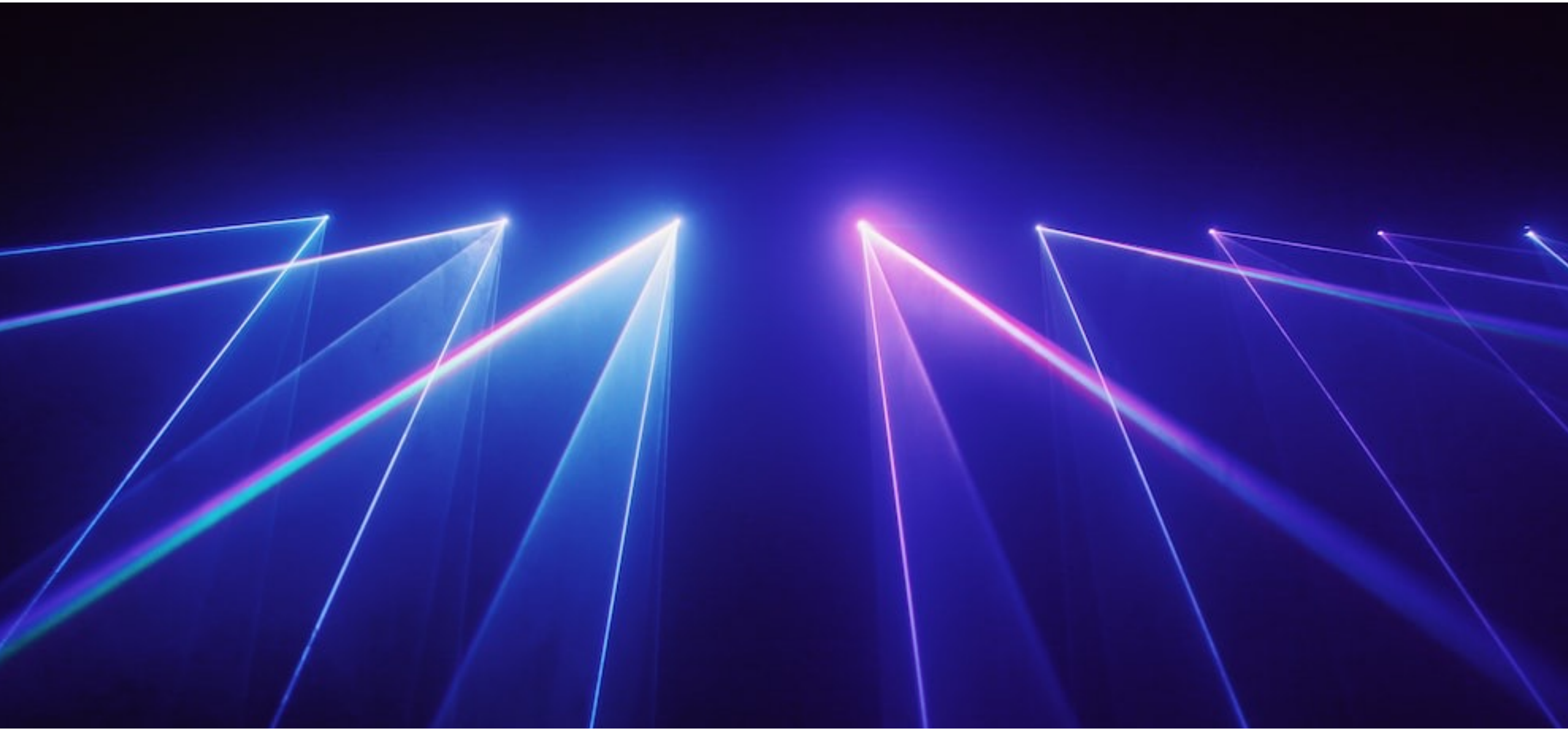
Epipolar Geometry

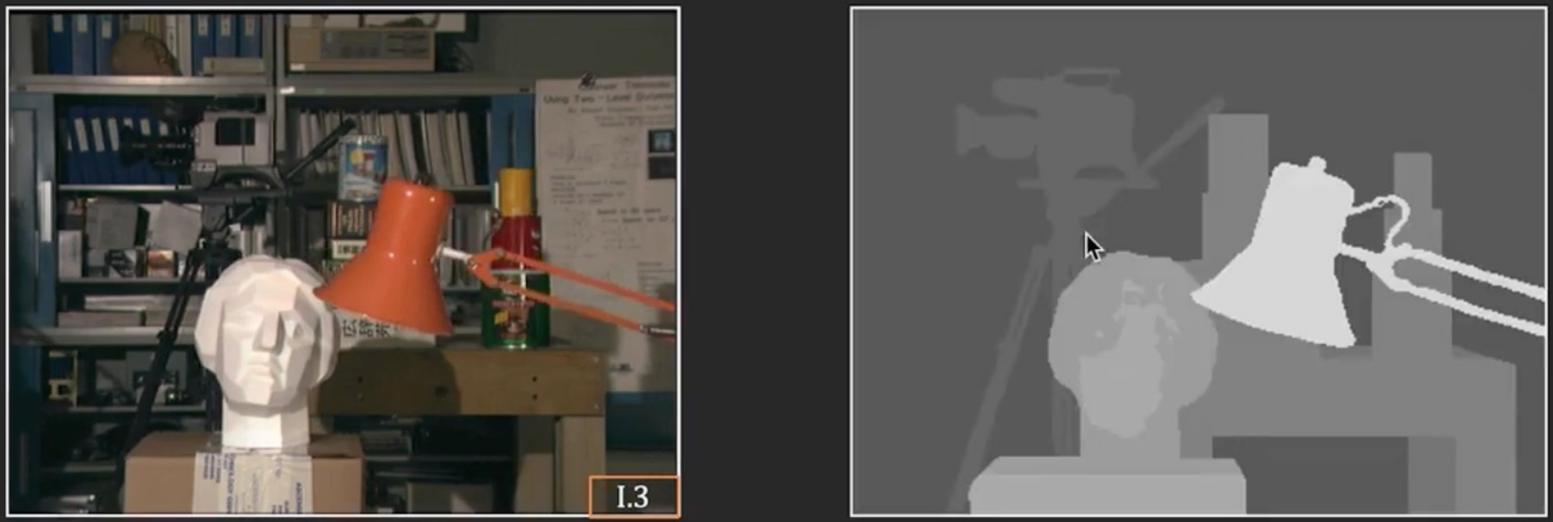


A lot of slides from Noah Snavely +
Shree Nayar's YT series: First principals of Computer Vision

CS180: Intro to Computer Vision and Comp. Photo
Angjoo Kanazawa & Alexei Efros, UC Berkeley, Fall 2023

So how do we get depth?

- Find the disparity! of corresponding points!
- Called: Stereo Matching

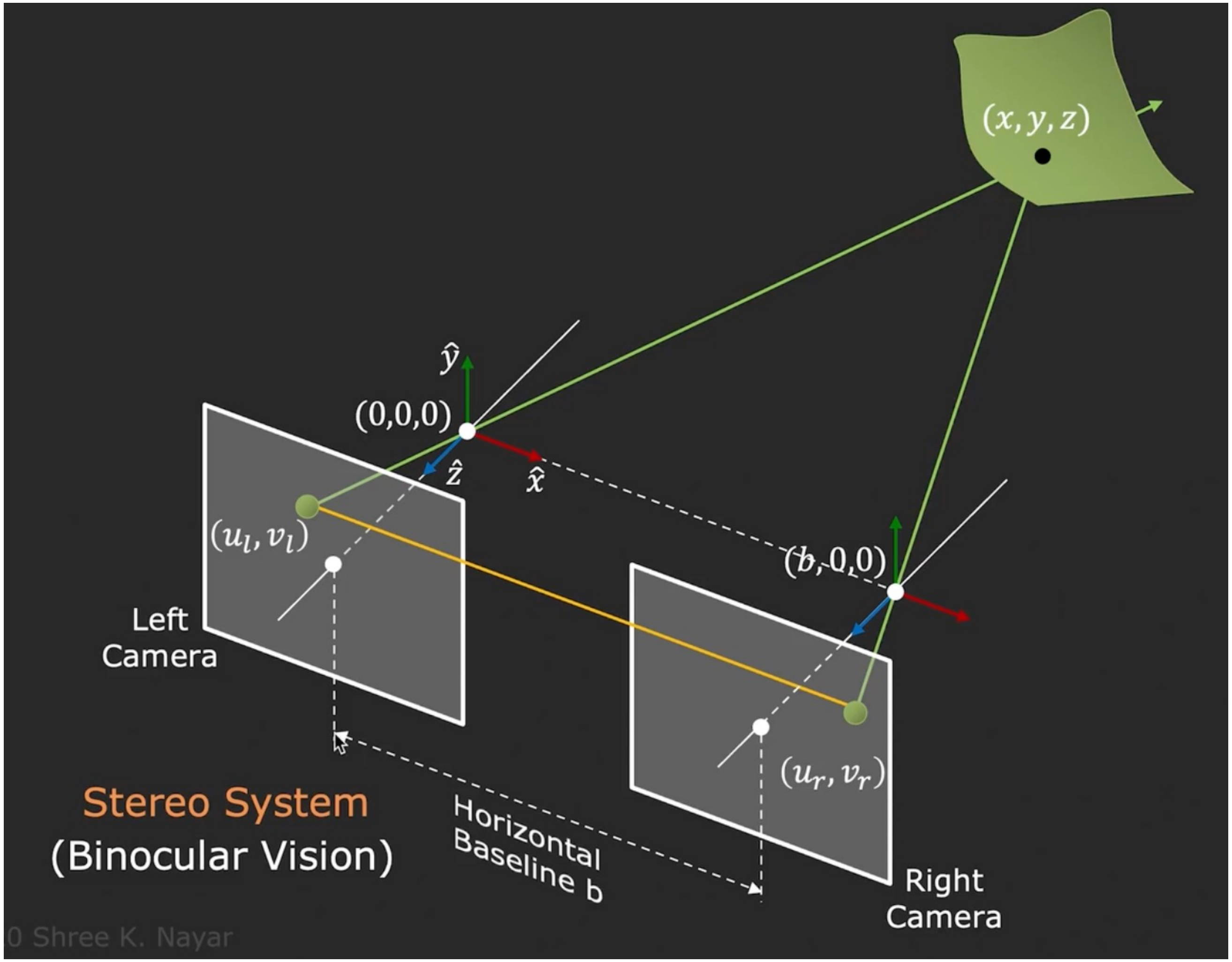


Left/Right Camera Images

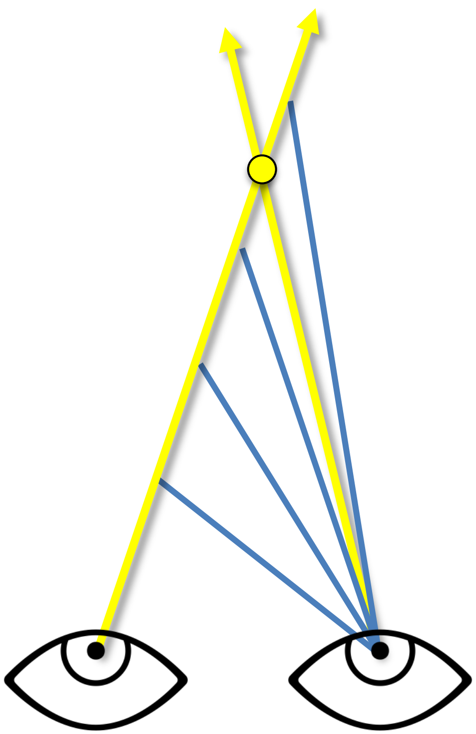
Disparity Map (Ground Truth)

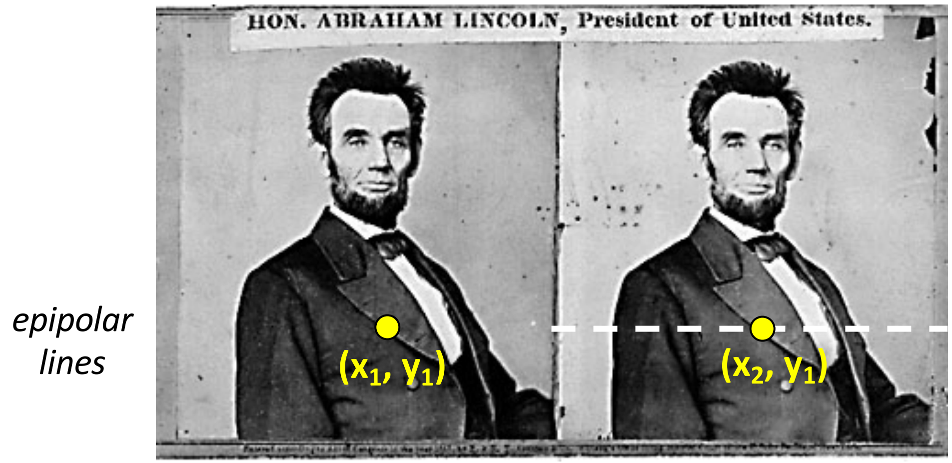
Where is the corresponding point going to be?

Hint



Epipolar Line

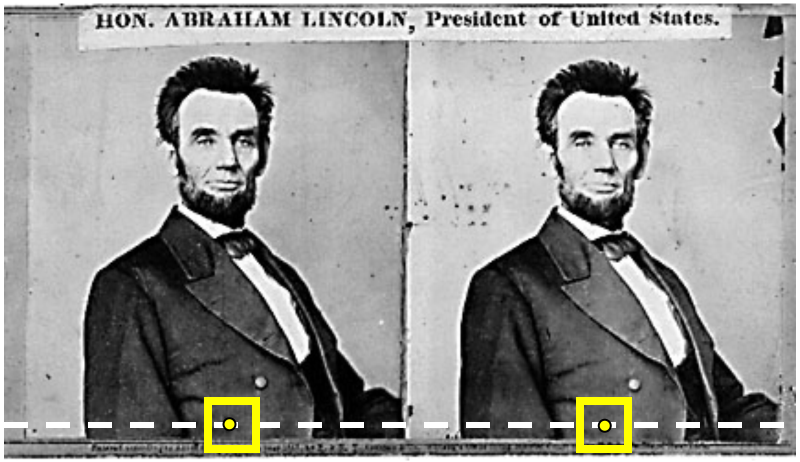




Two images captured by a purely horizontal translating camera
(*rectified* stereo pair)

$x_1 - x_2$ = the *disparity* of pixel (x_1, y_1)

Your basic stereo algorithm



For every epipolar line:

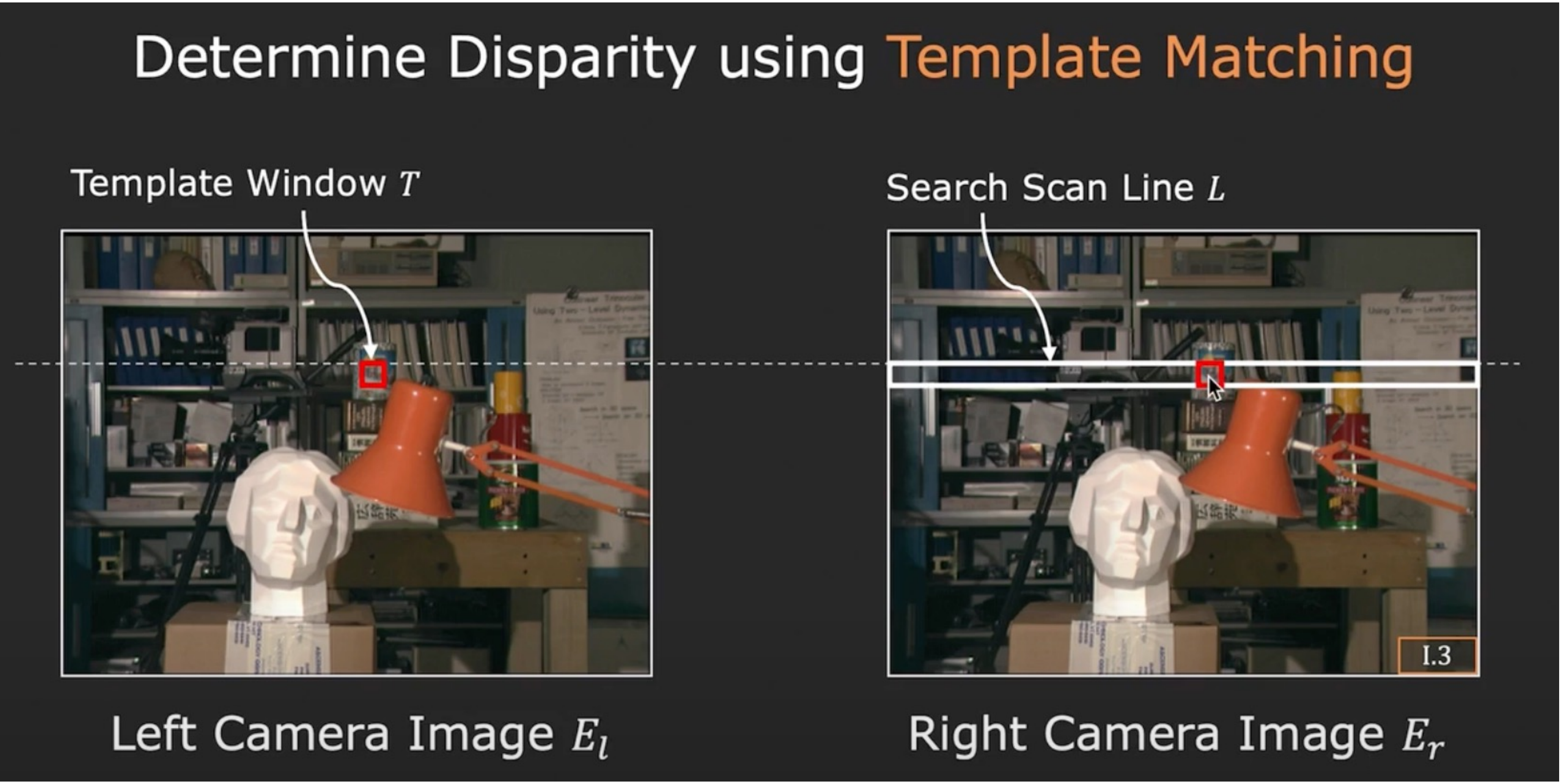
For each pixel in the left image

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

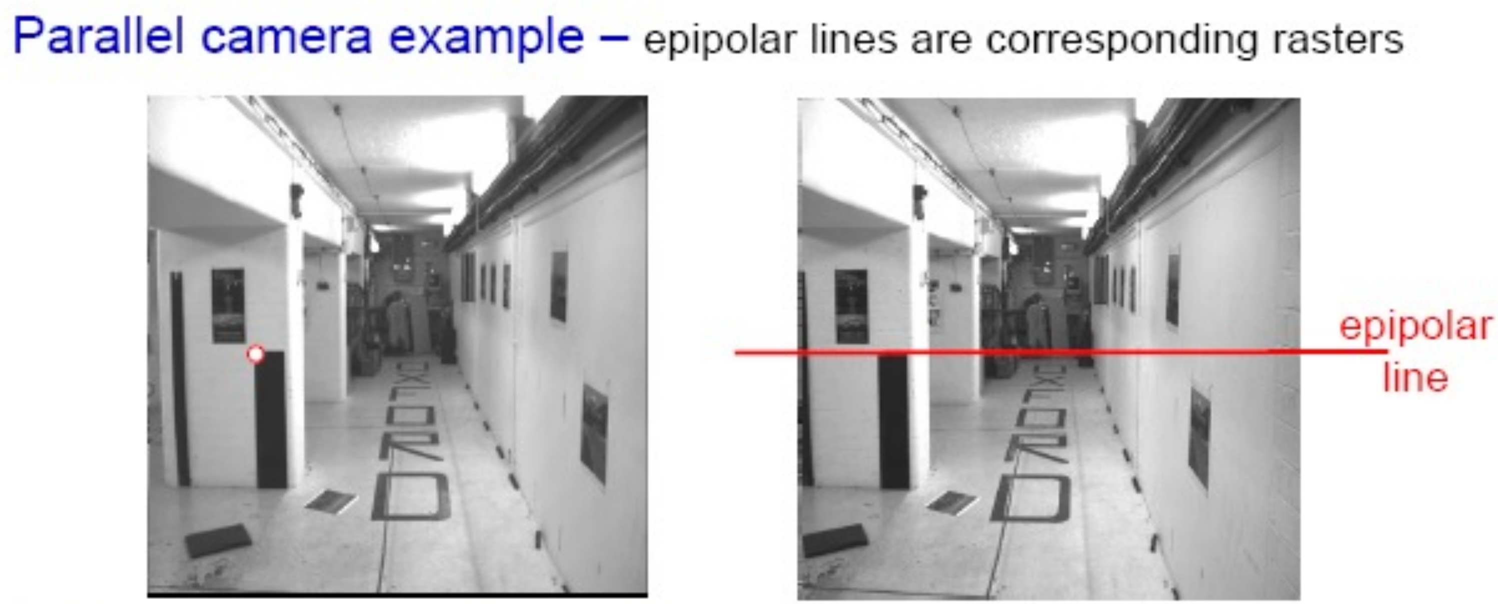
Improvement: match ***windows,*** *+ clearly lots of matching strategies*

