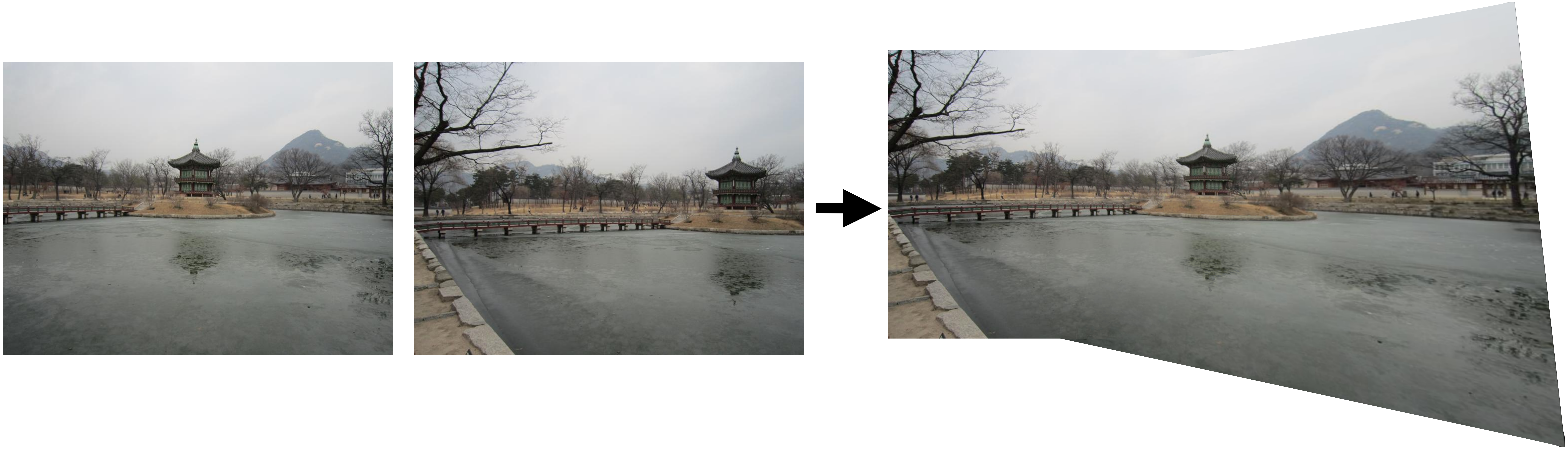
Lecture 17: Multi-view geometry

Today

- Epipolar geometry
- Stereo matching
- Image alignment

PS8: making panoramas



Source: N. Snavely

Recall: homogeneous coordinates

Representing translations:

$$(x, y) \Rightarrow \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

homogeneous image coordinates

$$\begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x + t_x \\ y + t_y \\ 1 \end{bmatrix}$$

Converting back to image coordinates

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} \Rightarrow (x/w, y/w)$$

Source: N. Snavely

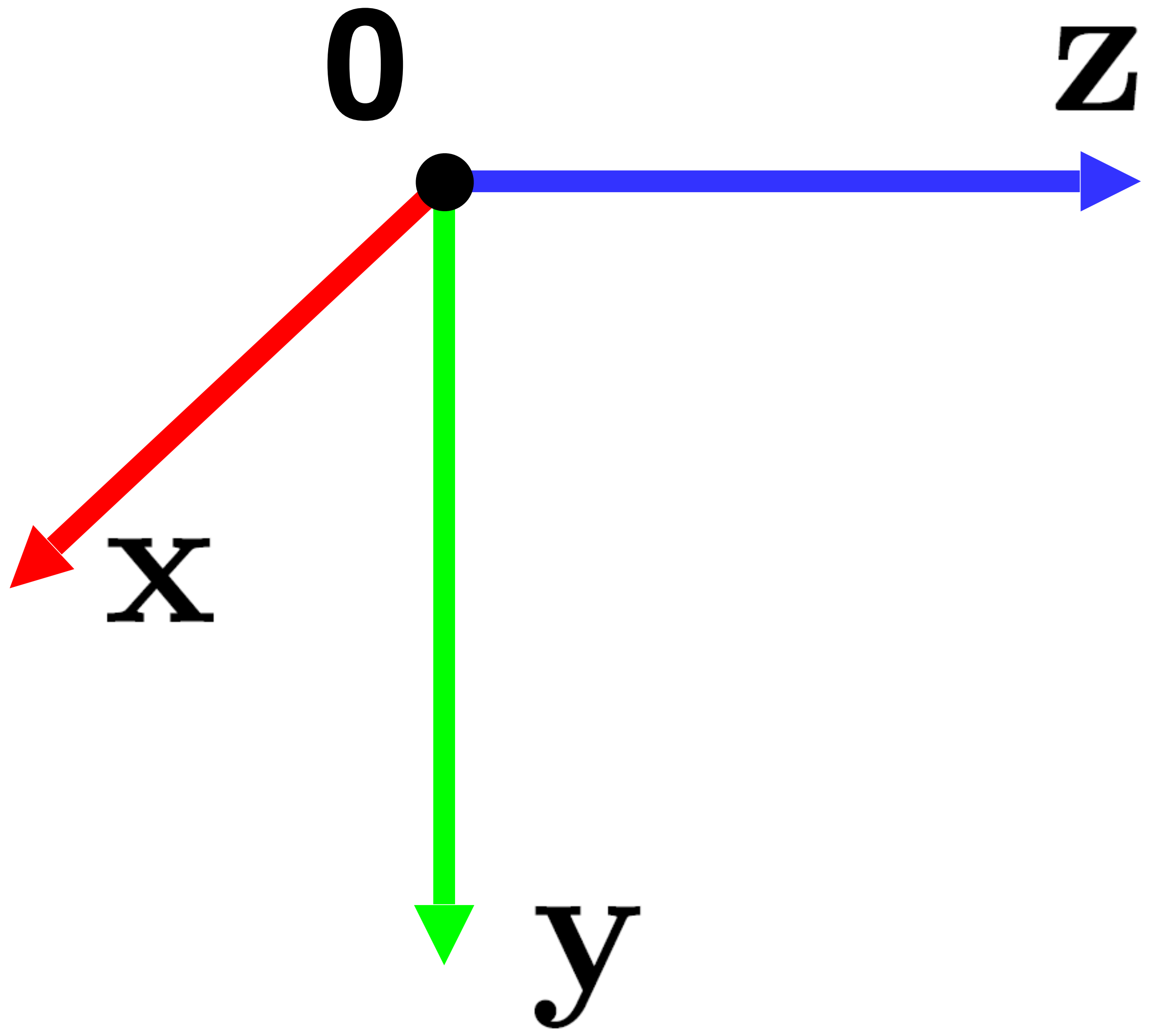
Recall: camera parameters

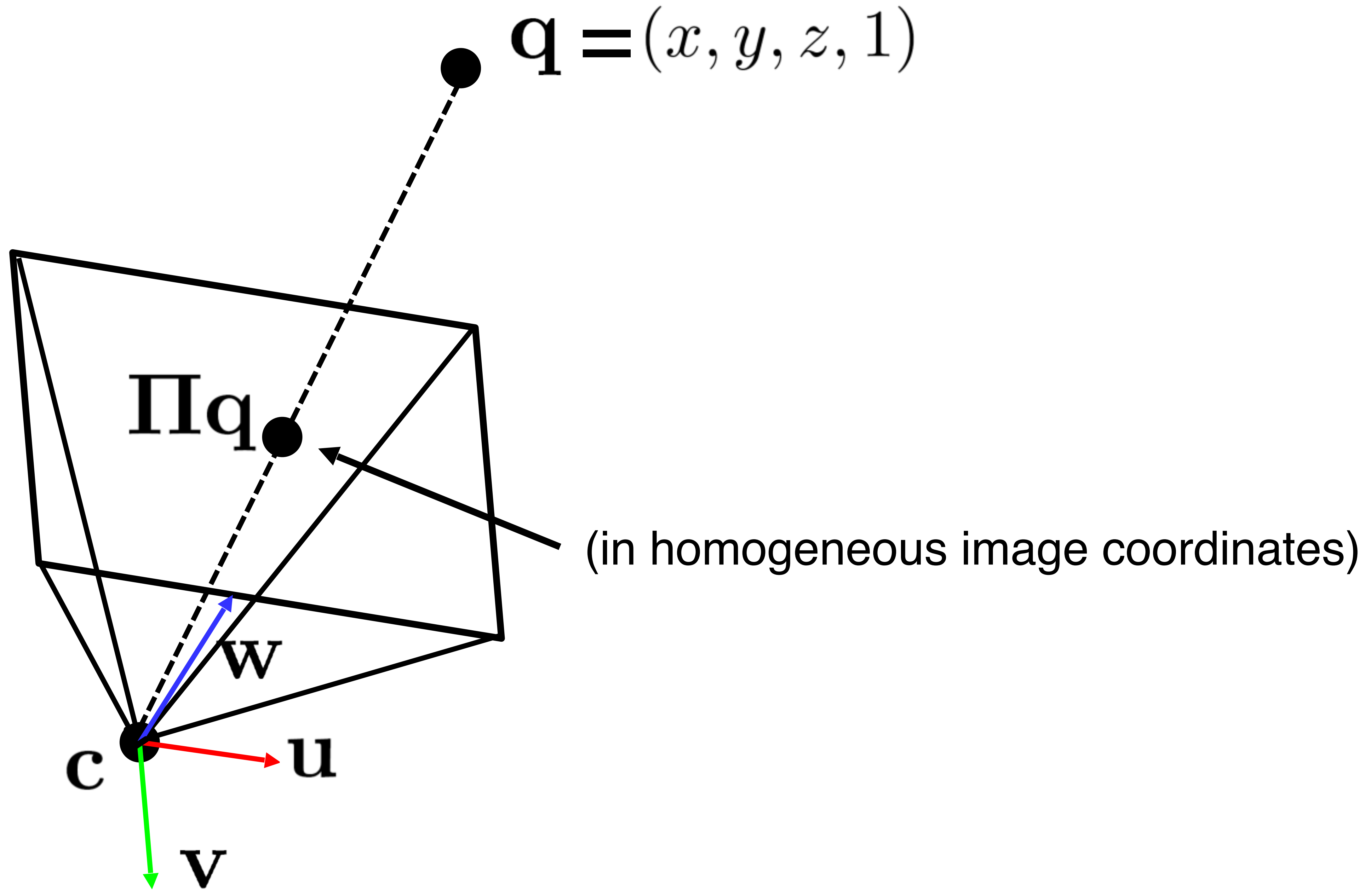
$$\mathbf{\Pi} = \begin{bmatrix} f & s & c_x \\ 0 & \alpha f & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{R}_{3\times 3} & \mathbf{0}_{3\times 1} \\ \mathbf{0}_{1\times 3} & 0 \end{bmatrix} \begin{bmatrix} \mathbf{I}_{3\times 3} & \mathbf{T}_{3\times 1} \\ \mathbf{0}_{1\times 3} & 0 \end{bmatrix}$$

intrinsics rotation translation

Source: N. Snavely

Projection matrix

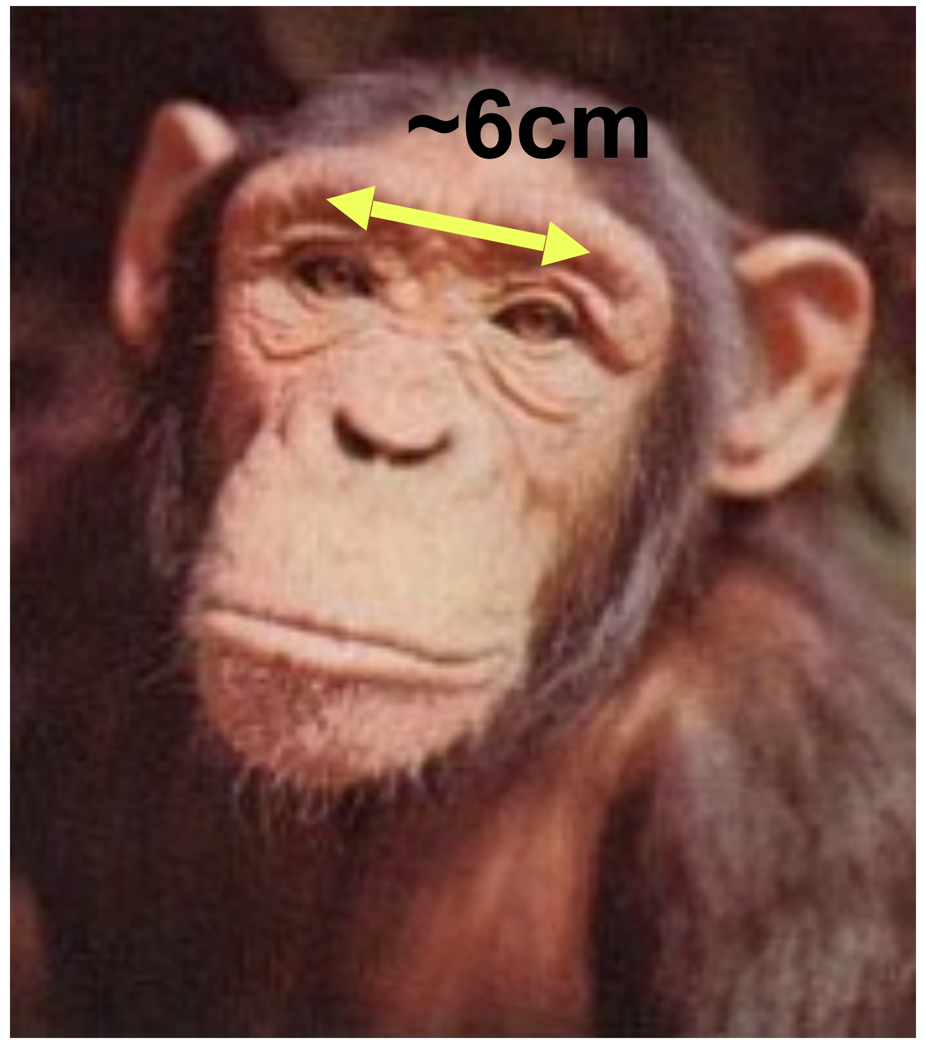


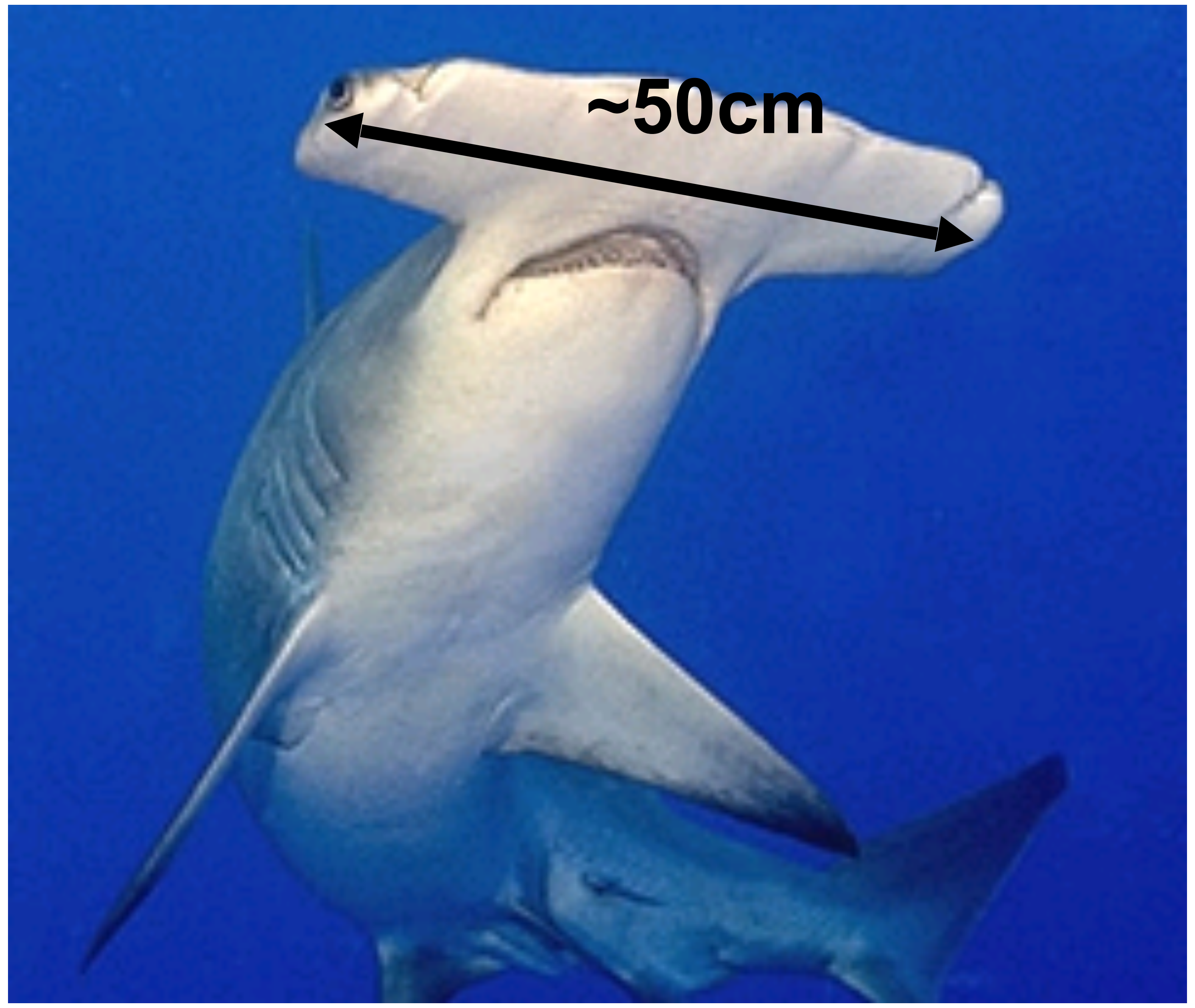


Source: N. Snavely

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

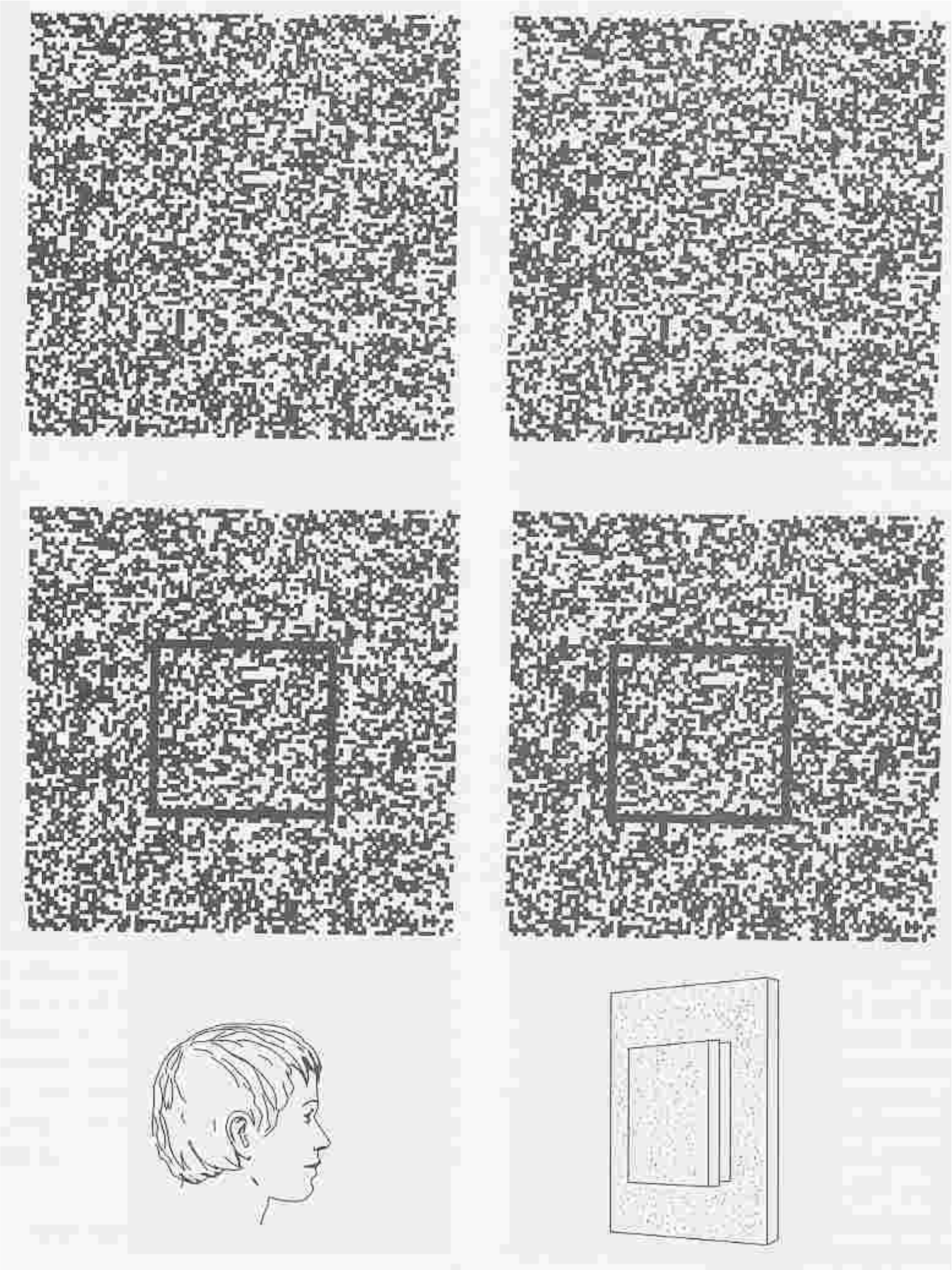


FIGURE 8.13

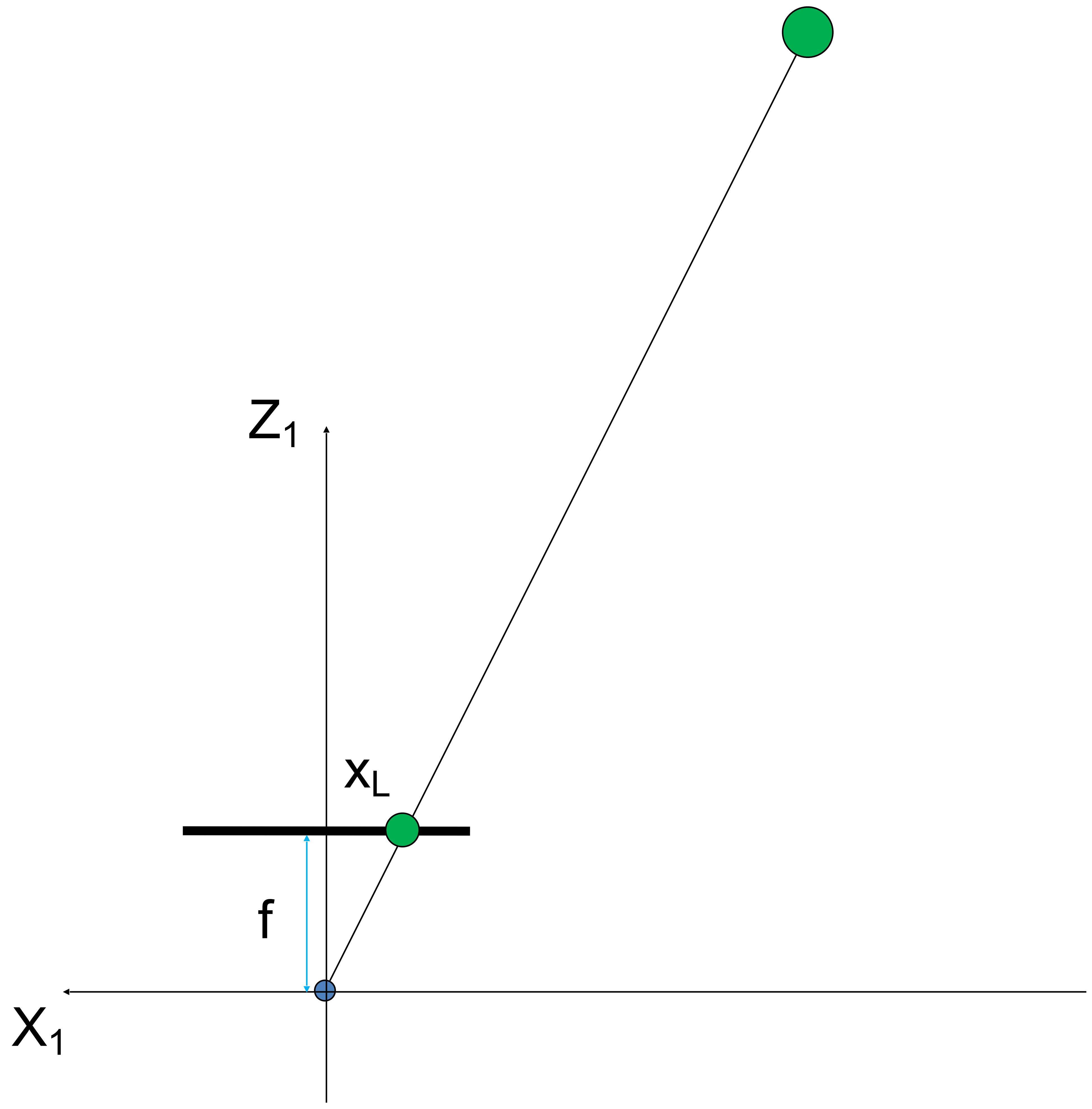
1	0	1	0	1	0	0	1	0	1
1	0	0	1	0	1	0	1	0	0
0	0	1	1	0	1	1	0	1	0
0	1	0	Y	A	A	B	B	0	1
1	1	1	X	B	A	B	A	0	1
0	0	1	X	A	A	B	A	1	0
1	1	1	Y	B	B	A	B	0	1
1	0	0	1	1	0	1	1	0	1
1	1	0	0	1	1	0	1	1	1
0	1	0	0	0	1	1	1	1	0

1	0	1	0	1	0	0	1	0	1
1	0	0	1	0	1	0	1	0	0
0	0	1	1	0	1	1	0	1	0
0	1	0	A	A	B	B	X	0	1
1	1	1	B	A	B	A	Y	0	1
0	0	1	A	A	B	A	Y	1	0
1	1	1	B	B	A	B	X	0	1
1	0	0	1	1	0	1	1	0	1
1	1	0	0	1	1	0	1	1	1
0	1	0	0	0	1	1	1	1	0

