# **Lecture 16:** Image formation

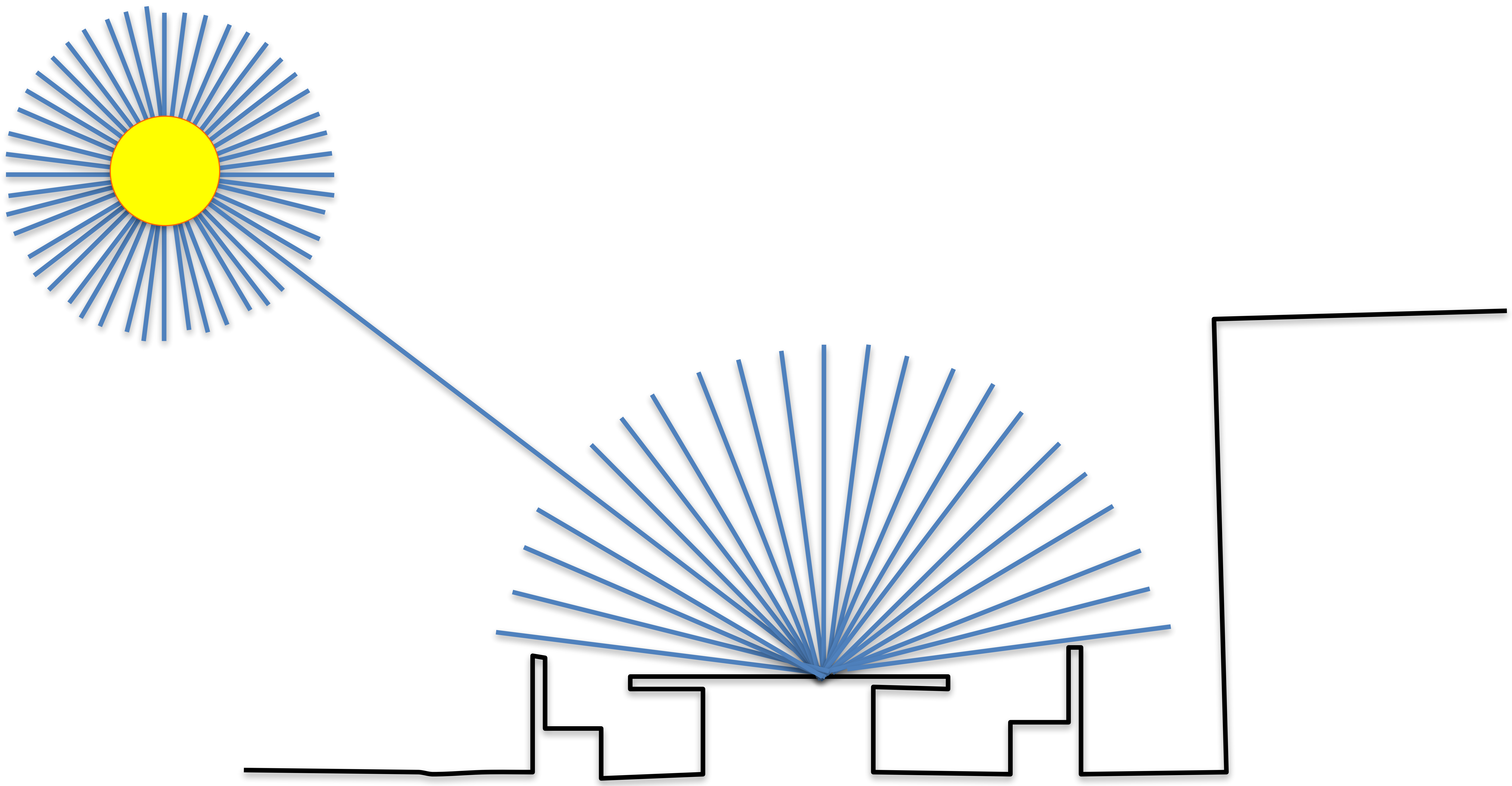
# Announcements

- Reminder: PS2 grades out
- Next problem sets:
  - PS7: representation learning
  - PS8: panorama stitching

# Today

- Camera models
- Projection equations

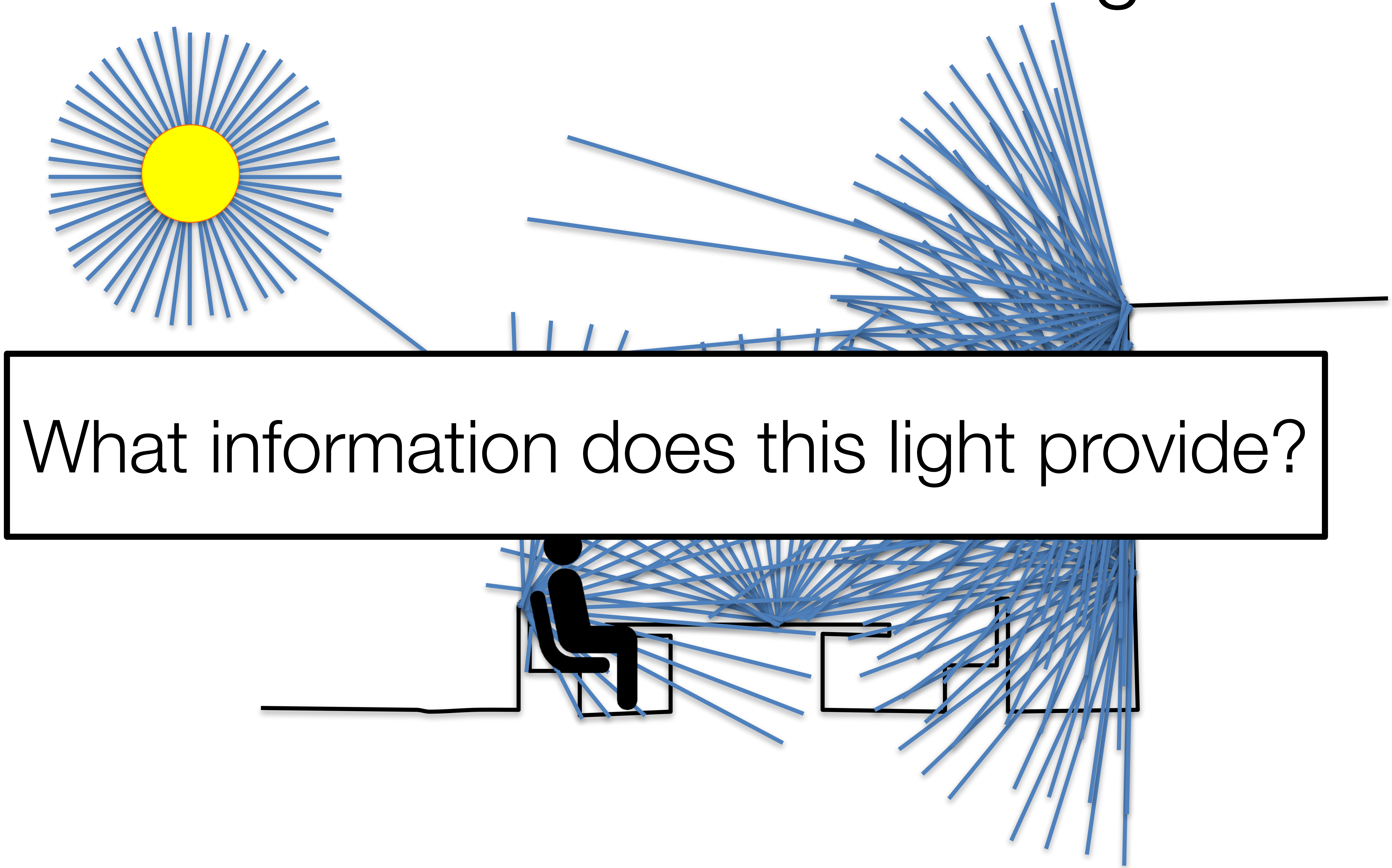
# The structure of ambient light





Source: Freeman, Torralba, Isola

# The structure of ambient light

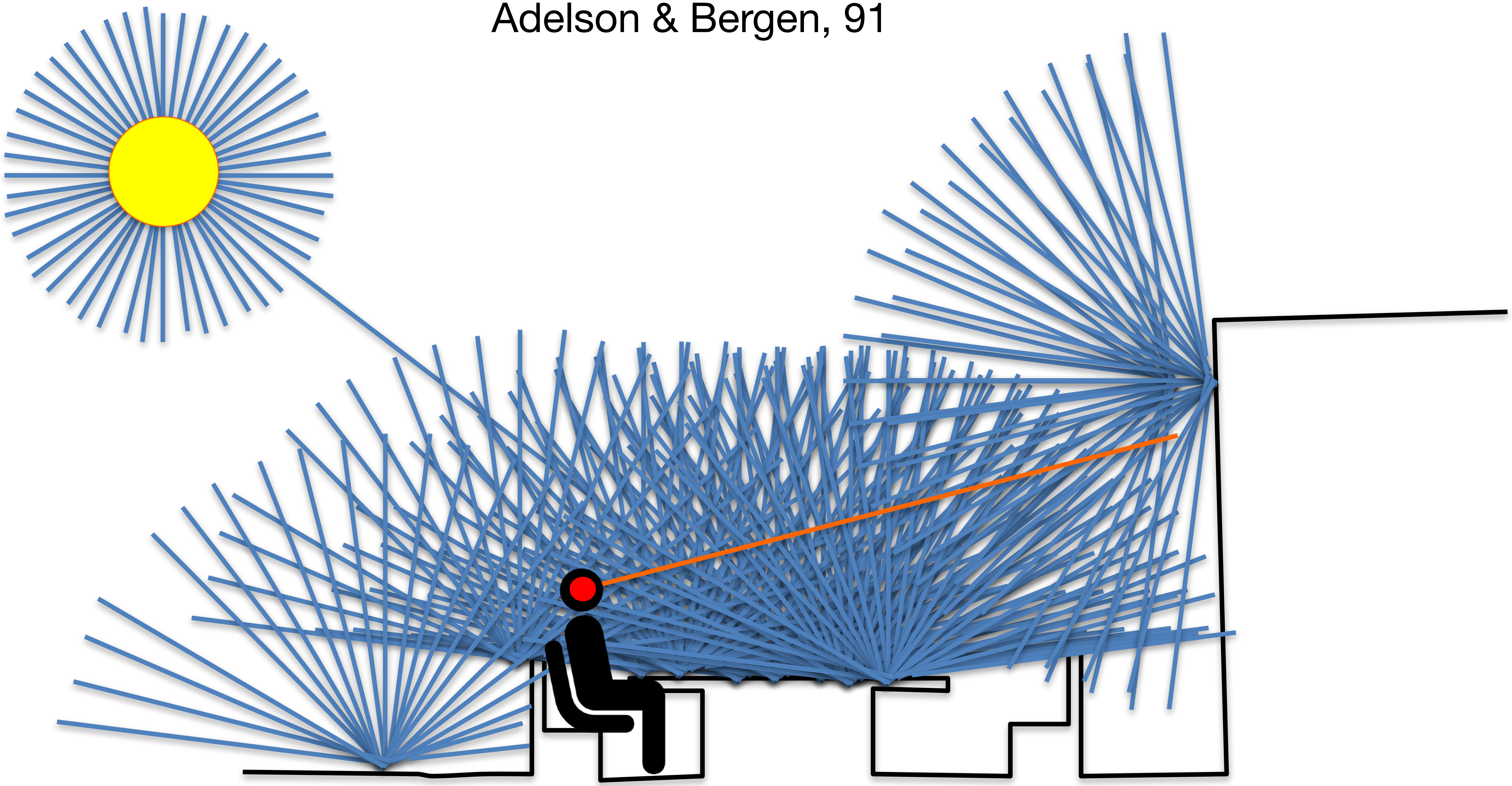





Source: Freeman, Torralba, Isola

# The Plenoptic Function

Adelson & Bergen, 91



The intensity P can be parameterized as:

