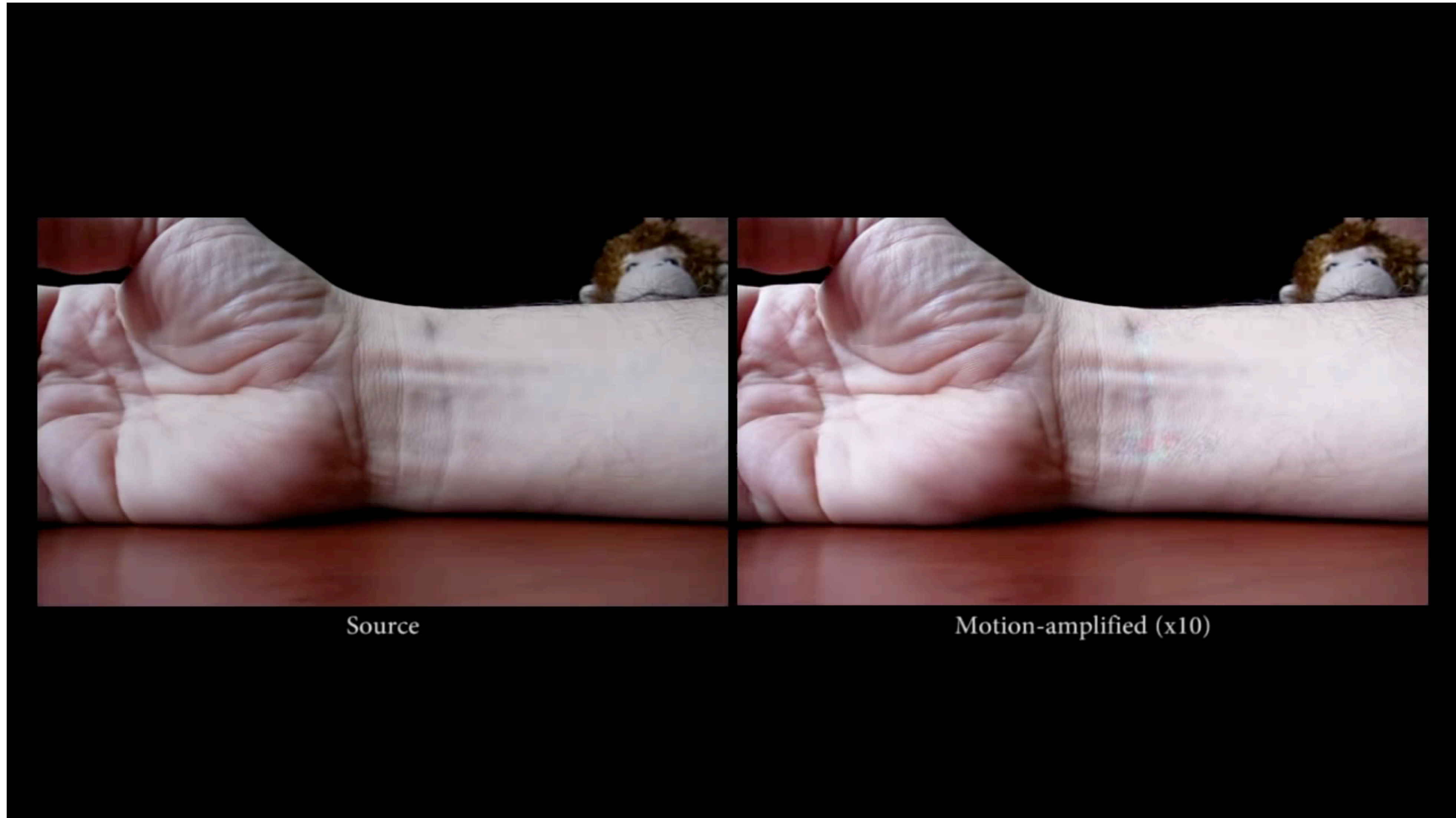
## Readings:

- **Today's** Lecture: Forsyth & Ponce (2nd ed.) 10.6, 6.2.2, 9.3.1, 9.3.3, 9.4.2
- **Next** Lecture: None

## Reminders:

- **Assignment 4:** RANSAC and Panoramas due **March 17th**
- **Midterm** is 96% graded

# Today's “**fun**” Example: Eulerian Video Magnification



# Today's "fun" Example: Eulerian Video Magnification

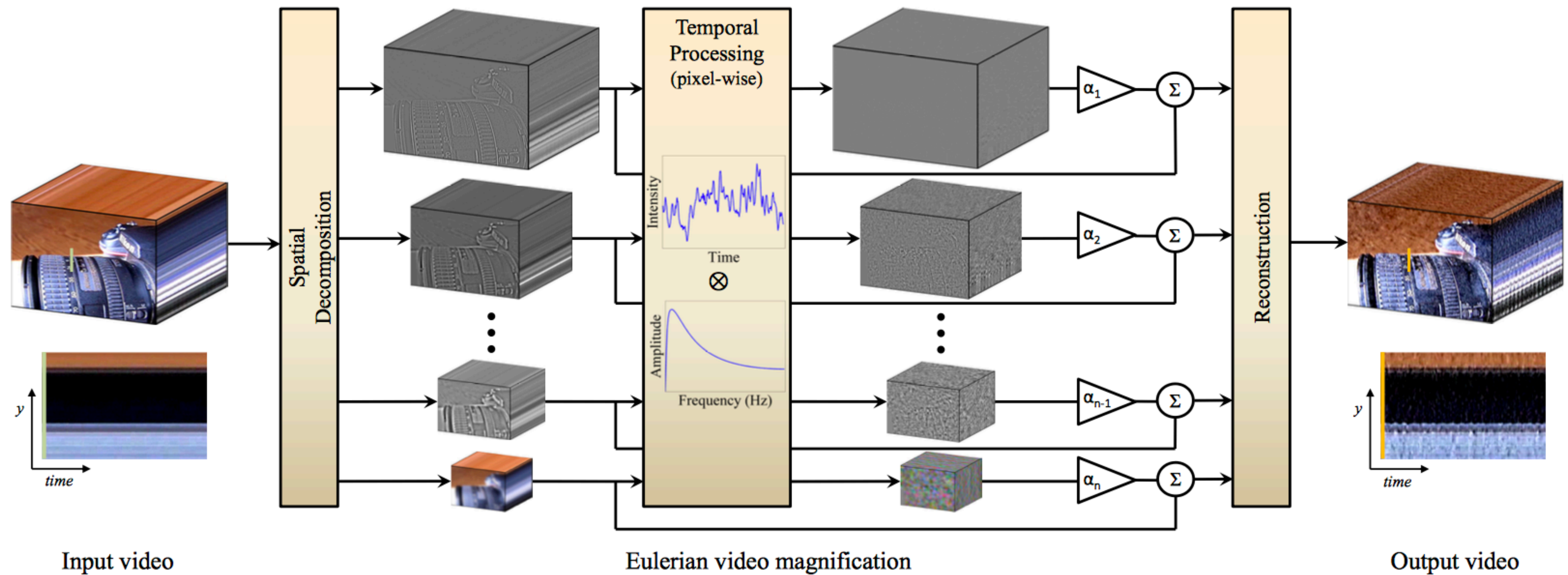


Figure From: Wu et al., Siggraph 2012

# 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

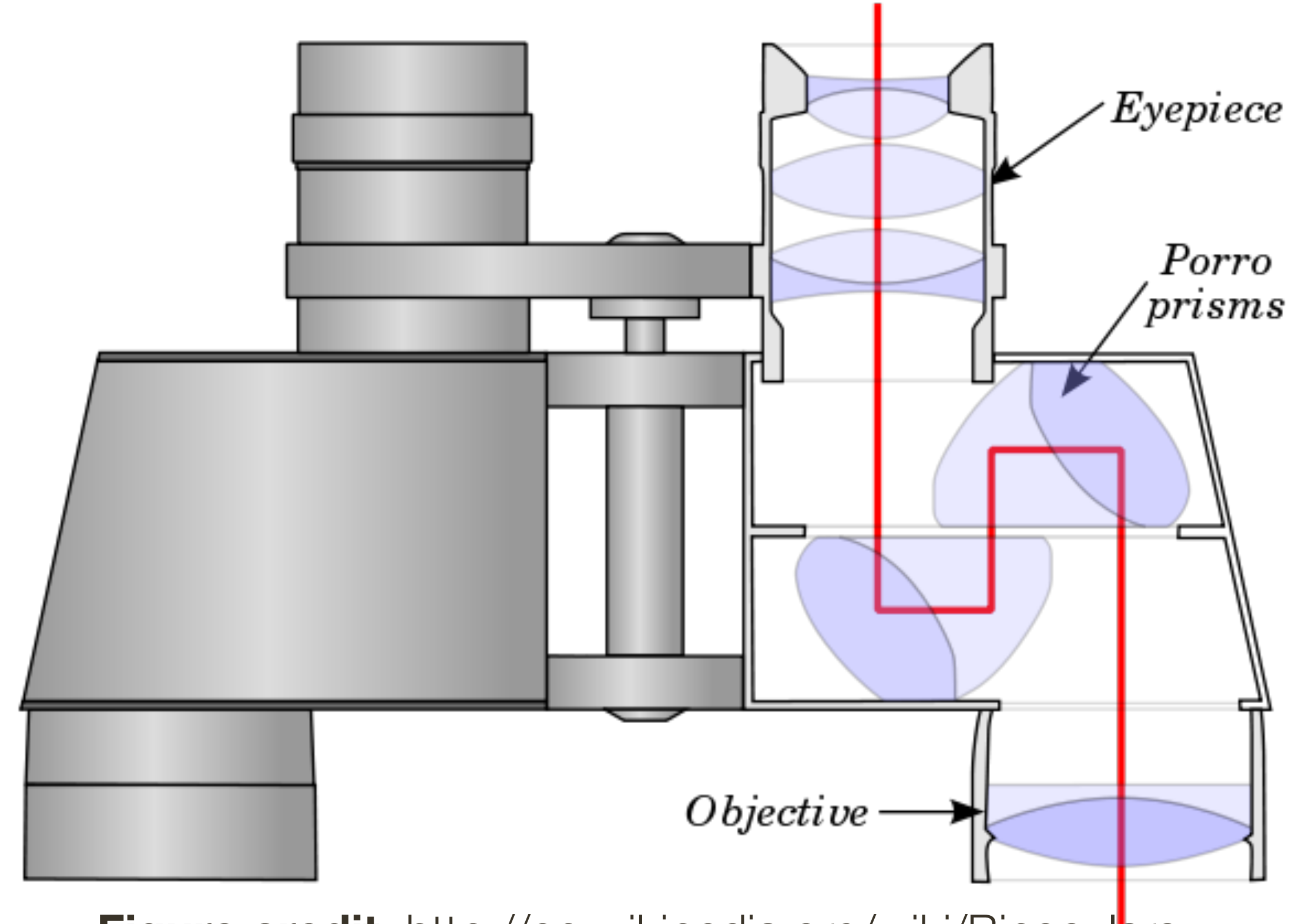
With two eyes, we acquire images of the world from slightly different viewpoints

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

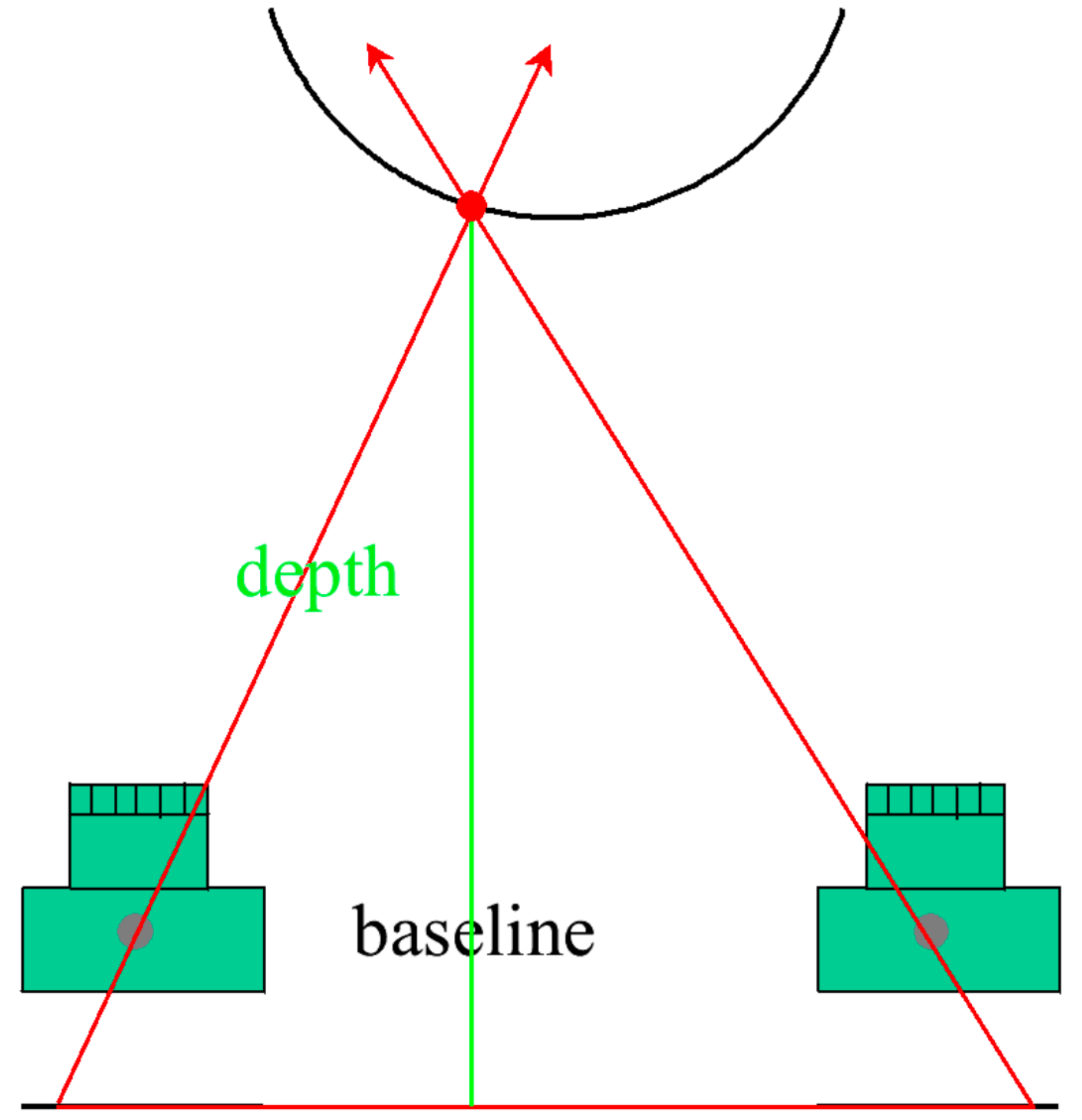
**Approach:** “Match” locations in one image to those in another

**Sub-tasks:**

- Calibrate cameras and camera positions
- Find all corresponding points (the hardest part)
- Compute depth and surfaces

