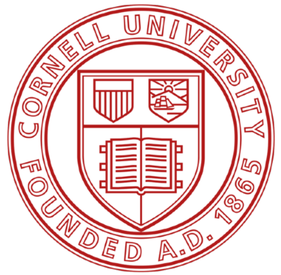
Lecture 19: Two-view geometry

CS 5670: Introduction to Computer Vision

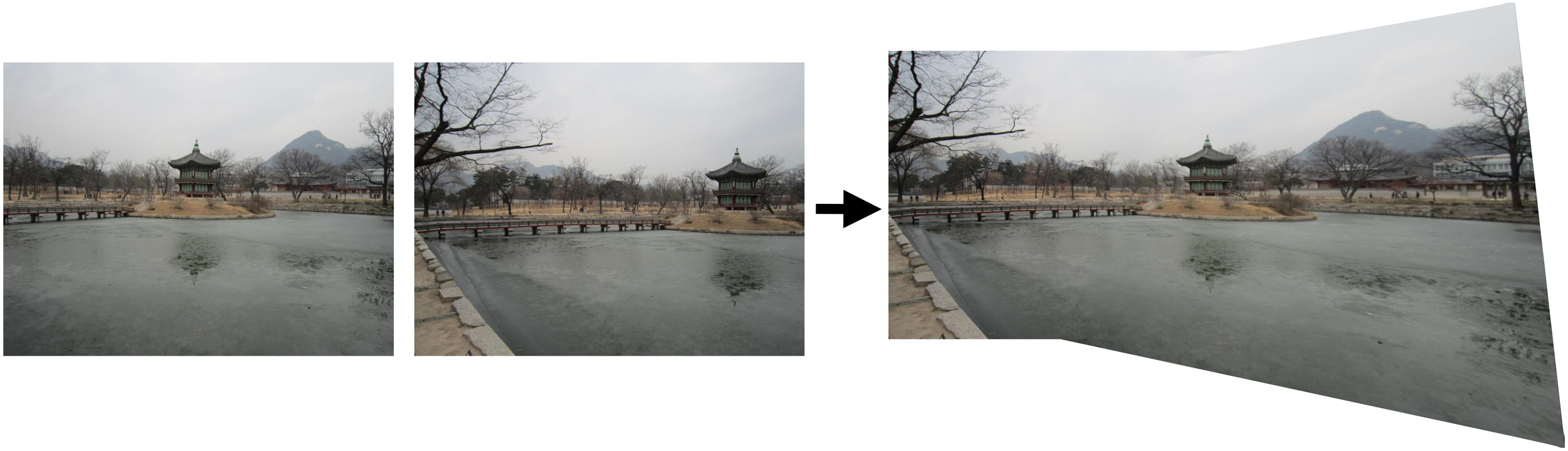


Most slides from Noah Snavely

Today

- Epipolar geometry
- Stereo matching
- Image alignment

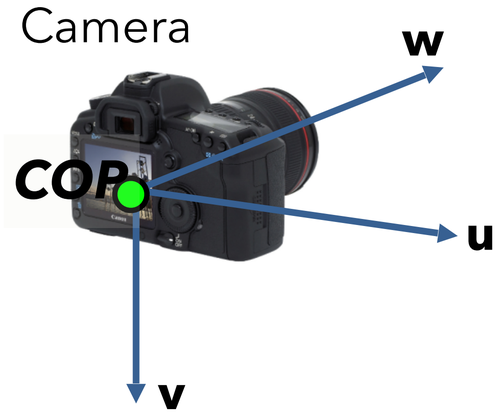
PS5: making panoramas

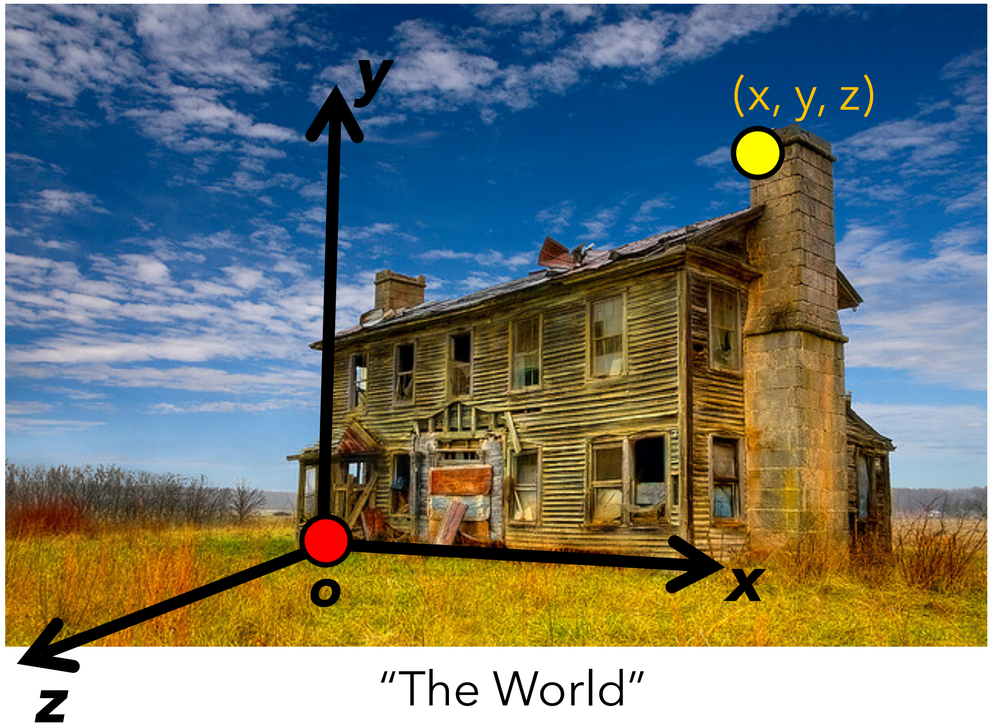


Source: N. Snavely

Camera parameters

- How can we model the geometry of a camera?





"The World"

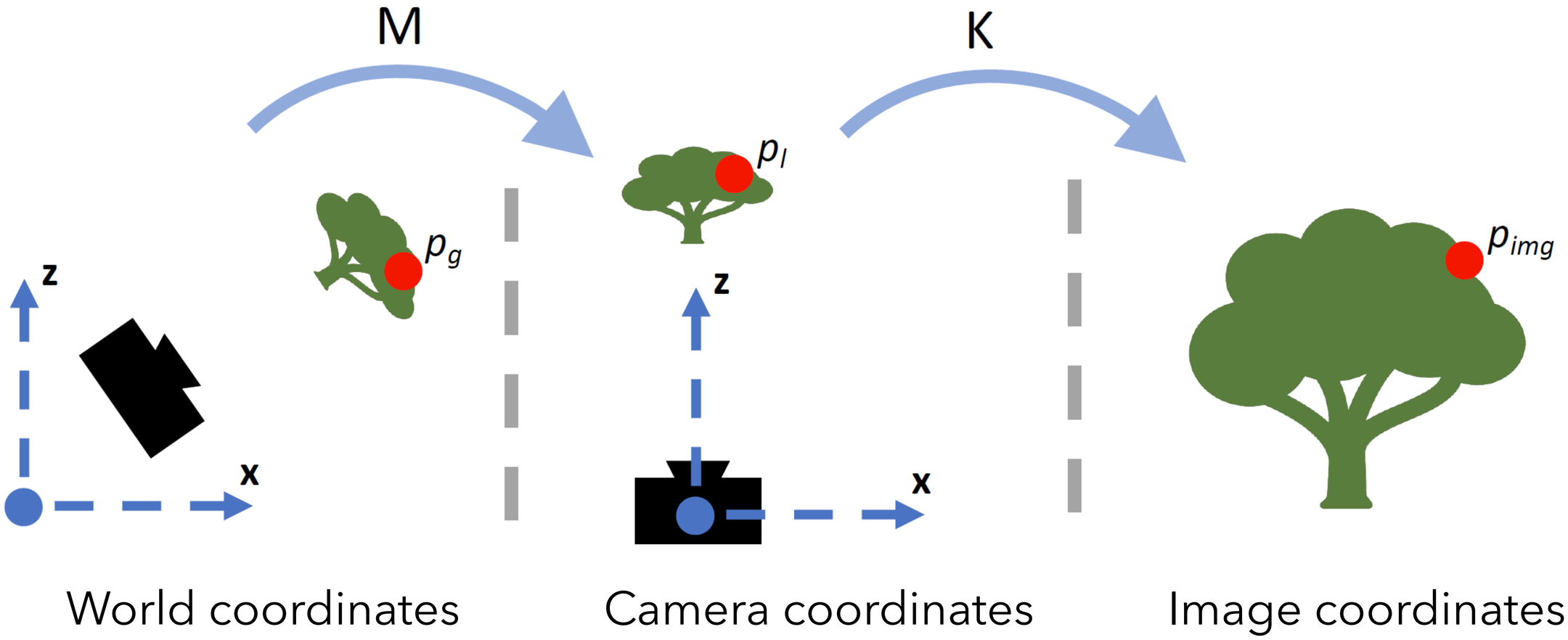
Three important coordinate systems:

1. *World* coordinates
2. *Camera* coordinates
3. *Image* coordinates

How do we project a given world point (x, y, z) to an image point?

Source: N. Snavely

Coordinate frames



Source: N. Snavely
Figure credit: Peter Hedman

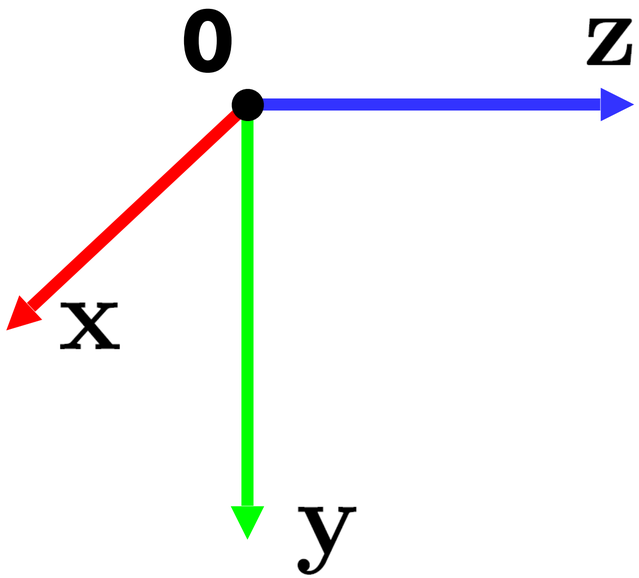
Camera parameters

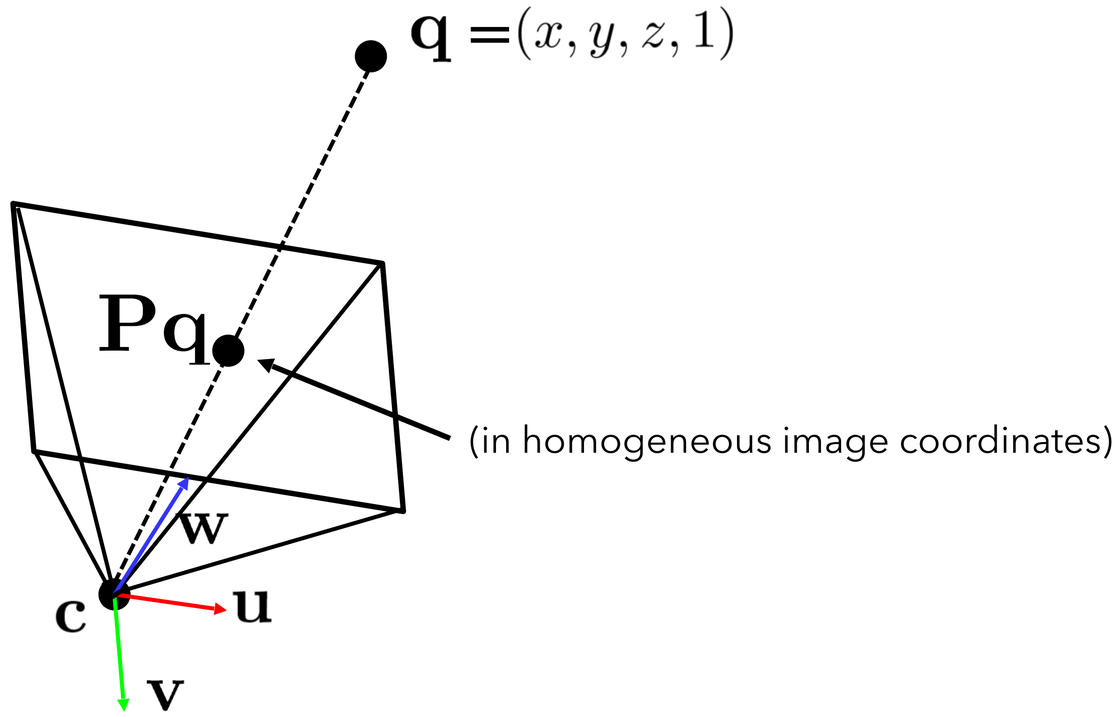
To project a point (*x*, *y*, *z*) in *world* coordinates into a camera

- First transform (*x*, *y*, *z*) into *camera* coordinates
- Need to know
  - Camera position (in world coordinates)
  - Camera orientation (in world coordinates)
- Then project into the image plane to get *image (pixel) coordinates*
  - Need to know camera *intrinsics*

Source: N. Snavely

Projection matrix

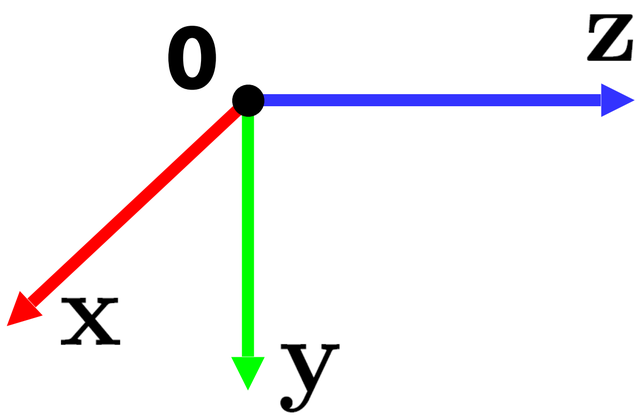




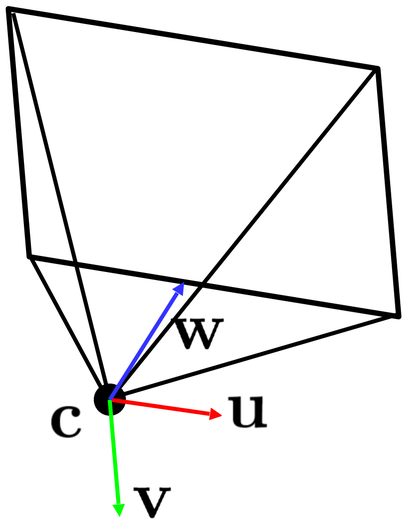
Source: N. Snavely

Extrinsics

- How do we get the camera to “canonical form”?
  - (Center of projection at the origin, x-axis points right, y-axis points up, z-axis points backwards)



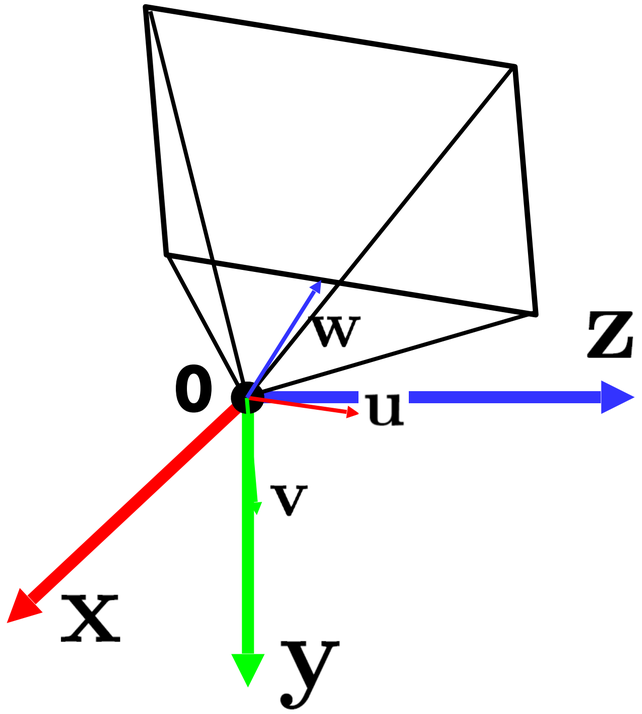
Step 1: Translate by -**c**



Source: N. Snavely

Extrinsics

- How do we get the camera to "canonical form"?
  - (Center of projection at the origin, x-axis points right, y-axis points up, z-axis points backwards)



Step 1: Translate by -$\mathbf{c}$

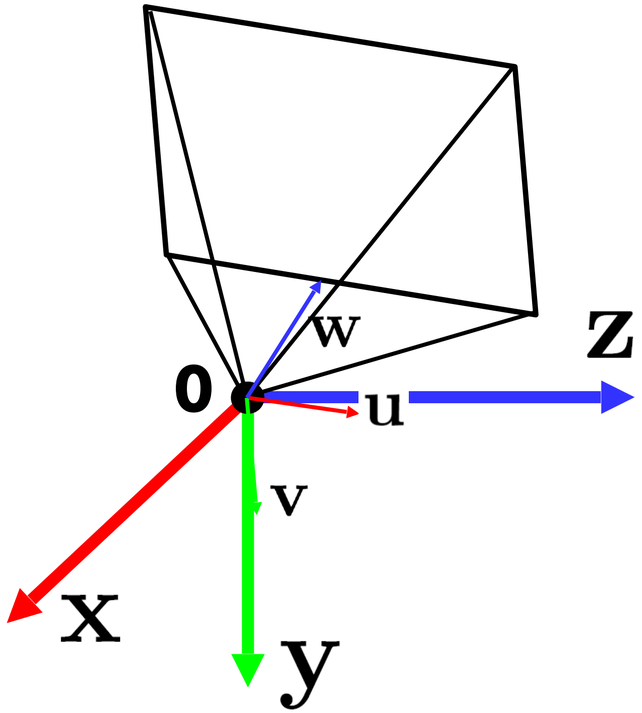
How do we represent translation as a matrix multiplication?

$$\mathbf{T} = \begin{bmatrix} \mathbf{I}_{3\times 3} & -\mathbf{c} \\ 0 \quad 0 \quad 0 & 1 \end{bmatrix}$$

Source: N. Snavely

Extrinsics

- How do we get the camera to "canonical form"?
  - (Center of projection at the origin, x-axis points right, y-axis points up, z-axis points backwards)



Step 1: Translate by -**c**

Step 2: Rotate by **R**

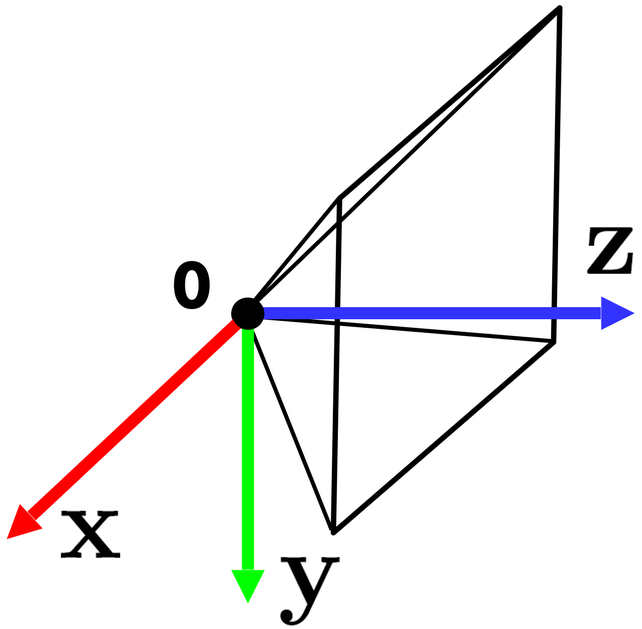
$$\mathbf{R} = \begin{bmatrix} \mathbf{u}^T \\ \mathbf{v}^T \\ \mathbf{w}^T \end{bmatrix}$$

3x3 rotation matrix

Source: N. Snavely

Extrinsics

- How do we get the camera to "canonical form"?
  - (Center of projection at the origin, x-axis points right, y-axis points up, z-axis points backwards)



Step 1: Translate by -**c**

Step 2: Rotate by **R**

$$\mathbf{R} = \begin{bmatrix} \mathbf{u}^T \\ \mathbf{v}^T \\ \mathbf{w}^T \end{bmatrix}$$

(with extra row/column of [0 0 0 1])

Source: N. Snavely

Perspective projection

$$\underbrace{\begin{bmatrix} f & 0 & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{bmatrix}}_{\mathbf{K}} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

(intrinsics)

(converts from 3D rays in camera coordinate system to pixel coordinates)

in general, $\mathbf{K} = \begin{bmatrix} f & s & c_x \\ 0 & \alpha f & c_y \\ 0 & 0 & 1 \end{bmatrix}$ (upper triangular matrix)

α: **aspect ratio** (1 unless pixels are not square)

s : **skew** (0 unless pixels are shaped like rhombi/parallelograms)

(c_x, c_y): **principal point** ((w/2,h/2) unless optical axis doesn't intersect projection plane at image center)

Source: N. Snavely

Typical intrinsics matrix

$$\mathbf{K} = \begin{bmatrix} f & 0 & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

- **2D affine transform** corresponding to a scale by f (focal length) and a translation by (c_x, c_y) (principal point)
- Maps 3D rays to 2D pixels

Source: N. Snavely

Projection matrix

$$\mathbf{P} = \mathbf{K} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} & \mathbf{R} & & \begin{matrix} 0 \\ 0 \\ 0 \end{matrix} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{I}_{3\times 3} & -\mathbf{c} \\ 0 \quad 0 \quad 0 & 1 \end{bmatrix}$$

intrinsics

projection

rotation

translation

The **K** matrix converts 3D rays in the camera's coordinate system to 2D image points in image (pixel) coordinates.

This part converts 3D points in world coordinates to 3D rays in the camera's coordinate system. There are 6 parameters represented (3 for position/translation, 3 for rotation).