P (                X, Y, Z )

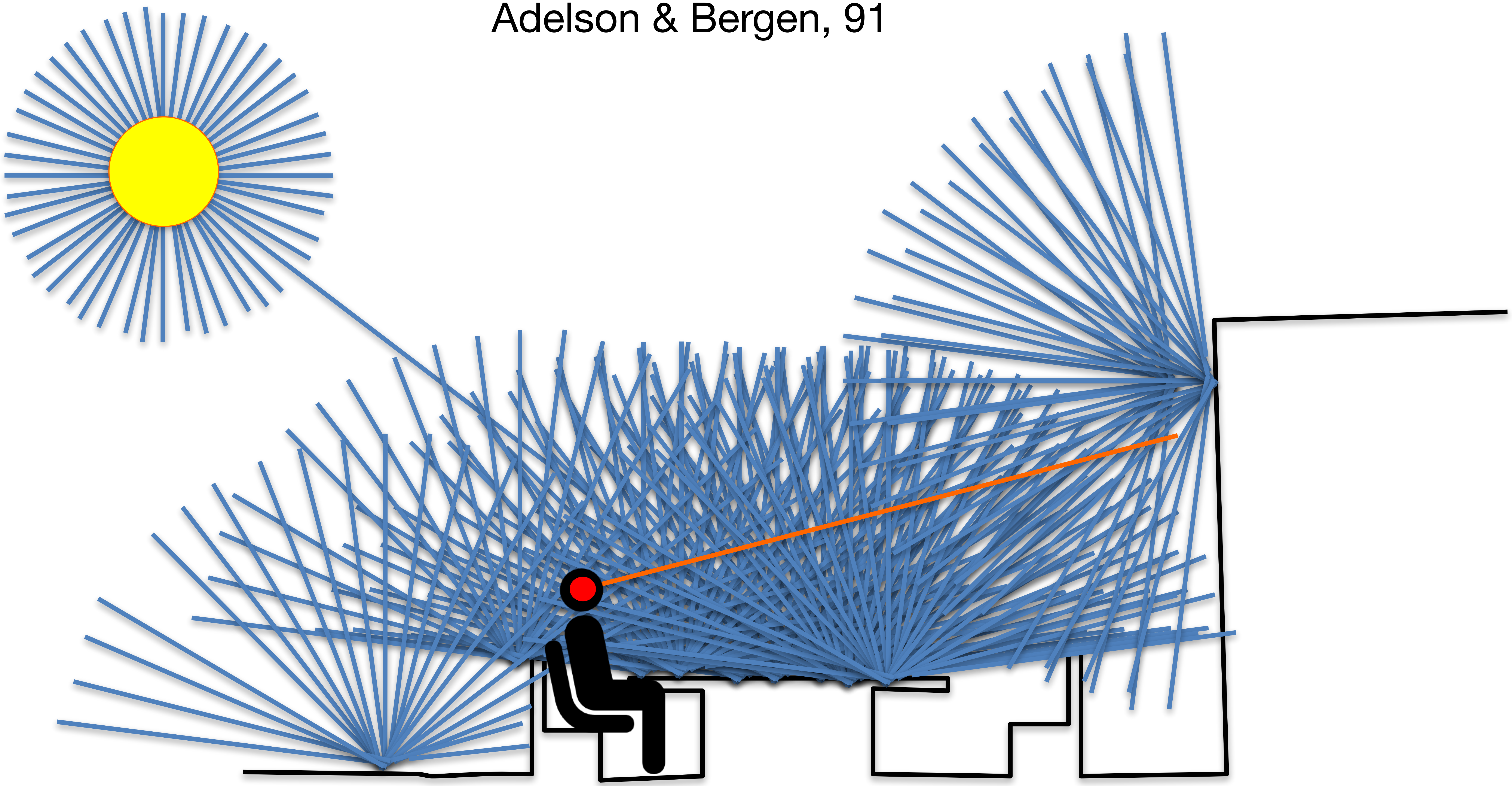
Eye position



Source: Freeman, Torralba, Isola

# The Plenoptic Function

Adelson & Bergen, 91



The intensity P can be parameterized as:

P (θ, ϕ, X, Y, Z)

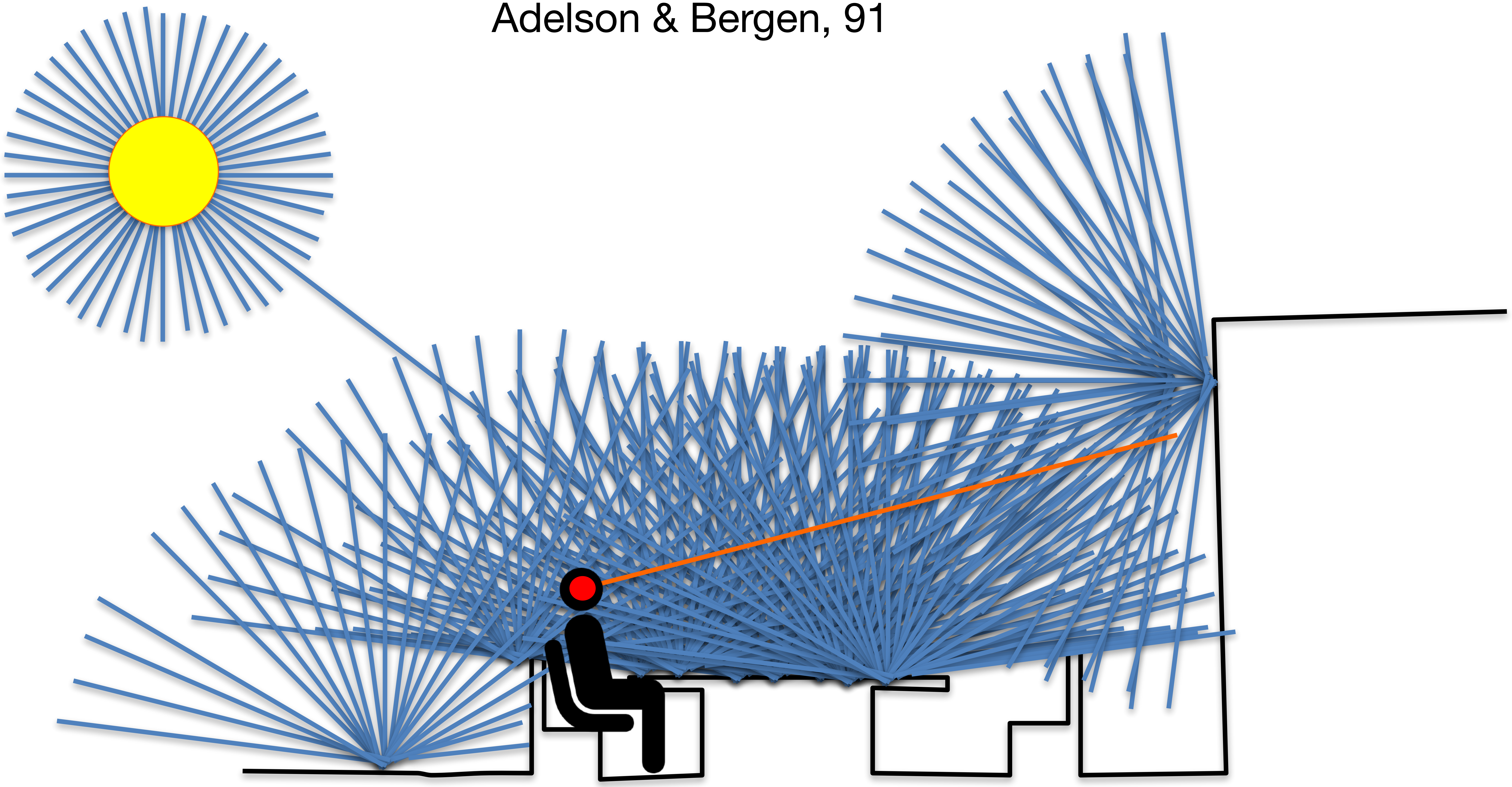
Angle



Source: Freeman, Torralba, Isola

# The Plenoptic Function

Adelson & Bergen, 91



The intensity P can be parameterized as:

$$P\,(\theta, \phi, \lambda, t, X, Y, Z)$$

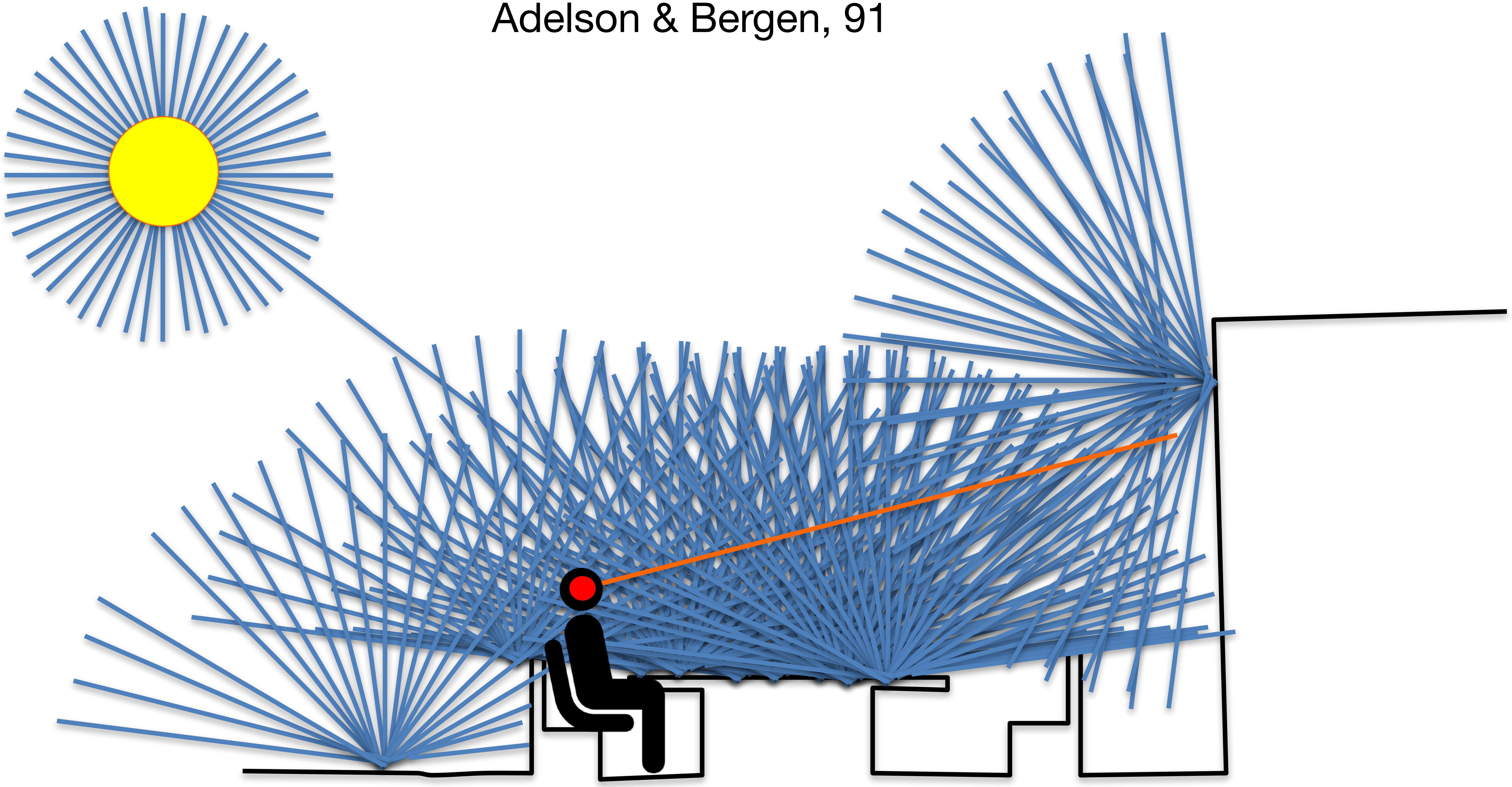
Wavelength, time



Source: Freeman, Torralba, Isola

# The Plenoptic Function

Adelson & Bergen, 91



The intensity P can be parameterized as:

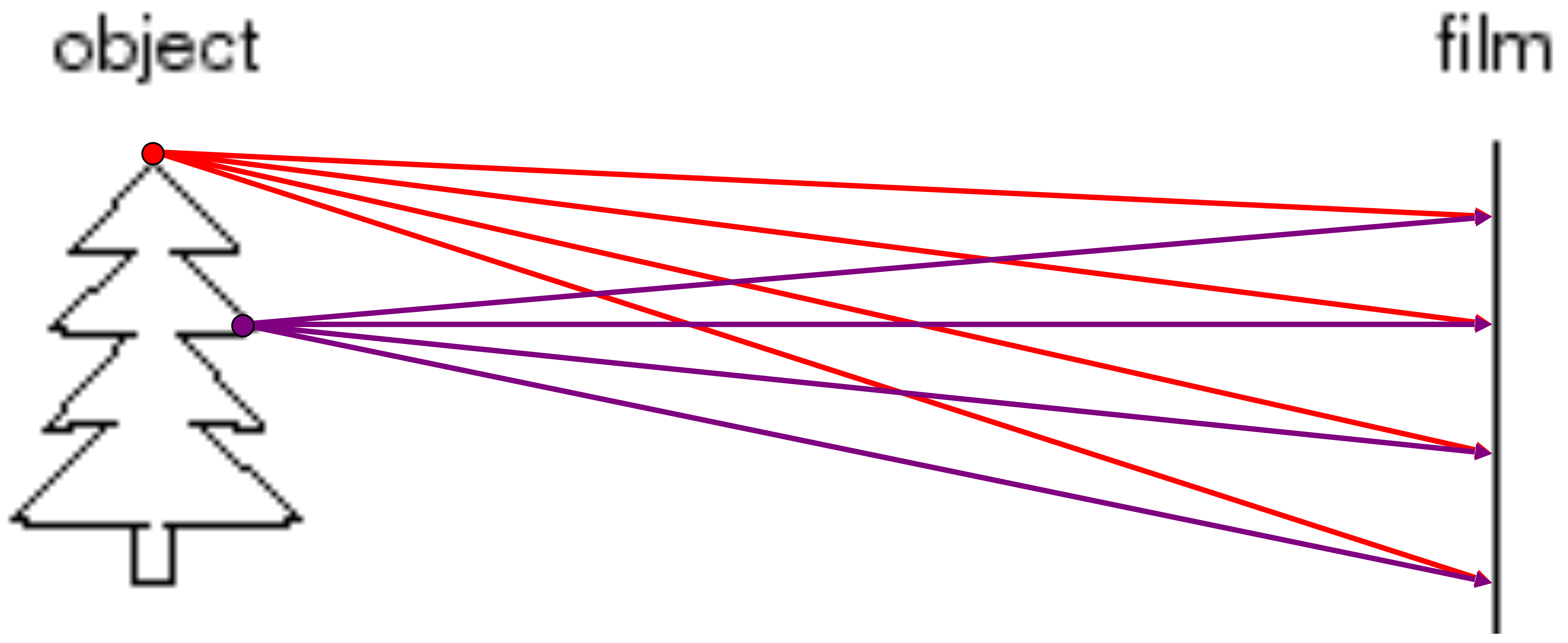
$$P(\theta, \phi, \lambda, t, X, Y, Z)$$

Full plenoptic function



Source: Freeman, Torralba, Isola

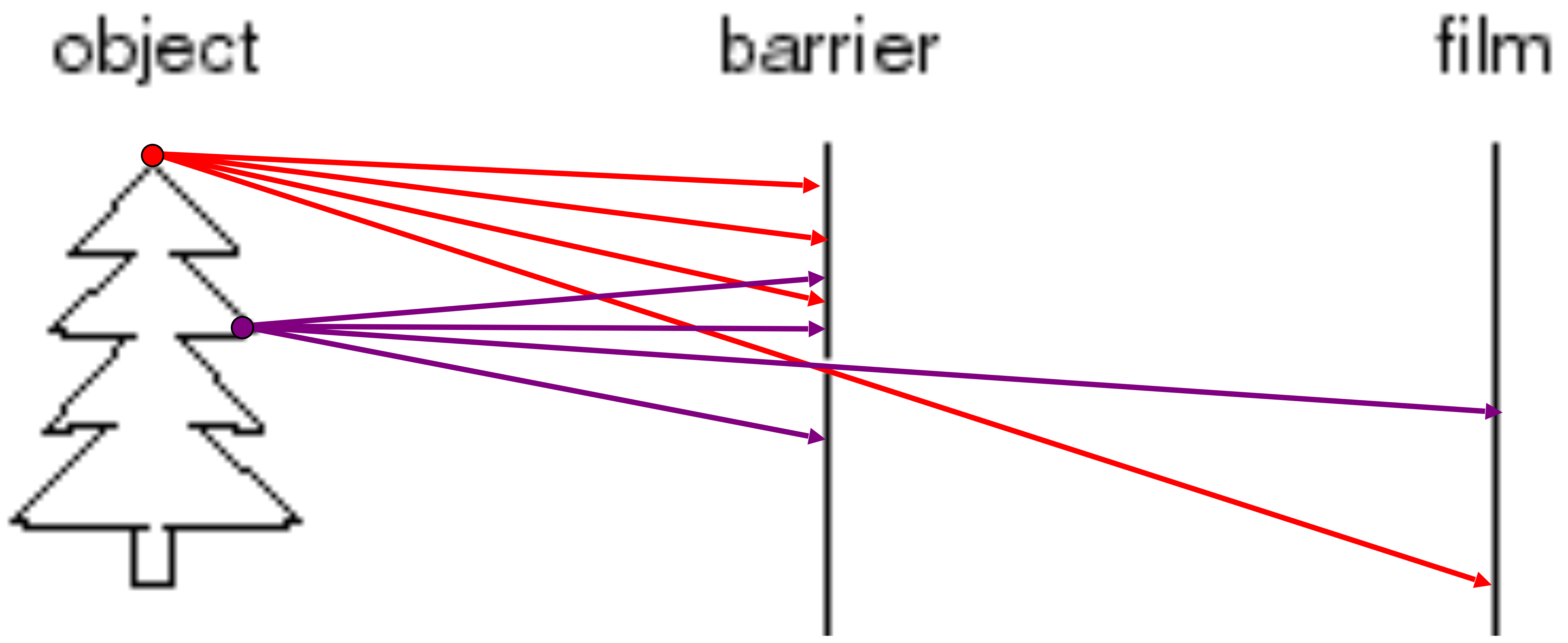
# Making a camera




Idea #1: put a piece of film in front of an object.