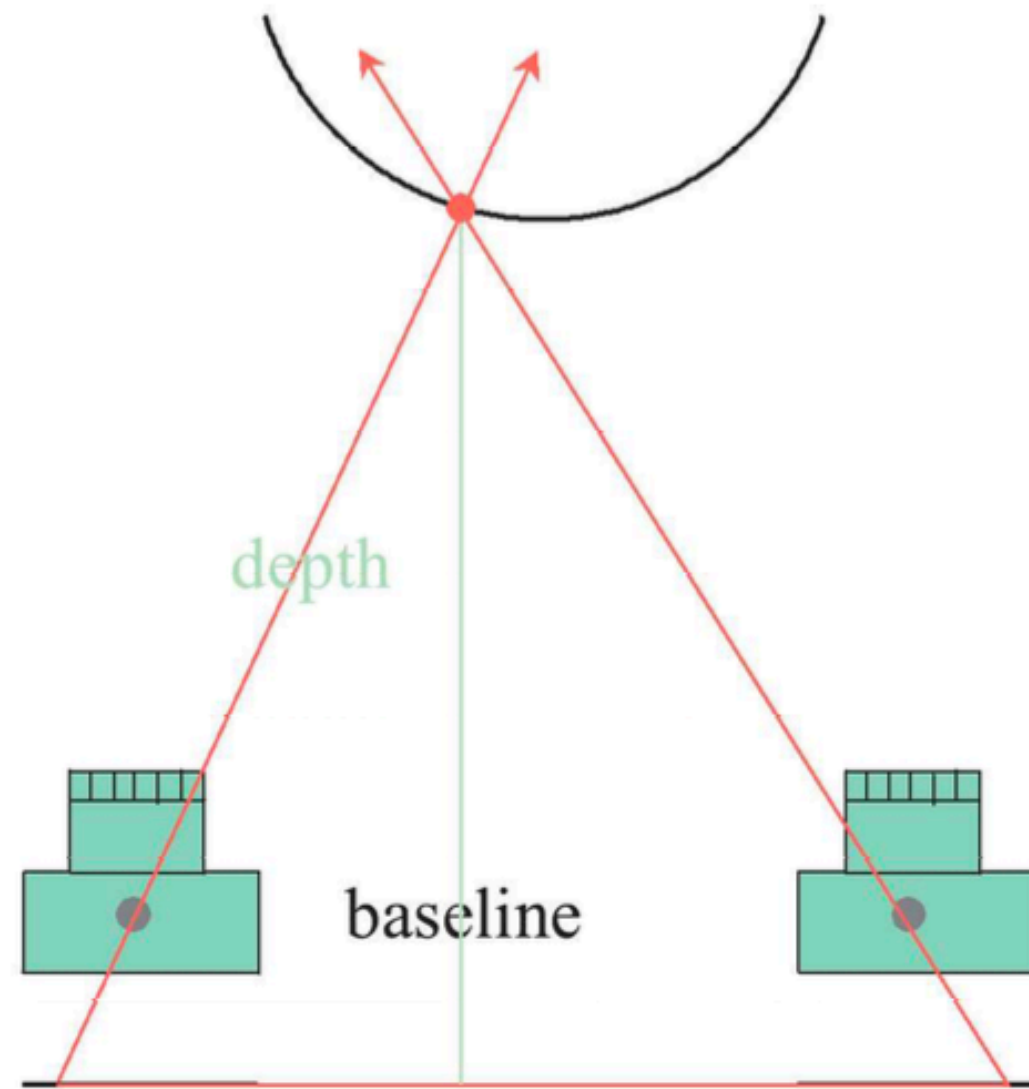


# Stereo Vision



Slide credit: Trevor Darrell

# Stereo Vision

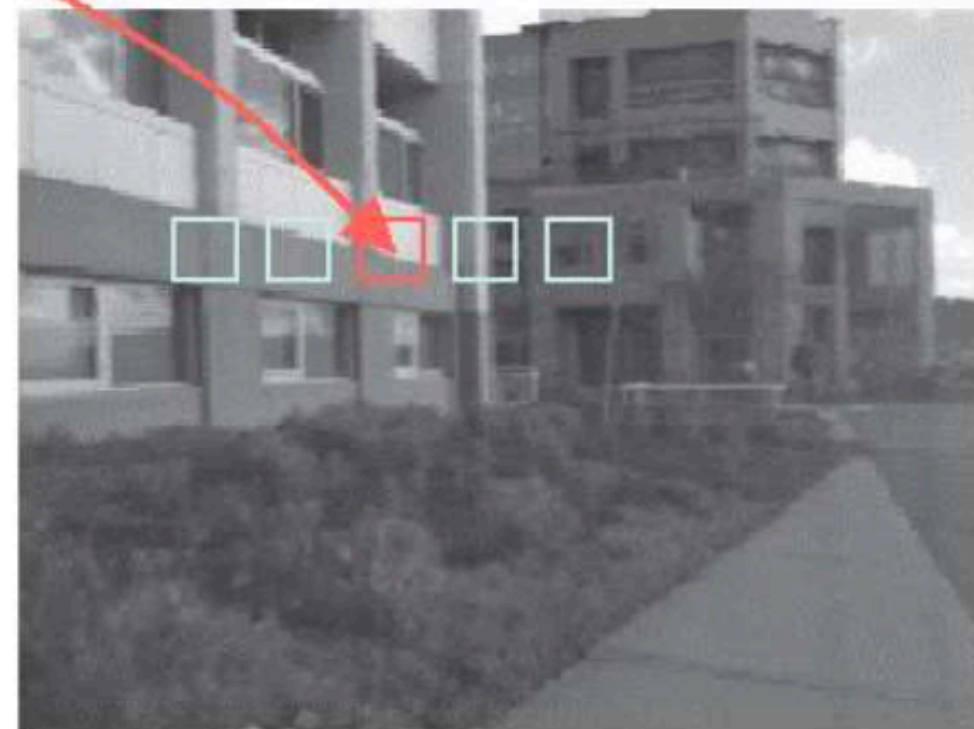


Triangulate on two images of the same point

Left



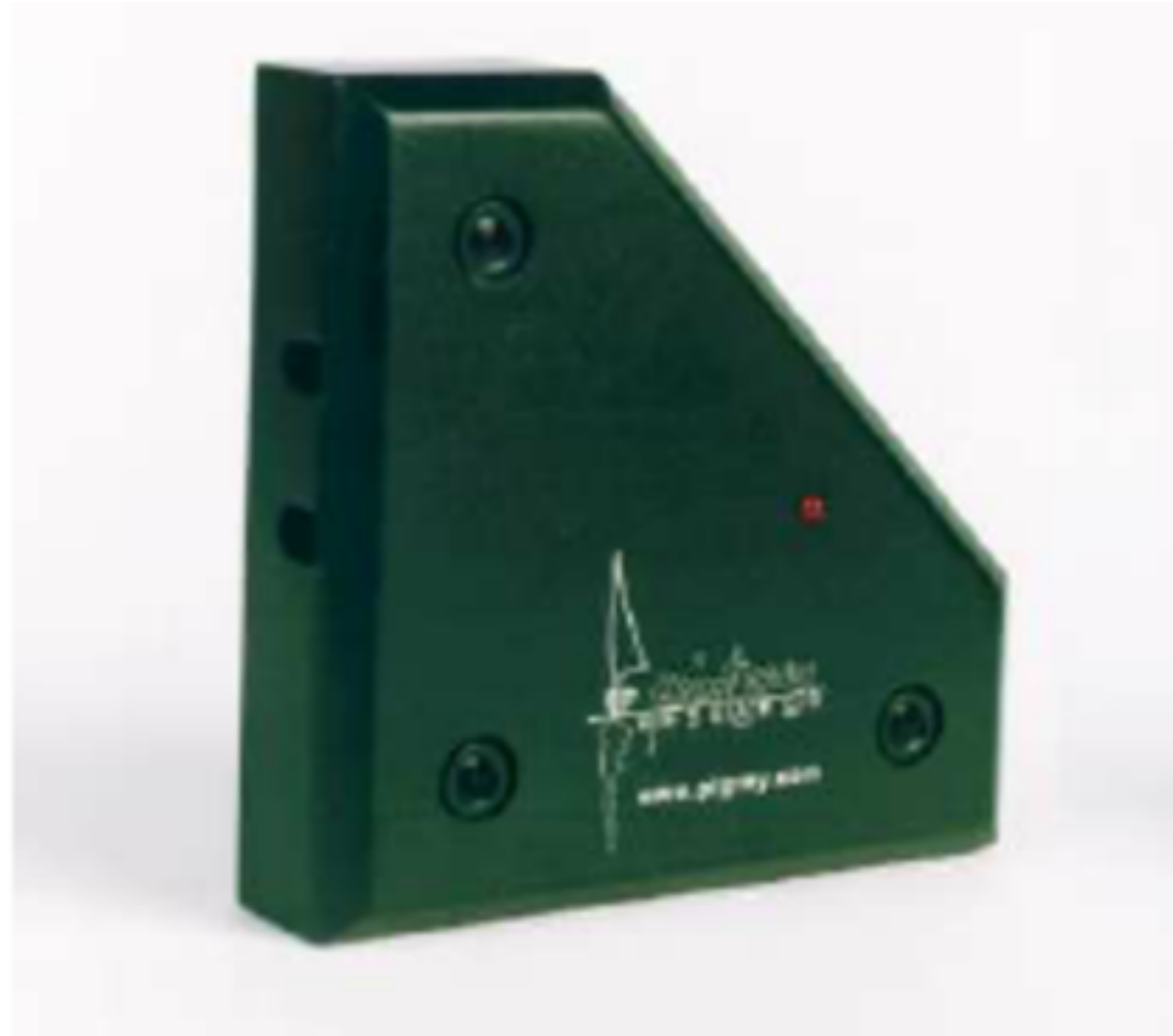
Right



Match correlation windows across scan lines

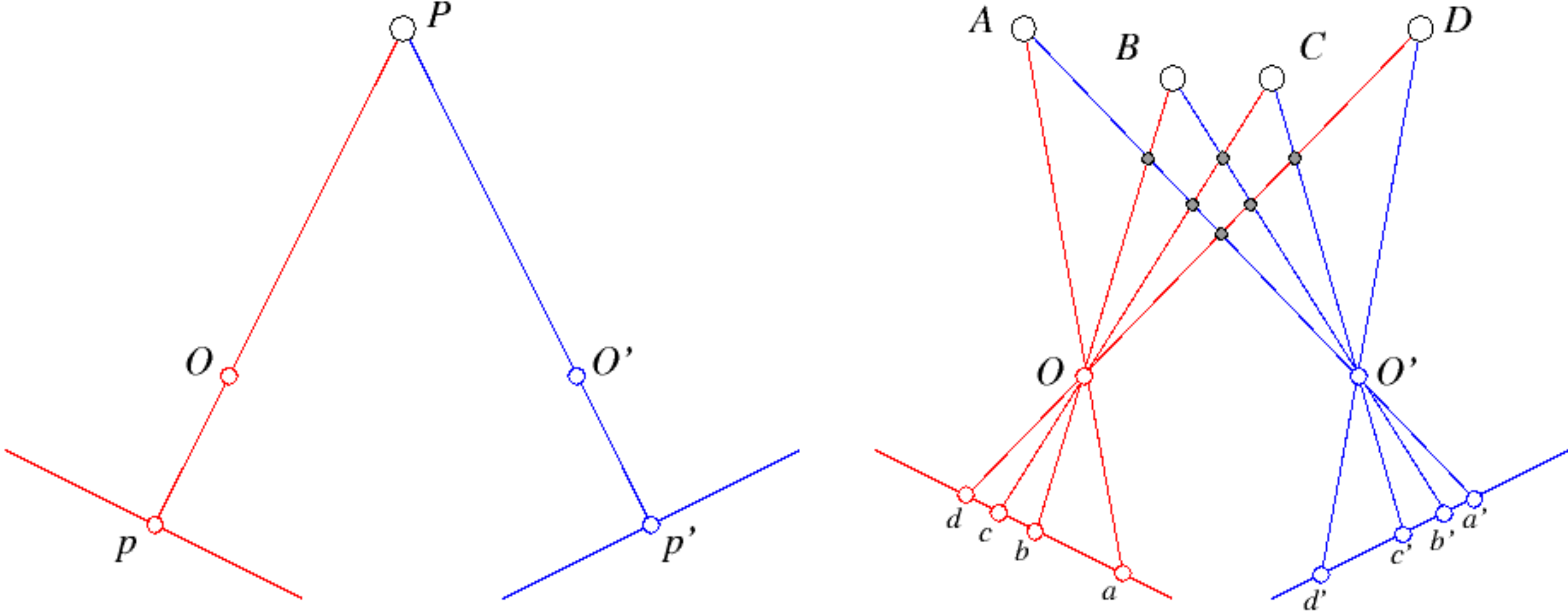
**Image credit:** Point Grey Research  
**Slide credit:** Trevor Darrell