Slide credit: Noah Snavely

Your basic stereo algorithm

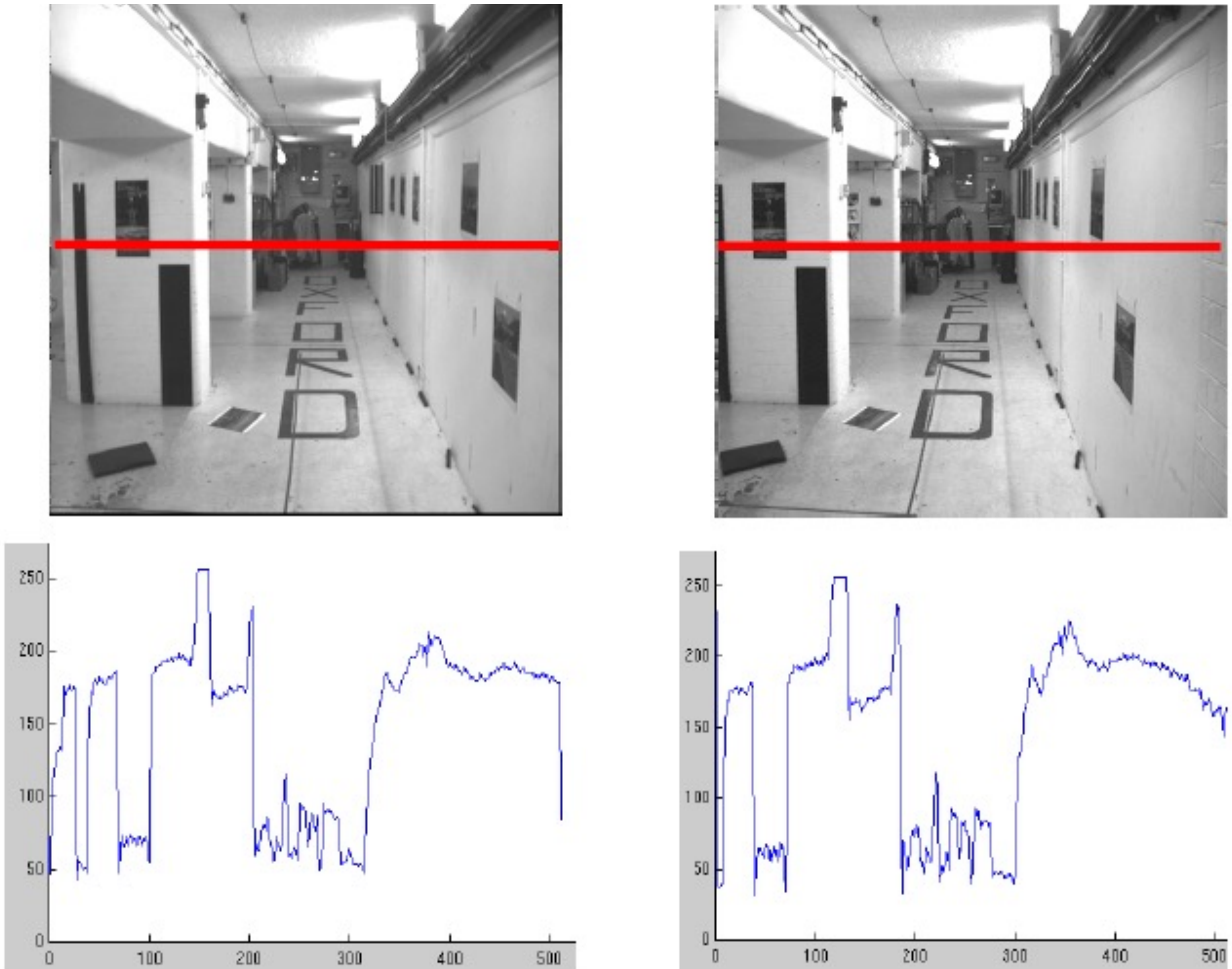


Correspondence problem



Source: Andrew Zisserman

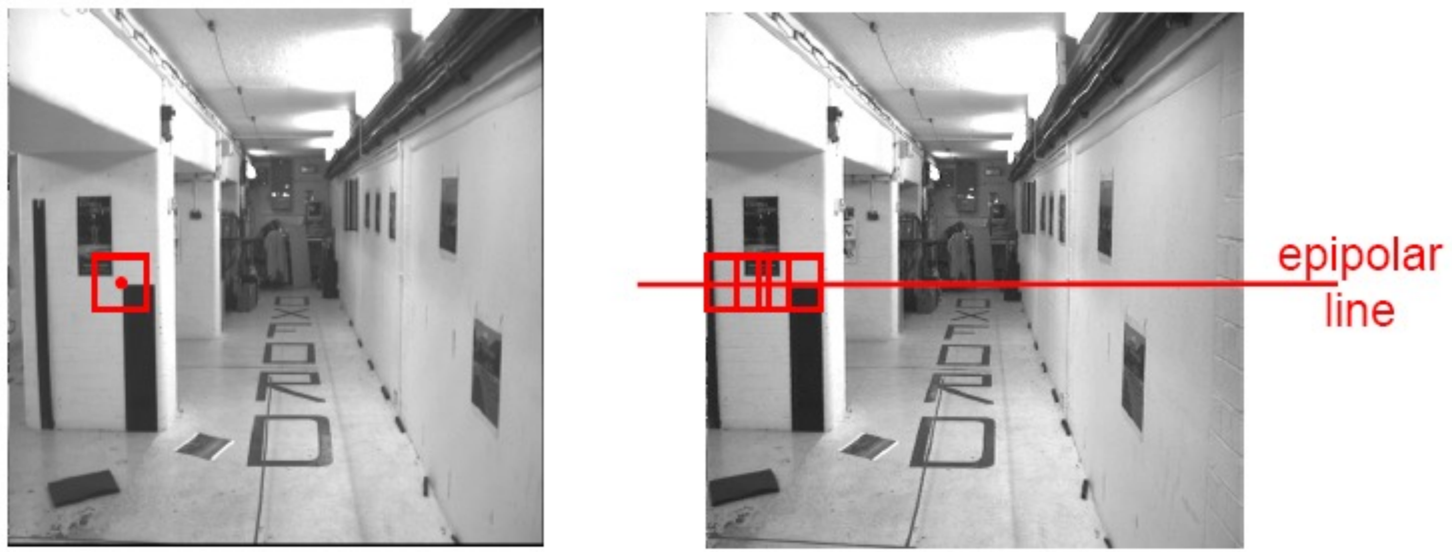
Intensity profiles



- Clear correspondence between intensities, but also noise and ambiguity

Source: Andrew Zisserman

Correspondence problem



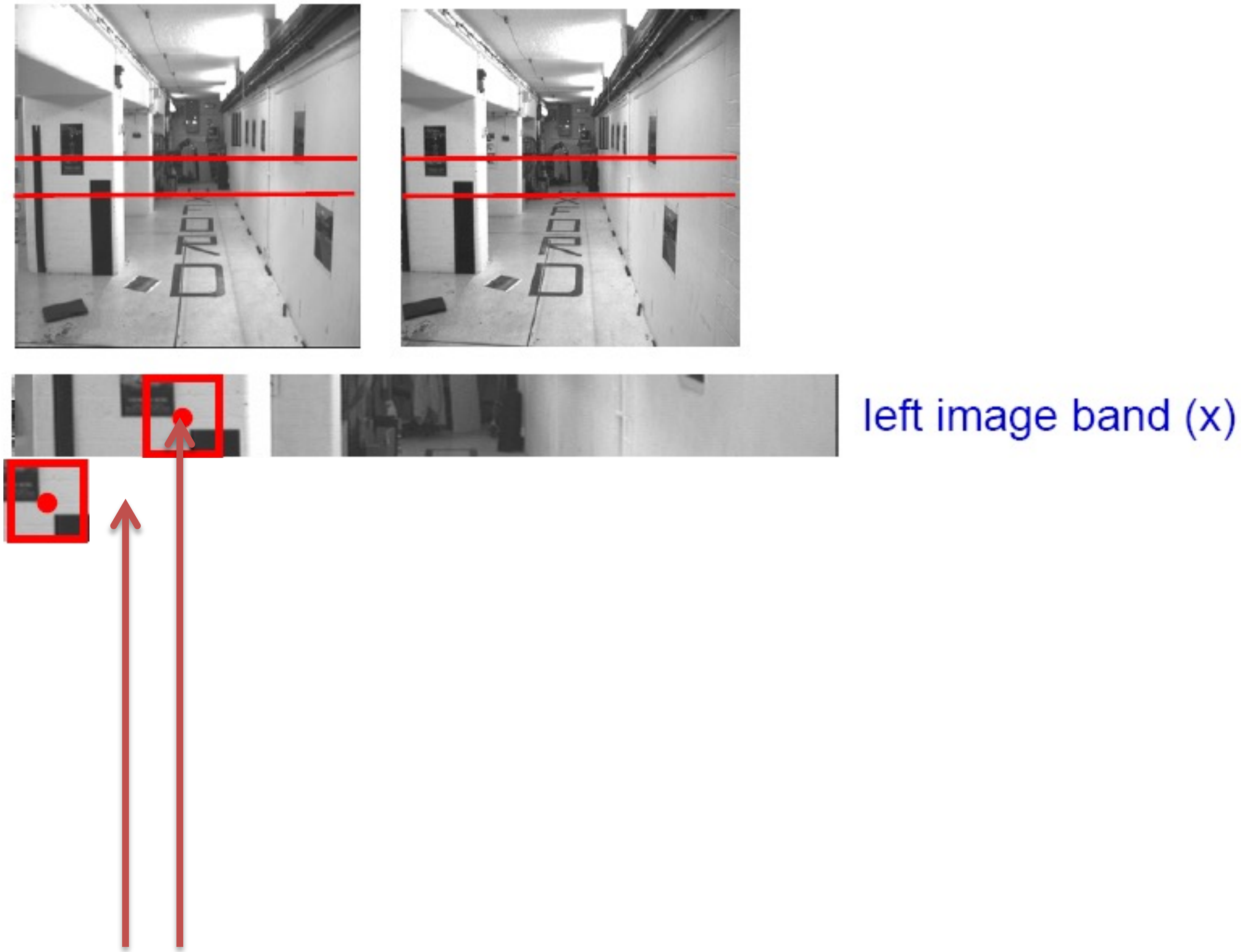
Neighborhood of corresponding points are similar in intensity patterns

- Use Normalized Cross Correlation (NCC)

or a distance in some descriptor within a window

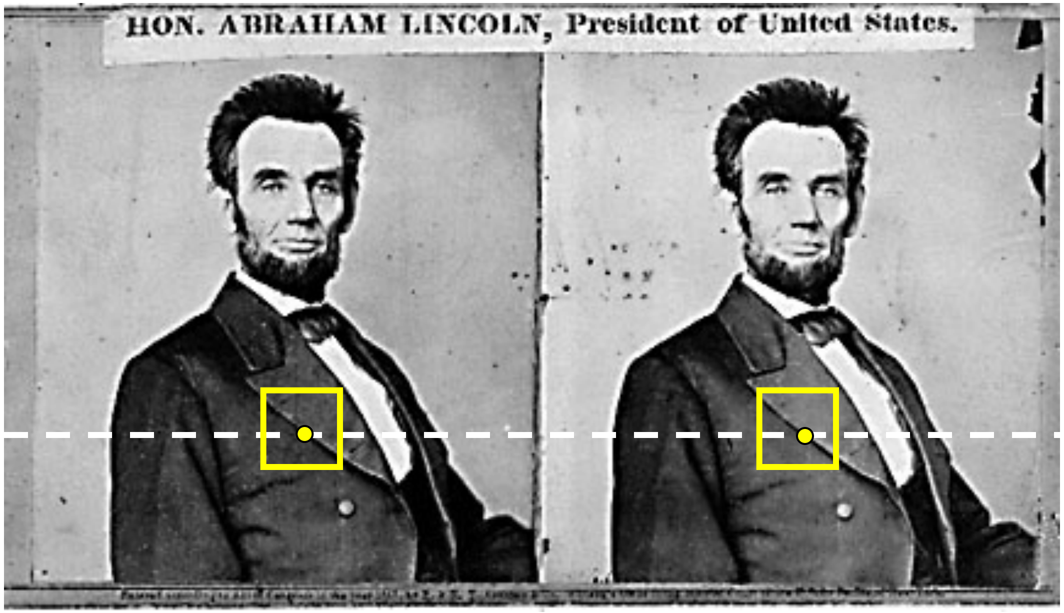
Source: Andrew Zisserman

Correlation-based window matching



Source: Andrew Zisserman

Dense correspondence search



For each epipolar line

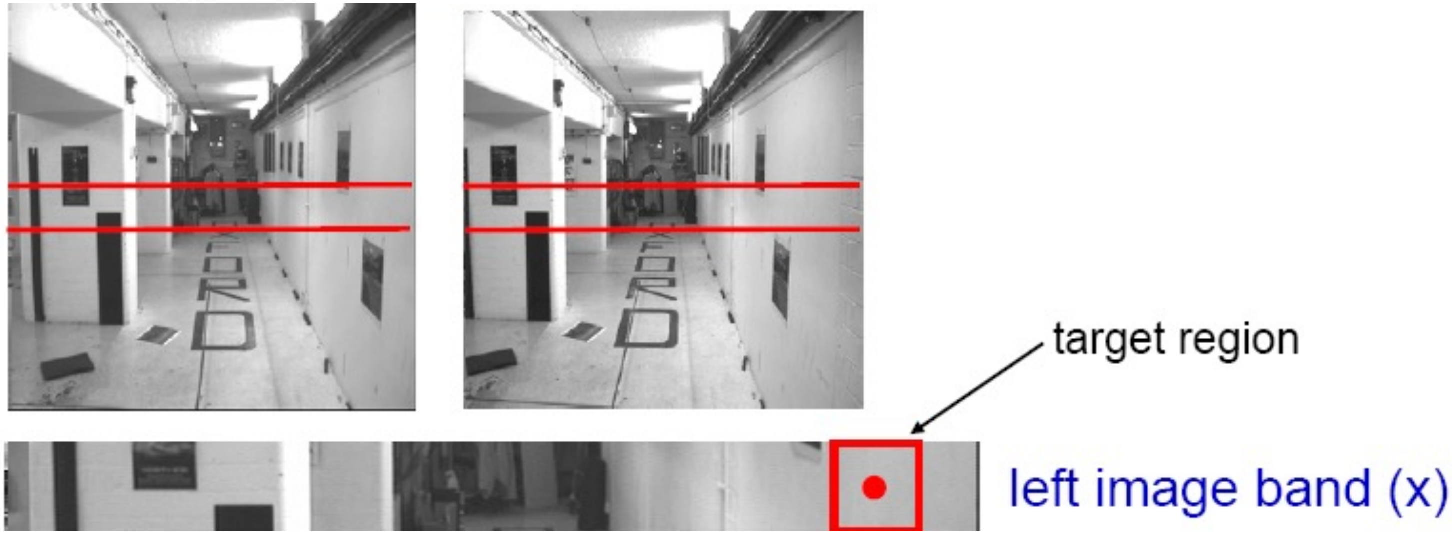
For each pixel / window in the left image

- compare with every pixel / window on same epipolar line in right image
- pick position with minimum match cost (e.g., SSD, correlation)

Adapted from Li Zhang

Grauman

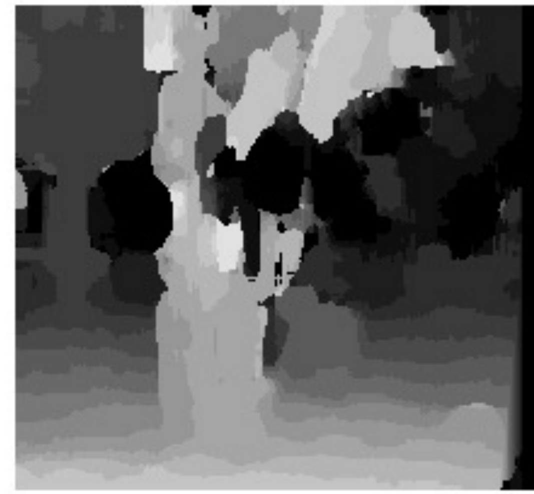
Textureless regions



Source: Andrew Zisserman

Effect of window size

W = 3



W = 20

Want window large enough to have sufficient intensity variation, yet small enough to contain only pixels with about the same disparity.

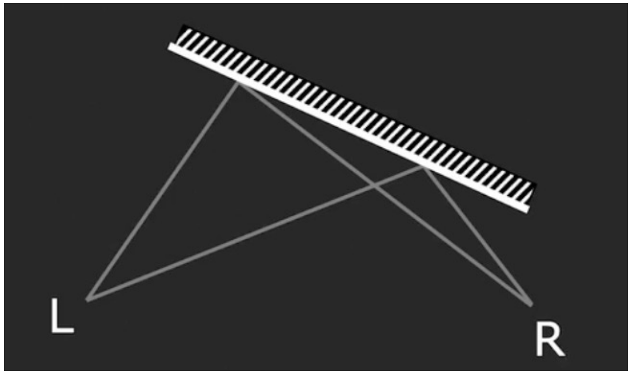
Figures from Li Zhang

Grauman

Issues with Stereo

- Surface must have non-repetitive texture

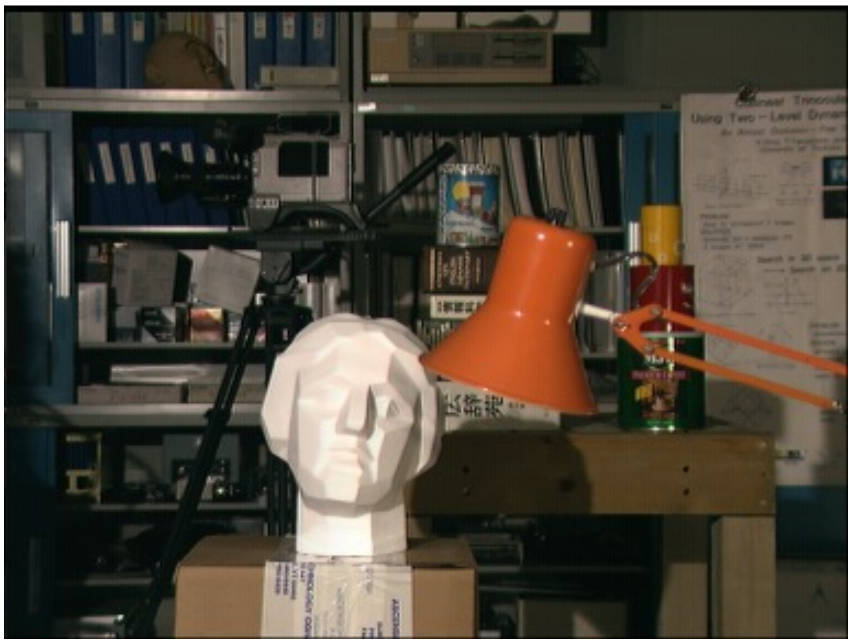
- Foreshortening effect makes matching a challenge



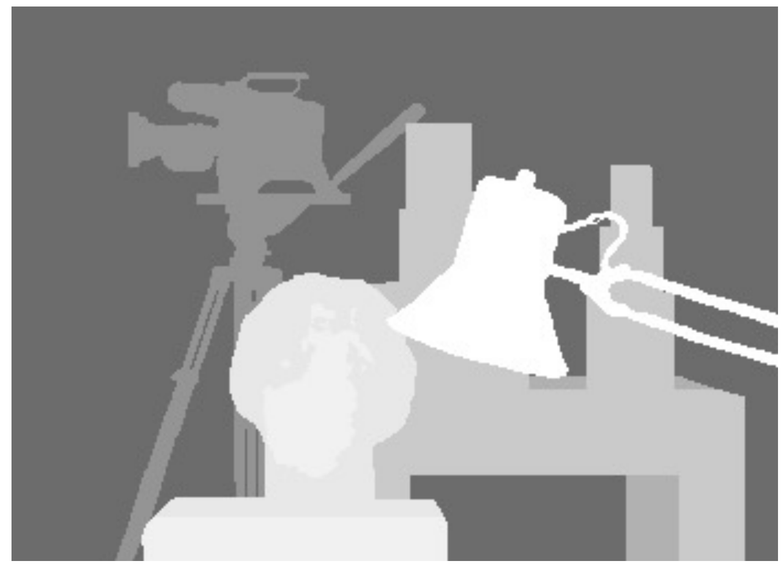
Slide Credit: Shree Nayar

Stereo Results

– Data from University of Tsukuba

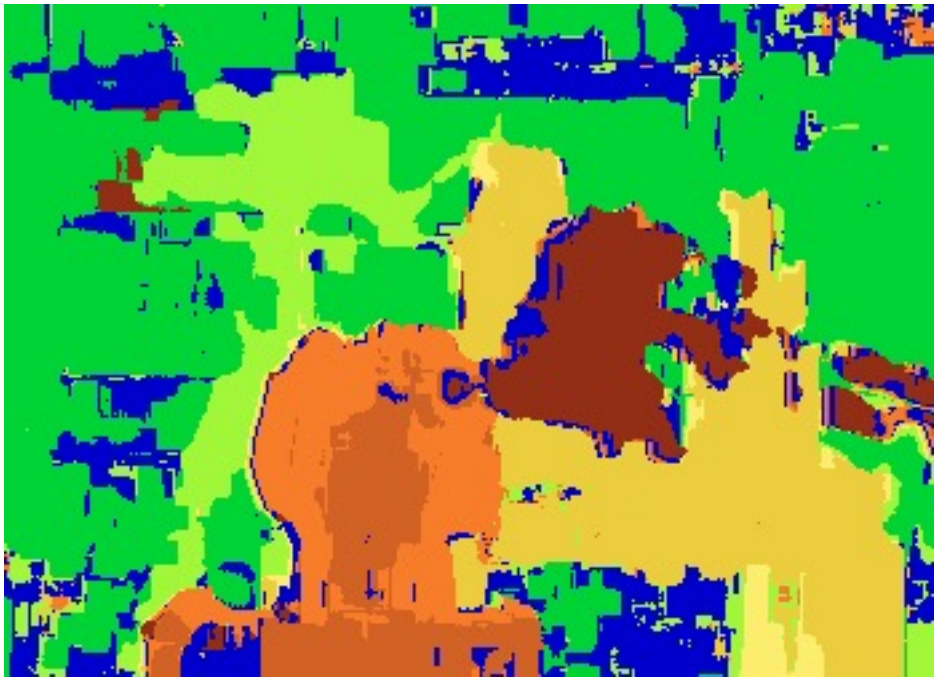


Scene



Ground truth

Results with Window Search



Window-based matching
(best window size)

Ground truth

Better methods exist...