Source: N. Snavely

# Pinhole camera

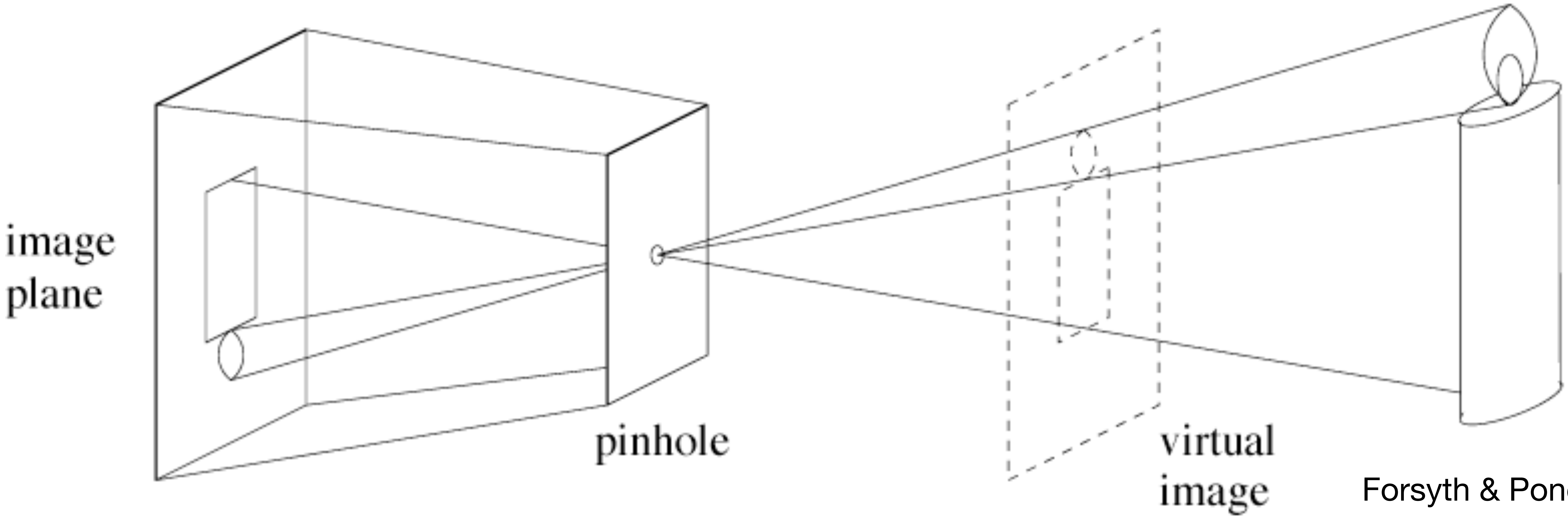



Add a barrier to block off most of the rays

- This reduces blurring
- The opening known as the **aperture**

Source: N. Snavely

# Upside down images




Forsyth & Ponce

Useful concept: virtual image



Source: Freeman, Torralba, Isola

# Pinhole camera

Photograph by Abelardo Morell, 1991



Source: Freeman, Torralba, Isola

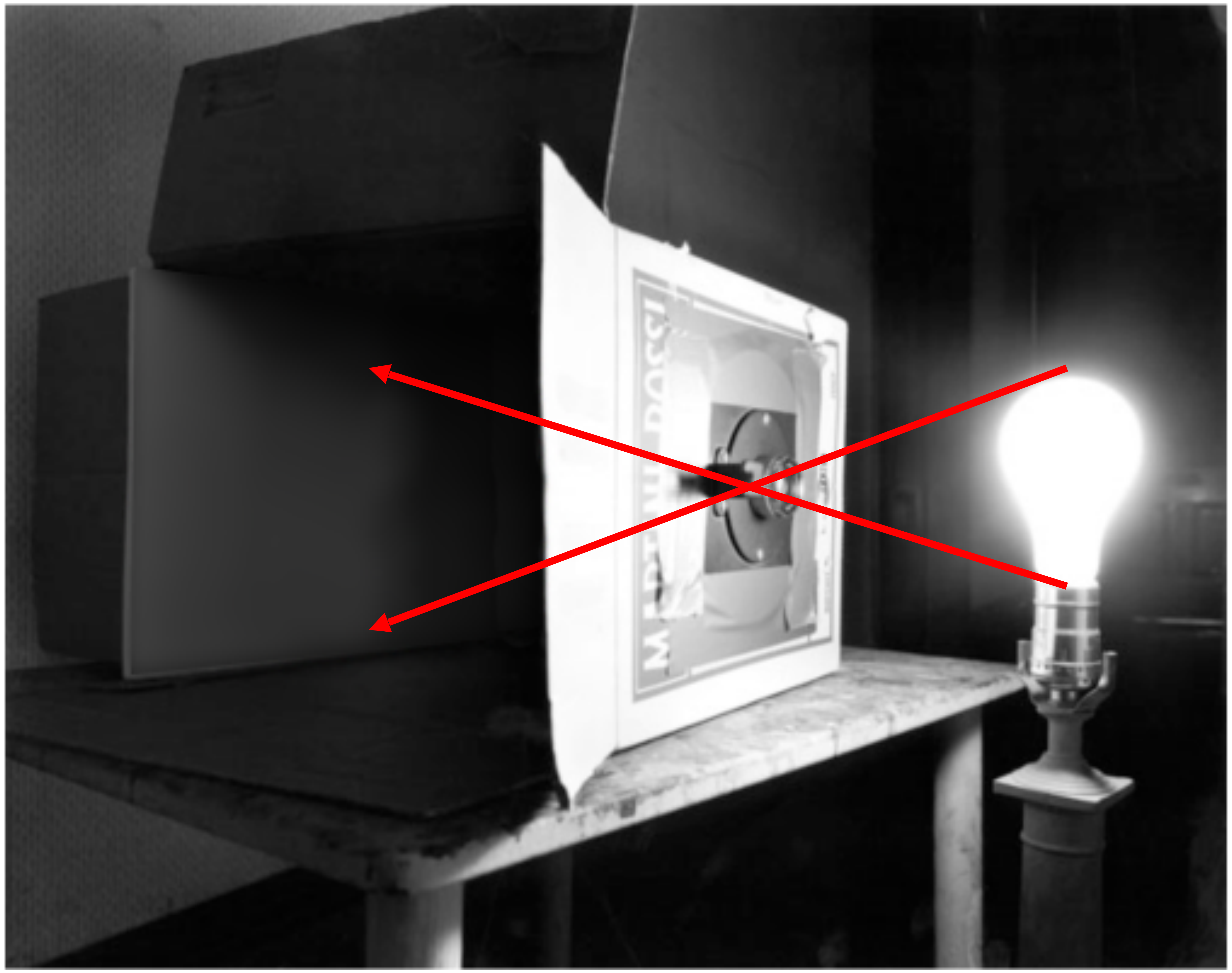
# Pinhole camera

Photograph by Abelardo Morell, 1991



Source: Freeman, Torralba, Isola

# Pinhole camera

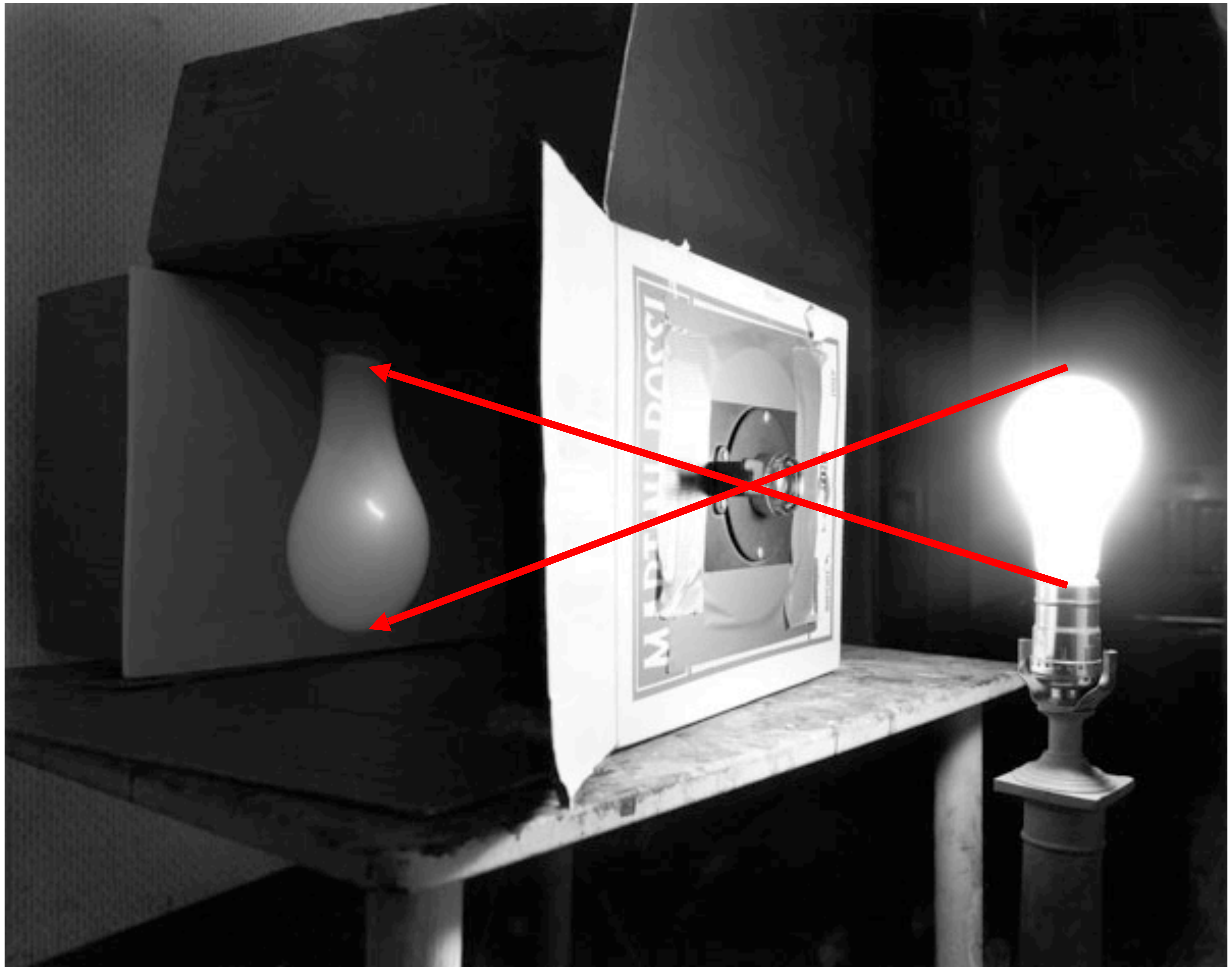


Photograph by Abelardo Morell, 1991



Source: Freeman, Torralba, Isola

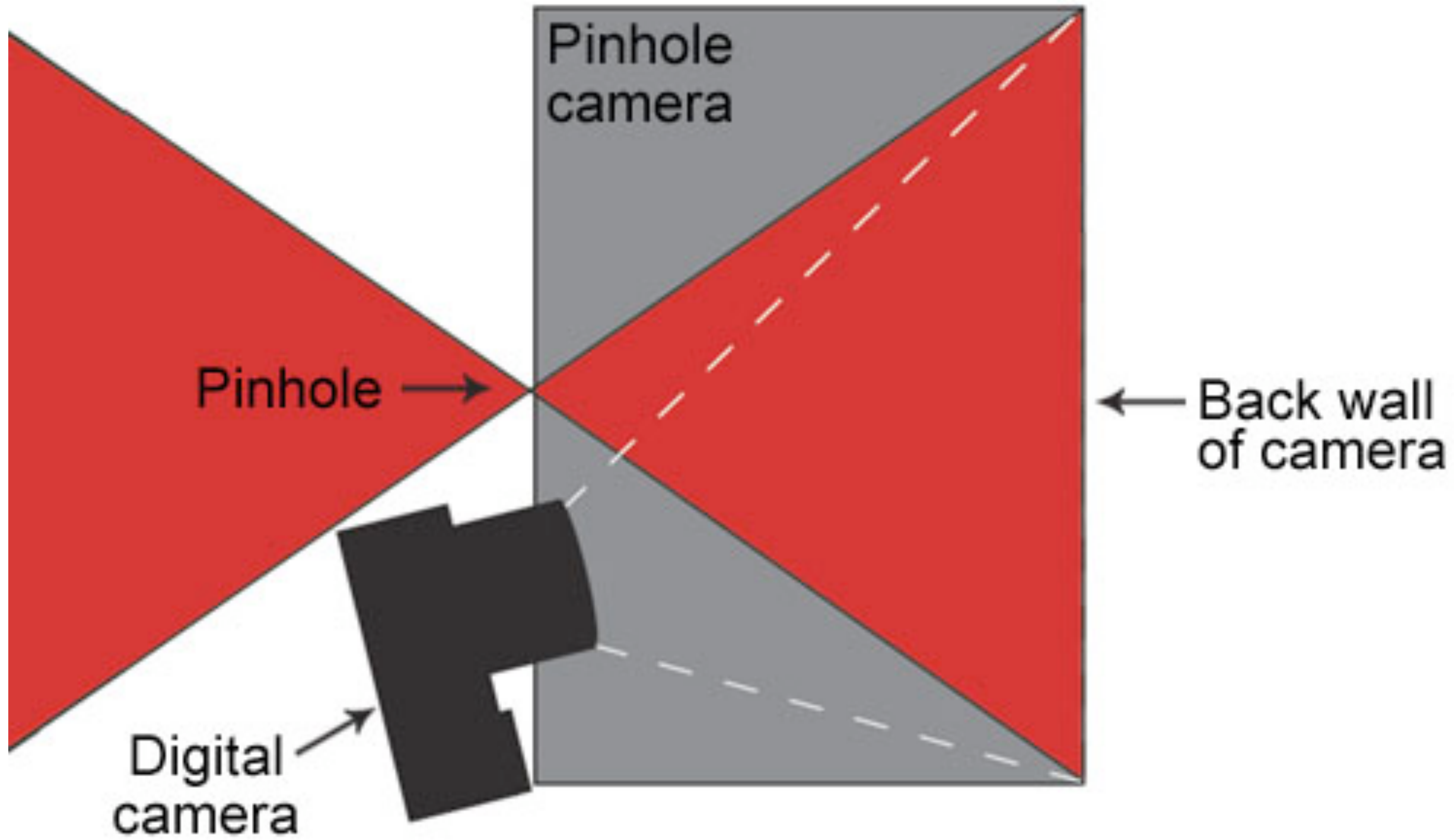
# Pinhole camera



Photograph by Abelardo Morell, 1991