# Point Grey Research **Digiclops**



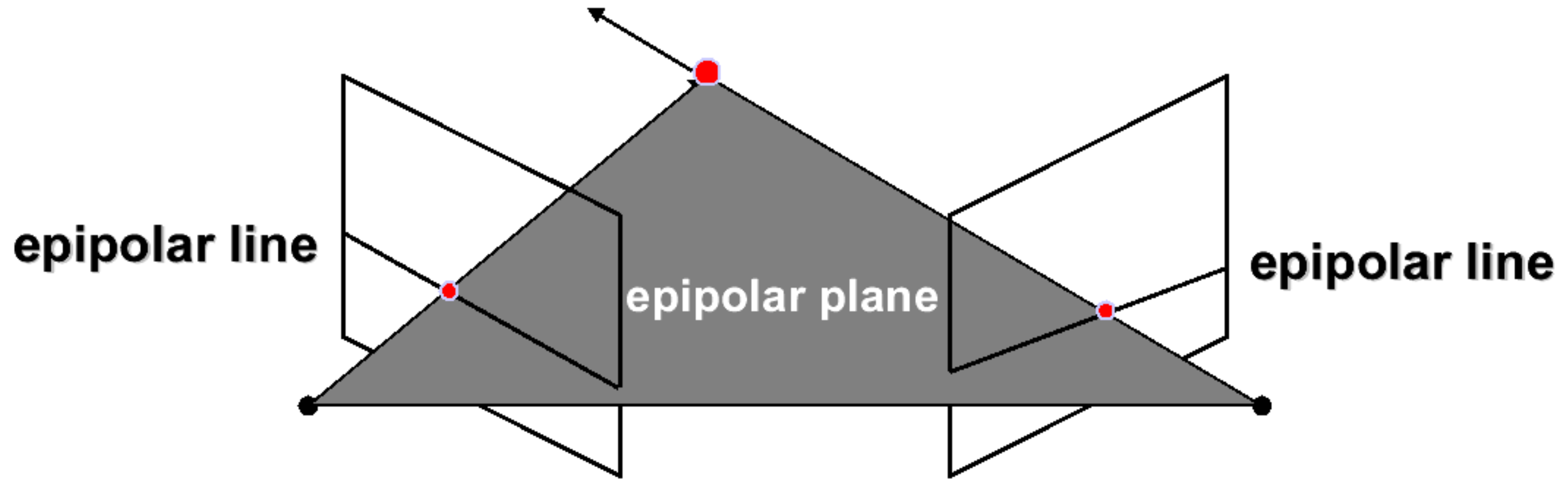
**Image credit:** Point Grey Research

# 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

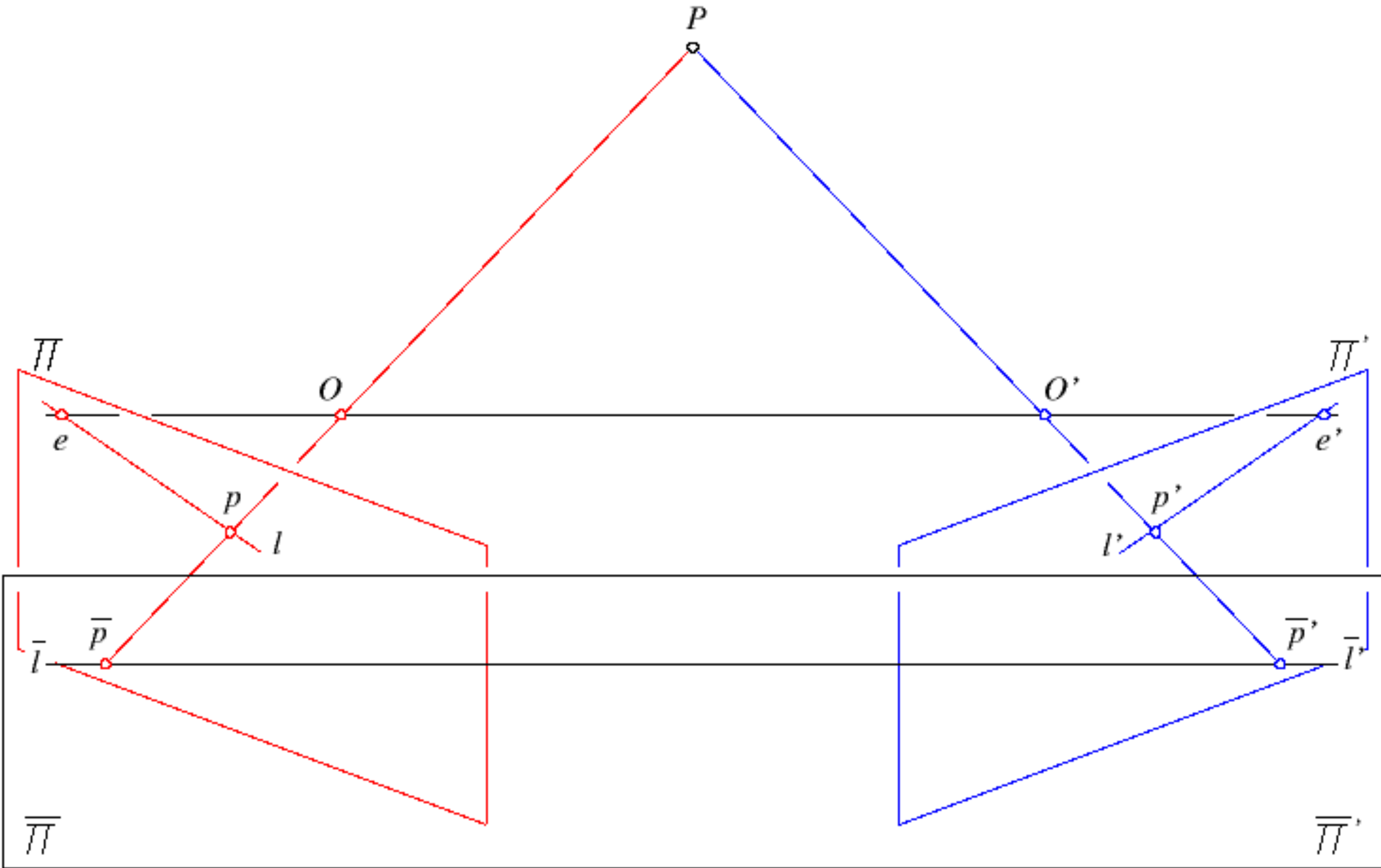
Focal **lengths** same

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

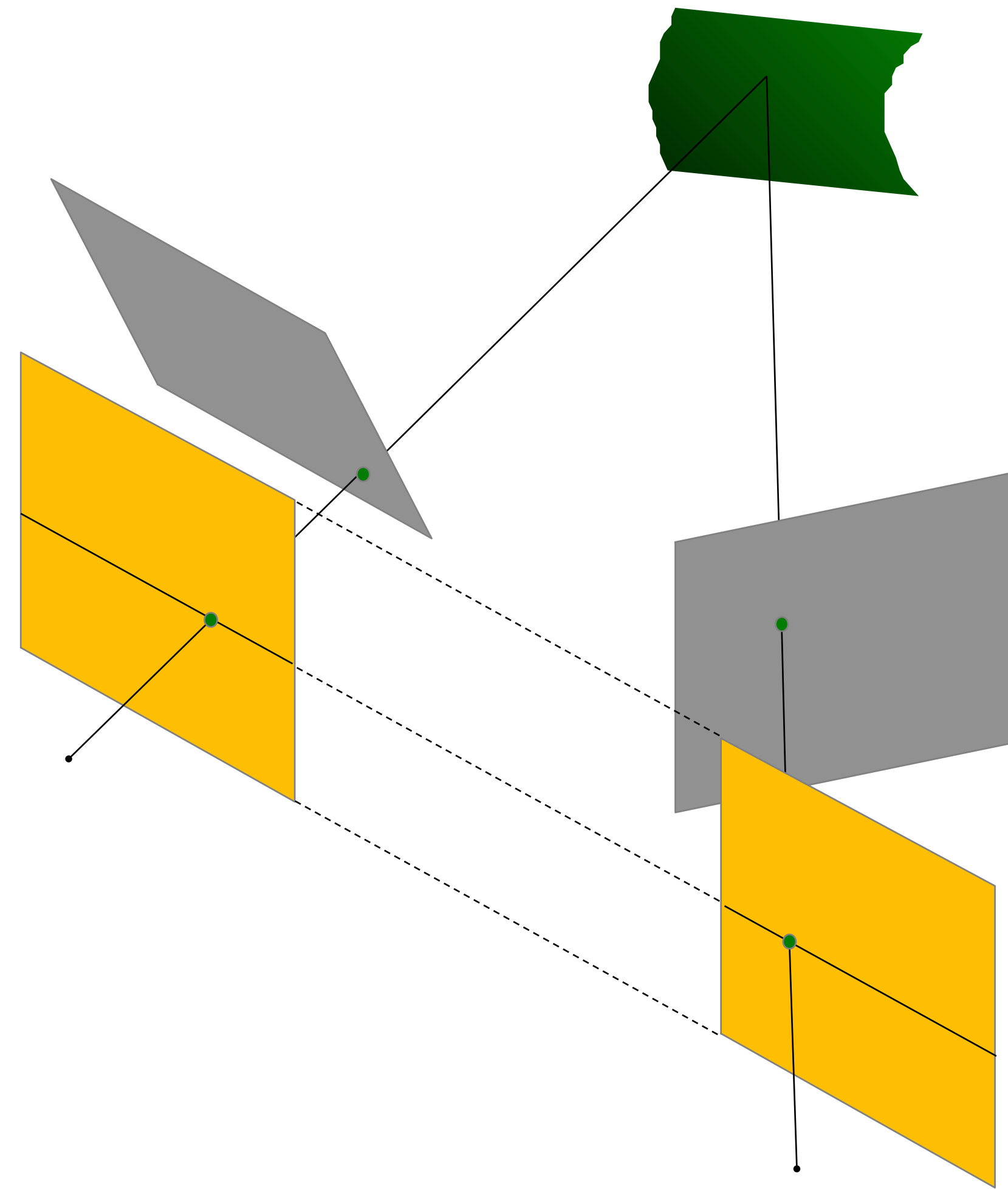
# Rectified Stereo Pair



# 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.



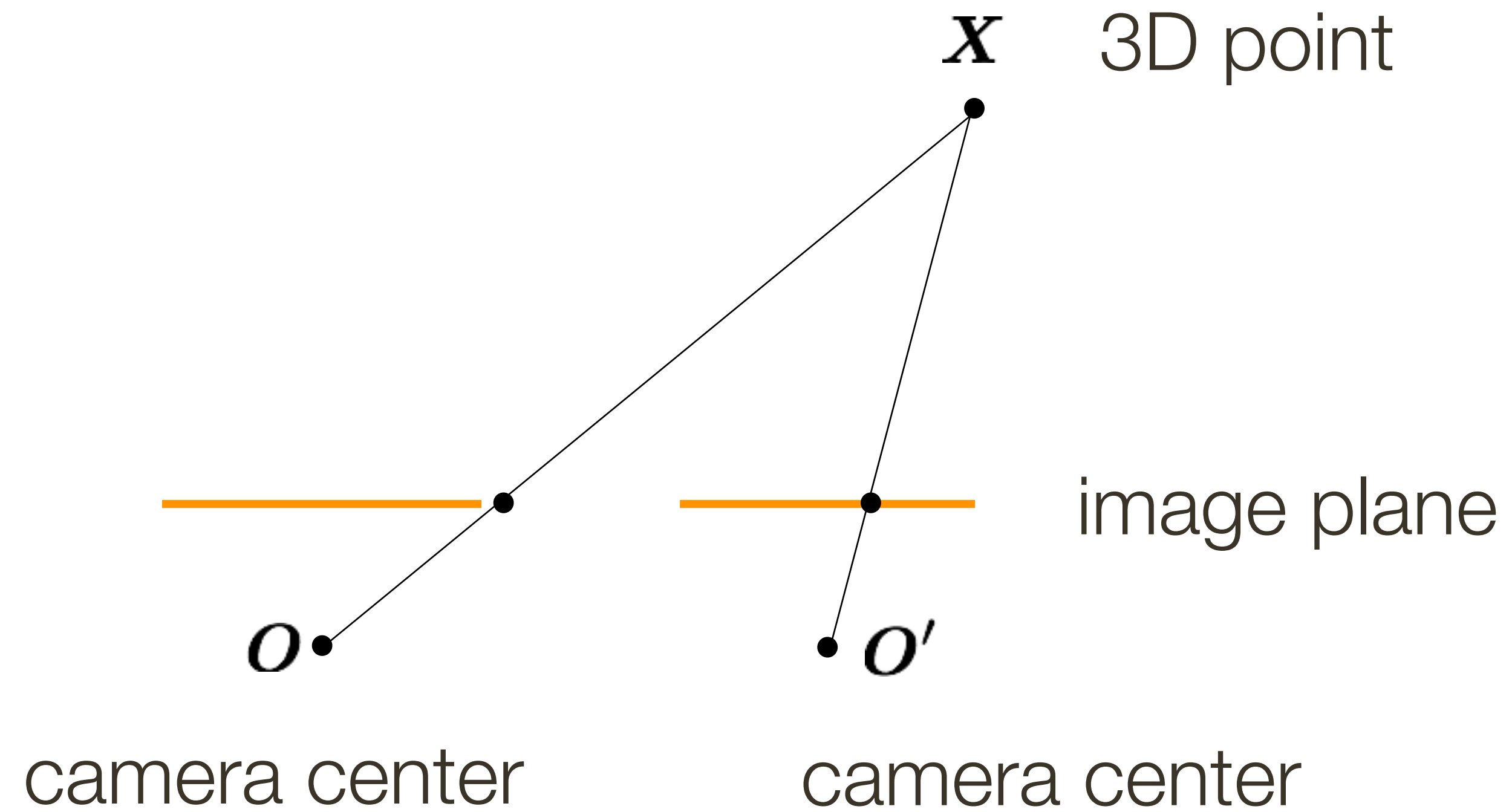
# Rectified Stereo Pair: Example

Before Rectification

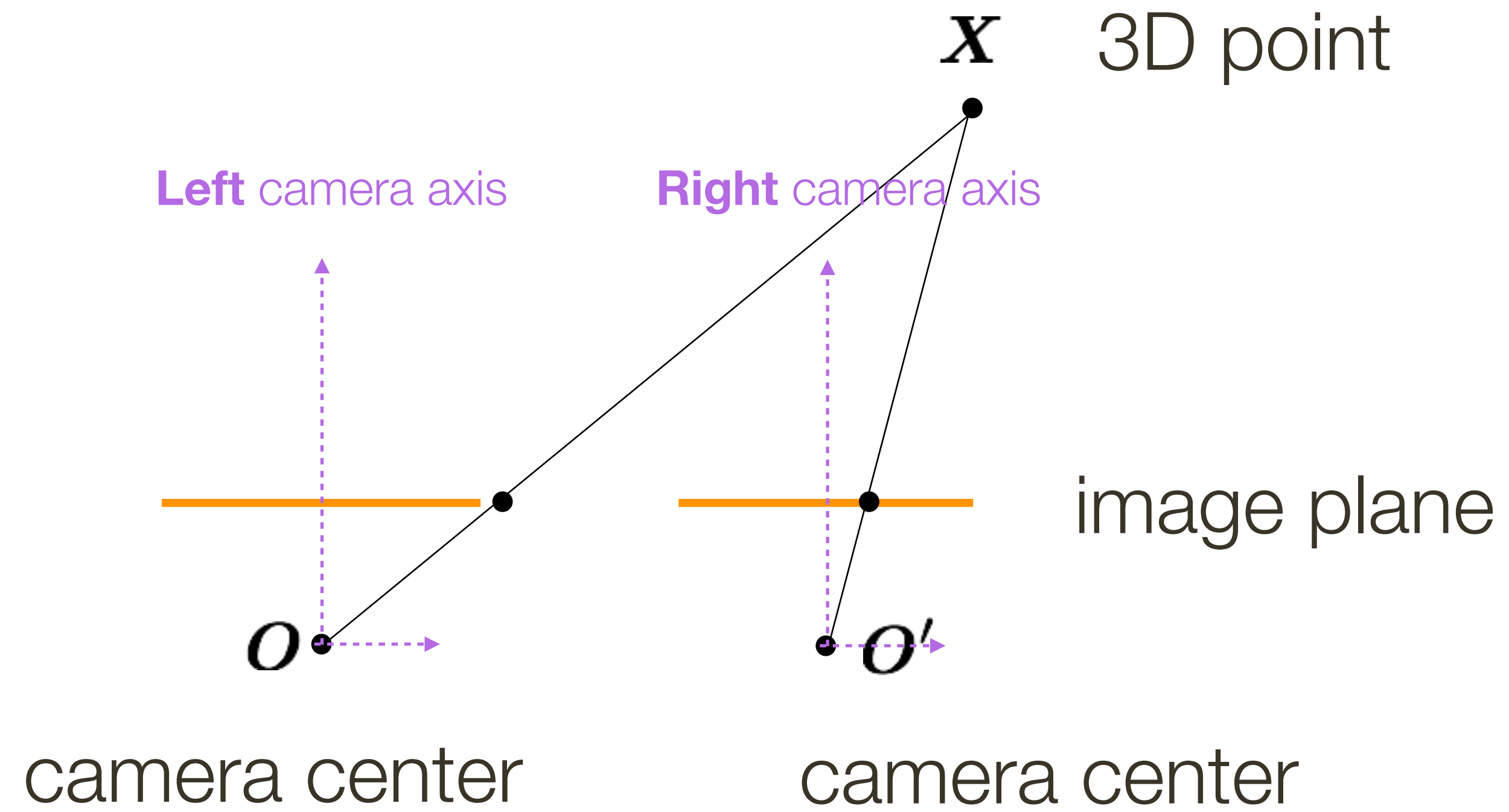


After Rectification

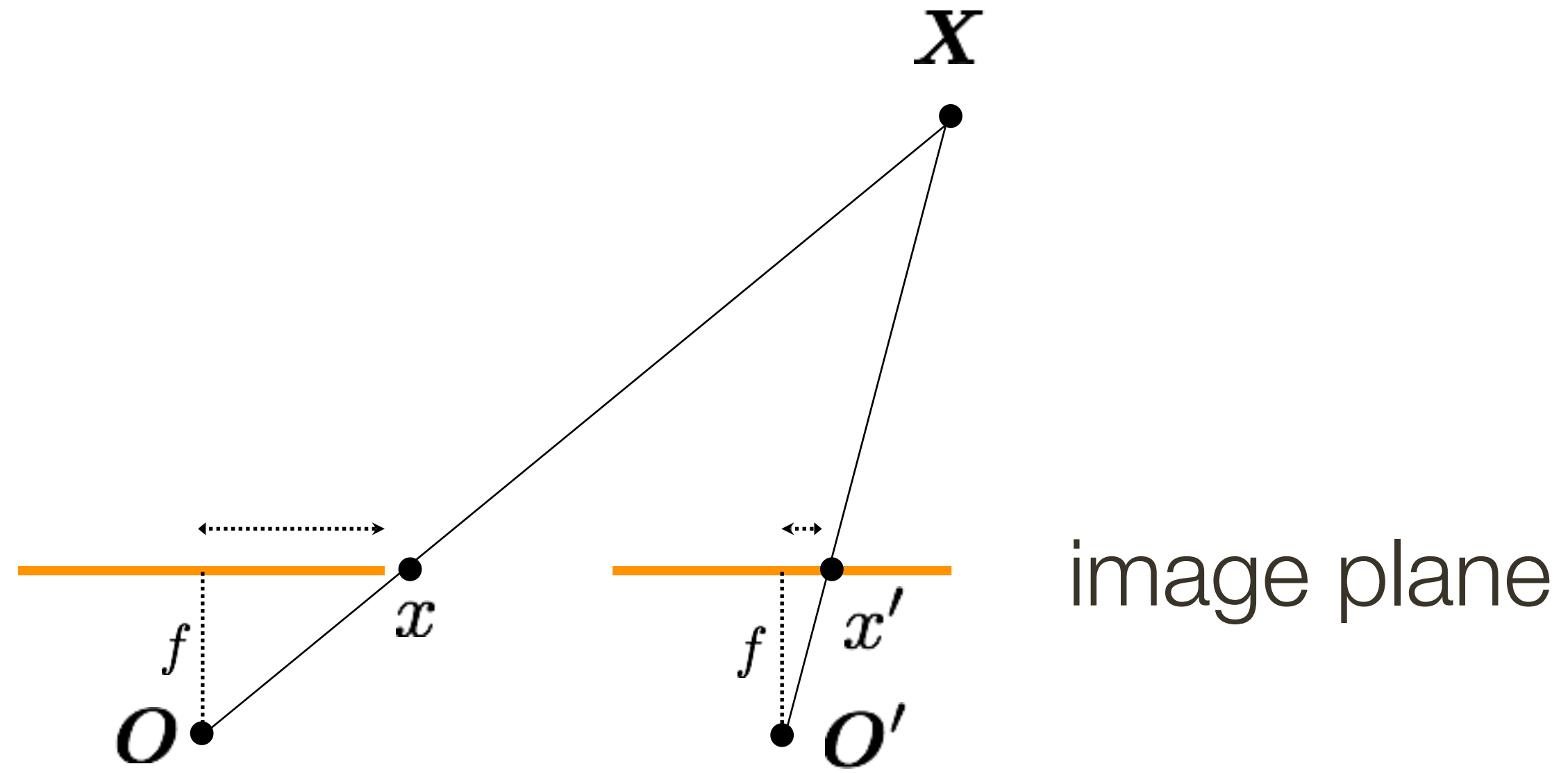
# Rectified Stereo Pair: Depth Estimate



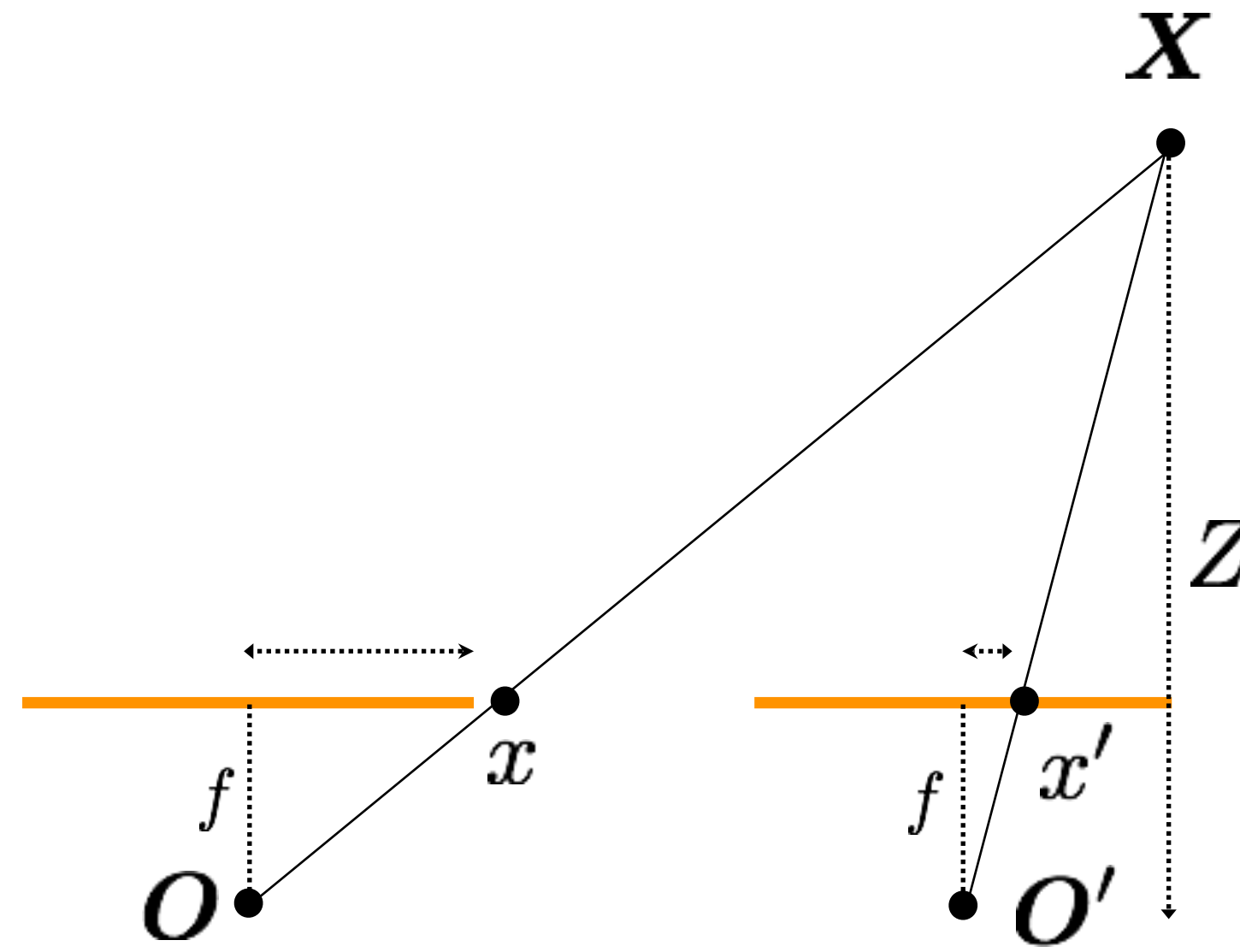
# Rectified Stereo Pair: Depth Estimate



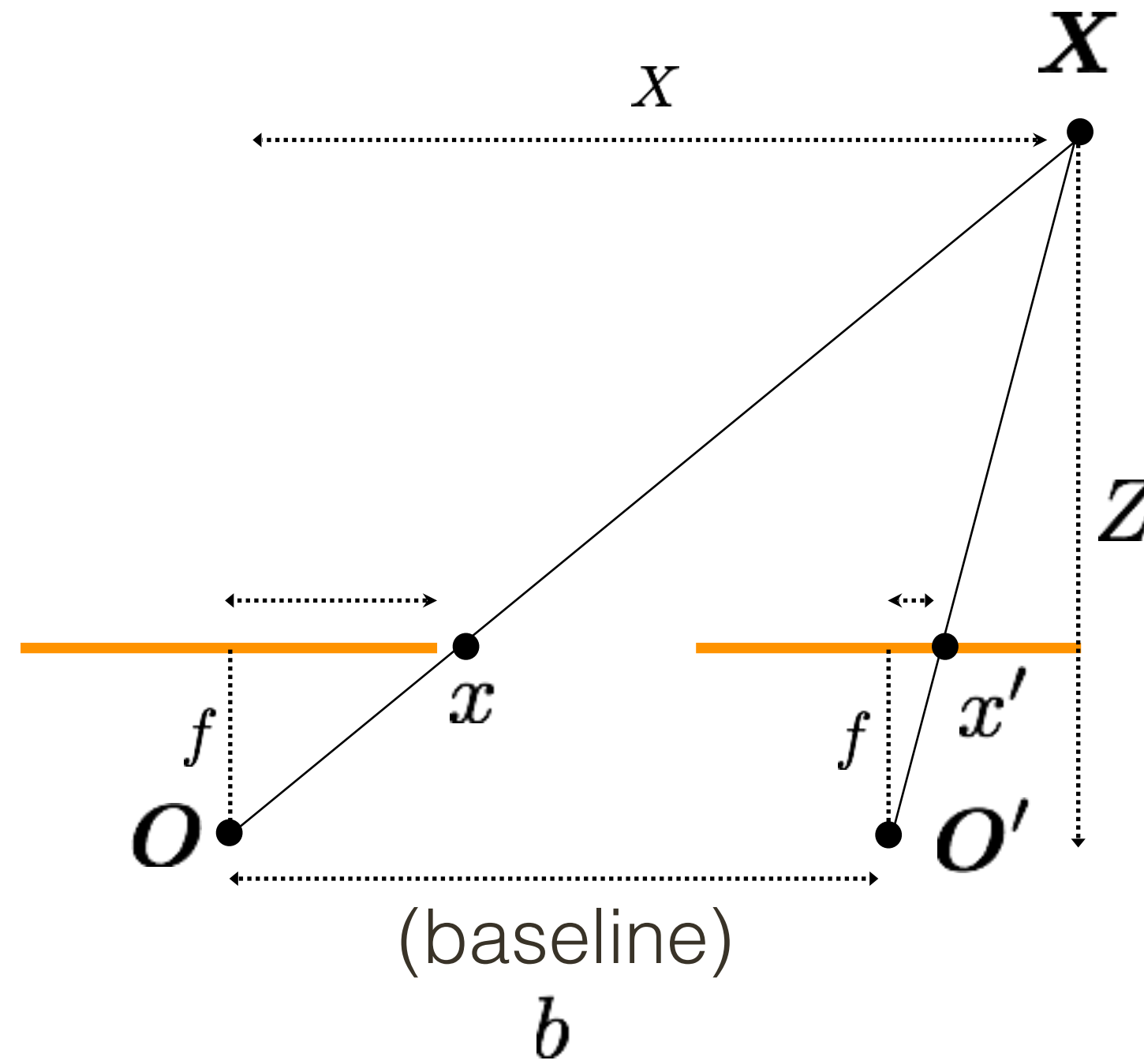
# Rectified Stereo Pair: Depth Estimate



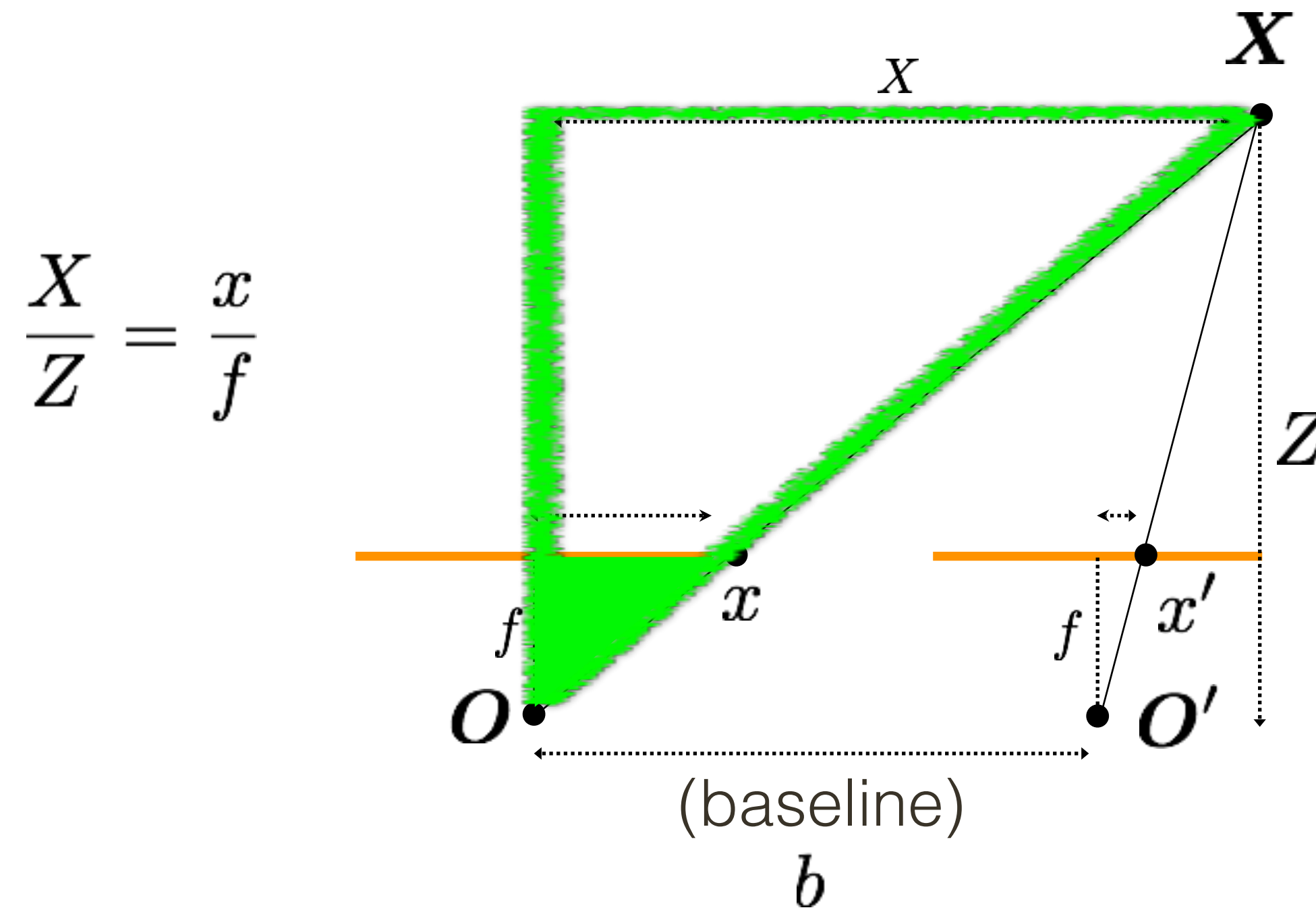
# Rectified Stereo Pair: Depth Estimate



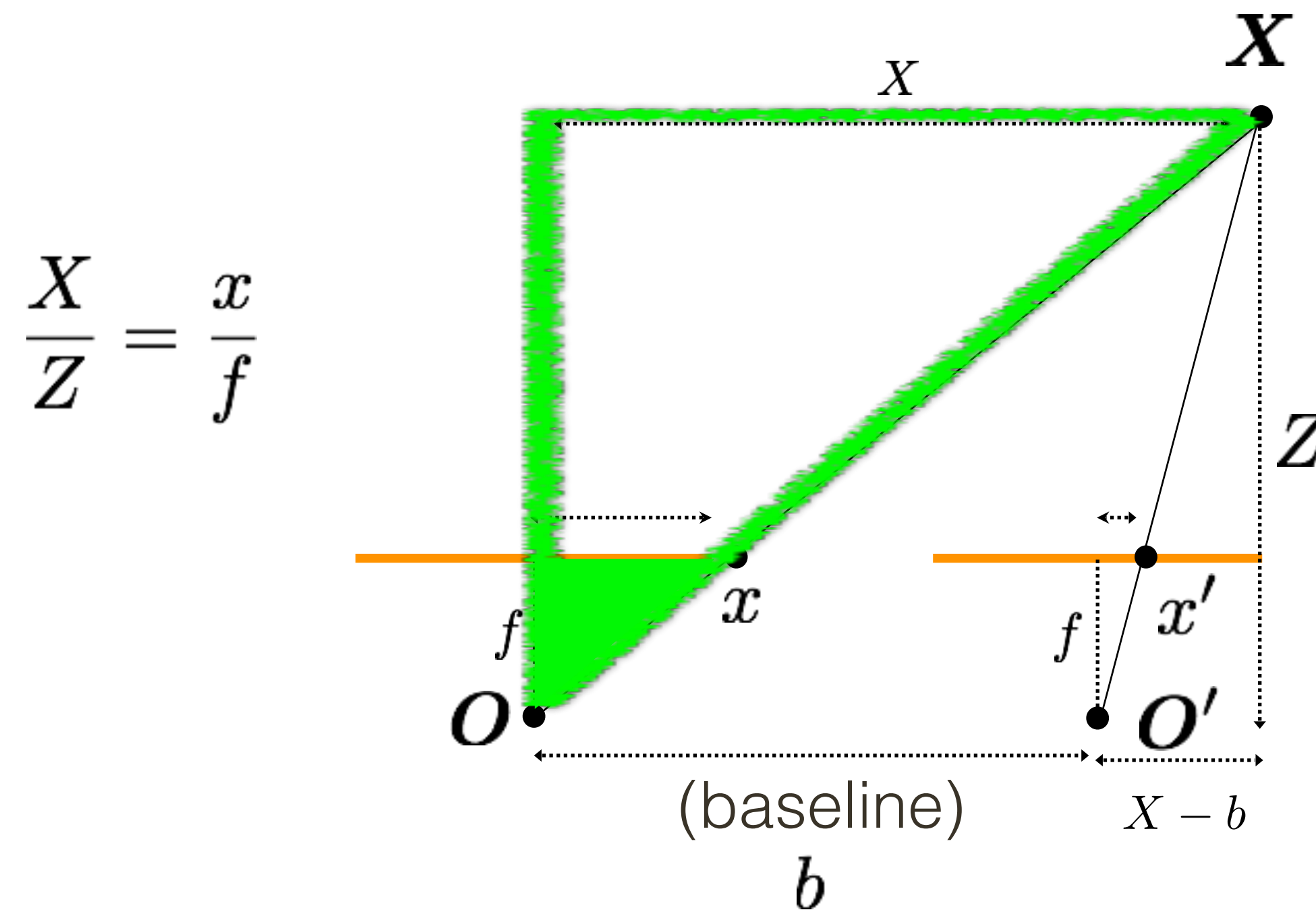
# Rectified Stereo Pair: Depth Estimate



# Rectified Stereo Pair: Depth Estimate

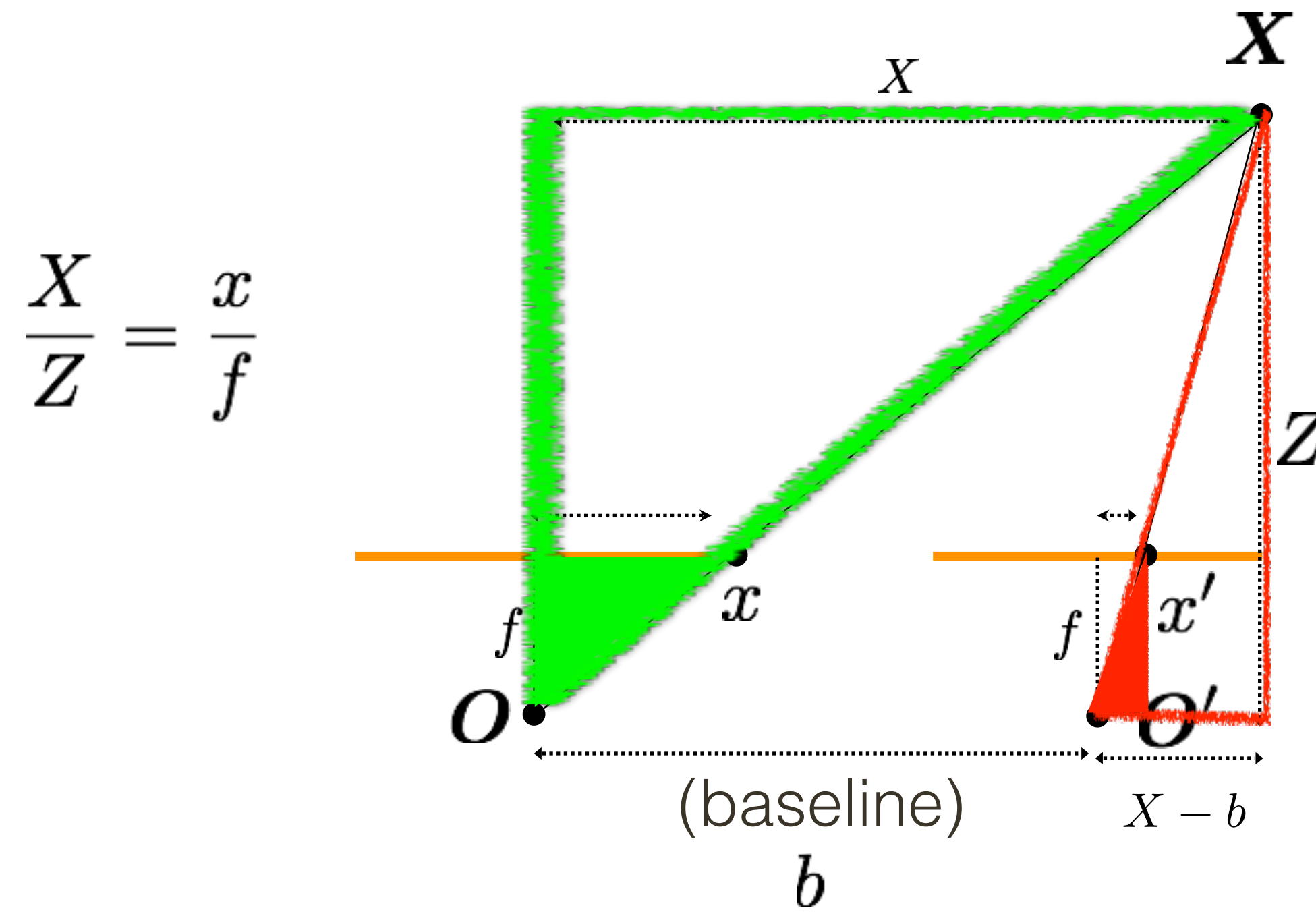