Energy Minimization

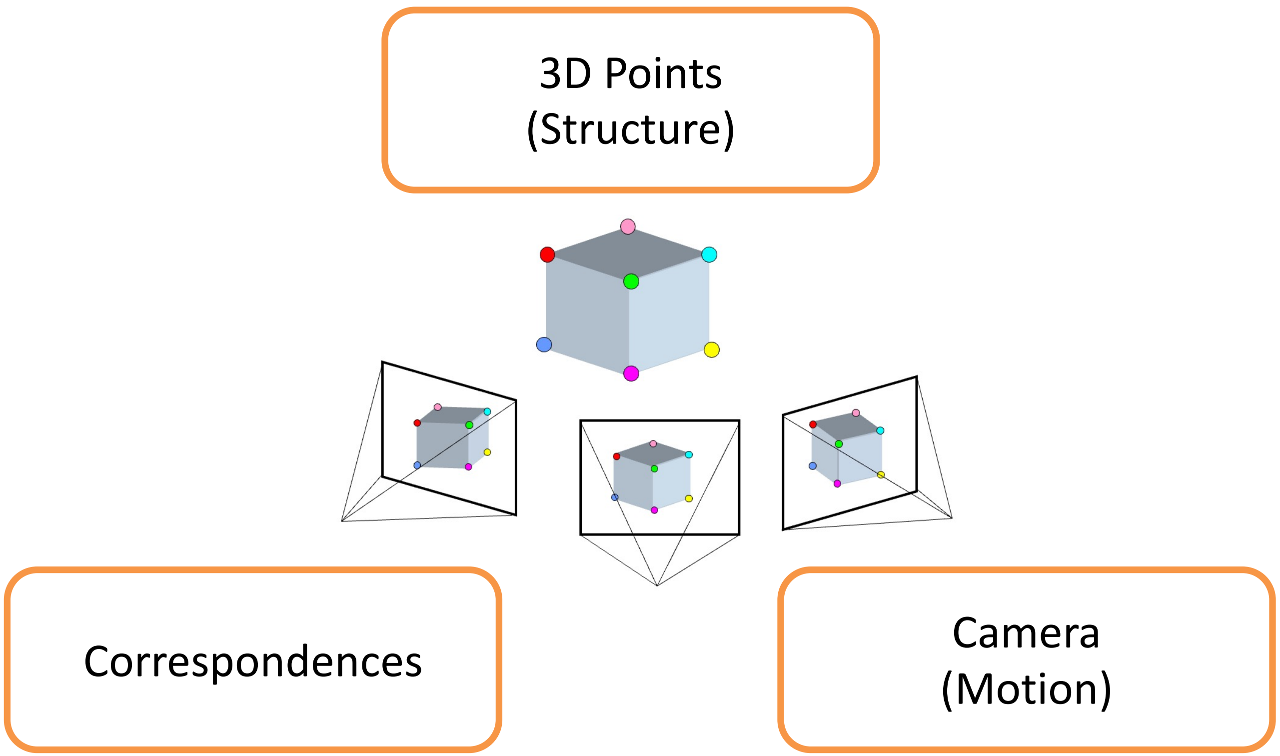
Ground truth

Boykov et al., Fast Approximate Energy Minimization via Graph Cuts, International Conference on Computer Vision, September 1999.

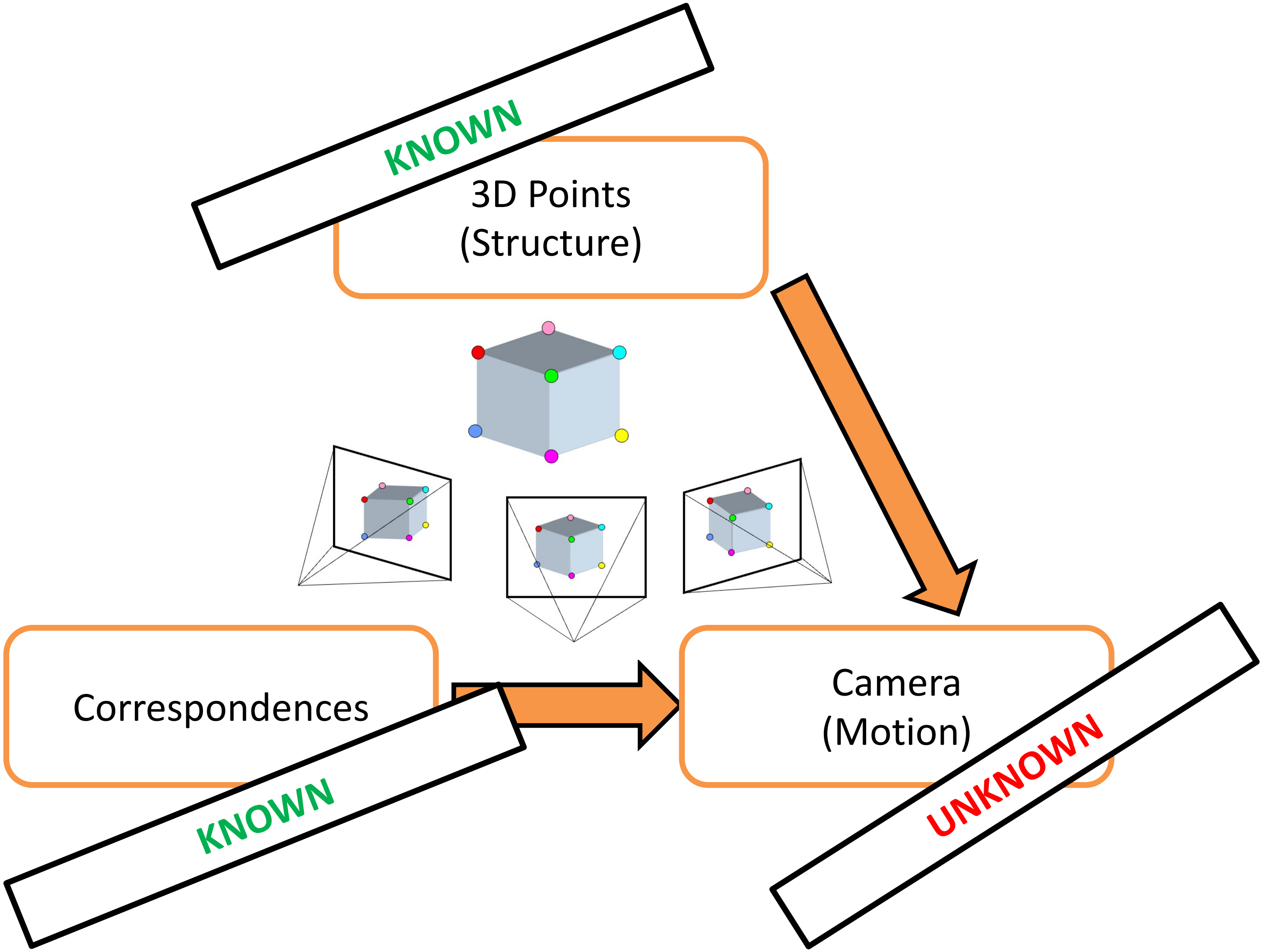
Summary

- With a simple stereo system, **how much pixels move, or “disparity”** give information about the depth
- Correspondences to measure the pixel disparity

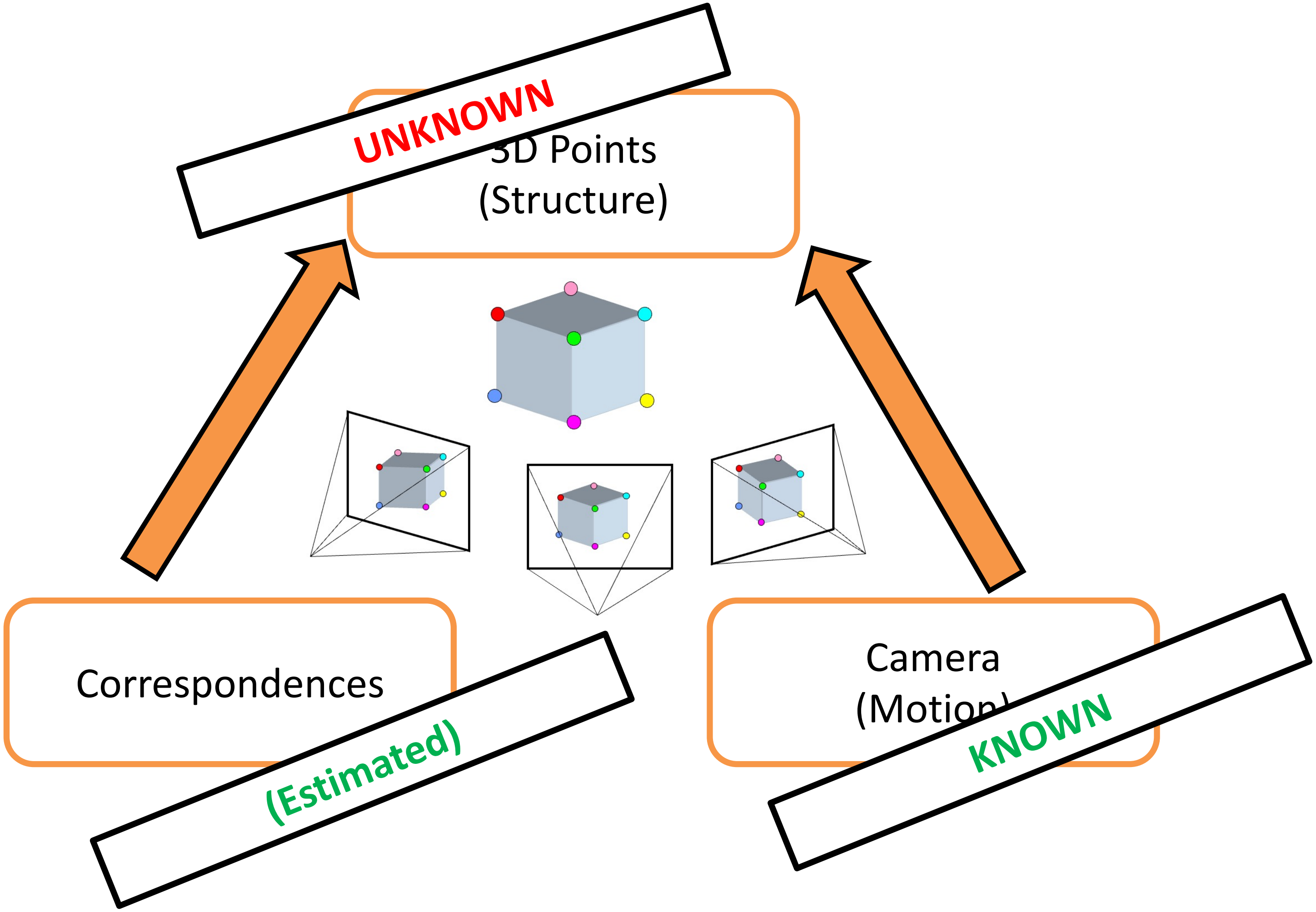
Many problems in 3D



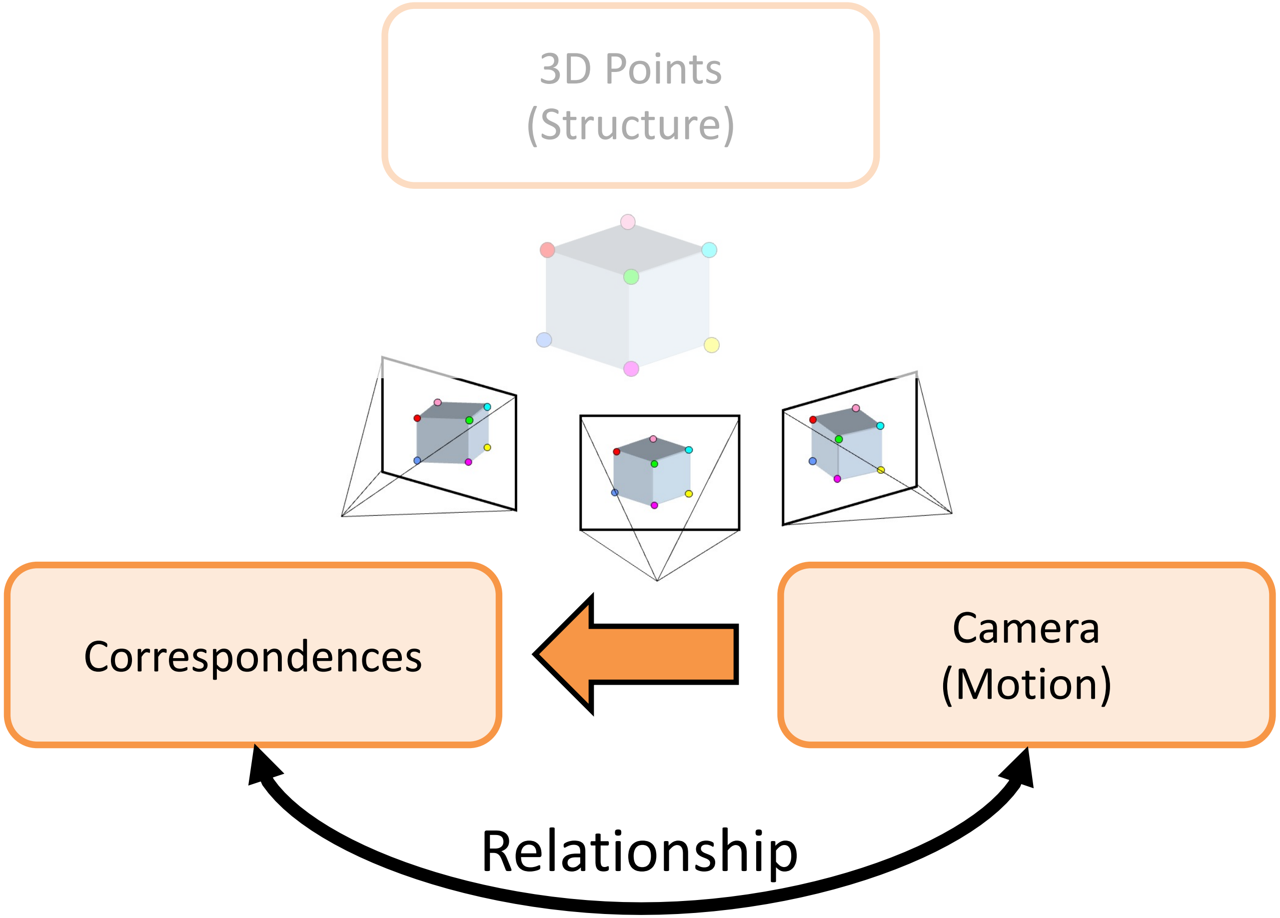
Camera Calibration



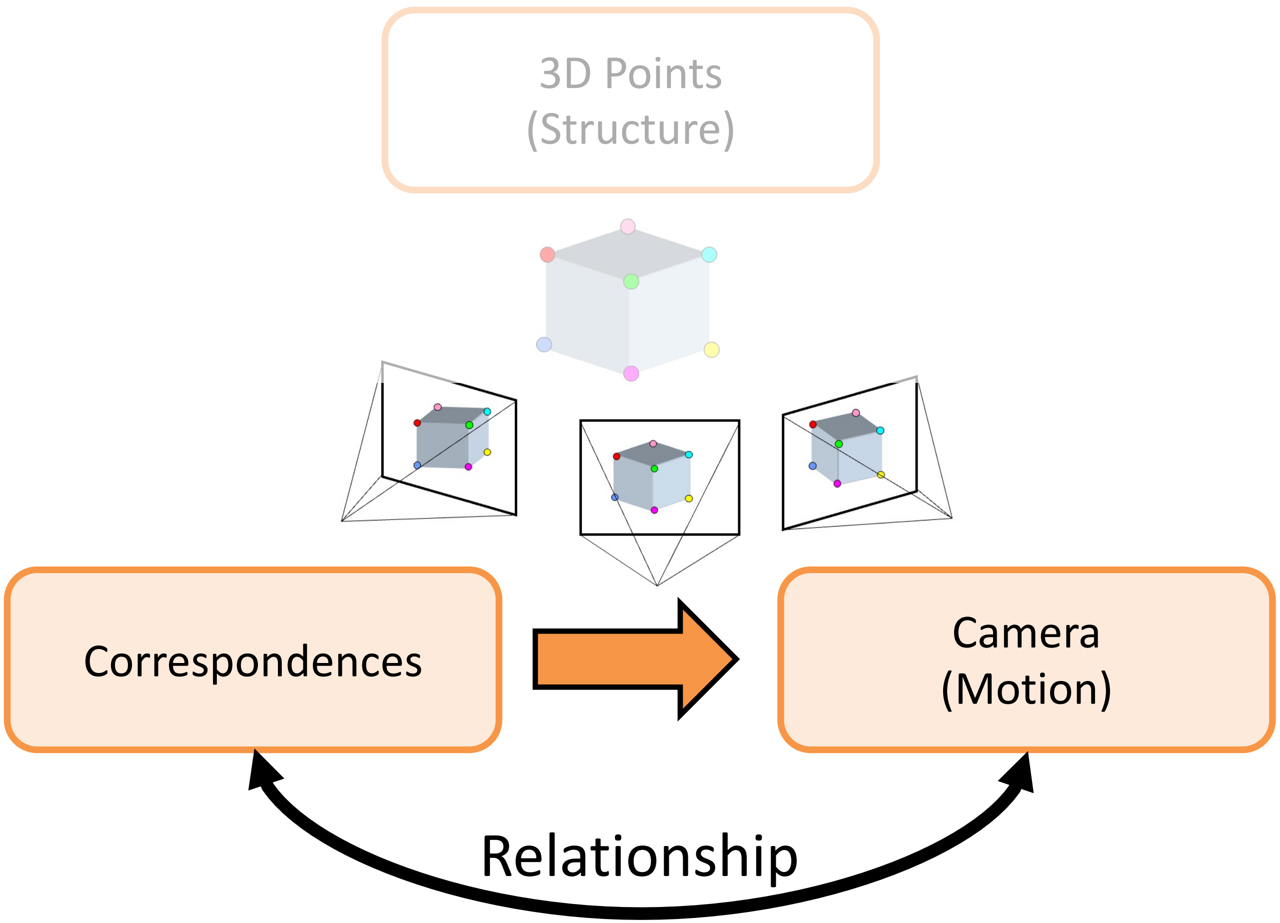
Stereo (w/2 cameras);
Multi-view Stereo / Triangulation



Camera helps Correspondence: **Epipolar Geometry**



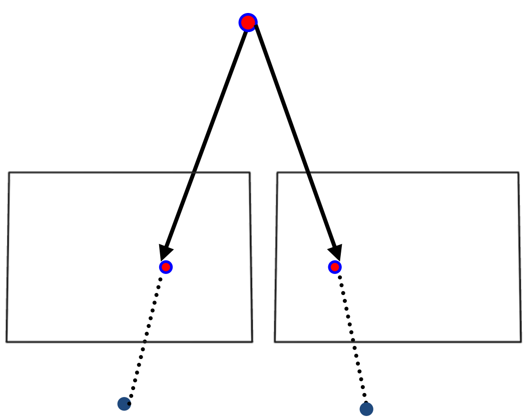
Correspondence gives camera:
Epipolar Geometry

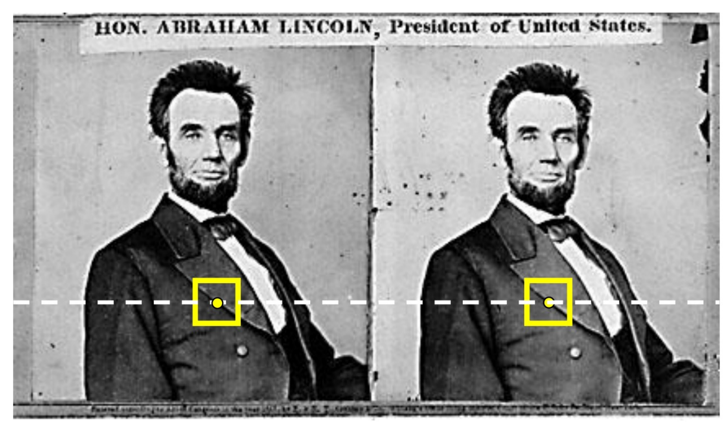


Recap

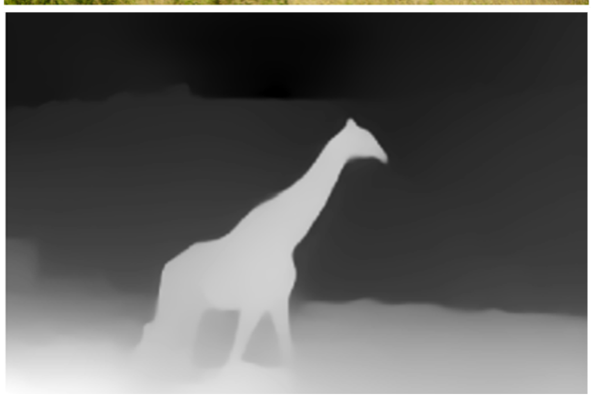
We covered:

- How to estimate the camera parameters
  - "Calibration"
  - Solve for intrinsics & extrinsics
- With a simple stereo, correspondences lie on horizontal lines
- depth is inversely proportional to disparity
  (how much the pixel moves)





What Depth Map provides



Monocular Depth Prediction [Ranftl et al. PAMI’20]

warping the pixel based on its depth as you change the views

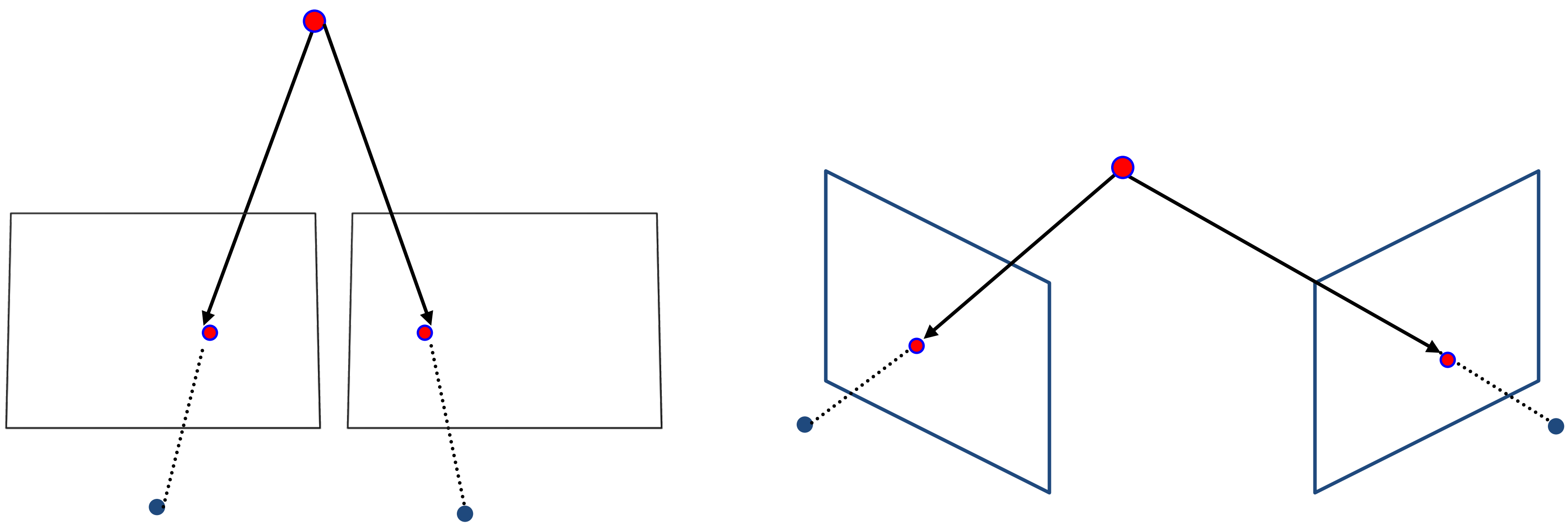
More cool things with Depth

3D photo

AR

Next: General case

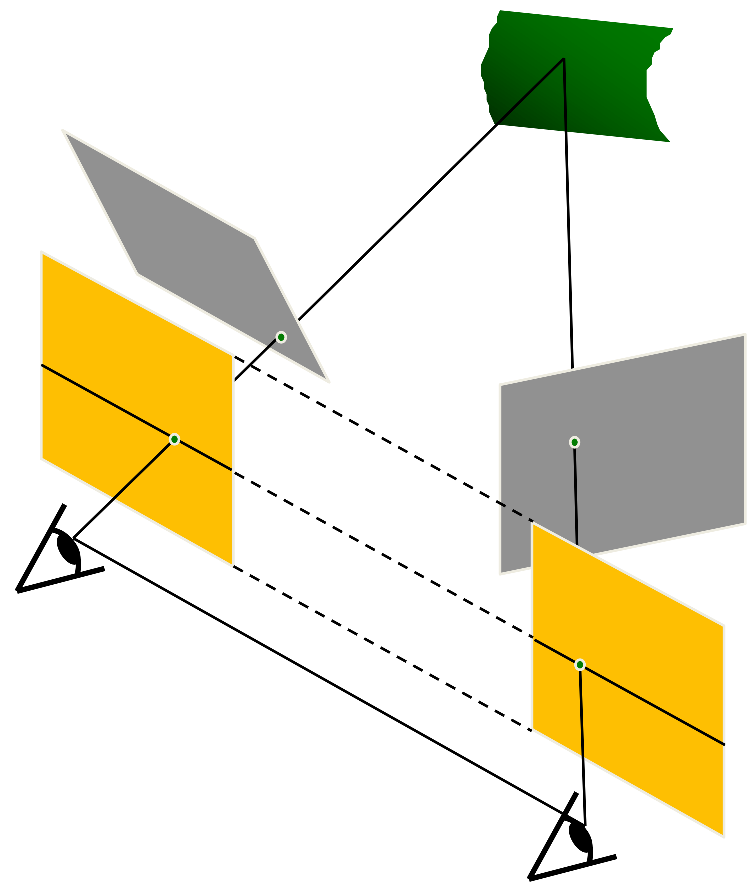
- The two cameras need not have parallel optical axes.
- Assume camera intrinsics are calibrated