Source: Freeman, Torralba, Isola



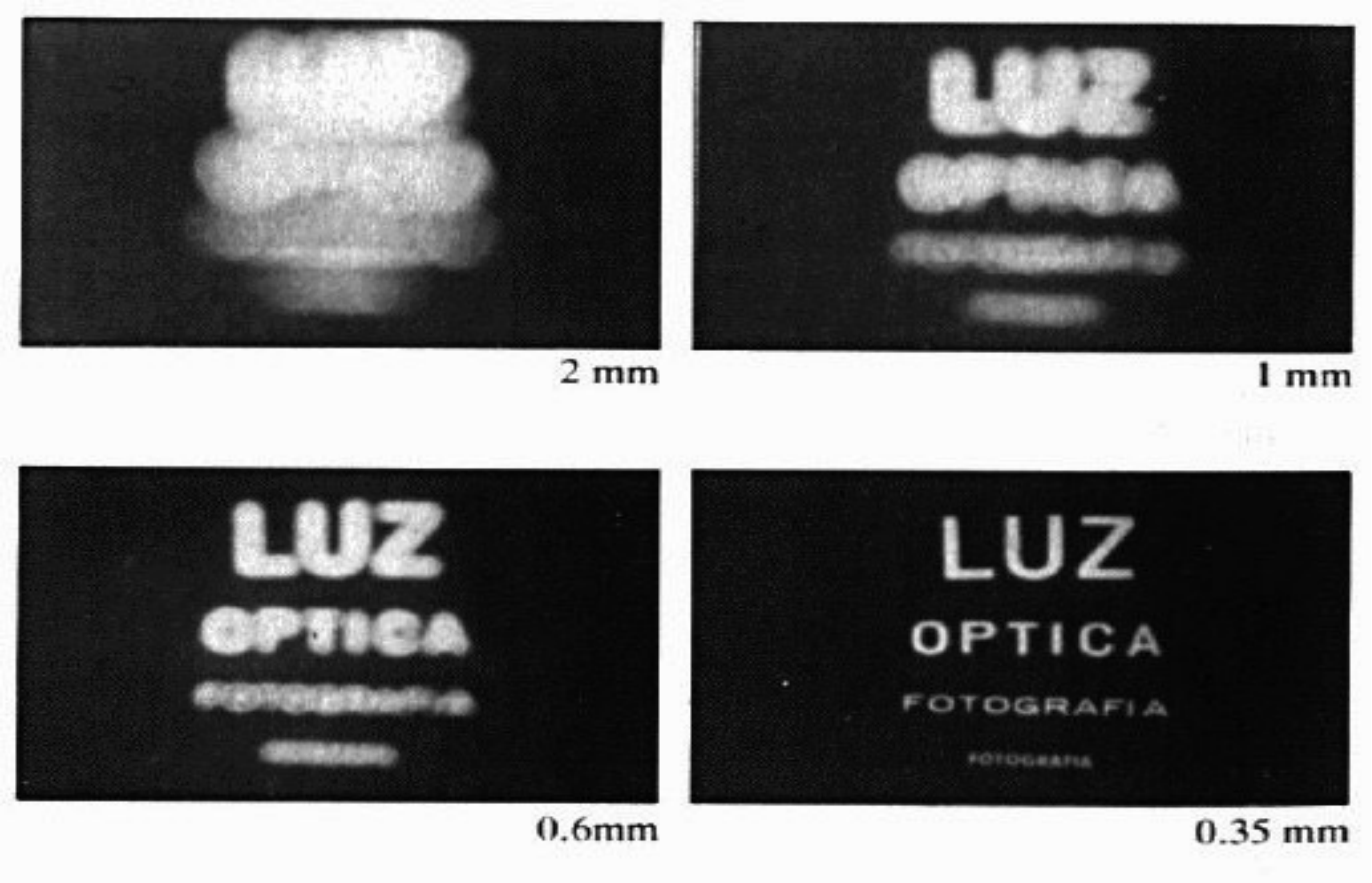

http://www.foundphotography.com/PhotoThoughts/archives/2005/04/pinhole_camera_2.html



Source: Freeman, Torralba, Isola


25

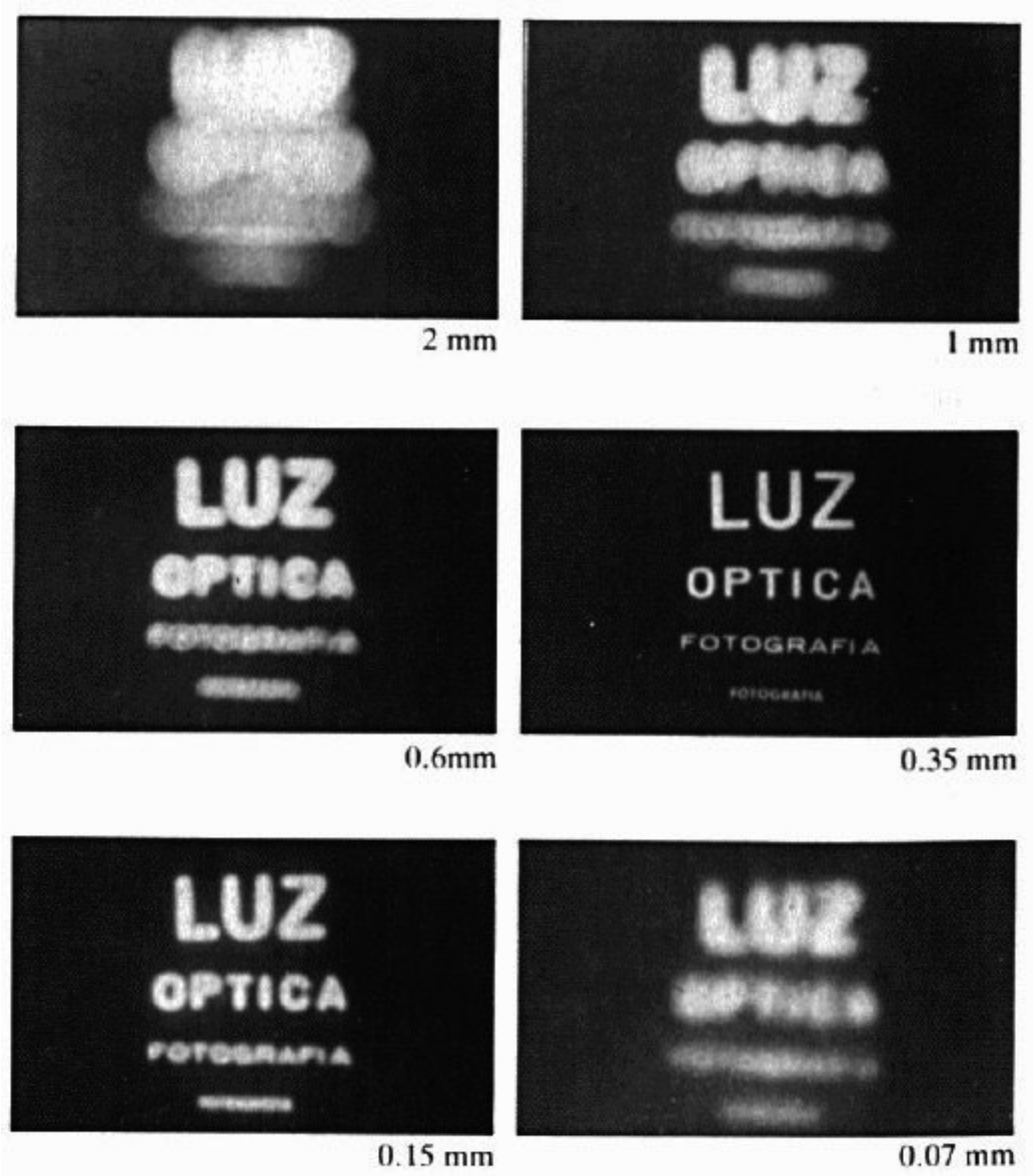
# Shrinking the aperture




- Why not make the aperture as small as possible?
  - Less light gets through
  - *Diffraction* effects...

Source: N. Snavely

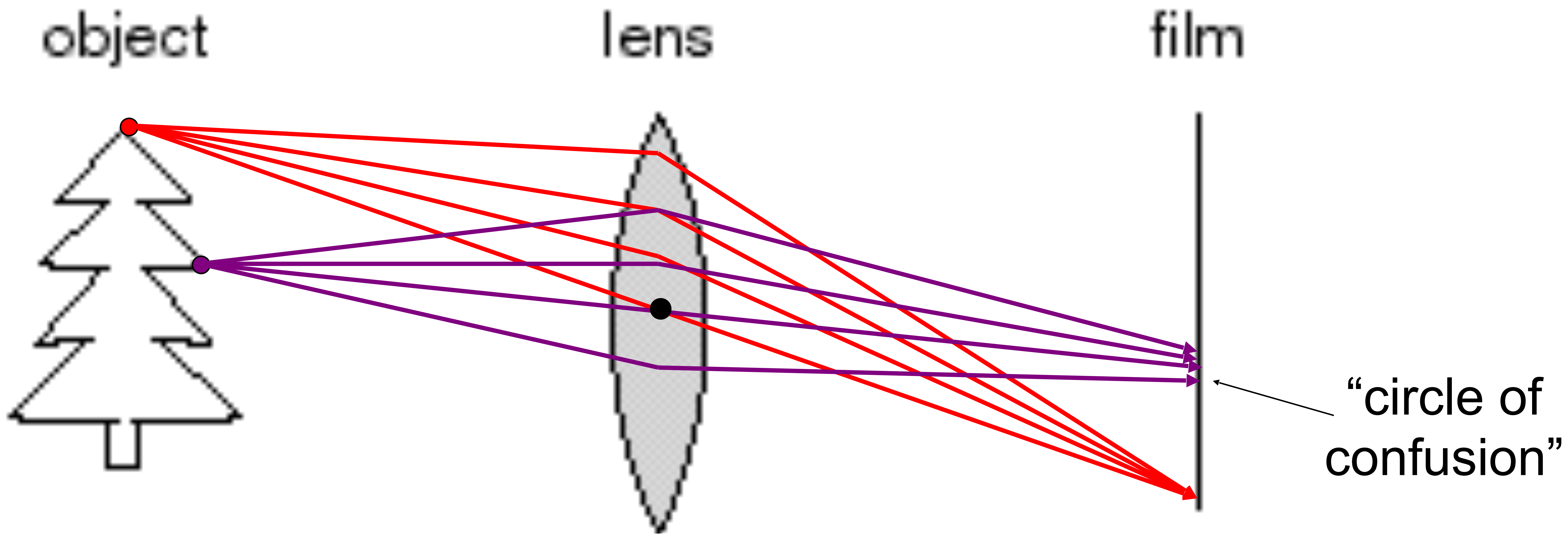
# Shrinking the aperture






Source: N. Snavely

# Adding a lens



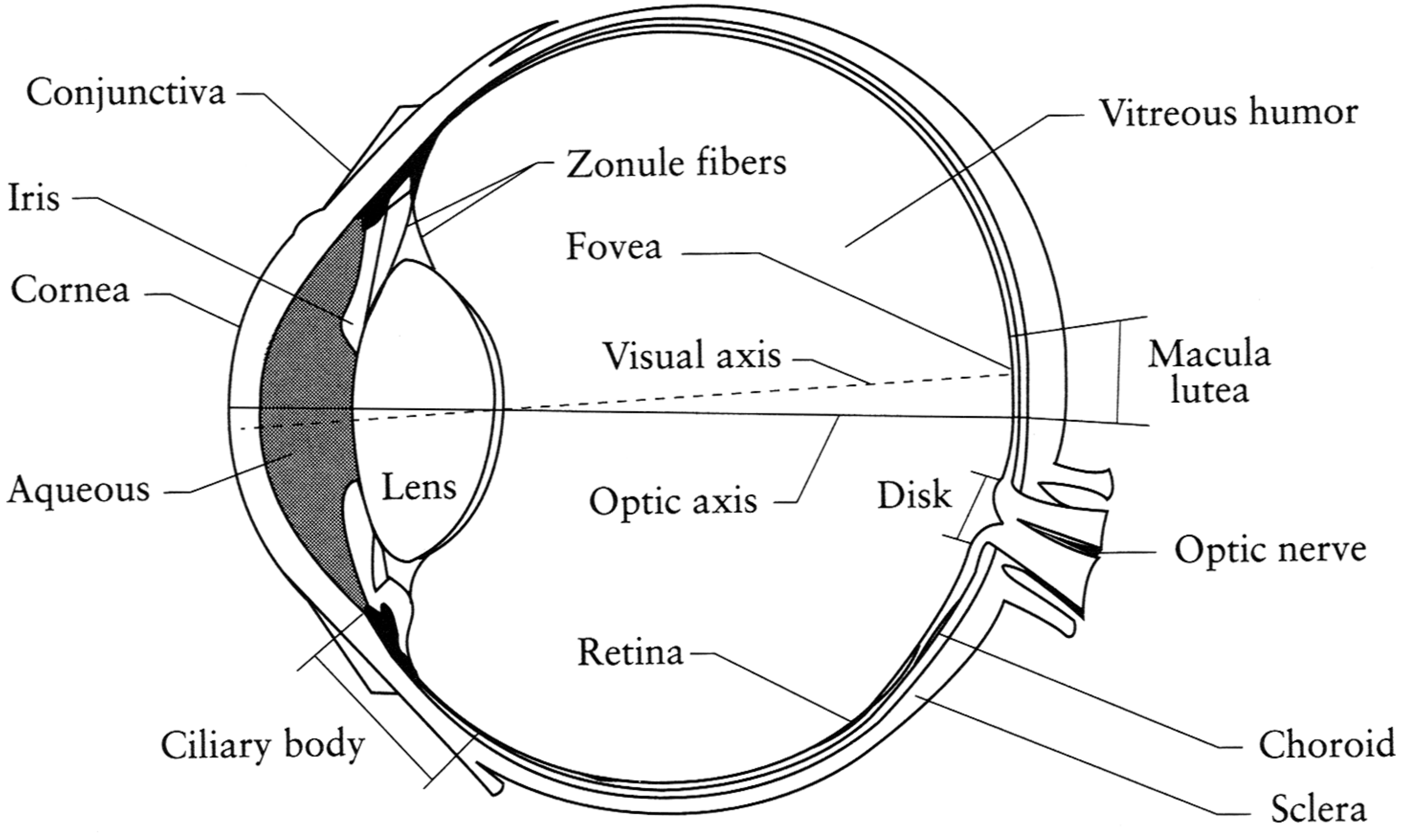

A lens focuses light onto the film

- There is a specific distance at which objects are "in focus"
  - other points project to a "circle of confusion" in the image
- Changing the shape of the lens changes this distance