# Rectified Stereo Pair: Depth Estimate





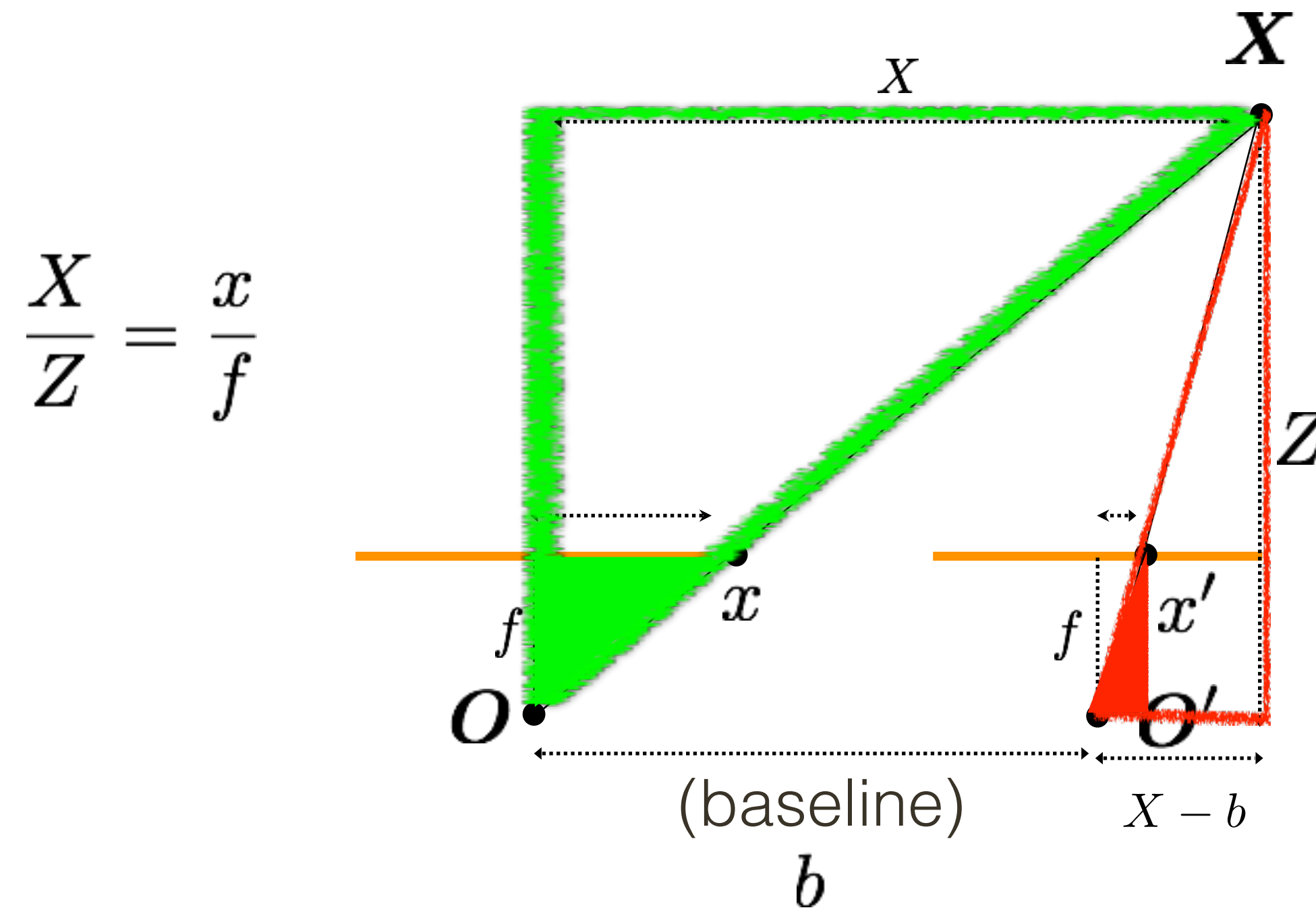
# Rectified Stereo Pair: Depth Estimate



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

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

# Rectified Stereo Pair: Depth Estimate

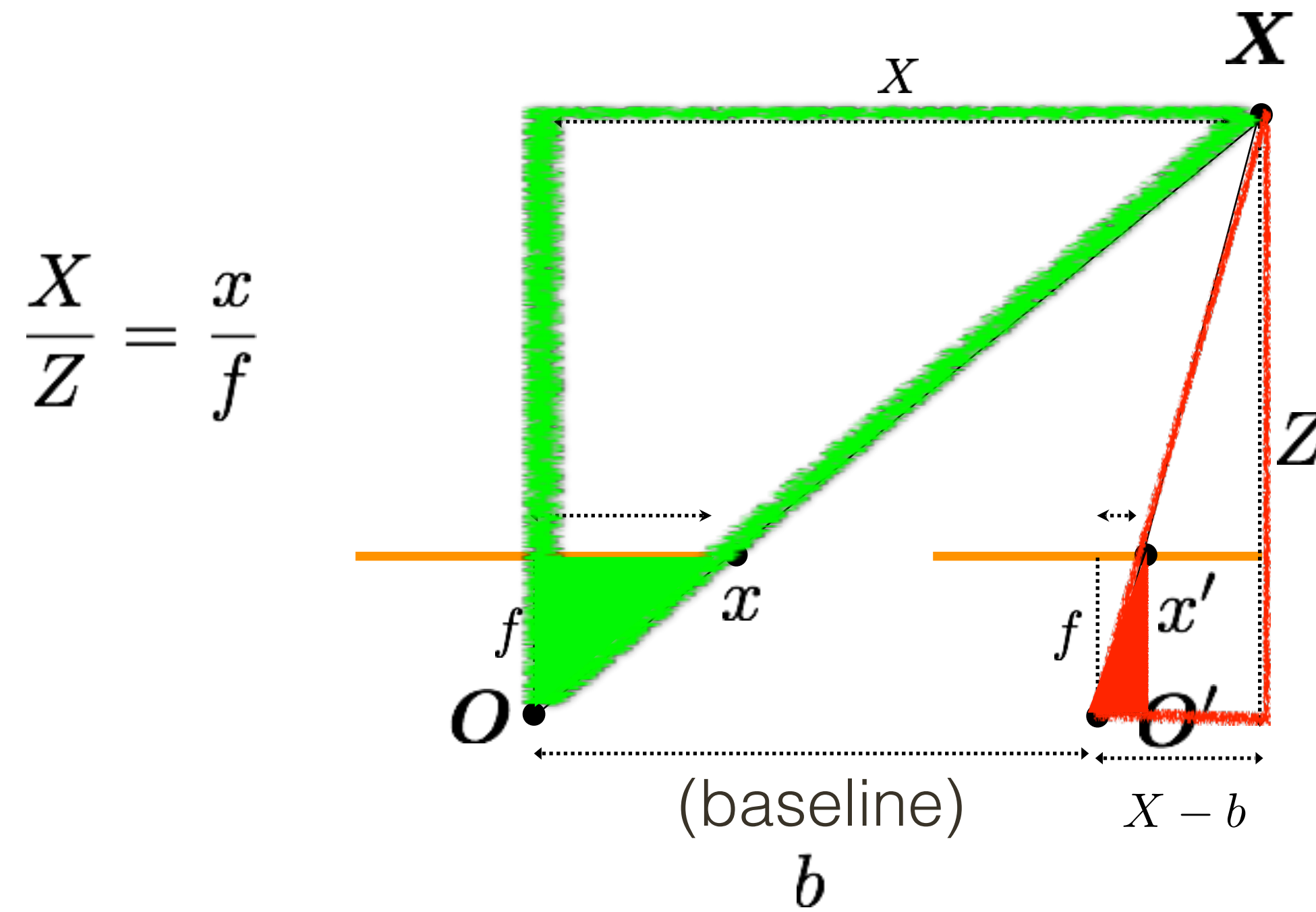


$$\frac{X}{Z} = \frac{x}{f}$$

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

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

# Rectified Stereo Pair: Depth Estimate



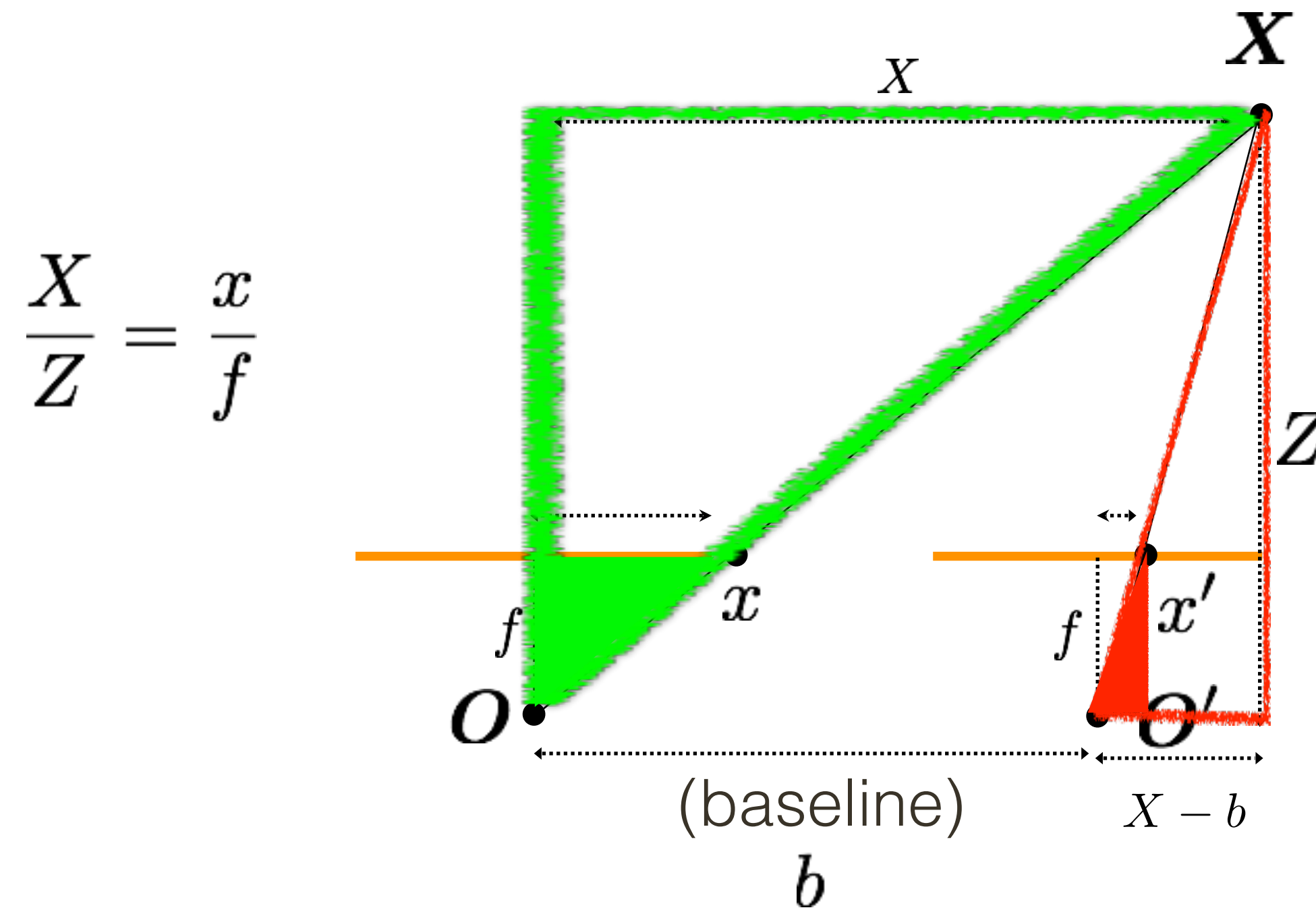
$$\frac{X}{Z} = \frac{x}{f}$$

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

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

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

# Rectified Stereo Pair: Depth Estimate



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

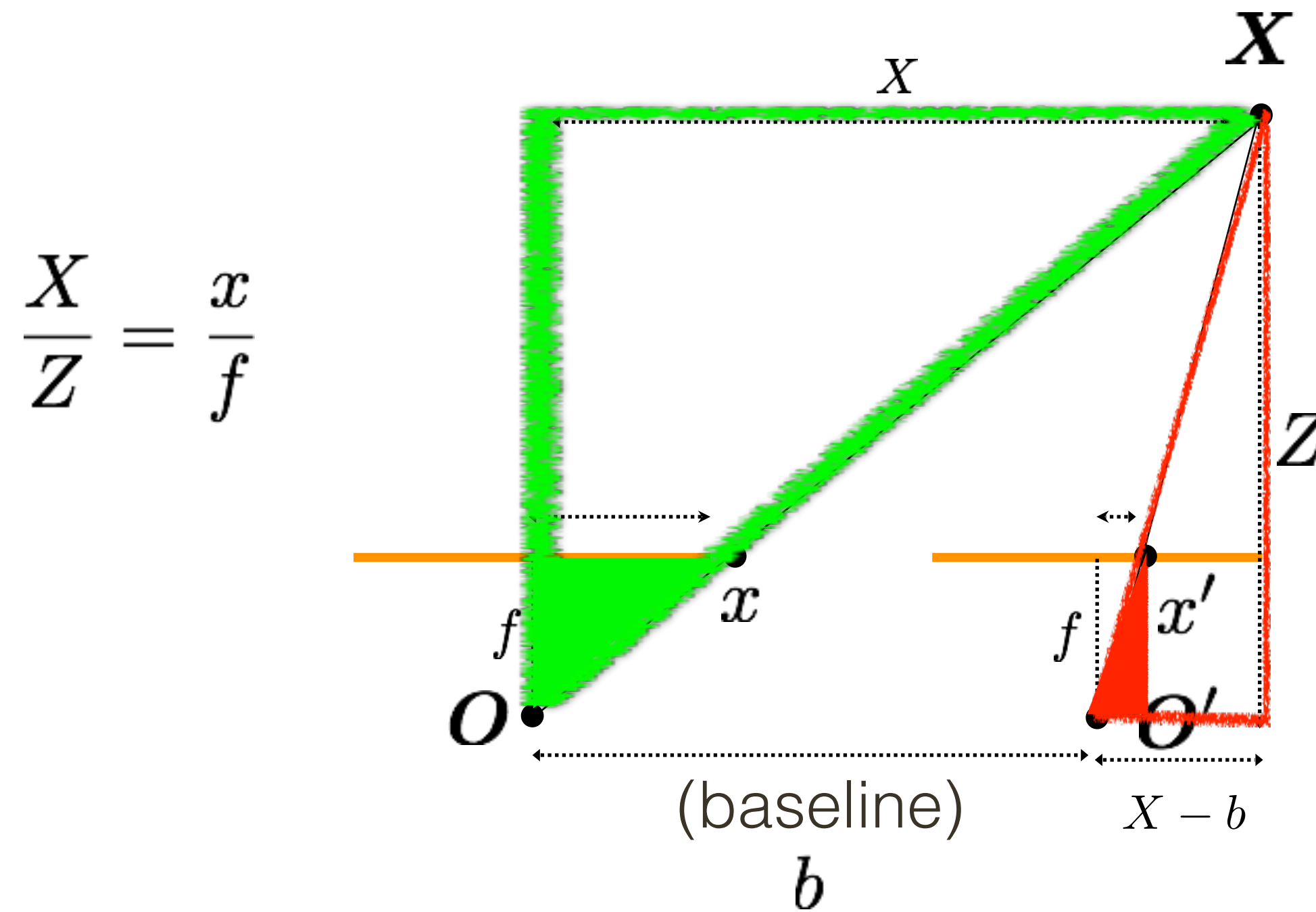
$$\frac{X - b}{Z} = \frac{x'}{f}$$

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

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

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