Source: N. Snavely

Projection matrix

$$\mathbf{P} = \mathbf{K} \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{\text{projection}} \underbrace{\begin{bmatrix} & \mathbf{R} & & \begin{matrix} 0 \\ 0 \\ 0 \end{matrix} \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{\text{rotation}} \underbrace{\begin{bmatrix} & & & \\ & \mathbf{I}_{3\times 3} & & -\mathbf{c} \\ & & & \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{\text{translation}}$$

intrinsics

$$\begin{bmatrix} \mathbf{R} & \underbrace{-\mathbf{Rc}} \end{bmatrix}$$

(sometimes called **t**)

$$\mathbf{P} = \mathbf{K} \begin{bmatrix} \mathbf{R} & -\mathbf{Rc} \end{bmatrix}$$

Source: N. Snavely

Focal length

- Can think of as "zoom"

24mm

50mm

200mm

800mm

- Also related to *field of view*

Source: N. Snavely

Changing focal length

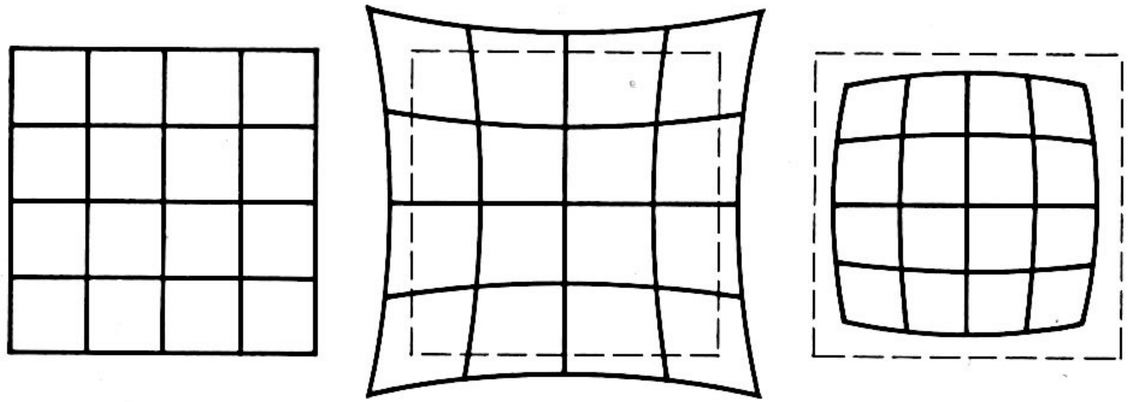
Wide angle | Standard | Telephoto

Source: N. Snavely

http://petapixel.com/2013/01/11/how-focal-length-affects-your-subjects-apparent-weight-as-seen-with-a-cat/

Source: N. Snavely

Distortion



No distortion Pin cushion Barrel

- Radial distortion of the image
  - Caused by imperfect lenses
  - Deviations are most noticeable for rays that pass through the edge of the lens

Source: N. Snavely

Source: N. Snavely

Modeling distortion

$(\hat{x}, \hat{y}, \hat{z})$

Project to "normalized" image coordinates

$$x'_n = \hat{x}/\hat{z}$$
$$y'_n = \hat{y}/\hat{z}$$

$$r^2 = {x'_n}^2 + {y'_n}^2$$

Apply radial distortion

$$x'_d = x'_n(1 + \kappa_1 r^2 + \kappa_2 r^4)$$
$$y'_d = y'_n(1 + \kappa_1 r^2 + \kappa_2 r^4)$$

Apply focal length translate image center

$$x' = f x'_d + x_c$$
$$y' = f y'_d + y_c$$

- To model lens distortion
  - Use above projection operation instead of standard projection matrix multiplication

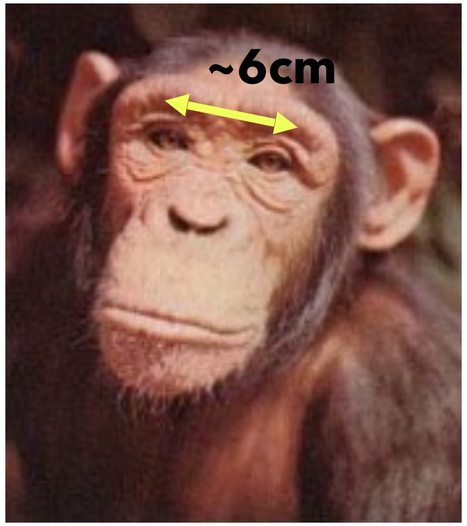
Source: N. Snavely

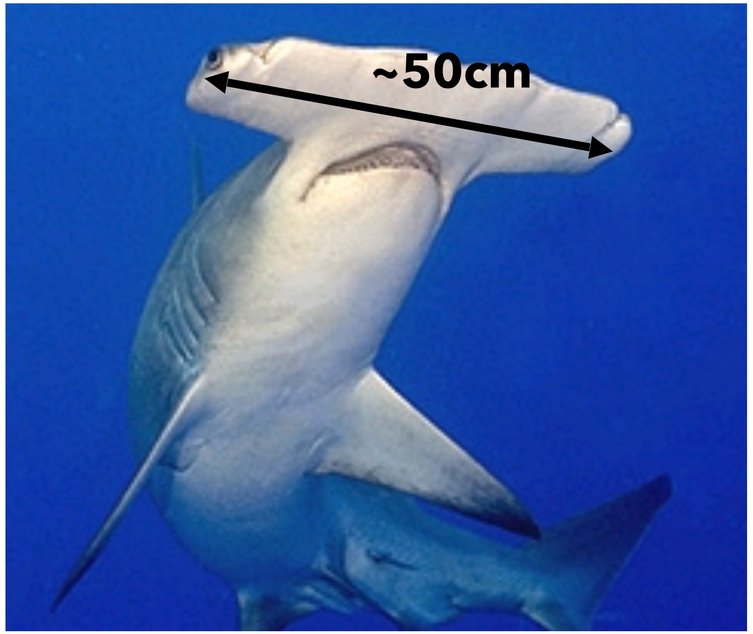
Correcting radial distortion

from Helmut Dersch

Estimating depth from multiple views

Stereo vision





Source: Torralba, Isola, Freeman

1 vs. 2 eyes

Source: Torralba, Isola, Freeman

1 vs. 2 eyes

Source: Torralba, Isola, Freeman

Stereoscopic card

Brewster stereoscope

Image source: wikipedia

Source: Torralba, Isola, Freeman

Depth without objects

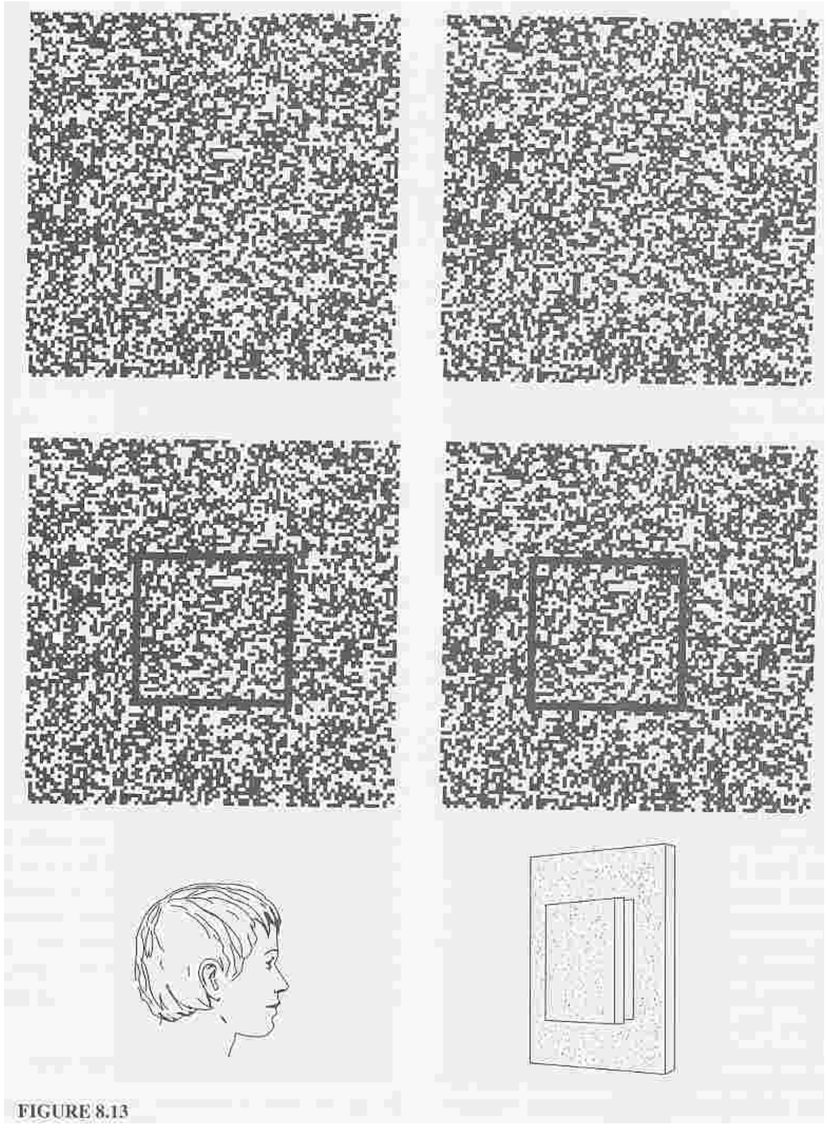
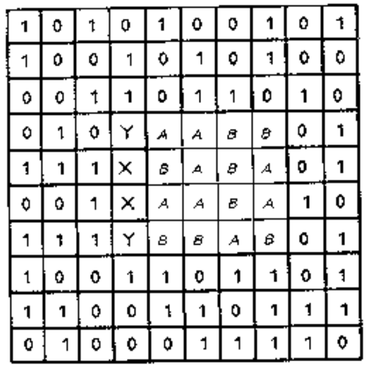
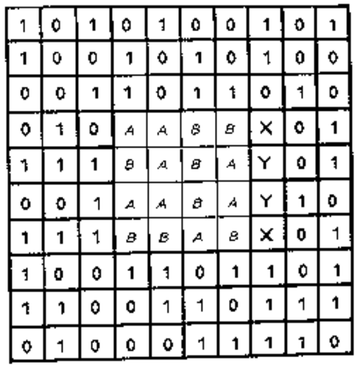