Julesz, 1971

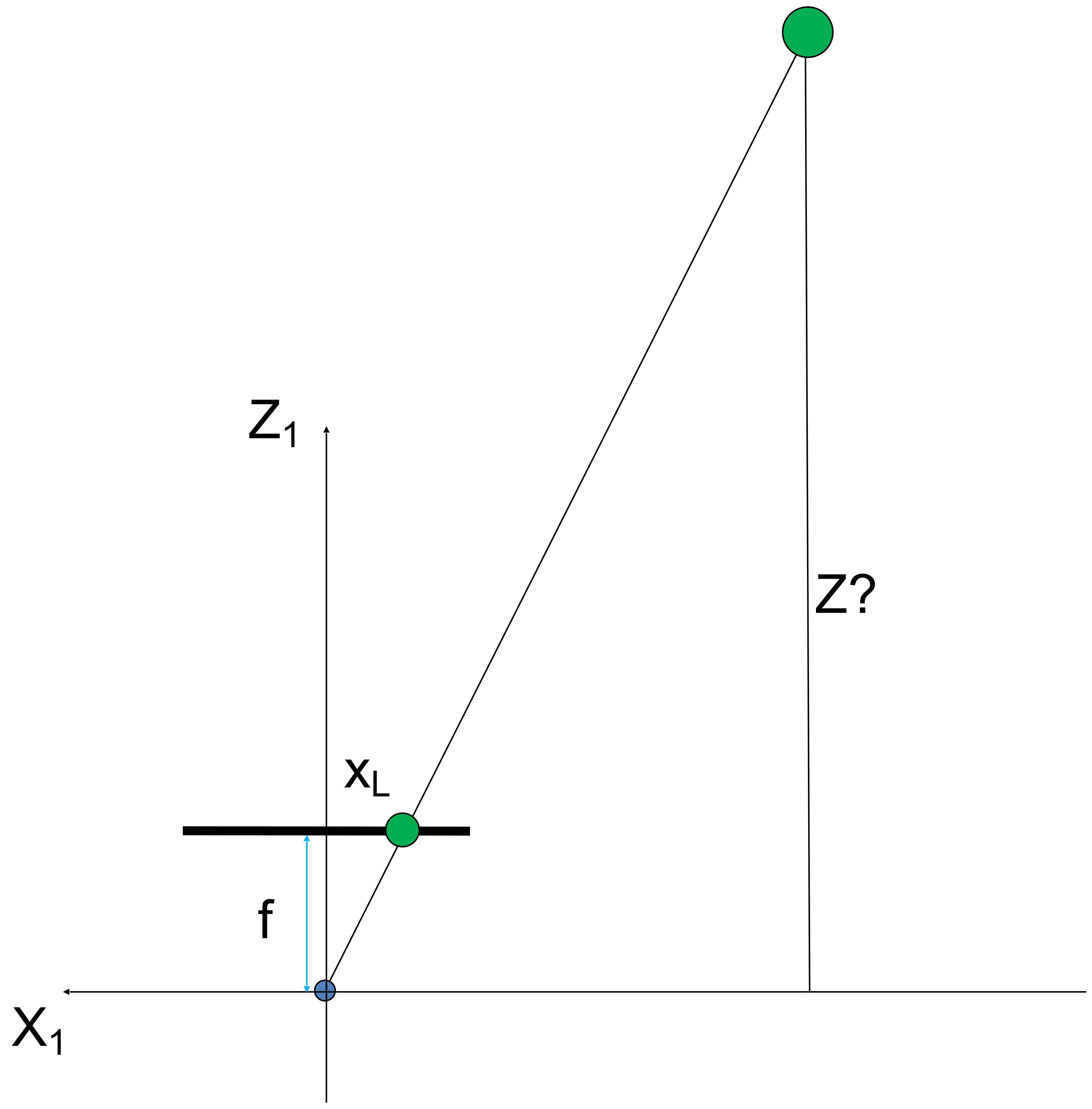
Source: Torralba, Isola, Freeman

Geometry for a simple stereo system



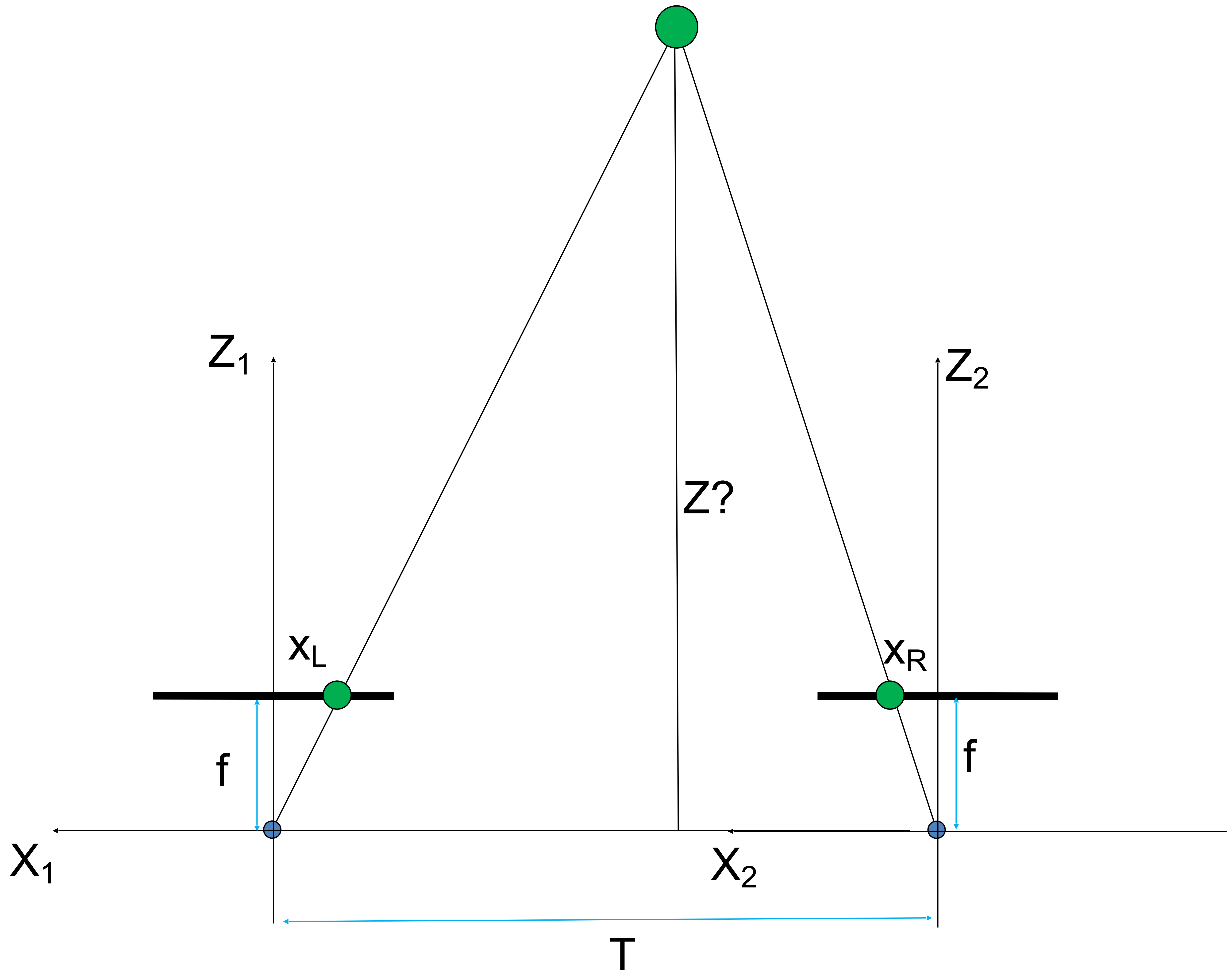
Source: Torralba, Isola, Freeman

Geometry for a simple stereo system



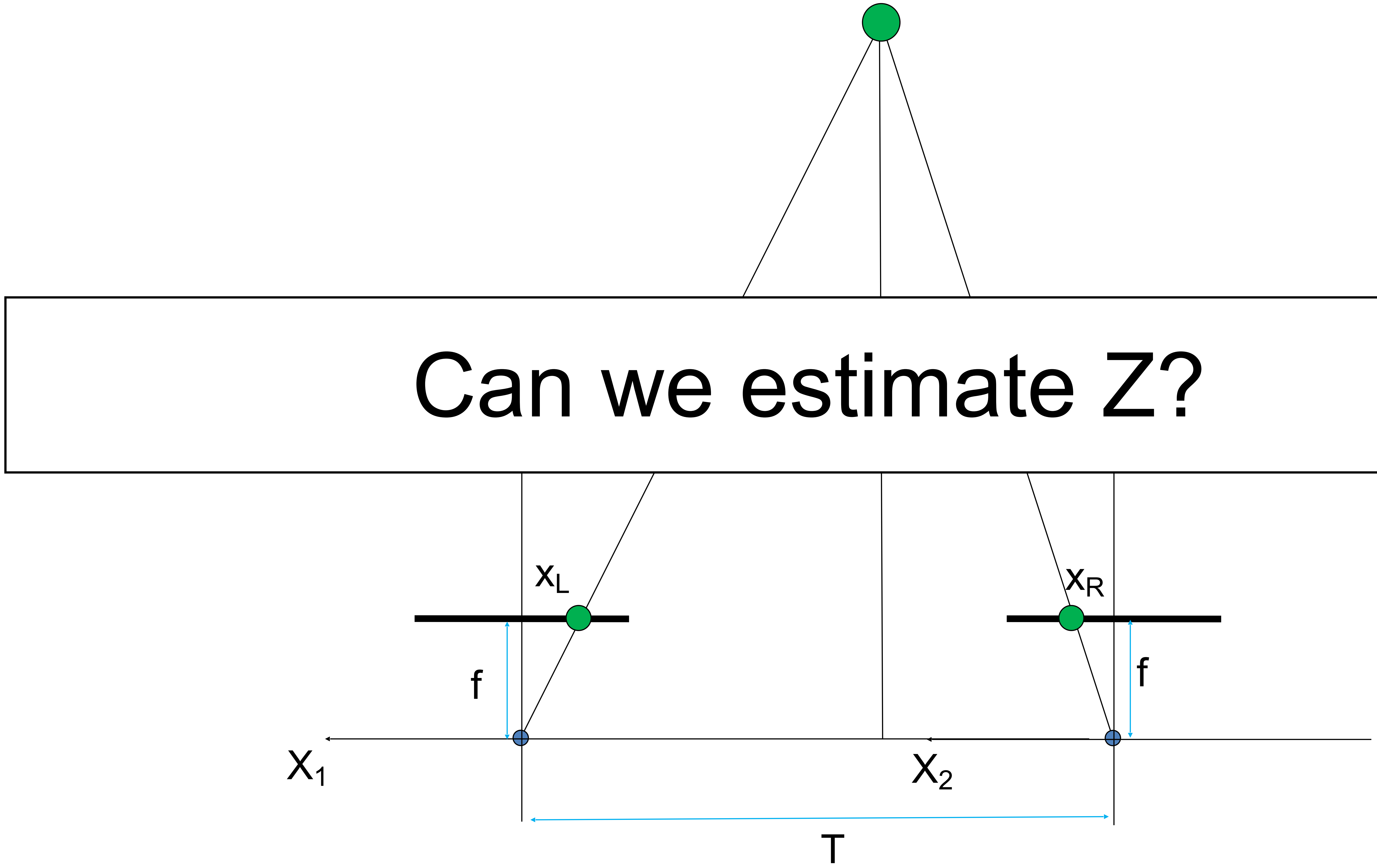
Source: Torralba, Isola, Freeman

Geometry for a simple stereo system



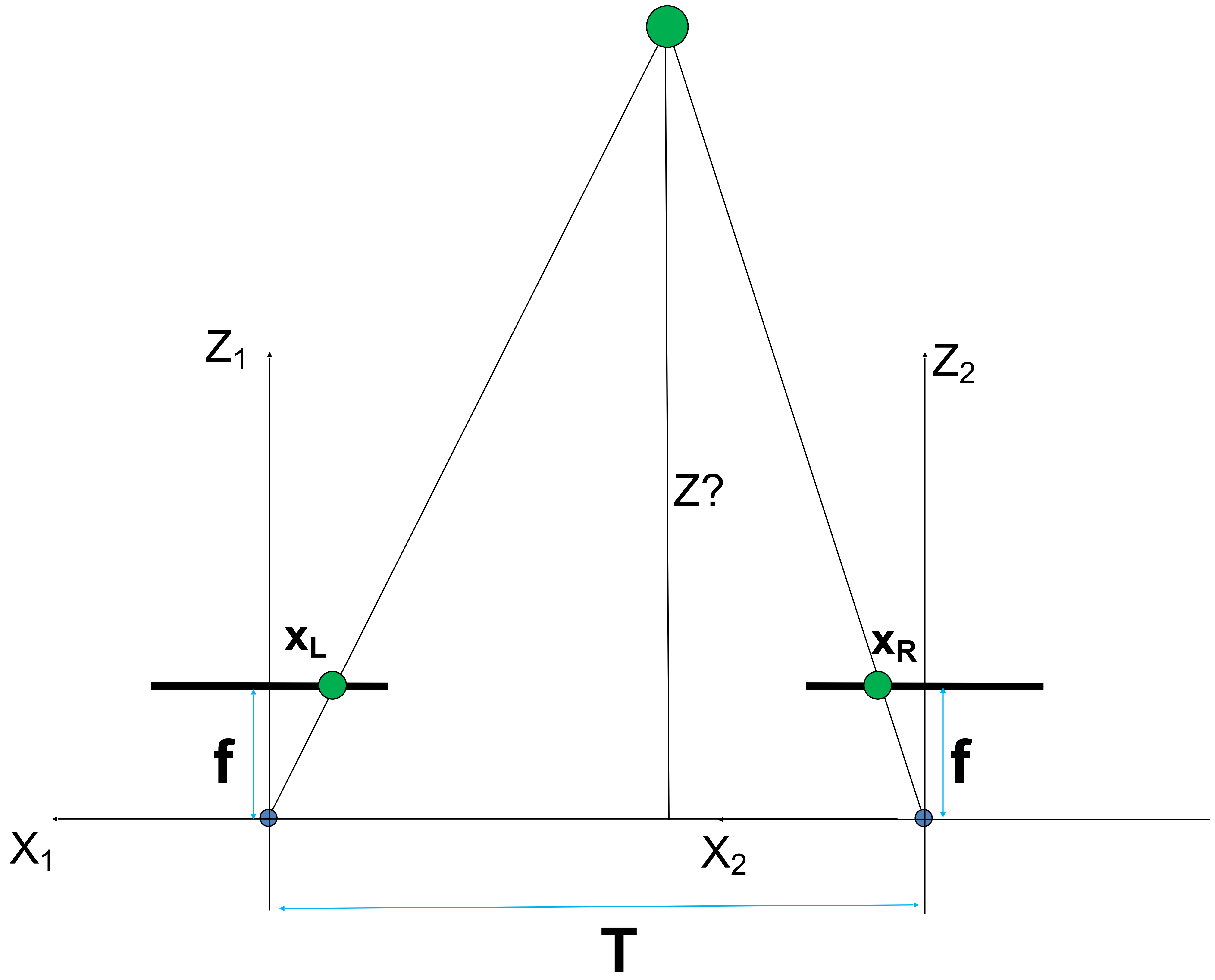
Source: Torralba, Isola, Freeman

Geometry for a simple stereo system



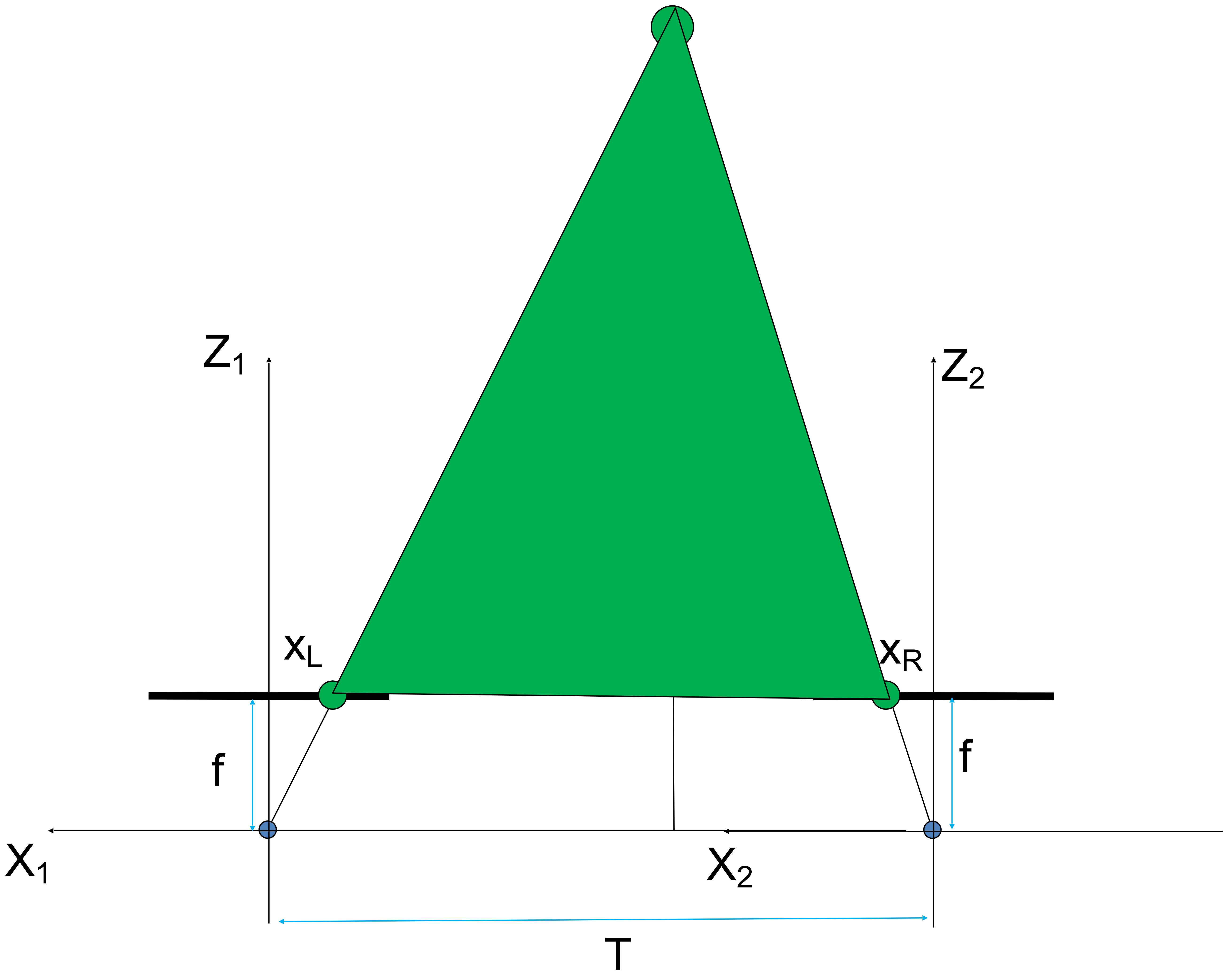
Source: Torralba, Isola, Freeman

Geometry for a simple stereo system



Source: Torralba, Isola, Freeman

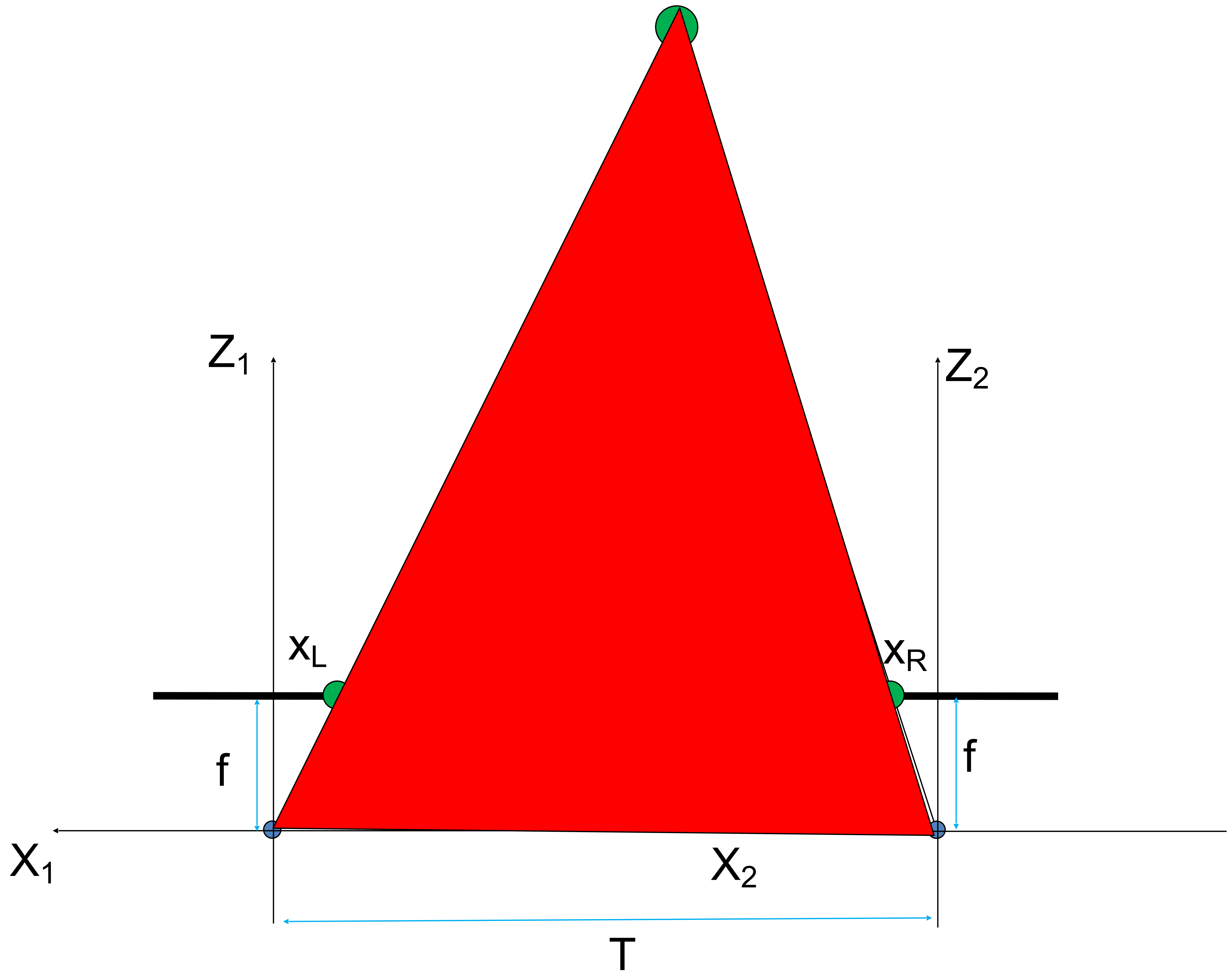
Geometry for a simple stereo system



Similar triangles

Source: Torralba, Isola, Freeman

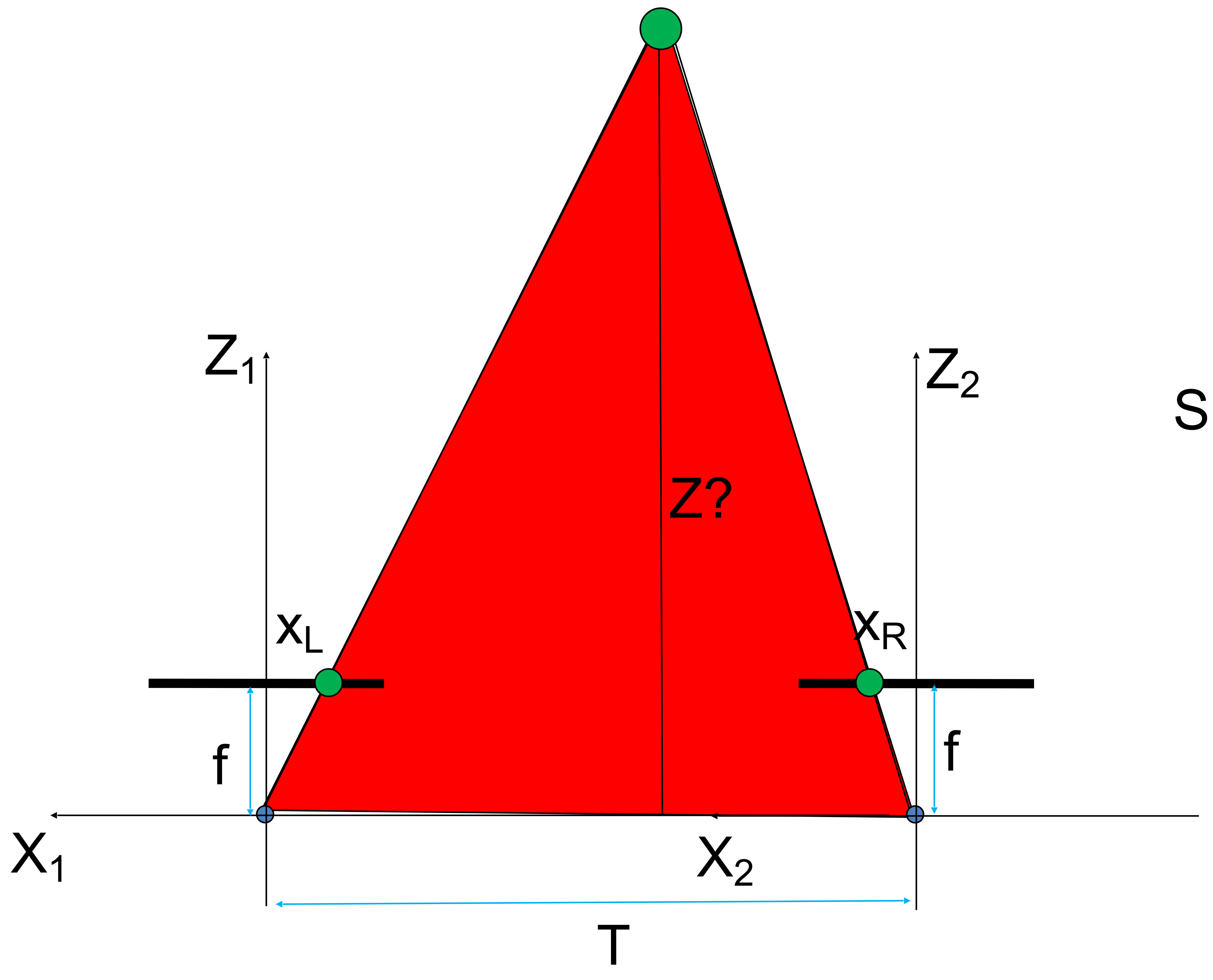
Geometry for a simple stereo system



Similar triangles

Source: Torralba, Isola, Freeman

Geometry for a simple stereo system

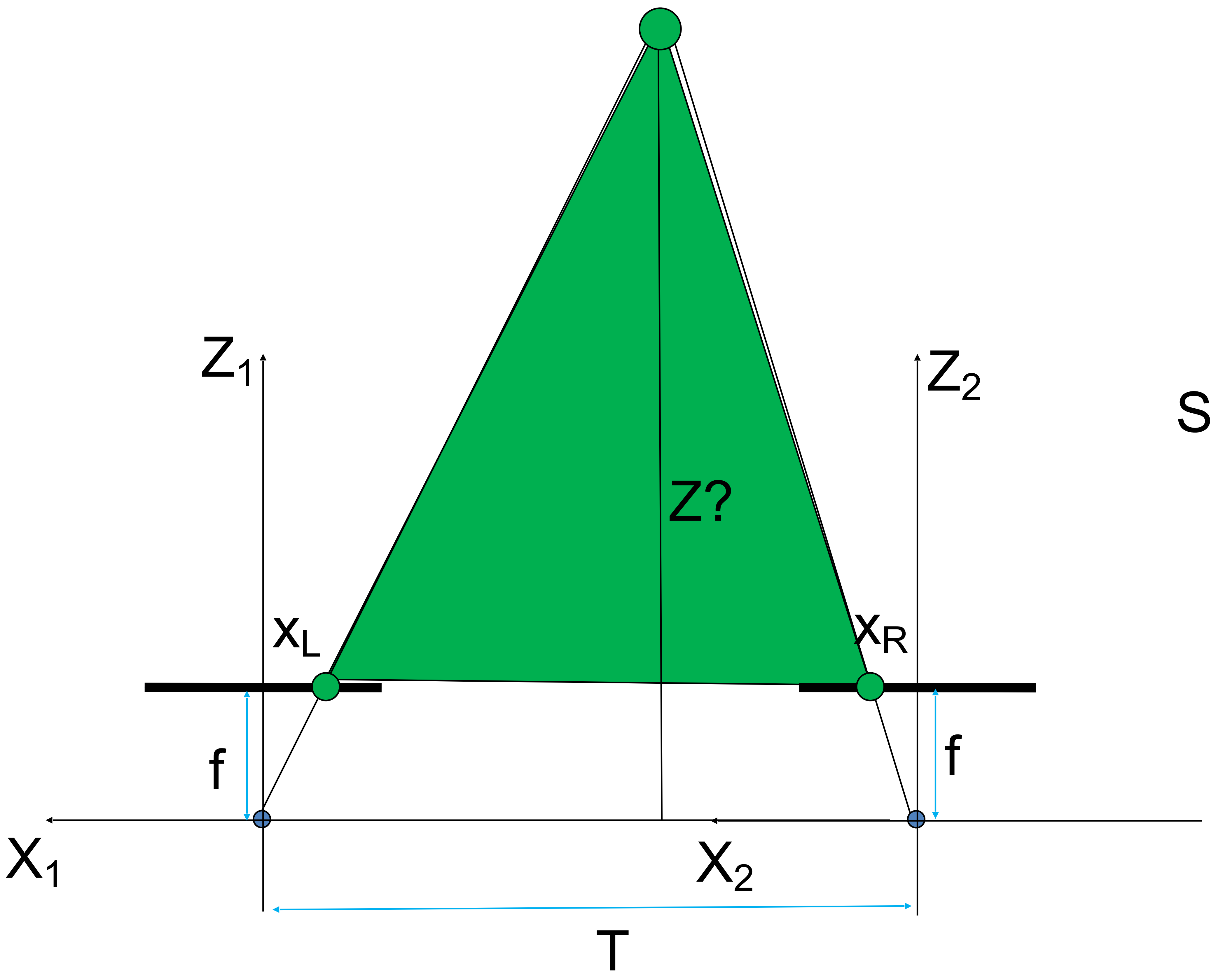


Similar triangles:

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

Source: Torralba, Isola, Freeman

Geometry for a simple stereo system

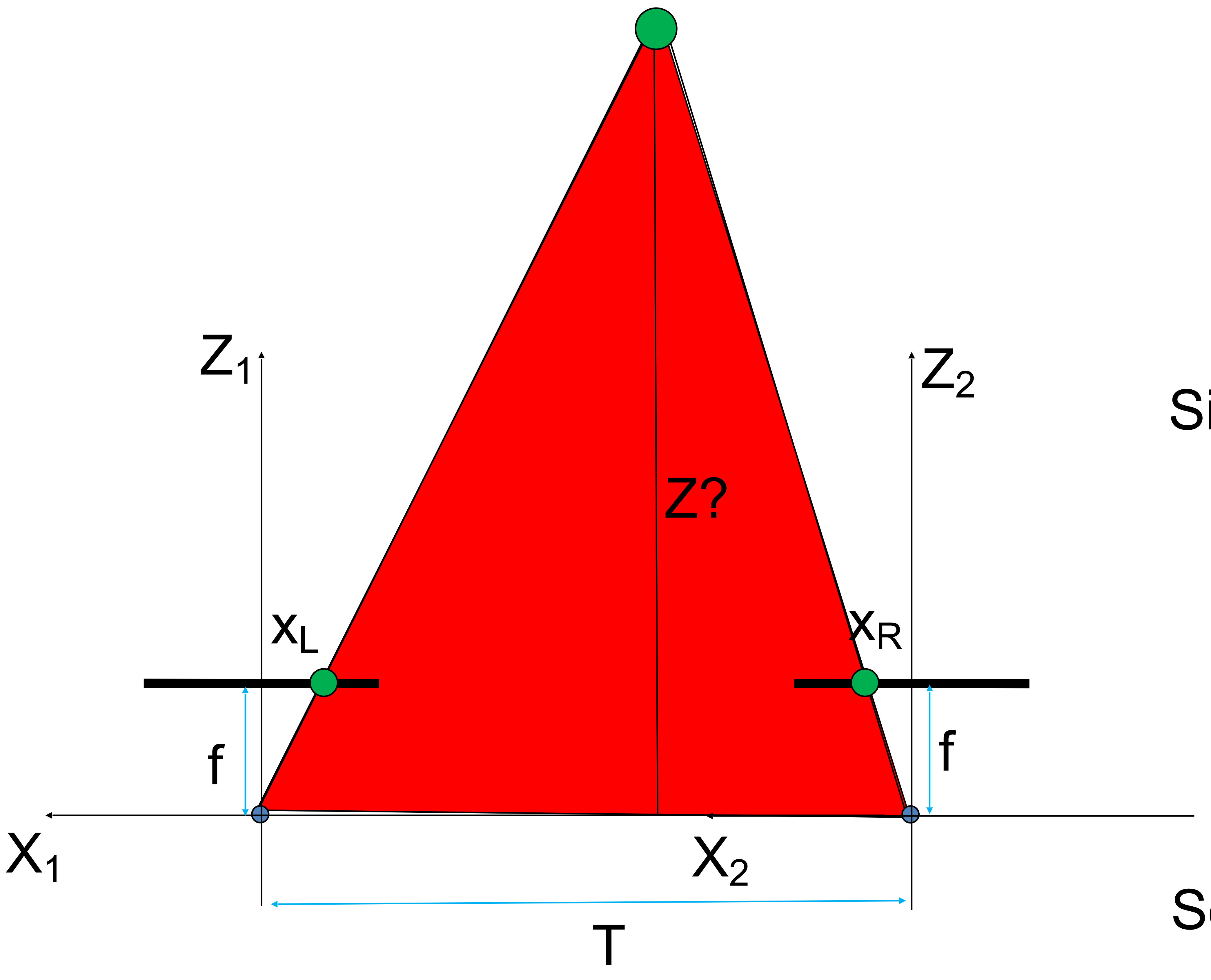


Similar triangles:

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

Source: Torralba, Isola, Freeman

Geometry for a simple stereo system



Similar triangles:

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

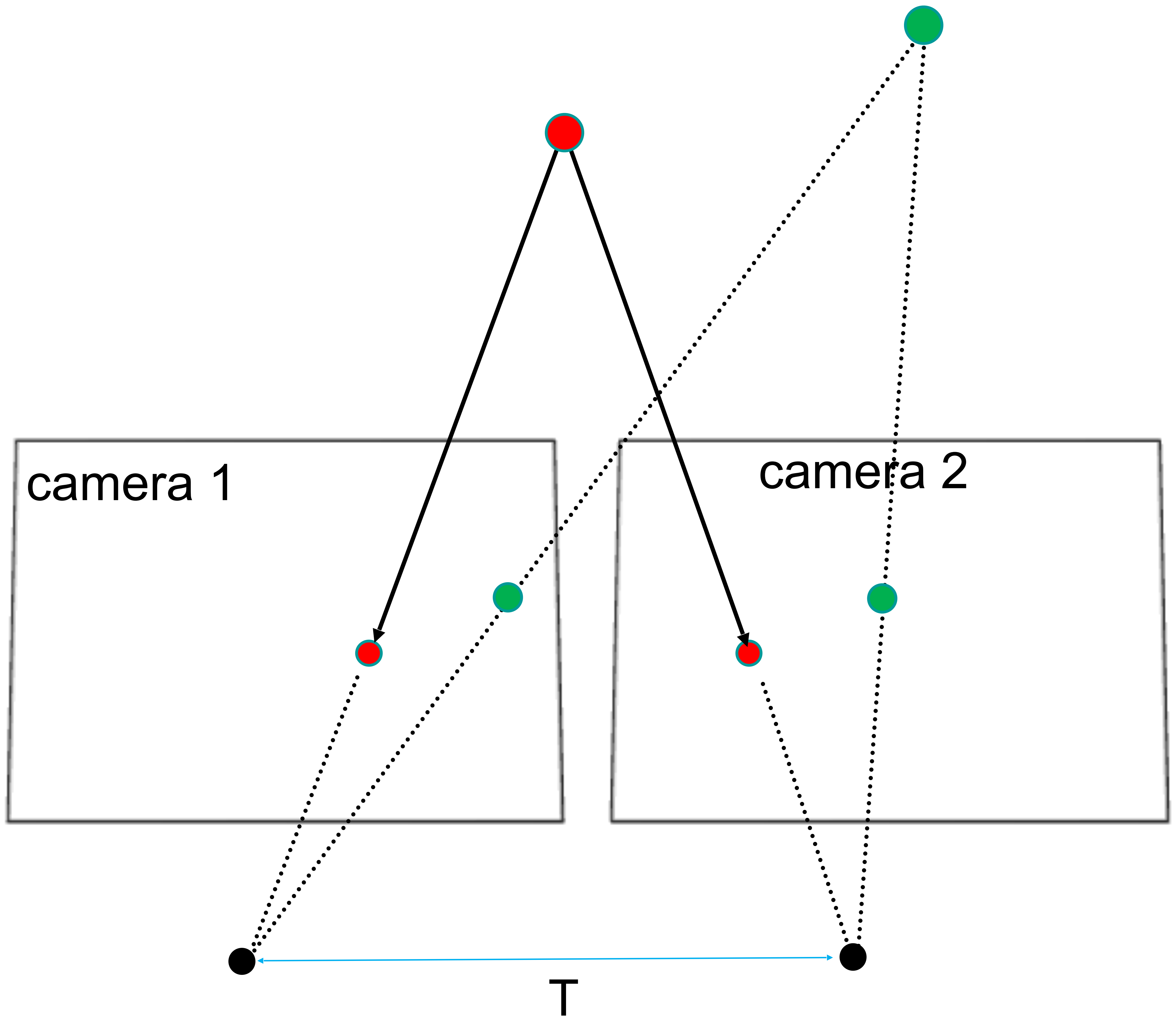
Solving for Z:

Disparity

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

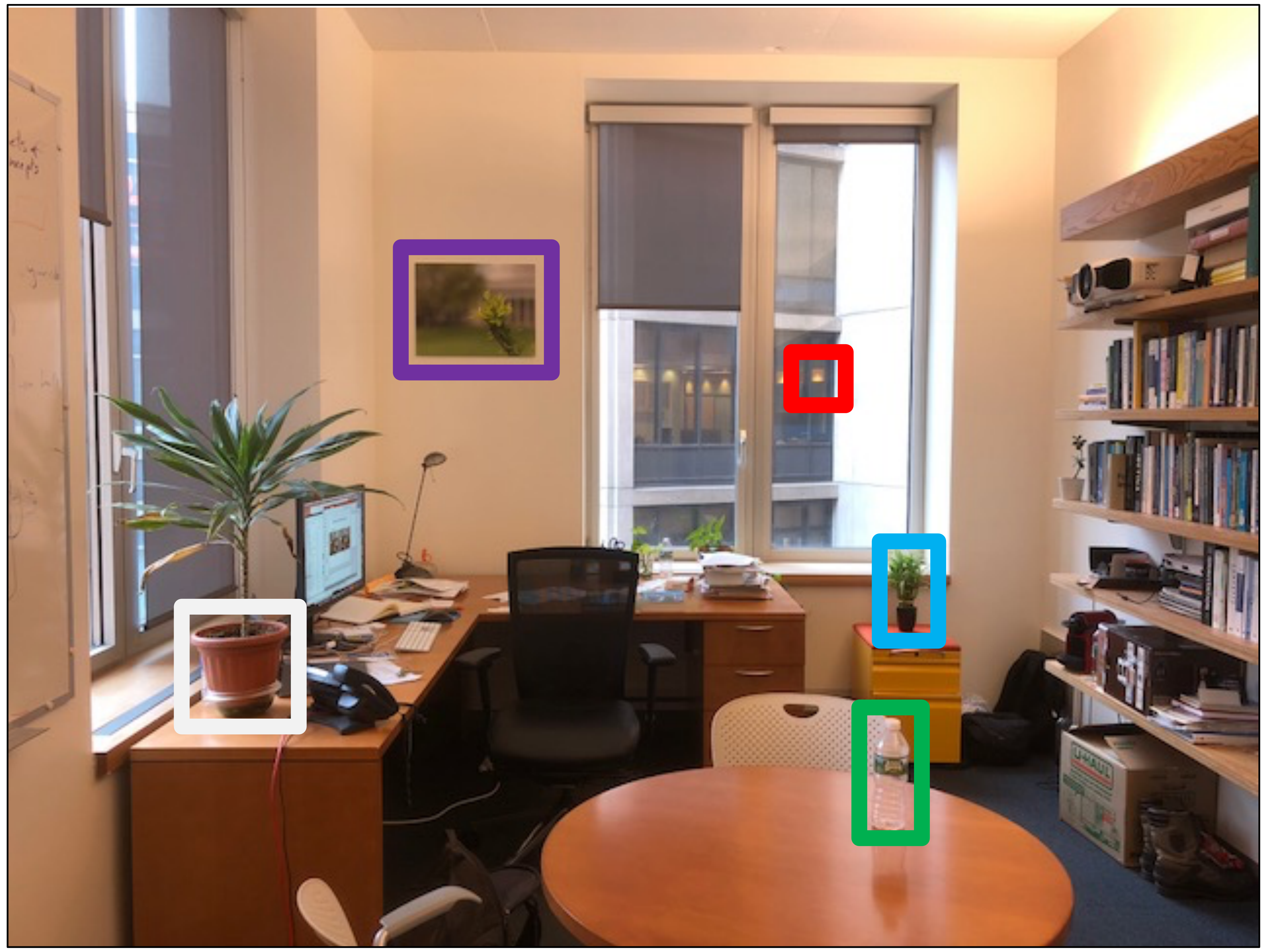
Source: Torralba, Isola, Freeman

In 3D



Source: Torralba, Isola, Freeman

Disparity map



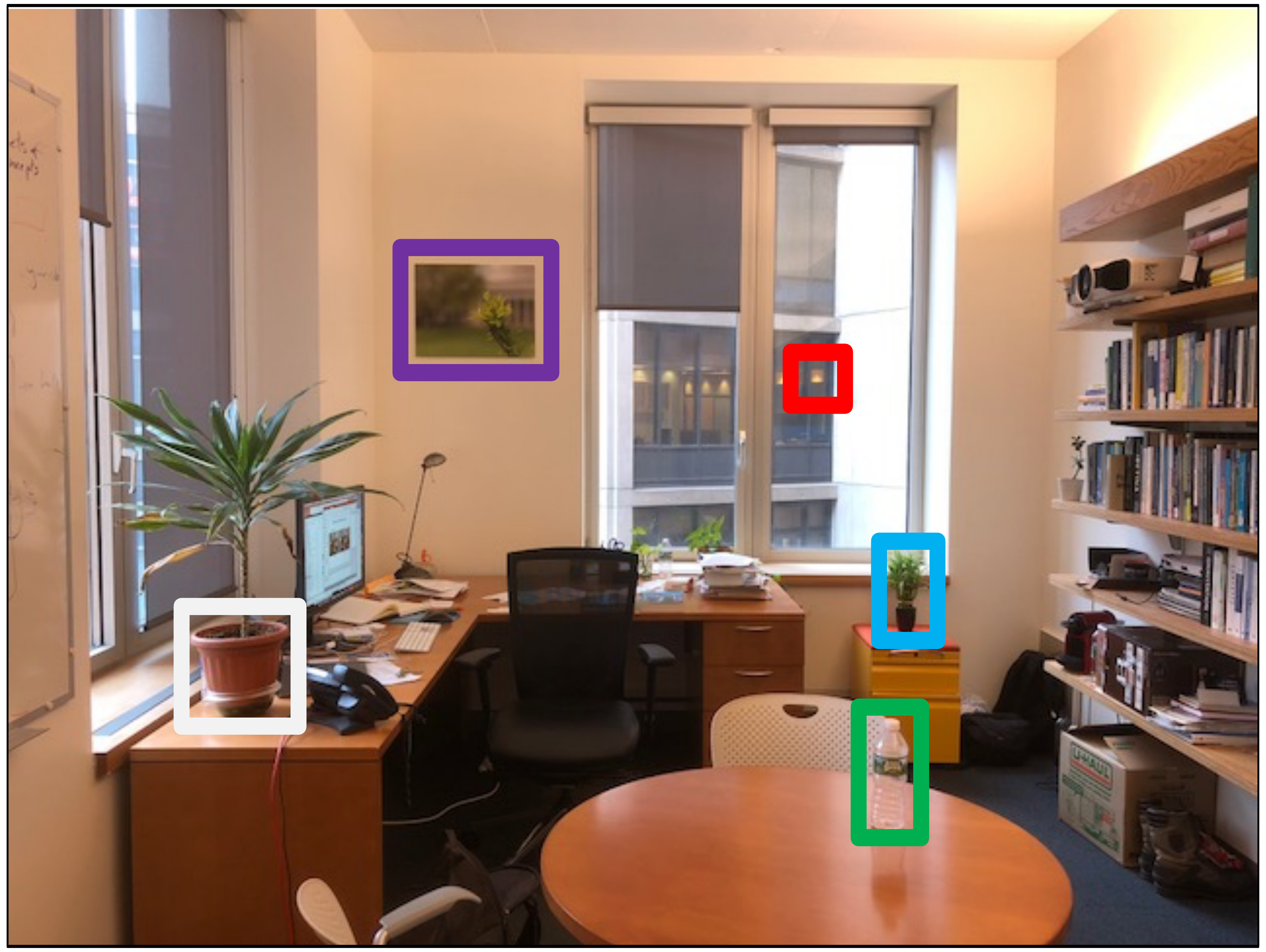
Left image

Right image

Second picture is ~1m to the right

Source: Torralba, Isola, Freeman

Disparity map

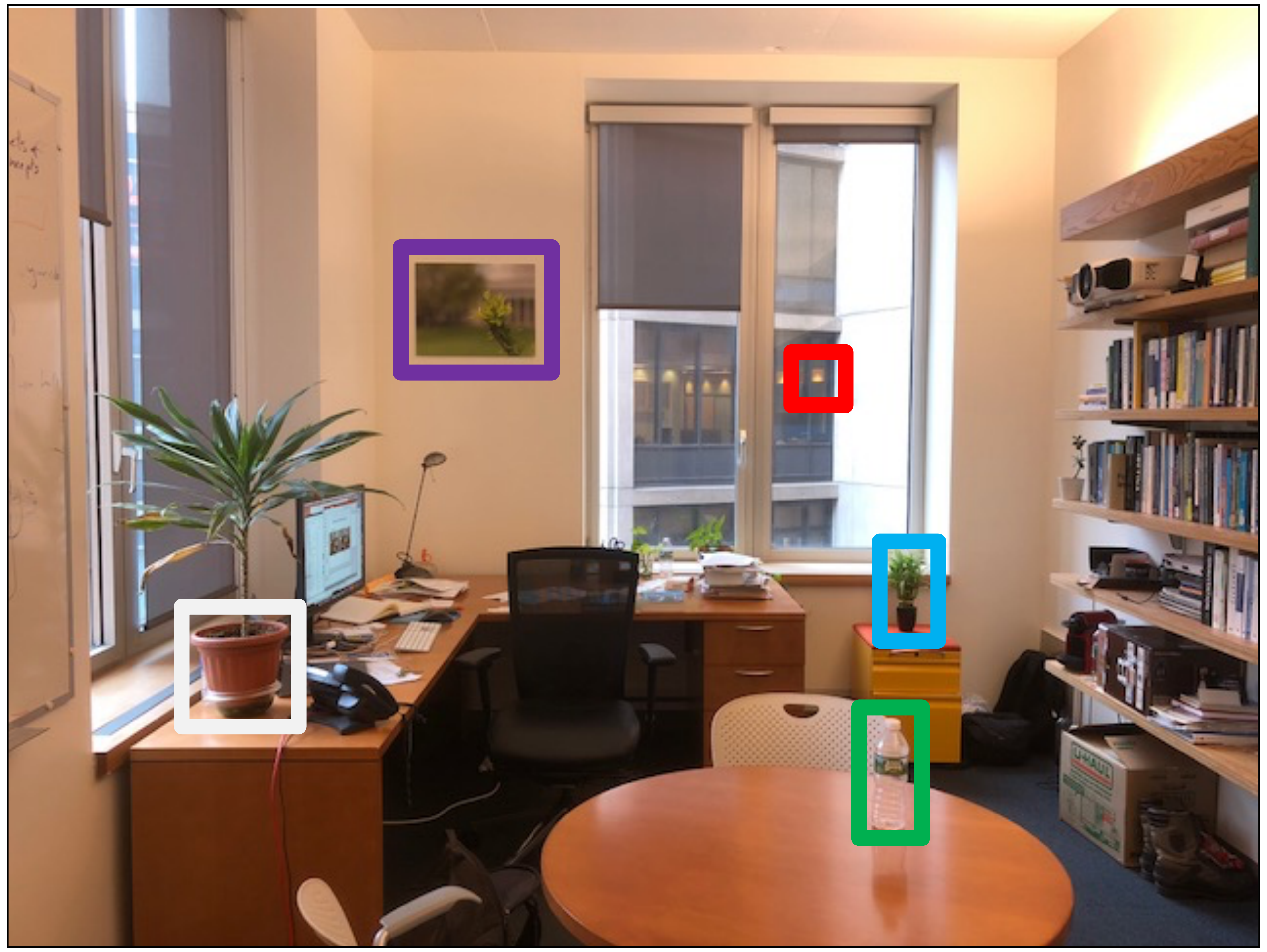


Left image

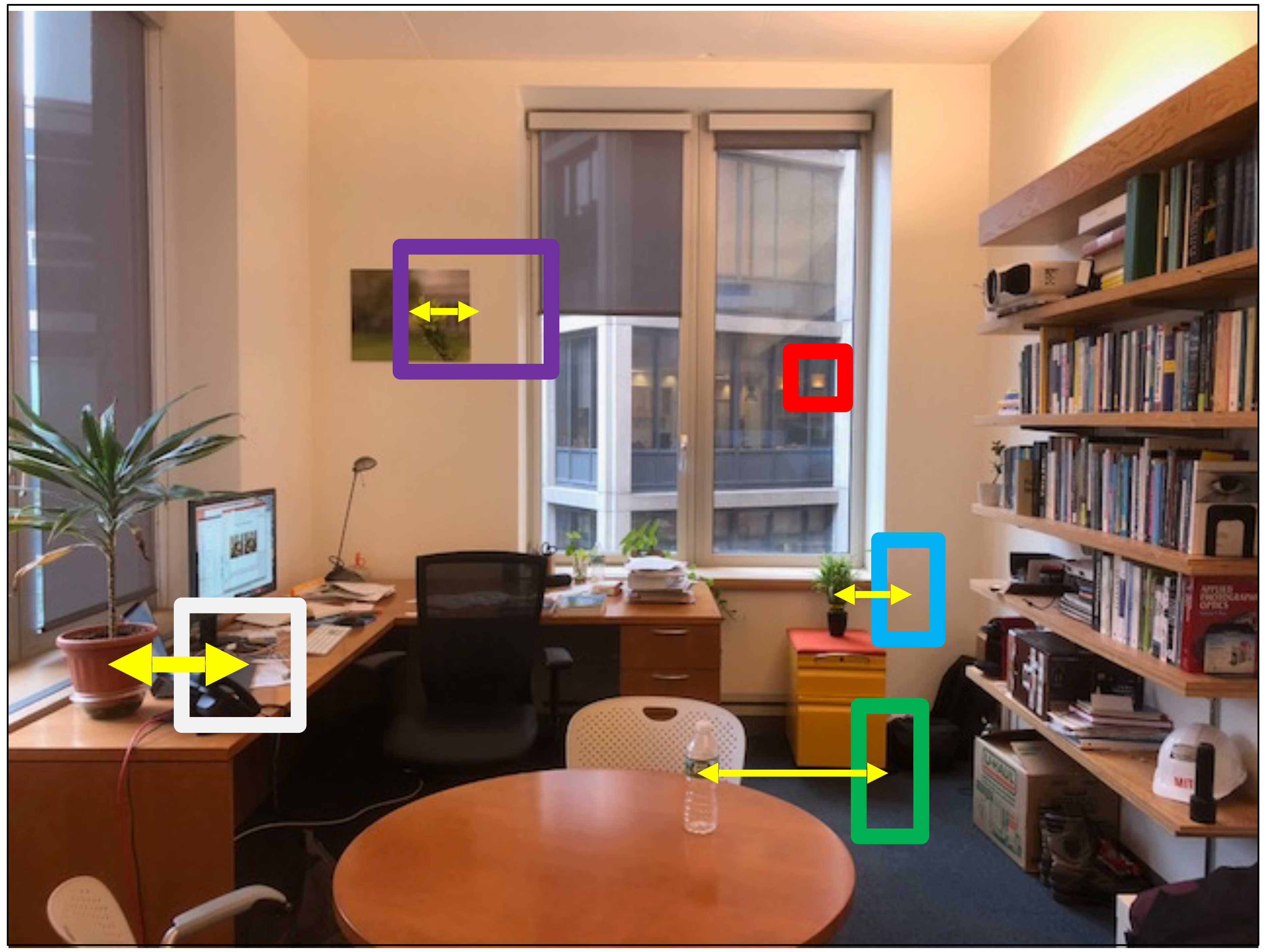
Right image

Source: Torralba, Isola, Freeman

Disparity map



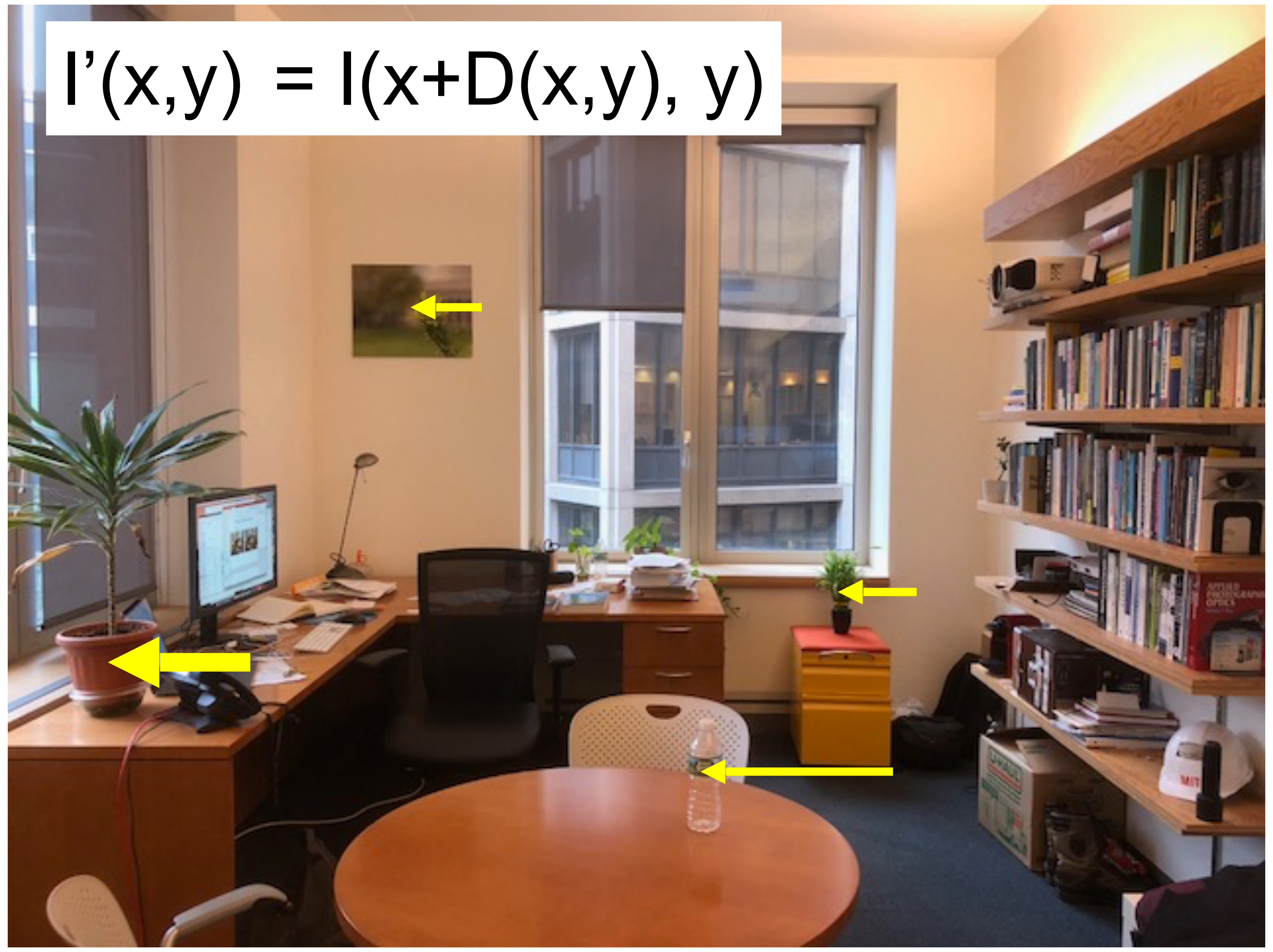
Left image



Right image

Source: Torralba, Isola, Freeman

Disparity map

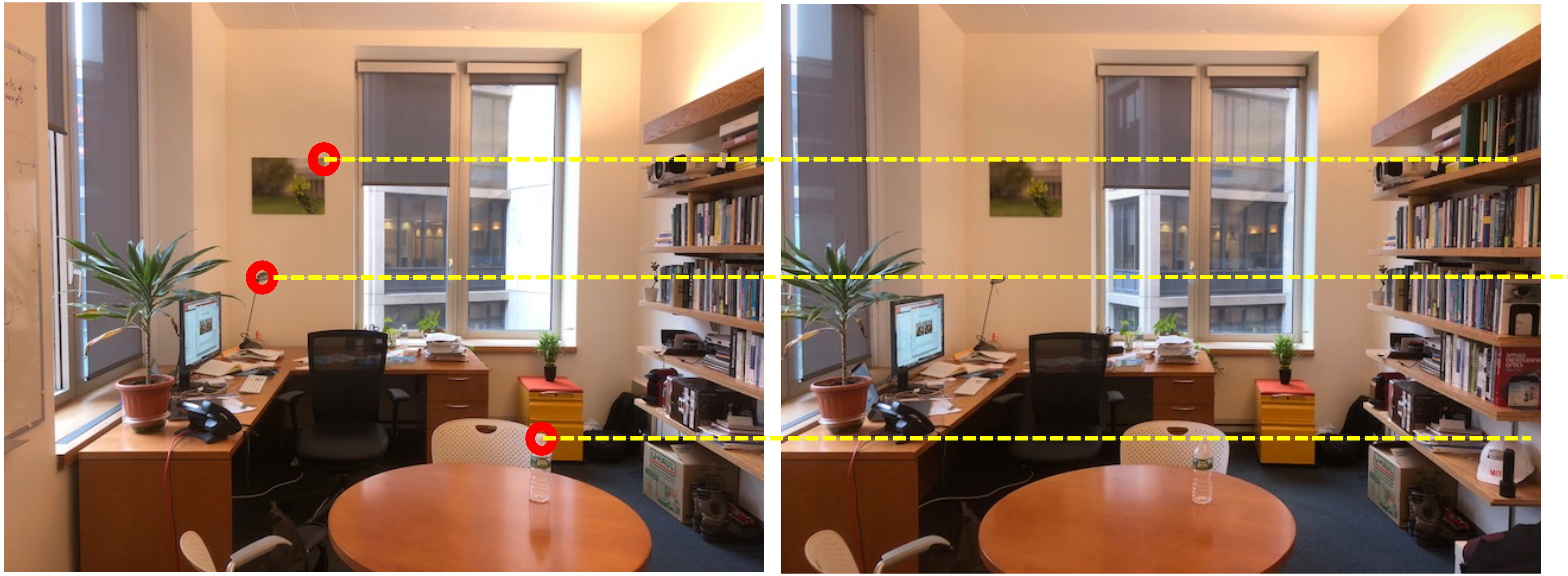


D(x,y)

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

Source: Torralba, Isola, Freeman

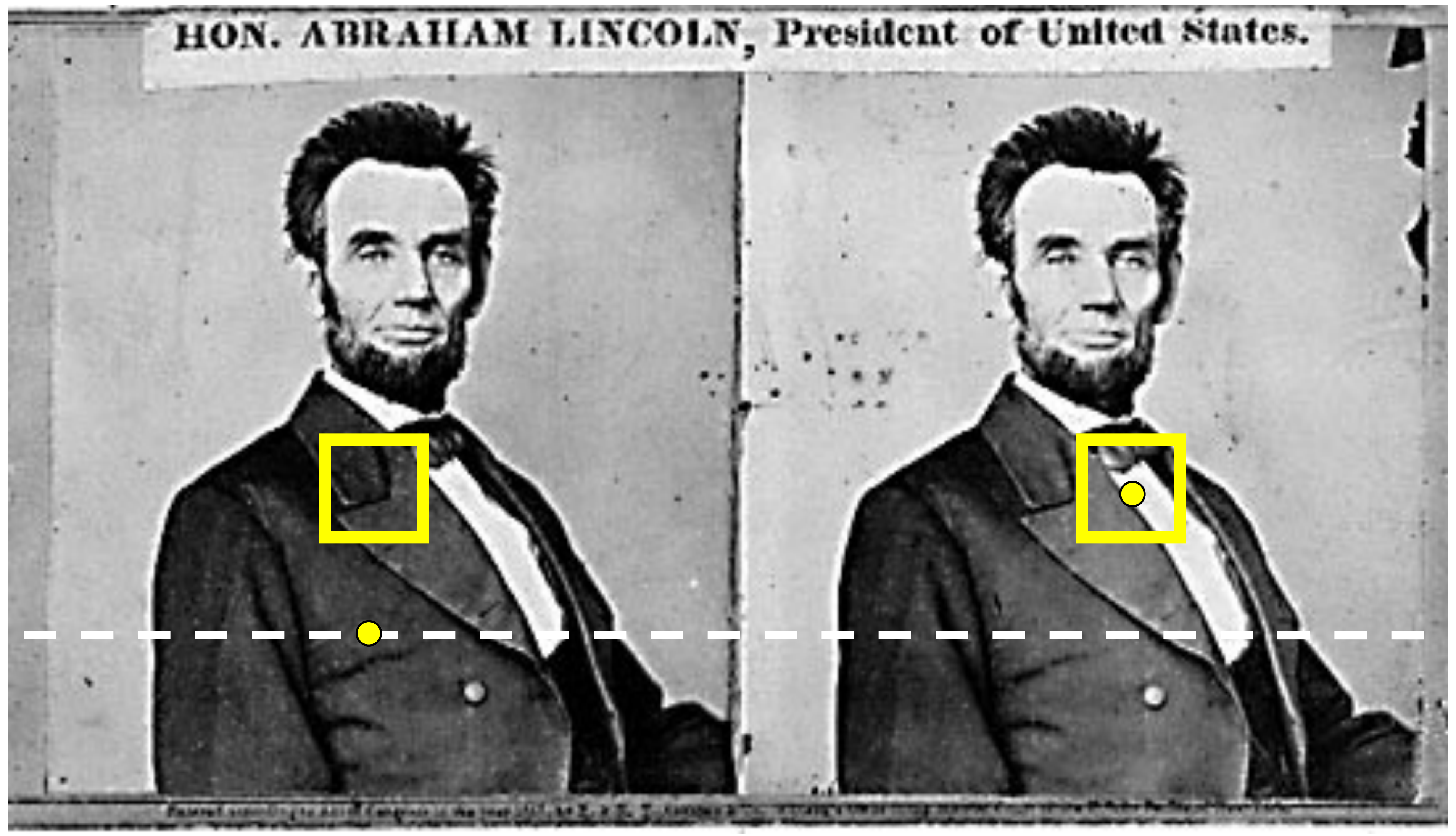
Finding correspondences



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

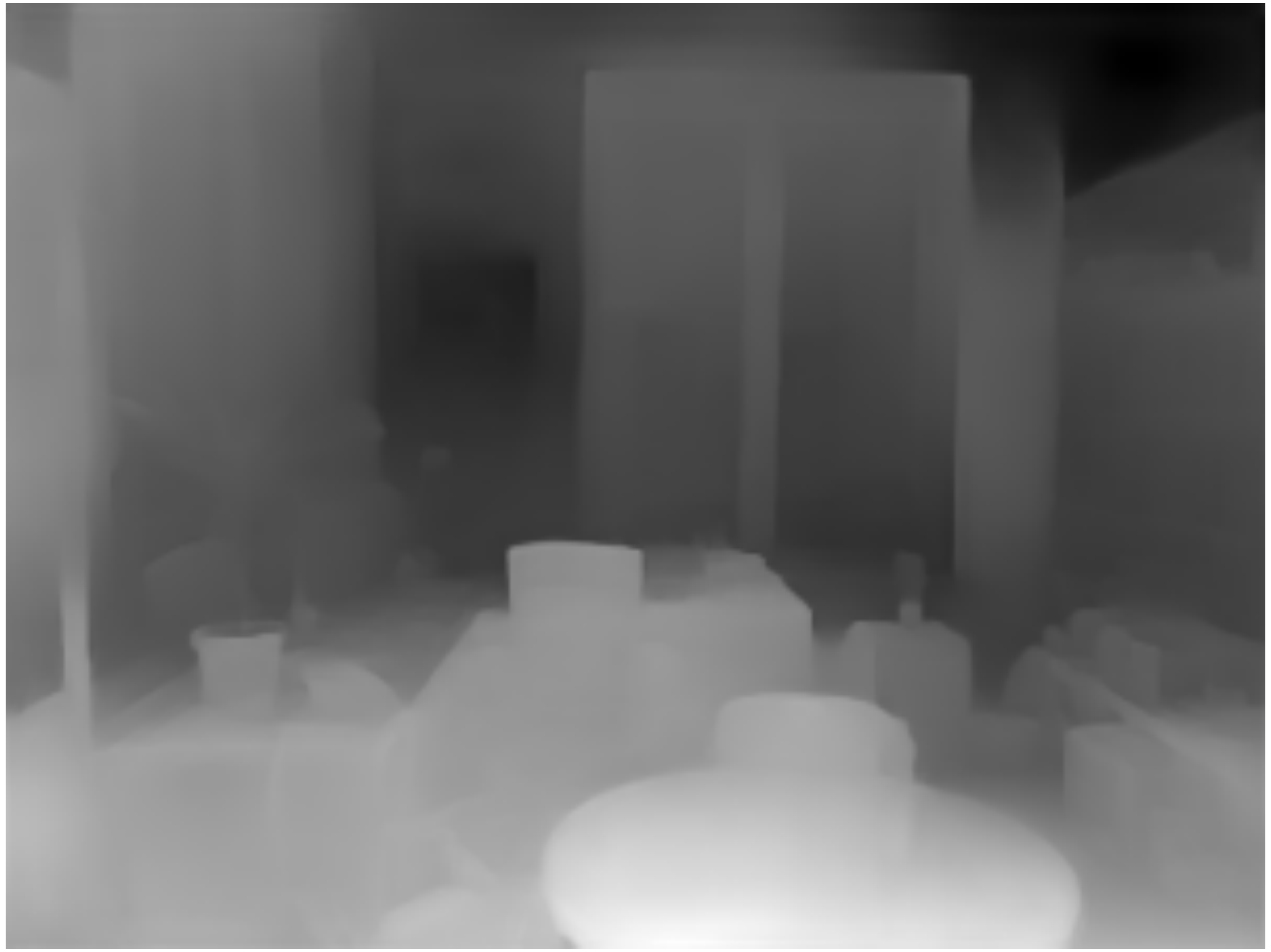
Source: Torralba, Isola, Freeman

Computing disparity

Semi-global matching [Hirschmüller 2008]

Source: Torralba, Isola, Freeman

Can also learn depth from a single image



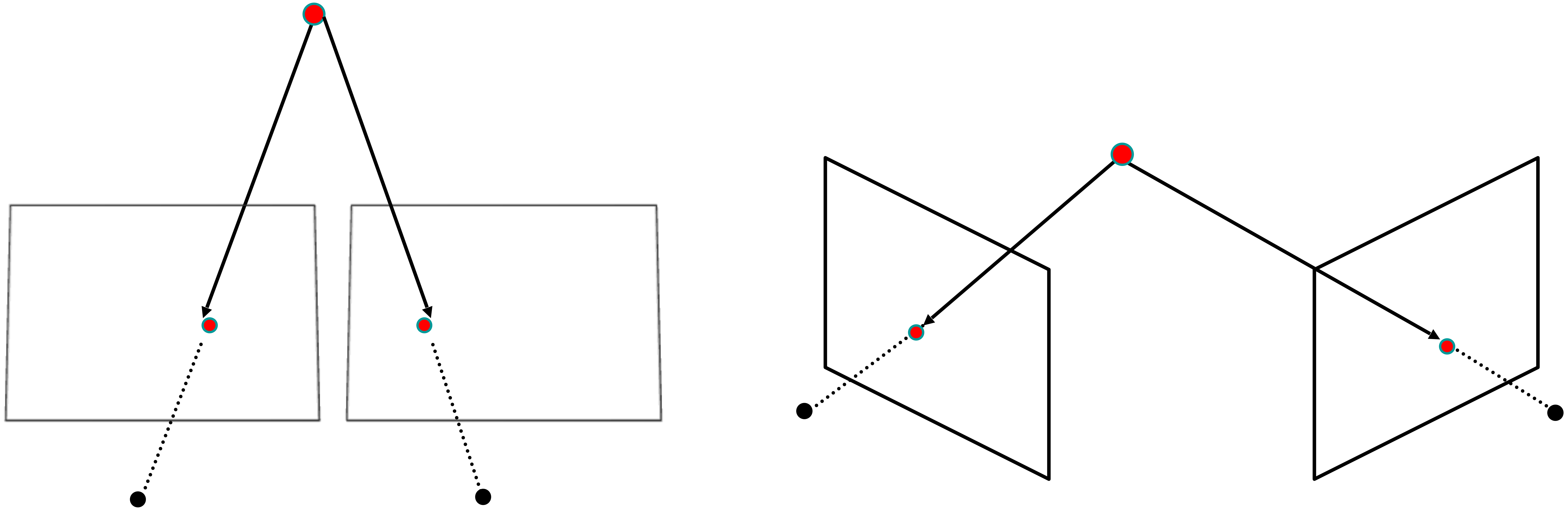
MegaDepth: Learning Single-View Depth Prediction from Internet Photos

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

Source: Torralba, Isola, Freeman

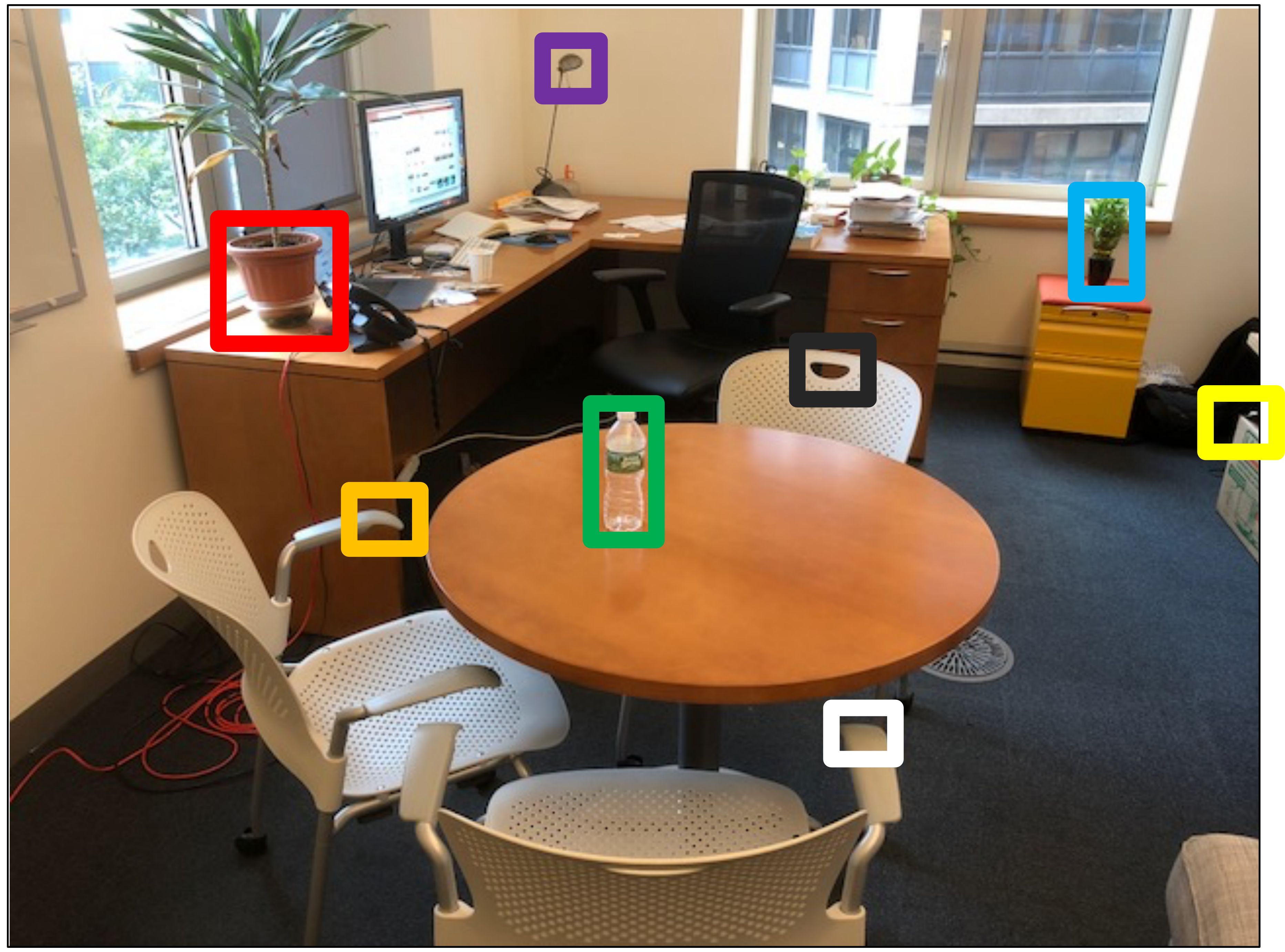
General case

- The two cameras need not have parallel optical axes.