Same hammer:
Find the correspondences, then solve for structure

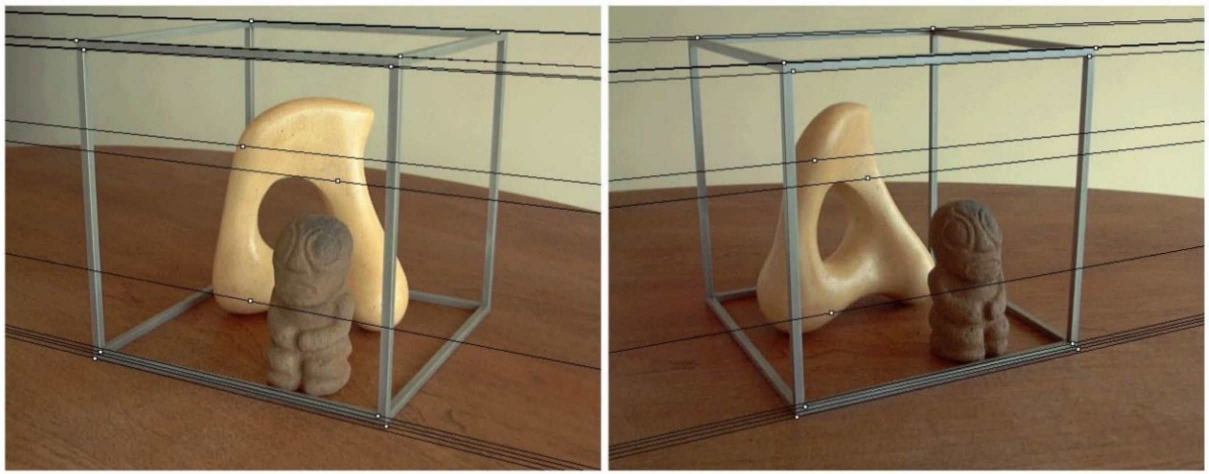
Option 1: Rectify via homography

- reproject image planes onto a common plane
  - plane parallel to the line between optical centers
- pixel motion is horizontal after this transformation
- Two homographies, one for each input image reprojection
  - C. Loop and Z. Zhang. [Computing Rectifying Homographies for Stereo Vision](). CVPR 1999.

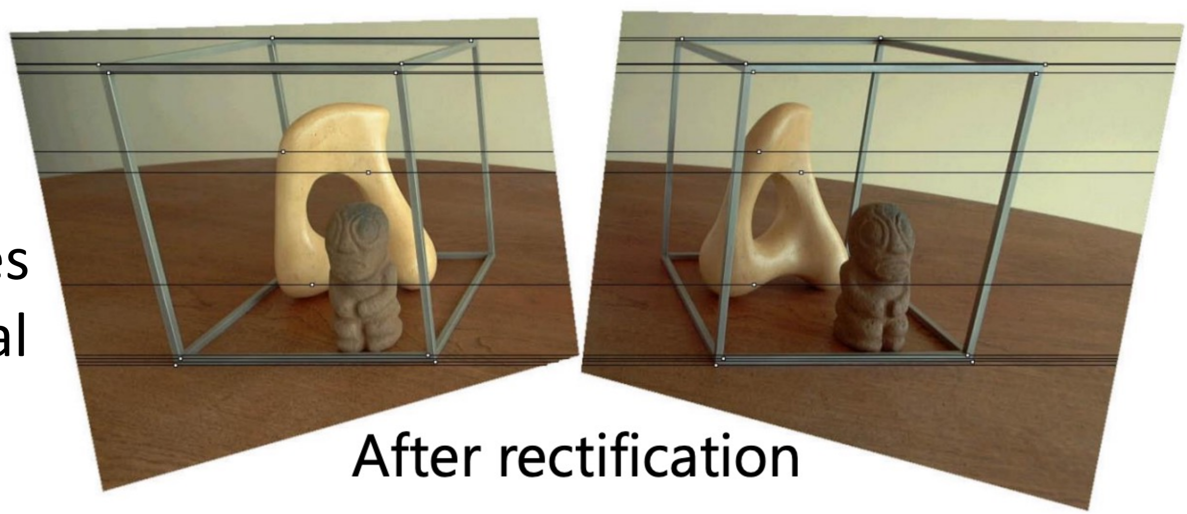


Slide courtesy of Noah Snavely

Option 1: Rectify via homography



Original stereo pair



After rectification

Then find correspondences on the horizontal scan line

Slide courtesy of Noah Snavely

General case, known camera, find depth: Option 2

1. Find correspondences
2. Triangulate

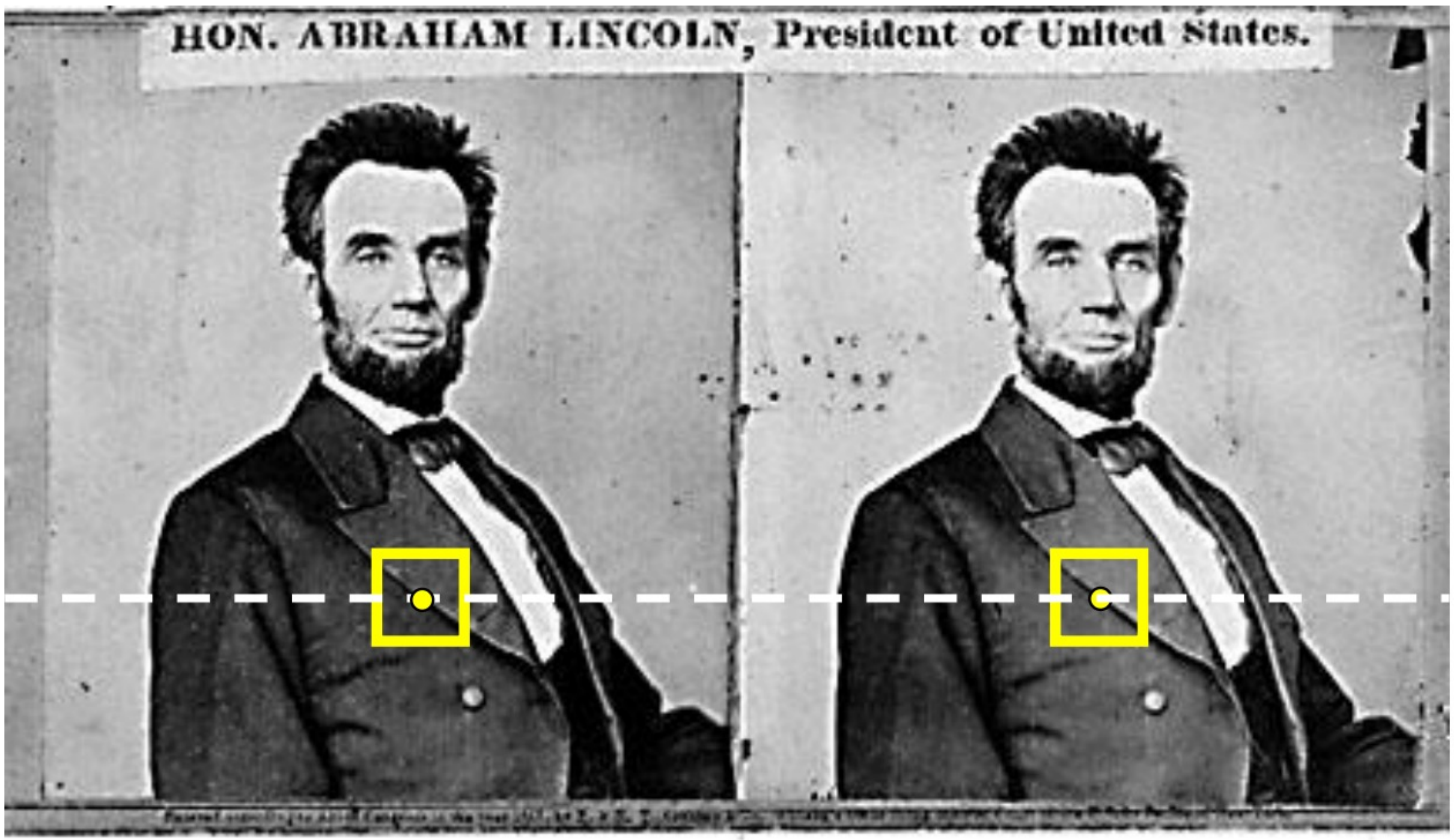
General case, known camera, find depth: Option 2

1. **Find correspondences**
2. Triangulate

Can we restrict the search space again to 1D?

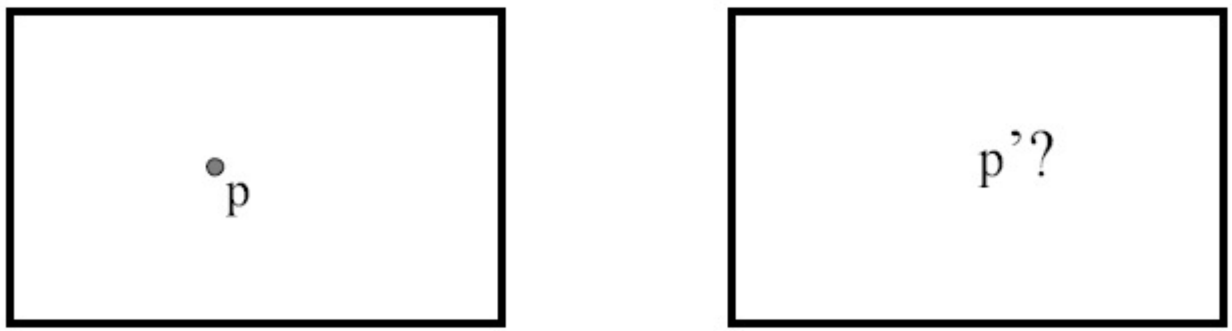
What is the relationship between the camera + the corresponding points?

Where do epipolar lines come from?



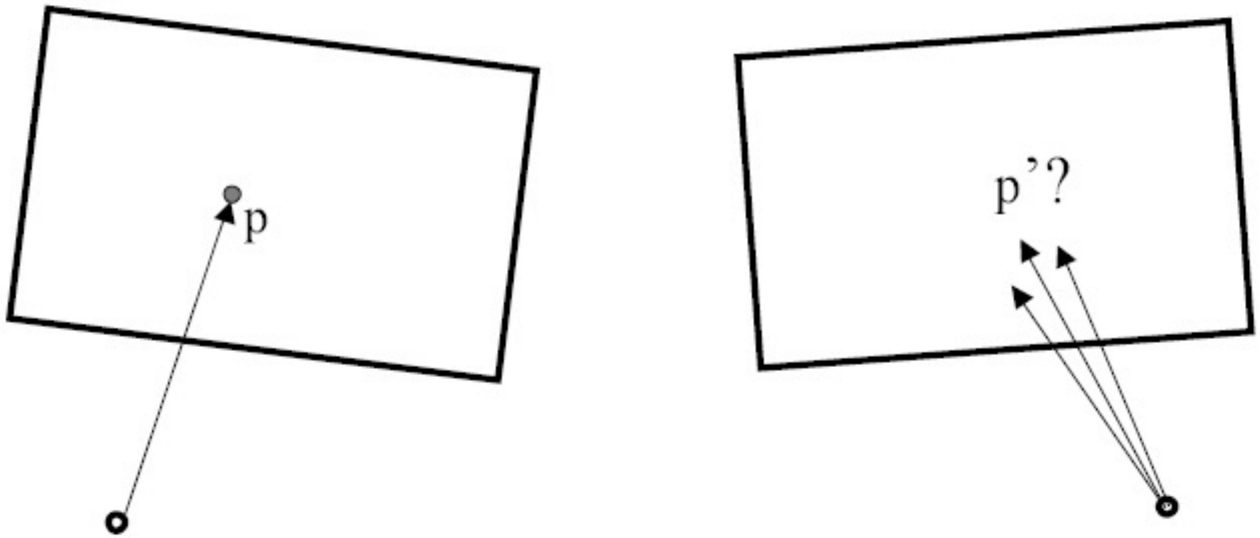
Slide courtesy of Noah Snavely

Stereo correspondence constraints



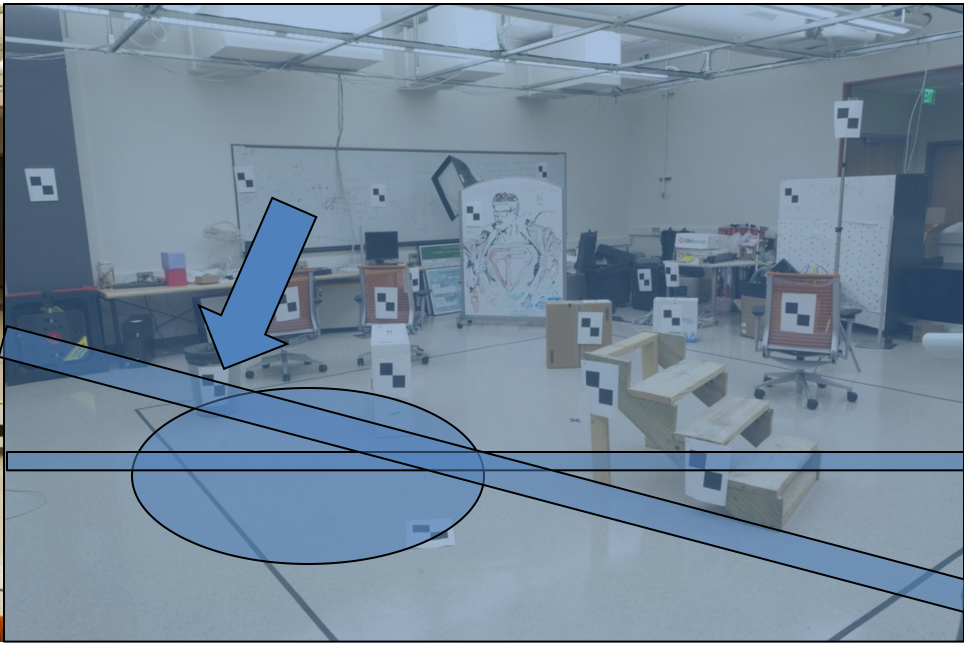
- Given p in left image, where can corresponding point p' be?

Stereo correspondence constraints



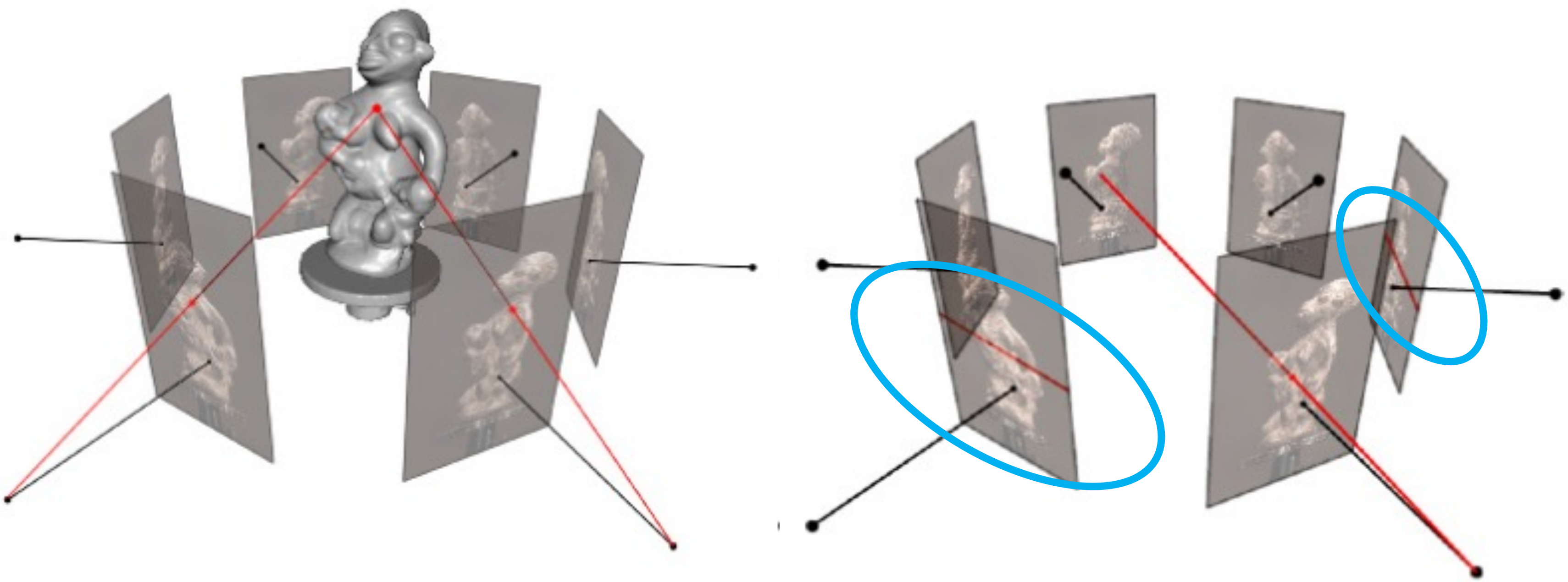
- Given p in left image, where can corresponding point p' be?

Where do we need to search?



Slide by James Hays

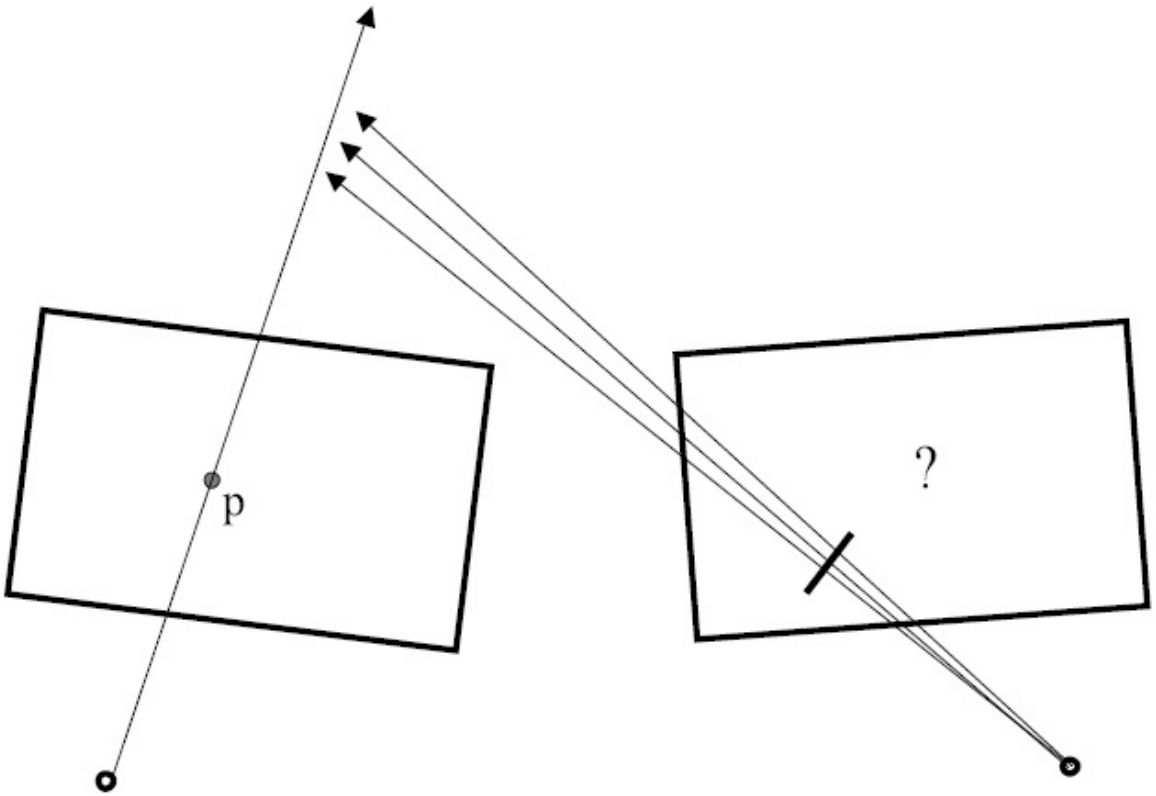
Epipolar Geometry



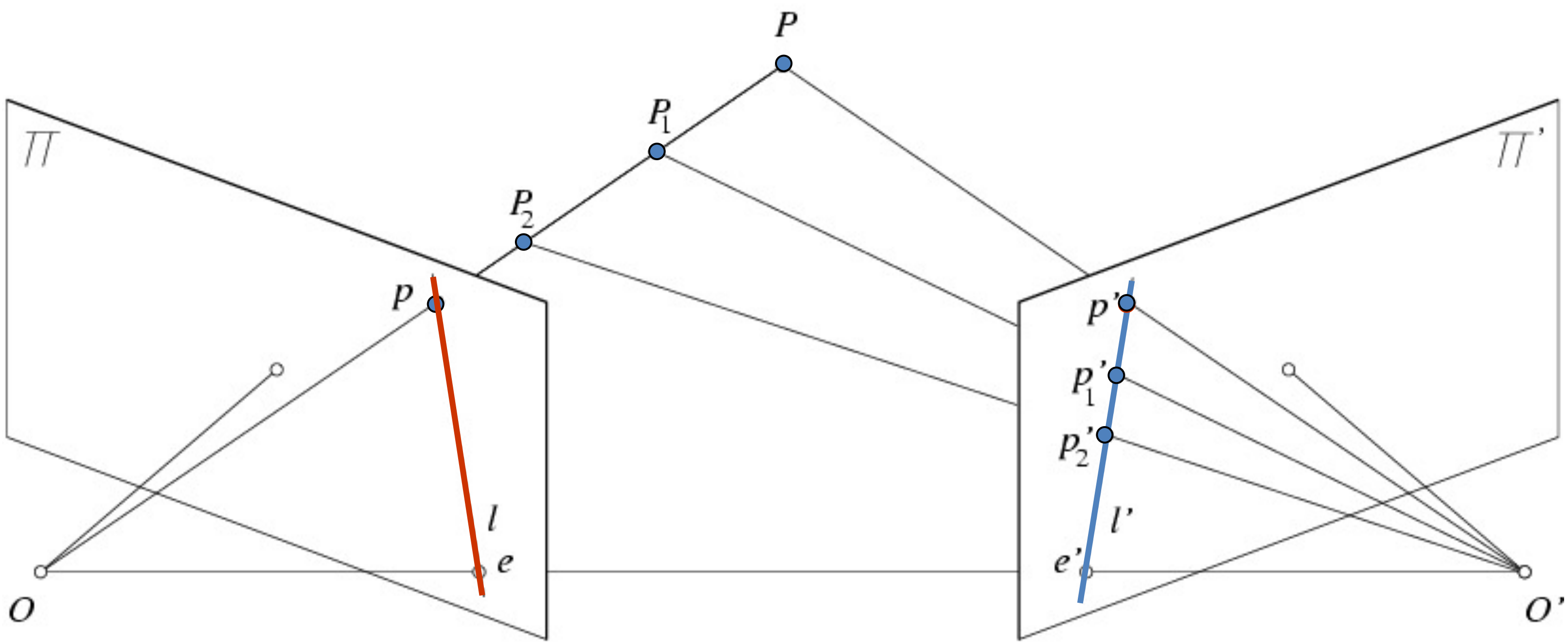
Figures by Carlos Hernandez

If you get confused with the following math, look at this picture again, it just describes this.