FIGURE 8.13

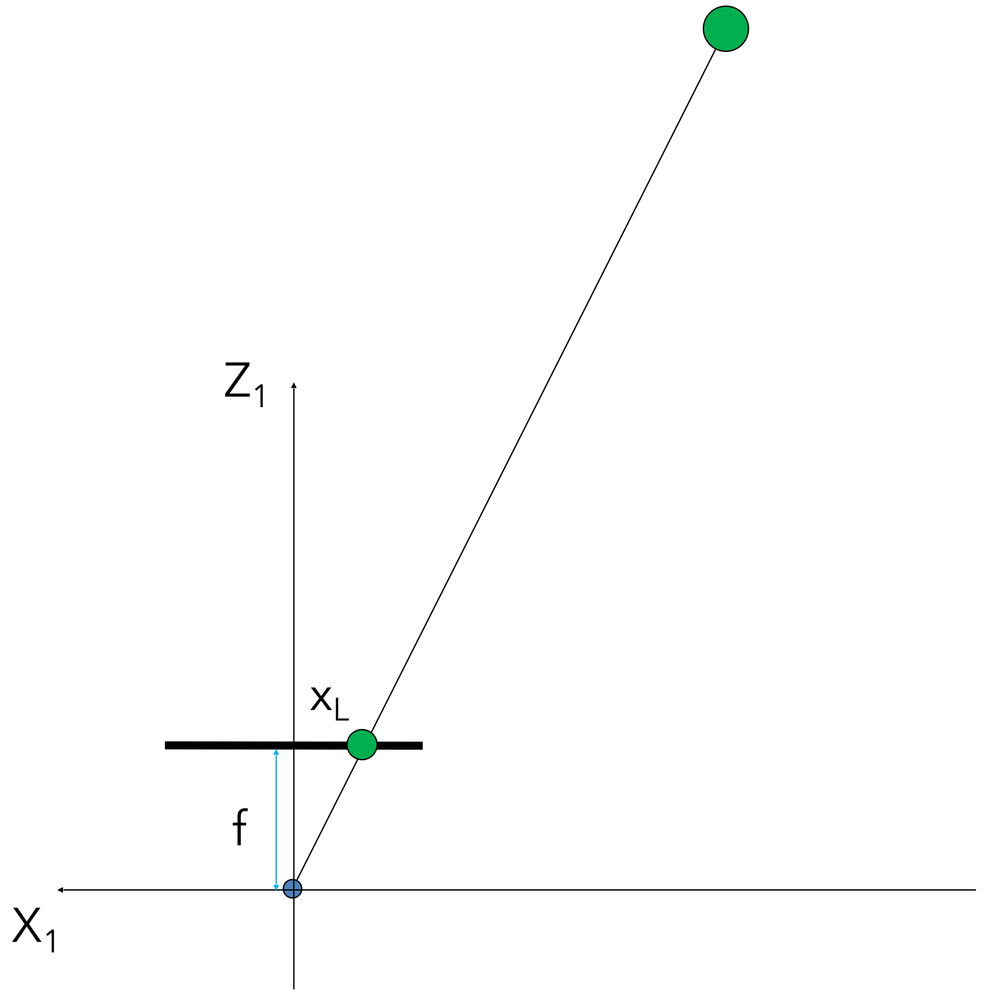




Julesz, 1971

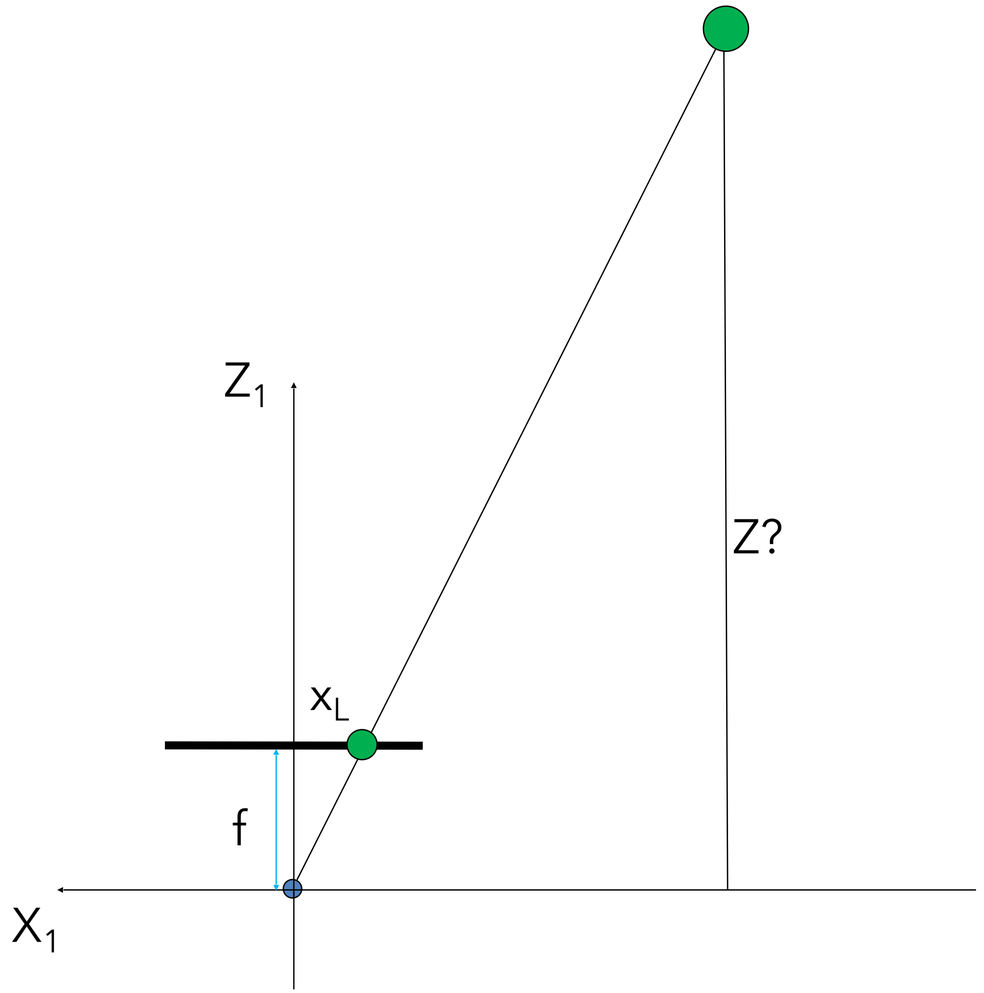
Source: Torralba, Isola, Freeman

Geometry for a simple stereo system



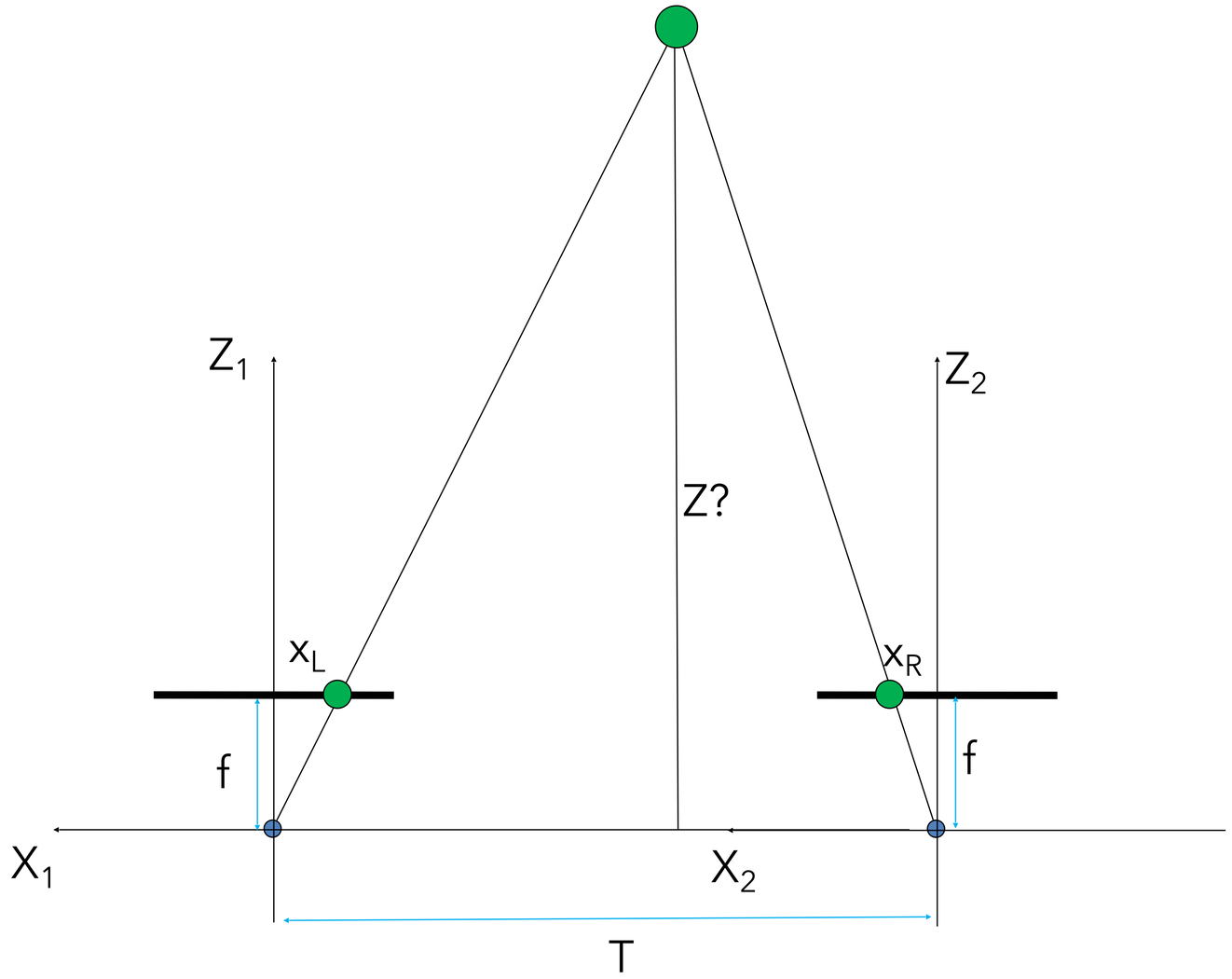
Source: Torralba, Isola, Freeman

Geometry for a simple stereo system



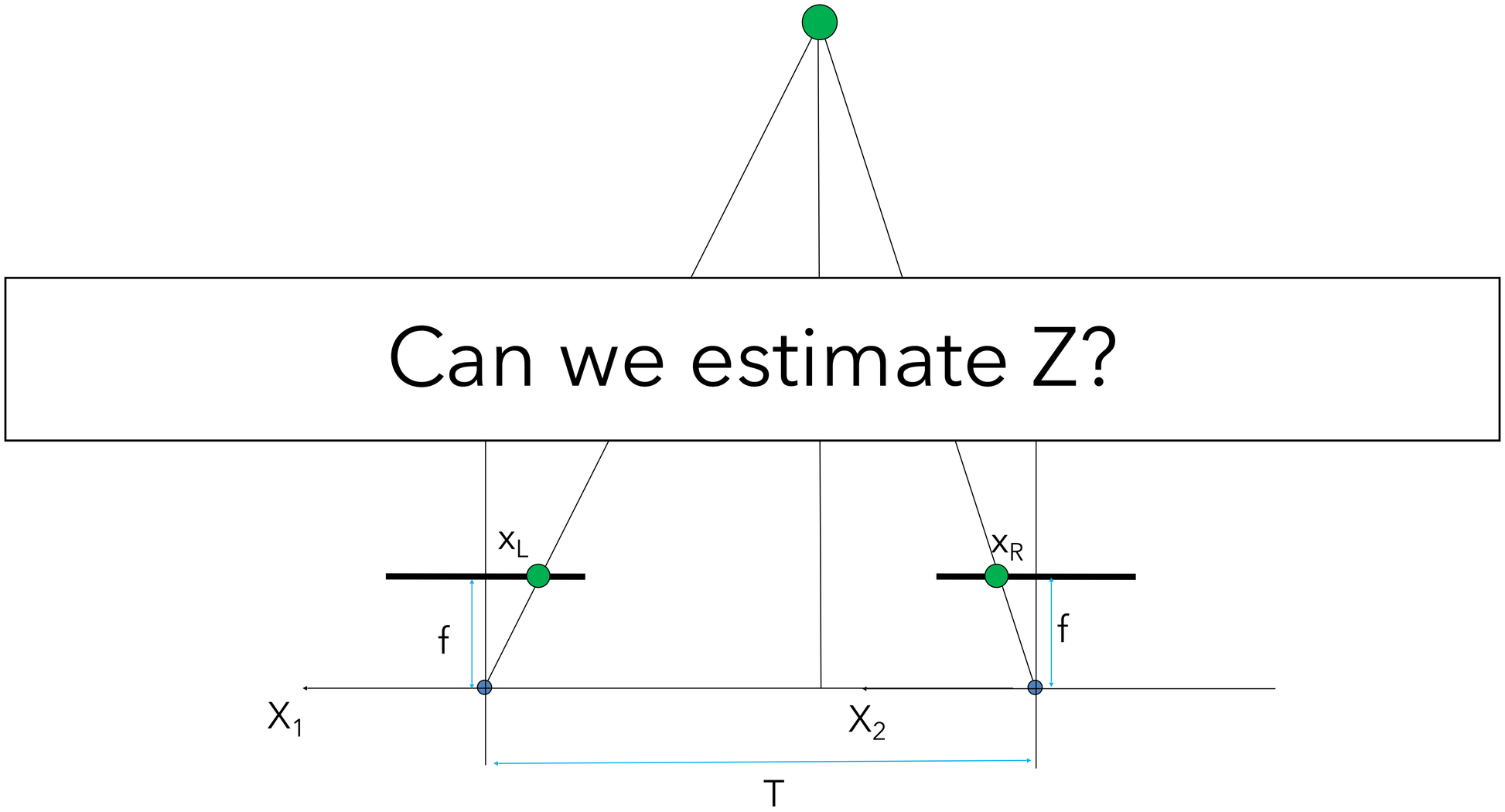
Source: Torralba, Isola, Freeman

Geometry for a simple stereo system



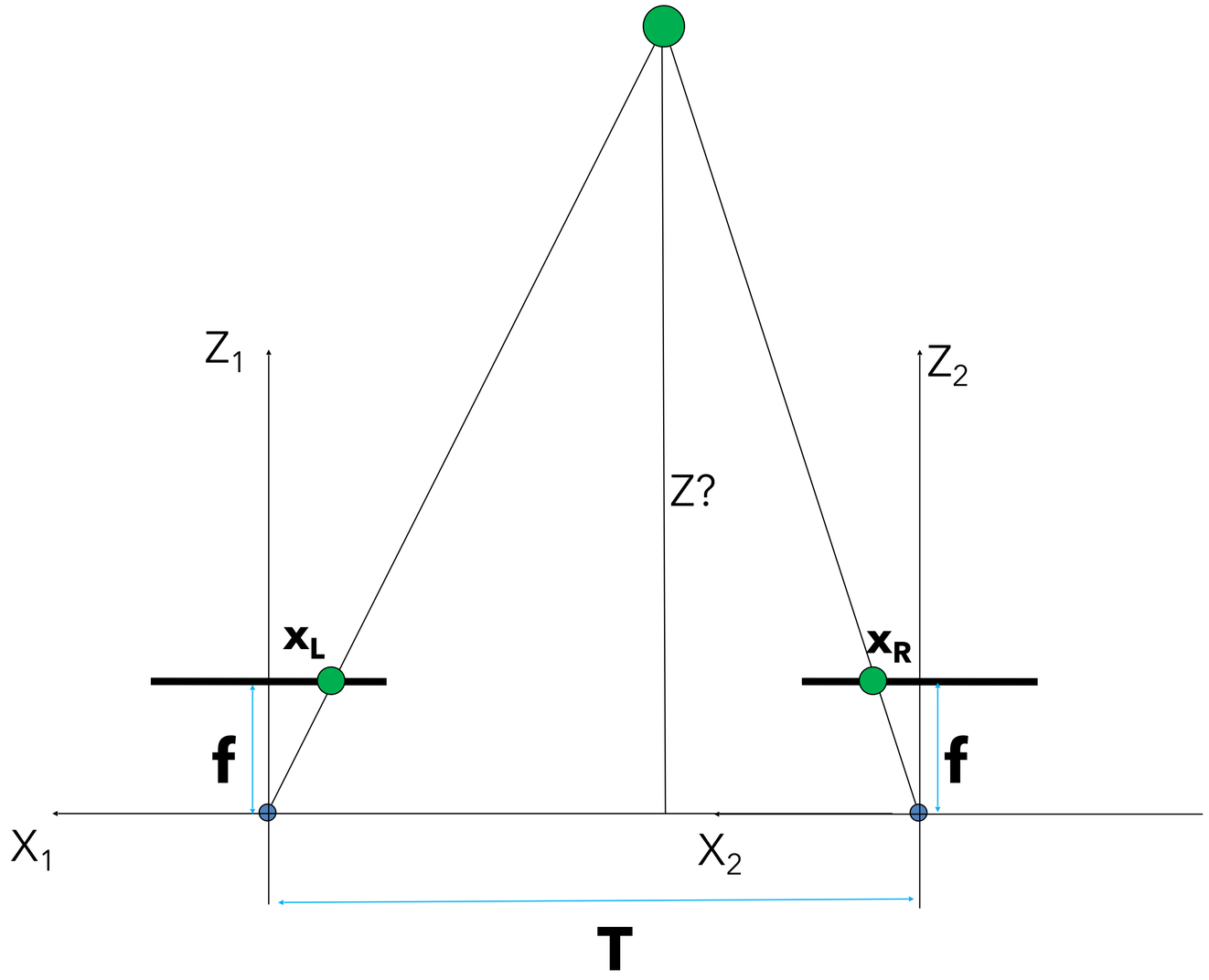
Source: Torralba, Isola, Freeman

Geometry for a simple stereo system



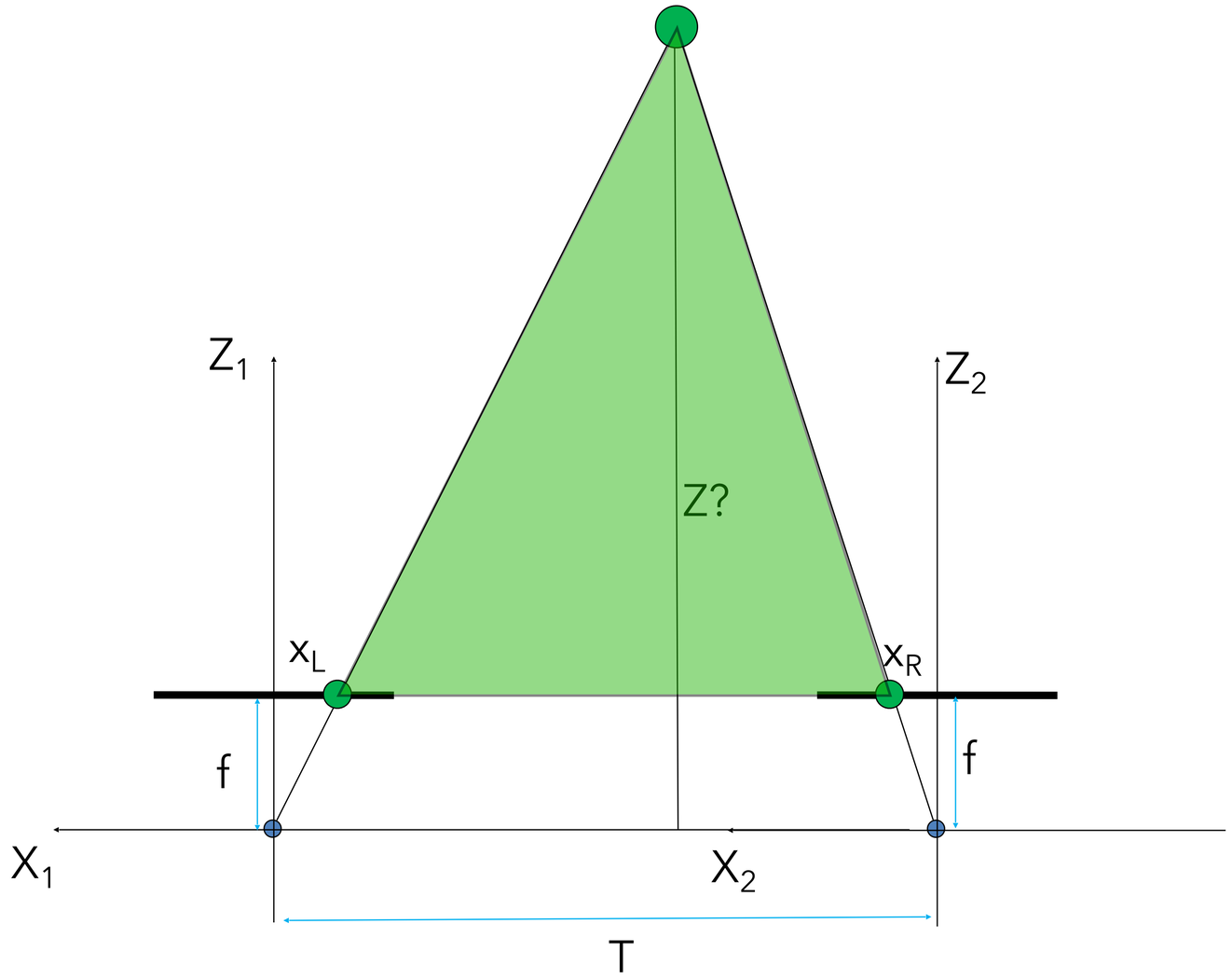
Source: Torralba, Isola, Freeman

Geometry for a simple stereo system



Source: Torralba, Isola, Freeman

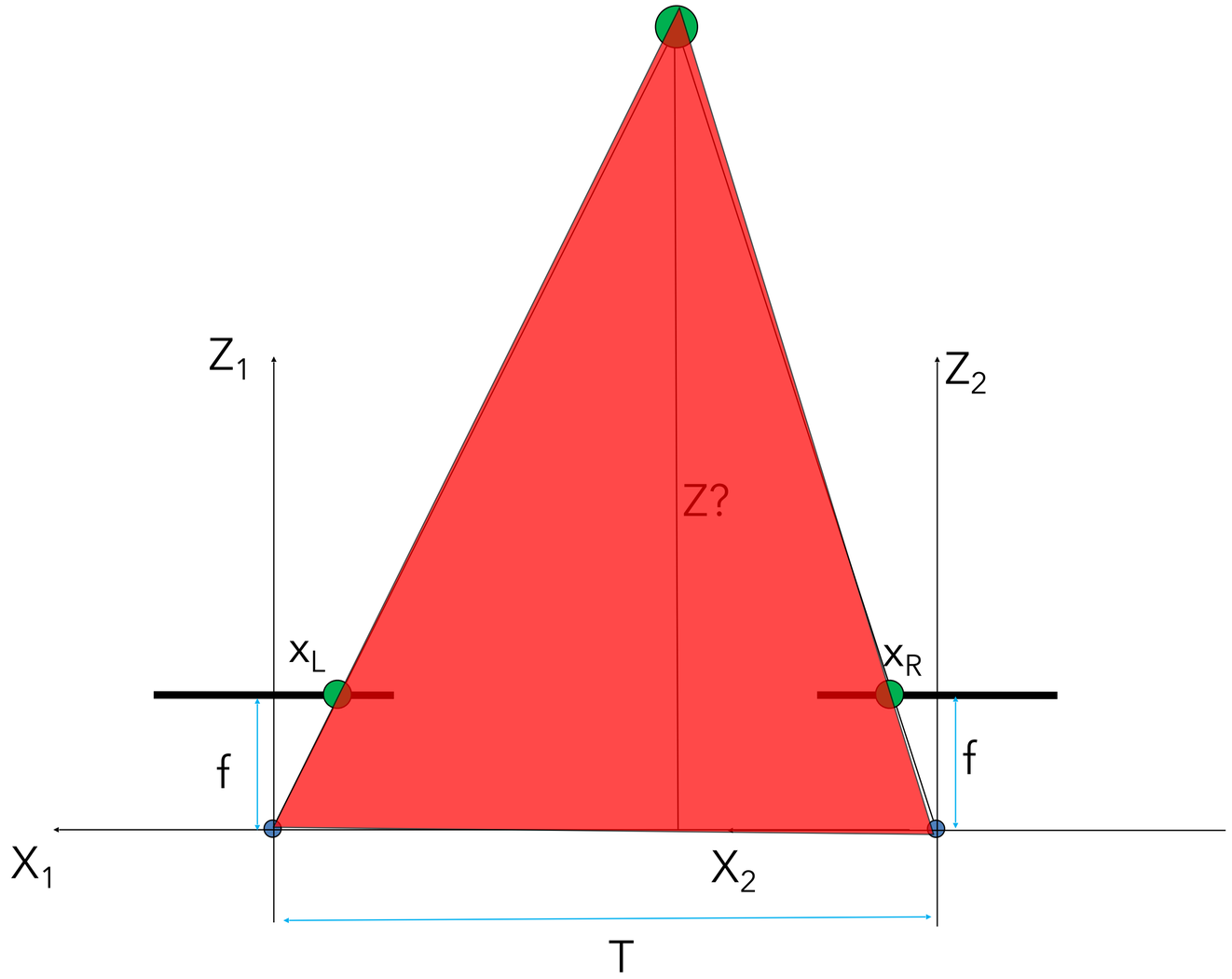
Geometry for a simple stereo system



Similar triangles!

Source: Torralba, Isola, Freeman

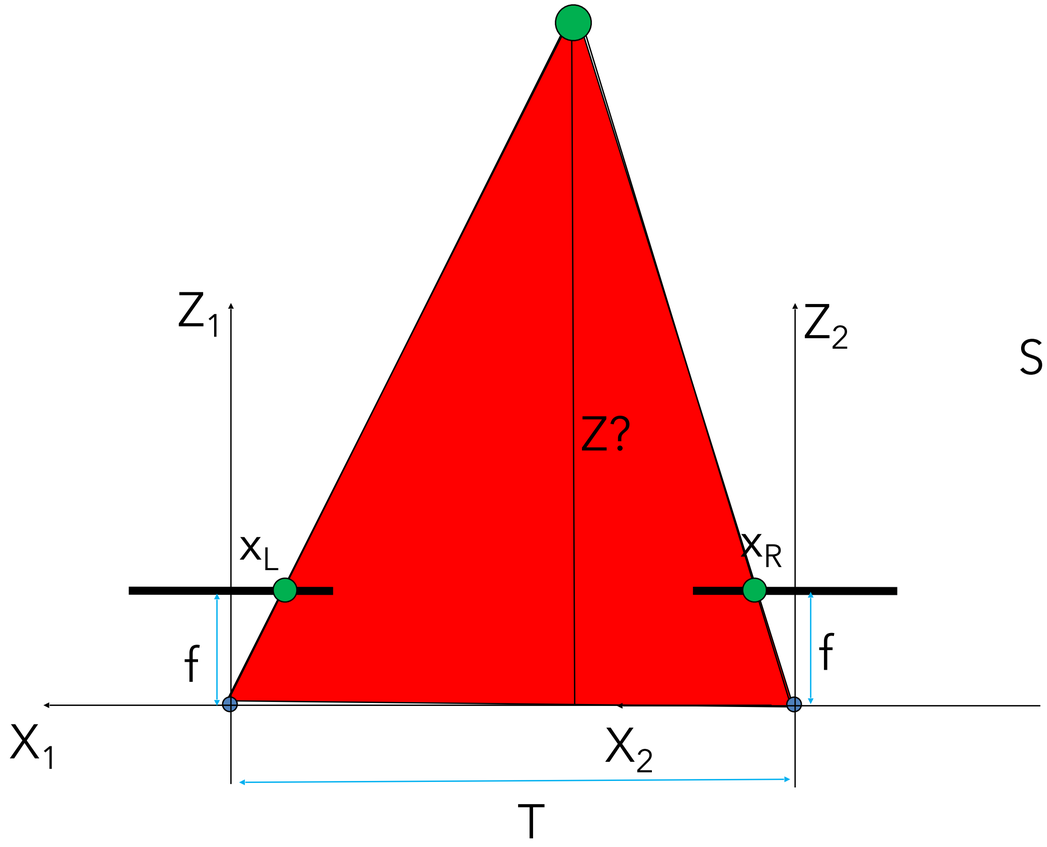
Geometry for a simple stereo system



Similar triangles!

Source: Torralba, Isola, Freeman

Geometry for a simple stereo system

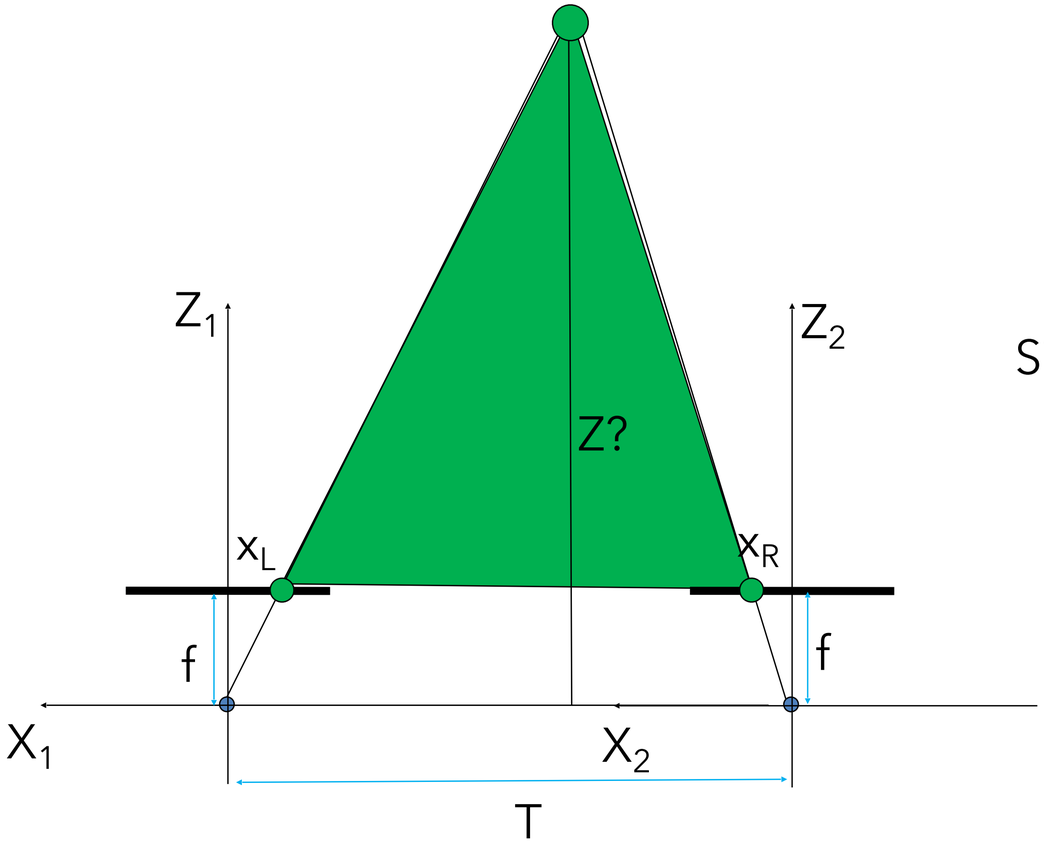


Similar triangles:

$$\frac{T}{Z}$$

Source: Torralba, Isola, Freeman

Geometry for a simple stereo system

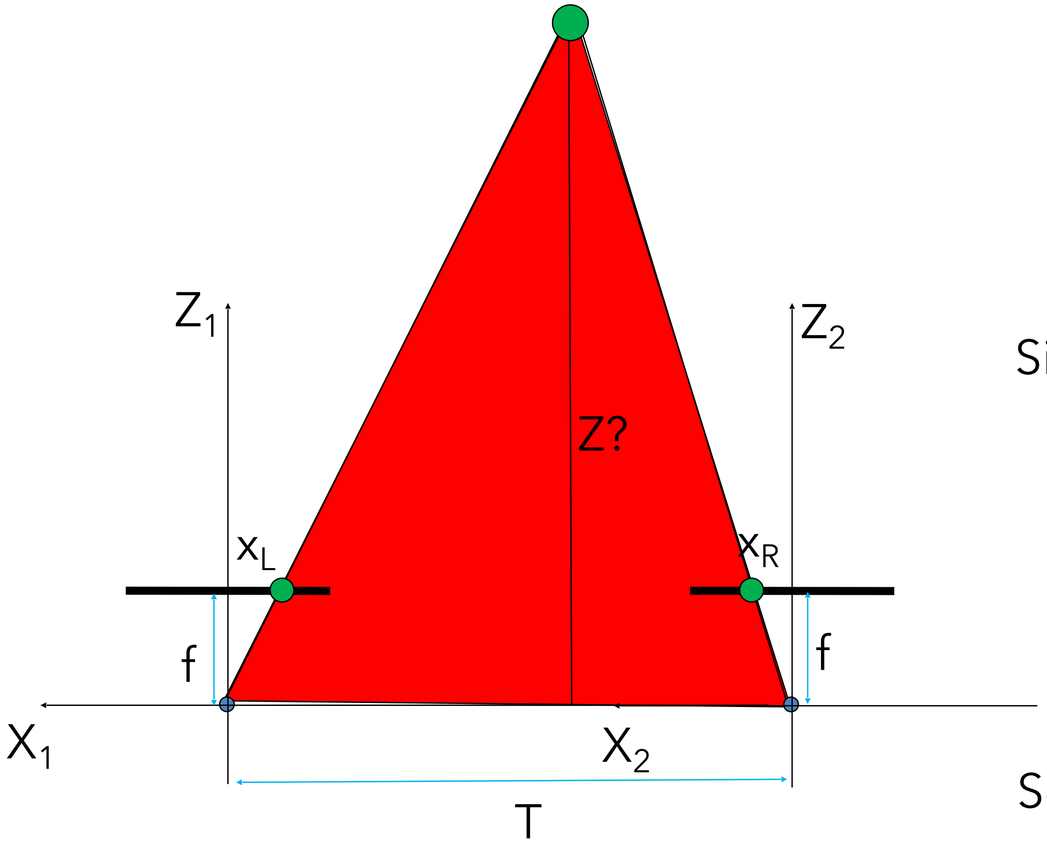


Similar triangles:

$$\frac{T+X_R-X_L}{Z-f} = \frac{T}{Z}$$

Source: Torralba, Isola, Freeman

Geometry for a simple stereo system



Similar triangles:

$$\frac{T+X_R-X_L}{Z-f} = \frac{T}{Z}$$

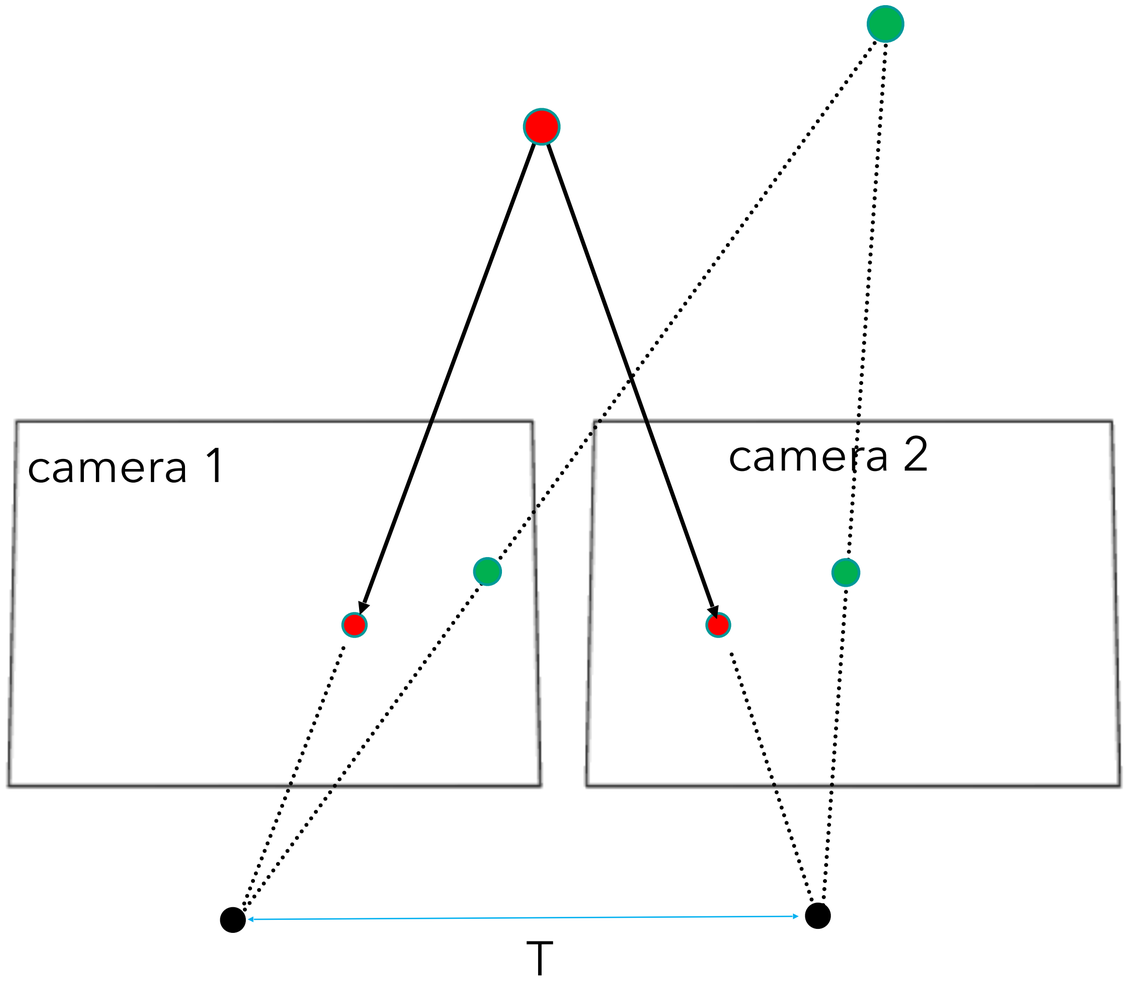
Solving for Z:

$$Z = f\,\frac{T}{X_L - X_R}$$

Disparity

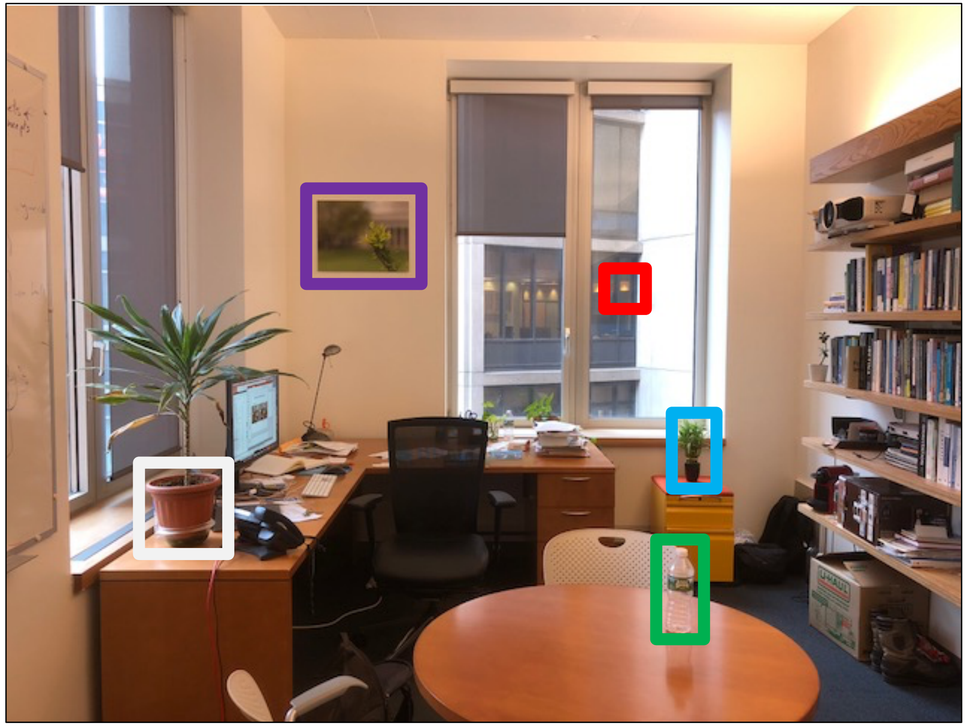
Source: Torralba, Isola, Freeman

In 3D



Source: Torralba, Isola, Freeman

Disparity map



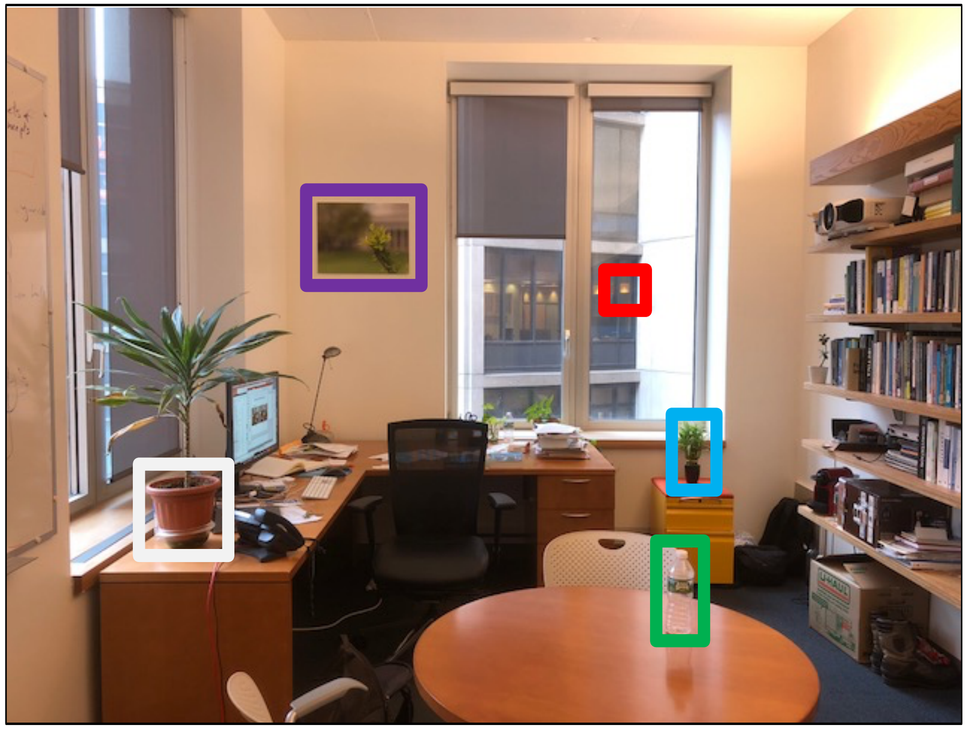
Left image

Right image

Second picture is ~1m to the right

Source: Torralba, Isola, Freeman

Disparity map

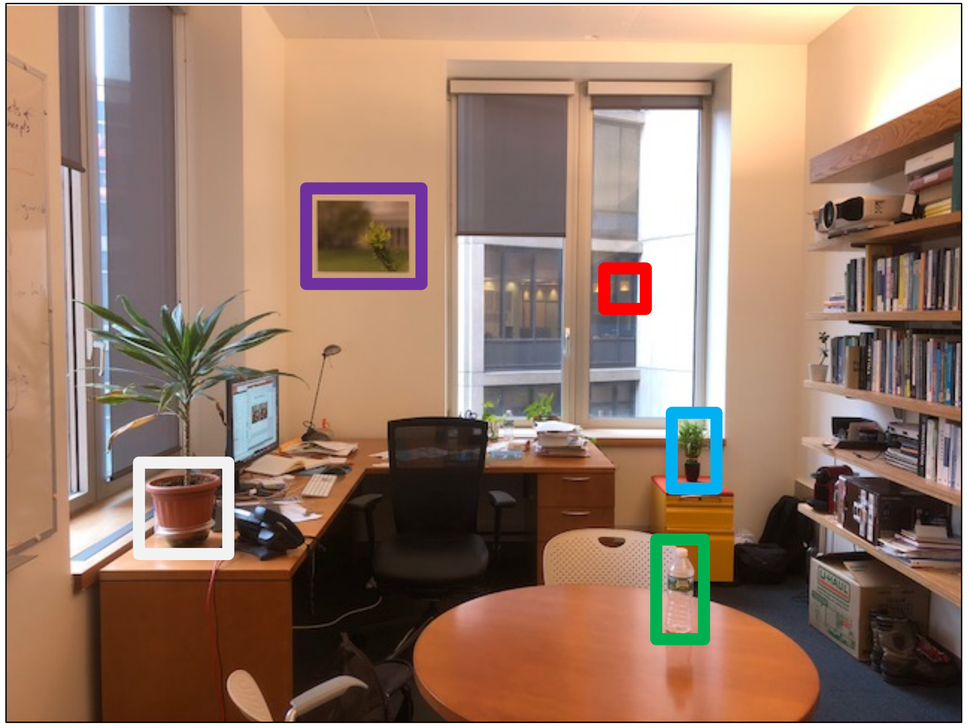


Left image

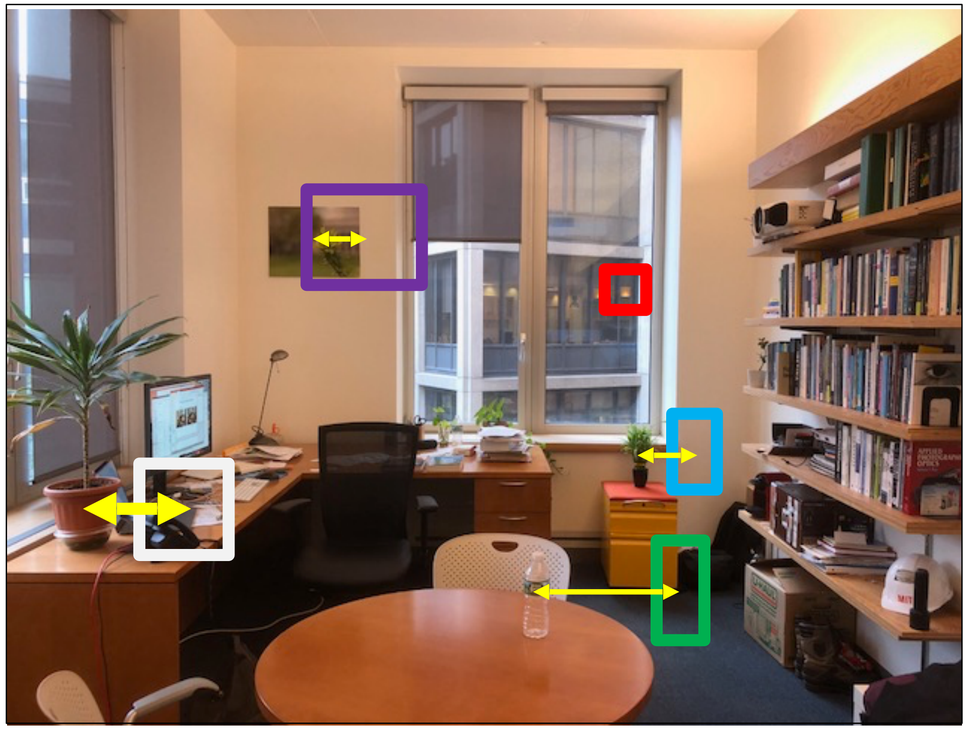
Right image

Source: Torralba, Isola, Freeman

Disparity map



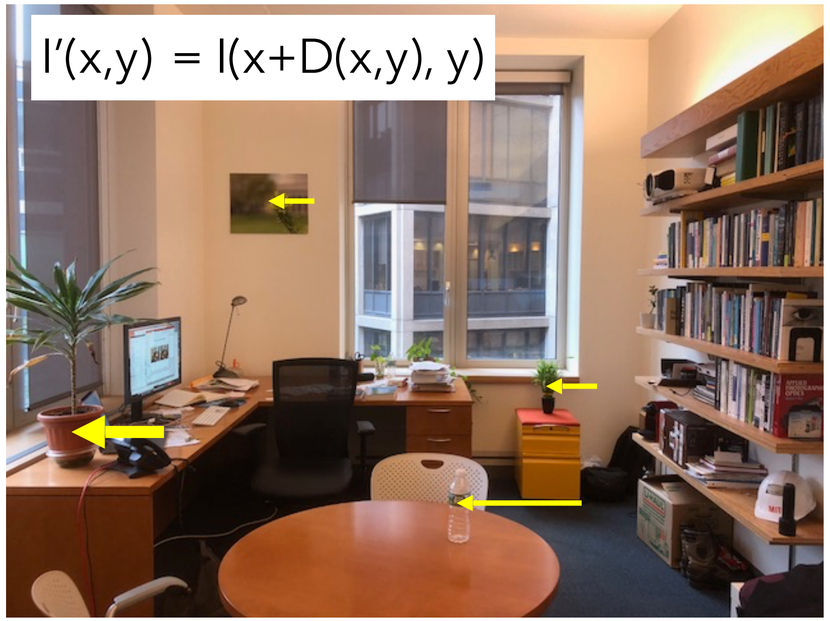
Left image



Right image

Source: Torralba, Isola, Freeman

Disparity map



D(x,y)

$$Z(x,y) = \frac{f}{D(x,y)}$$

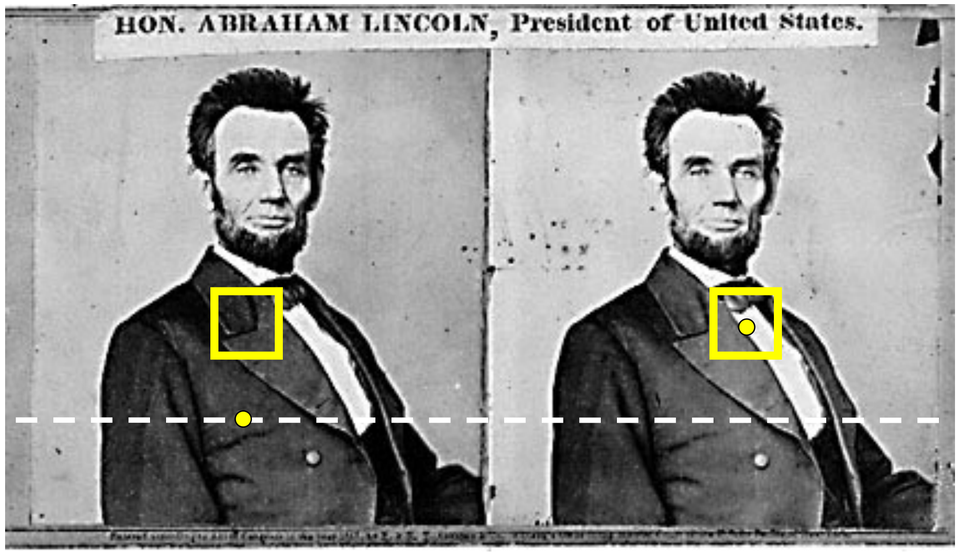
Source: Torralba, Isola, Freeman

Finding correspondences

We only need to search for matches along horizontal lines.

Source: Torralba, Isola, Freeman

Basic stereo algorithm



For each “epipolar line”

For each pixel in the left image

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

Source: R. Szeliski

Computing disparity

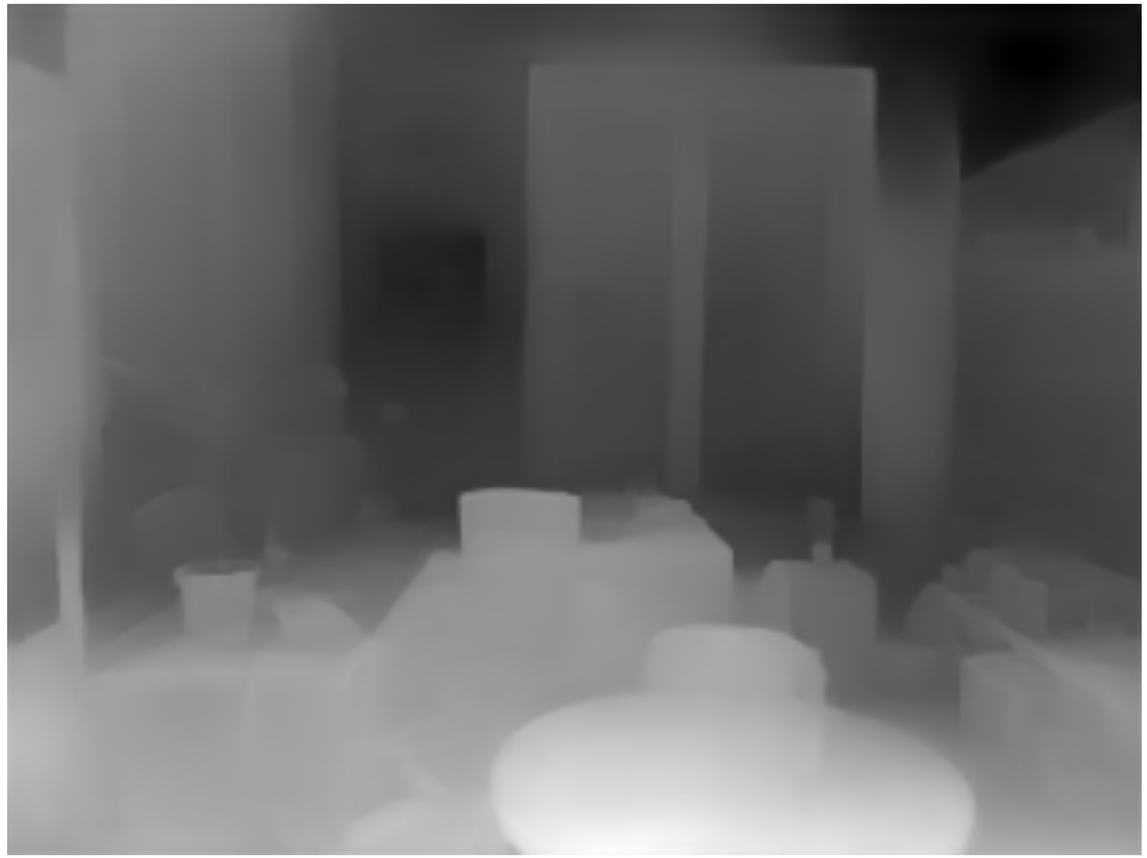
Source: Torralba, Isola, Freeman

Computing disparity

Semi-global matching [Hirschmüller 2008]

Source: Torralba, Isola, Freeman

Can also learn depth from a single image

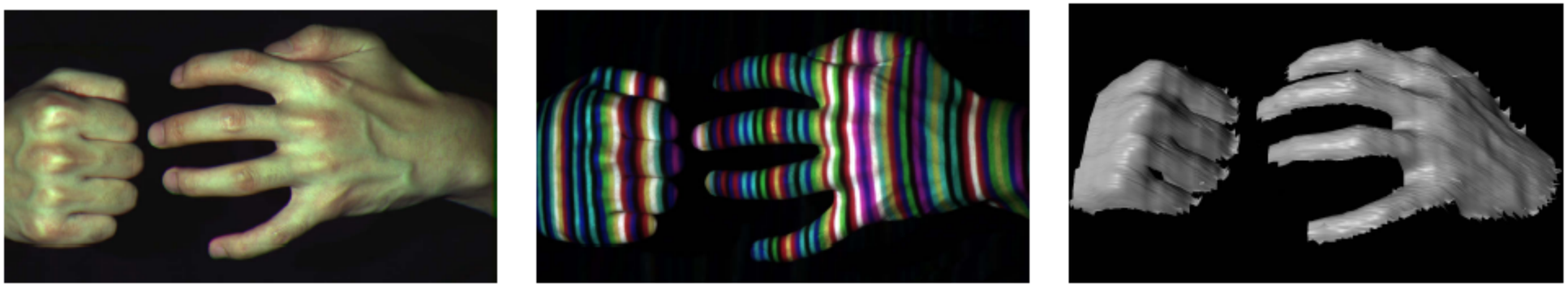


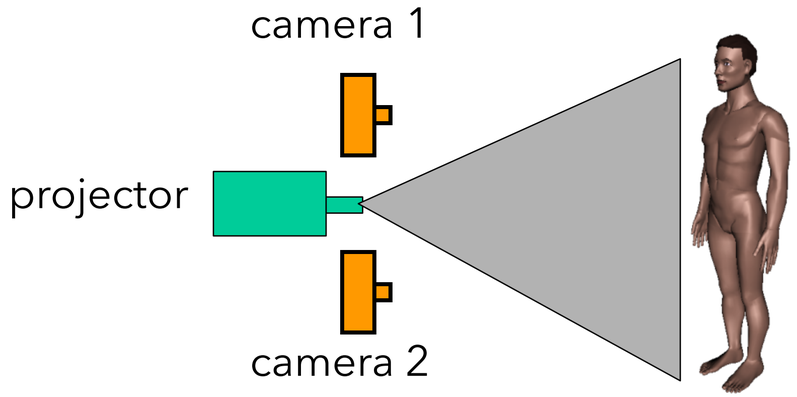
MegaDepth: Learning Single-View Depth Prediction from Internet Photos

Zhengqi Li Noah Snavely
Department of Computer Science & Cornell Tech, Cornell University

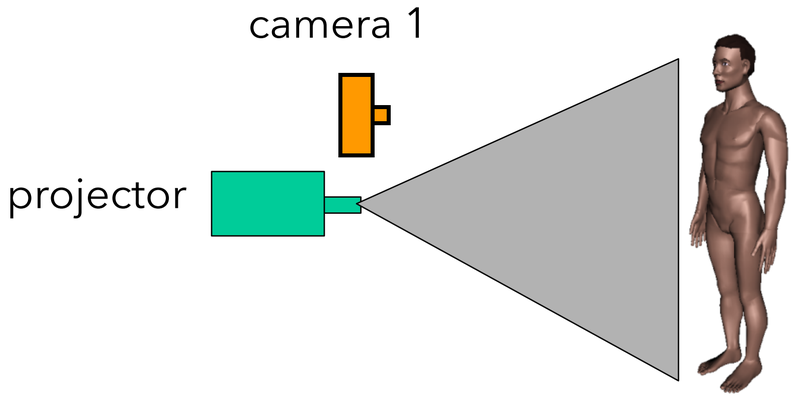
Source: Torralba, Isola, Freeman

Active stereo with structured light





Easy-to-match pattern

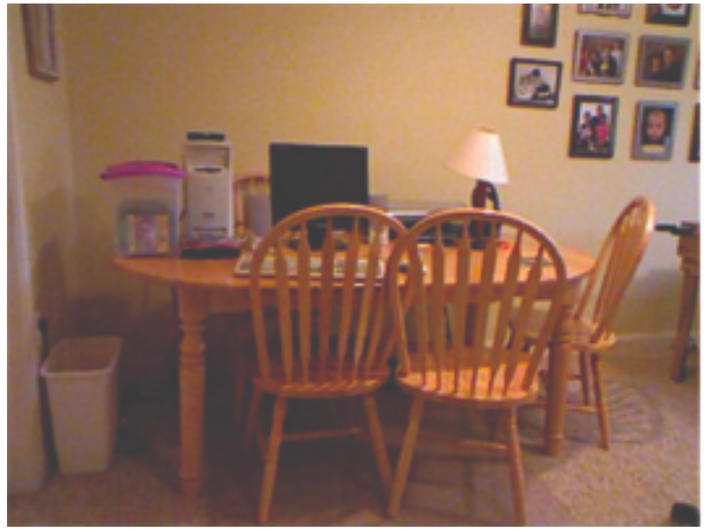


Do we really need the second camera?