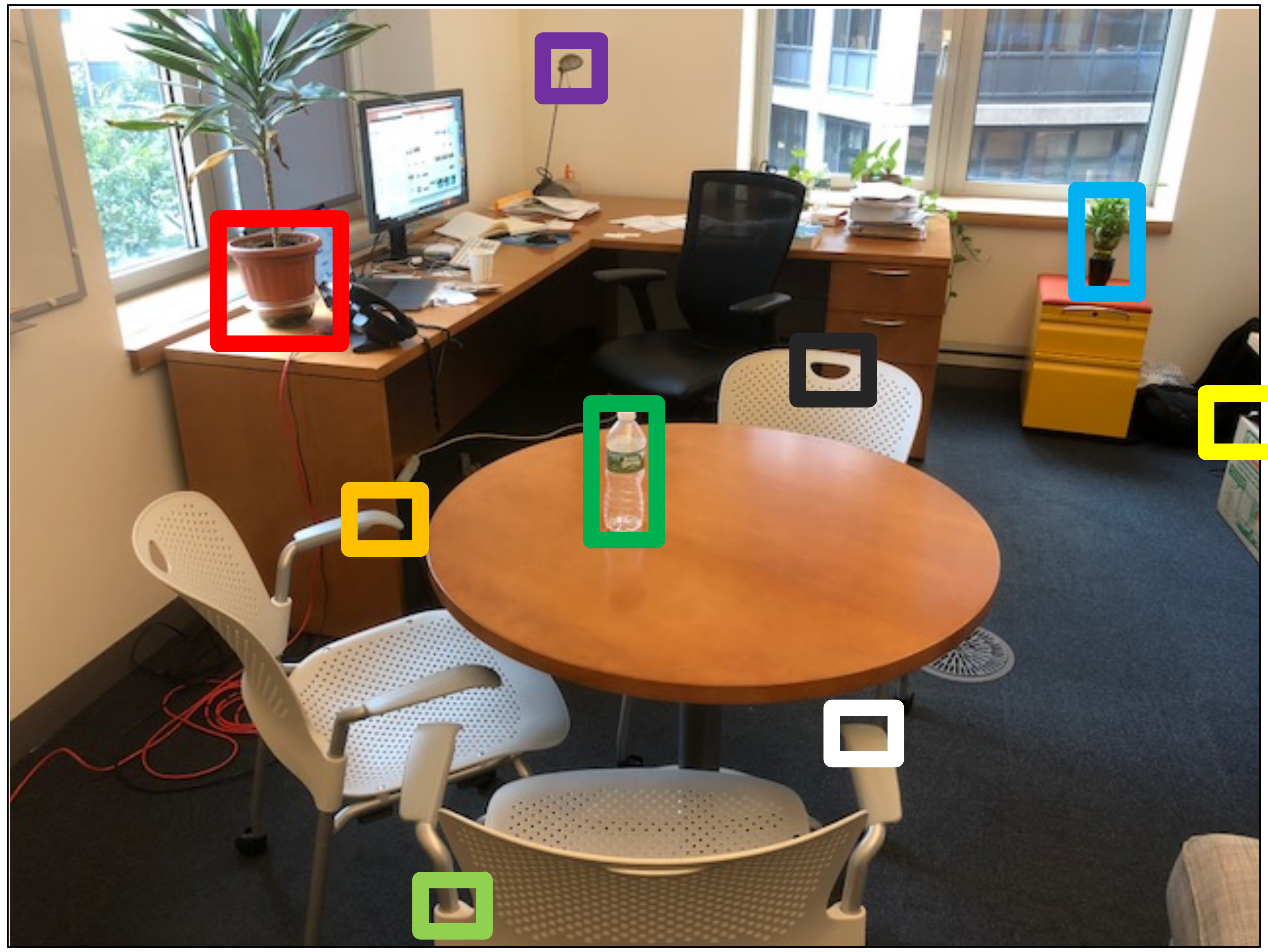
Source: Torralba, Isola, Freeman

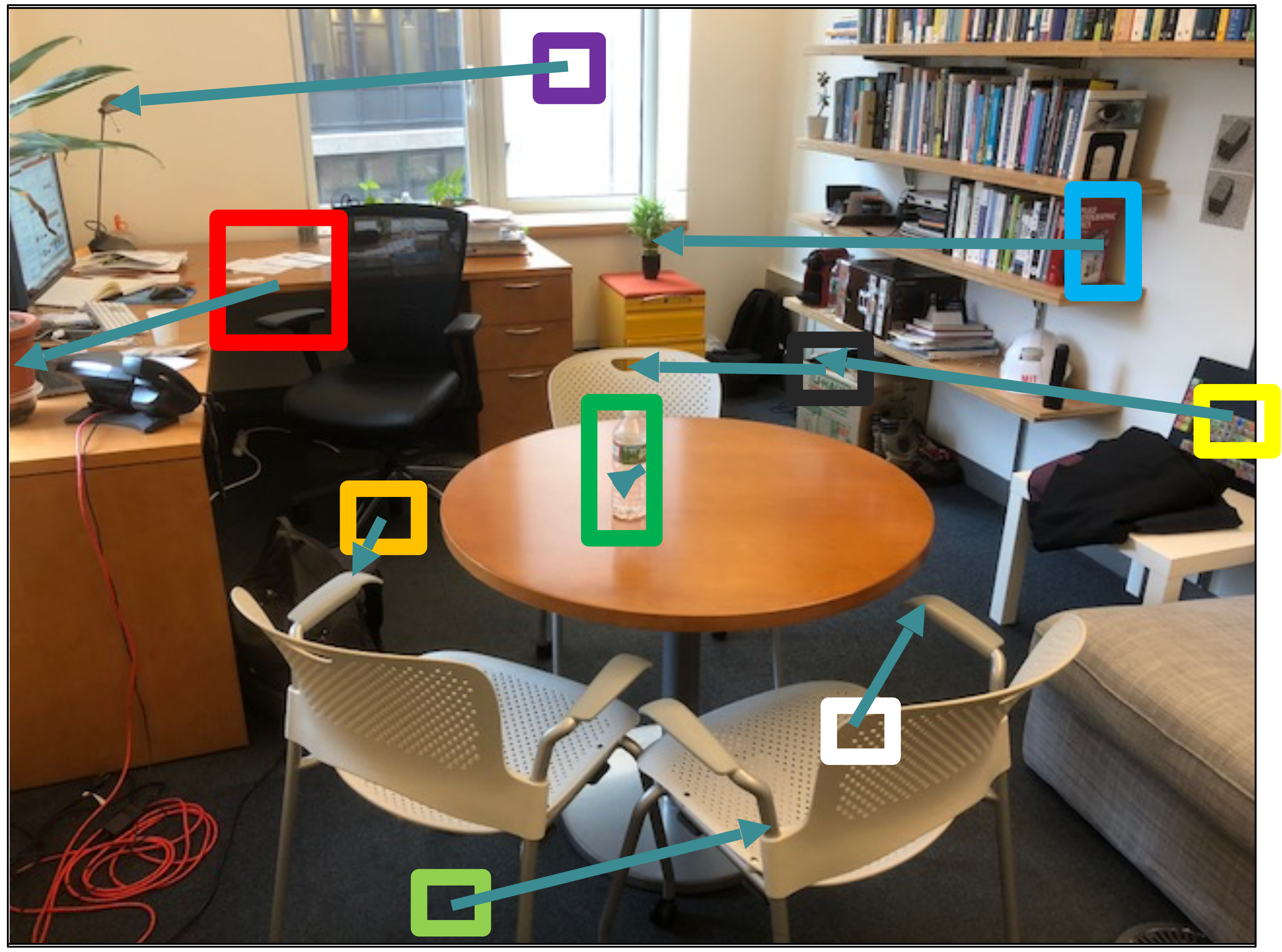
Source: Torralba, Isola, Freeman

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

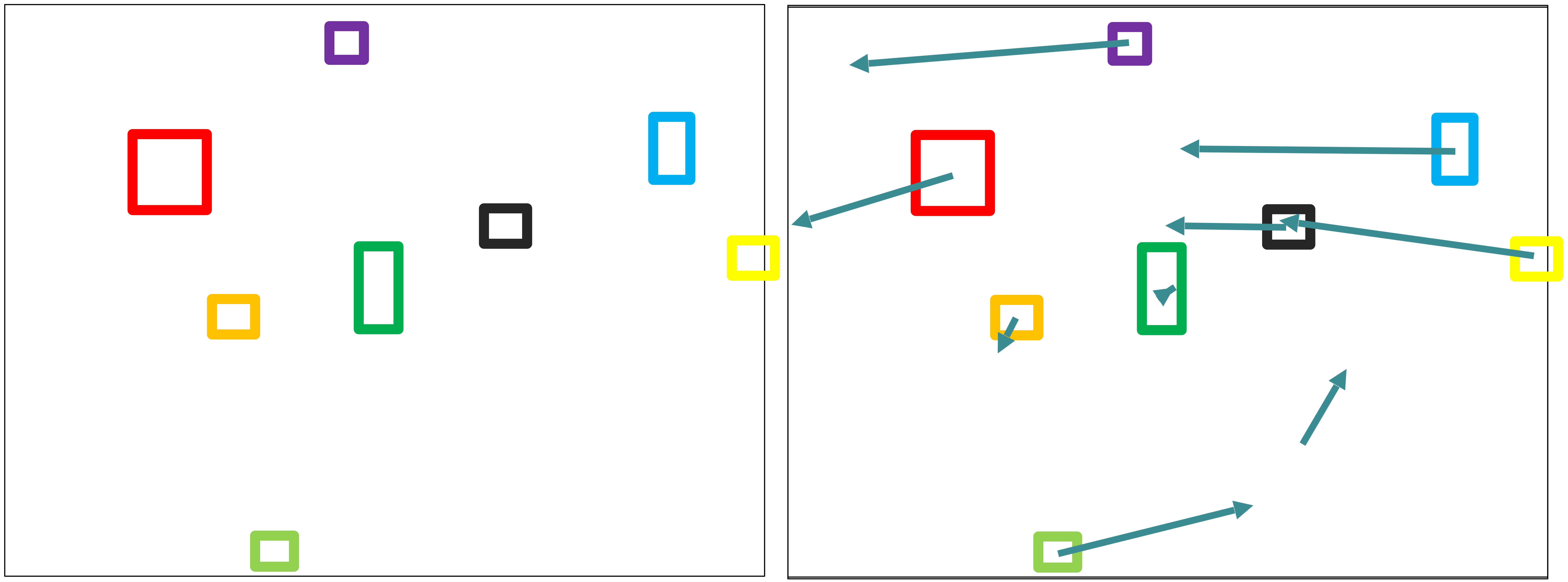
Source: Torralba, Isola, Freeman





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

Source: Torralba, Isola, Freeman

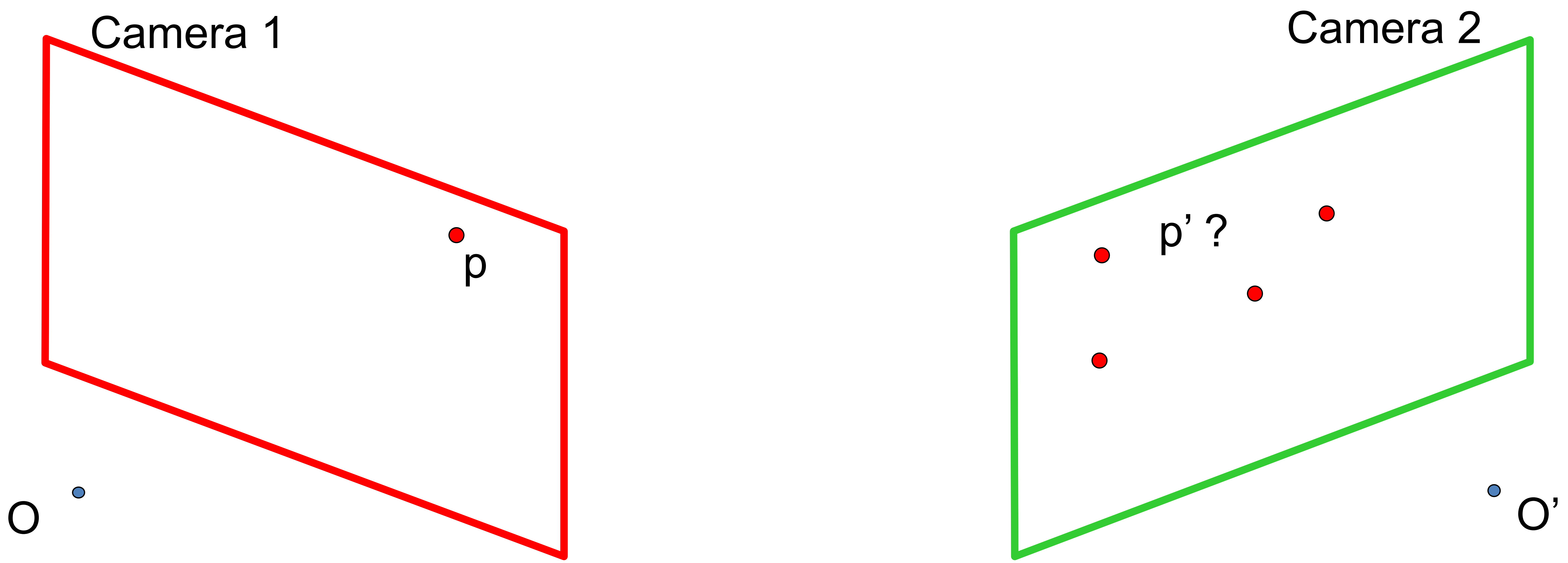


~~Do we need to search for matches only along horizontal lines?~~

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

Source: Torralba, Isola, Freeman

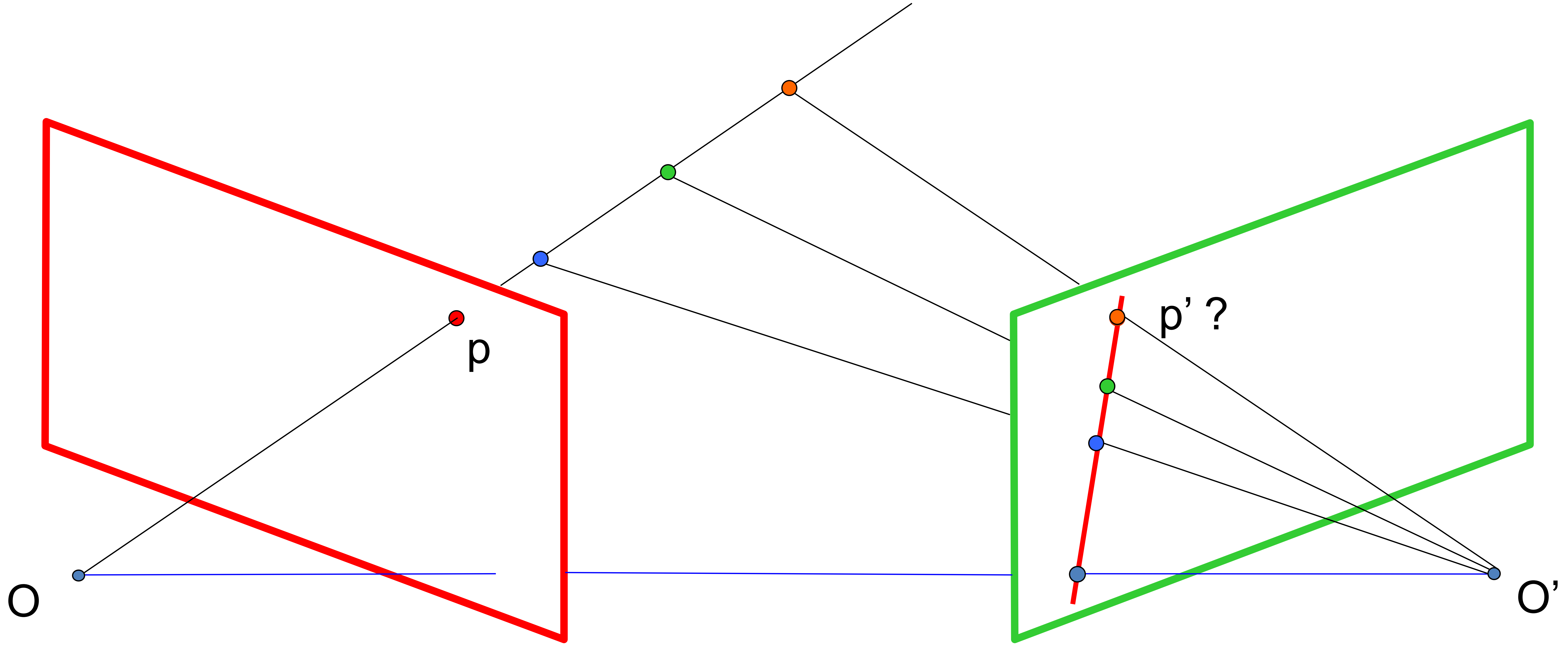
Stereo correspondence constraints



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

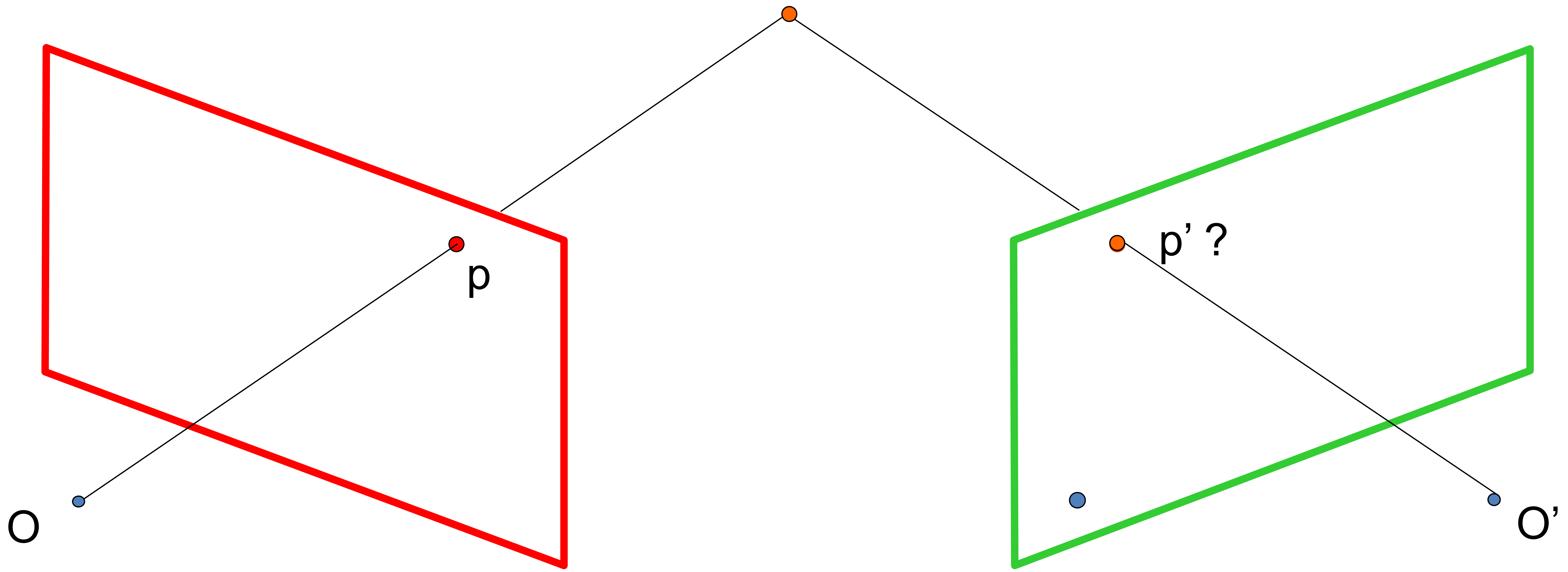
Source: Torralba, Isola, Freeman

Stereo correspondence constraints



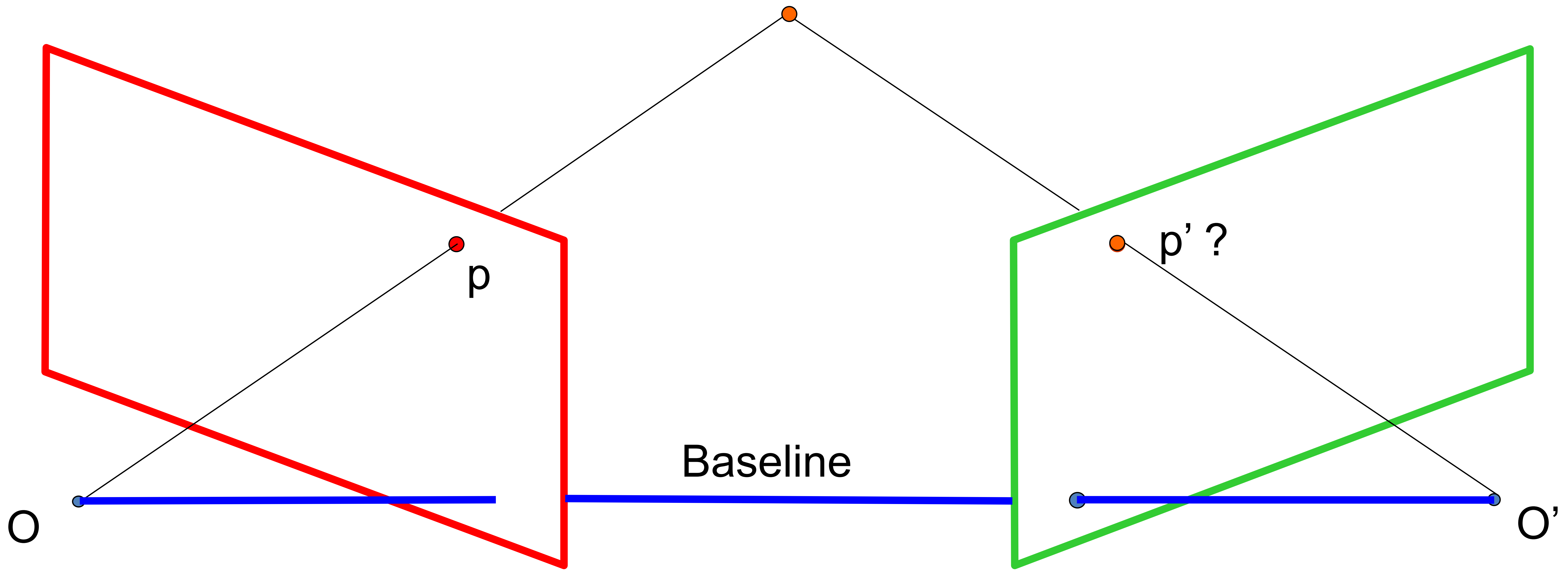
Source: Torralba, Isola, Freeman

Some terminology



Source: Torralba, Isola, Freeman

Some terminology

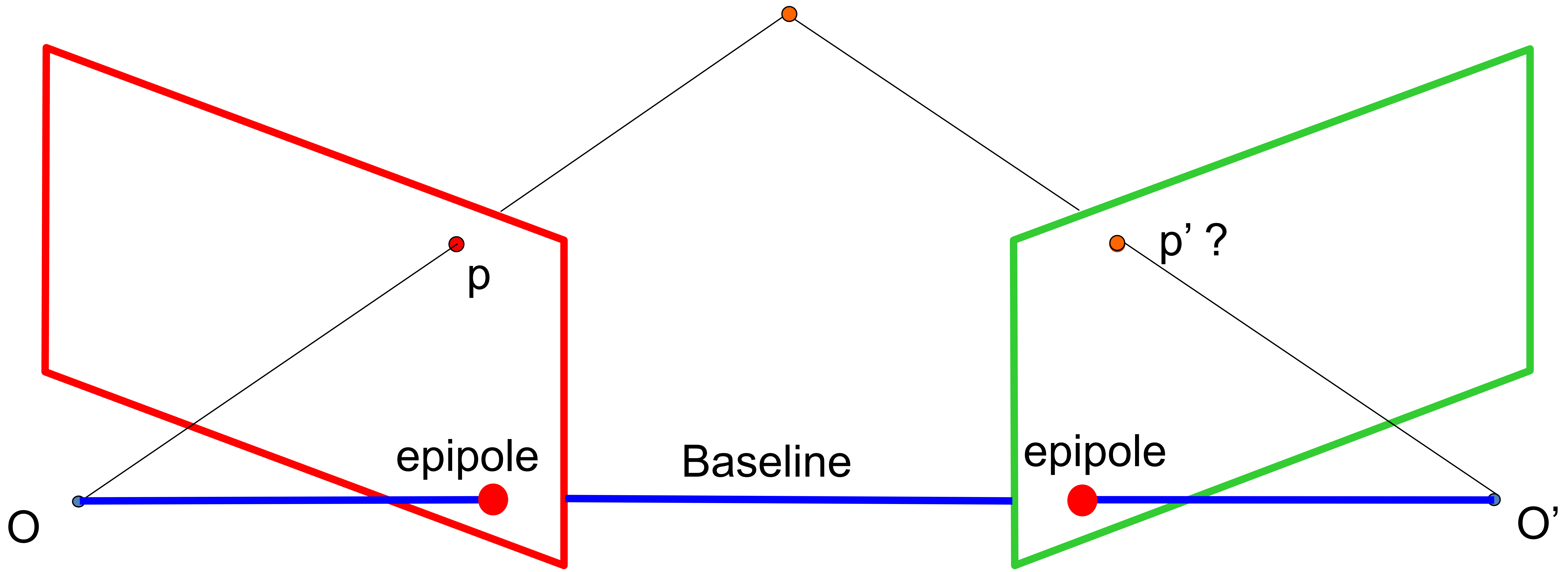


Baseline: the line connecting the two camera centers

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

Source: Torralba, Isola, Freeman

Some terminology

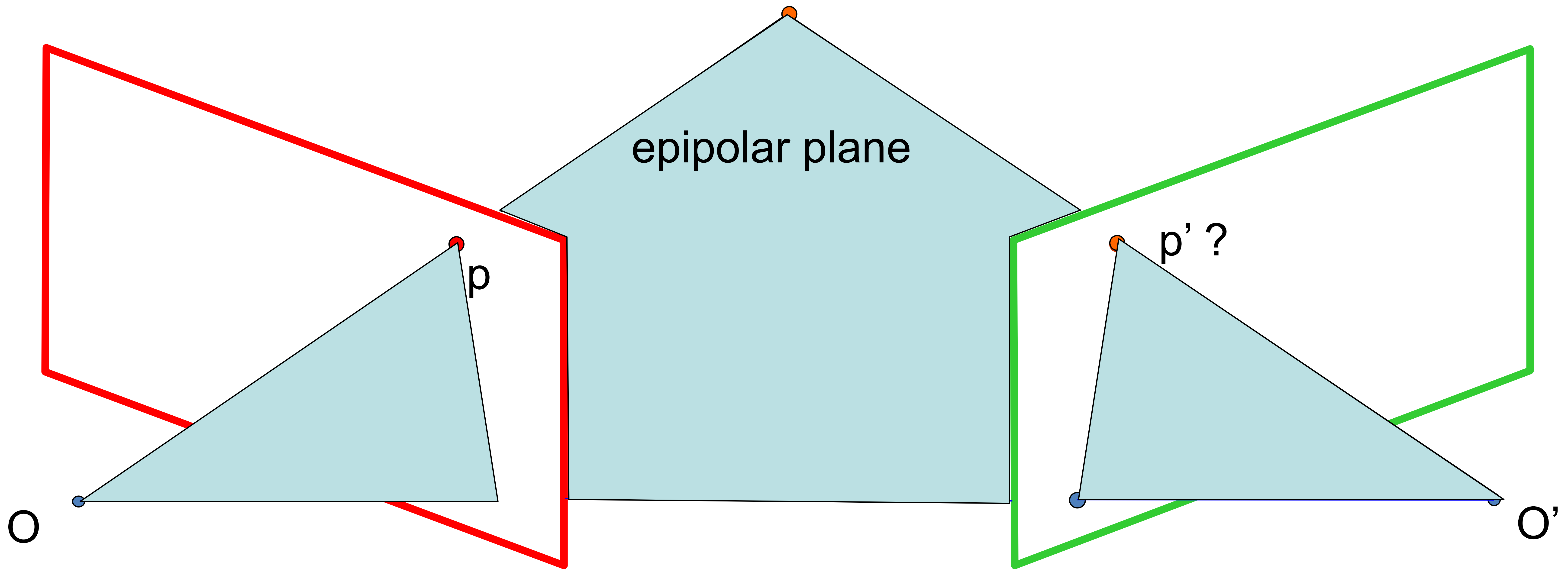


Baseline: the line connecting the two camera centers

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

Source: Torralba, Isola, Freeman

Some terminology



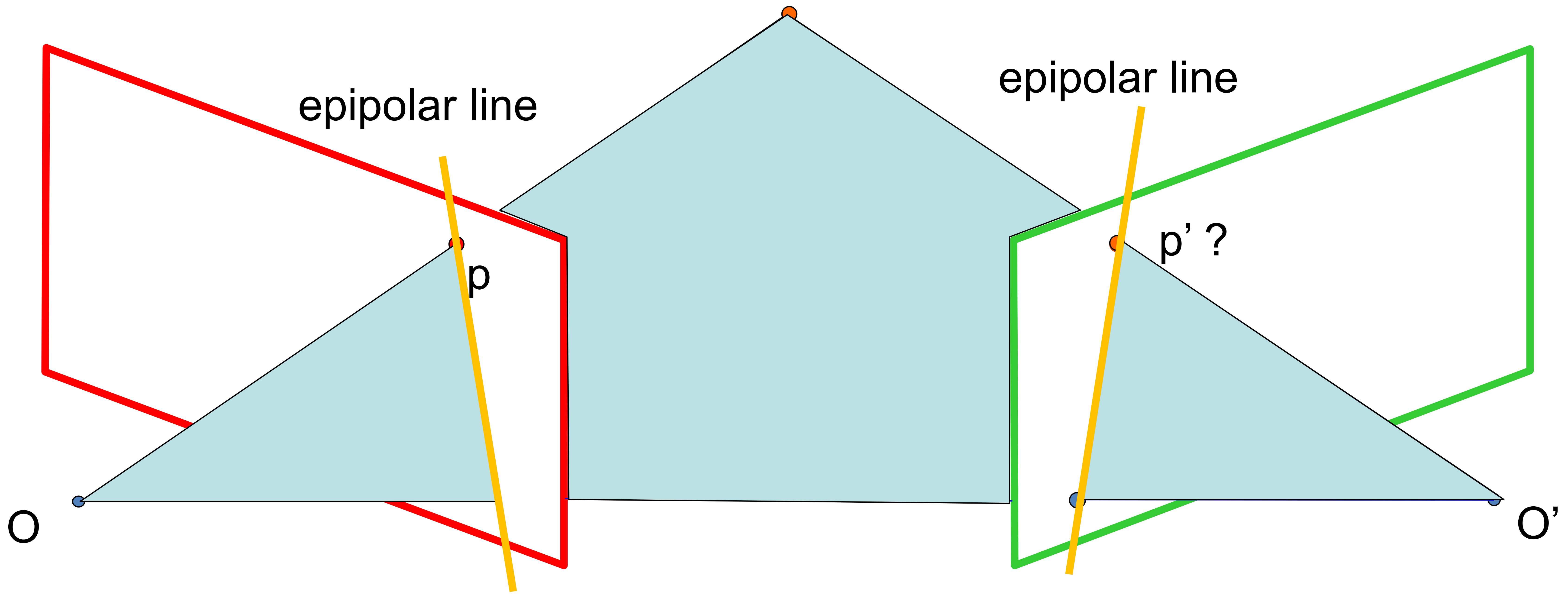
Baseline: the line connecting the two camera centers