Source: N. Snavely

# The eye




The human eye is a camera

- **Iris -** colored annulus with radial muscles
- **Pupil -** the hole (aperture) whose size is controlled by the iris
- What's the "film"?
  - Photoreceptor cells (rods and cones) in the **retina**

Source: N. Snavely

# Eyes in nature: eyespots to pinhole camera

http://upload.wikimedia.org/wikipedia/commons/6/6d/Mantis_shrimp.jpg



Source: Freeman, Torralba, Isola

# Pinhole cameras in unexpected places

Tree shadow during a solar eclipse

photo credit: Nils van der Burg

http://www.physicstogo.org/index.cfm

Source: Steve Seitz




Source: Freeman, Torralba, Isola



Source: Freeman, Torralba, Isola



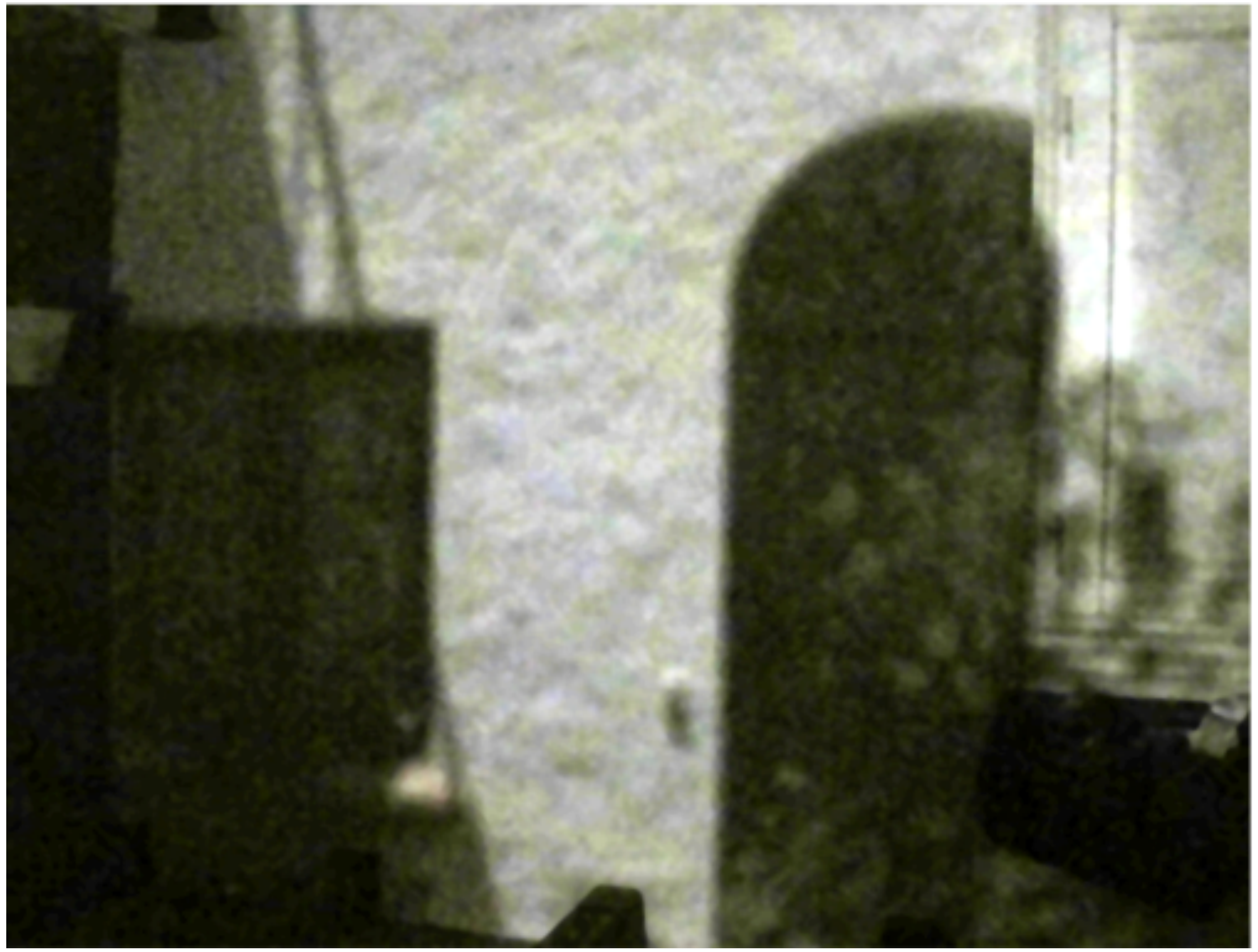
Source: Freeman, Torralba, Isola

# Accidental pinhole camera



Source: Freeman, Torralba, Isola

Window turned into a pinhole



View outside



Source: Freeman, Torralba, Isola

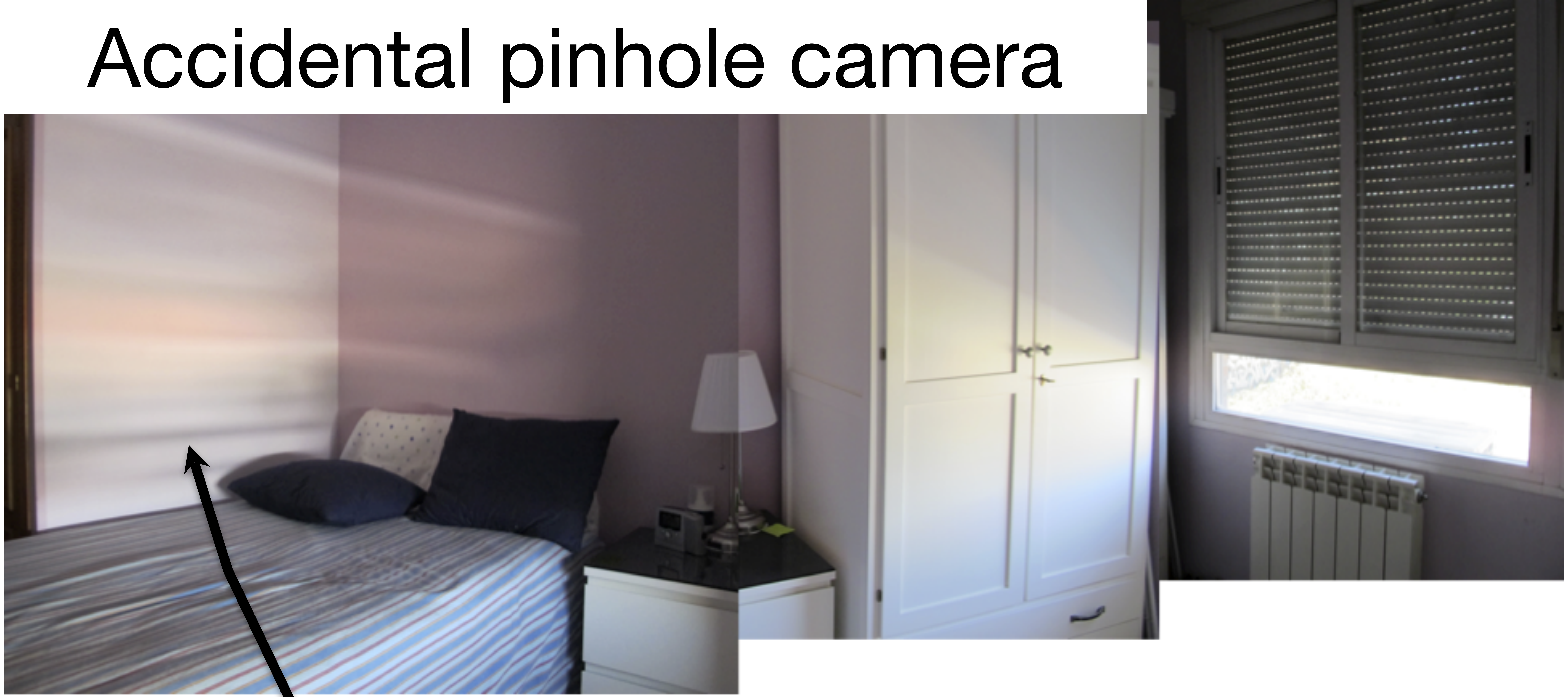
Window open

Window turned into a pinhole



Source: Freeman, Torralba, Isola

# Accidental pinhole camera



Outside scene

*

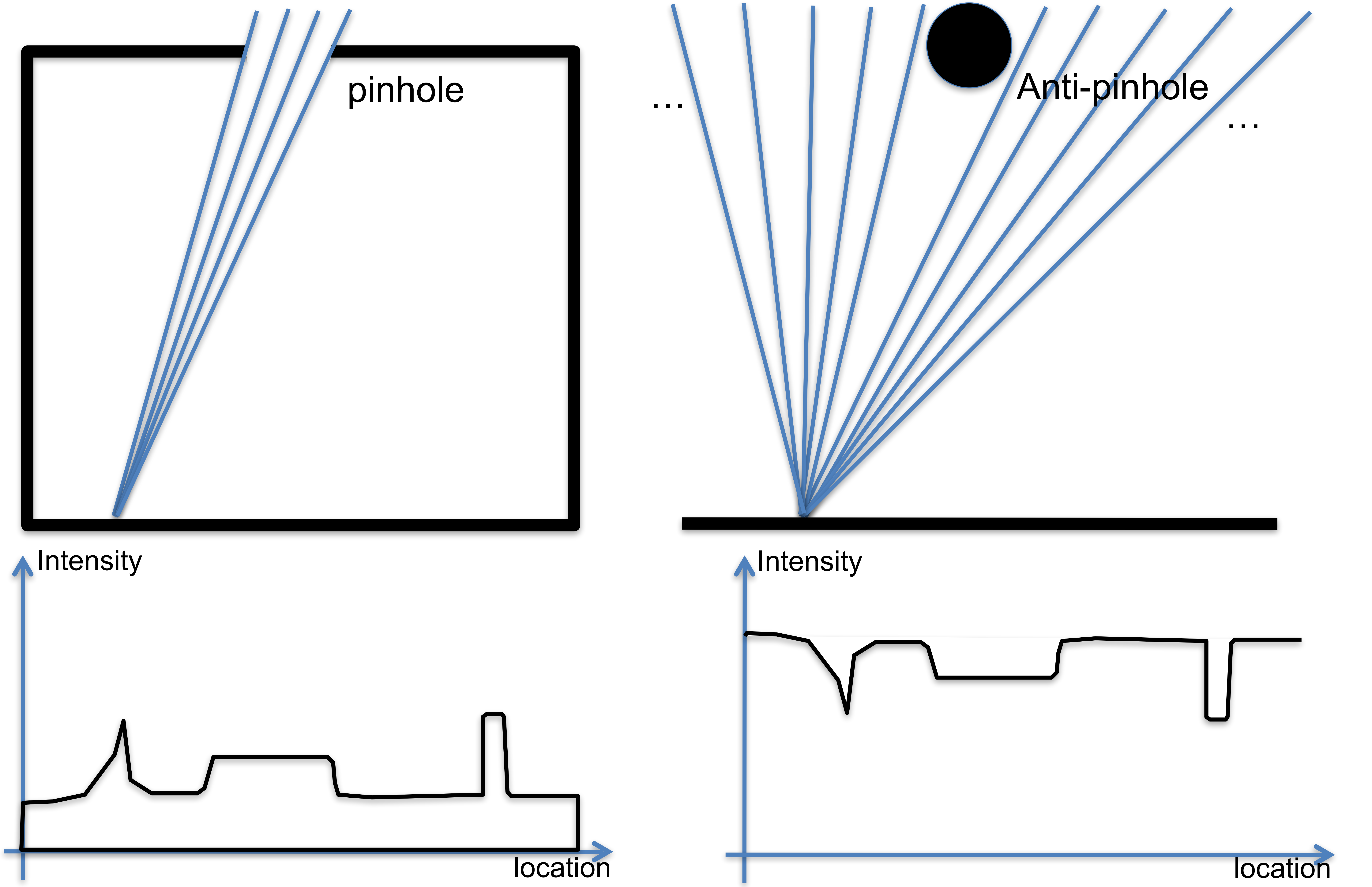
Aperture

See Zomet, A.; Nayar, S.K. CVPR 2006 for a detailed analysis.



Source: Freeman, Torralba, Isola

# Pinhole and Anti-pinhole cameras






Adam L. Cohen, 1982

Source: Freeman, Torralba, Isola

# Mixed accidental pinhole and anti-pinhole cameras

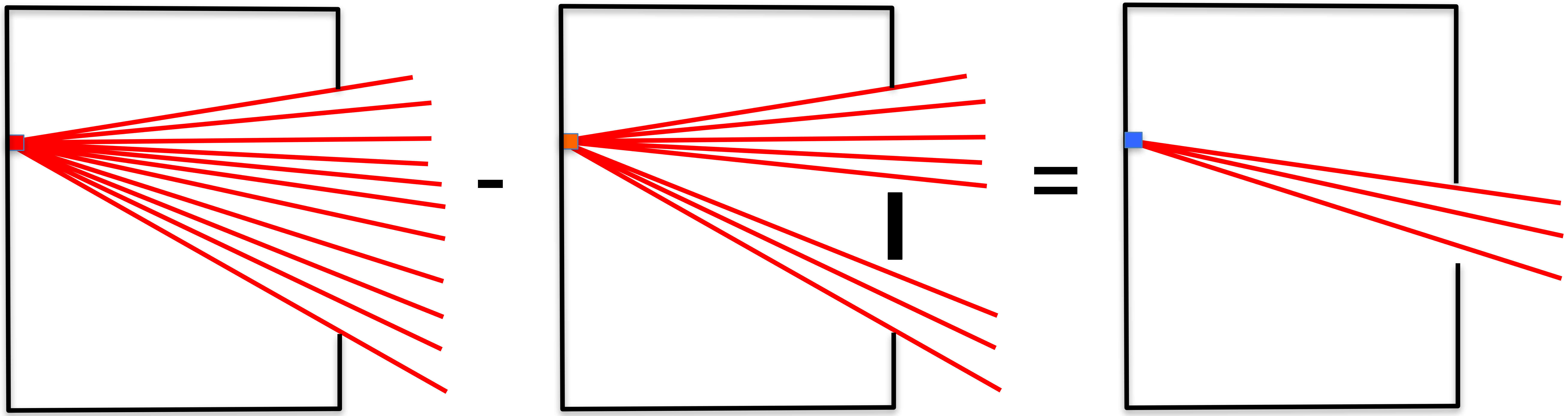
# Mixed accidental pinhole and anti-pinhole cameras



Source: Freeman, Torralba, Isola

# Mixed accidental pinhole and anti-pinhole cameras

Room with a window | Person in front of the window | Difference image



= ?