# Rectified Stereo Pair: Depth Estimate



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

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

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

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

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

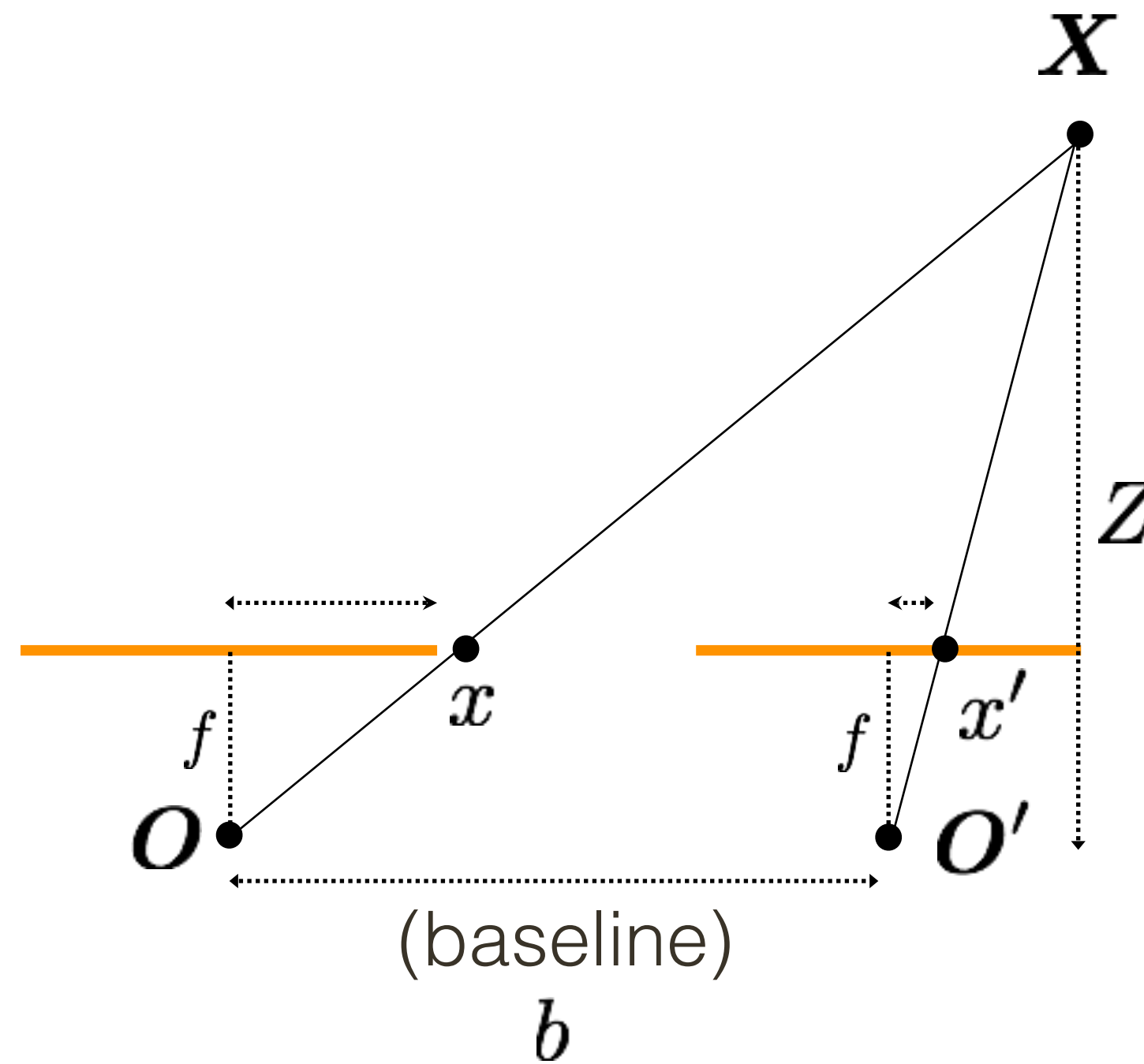
## Disparity

(wrt to camera origin of image plane)

$$d = x - x'$$

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

# Rectified Stereo Pair: Depth Estimate



## Disparity

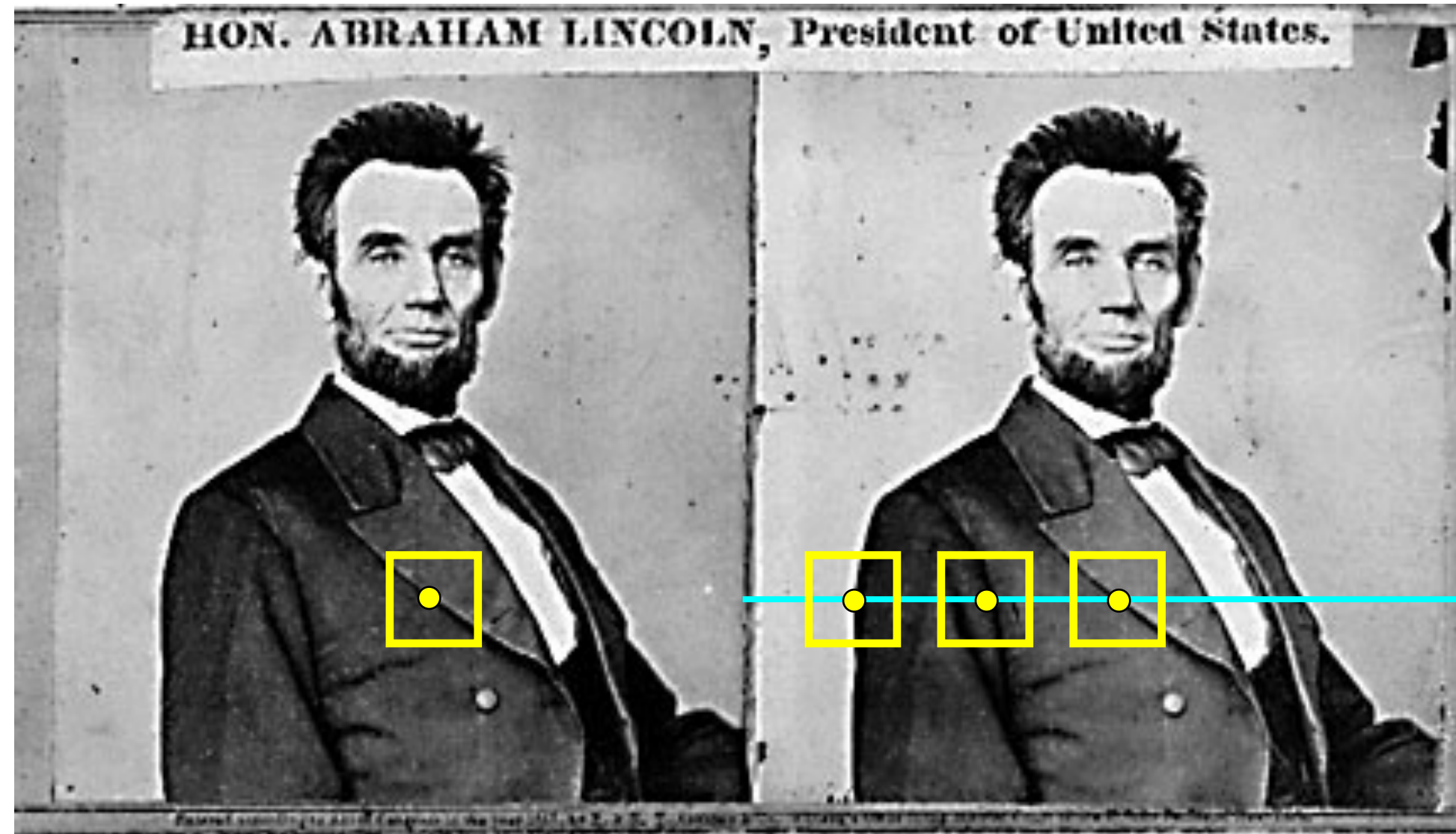
(wrt to camera origin of image plane)

$$d = x - x'$$

inversely proportional to depth

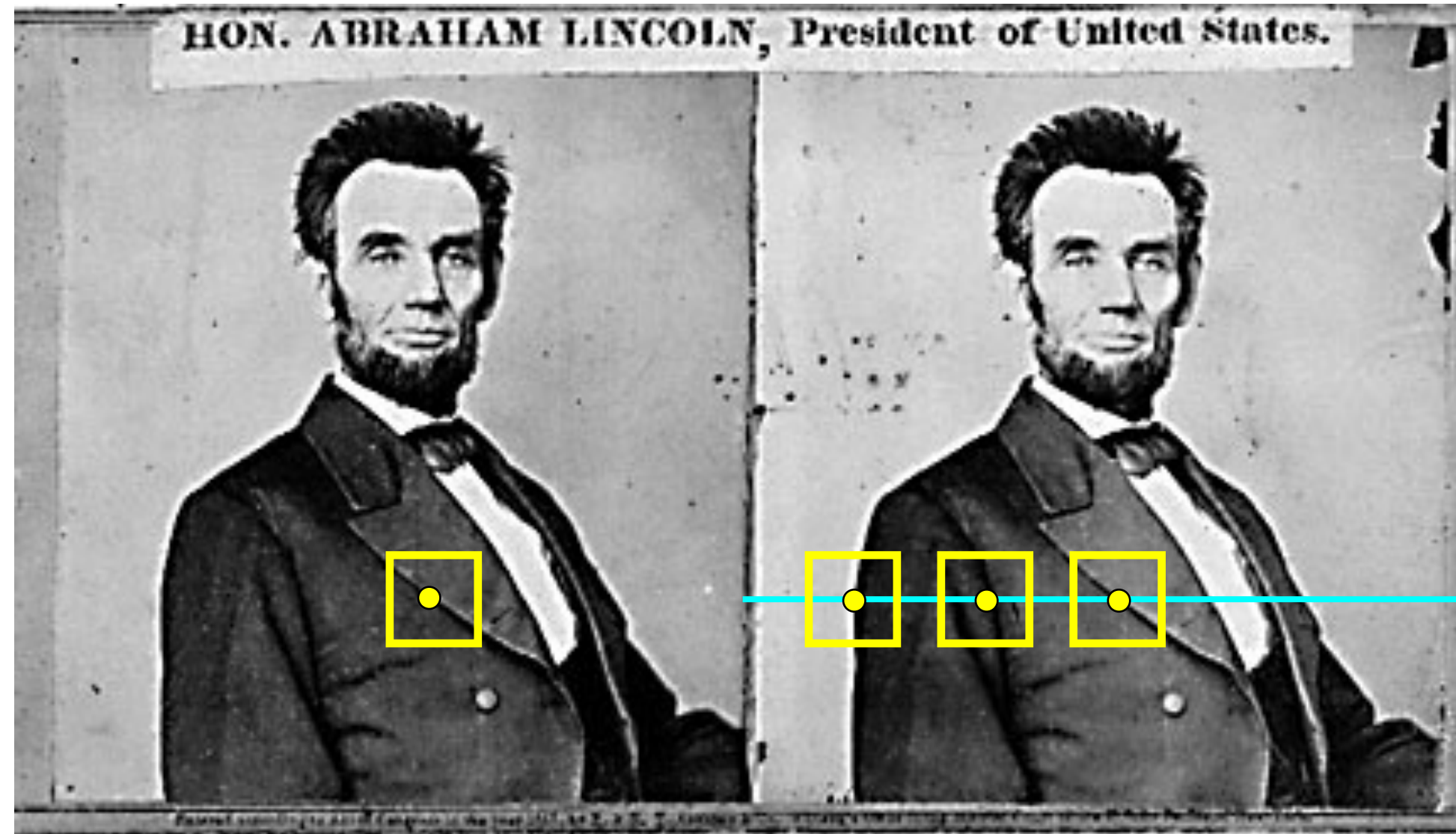
$$= \frac{bf}{Z}$$

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

# (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

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



# Correspondence: What should we match?

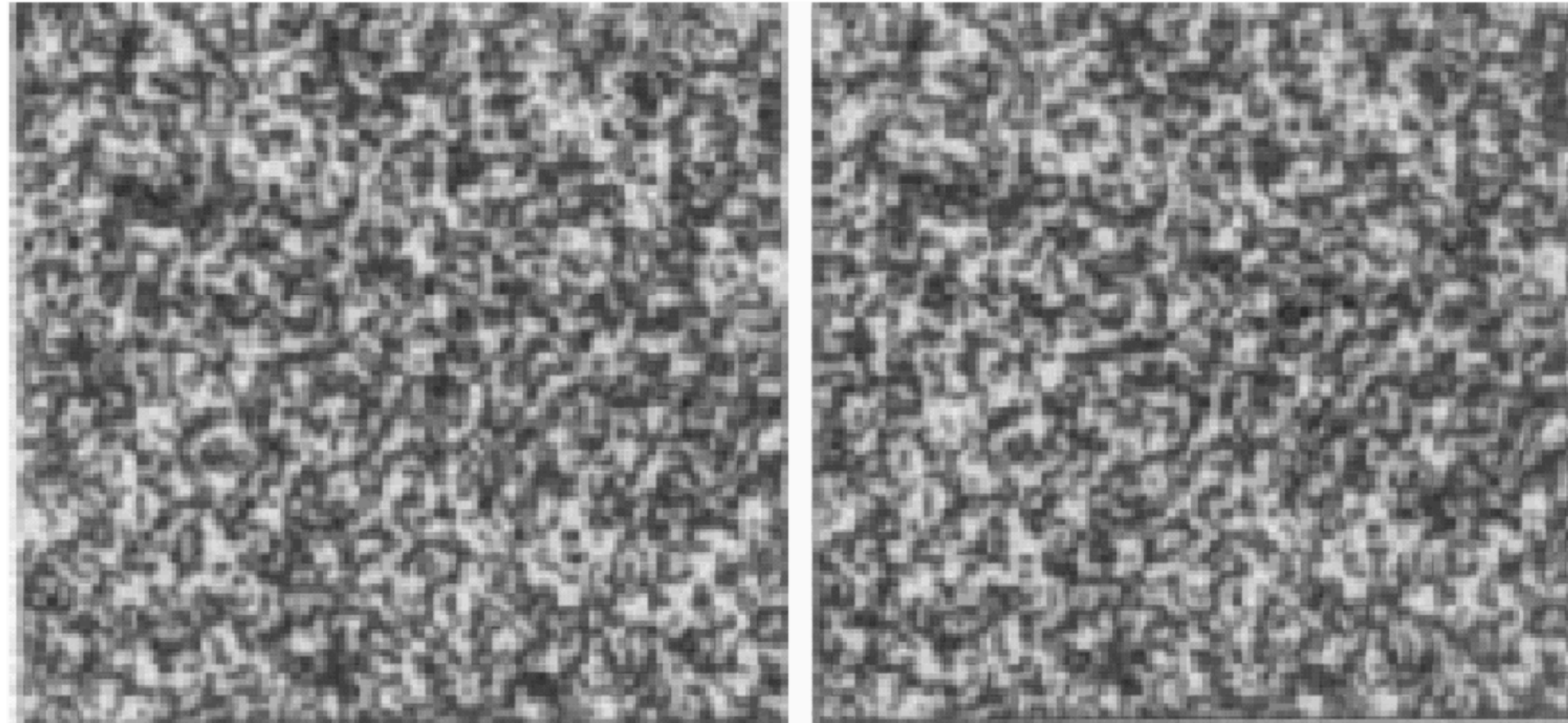
Objects?

Edges?

Pixels?

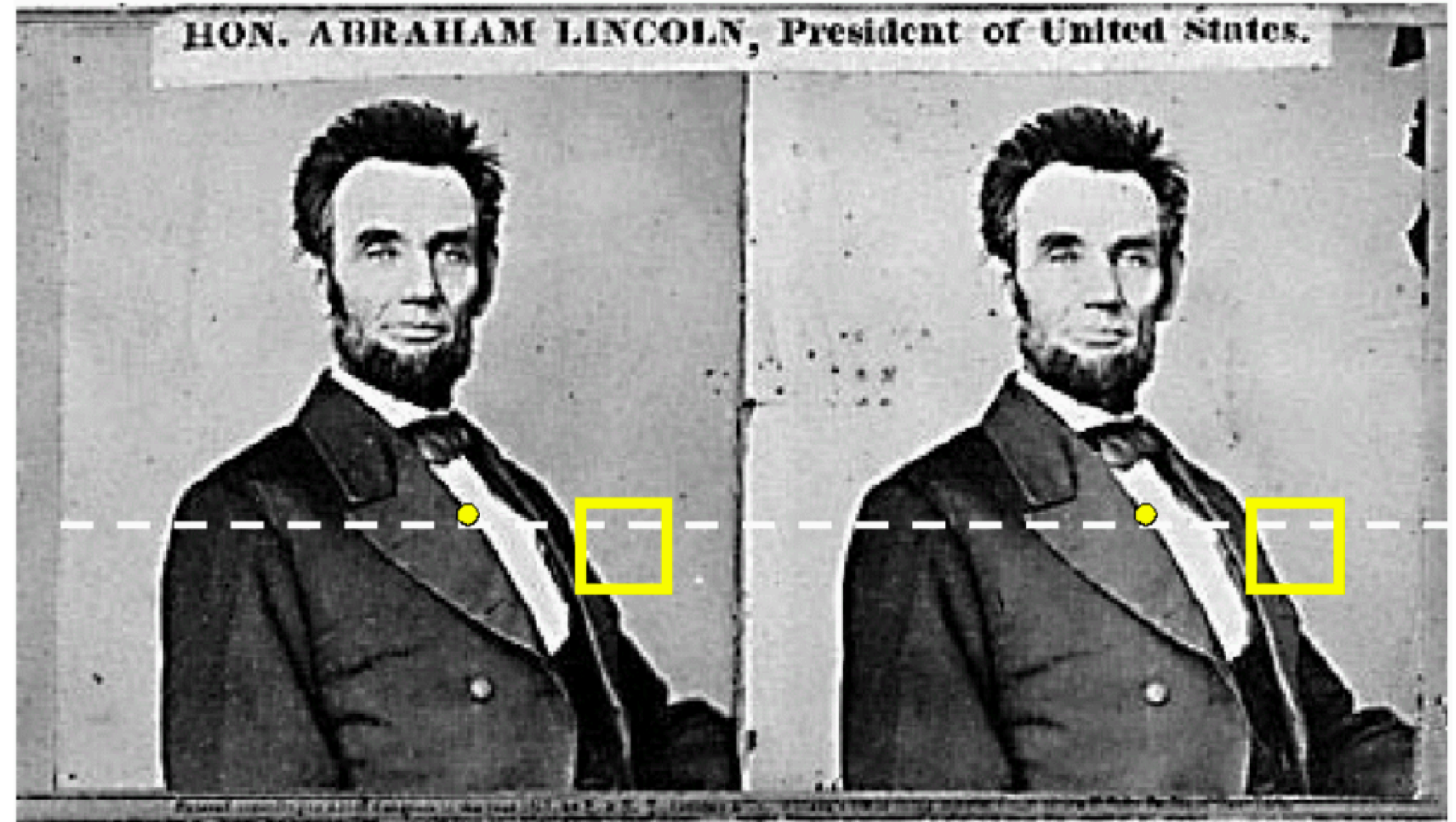
Collections of pixels?