[Zhang, Curless, Seitz, 2002]

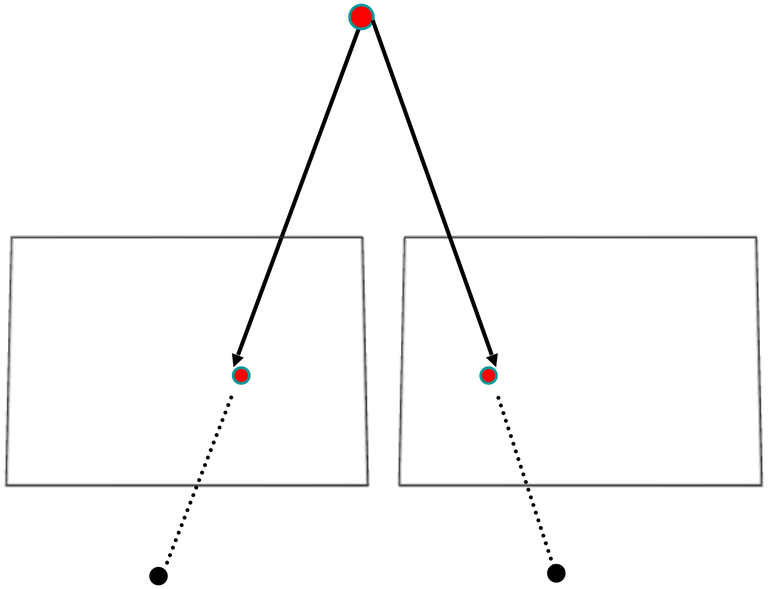
Source: R. Szeliski

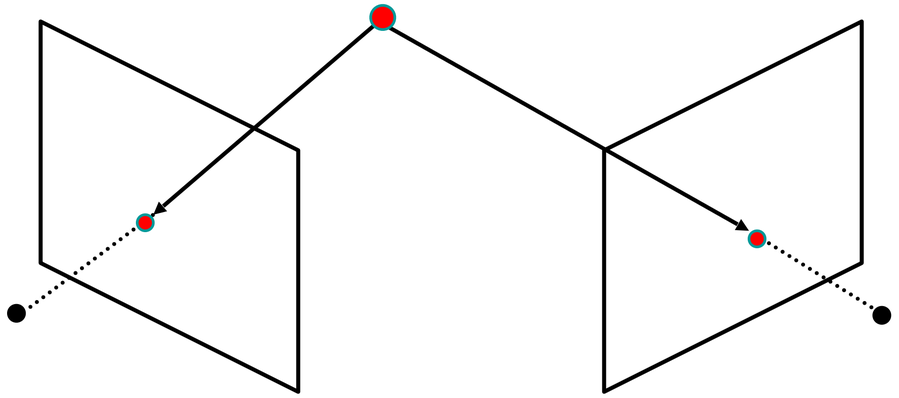
RGB-D sensors



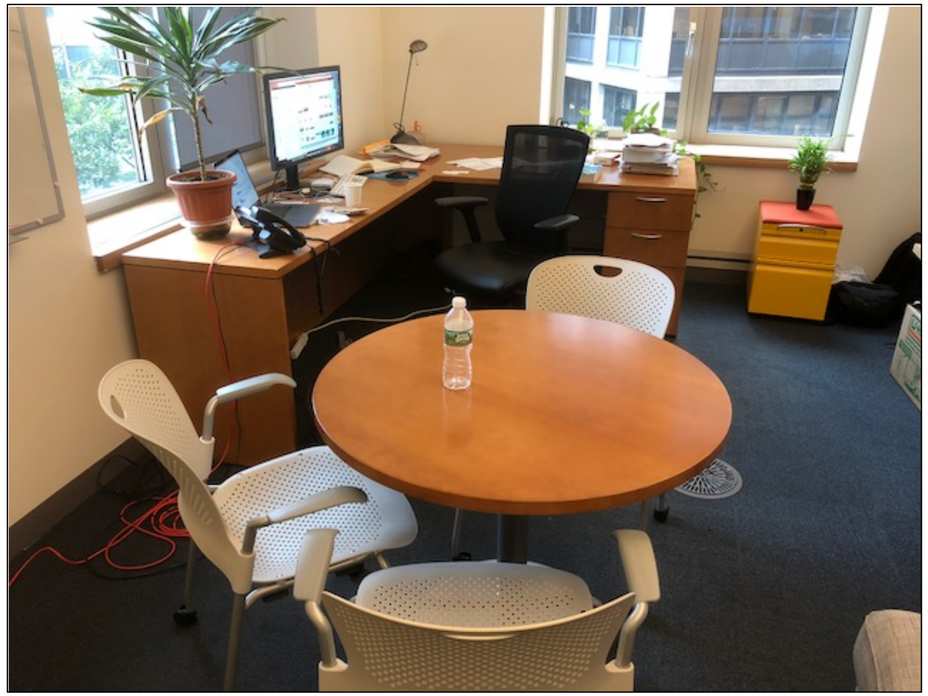
General case

- The two cameras need not have parallel optical axes.

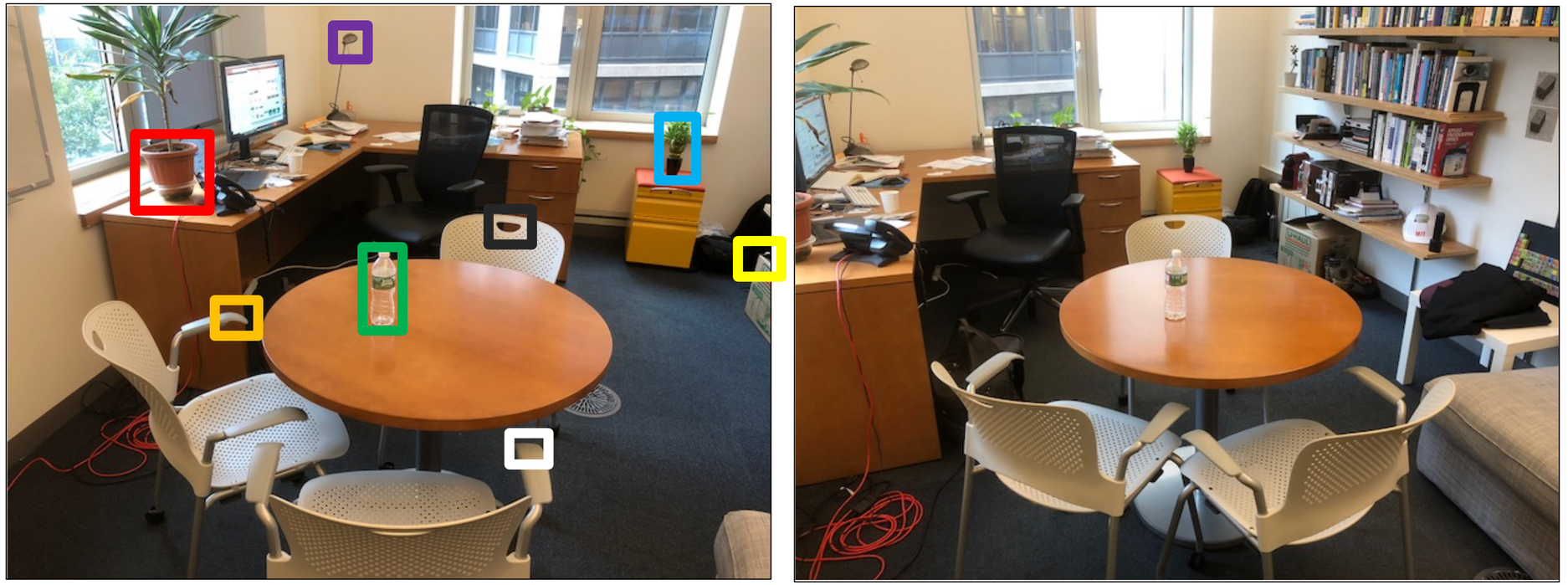




Source: Torralba, Isola, Freeman

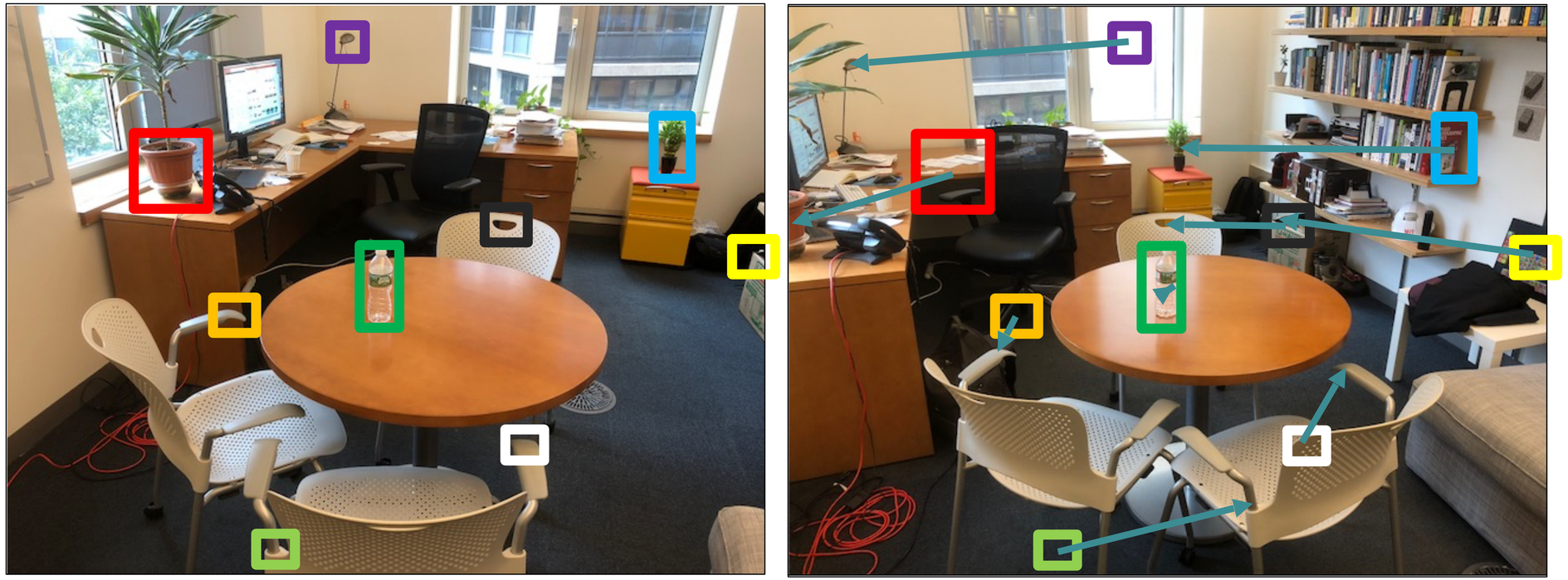


Source: Torralba, Isola, Freeman



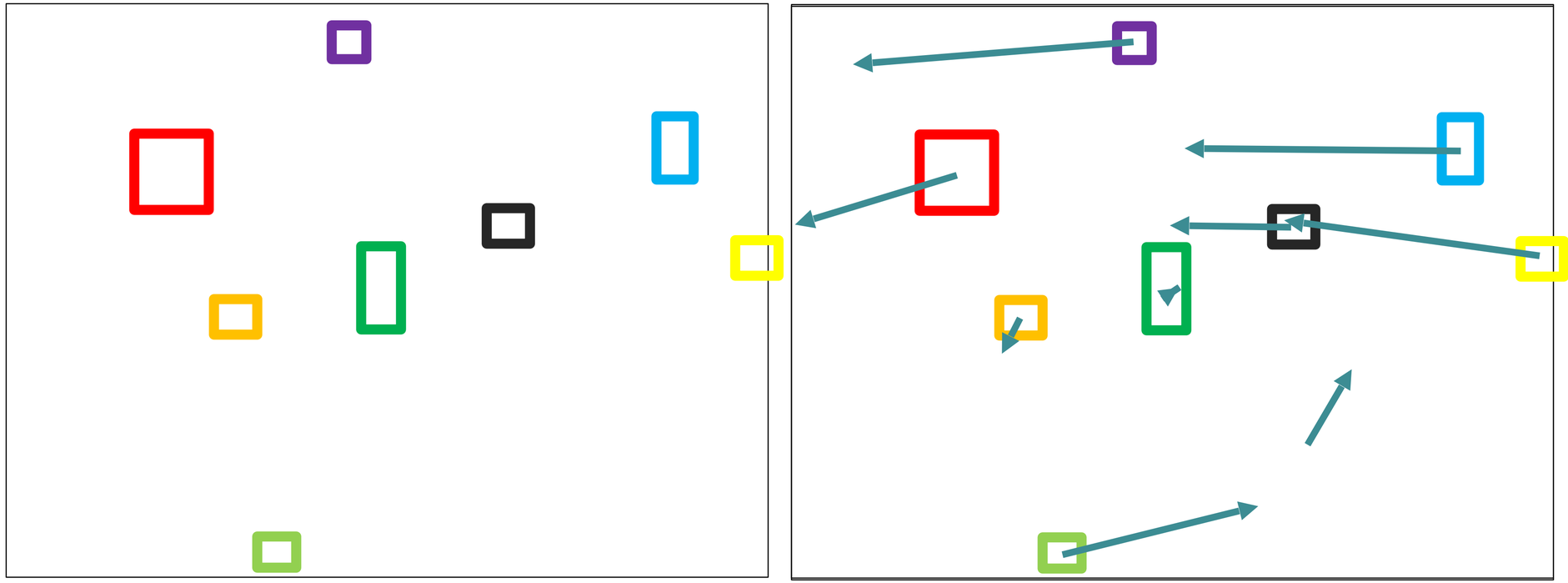
Do we need to search for matches only along horizontal lines?

Source: Torralba, Isola, Freeman



Do we need to search for matches only along horizontal lines?

Source: Torralba, Isola, Freeman

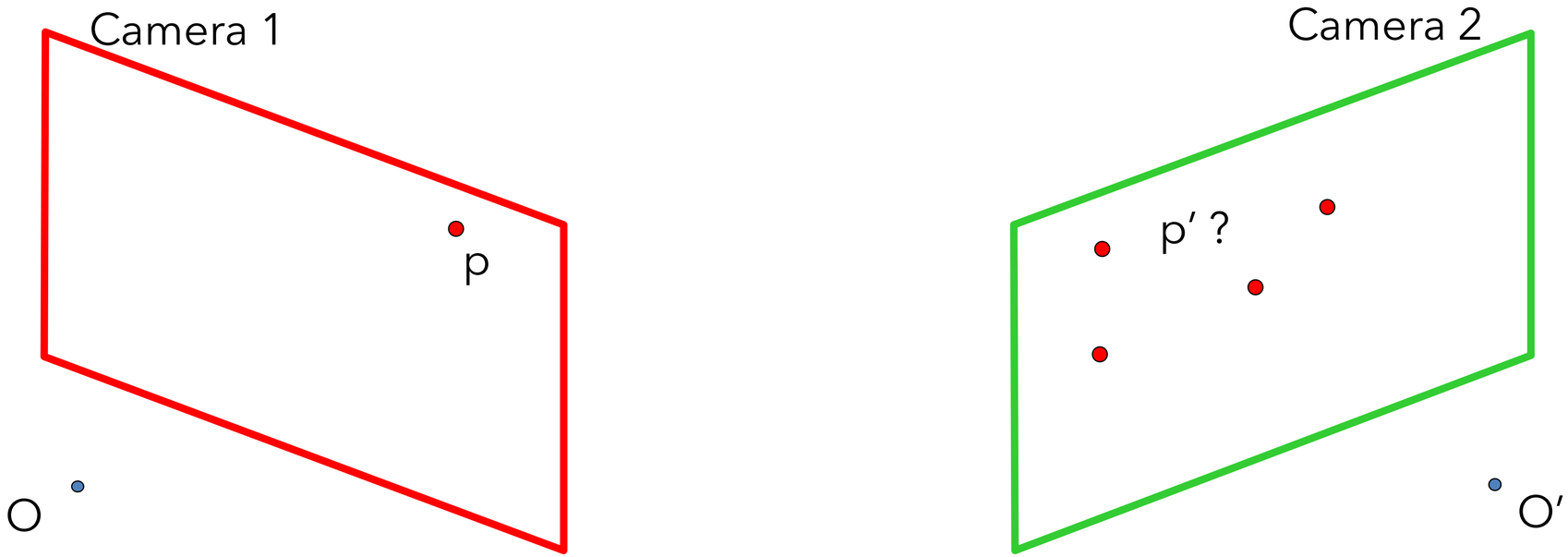


~~Do we need to search for matches only along horizontal lines?~~

It looks like we might need to search everywhere… are there any constraints that can guide the search?

Source: Torralba, Isola, Freeman

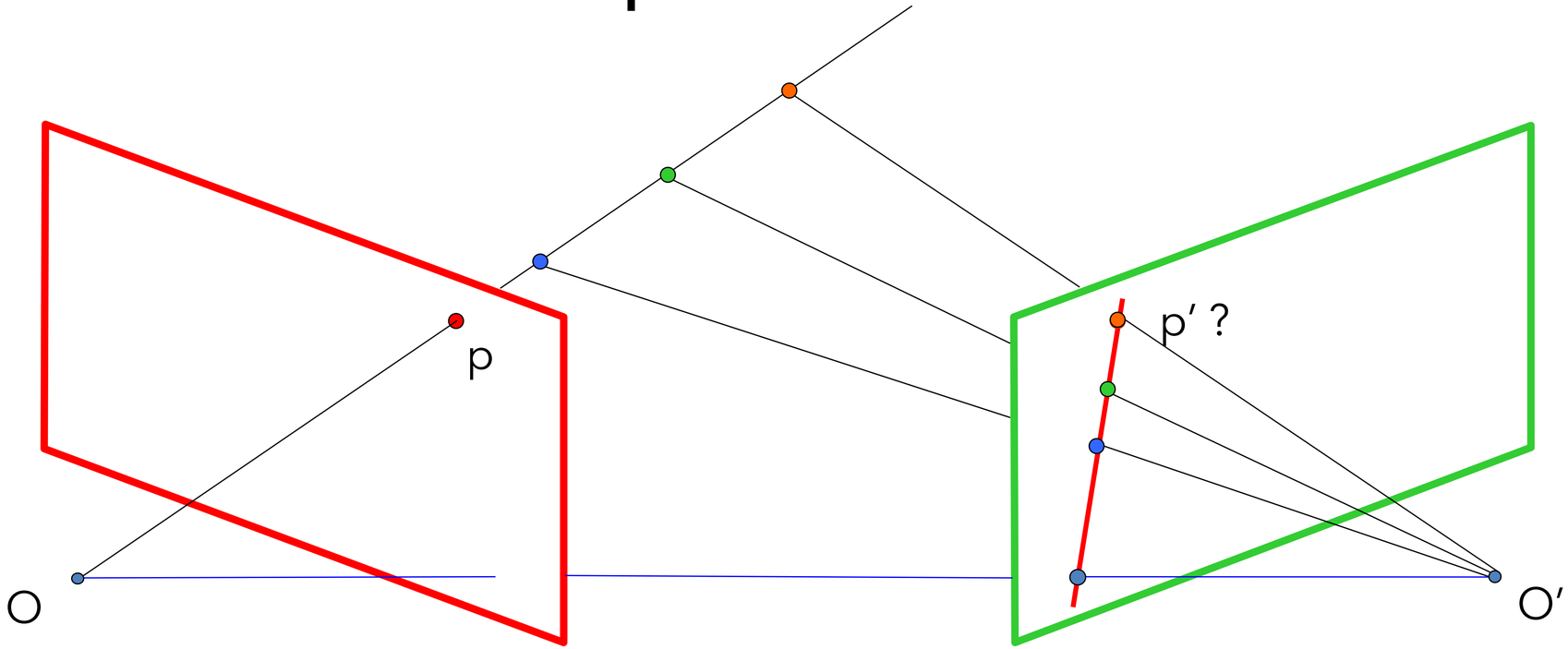
Stereo correspondence constraints



If we see a point in camera 1, are there any constraints on where we will find it on camera 2?