Stereo correspondence constraints



Epipolar constraint

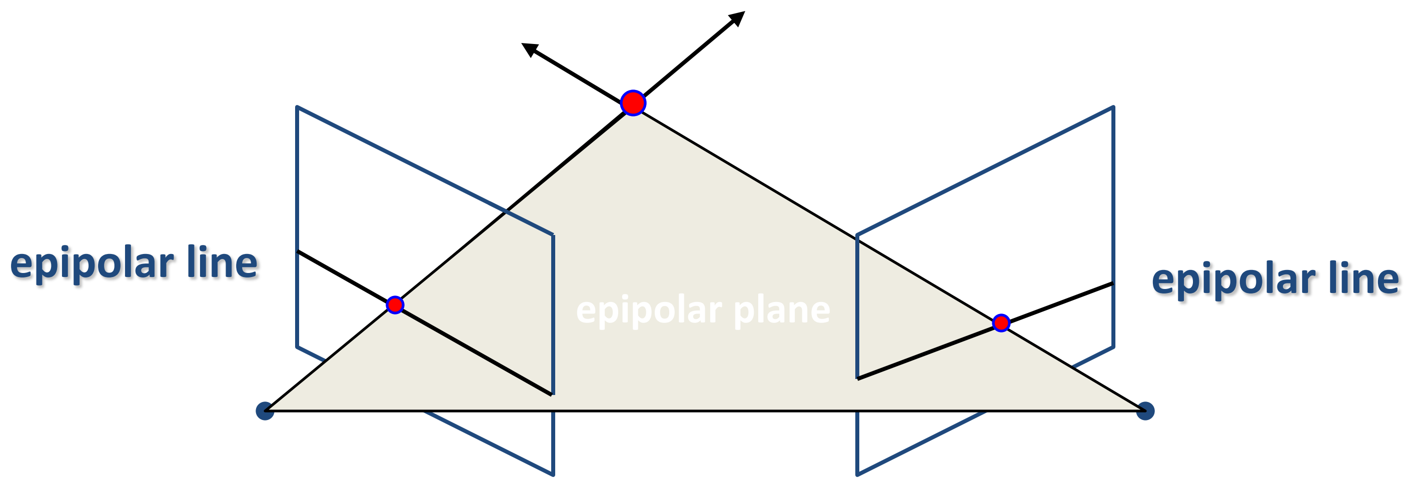


- Potential matches for *p* have to lie on the corresponding epipolar line *l'*.
- Potential matches for *p'* have to lie on the corresponding epipolar line *l*.

Source: M. Pollefeys

Stereo correspondence constraints

•Geometry of two views allows us to constrain where the corresponding pixel for some image point in the first view must occur in the second view.



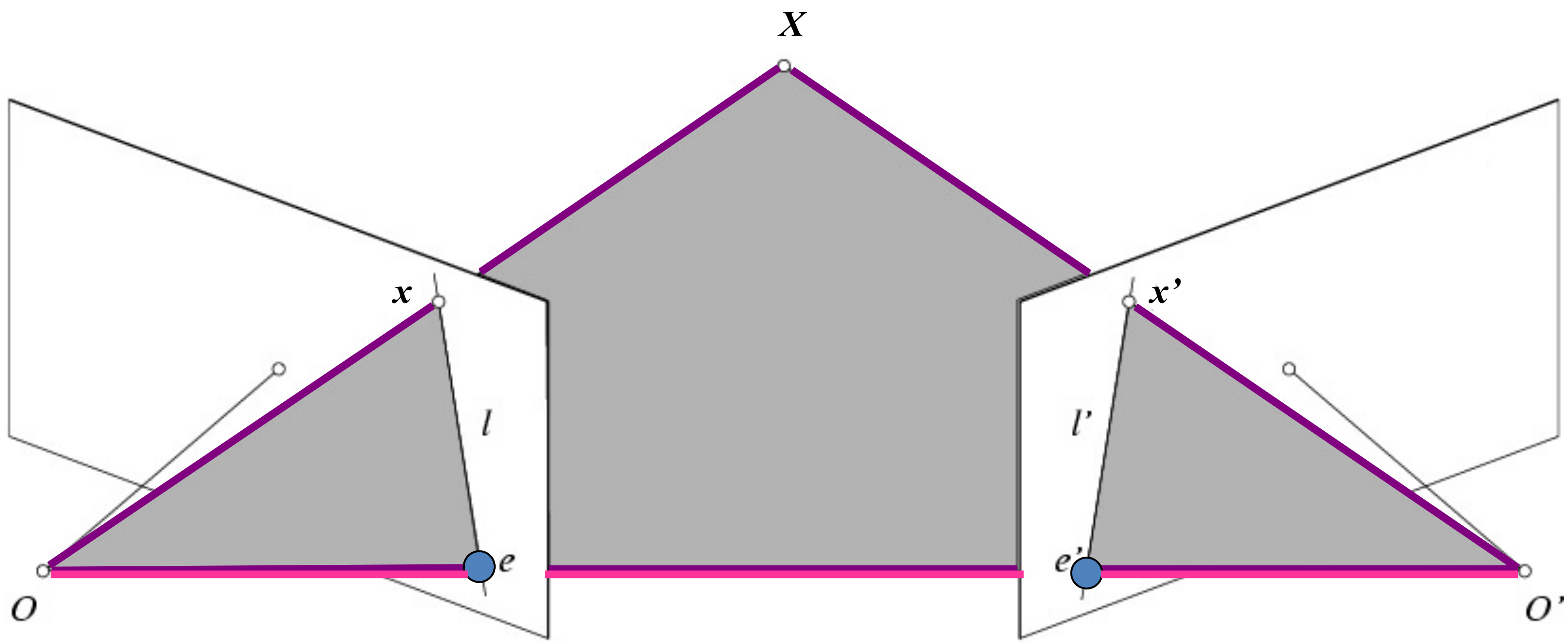
Epipolar constraint: Why is this useful?

- Reduces correspondence problem to 1D search along *conjugate epipolar lines*

http://www.ai.sri.com/~luong/research/Meta3DViewer/EpipolarGeo.html

Adapted from Steve Seitz

Parts of Epipolar geometry



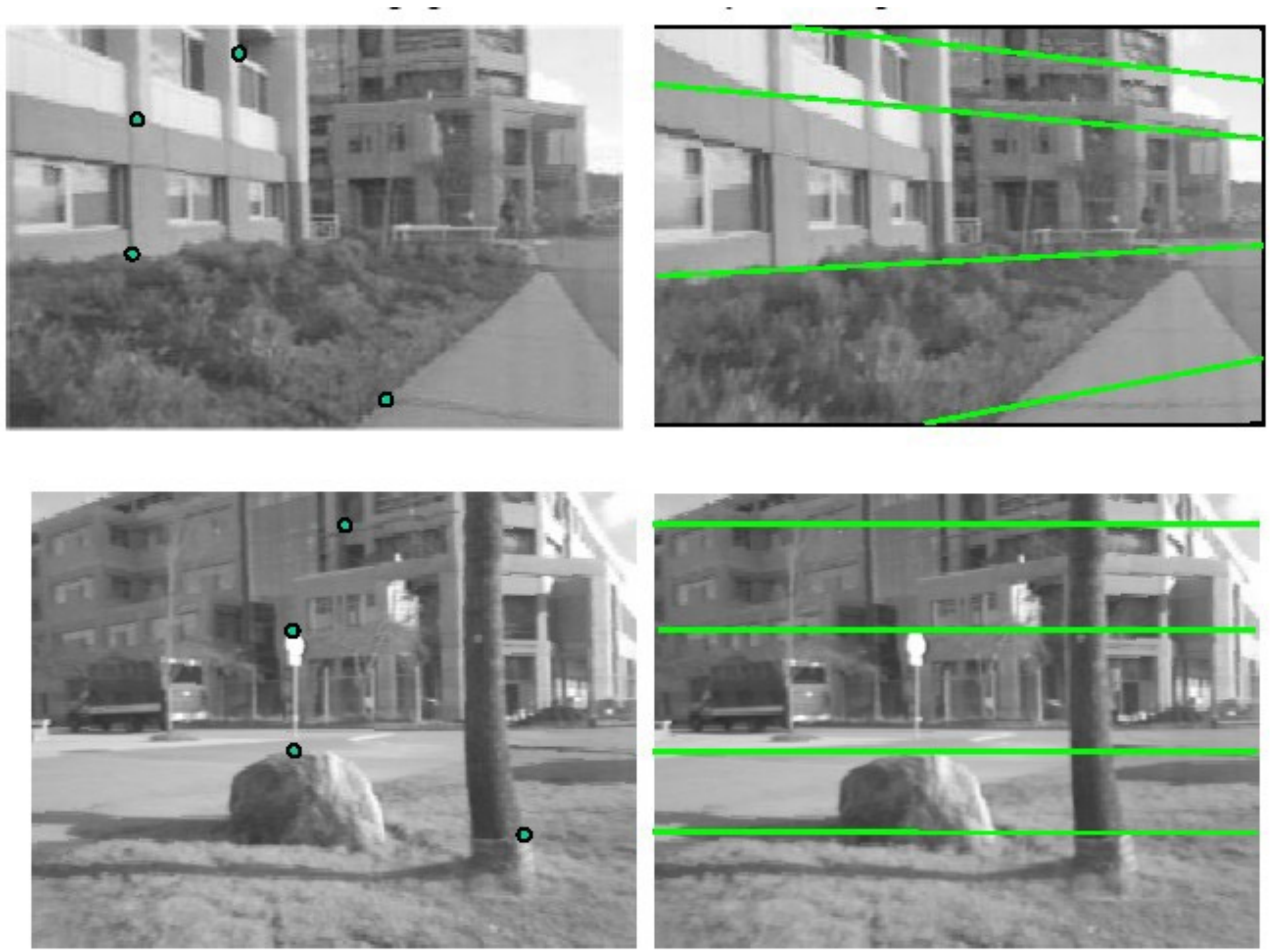
- **Baseline** – line connecting the two camera centers
- **Epipolar Plane** – plane containing baseline (1D family)
- **Epipoles**

= intersections of baseline with image planes
= projections of the other camera center
= vanishing points of the baseline

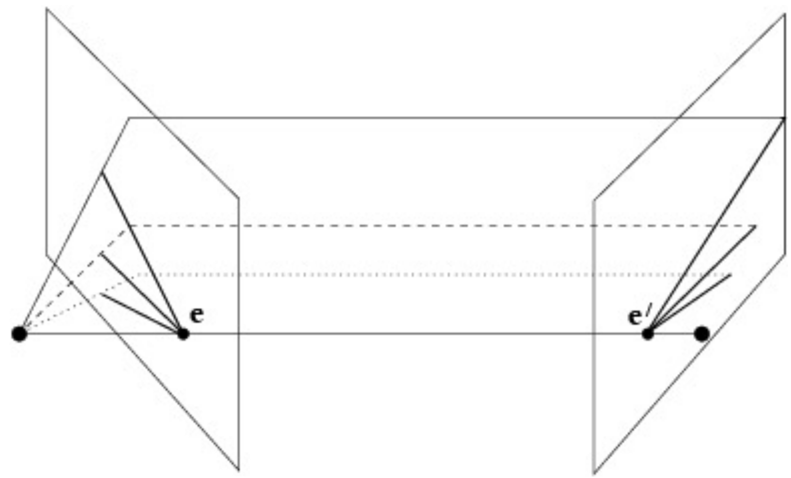
The Epipole

Photo by Frank Dellaert

Example



Example: converging cameras



As position of 3d point varies, epipolar lines "rotate" about the baseline

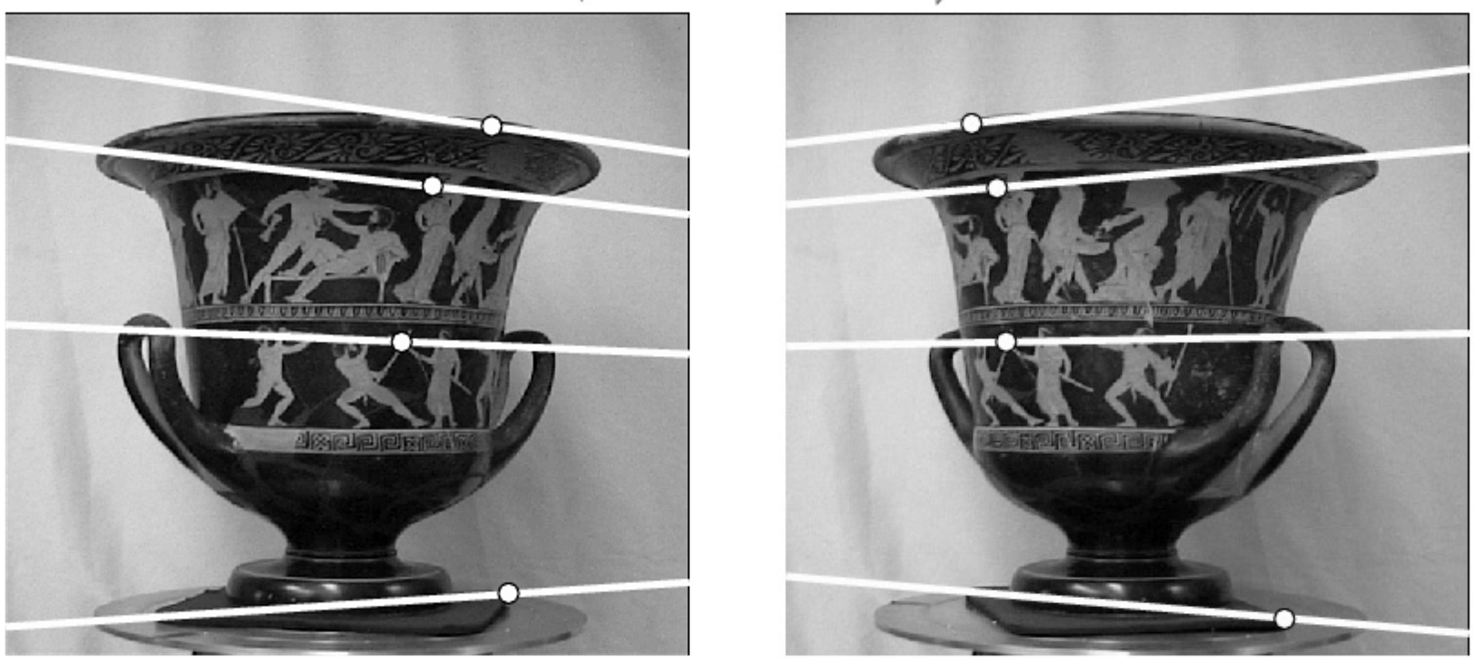
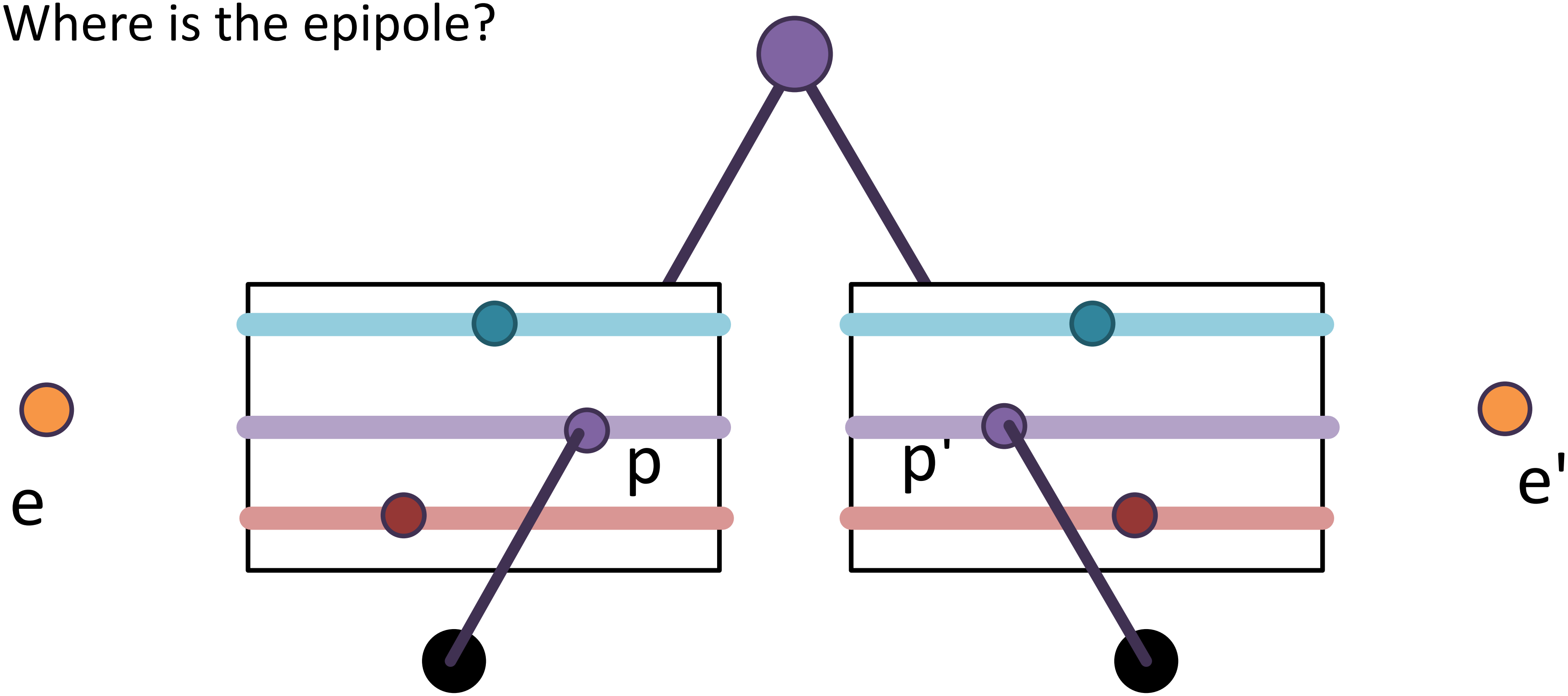


Figure from Hartley & Zisserman

Example: Parallel to Image Plane

Where is the epipole?