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

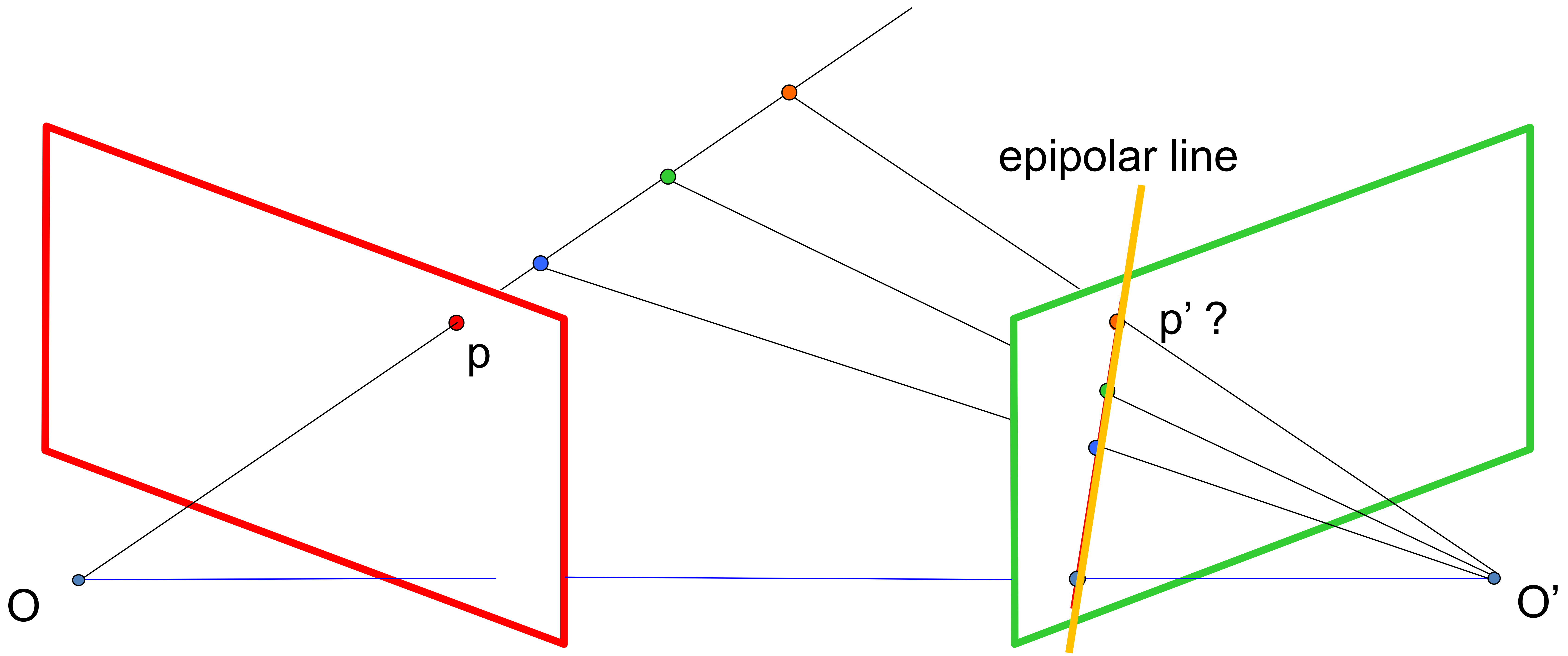
Epipole: point of intersection of *baseline* with the image plane

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

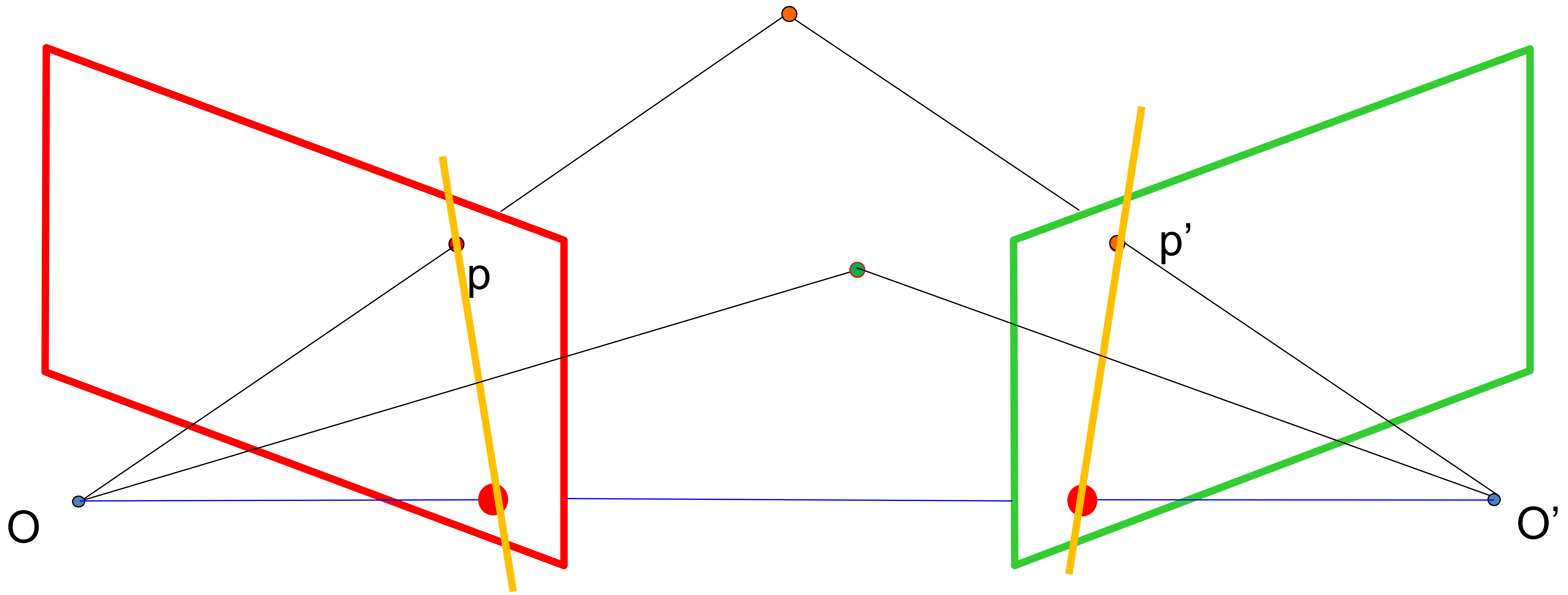


We can search for matches across epipolar lines

All epipolar lines intersect at the epipoles

Source: Torralba, Isola, Freeman

Epipolar constraint



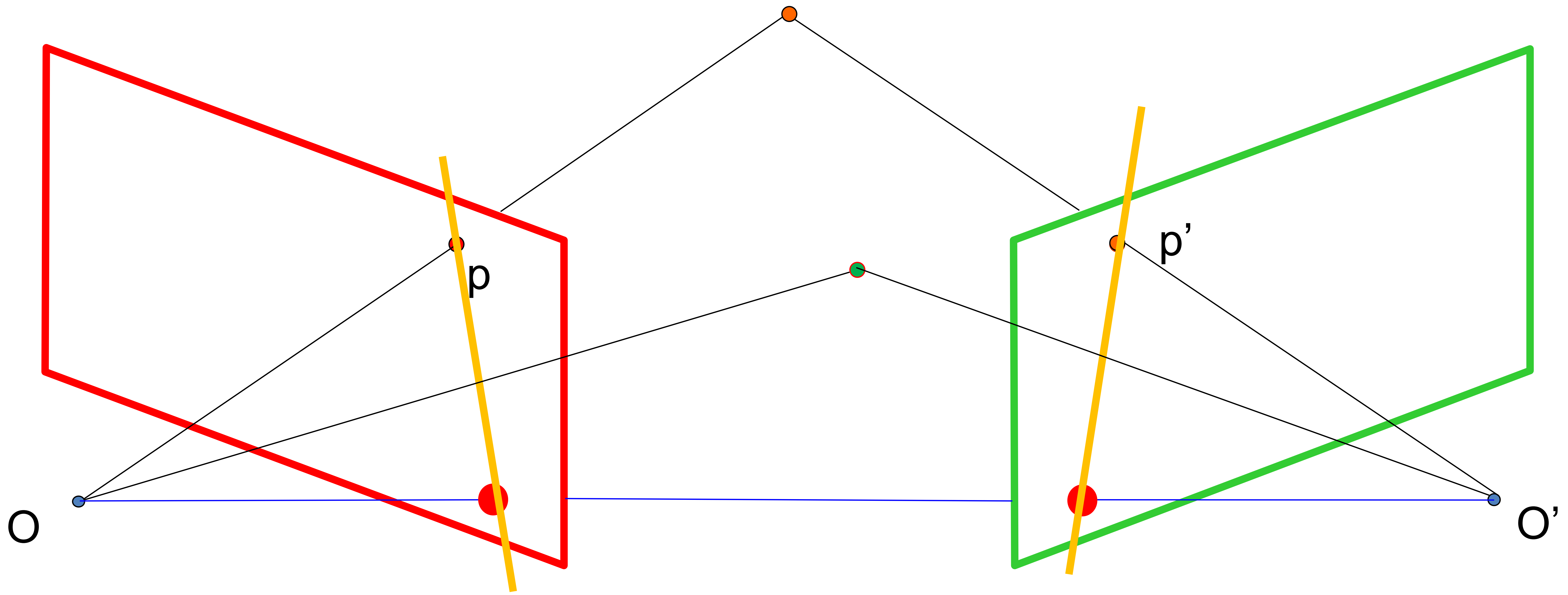
If we observe a point in one image, its position in the other image is constrained to lie on the epipolar line.

How do we get this line? We want a function that, given a point $\mathbf{p}$ tells us what this line is:

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

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

The fundamental matrix

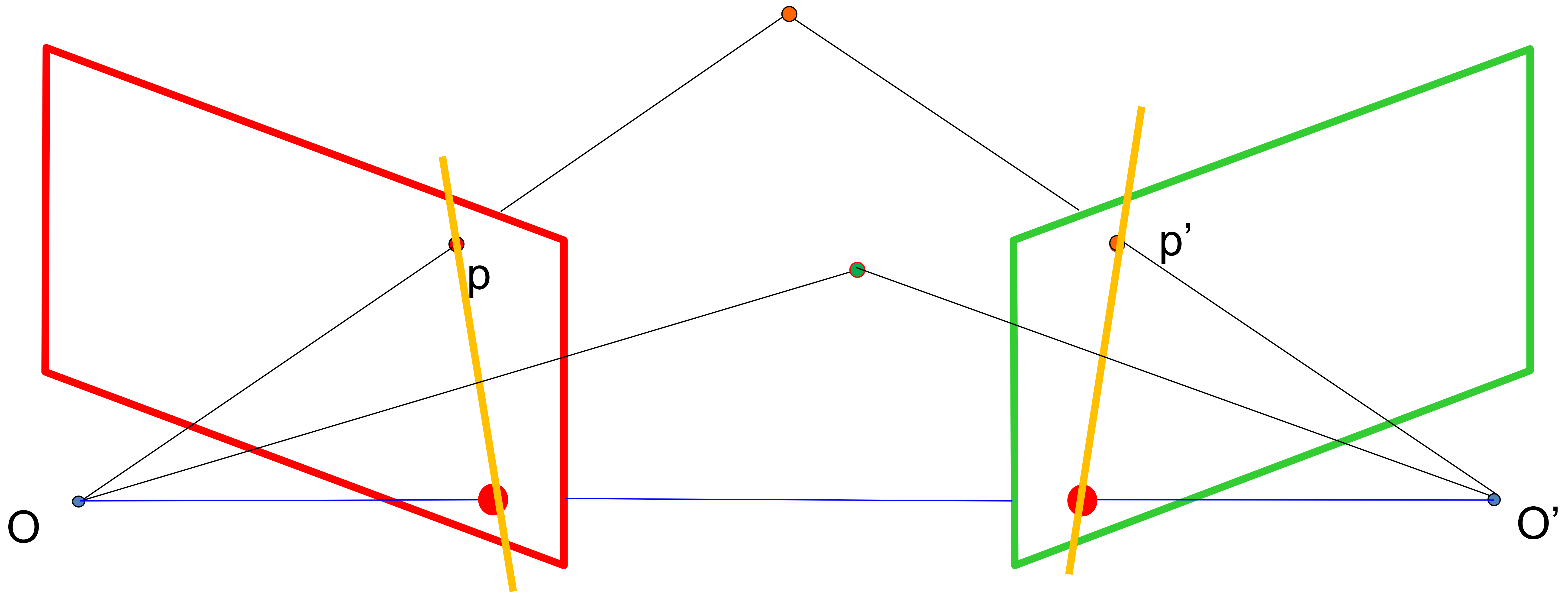


It can be shown that our function, f, can be written as a **matrix multiplication in homogeneous coordinates**.

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

F: fundamental matrix
p image point in homogeneous coordinates

The fundamental matrix



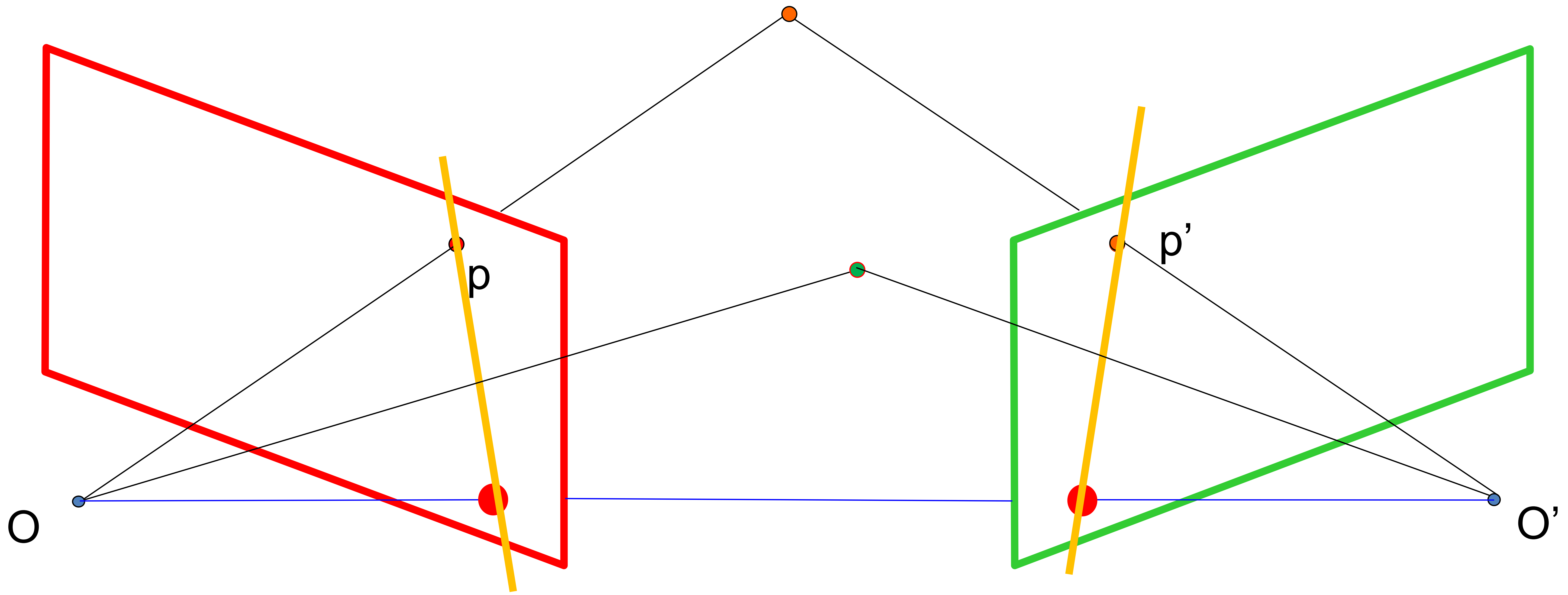
It can be shown that our function, f, can be written as a **matrix multiplication in homogeneous coordinates**.

More concisely: $$\mathbf{p}^\top F\,\mathbf{p}' = 0$$

F: fundamental matrix
p image point in homogeneous coordinates

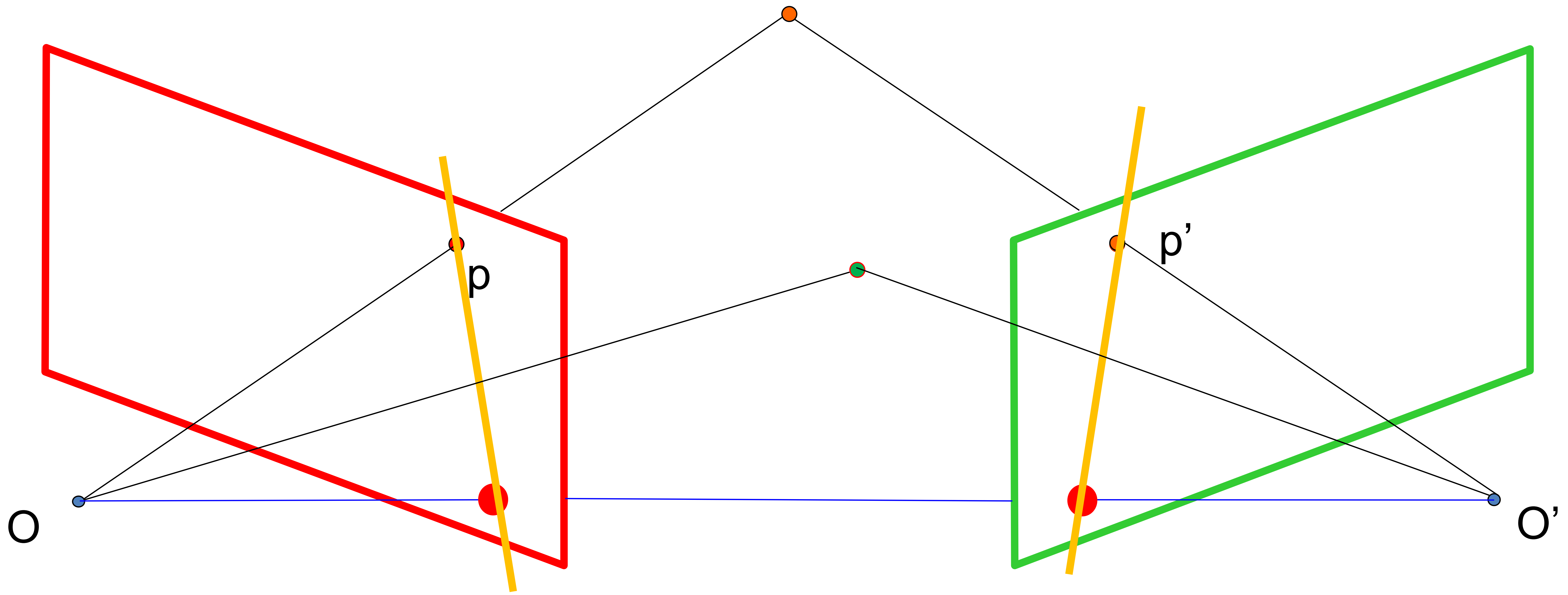
Source: Torralba, Isola, Freeman

The fundamental matrix



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

The fundamental matrix

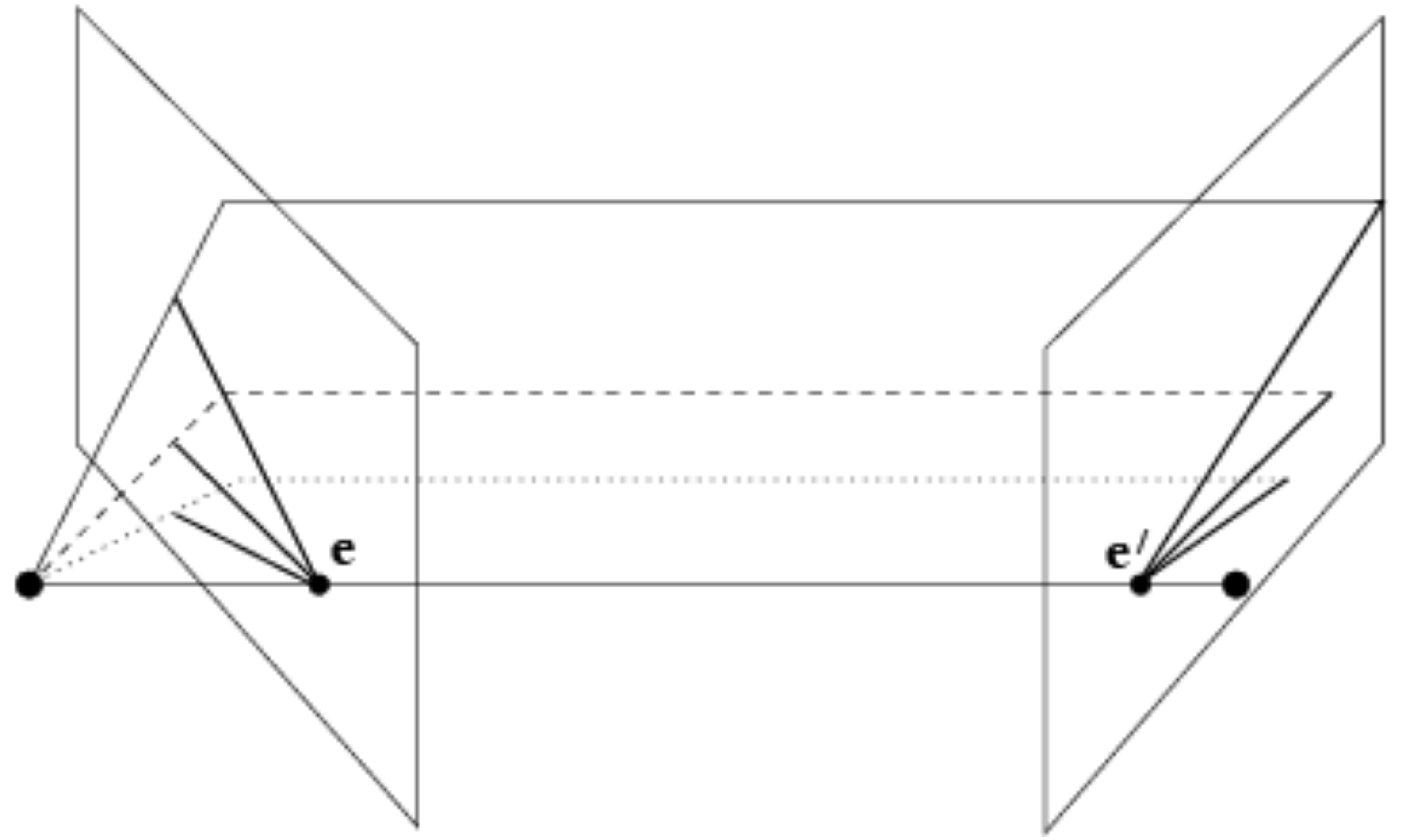


$$(\mathbf{p}^{\top} F)\ \mathbf{p}' = 0$$

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

u: a line induced by p
p, p': image points in homogeneous coordinates

Example: converging cameras



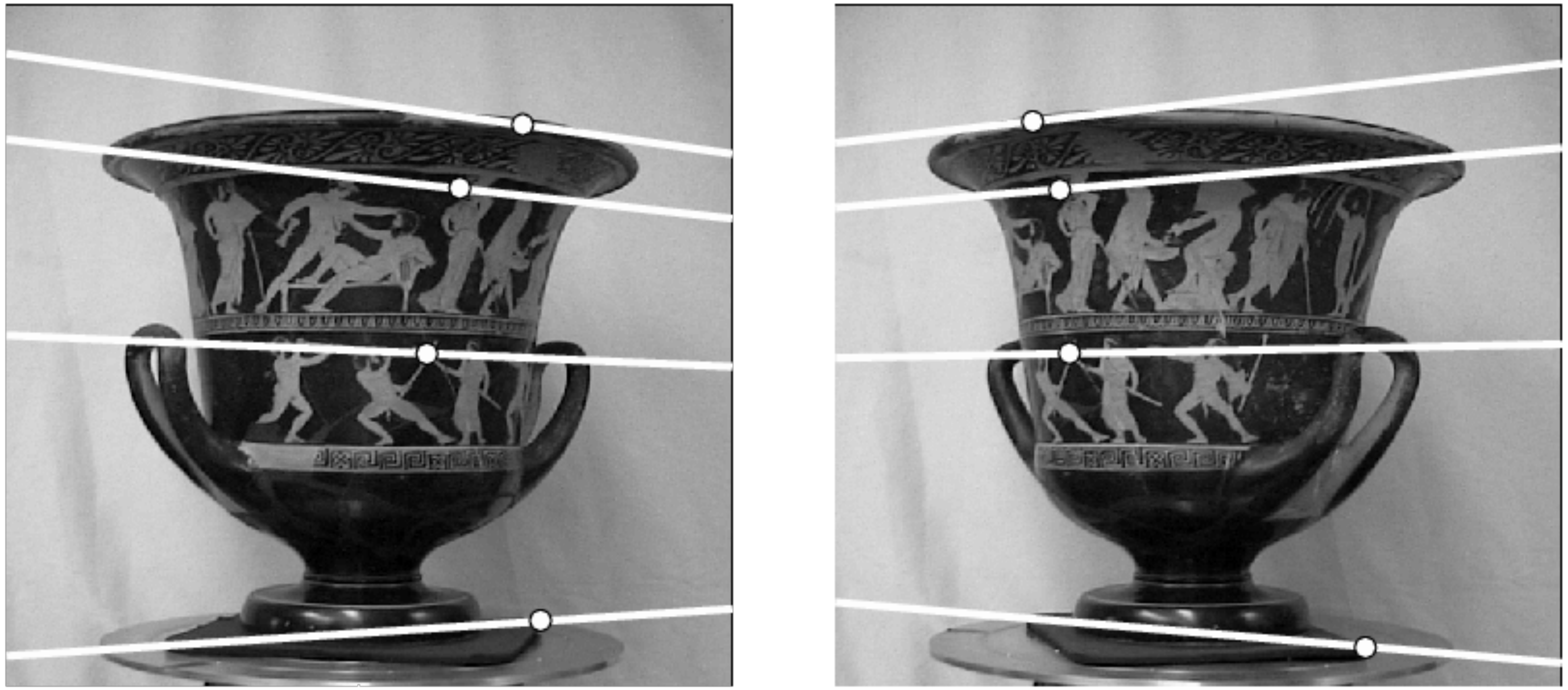
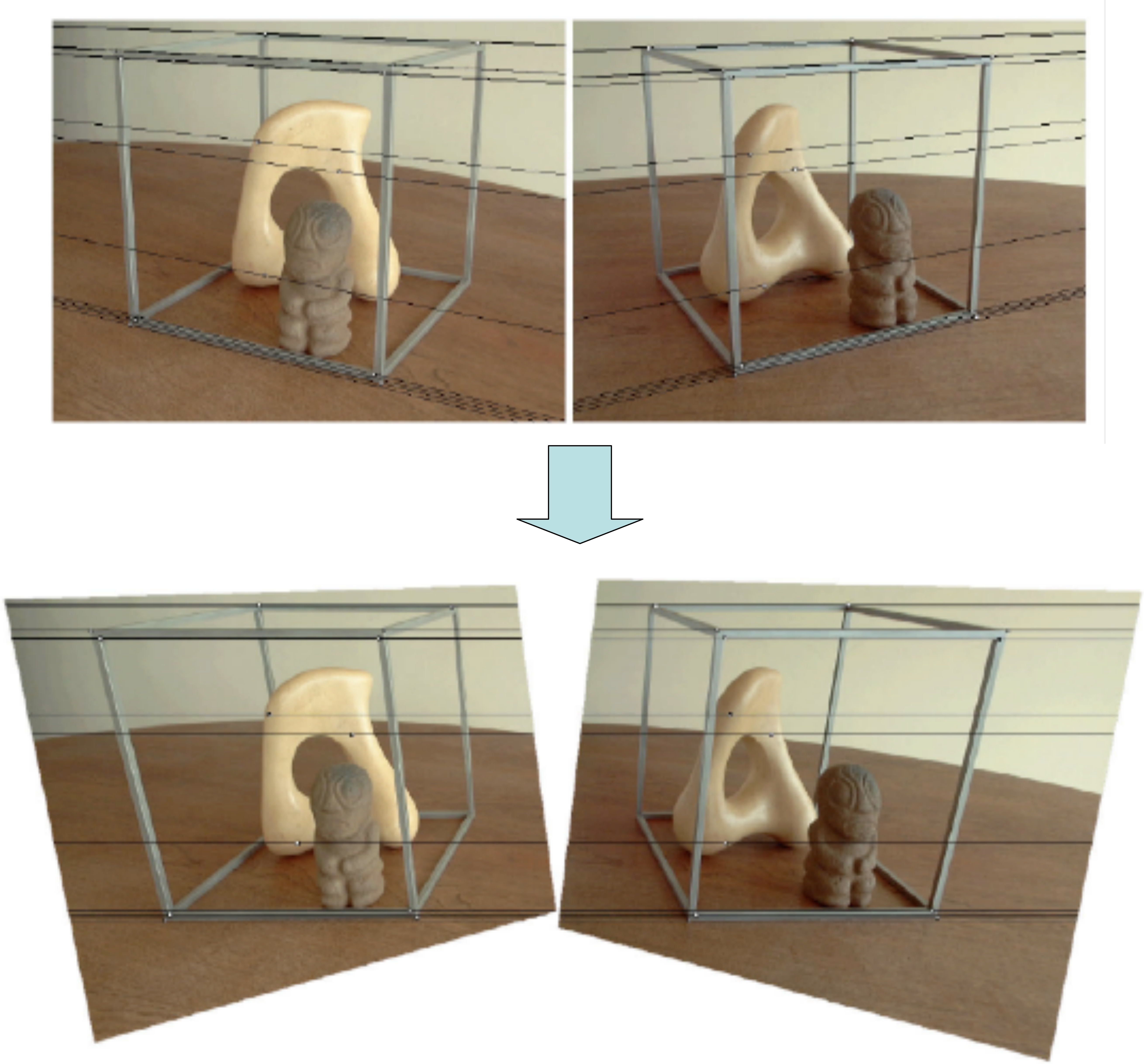