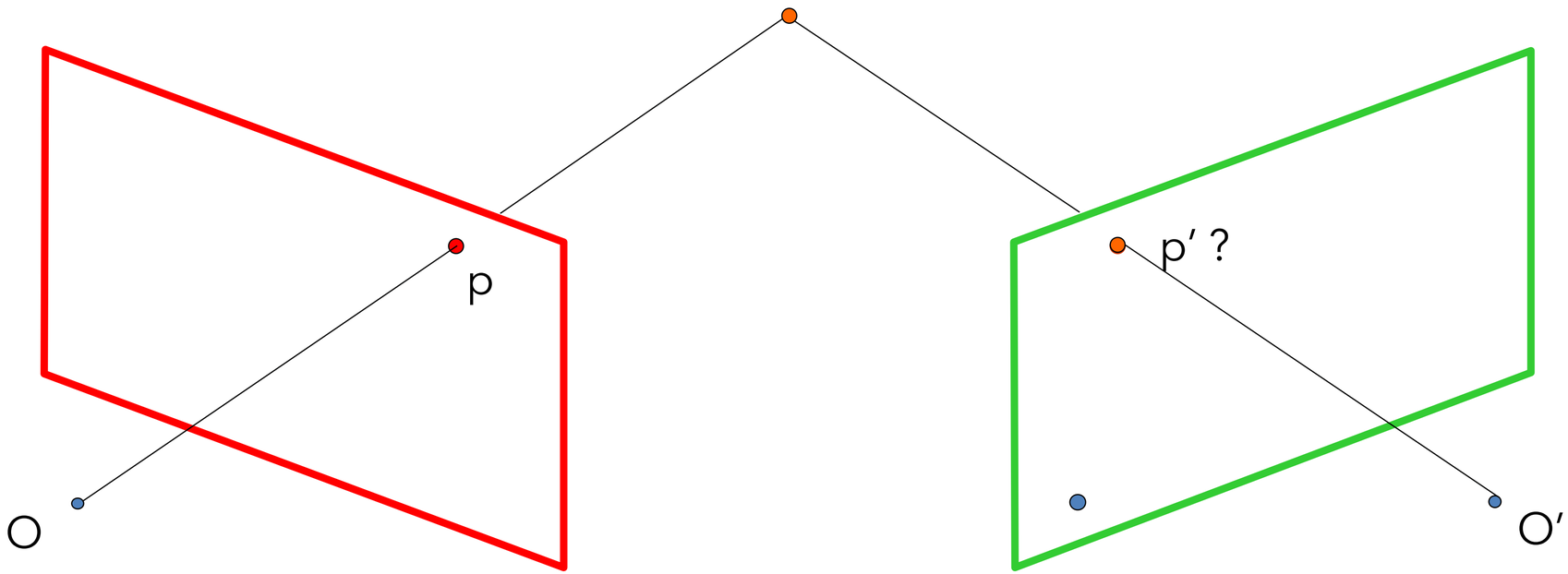
Source: Torralba, Isola, Freeman

Stereo correspondence constraints



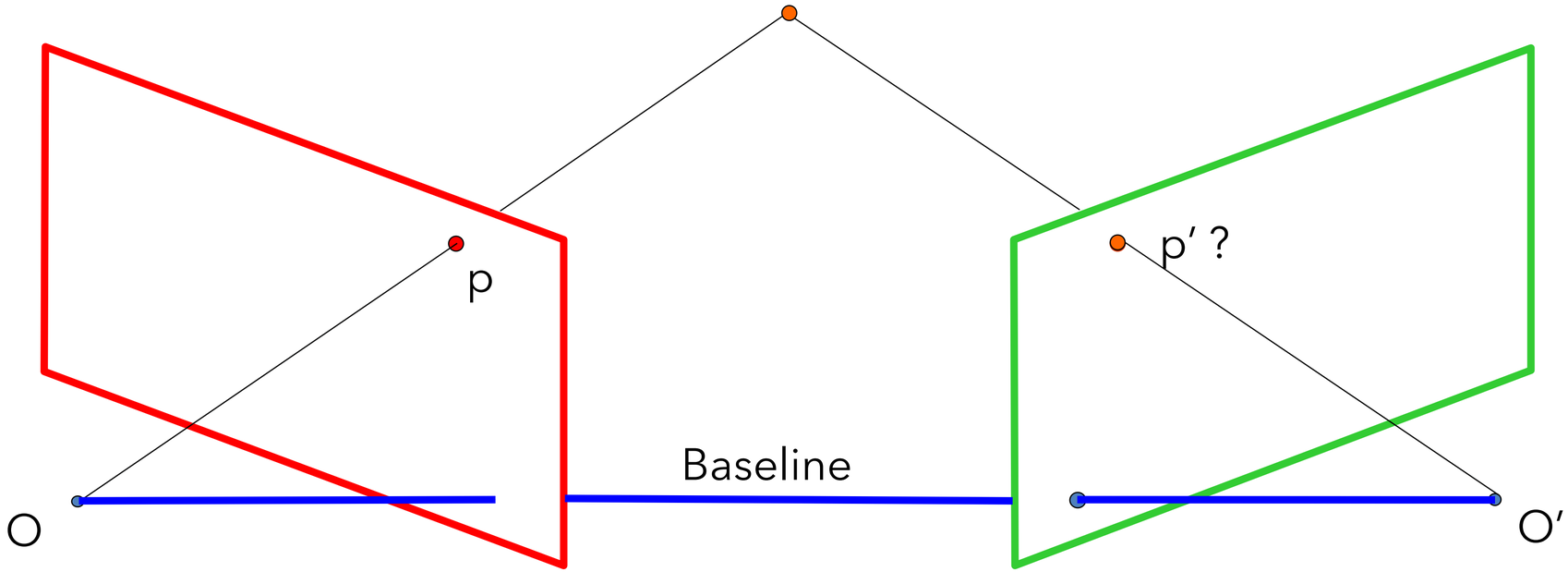
Source: Torralba, Isola, Freeman

Some terminology



Source: Torralba, Isola, Freeman

Some terminology

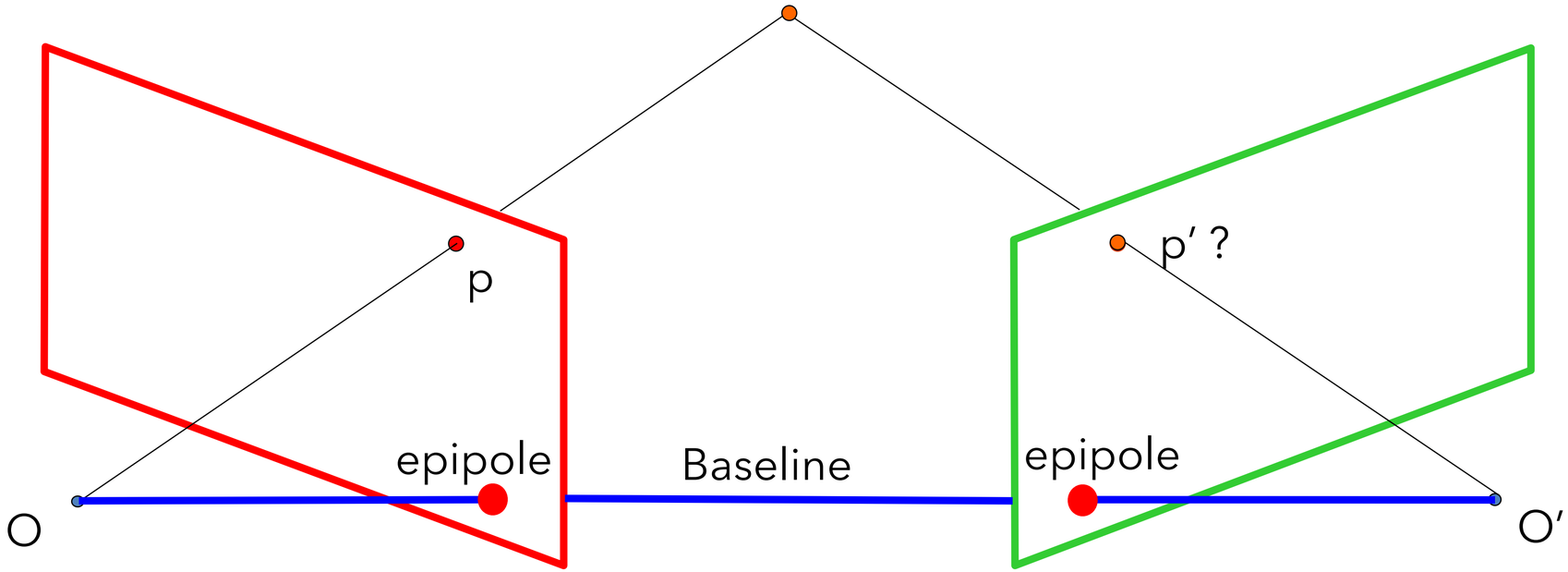


Baseline: the line connecting the two camera centers

Epipole: point of intersection of *baseline* with the image plane

Source: Torralba, Isola, Freeman

Some terminology

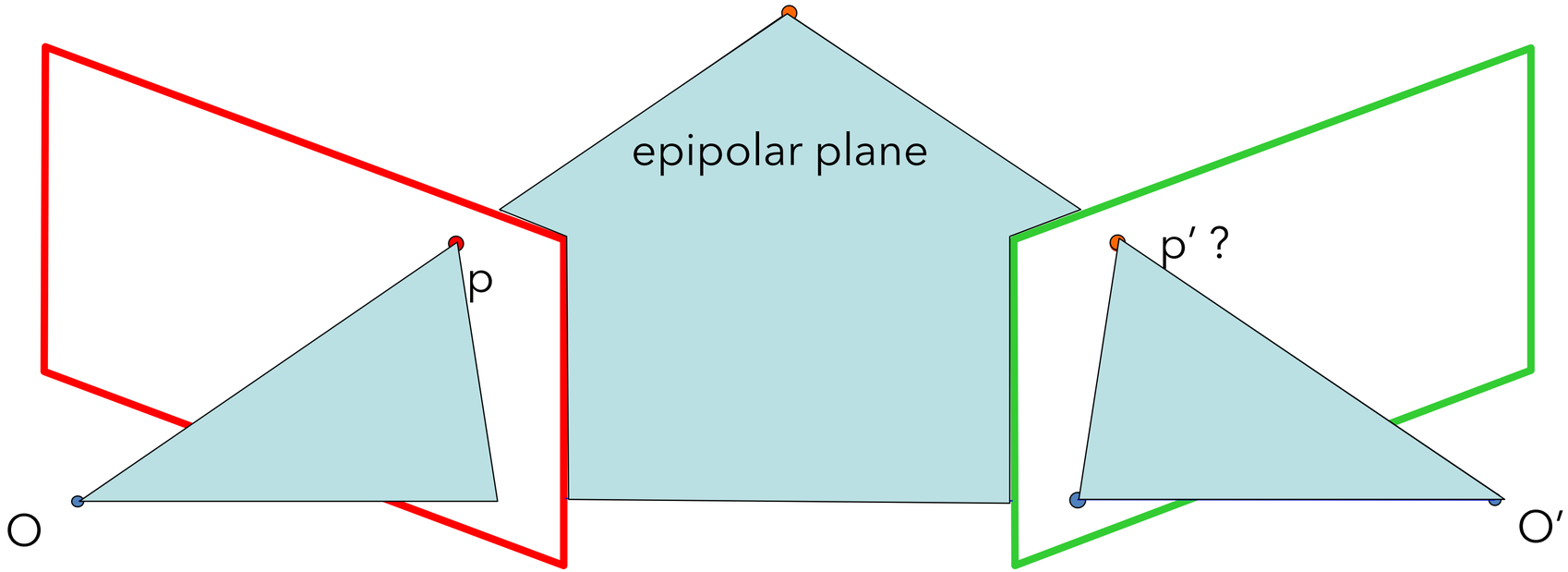


Baseline: the line connecting the two camera centers

Epipole: point of intersection of *baseline* with the image plane

Source: Torralba, Isola, Freeman

Some terminology



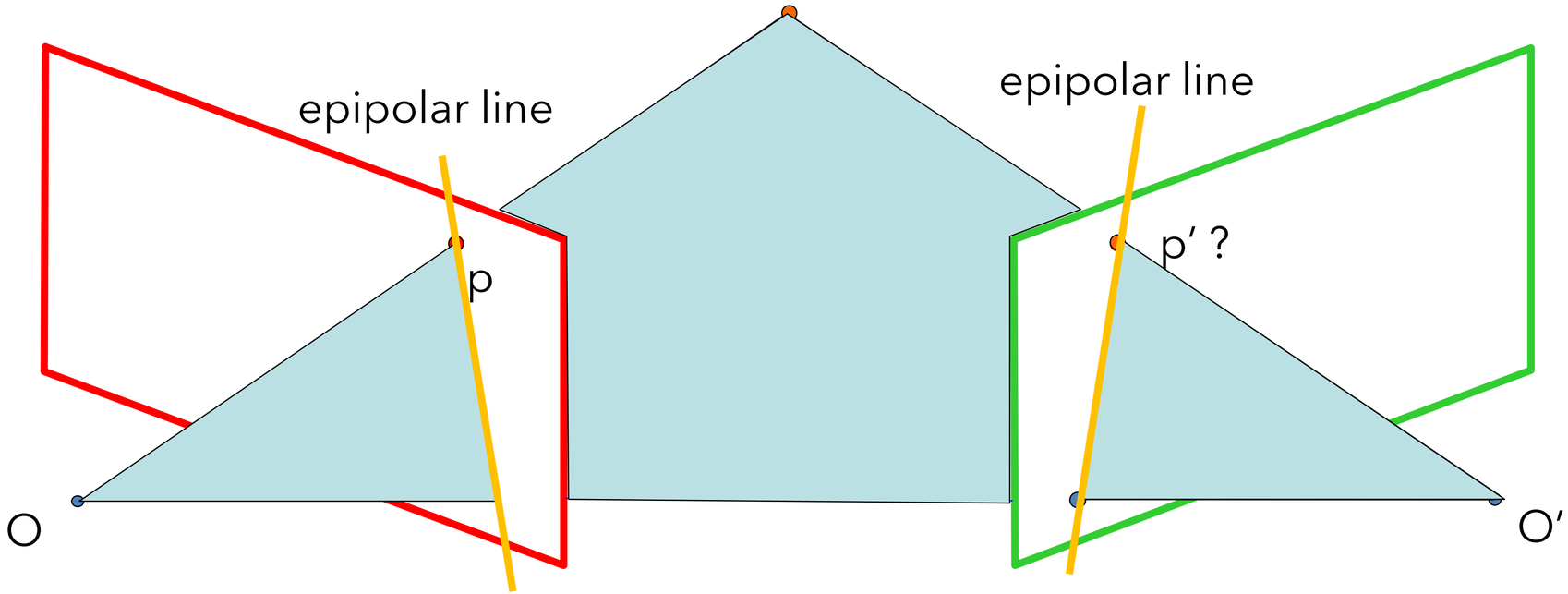
Baseline: the line connecting the two camera centers

Epipole: point of intersection of *baseline* with the image plane

Epipolar plane: the plane that contains the two camera centers and a 3D point in the world

Source: Torralba, Isola, Freeman

Some terminology



Baseline: the line connecting the two camera centers

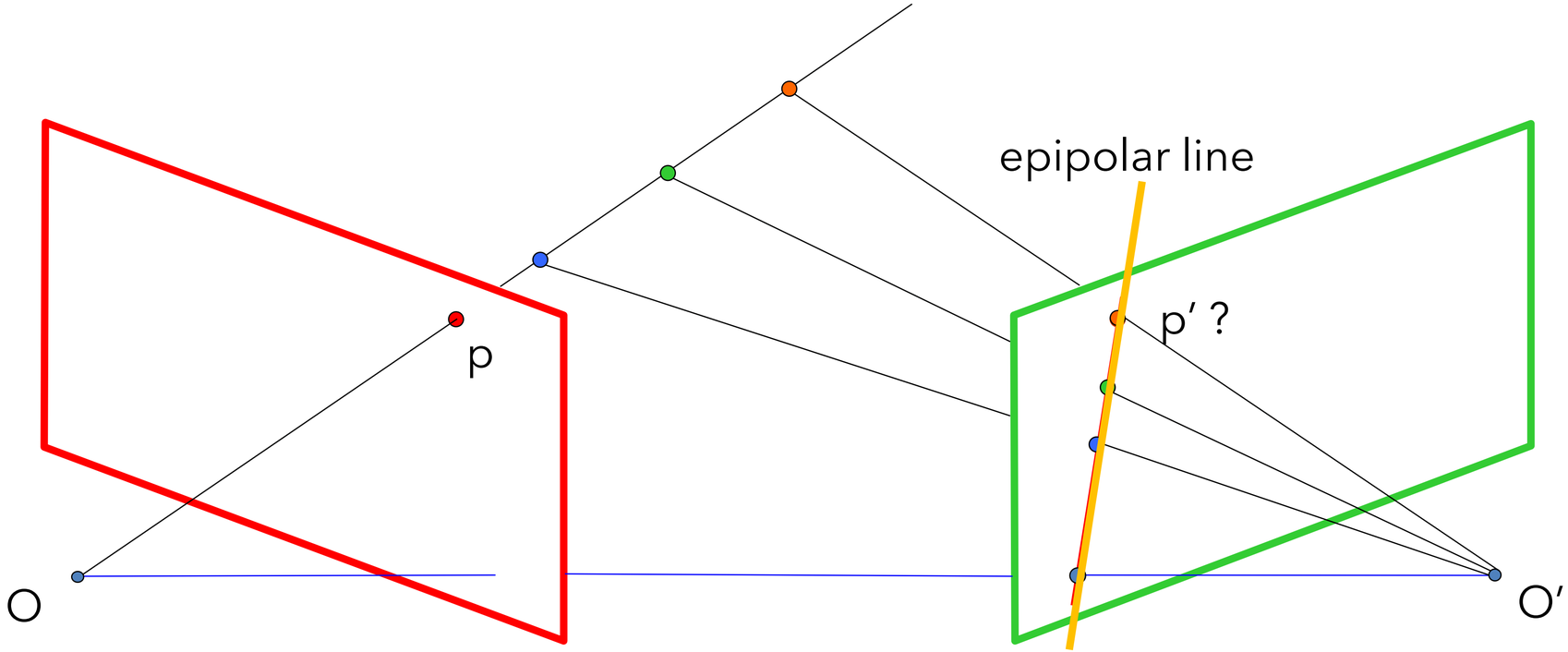
Epipole: point of intersection of *baseline* with the image plane

Epipolar plane: the plane that contains the two camera centers and a 3D point in the world

Epipolar line: intersection of the *epipolar plane* with each image plane

Source: Torralba, Isola, Freeman

Epipolar constraint

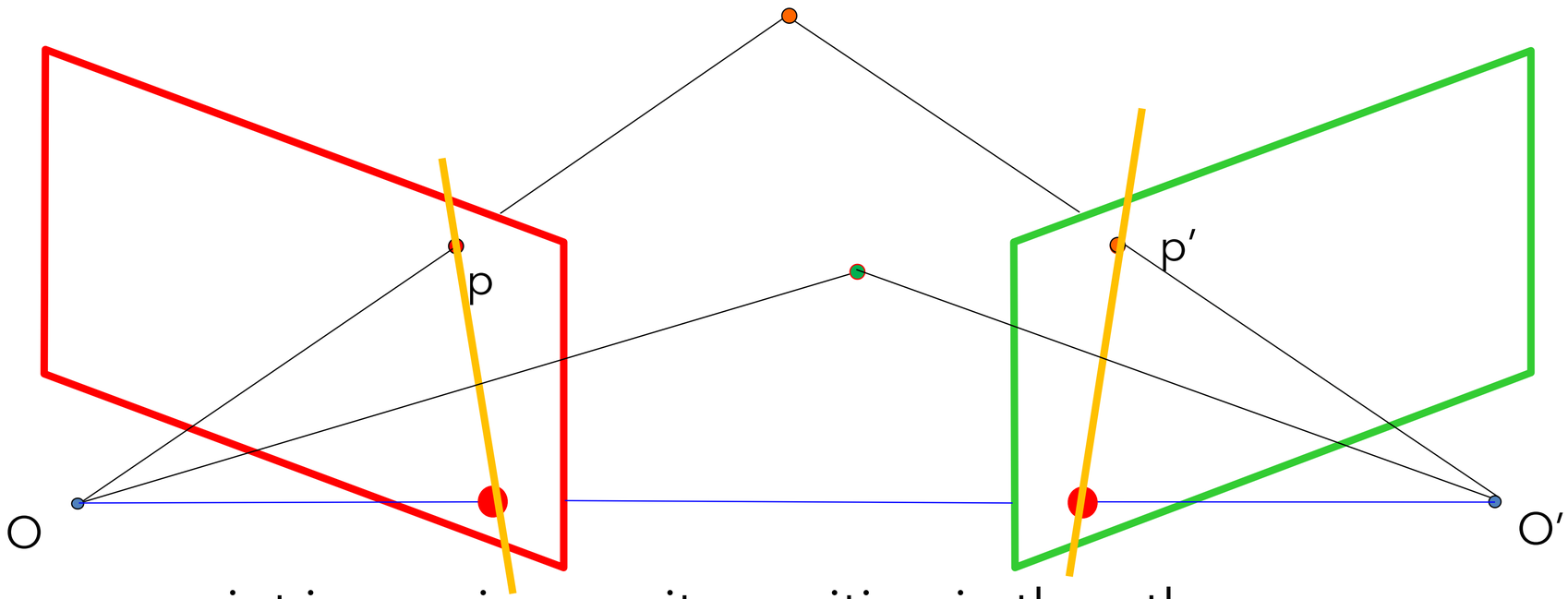


We can search for matches across epipolar lines

All epipolar lines intersect at the epipoles

Source: Torralba, Isola, Freeman

Epipolar constraint

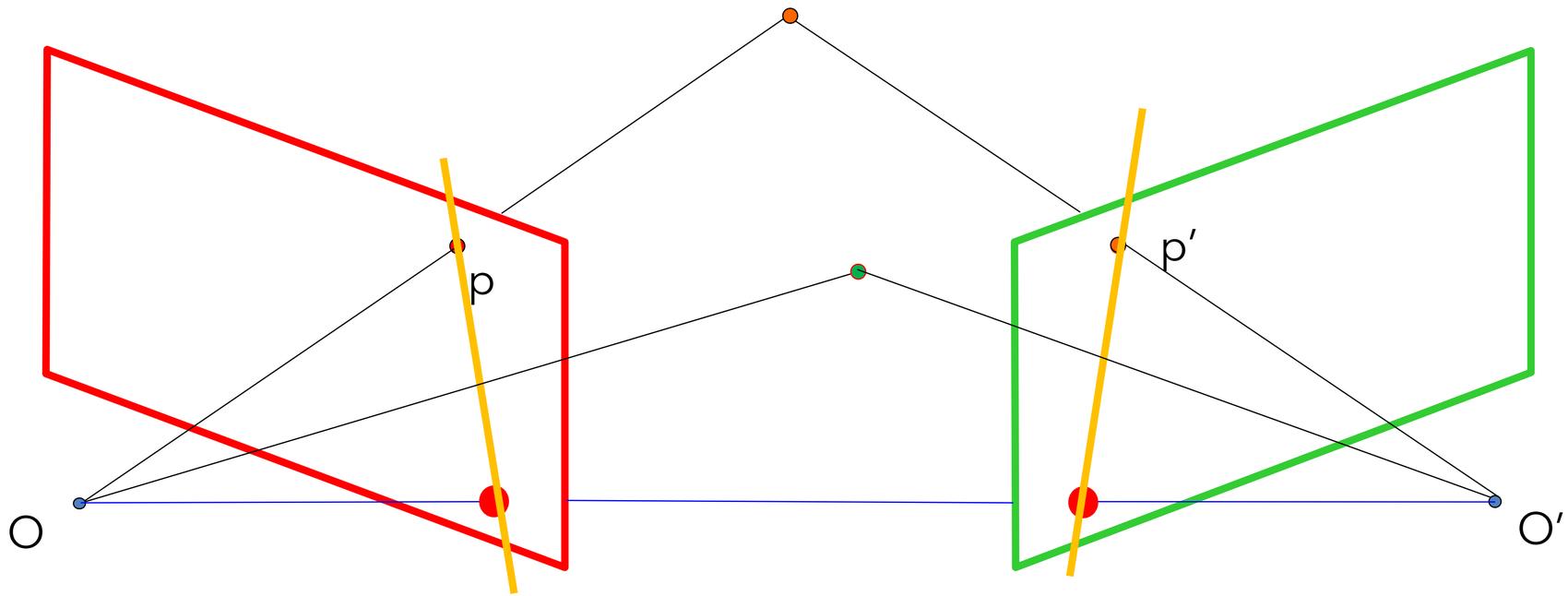


- If we observe a point in one image, its position in the other image is constrained to lie on the epipolar line.
- How do we get this line? We want a function that, given a point $\mathbf{p}$ tells us the line:

$$f(\mathbf{p}) = [a, b, c] \quad \text{such that} \quad ax' + by' + c = 0$$

where $\mathbf{p}' = [x', y']$. In other words: $f(\mathbf{p})^\top \mathbf{p}' = 0$