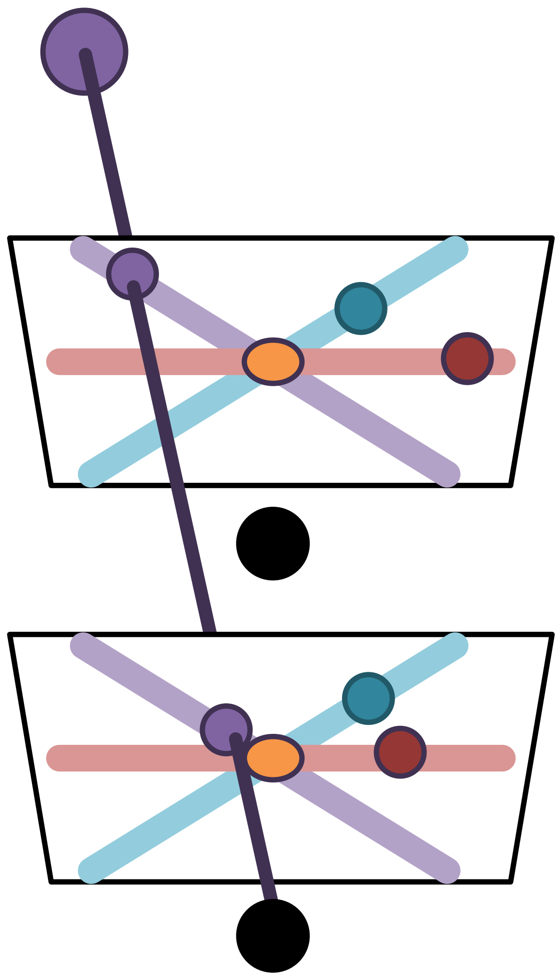


Epipoles *infinitely* far away, epipolar lines parallel

Slide credit: David Fouhey

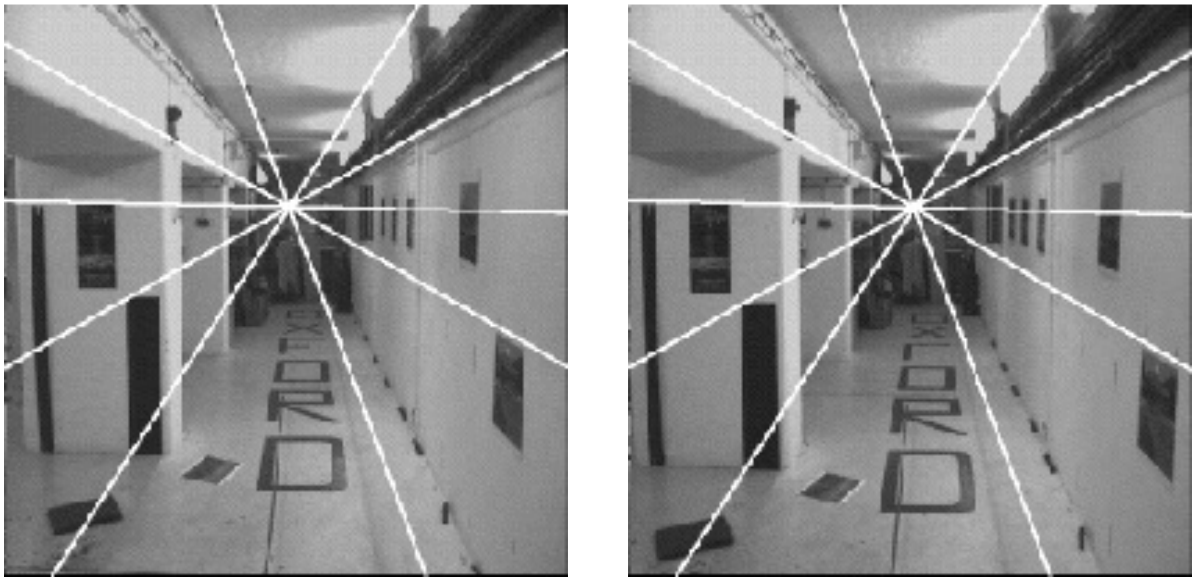
Example: Forward Motion

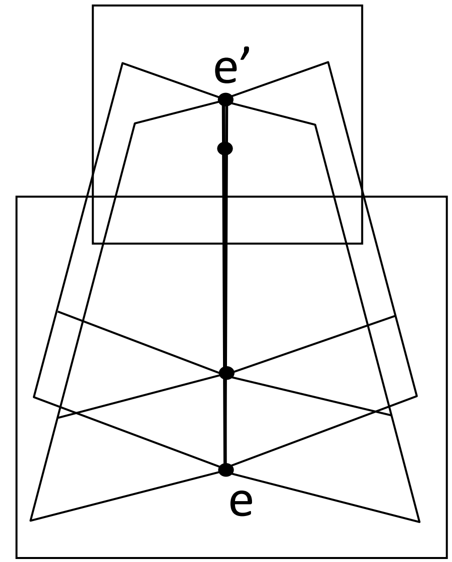
Epipole is focus of expansion / principal point of the camera.



Epipolar lines go out from principal point

Example: forward motion

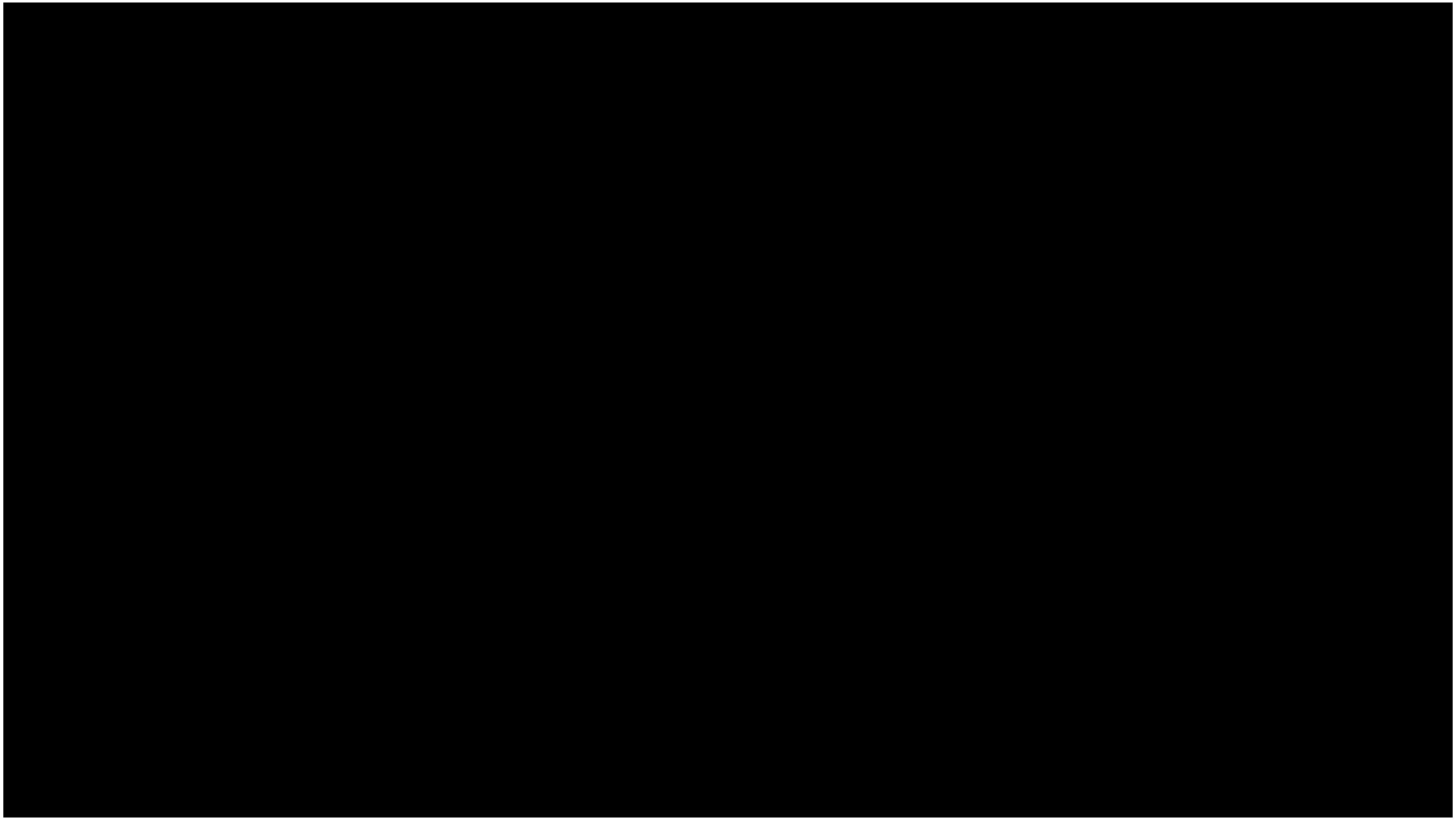




Epipole has same coordinates in both images.
Points move along lines radiating from e: “Focus of expansion”

Figure from Hartley & Zisserman

Motion perpendicular to image plane



http://vimeo.com/48425421

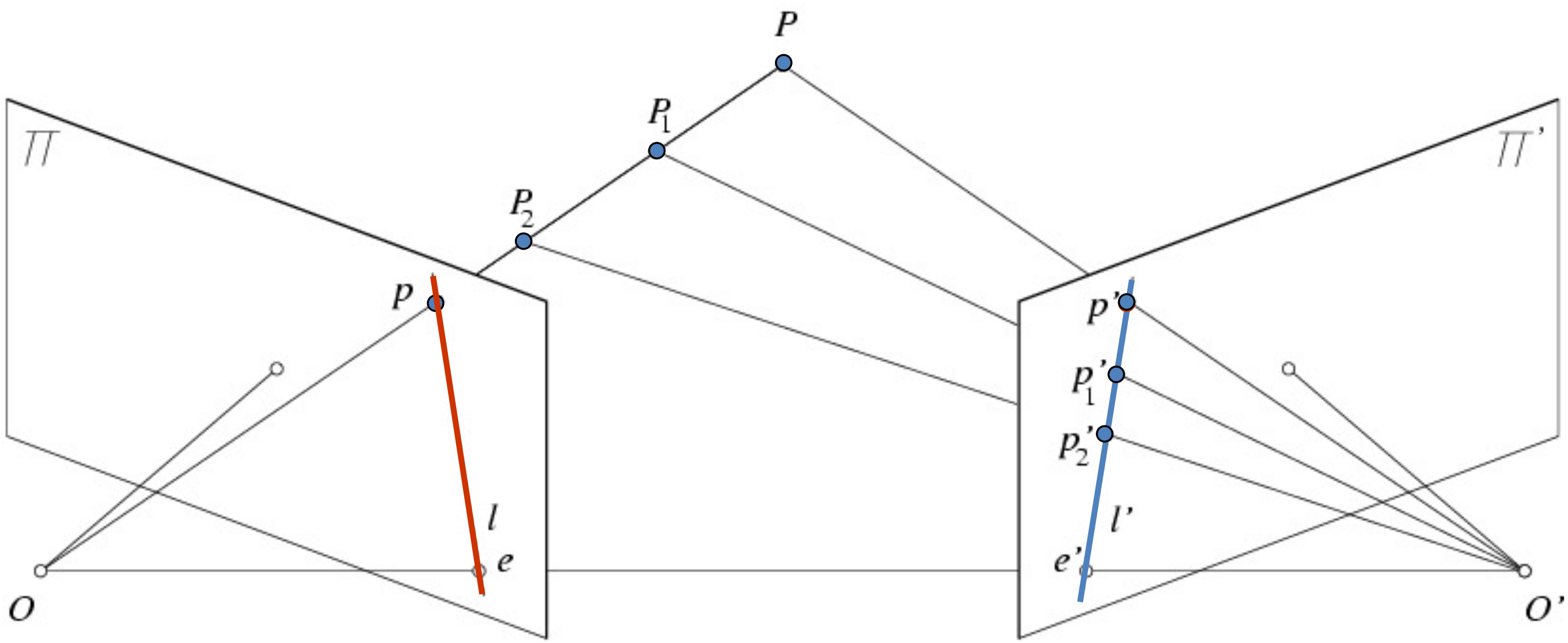
Slide credit: David Fouhey

Ok so where were we?

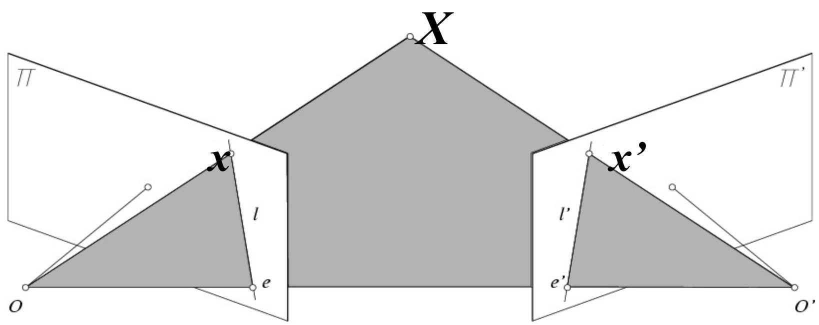
- Setup: Calibrated Camera (both extrinsic & intrinsic)
- Goal: 3D reconstruction of corresponding points in the image
- We need to find correspondences!

➔ 1D search along the epipolar line!

Ok so what exactly are l and l'?



Step 0: Normalized image coordinates



$$x = K[R \;\; t]X$$

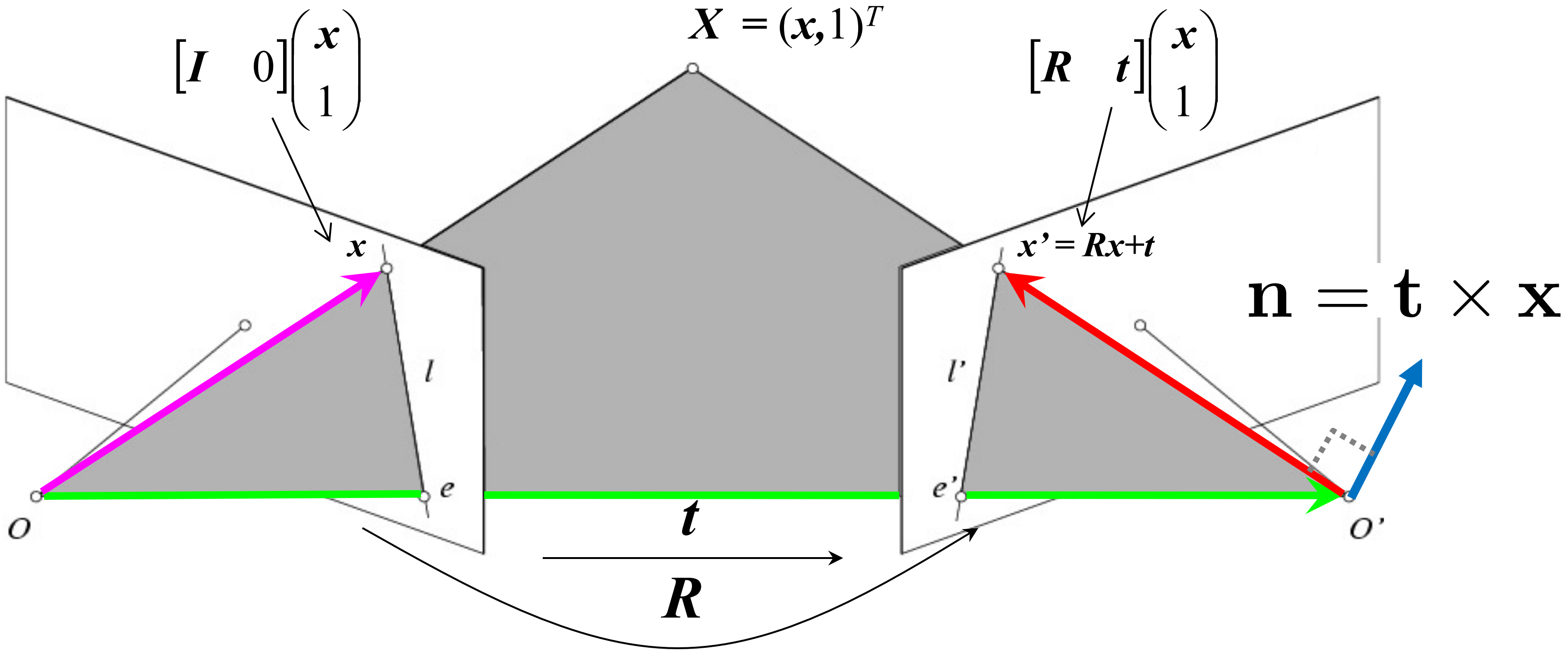
$$K^{-1}x = [R \;\; t]X$$

- Let's factor out the effect of K (do everything in 3D)
- Since we know the intrinsics K, apply its inverse to x with depth = 1
- This is called the *normalized* image coordinates. It may be thought of as a set of points with K = Identity

$$\boldsymbol{x}_{\text{norm}} = \boldsymbol{K}^{-1}\boldsymbol{x}_{\text{pixel}} = [\boldsymbol{I} \;\; 0]\boldsymbol{X}, \qquad \boldsymbol{x}'_{\text{norm}} = \boldsymbol{K}'^{-1}\boldsymbol{x}'_{\text{pixel}} = [\boldsymbol{R} \;\; \boldsymbol{t}]\boldsymbol{X}$$

- Assume that the points are normalized from here on

Epipolar constraint: Calibrated case



The vectors *x*, *t*, and *x'* are coplanar

What can you say about their relationships, given $\mathbf{n} = \mathbf{t} \times \mathbf{x'}$?

$$\boxed{\mathbf{x'} \cdot (\mathbf{t} \times \mathbf{x'}) = 0}$$

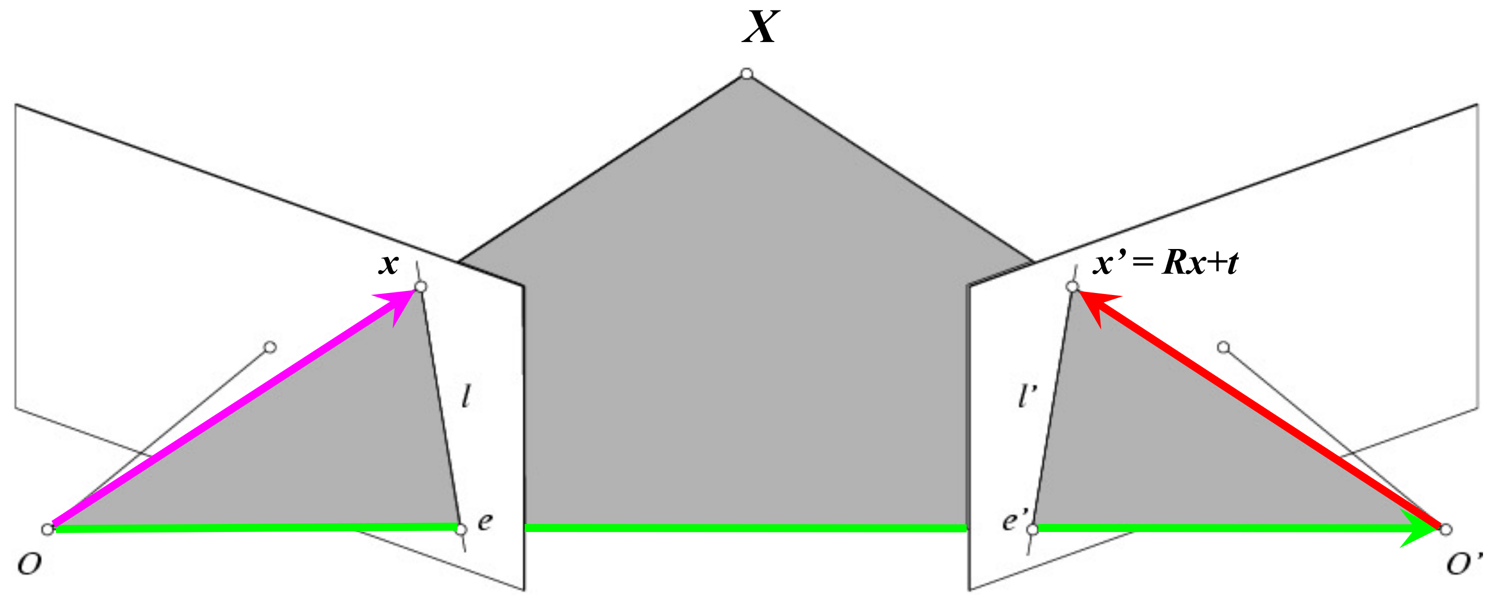
$$\mathbf{x'} \cdot (\mathbf{t} \times (R\mathbf{x} + \mathbf{t})) = 0$$

$$\mathbf{x'} \cdot (\mathbf{t} \times R\mathbf{x} + \mathbf{t} \times \mathbf{t})) = 0$$

0

$$\boxed{\mathbf{x'} \cdot (\mathbf{t} \times R\mathbf{x}) = 0}$$

Epipolar constraint: Calibrated case

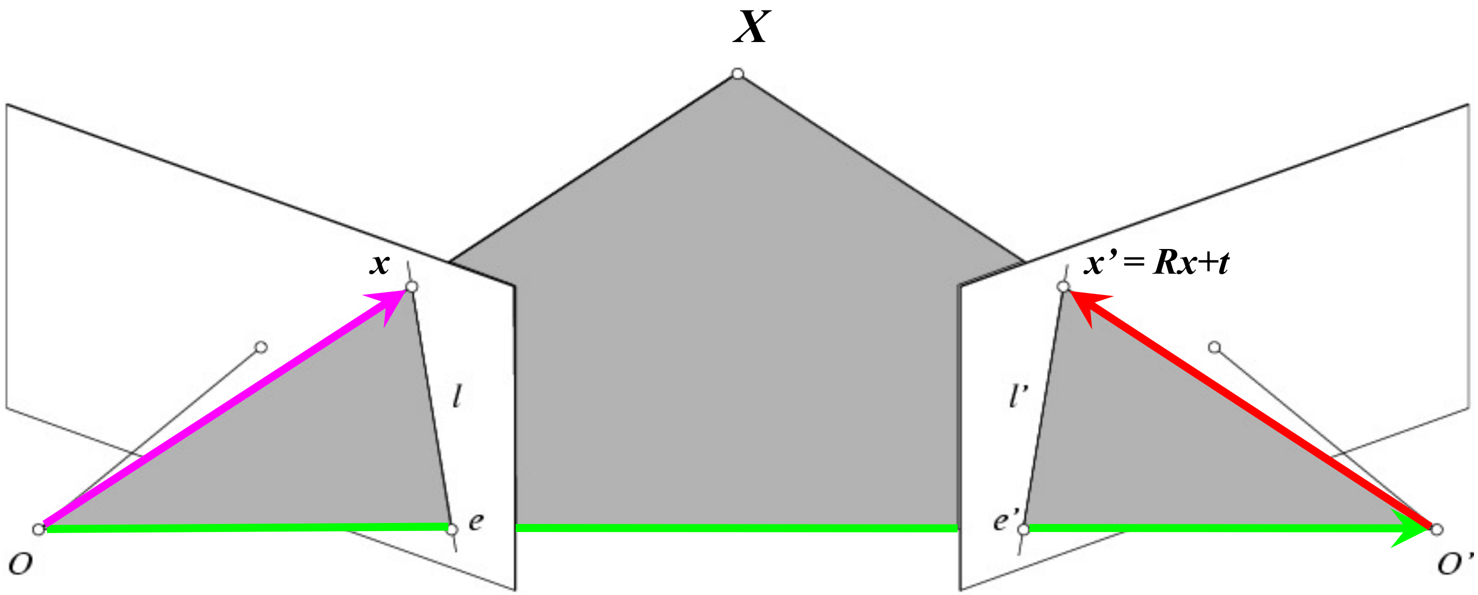


$$\boldsymbol{x}' \cdot [\boldsymbol{t} \times (\boldsymbol{R}\boldsymbol{x})] = 0 \quad \Rightarrow \quad \boldsymbol{x}'^{T} [\boldsymbol{t}_{\times}] \boldsymbol{R}\boldsymbol{x} = 0$$

Recall: $\mathbf{a} \times \mathbf{b} = \begin{bmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{bmatrix} \begin{bmatrix} b_x \\ b_y \\ b_z \end{bmatrix} = [\mathbf{a}_{\times}]\mathbf{b}$

The vectors $\boldsymbol{x}$, $\boldsymbol{t}$, and $\boldsymbol{x}'$ are coplanar