Source: Freeman, Torralba, Isola

# Mixed accidental pinhole and anti-pinhole cameras

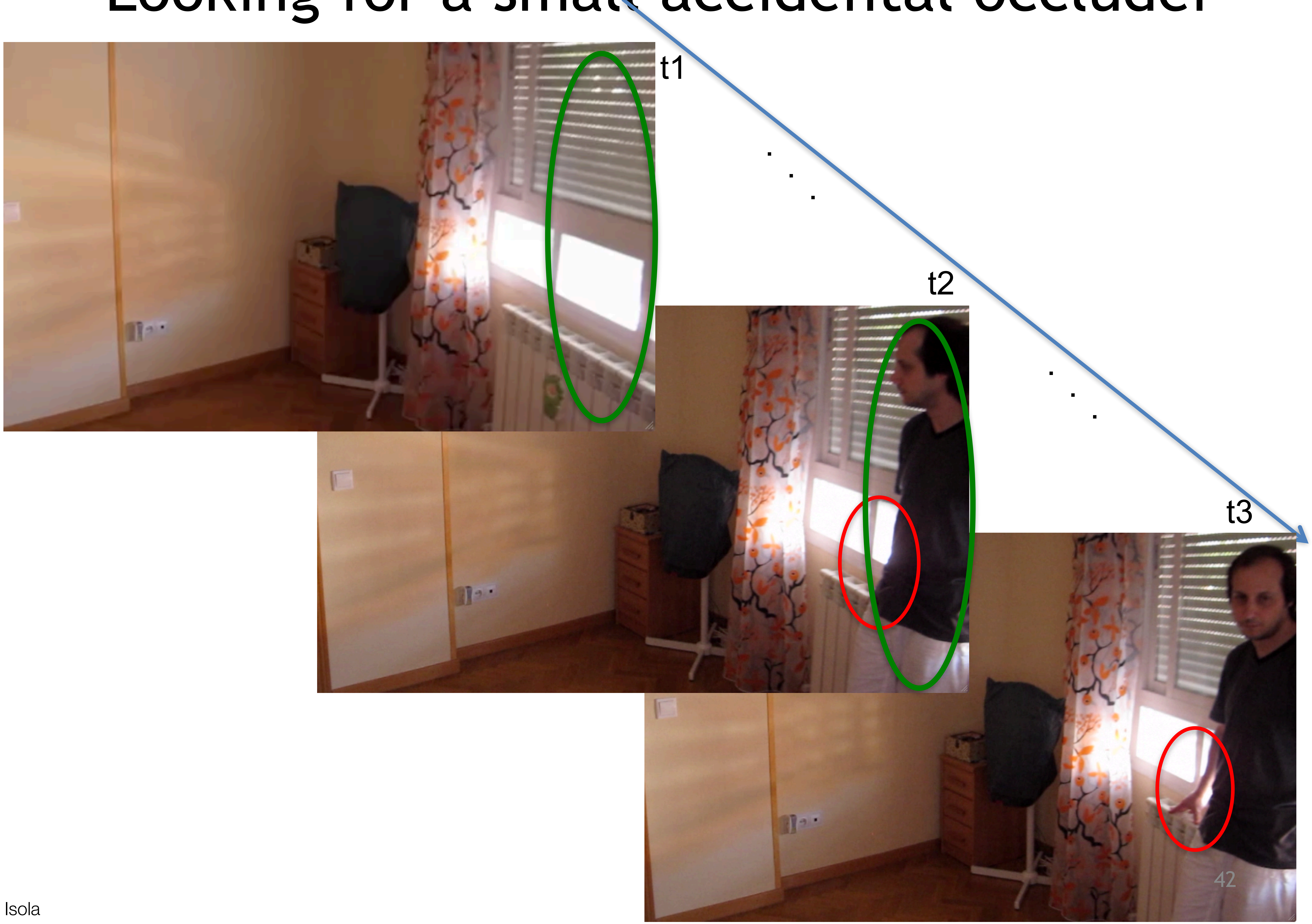
Body as the occluder

View outside the window



Source: Freeman, Torralba, Isola

# Looking for a small accidental occluder




Source: Freeman, Torralba, Isola

# Looking for a small accidental occluder

Body as the occluder

Hand as the occluder

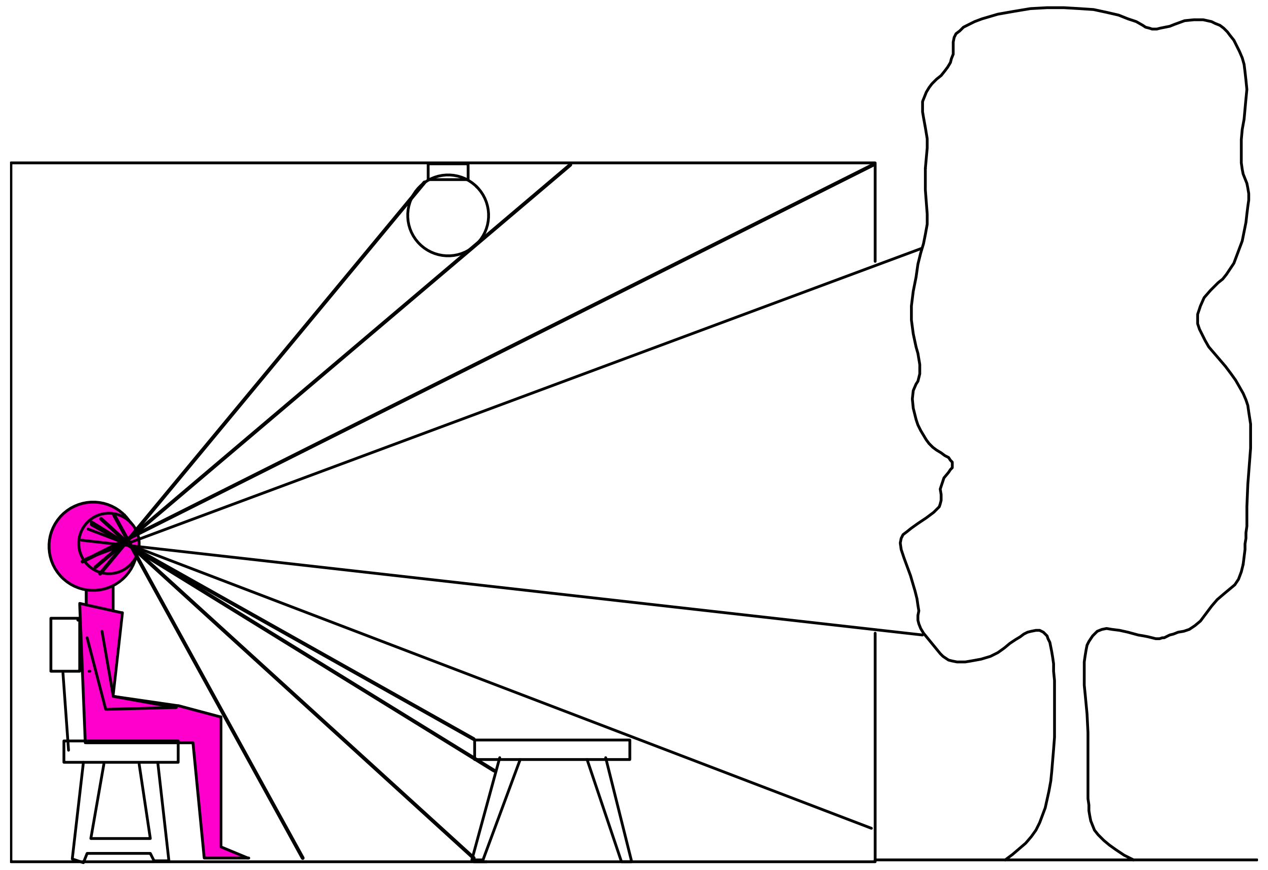
View outside the window



Source: Freeman, Torralba, Isola

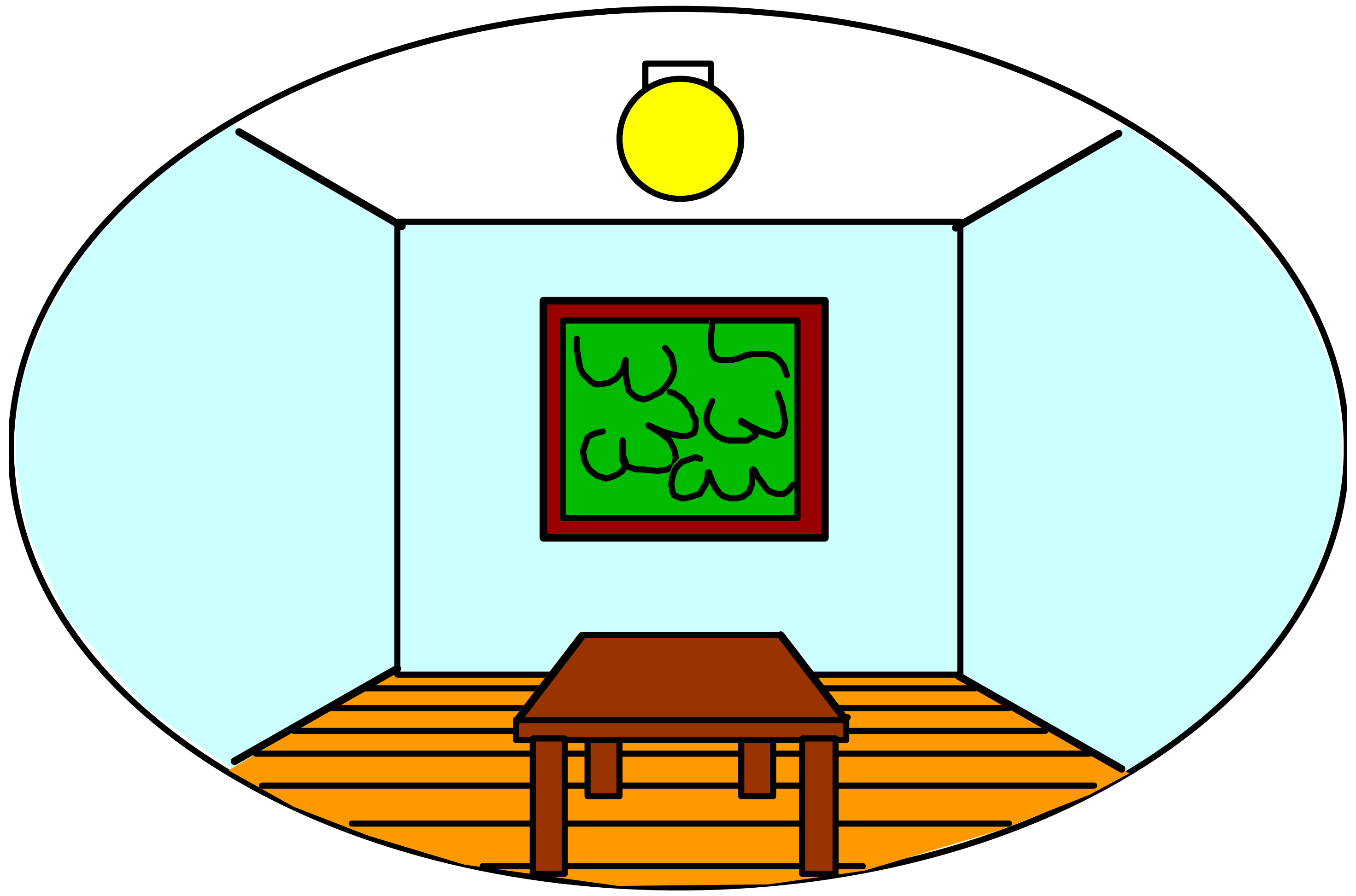