The fundamental matrix



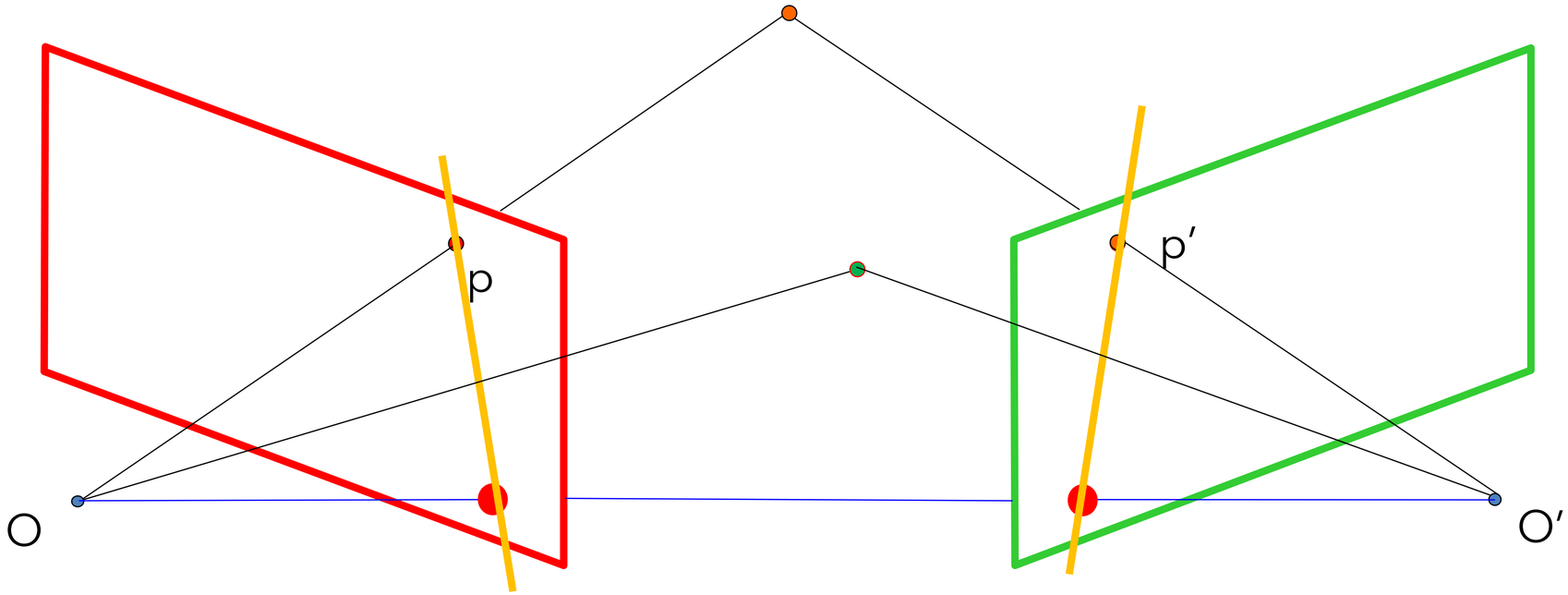
It can be shown that our function, f, can be written as a matrix multiplication in homogeneous coordinates!

$$f(\mathbf{p}) = \mathbf{p}^\top F = [a, b, c]$$

F: the **fundamental matrix**

p: image point in homogeneous coordinates

The fundamental matrix



It can be shown that our function, f, can be written as a matrix multiplication in homogeneous coordinates!

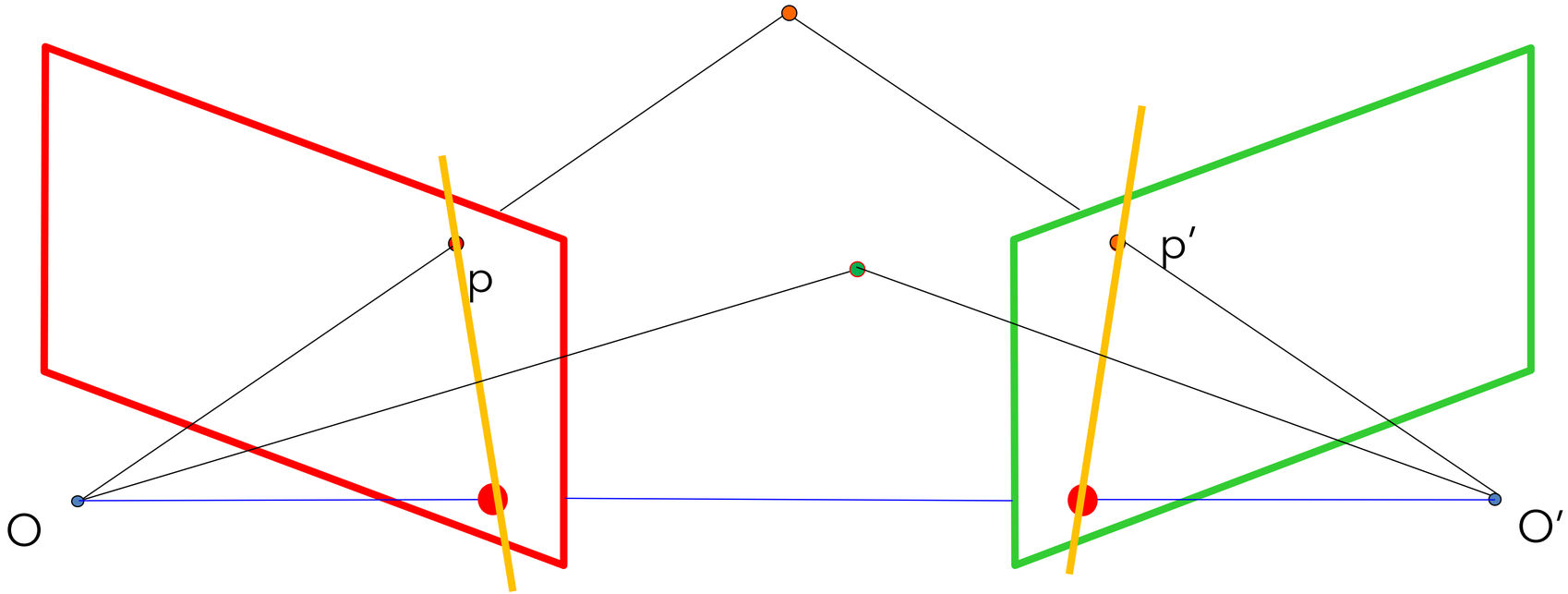
More concisely:

$$\mathbf{p}^\top F\ \mathbf{p}' = 0$$

F: the **fundamental matrix**
p, p′: corresponding image points

Source: Torralba, Isola, Freeman

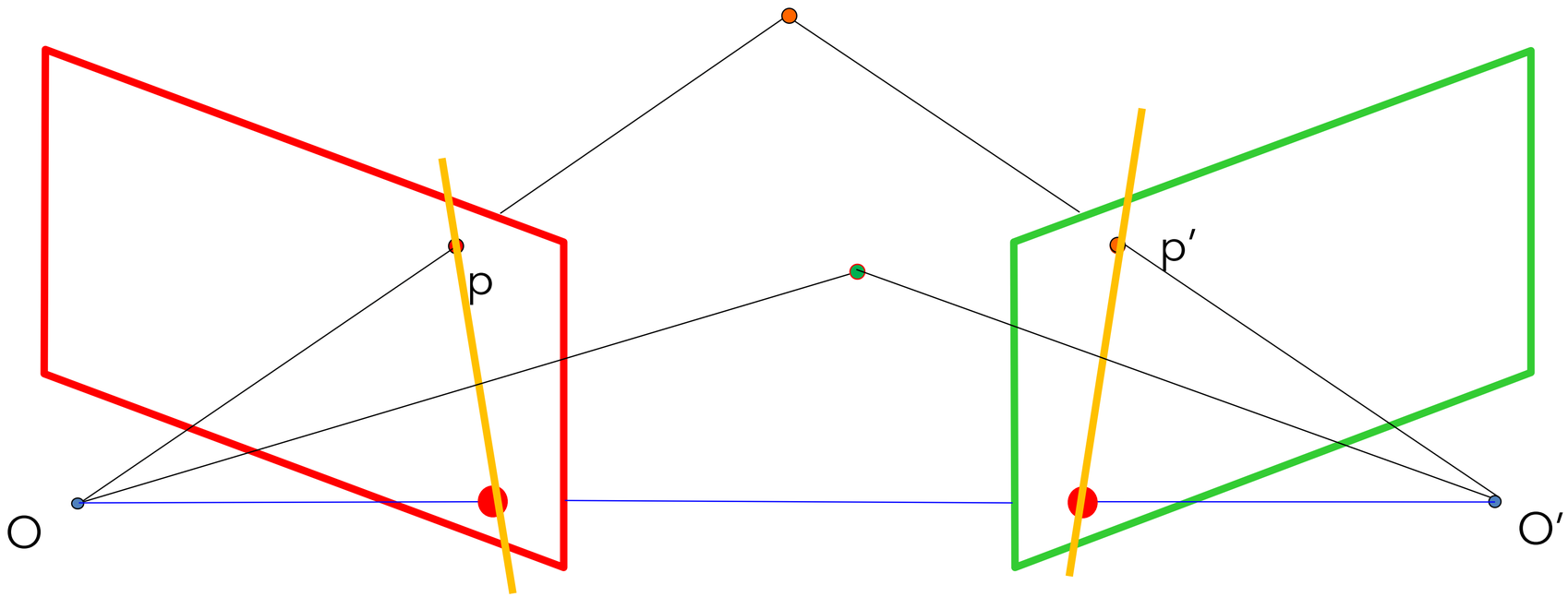
The fundamental matrix



How this works:

$$\mathbf{p}^\top F\ \mathbf{p}' = 0$$

The fundamental matrix



How this works:

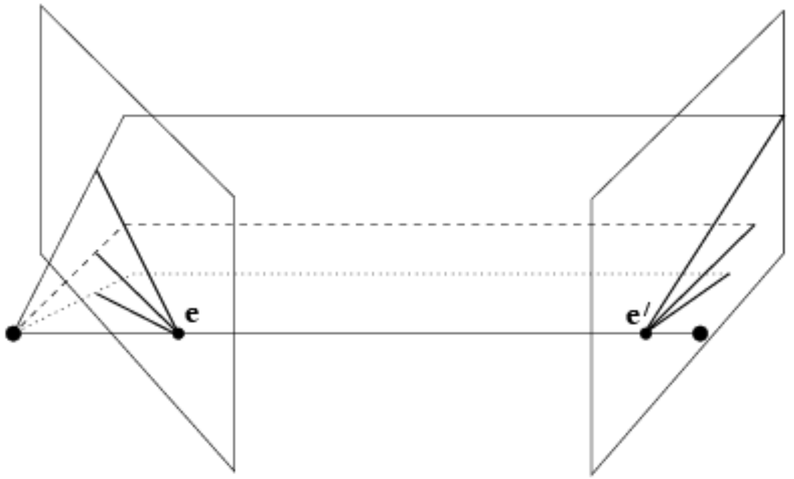
$$(\mathbf{p}^\top F)\ \mathbf{p}' = 0$$

$$\mathbf{u}^\top \mathbf{p}' = 0$$

u: a line induced by p

p, p’: image points in homogeneous coordinates

Example: converging cameras



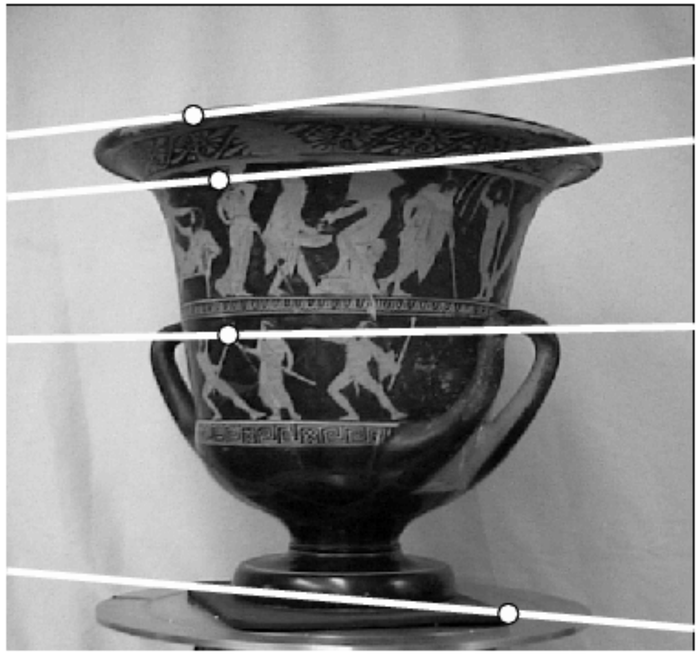
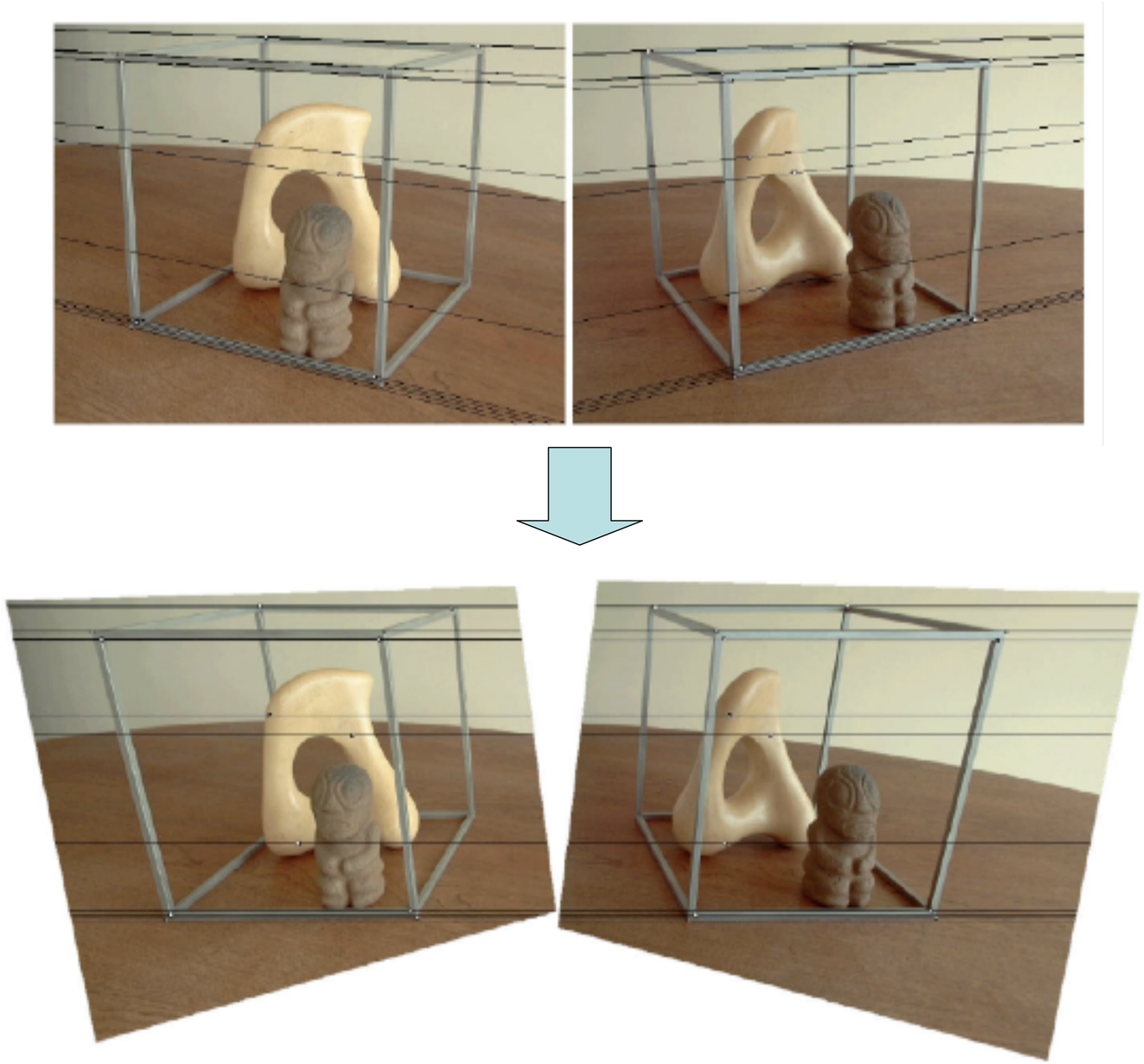


Figure from Hartley & Zisserman

Source: Kristen Grauman

Image rectification

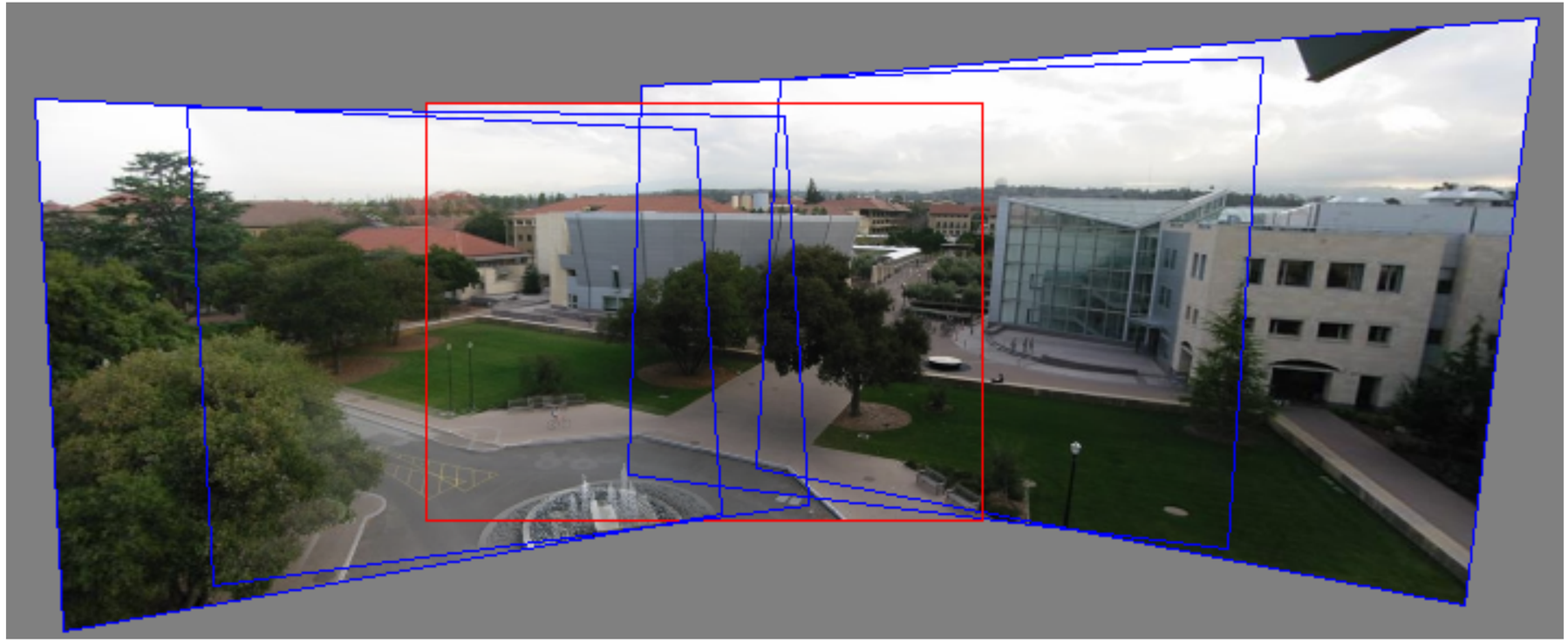


Source: A. Efros

Example

Source: N. Snavely

Making panoramas



Source: N. Snavely

Making panoramas

What is the geometric relationship between these two images?