# 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**

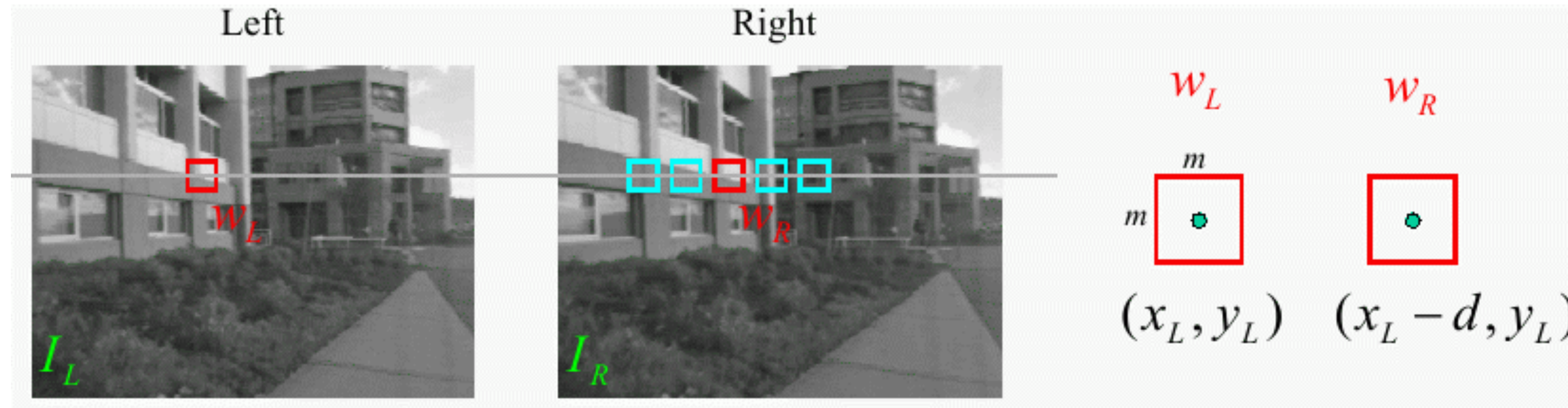
For each **pixel** in the left image

- 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

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

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

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

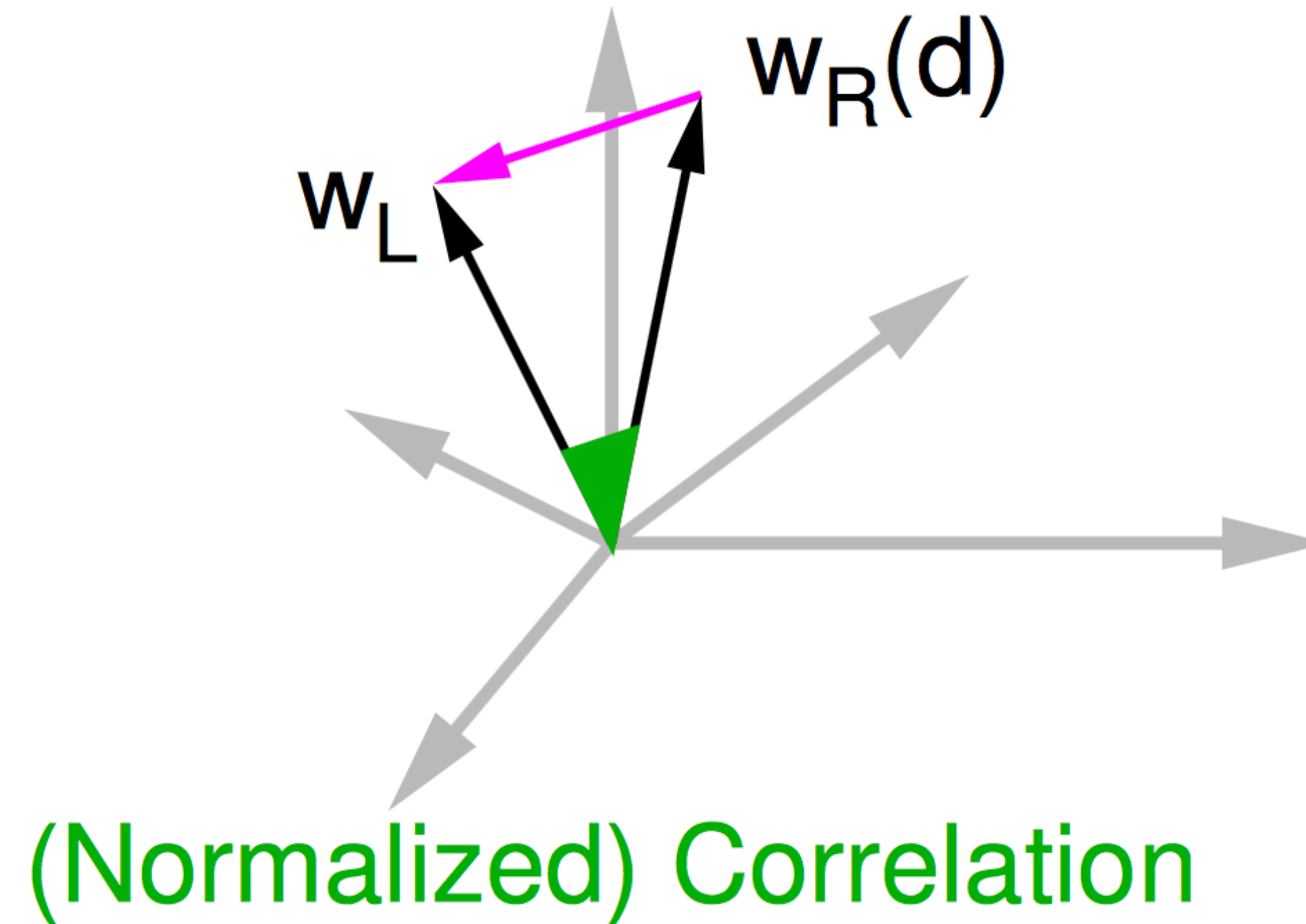
Window Magnitude

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

**Normalized Pixel:** subtract the mean, normalize to unit length

# Image Metrics

(Normalized) Sum of Squared Differences



# Image Metrics

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

**Sum of Squared Differences:**

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

(Normalized) **Correlation:**

$$\begin{aligned} C_{NC}(d) &= \sum_{(u,v) \in \mathbf{W}_m(x,y)} \hat{I}_L(u,v) \hat{I}_R(u-d,v) \\ &= \mathbf{w}_L \cdot \mathbf{w}_R(d) = \cos \theta \end{aligned}$$

# 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

Left

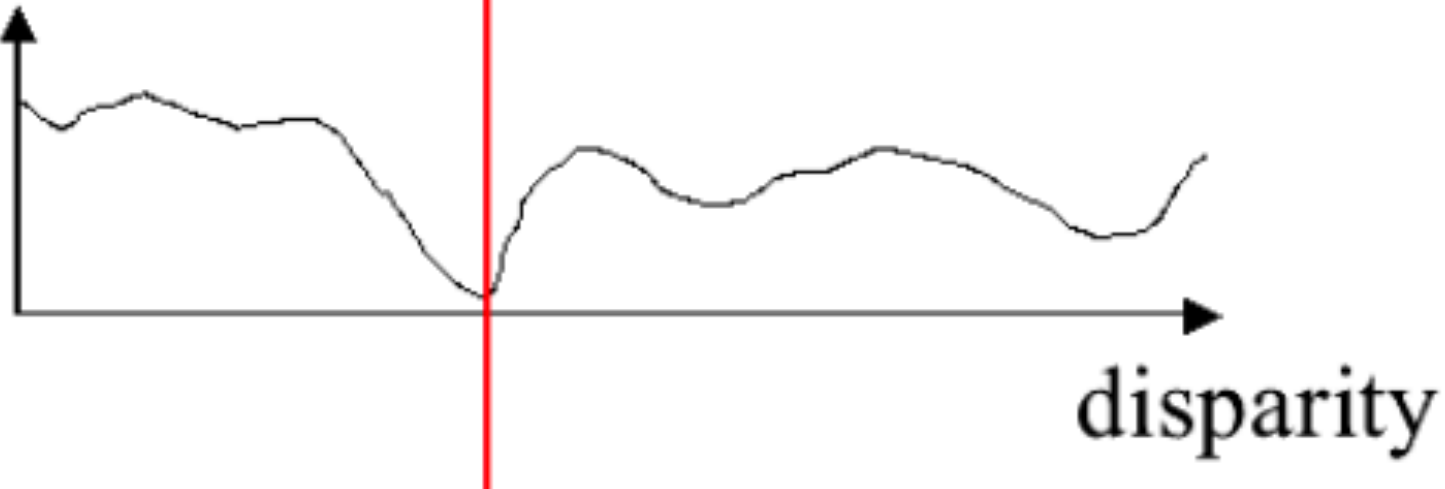


scanline

Right



SSD error



# Similarity Measure

Sum of Absolute Differences (SAD)

Sum of Squared Differences (SSD)

Zero-mean SAD

Locally scaled SAD

Normalized Cross Correlation (NCC)

# Formula

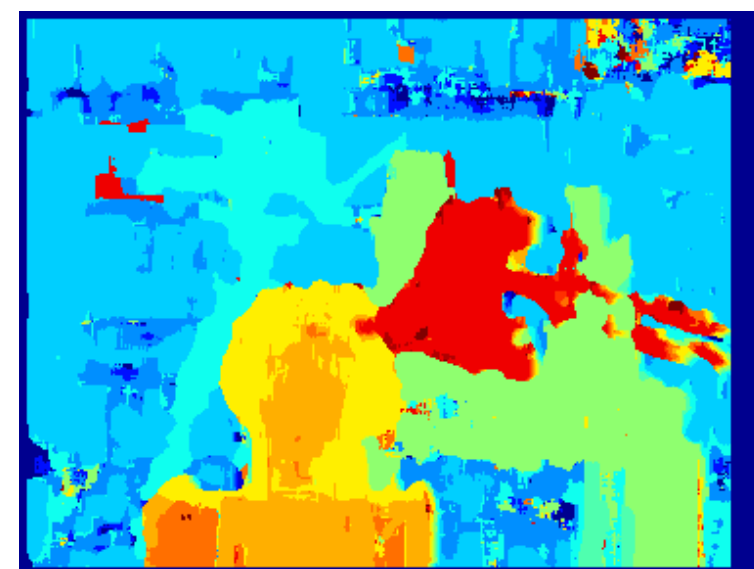
$$\sum_{(i,j) \in W} |I_1(i,j) - I_2(x+i, y+j)|$$

$$\sum_{(i,j) \in W} (I_1(i,j) - I_2(x+i, y+j))^2$$

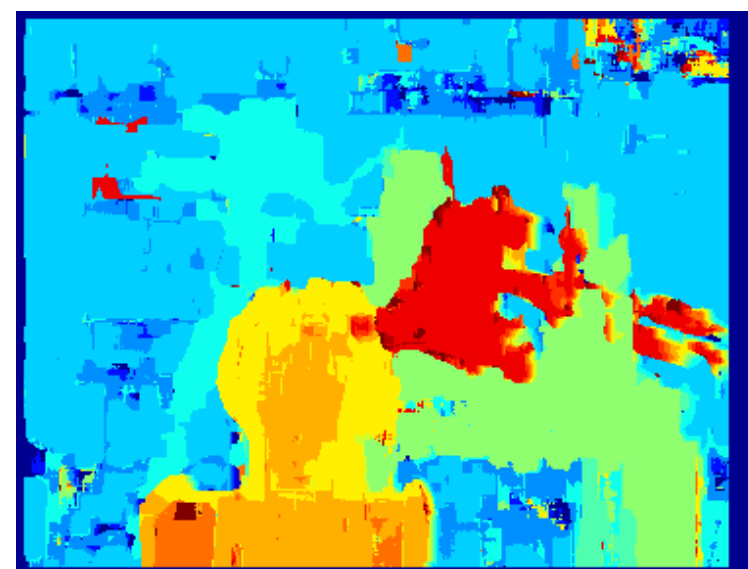
$$\sum_{(i,j) \in W} |I_1(i,j) - \bar{I}_1(i,j) - I_2(x+i, y+j) + \bar{I}_2(x+i, y+j)|$$

$$\sum_{(i,j) \in W} |I_1(i,j) - \frac{\bar{I}_1(i,j)}{\bar{I}_2(x+i, y+j)} I_2(x+i, y+j)|$$

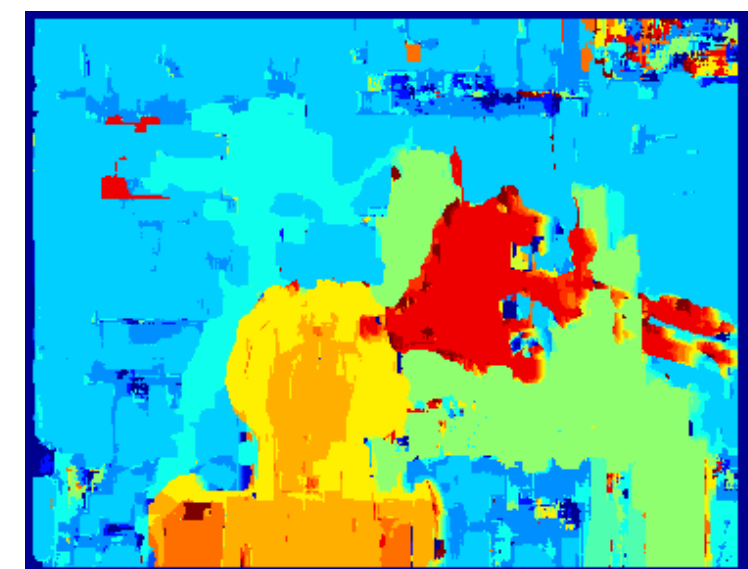
$$\frac{\sum_{(i,j) \in W} I_1(i,j) \cdot I_2(x+i, y+j)}{\sqrt{\sum_{(i,j) \in W} I_1^2(i,j) \cdot \sum_{(i,j) \in W} I_2^2(x+i, y+j)}}$$



SAD



SSD



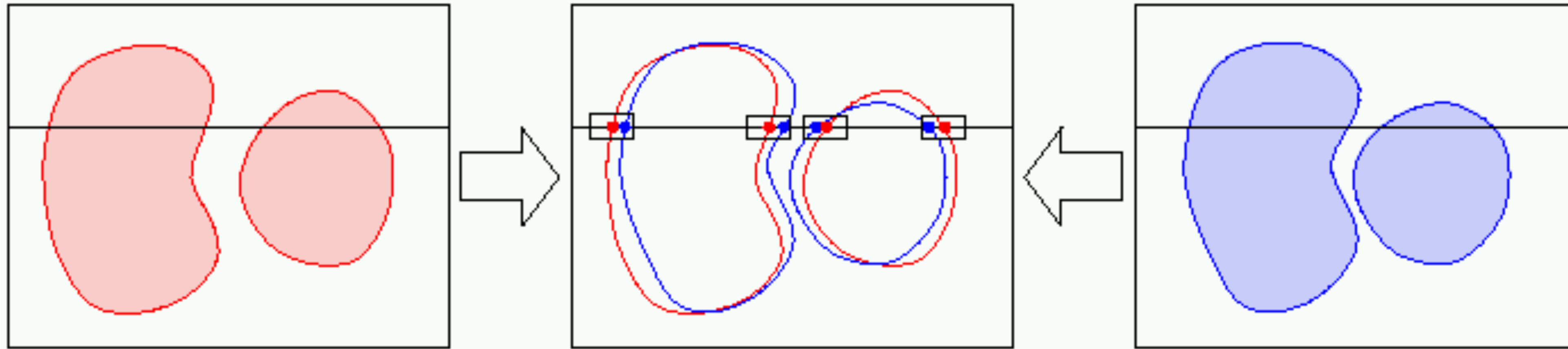
NCC



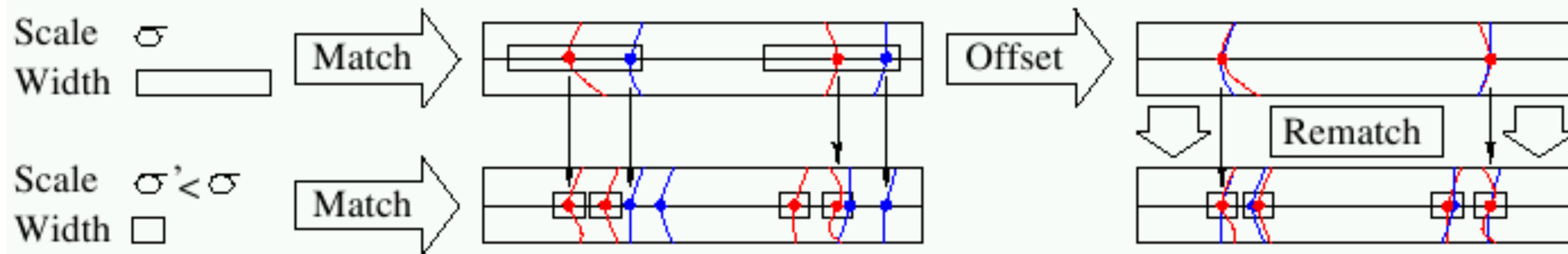
Ground truth

# Method: Edges

Matching zero-crossings at a single scale



Matching zero-crossings at multiple scales



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  $\sigma$ , 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”