Figure from Hartley & Zisserman

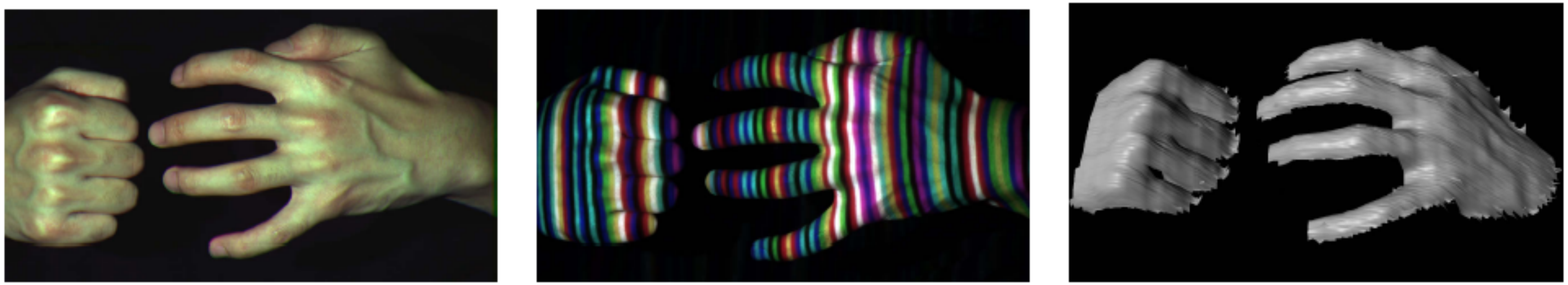
Source: Kristen Grauman

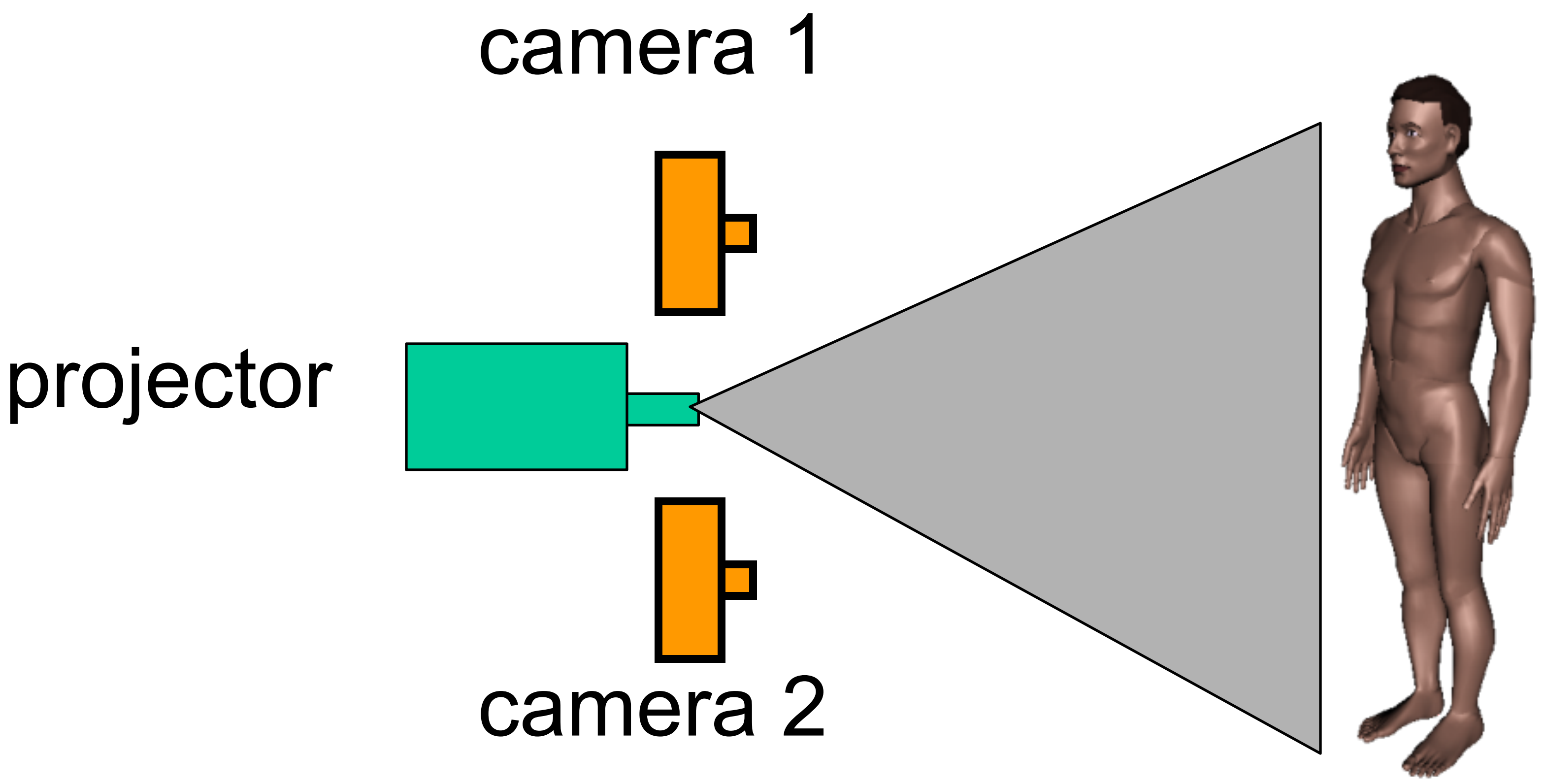
Image rectification



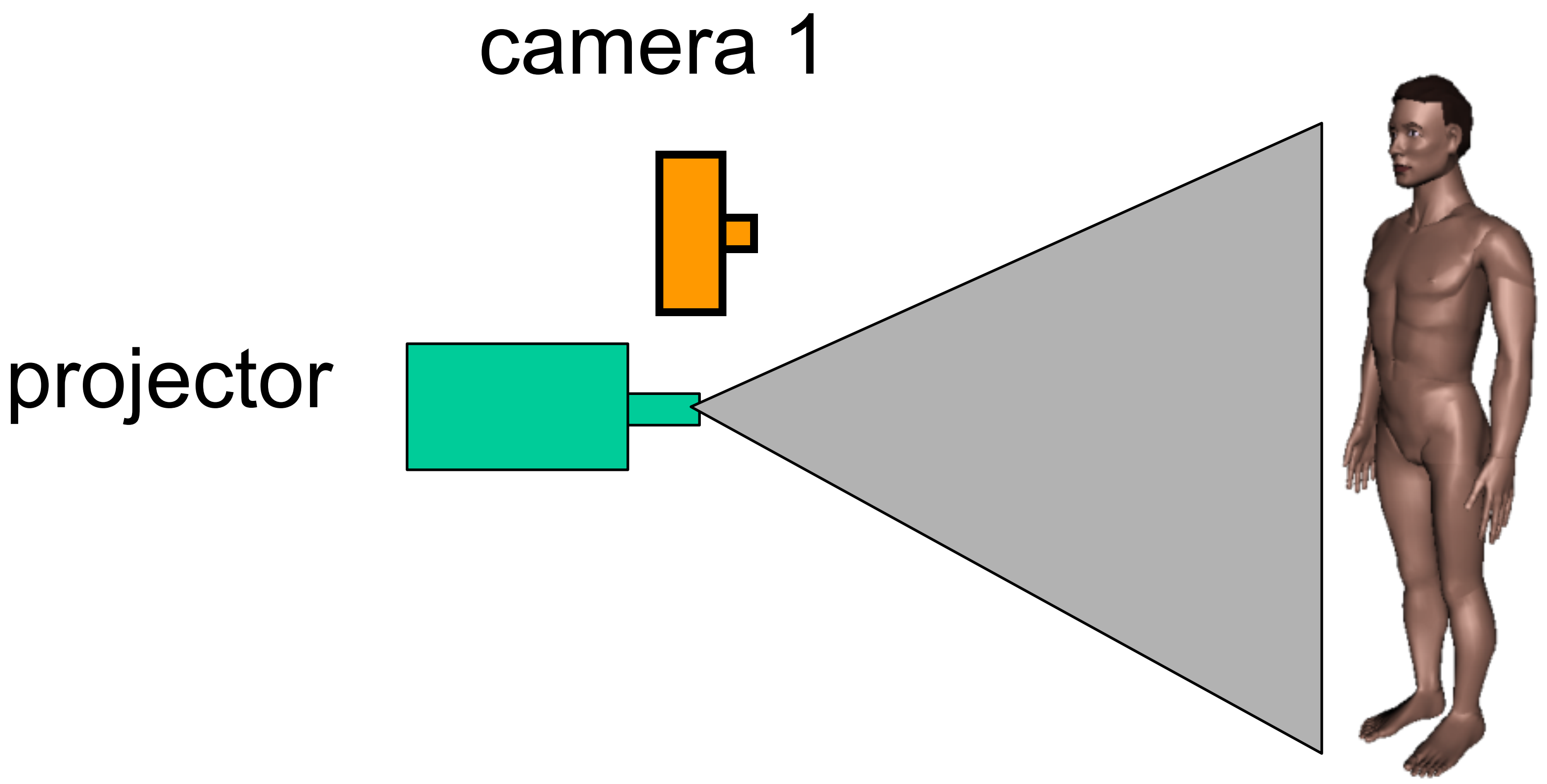
Source: A. Efros

Active stereo with structured light





Easy-to-match pattern

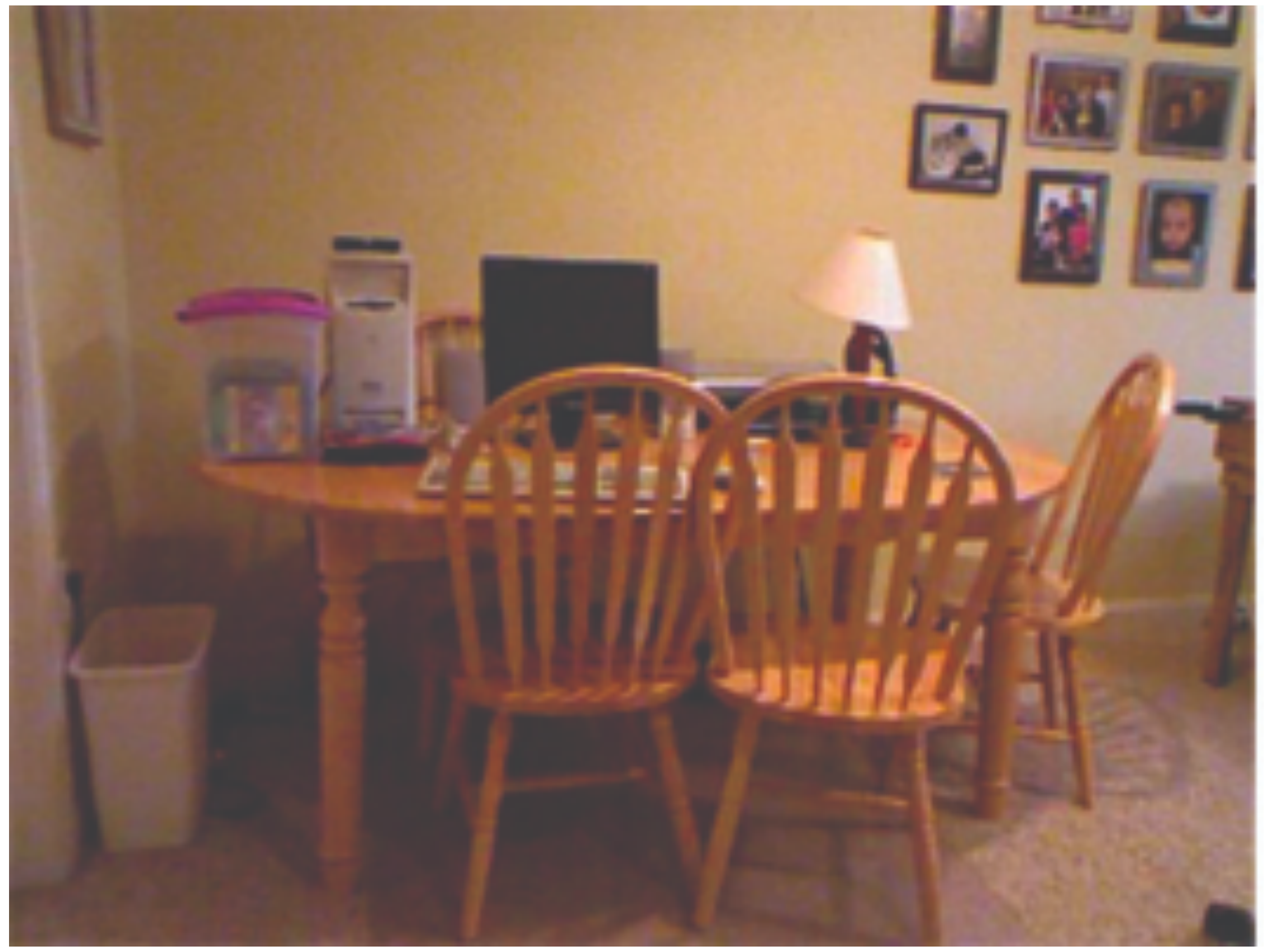


Do we really need the second camera?

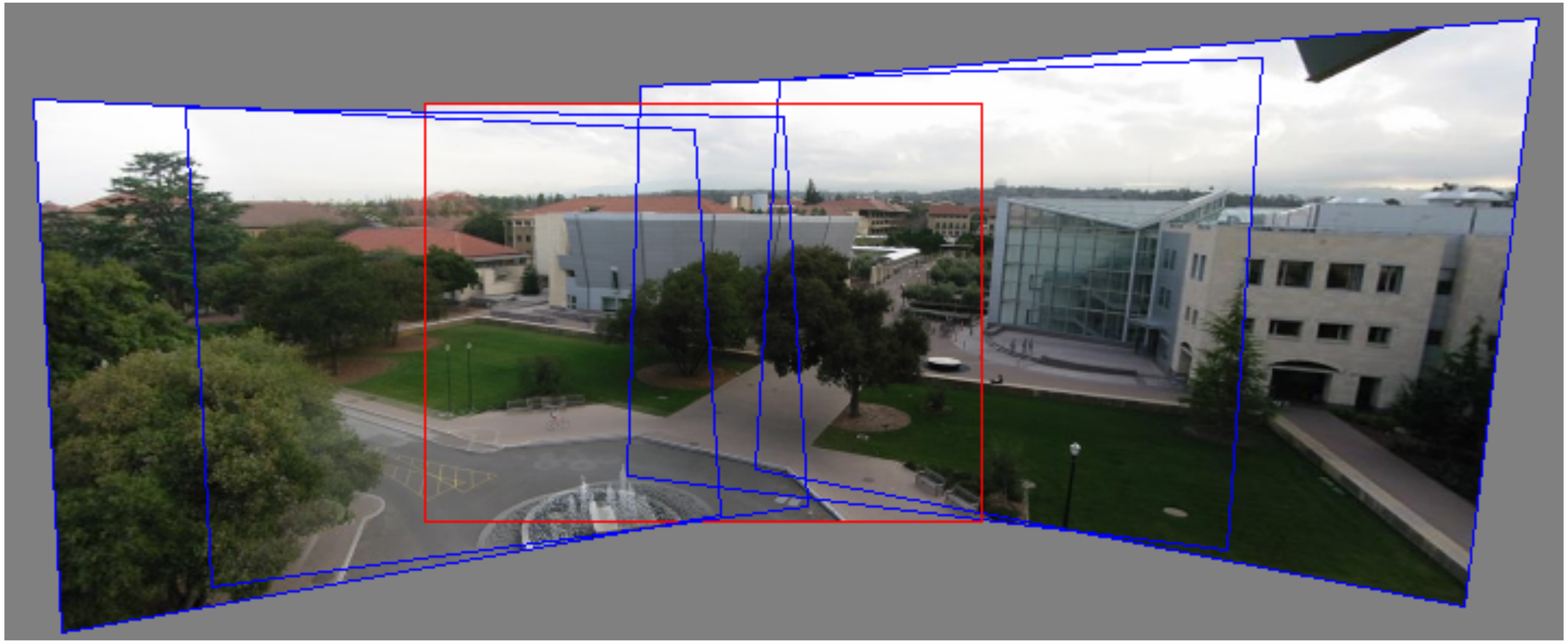
[Zhang, Curless, Seitz, 2002]

Source: R. Szeliski

RGB-D sensors



Making panoramas



Source: N. Snavely

Making panoramas

What is the geometric relationship between these two images?

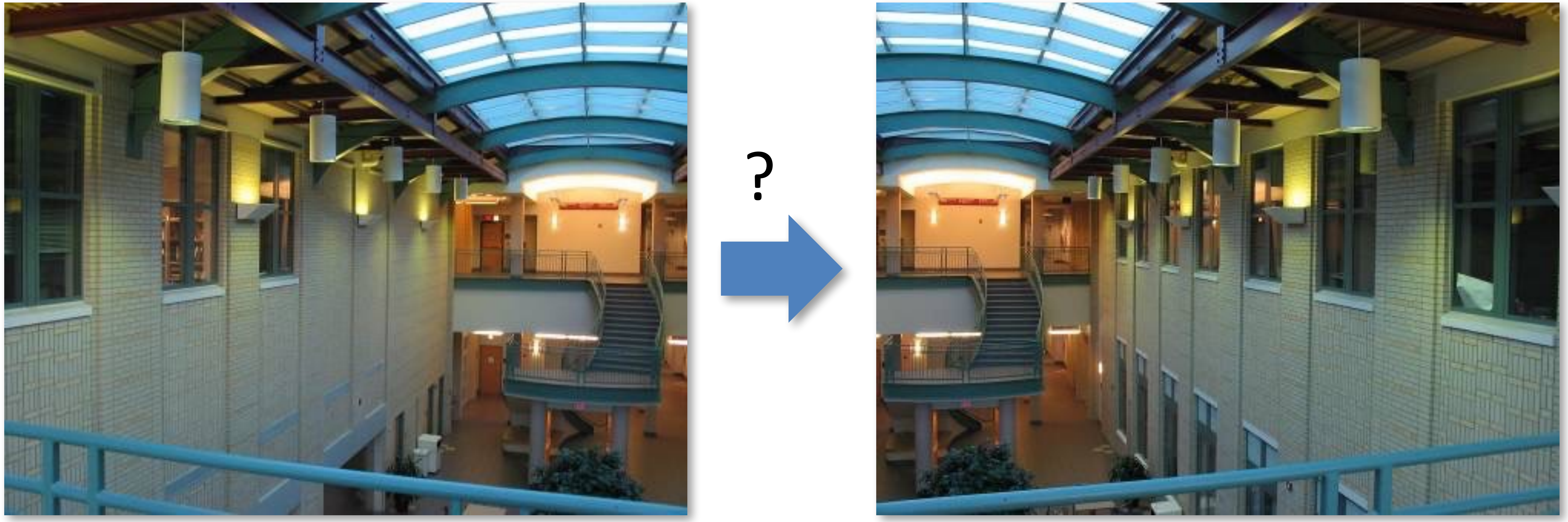


Image alignment

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

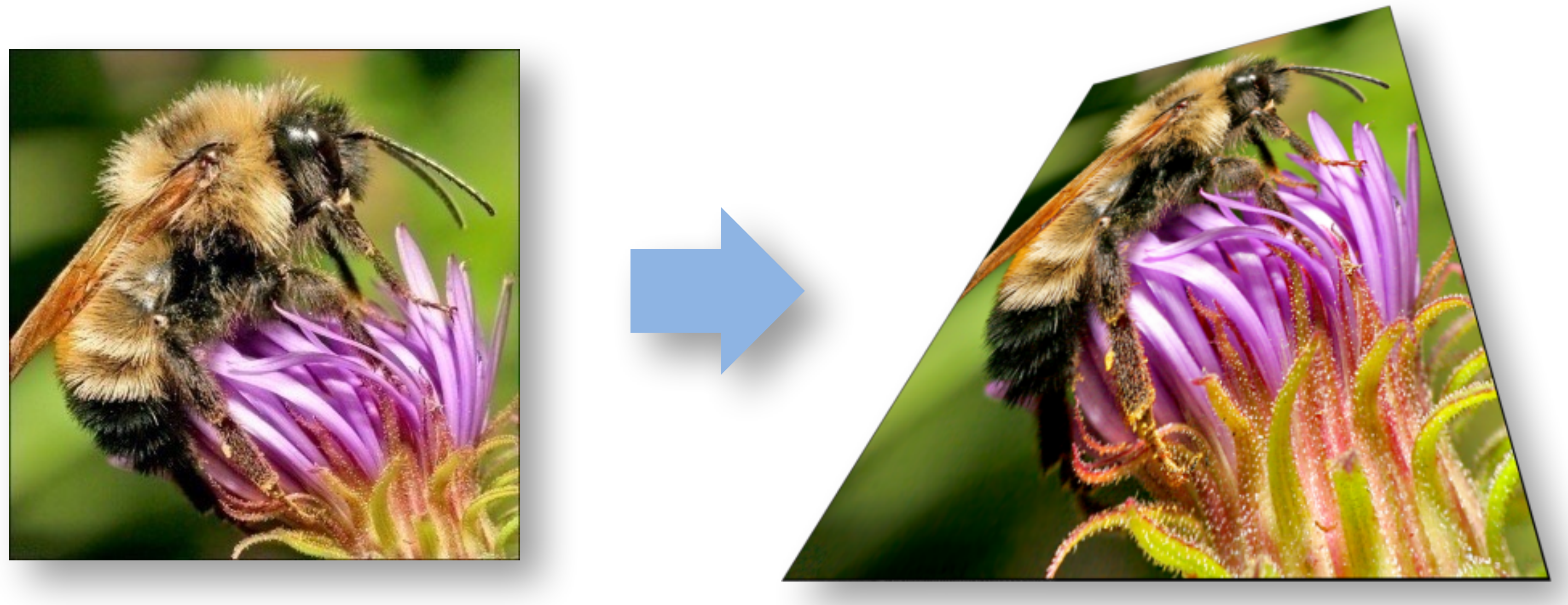
what happens when we change this row?