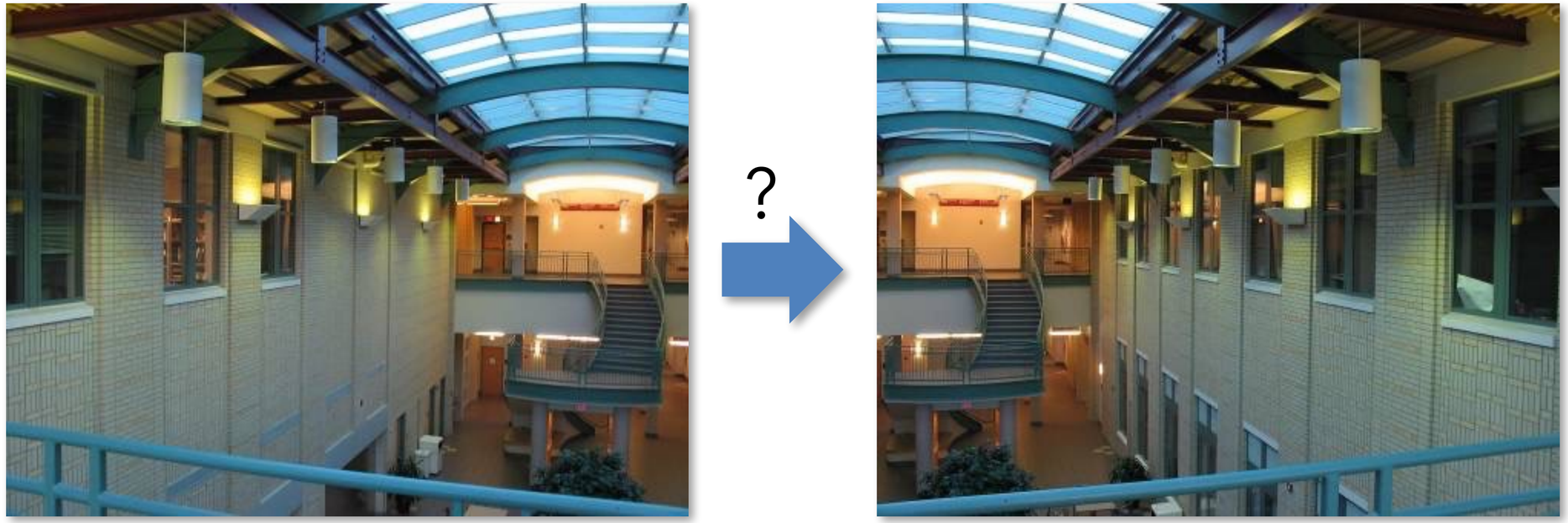


Image alignment

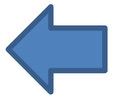
Why don’t these image line up exactly?

Source: N. Snavely

Recall: affine transformations

$$\begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix}$$

affine transformation



what happens when we change this row?

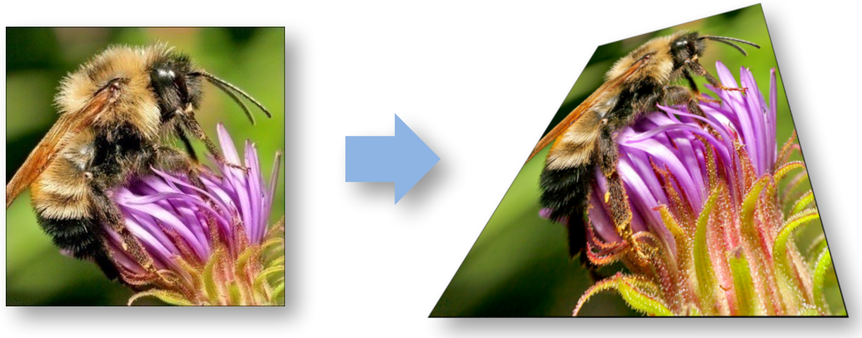
Source: N. Snavely

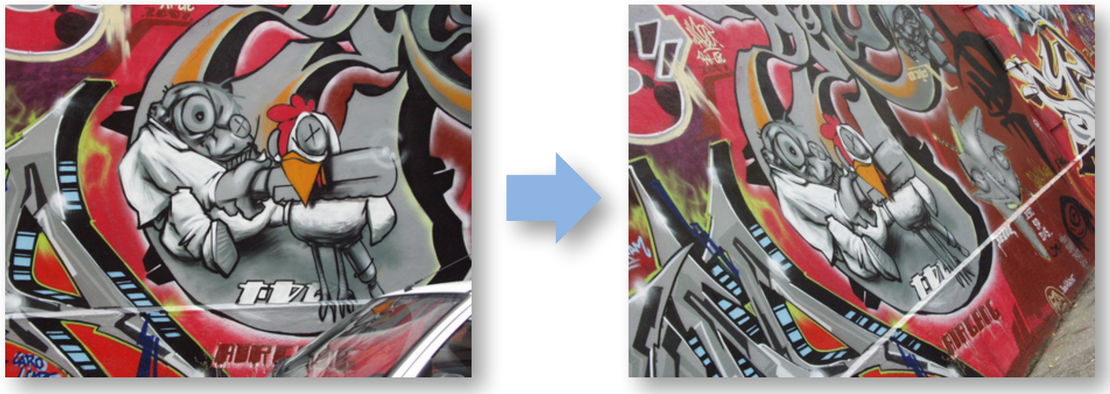
Projective Transformations aka Homographies aka Planar Perspective Maps

$$\mathbf{H} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & 1 \end{bmatrix}$$

Called a **homography**

(or planar perspective map)





Source: N. Snavely

Homographies

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Note that this can be 0!
A "point at infinity"

$$\sim \begin{bmatrix} \frac{ax+by+c}{gx+hy+1} \\ \frac{dx+ey+f}{gx+hy+1} \\ 1 \end{bmatrix}$$

Source: N. Snavely

Homography

Example: two pictures taken by rotating the camera:

If we try to build a panorama by overlapping them:

Source: Torralba, Isola, Freeman

Homography

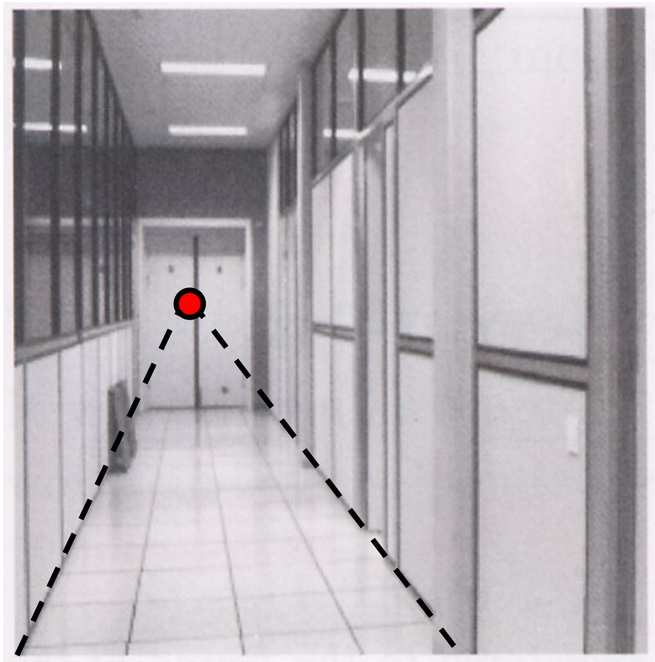
Example: two pictures taken by rotating the camera:

With a homography you can map both images into a single camera:

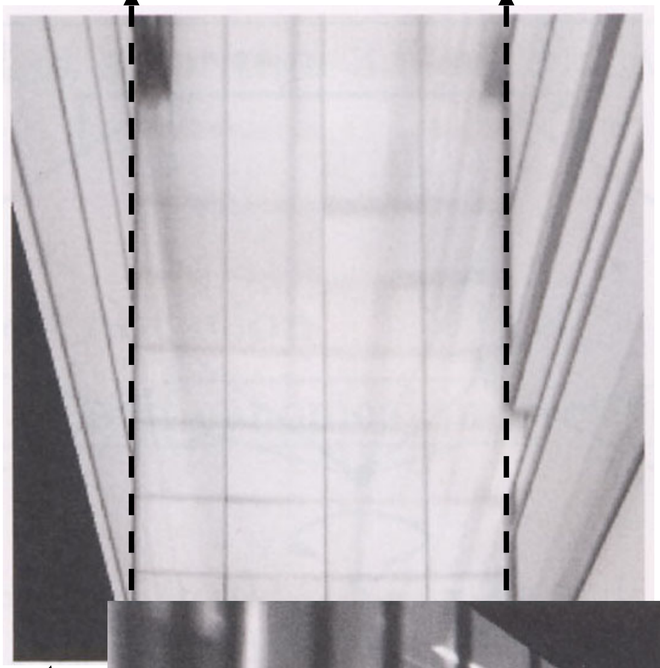
We'll see why in PS5!

Source: Torralba, Isola, Freeman

Plane-to-plane homography



$\mathbf{H}_1$



$\mathbf{H}_2$

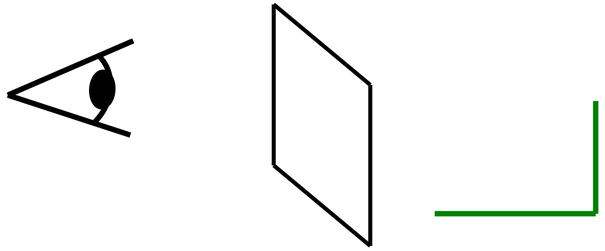


image plane in front

black area
where no pixel
maps to

Source: N. Snavely

Homographies

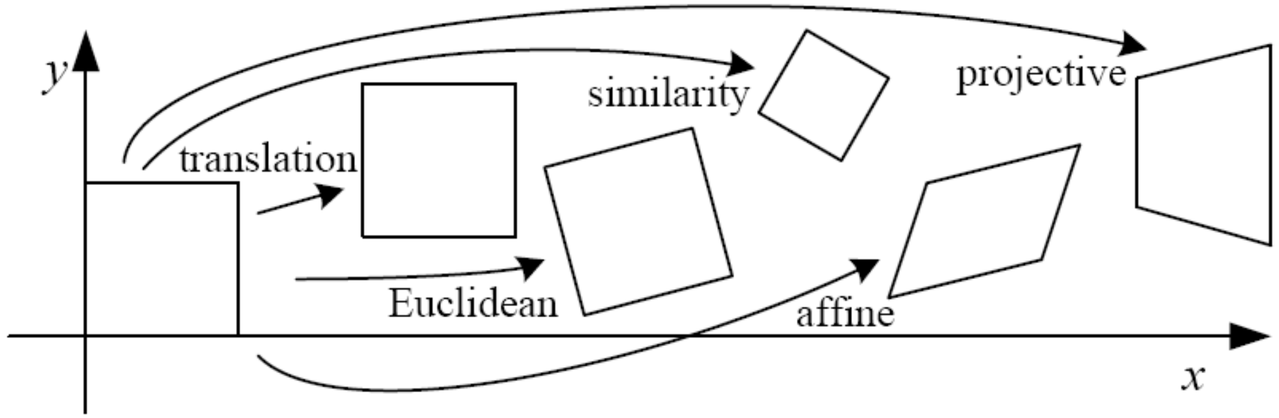
- Homographies …
  - Affine transformations, and
  - Projective warps

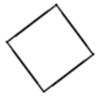
$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

- Properties of projective transformations:
  - Origin does not necessarily map to origin
  - Lines map to lines
  - Parallel lines do not necessarily remain parallel
  - Closed under composition

Source: N. Snavely

2D image transformations



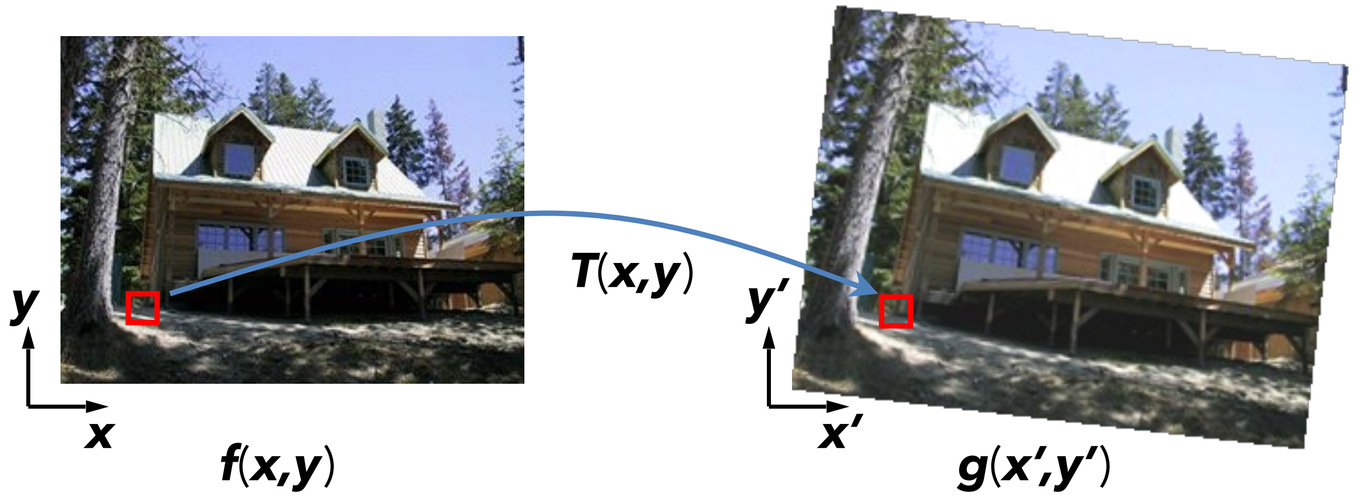
Name	Matrix	# D.O.F.	Preserves:	Icon
translation	$\begin{bmatrix} \boldsymbol{I} \mid \boldsymbol{t} \end{bmatrix}_{2\times3}$	2	orientation $+\cdots$	
rigid (Euclidean)	$\begin{bmatrix} \boldsymbol{R} \mid \boldsymbol{t} \end{bmatrix}_{2\times3}$	3	lengths $+\cdots$	
similarity	$\begin{bmatrix} s\boldsymbol{R} \mid \boldsymbol{t} \end{bmatrix}_{2\times3}$	4	angles $+\cdots$	
affine	$\begin{bmatrix} \boldsymbol{A} \end{bmatrix}_{2\times3}$	6	parallelism $+\cdots$	
projective	$\begin{bmatrix} \tilde{\boldsymbol{H}} \end{bmatrix}_{3\times3}$	8	straight lines	

Source: N. Snavely

How do we perform this warp?

Image warping

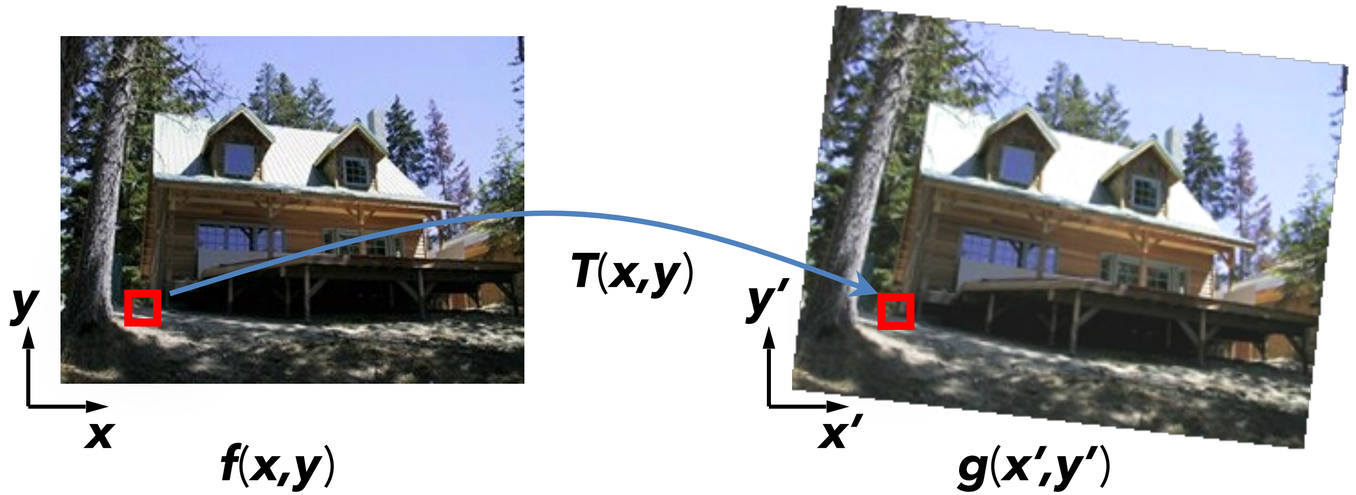
Given a coordinate transformation $(x', y') = T(x, y)$ and a source image $f(x, y)$, how do we compute a transformed image $g(x', y') = f(T(x, y))$?



Source: N. Snavely

Forward warping

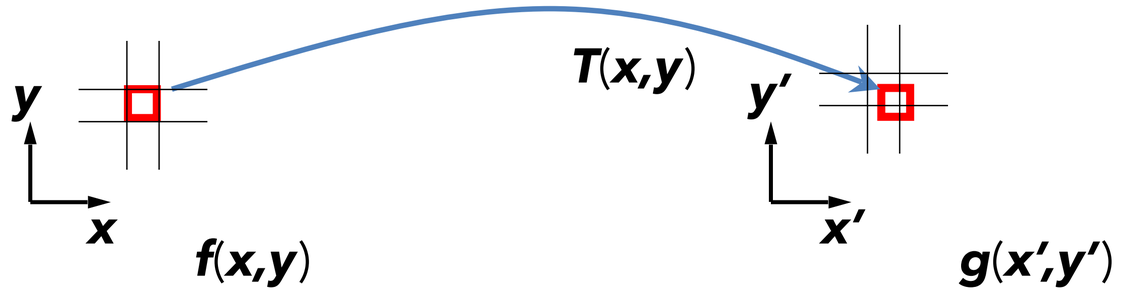
- Send each pixel $\boldsymbol{f(x)}$ to its corresponding location $\boldsymbol{(x',y')} = \boldsymbol{T(x,y)}$ in $\boldsymbol{g(x',y')}$
- What if a pixel lands "between" two pixels?



Source: N. Snavely

Forward warping

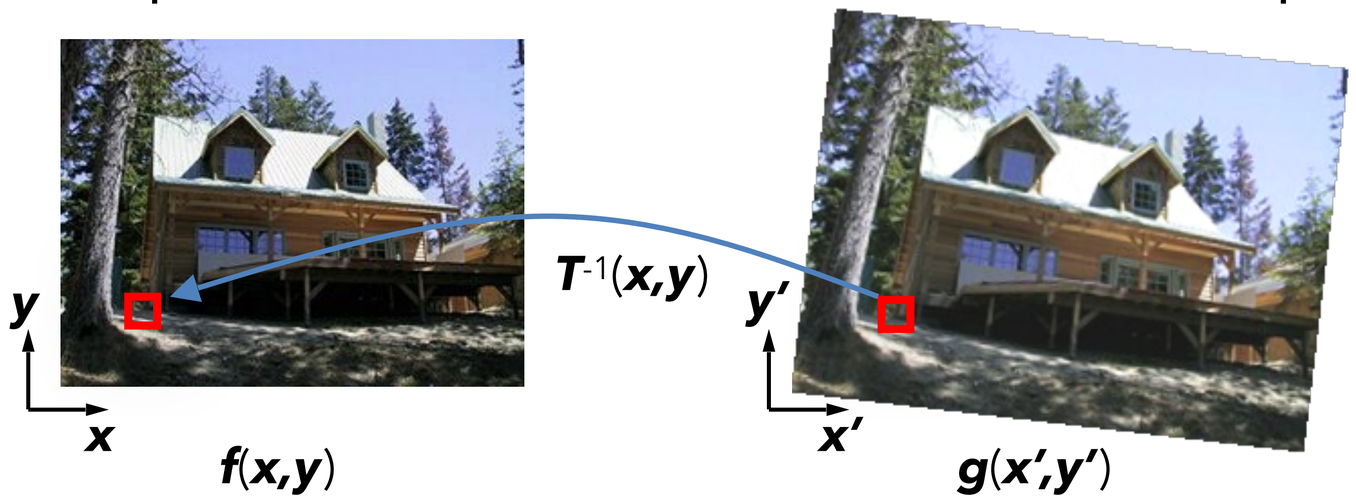
- Send each pixel $\boldsymbol{f(x)}$ to its corresponding location $\boldsymbol{(x',y')} = \boldsymbol{T(x,y)}$ in $\boldsymbol{g(x',y')}$
- What if a pixel lands "between" two pixels?
  - Answer: add "contribution" to several pixels, normalize later (*splatting*)
  - Can still result in holes



Source: N. Snavely

Backward (inverse) warping

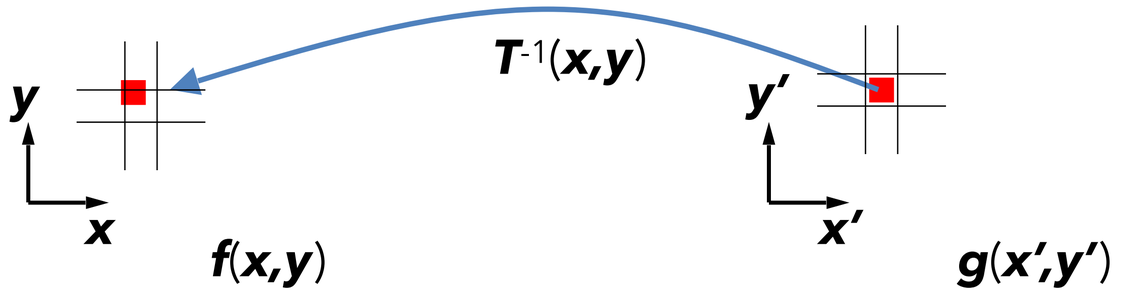
- Get each pixel $g(x',y')$ from its corresponding location $(x,y) = T^{-1}(x,y)$ in $f(x,y)$
  - Requires taking the inverse of the transform
  - What if pixel comes from "between" two pixels?



Source: N. Snavely

Backward (inverse) warping

- Get each pixel $g(x')$ from its corresponding location $x' = h(x)$ in $f(x)$
  - What if pixel comes from "between" two pixels?
  - Answer: *resample* color value from *interpolated* source image



Source: N. Snavely

Next class: estimating geometry from images