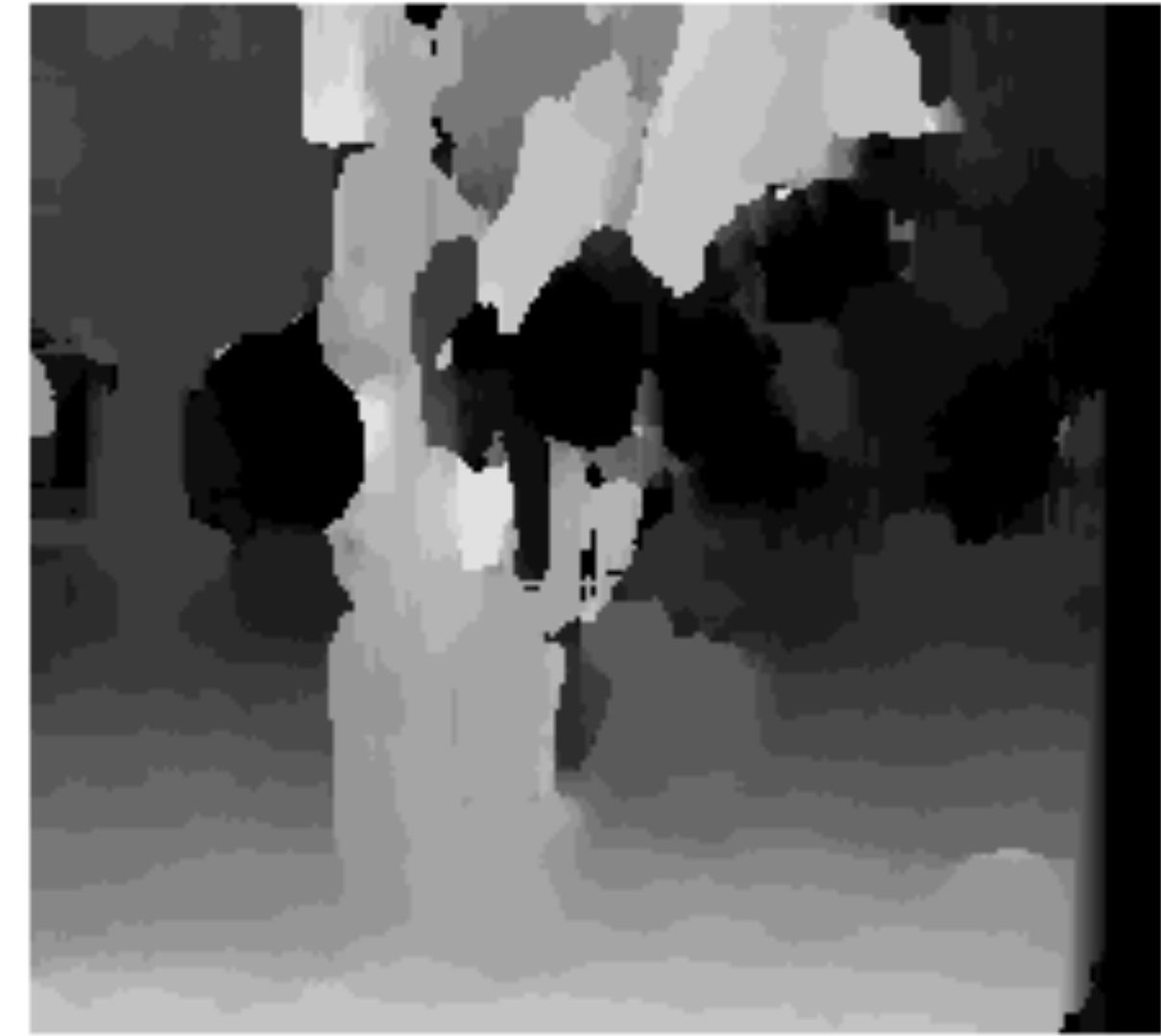
# Effect of **Window Size**



$W = 3$

**Smaller** window

- + More detail
- More noise



$W = 20$

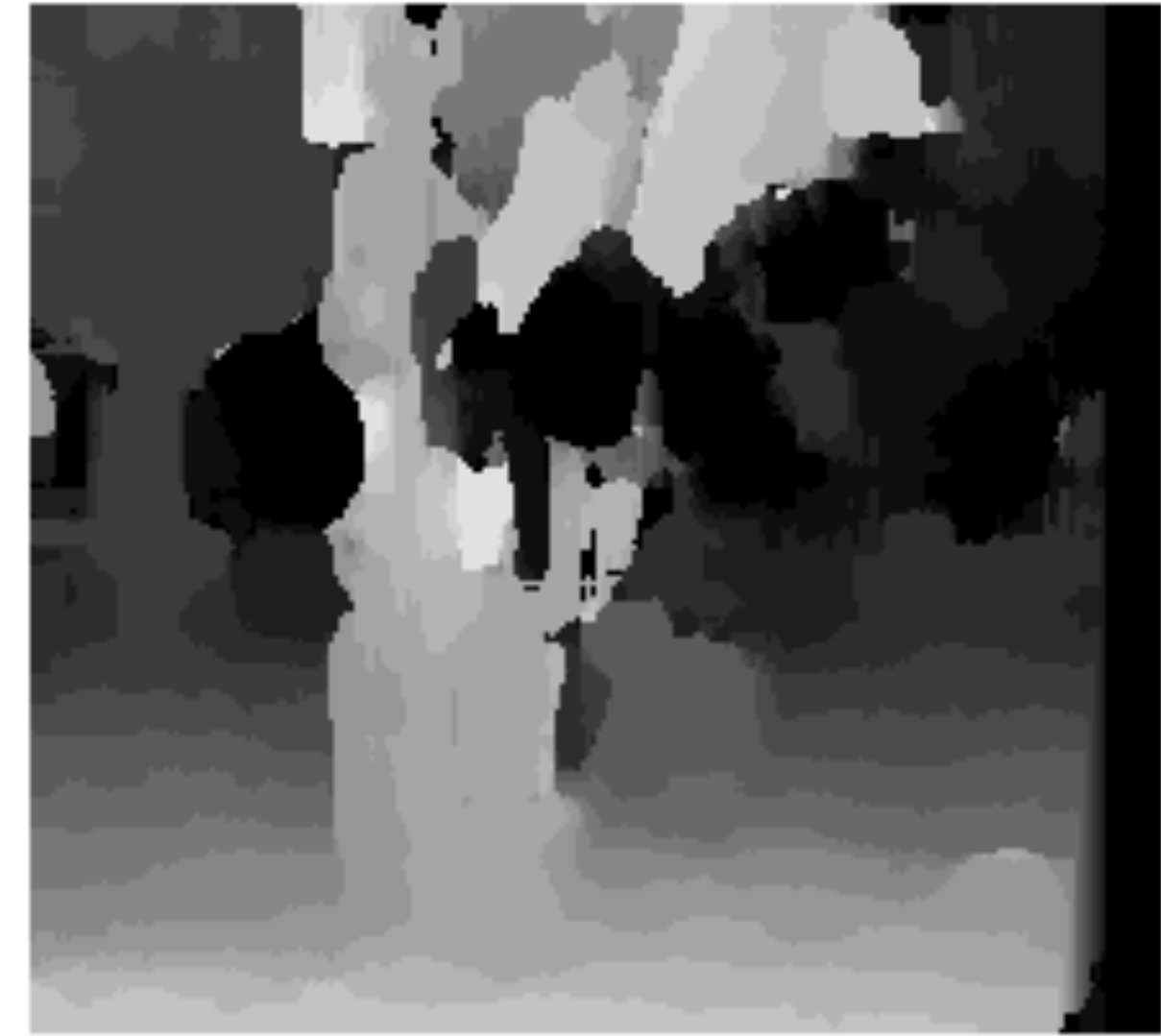
**Larger** window

- + Smoother disparity maps
- Less detail
- Fails near boundaries

# Effect of **Window Size**



$W = 3$

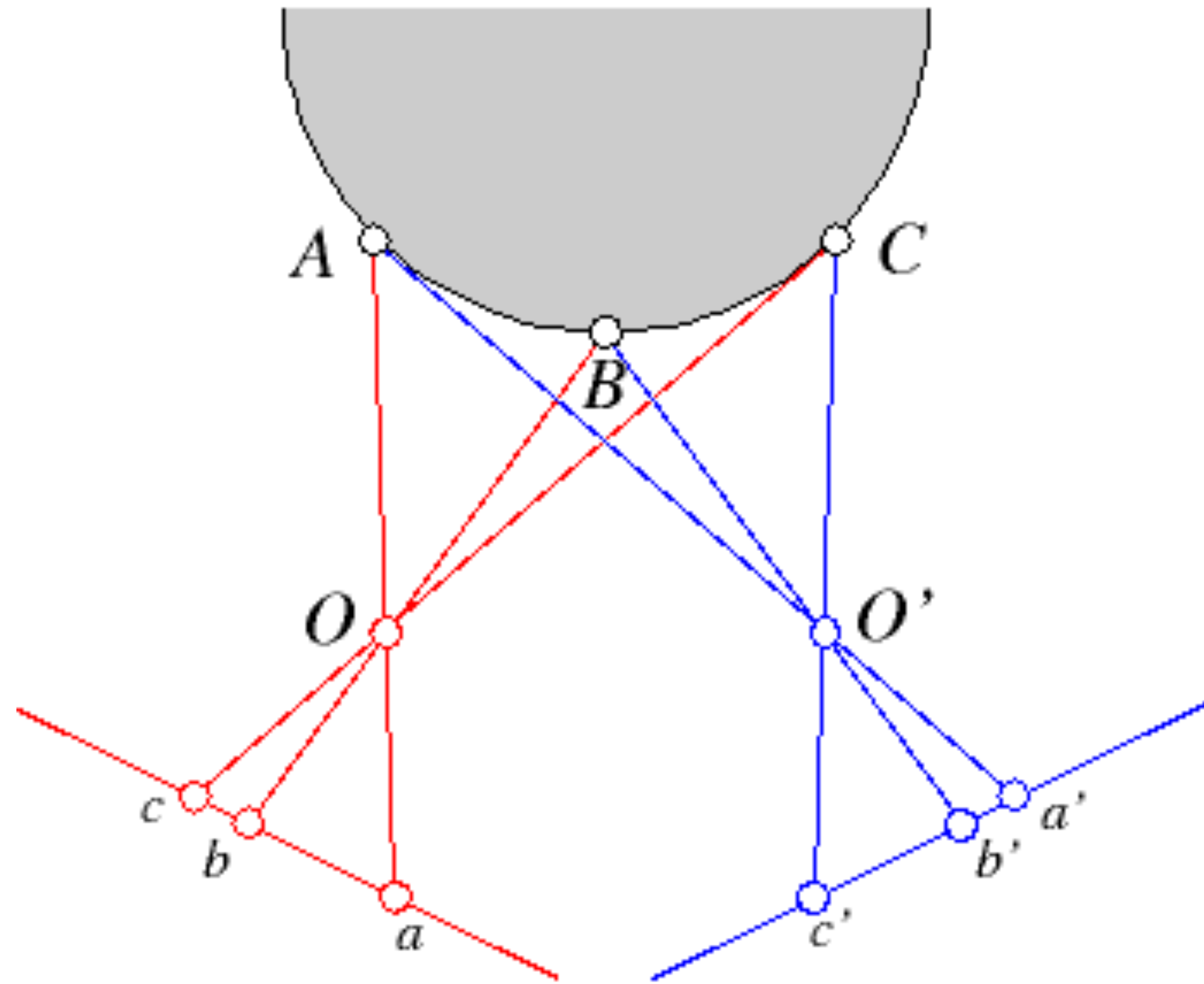


$W = 20$

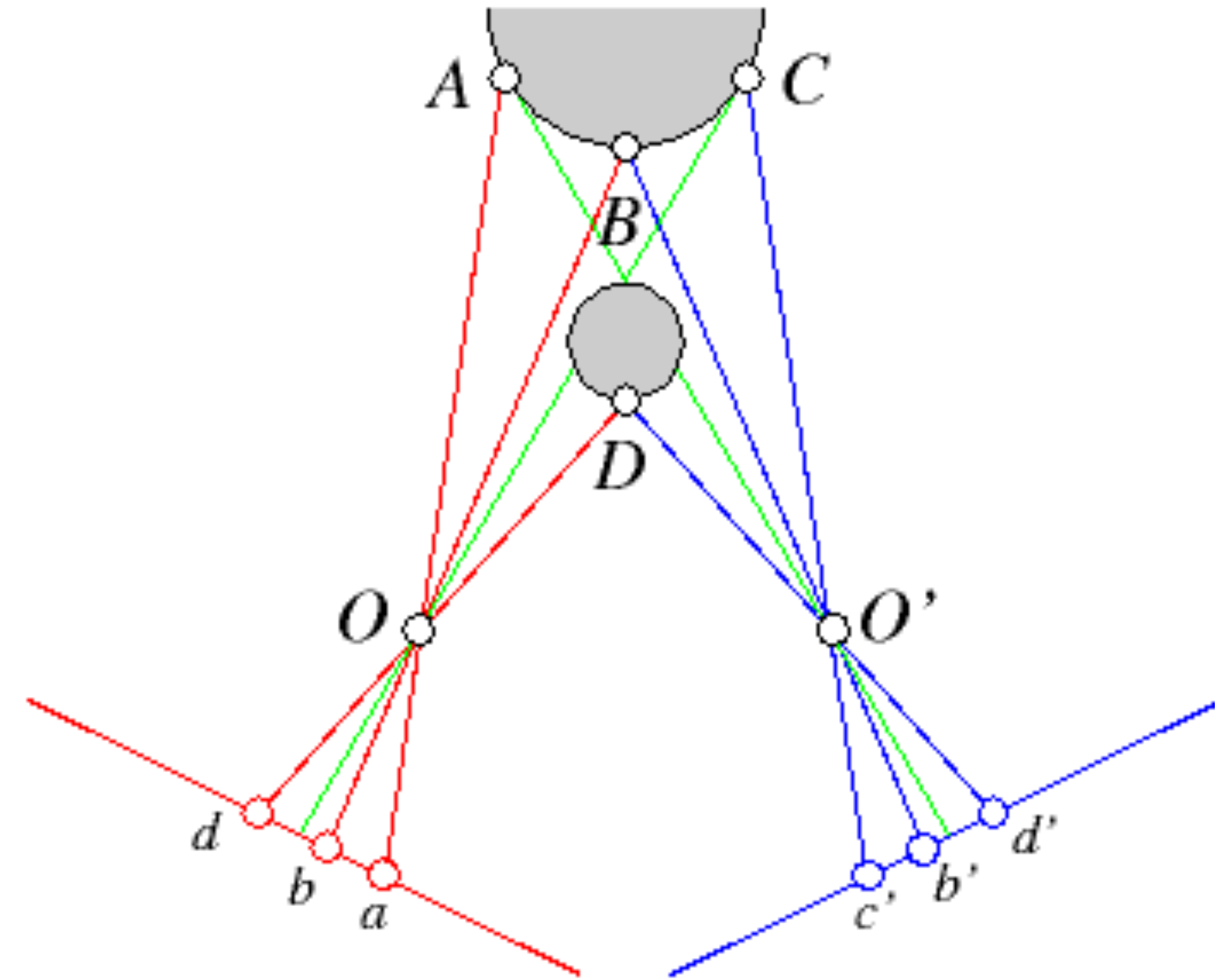
**Note:** Some approaches use an adaptive window size  
— try multiple sizes and select best match

# Ordering Constraints

Ordering constraint ...



.... and a **failure** case



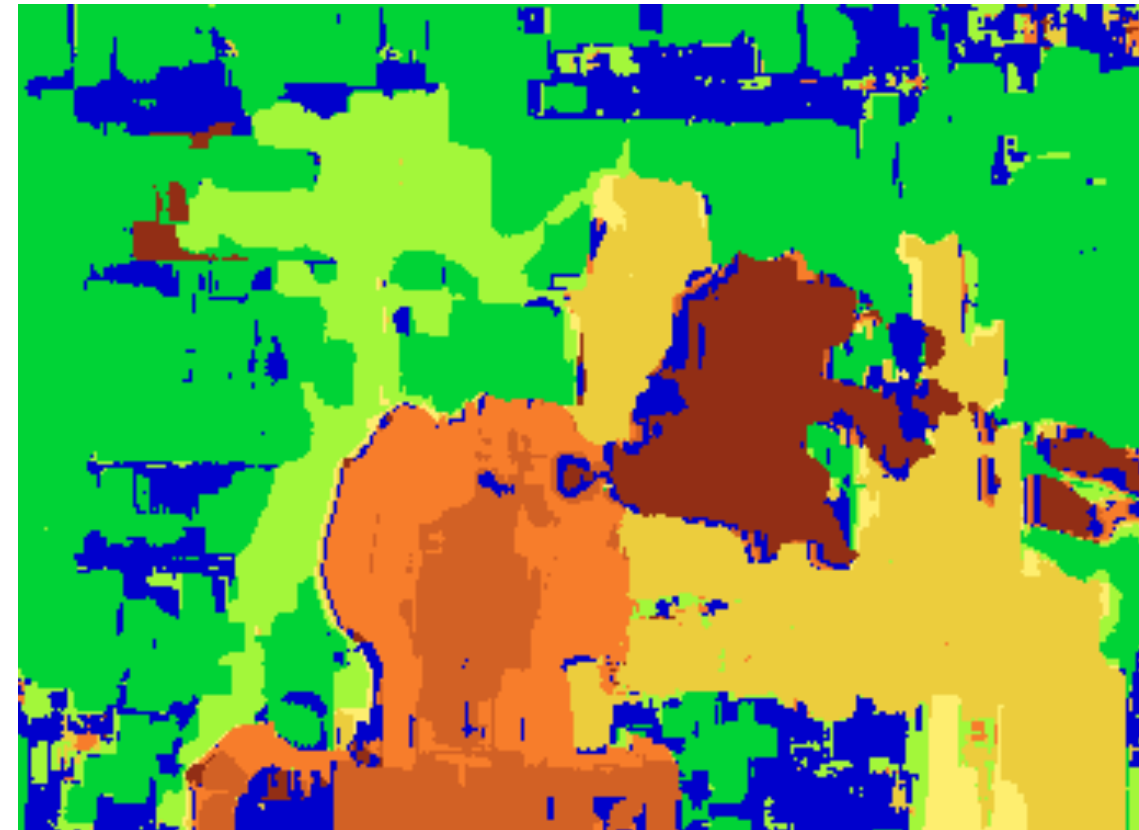
Forsyth & Ponce (2nd ed.) Figure 7.13



# Block Matching Techniques: Result



Block matching



Ground truth



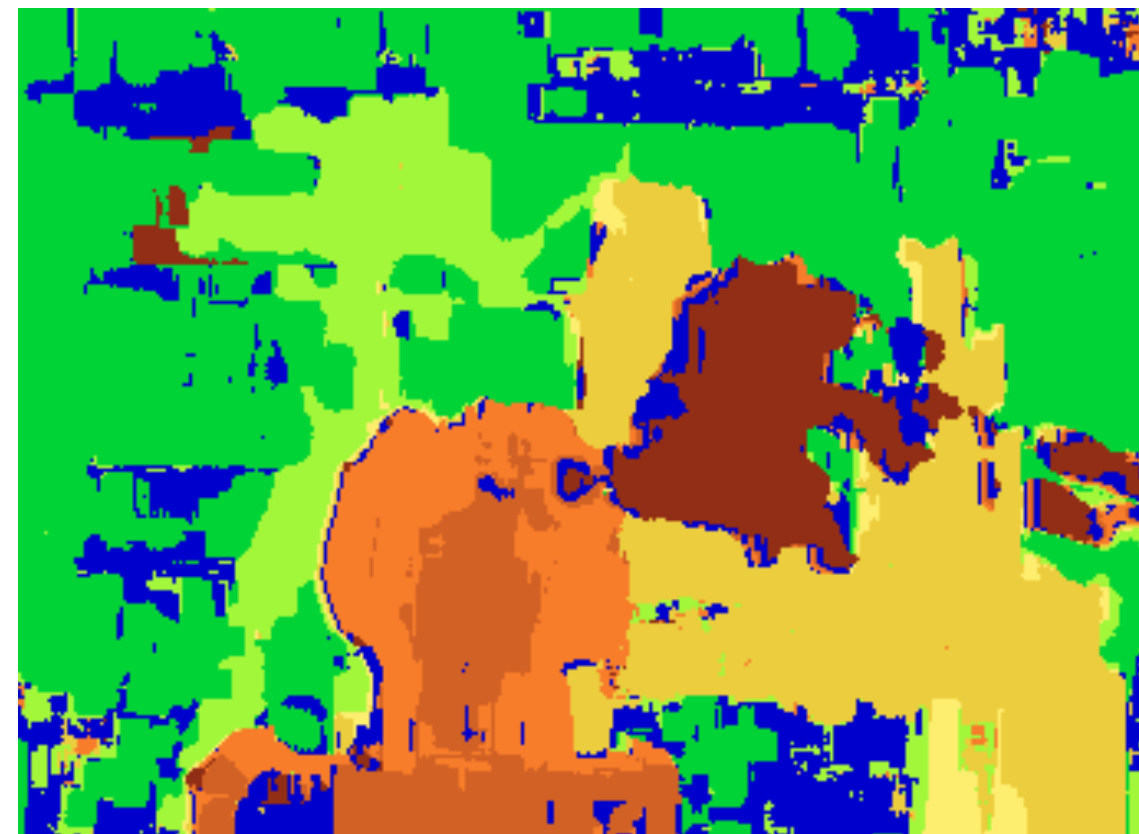
# Block Matching Techniques: Result

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

Let's make an assumption:  
depth should change smoothly



Block matching



Ground truth



# 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)

# Stereo Matching as **Energy Minimization**

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

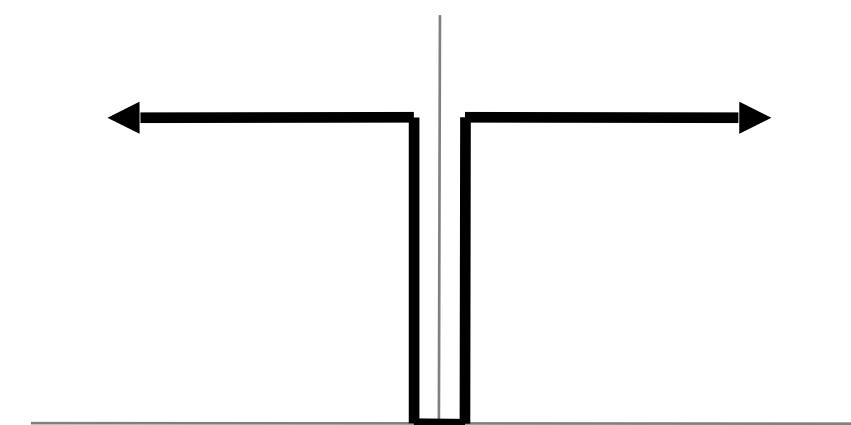
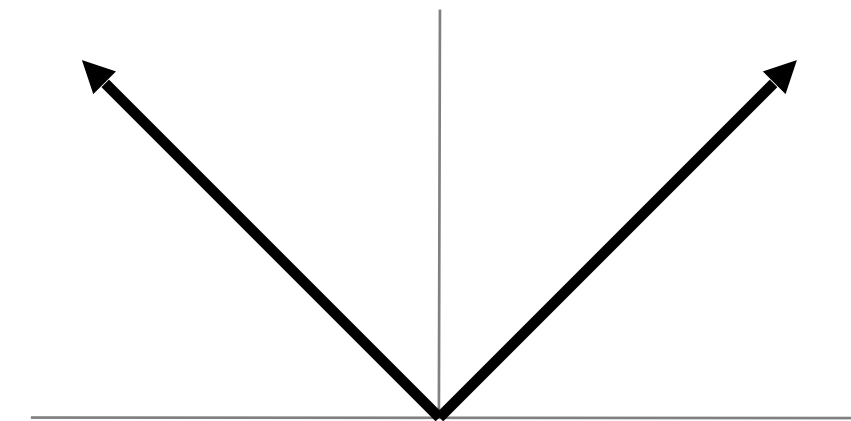
smoothness term

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

$L_1$  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”