Epipolar constraint: Calibrated case



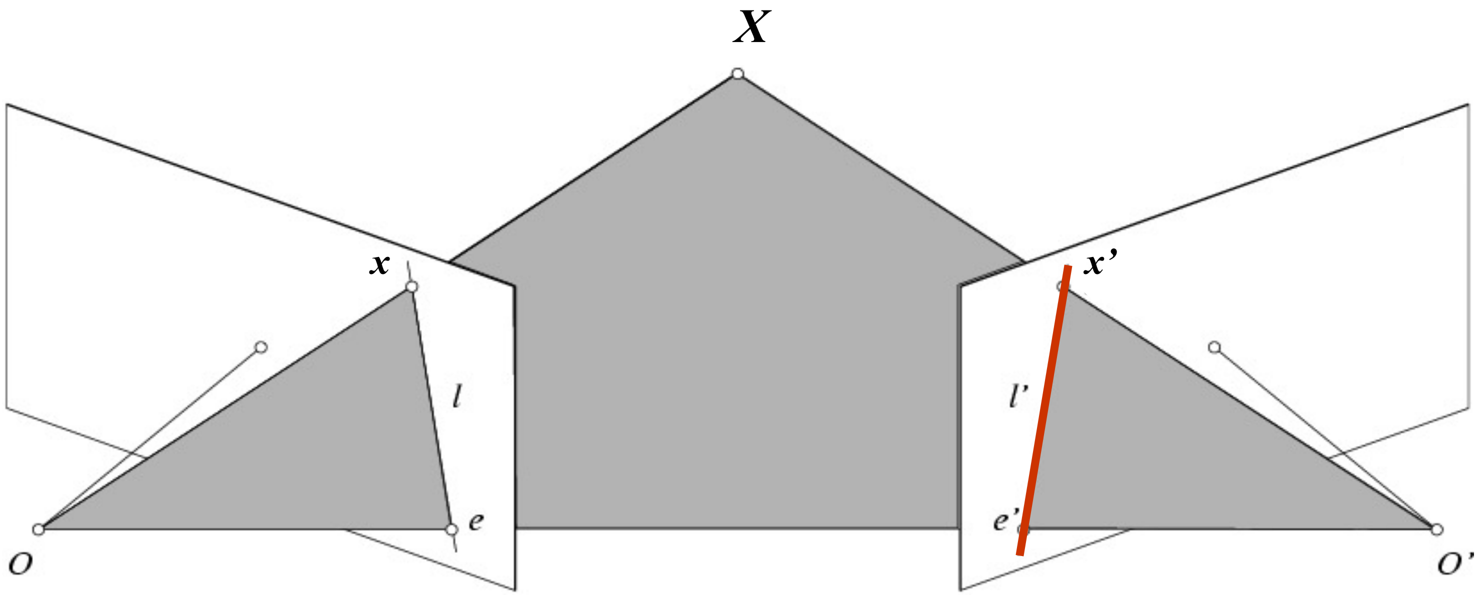
$$\boldsymbol{x}' \cdot [\boldsymbol{t} \times (\boldsymbol{R}\boldsymbol{x})] = 0 \quad \Rightarrow \quad \boldsymbol{x}'^{T} \underbrace{[\boldsymbol{t}_{\times}]\boldsymbol{R}}_{E} \boldsymbol{x} = 0 \quad \Rightarrow \quad \boldsymbol{x}'^{T} E \boldsymbol{x} = 0$$

Recall: $\mathbf{a} \times \mathbf{b} = \begin{bmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{bmatrix} \begin{bmatrix} b_x \\ b_y \\ b_z \end{bmatrix} = [\mathbf{a}_{\times}]\mathbf{b}$

Essential Matrix
(Longuet-Higgins, 1981)

The vectors $\boldsymbol{x}$, $\boldsymbol{t}$, and $\boldsymbol{x}'$ are coplanar

Epipolar constraint: Calibrated case

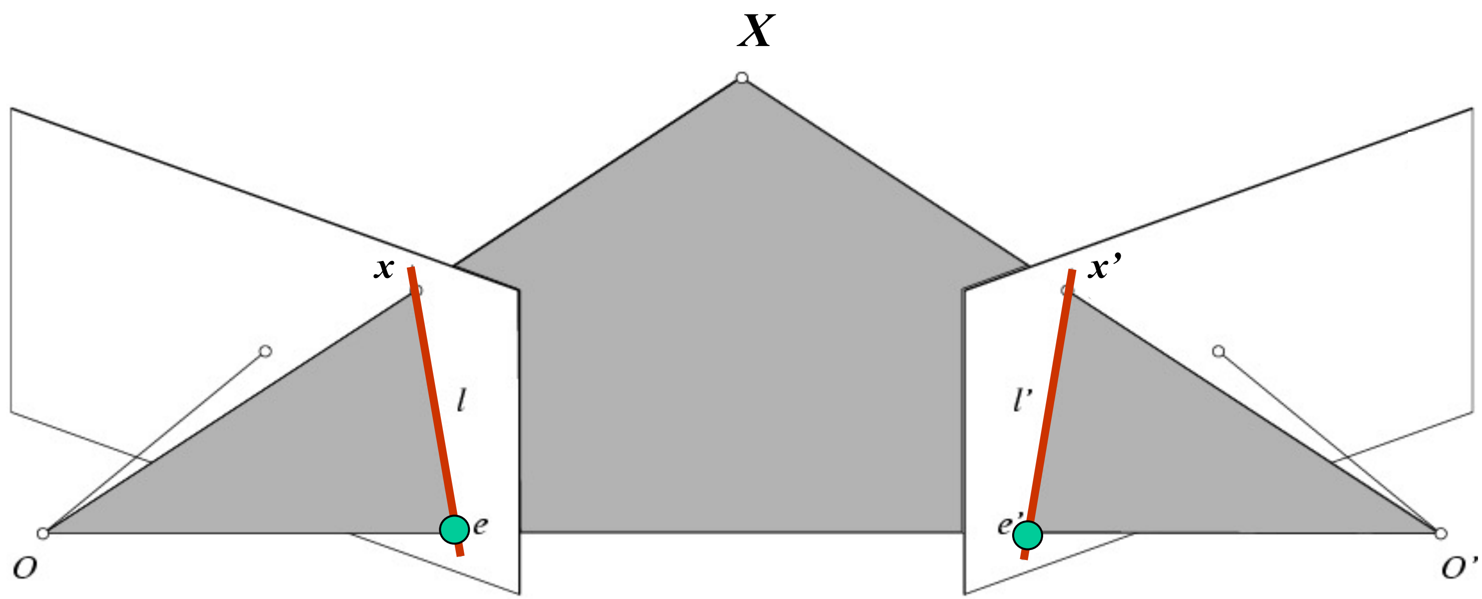


$$\boldsymbol{x}'^{T} E \boldsymbol{x} = 0$$

- ***E x*** is the epipolar line associated with ***x*** (***l'*** = ***E x***)
  - Recall: a line is given by $ax + by + c = 0$ or

$$\mathbf{l}^T \mathbf{x} = 0 \quad \text{where} \quad \mathbf{l} = \begin{bmatrix} a \\ b \\ c \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Epipolar constraint: Calibrated case



$$\boldsymbol{x}'^T E \boldsymbol{x} = 0$$

- ***E x*** is the epipolar line associated with ***x*** (***l'*** = ***E x***)
- ***E****^{T}**x'*** is the epipolar line associated with ***x'*** (***l*** = ***E****^{T}**x'***)
- ***E* e** = 0 and ***E****^{T}***e'** = *0*
- ***E*** is singular (rank two)
- ***E*** has five degrees of freedom

Epipolar constraint: Uncalibrated case

- Recall that we normalized the coordinates

$$x = K^{-1}\hat{x} \quad x' = K'^{-1}\hat{x}'$$

$$\hat{x} = \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}$$

where $\hat{x}$ is the image coordinates

- But in the *uncalibrated* case, ***K*** and ***K'*** are unknown!
- We can write the epipolar constraint in terms of *unknown* normalized coordinates:

$$x'^T E x = 0$$

$$(K'^{-1}\hat{x}')'^T E (K^{-1}\hat{x}) = 0$$

$$\hat{x}'^T \underbrace{K'^{-T} E K^{-1}}\hat{x} = 0$$

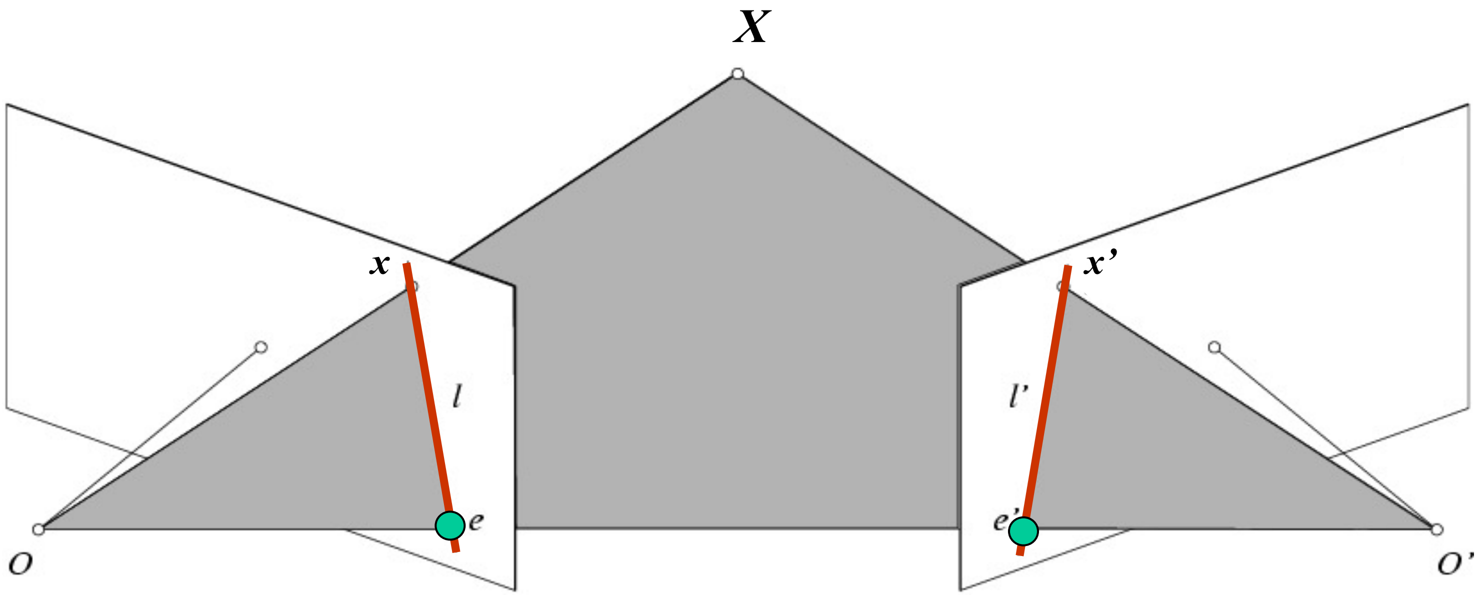
$$\hat{x}'^T F \hat{x} = 0$$

$$F = K'^{-T} E K^{-1}$$

Fundamental Matrix

(Faugeras and Luong, 1992)

Epipolar constraint: Uncalibrated case



$$x'^T E x = 0 \quad \Rightarrow \quad \hat{x}'^T F \hat{x} = 0 \quad \text{with} \quad F = K'^{-T} E K^{-1}$$

- $\boldsymbol{F}\hat{\boldsymbol{x}}$ is the epipolar line associated with $\hat{\boldsymbol{x}}$ ($\boldsymbol{l}' = \boldsymbol{F}\hat{\boldsymbol{x}}$)
- $\boldsymbol{F}^T\hat{\boldsymbol{x}}'$ is the epipolar line associated with $\hat{\boldsymbol{x}}'$ ($\boldsymbol{l} = \boldsymbol{F}^T\hat{\boldsymbol{x}}'$)
- $\boldsymbol{F}\mathbf{e} = 0$ and $\boldsymbol{F}^T\mathbf{e}' = 0$
- ***F*** is singular (rank two)
- ***F*** has *seven* degrees of freedom

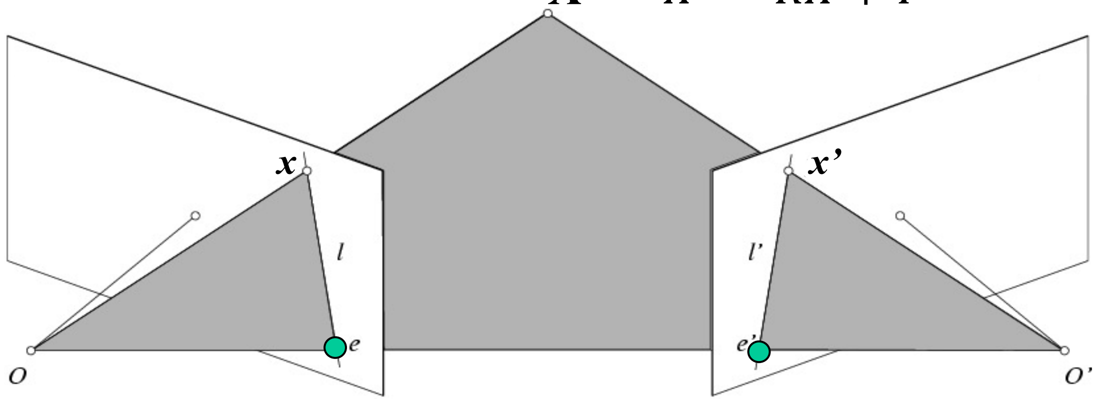
Where are we? (in the original setup)

We have two images with calibrated cameras, want the 3D points!

1. Solve for correspondences using epipolar constraints from known camera (1D search)
   - **Now we know the exact equation of this line**
2. Triangulate to get depth!

Finally: computing depth by triangulation

We know about the camera, K_1, K_2 and [R t]:



and found the corresponding points: $x \leftrightarrow x'$

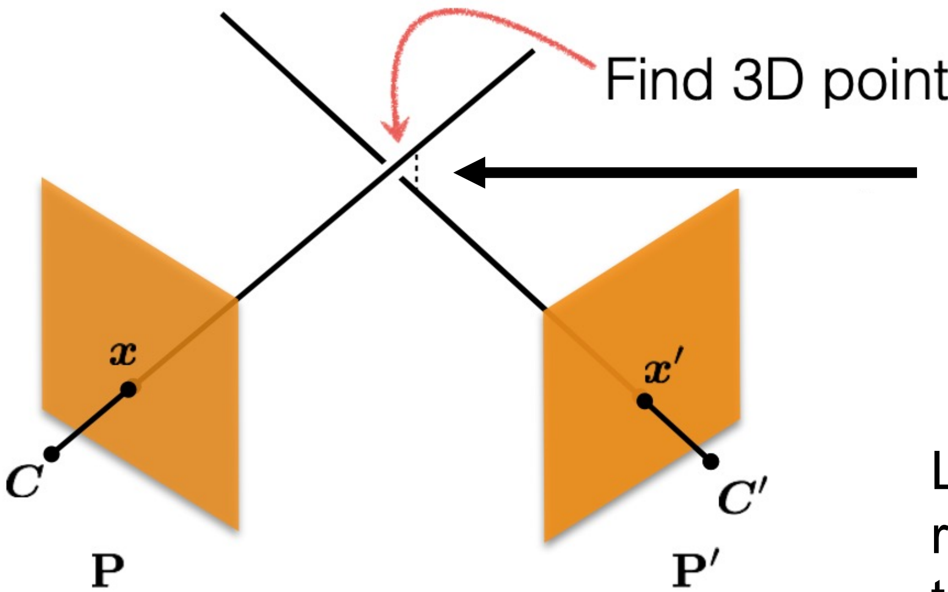
$$x = KX \qquad x' = K'X'$$
$$= K'(RX + T)$$

only unknowns!

How many unknowns
+ how many equations
do we have?

Solve by least squares

Triangulation Disclaimer: Noise



Ray's don't always intersect because of noise!!!

$$\mathbf{X} \text{ s.t.}$$

$$\mathbf{x} = \mathbf{P}\mathbf{X}, \ \mathbf{x}' = \mathbf{P}'\mathbf{X}$$

Least squares get you to a reasonable solution but it's not the actual geometric error (it's how far away the solution is from Ax = 0)

In practice with noise, you do non-linear least squares, or **"bundle adjustment"** (more than 2 image case, next lecture..)

Slide credit: Shubham Tulsiani

Summary: Two-view, known camera

0. Calibrate the camera.

1. Find correspondences:

   - Reduce this to 1D search with Epipolar Geometry!

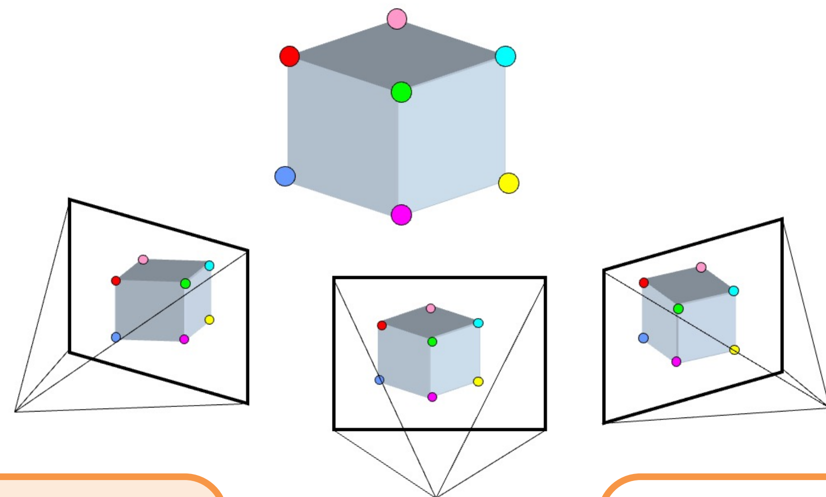
2. Get depth:

   - If simple stereo, disparity (difference of corresponding points) is inversely proportional to depth

   - In the general case, triangulate.

What if we don't know the camera?

3D Points
(Structure)



Correspondences → Camera
(Motion)

What if we don’t know the camera?

Assume we know the correspondences:

$\hat{x}'$ and $\hat{x}$ in the image

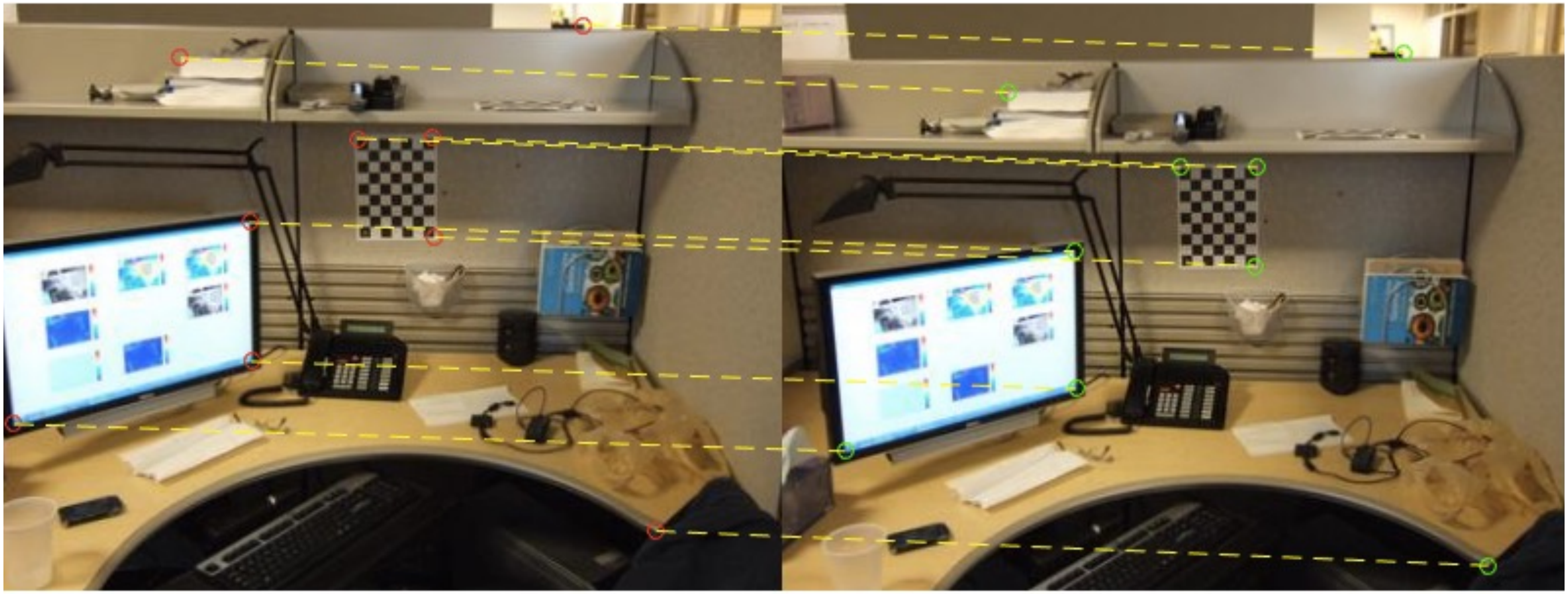
$$\hat{x}'^T F \hat{x} = 0$$

$$\hat{x} = \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} u' & v' & 1 \end{bmatrix} \begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = 0$$

How many correspondences do we need?