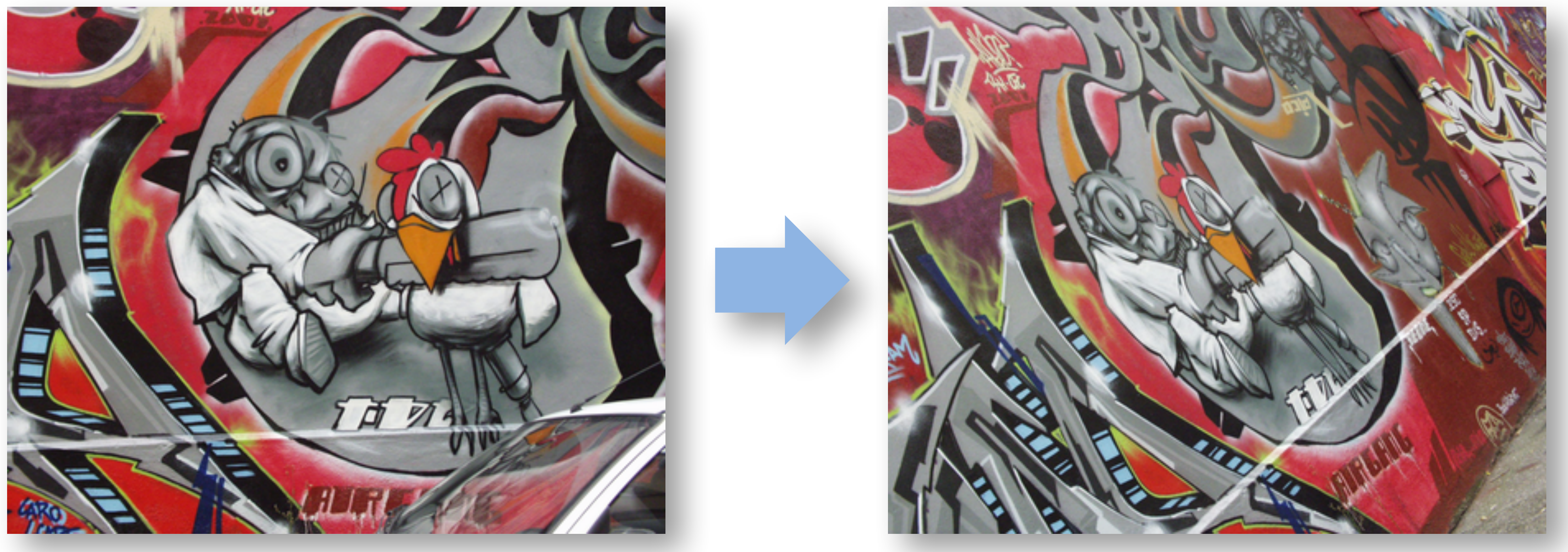
Source: N. Snavely

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

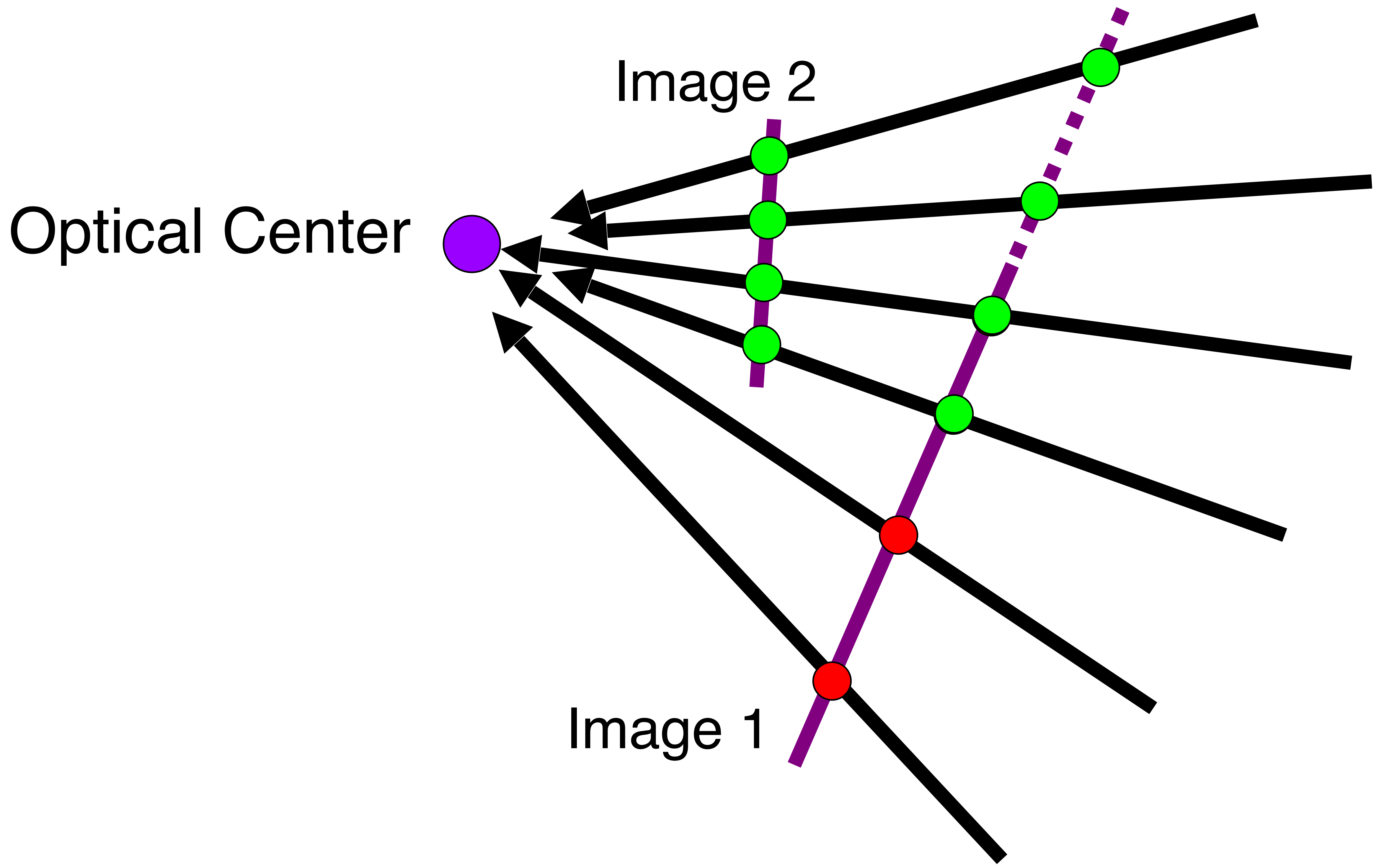
Homography

Example: two pictures taken by rotating the camera:

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

Source: Torralba, Isola, Freeman

Why does this work?



How do we map points in image 2 into image 1?

	image 1	image 2
intrinsics	$\mathbf{K}_1$	$\mathbf{K}_2$
extrinsics (rotation only)	$\mathbf{R}_1 = \mathbf{I}_{3\times 3}$	$\mathbf{R}_2$

Step 1: Convert pixels in image 2 to rays in camera 2's coordinate system.

$$\begin{bmatrix} X_2 \\ Y_2 \\ Z_2 \end{bmatrix} = \mathbf{K}_2^{-1} \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix}$$

Step 2: Convert rays in camera 2's coordinates to rays in camera 1's coordinates.

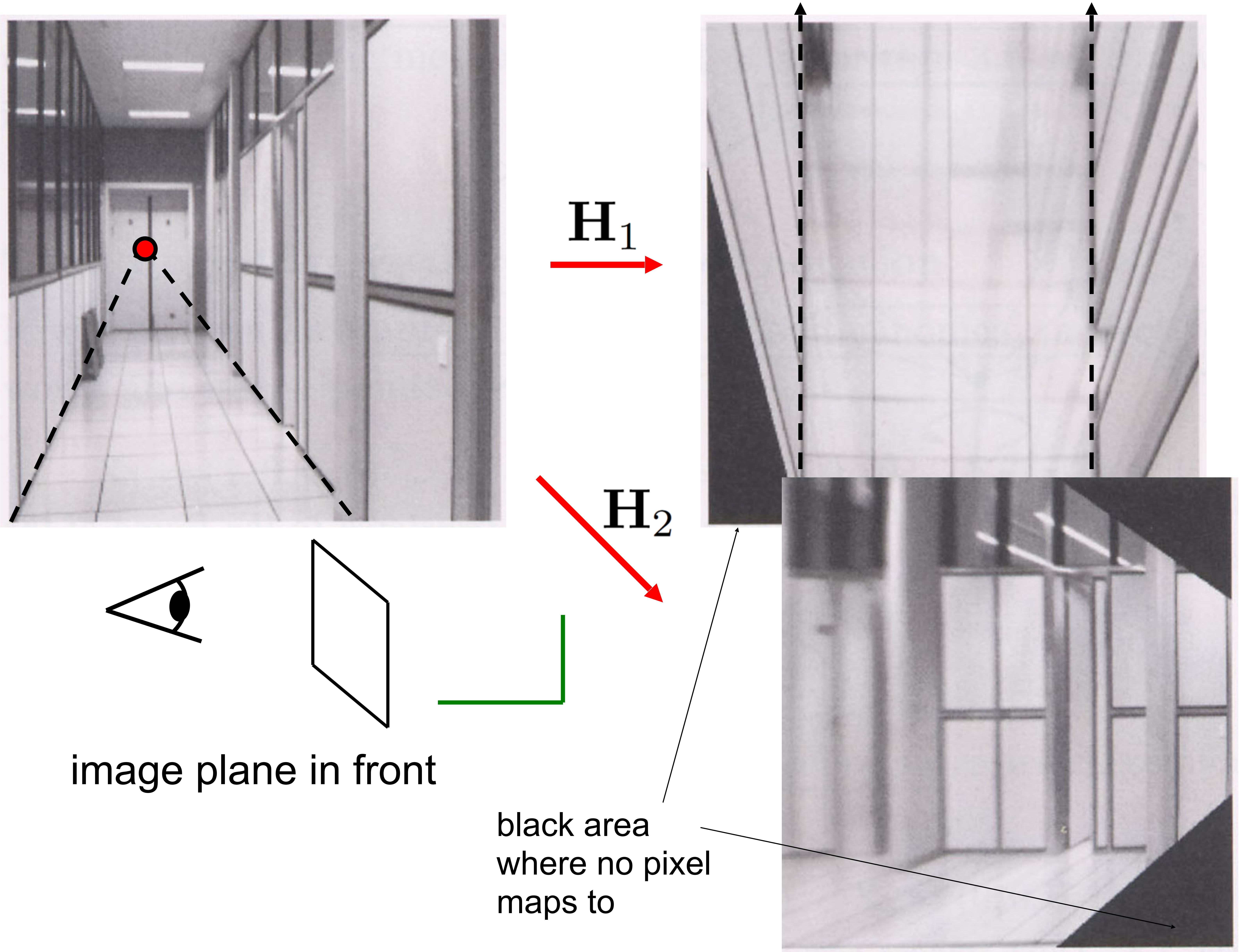
$$\begin{bmatrix} X_1 \\ Y_1 \\ Z_1 \end{bmatrix} = \mathbf{R}_2^T \mathbf{K}_2^{-1} \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix}$$

Step 3: Convert rays in camera 1's coordinates to pixels in image 1's coordinates.

$$\begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} \sim \underbrace{\mathbf{K}_1 \mathbf{R}_2^T \mathbf{K}_2^{-1}}_{\text{3x3 homography}} \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix}$$

Source: N. Snavely

Plane-to-plane homography



Source: N. Snavely

Homographies

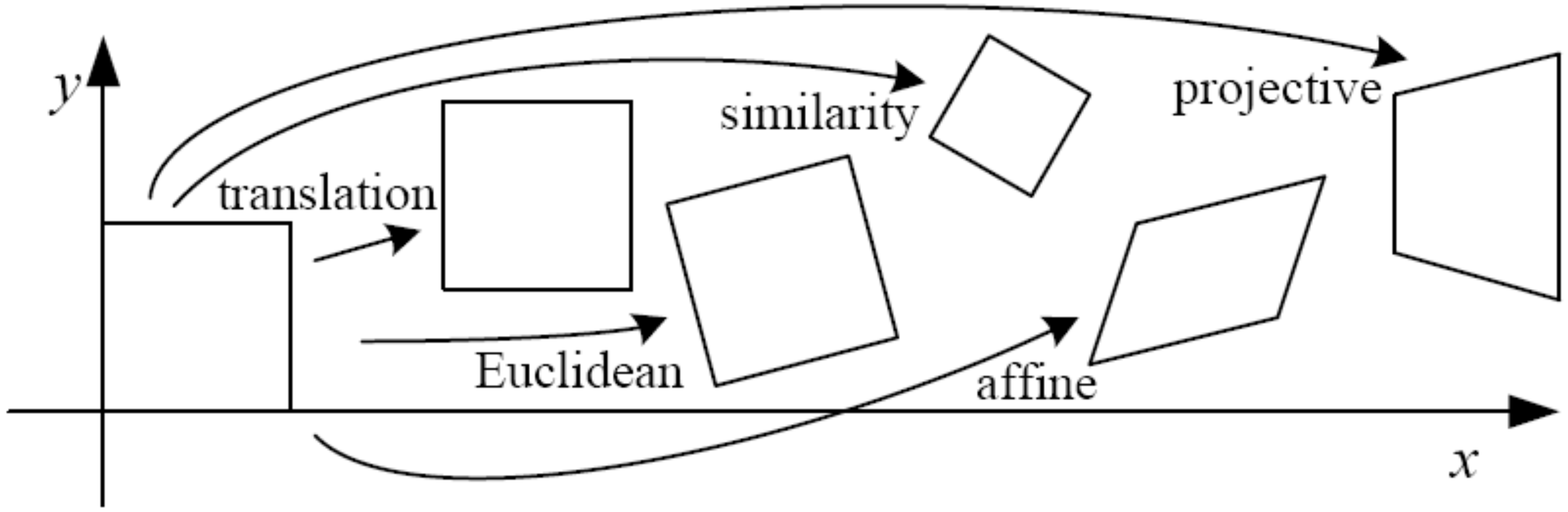
- Homographies …
  – Affine transformations, and
  – Projective warps

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

- 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



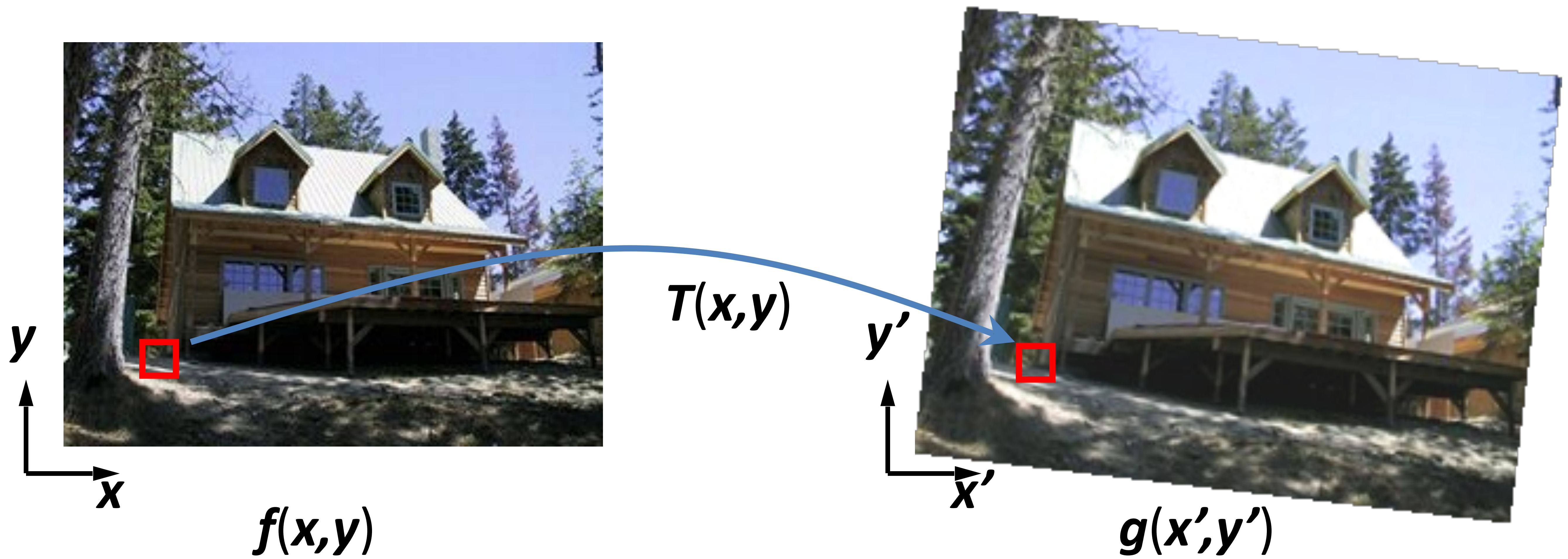
Name	Matrix	# D.O.F.	Preserves:	Icon
translation	$\begin{bmatrix} I \mid t \end{bmatrix}_{2\times3}$	2	orientation $+\cdots$	
rigid (Euclidean)	$\begin{bmatrix} R \mid t \end{bmatrix}_{2\times3}$	3	lengths $+\cdots$	
similarity	$\begin{bmatrix} sR \mid t \end{bmatrix}_{2\times3}$	4	angles $+\cdots$	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

Source: N. Snavely

How do we perform this warp?

Image warping

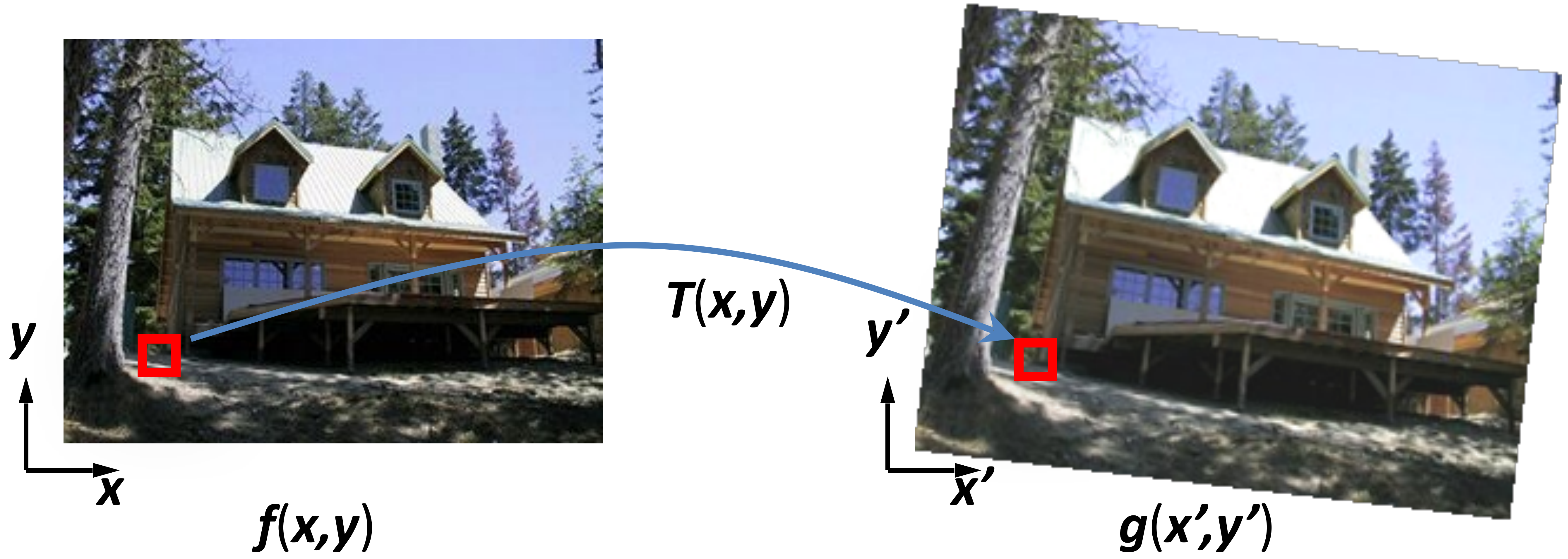
Given a coordinate transformation $(\boldsymbol{x'},\boldsymbol{y'}) = \boldsymbol{T}(\boldsymbol{x},\boldsymbol{y})$ and a source image $\boldsymbol{f}(\boldsymbol{x},\boldsymbol{y})$, how do we compute a transformed image $\boldsymbol{g}(\boldsymbol{x'},\boldsymbol{y'}) = \boldsymbol{f}(\boldsymbol{T}(\boldsymbol{x},\boldsymbol{y}))$?



Source: N. Snavely

Forward warping

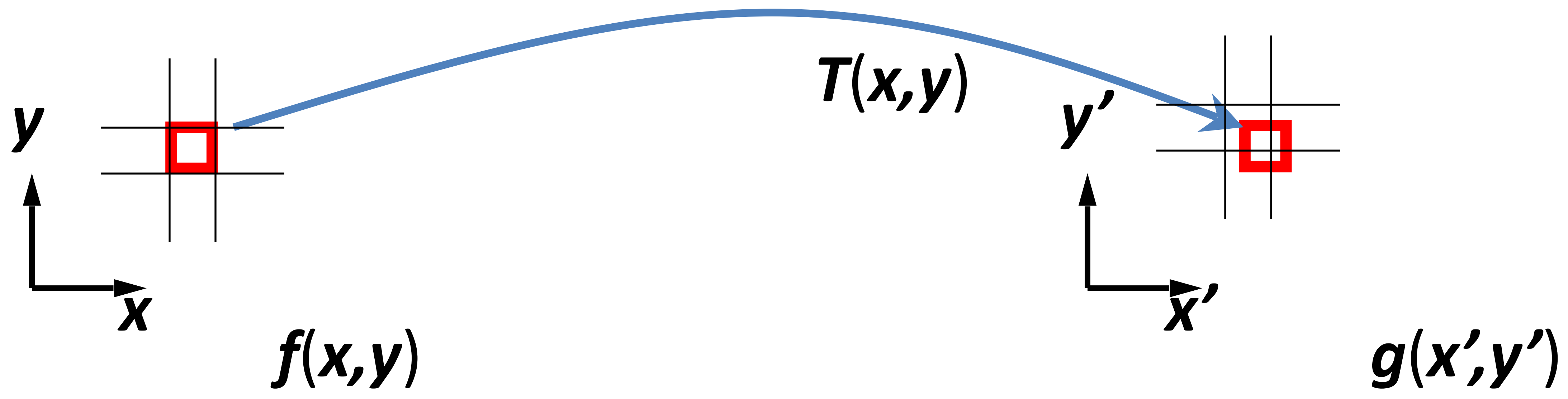
- Send each pixel ***f***(***x***) to its corresponding location (***x'***,***y'***) = ***T***(***x***,***y***) in ***g***(***x'***,***y'***)
- What if a pixel lands “between” two pixels?



Source: N. Snavely

Forward warping

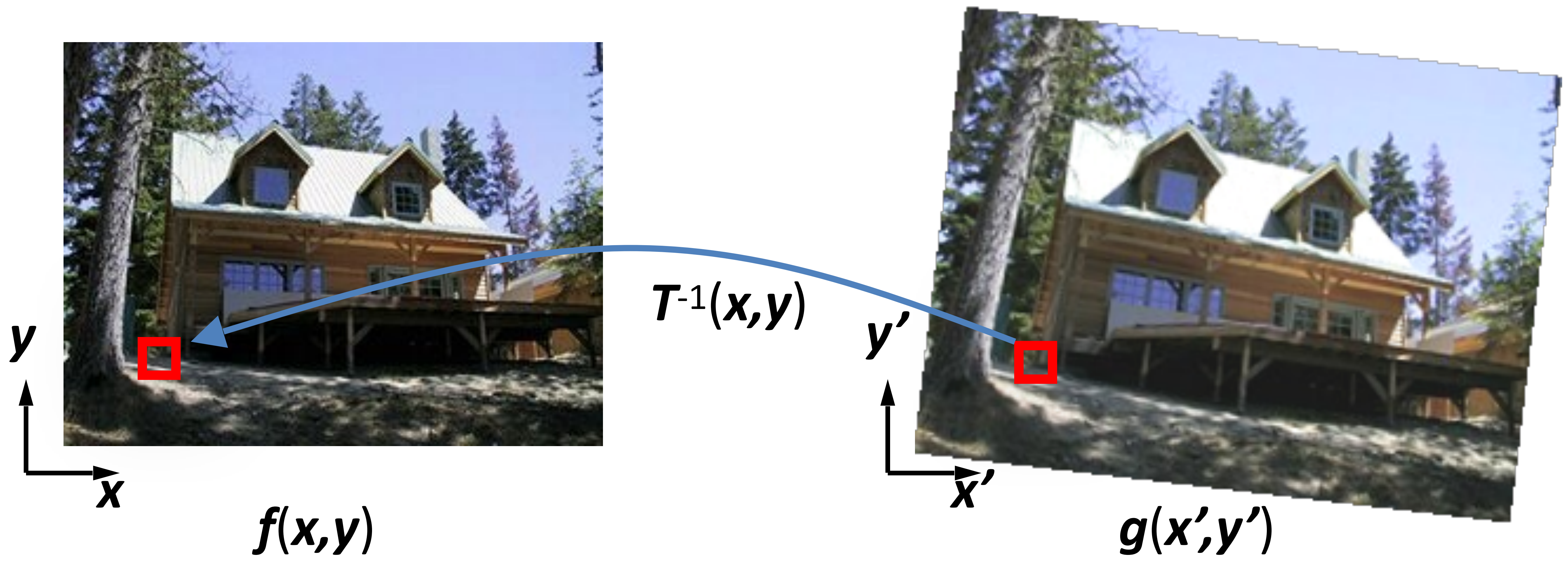
- Send each pixel $\boldsymbol{f}(\boldsymbol{x})$ to its corresponding location $(\boldsymbol{x'},\boldsymbol{y'}) = \boldsymbol{T}(\boldsymbol{x},\boldsymbol{y})$ in $\boldsymbol{g}(\boldsymbol{x'},\boldsymbol{y'})$
- What if a pixel lands "between" two pixels?
  - Answer: add "contribution" to several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



Source: N. Snavely

Inverse warping

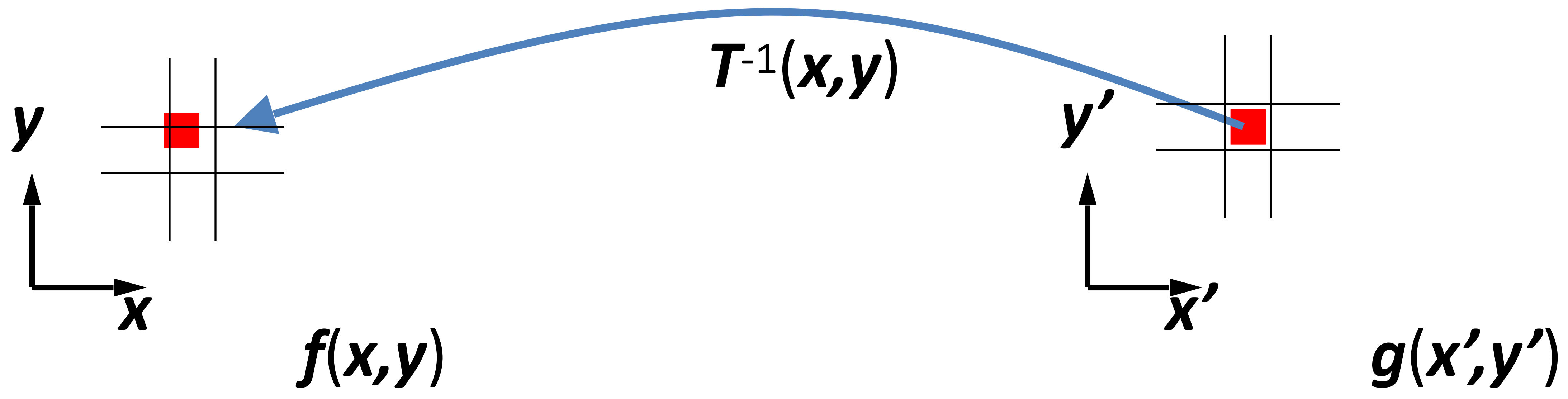
- Get each pixel $\boldsymbol{g}(\boldsymbol{x'},\boldsymbol{y'})$ from its corresponding location $(\boldsymbol{x},\boldsymbol{y}) = \boldsymbol{T}^{-1}(\boldsymbol{x},\boldsymbol{y})$ in $\boldsymbol{f}(\boldsymbol{x},\boldsymbol{y})$
  - Requires taking the inverse of the transform
  - What if pixel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



Source: N. Snavely

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 lecture: estimating geometry from images