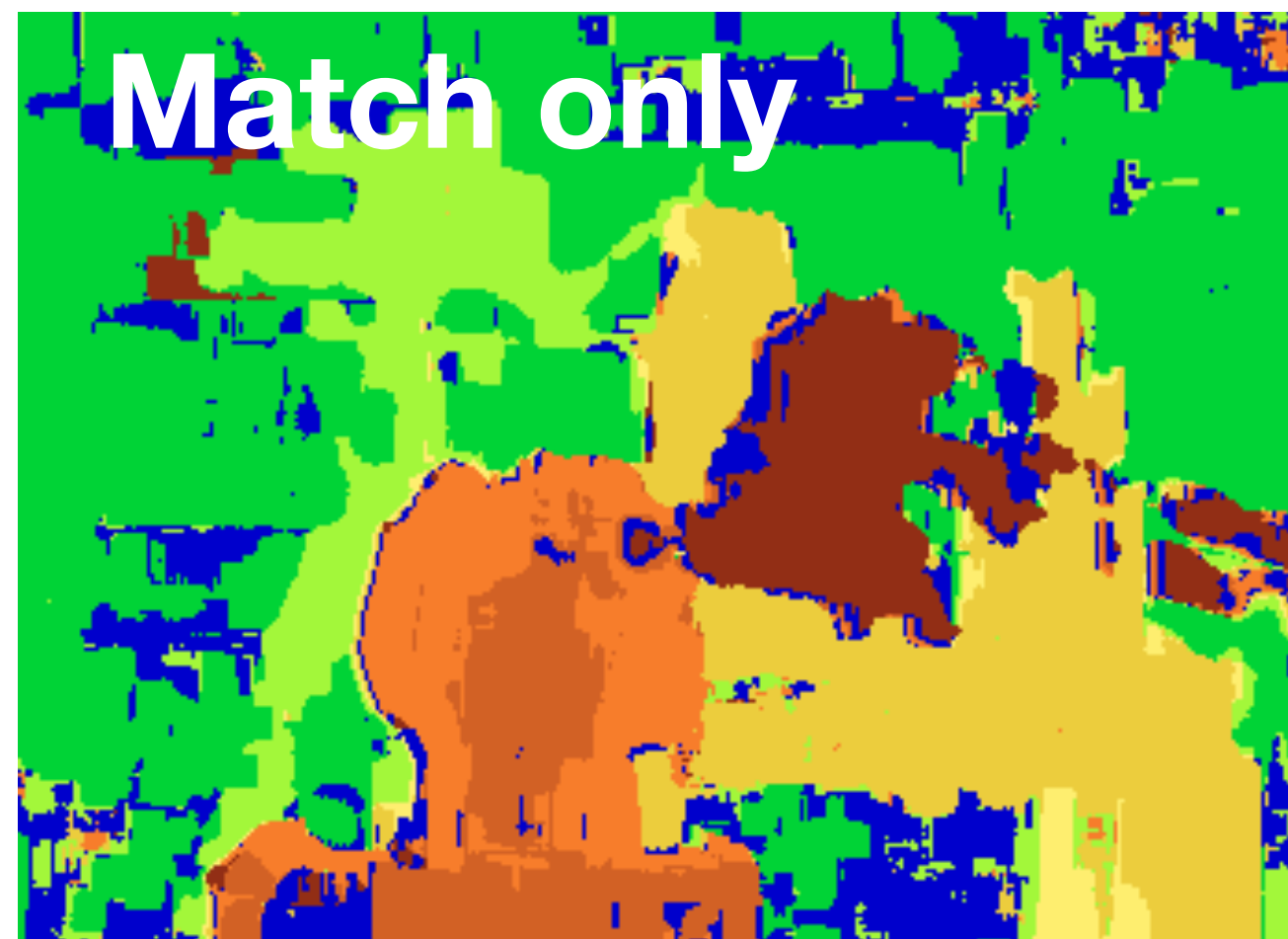


# Stereo Matching as **Energy Minimization**: Solution

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

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

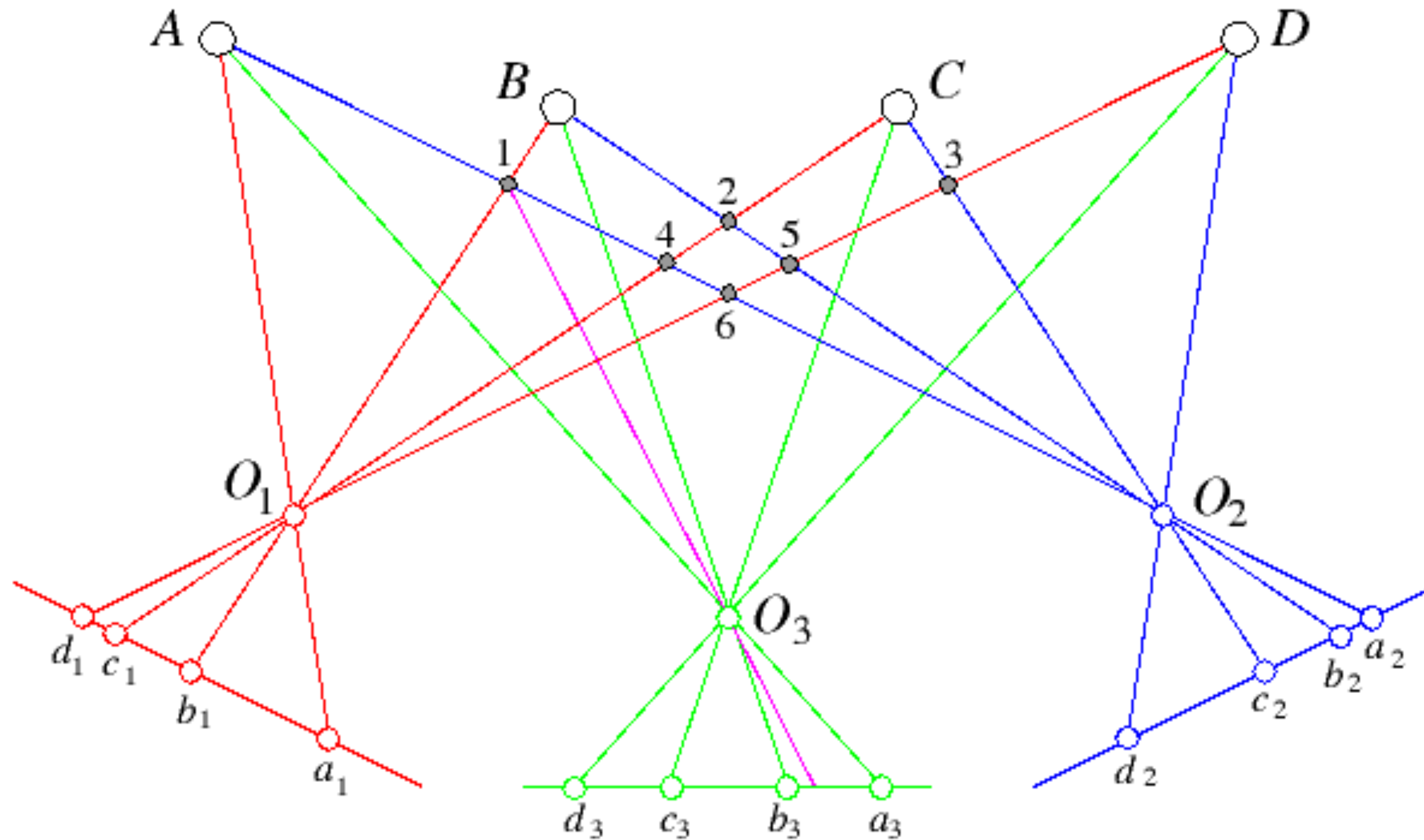
# Stereo Matching as **Energy Minimization**



Y. Boykov, O. Veksler, and R. Zabih, [Fast Approximate Energy Minimization via Graph Cuts](#), PAMI 2001

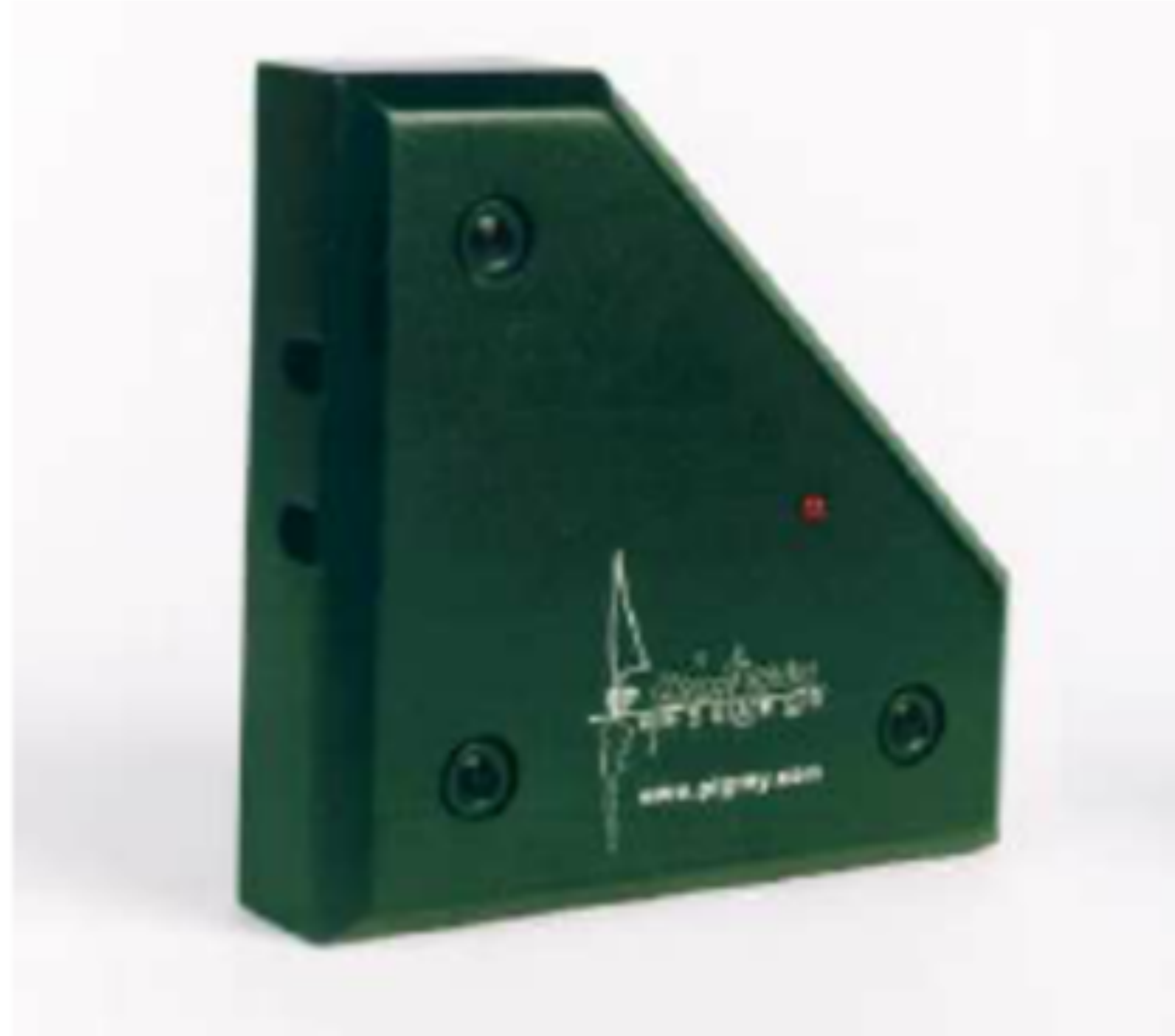
# Idea: Use More Cameras

Adding a third camera reduces ambiguity in stereo matching



Forsyth & Ponce (2nd ed.) Figure 7.17

# Point Grey Research **Digiclops**

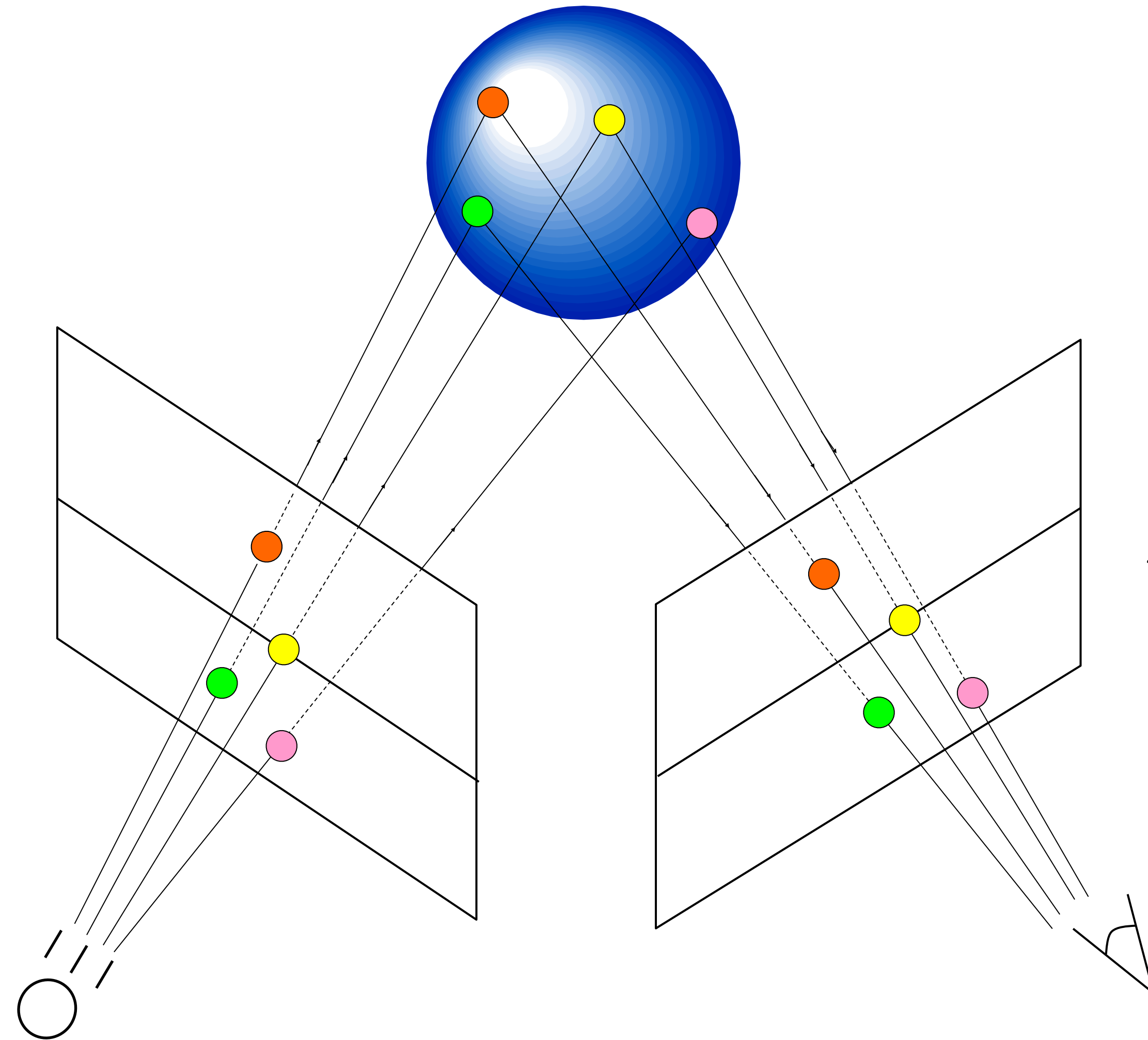


**Image credit:** Point Grey Research

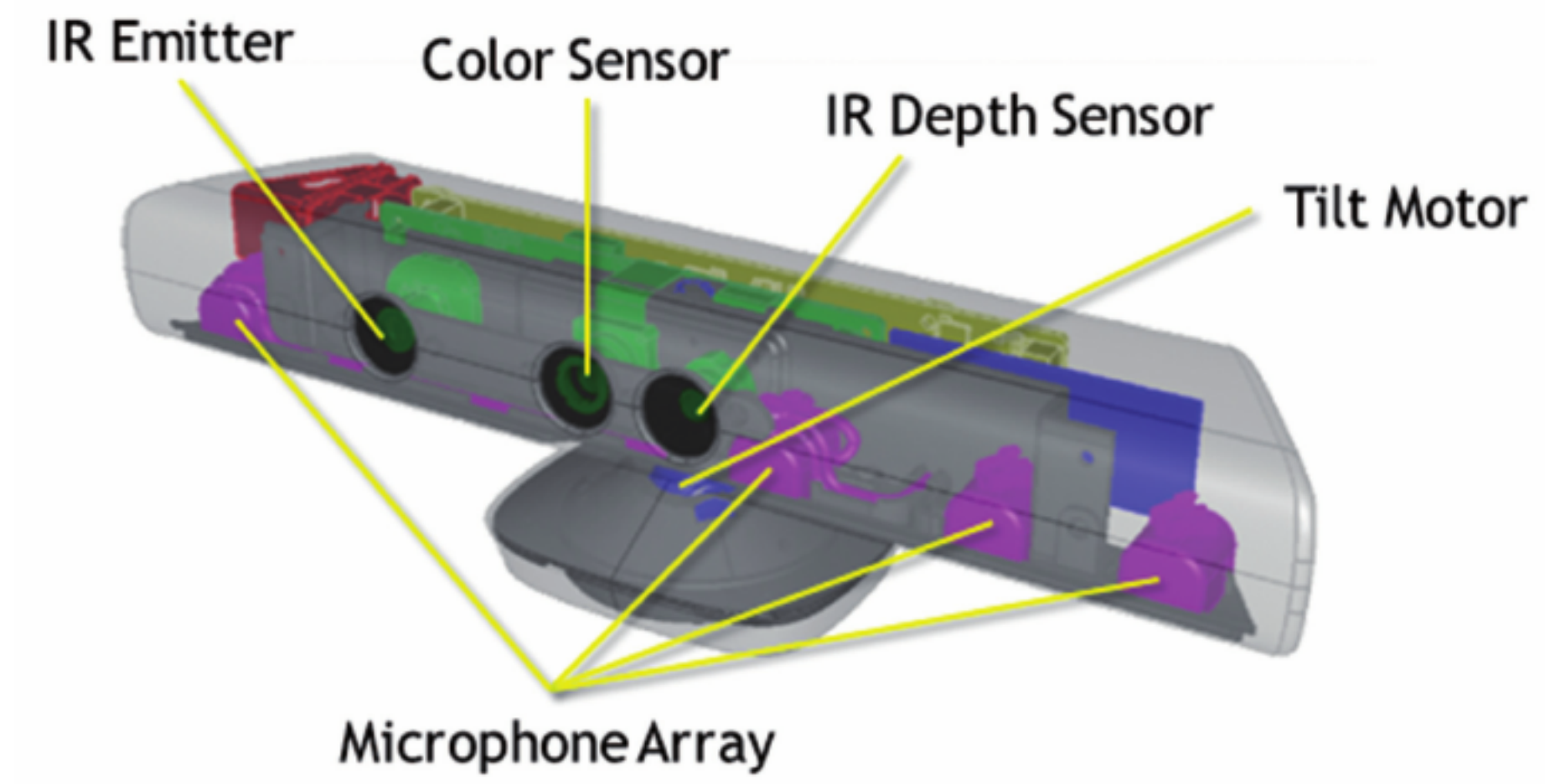


# Structured Light Imaging: Structured Light and One Camera

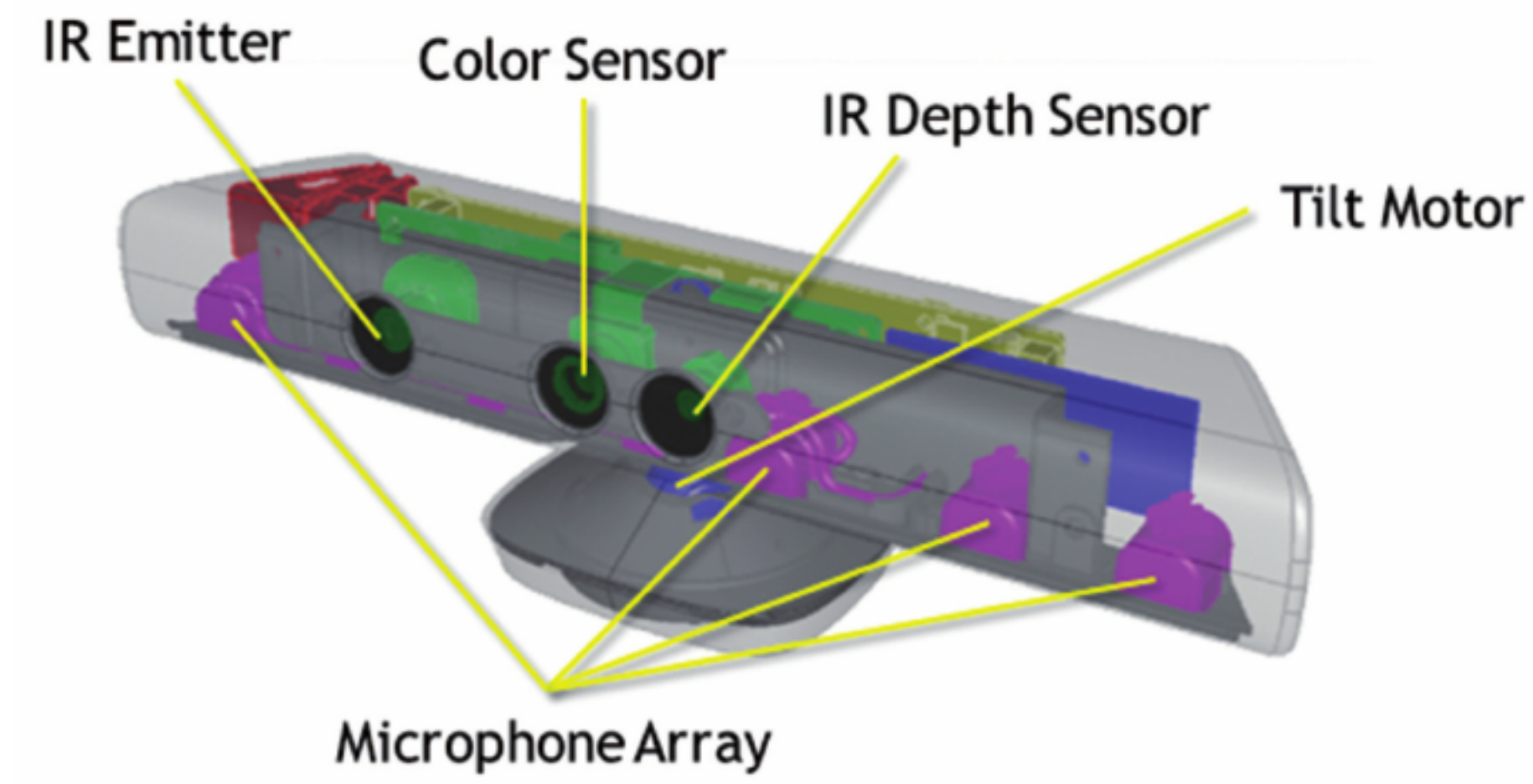
Projector acts like  
“reverse” camera



# 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?