Estimating the fundamental matrix



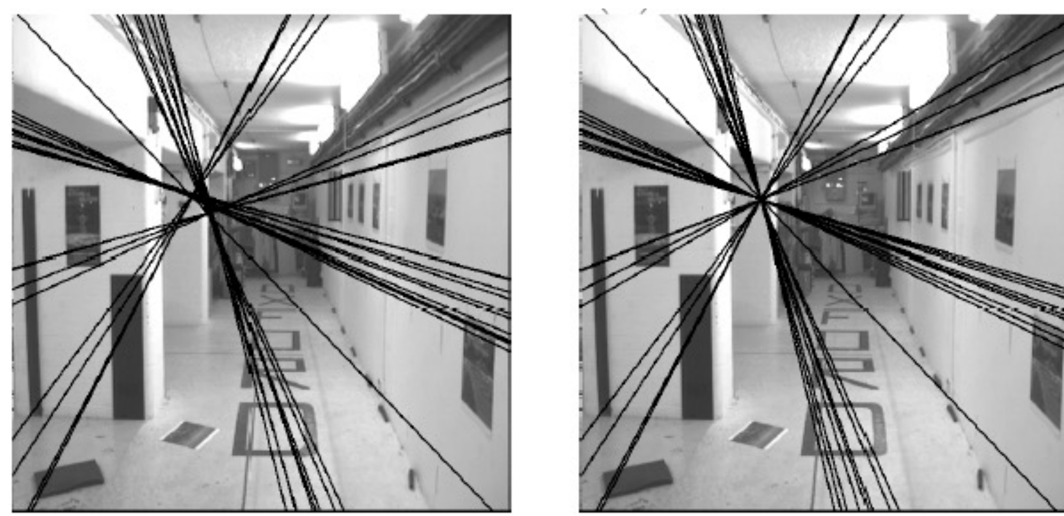
The eight-point algorithm

$$\boldsymbol{x} = (u, v, 1)^T, \quad \boldsymbol{x}' = (u', v', 1)$$

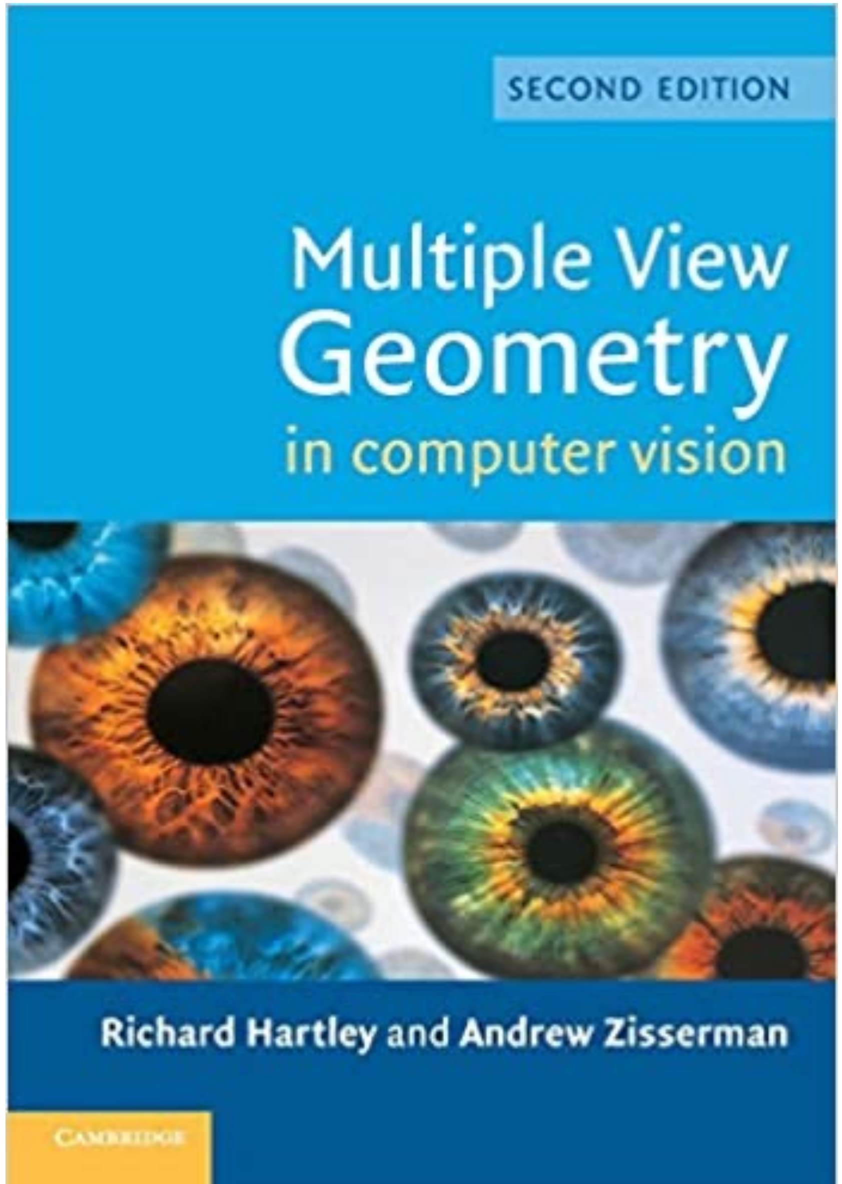
$$\begin{bmatrix} u' & v' & 1 \end{bmatrix} \begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = 0 \;\Rightarrow\; \begin{bmatrix} u'u & u'v & u' & v'u & v'v & v' & u & v & 1 \end{bmatrix} \begin{bmatrix} f_{11} \\ f_{12} \\ f_{13} \\ f_{21} \\ f_{22} \\ f_{23} \\ f_{31} \\ f_{32} \\ f_{33} \end{bmatrix} = 0$$

Solve homogeneous linear system using eight or more matches

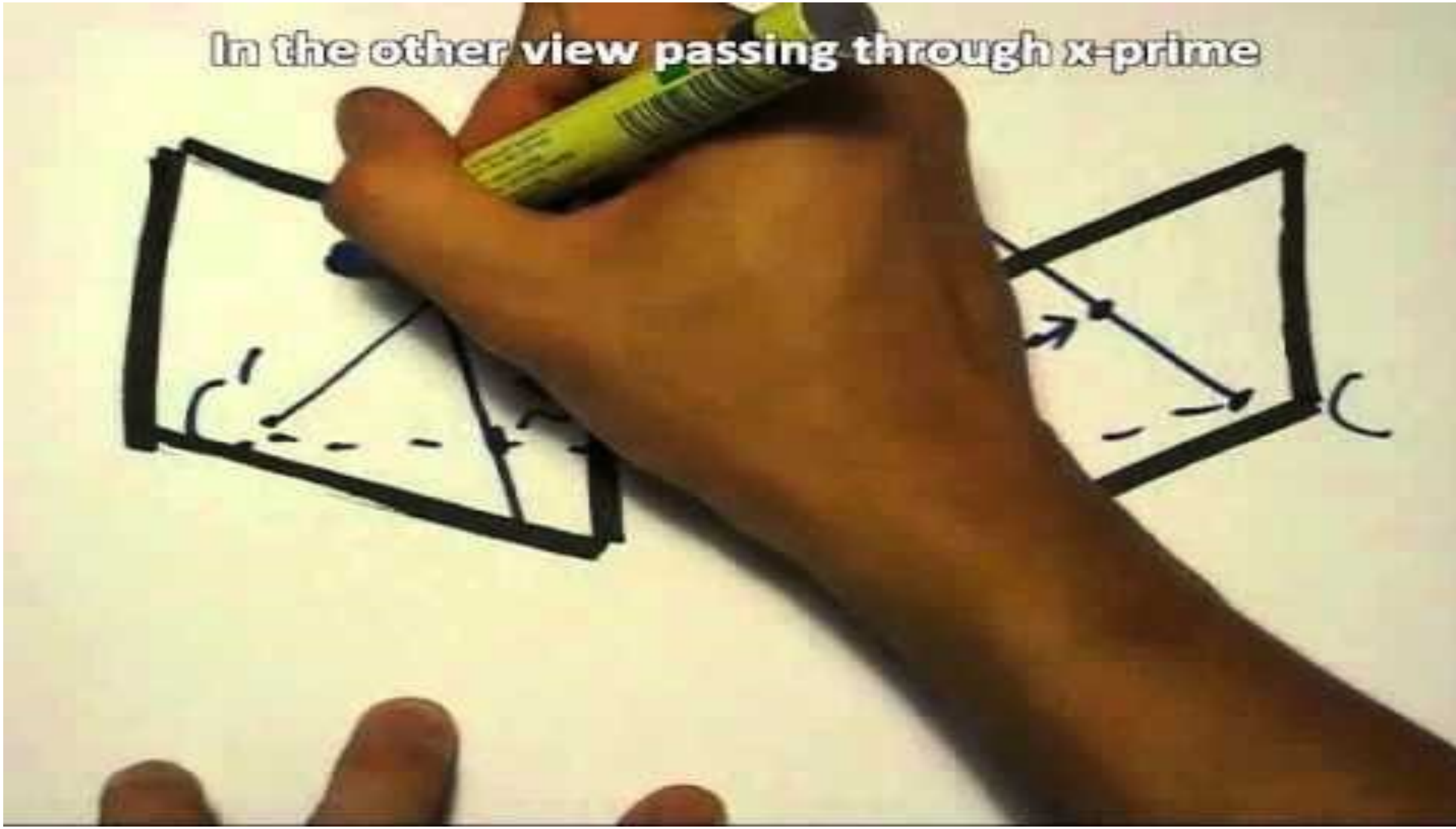
Enforce rank-2 constraint (take SVD of ***F*** and throw out the smallest singular value)



The Bible by Hartley & Zisserman



The Fundamental Matrix Song



http://danielwedge.com/fmatrix/
https://www.youtube.com/watch?time_continue=8&v=DgGV3l82NTk&feature=emb_title

Going from F to the Camera

Get the essential matrix with K (or some estimates of K)

$$E = K'^{T} F K.$$

How the 2D lines relate with 3D lines is captured by intrinsics!

Essential matrix can be decomposed

$E = T_x R$

If we know E, we can recover t and R

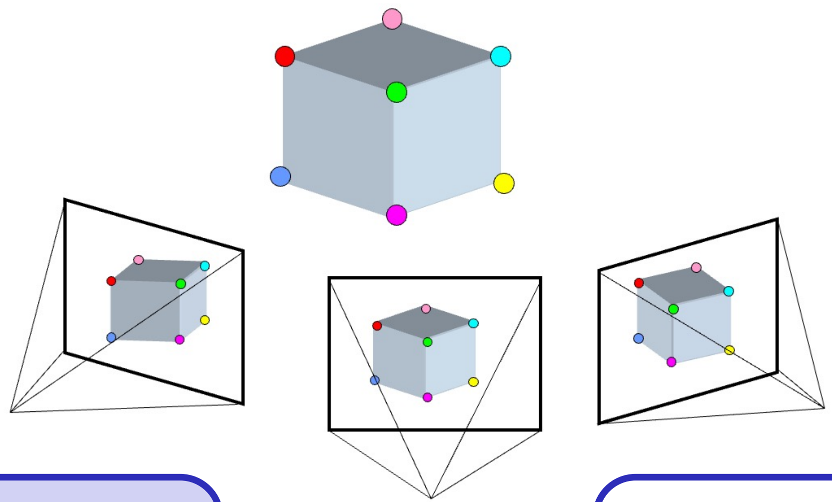
$$\begin{bmatrix} e_{11} & e_{12} & e_{13} \\ e_{21} & e_{22} & e_{23} \\ e_{31} & e_{32} & e_{33} \end{bmatrix} = \begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$$

Given that $T_\times$ is a Skew-Symmetric matrix (a_{ij} = - a_{ji}) and R is an Orthonormal matrix, it is possible to “decouple” $T_\times$ and R from their product using “Singular Value Decomposition”.

Slide from Shree Nayar

This completes: Corresp to Camera

3D Points
(Structure)



Correspondences

Camera
(Motion)

What about more than two views?

The geometry of three views is described by a 3 x 3 x 3 tensor called the *trifocal tensor*

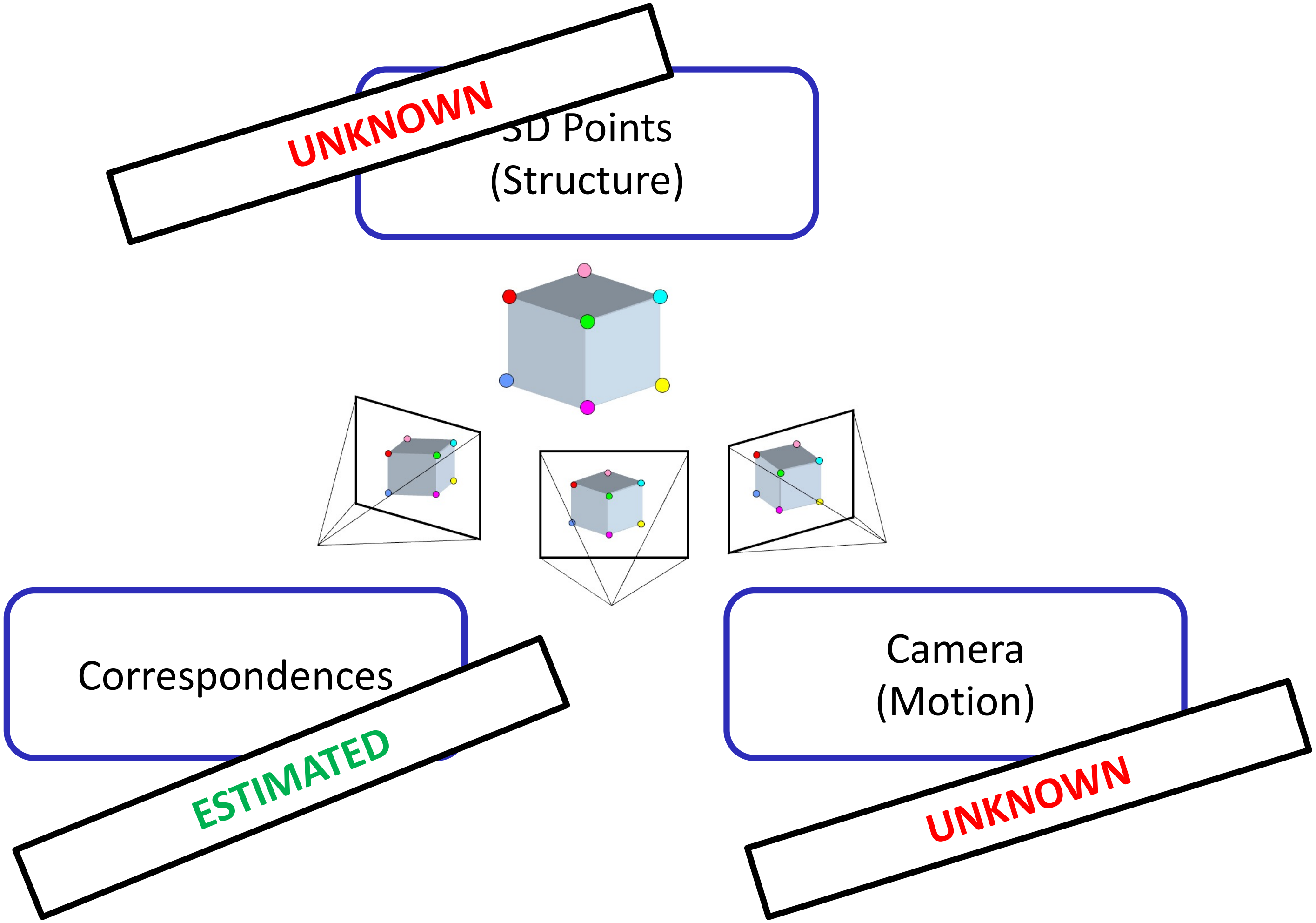
The geometry of four views is described by a 3 x 3 x 3 x 3 tensor called the *quadrifocal tensor*

After this it starts to get complicated…

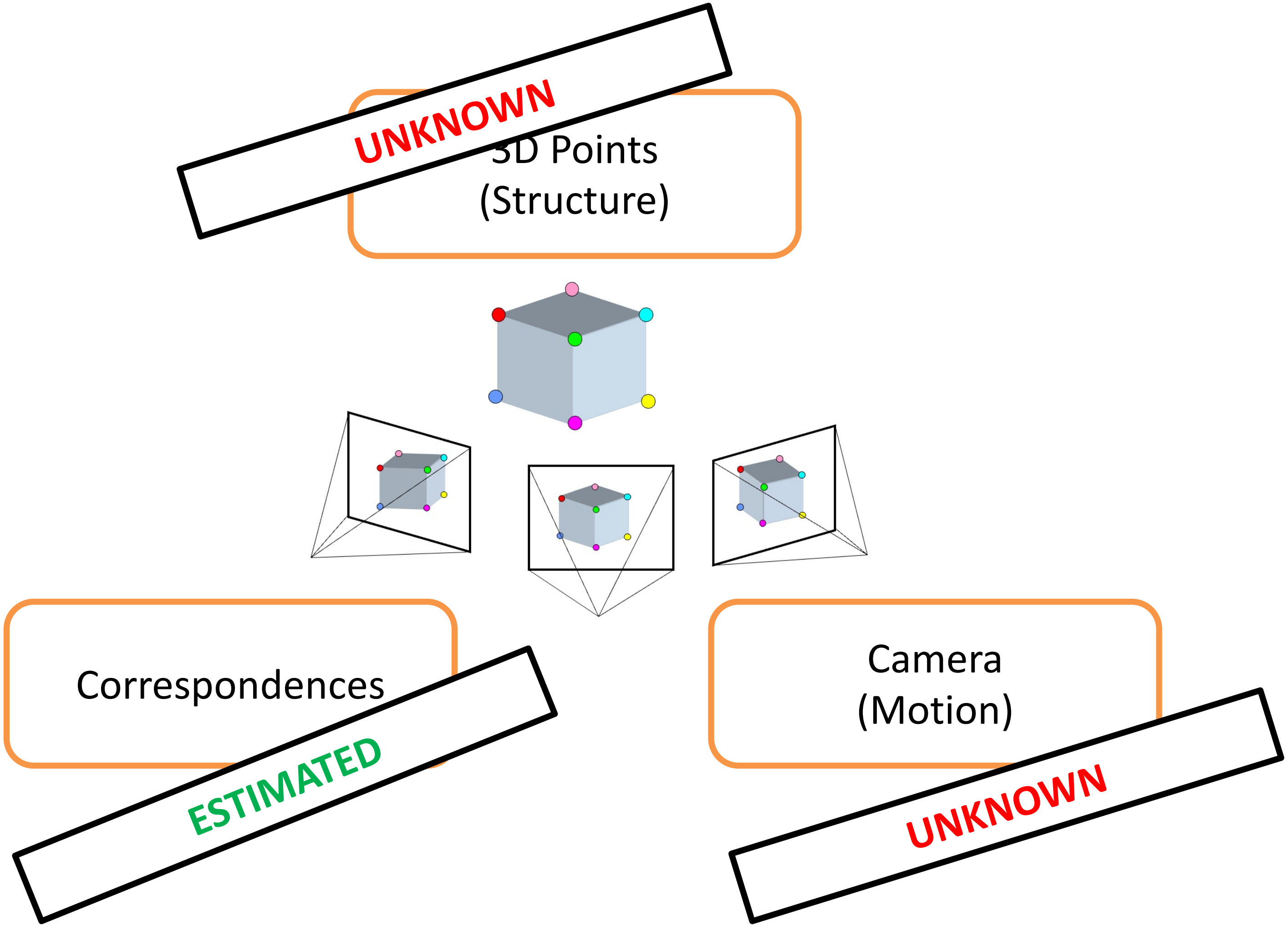
Putting it all together

Structure-from-Motion: You know nothing!

(except ok maybe intrinsics)

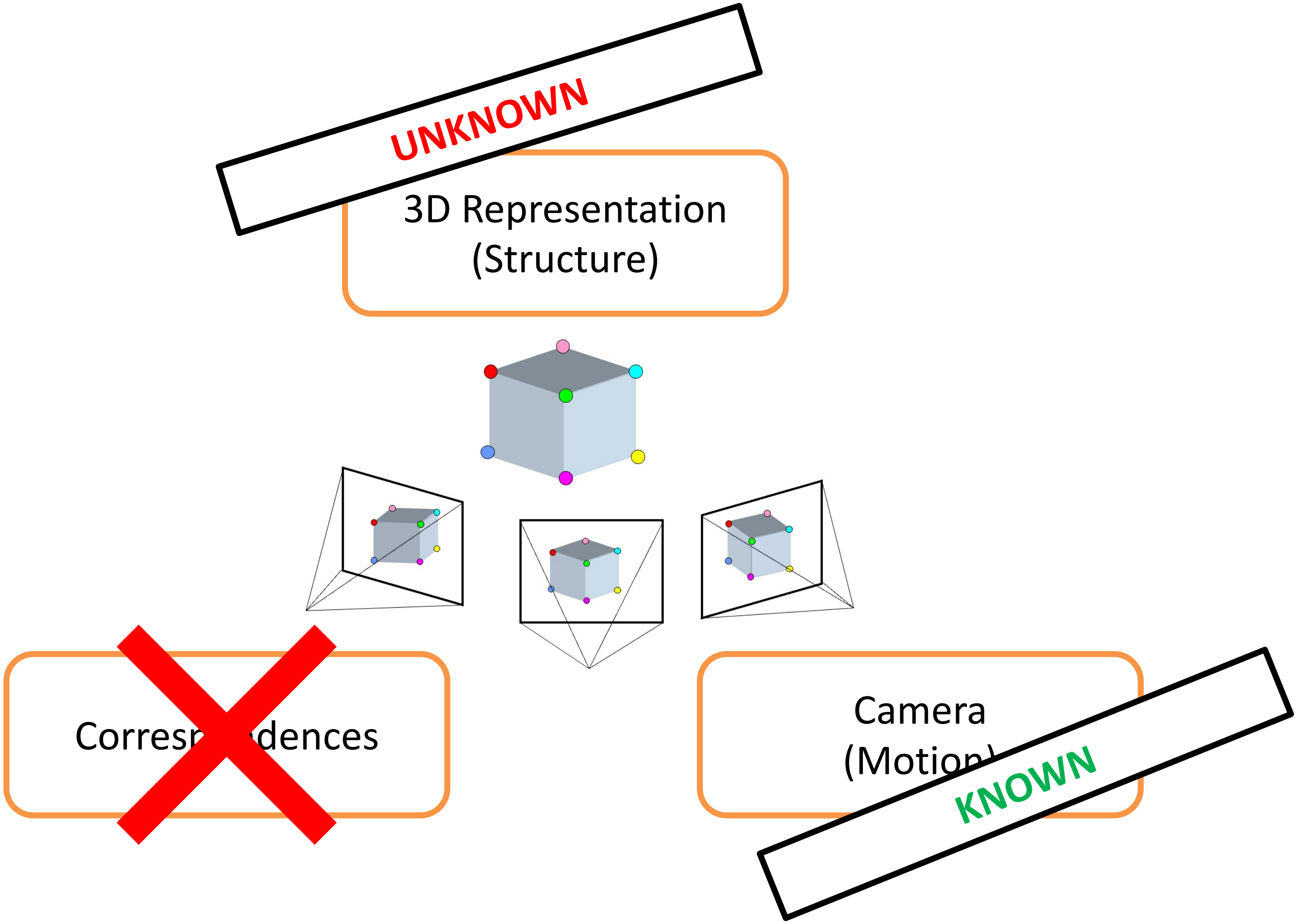


(Next lecture) Ultimate: Structure-from-Motion/SLAM



The starting point for all problems where you can't calibrate actively

(after that): Neural Rendering



A form of multi-view stereo, more on this in the NeRF lecture.

Next: Large-scale structure from motion

Dubrovnik, Croatia. 4,619 images (out of an initial 57,845).
Total reconstruction time: 23 hours
Number of cores: 352

Building Rome in a Day, Agarwal et al. ICCV 2009

Slide courtesy of Noah Snavely

Large-scale structure from motion

Result using COLMAP: Schönberger et al. CVPR ’16