# Projection from 3D to 2D

*3D world*



Point of observation

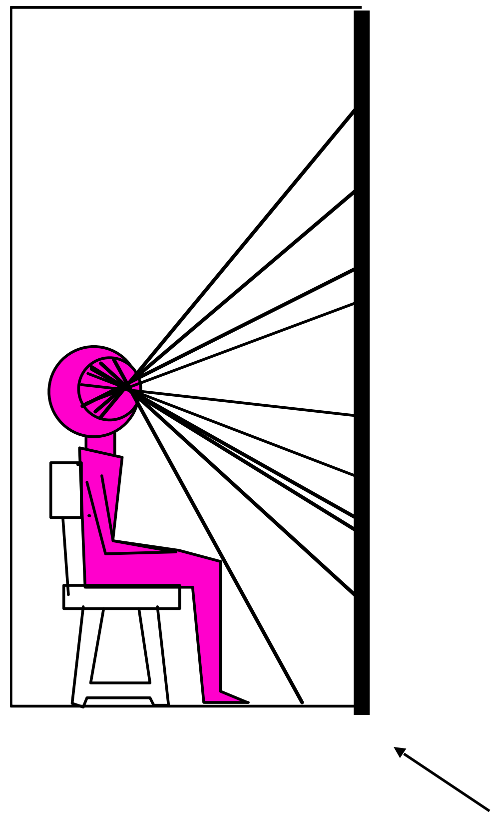
*2D image*



Slide by A. Efros
Figure source: Stephen E. Palmer, 2002

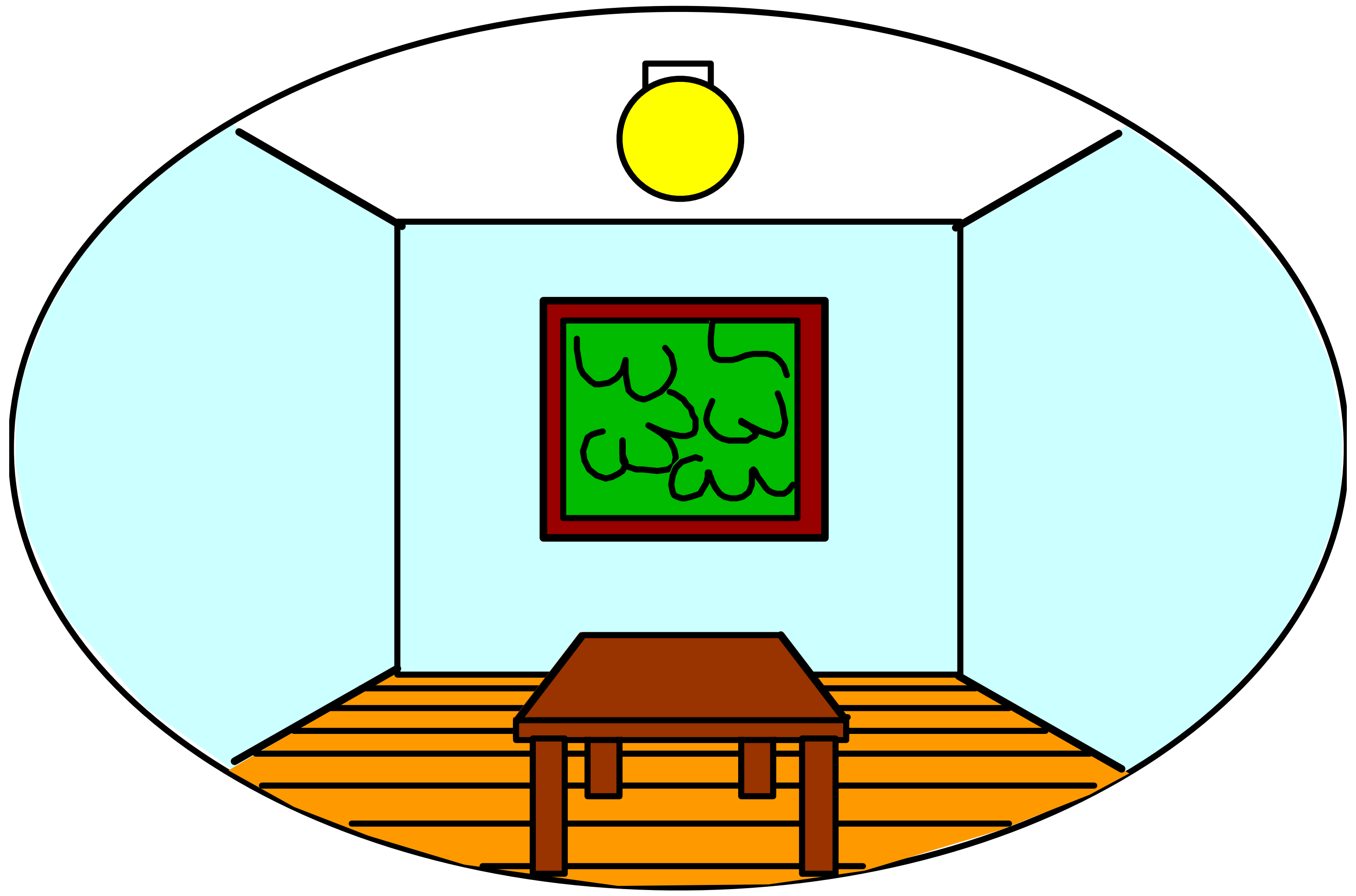
# Projection from 3D to 2D

*3D world*



Painted backdrop

*2D image*





Source: S. Lazebnik

# Fooling the eye



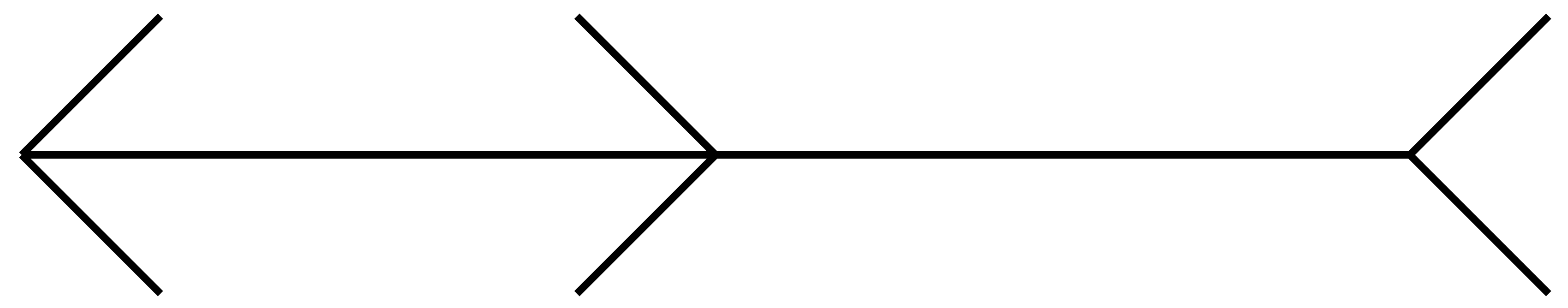

Source: S. Lazebnik

# Fooling the eye


Making of 3D sidewalk art: http://www.youtube.com/watch?v=3SNYtd0Ayt0

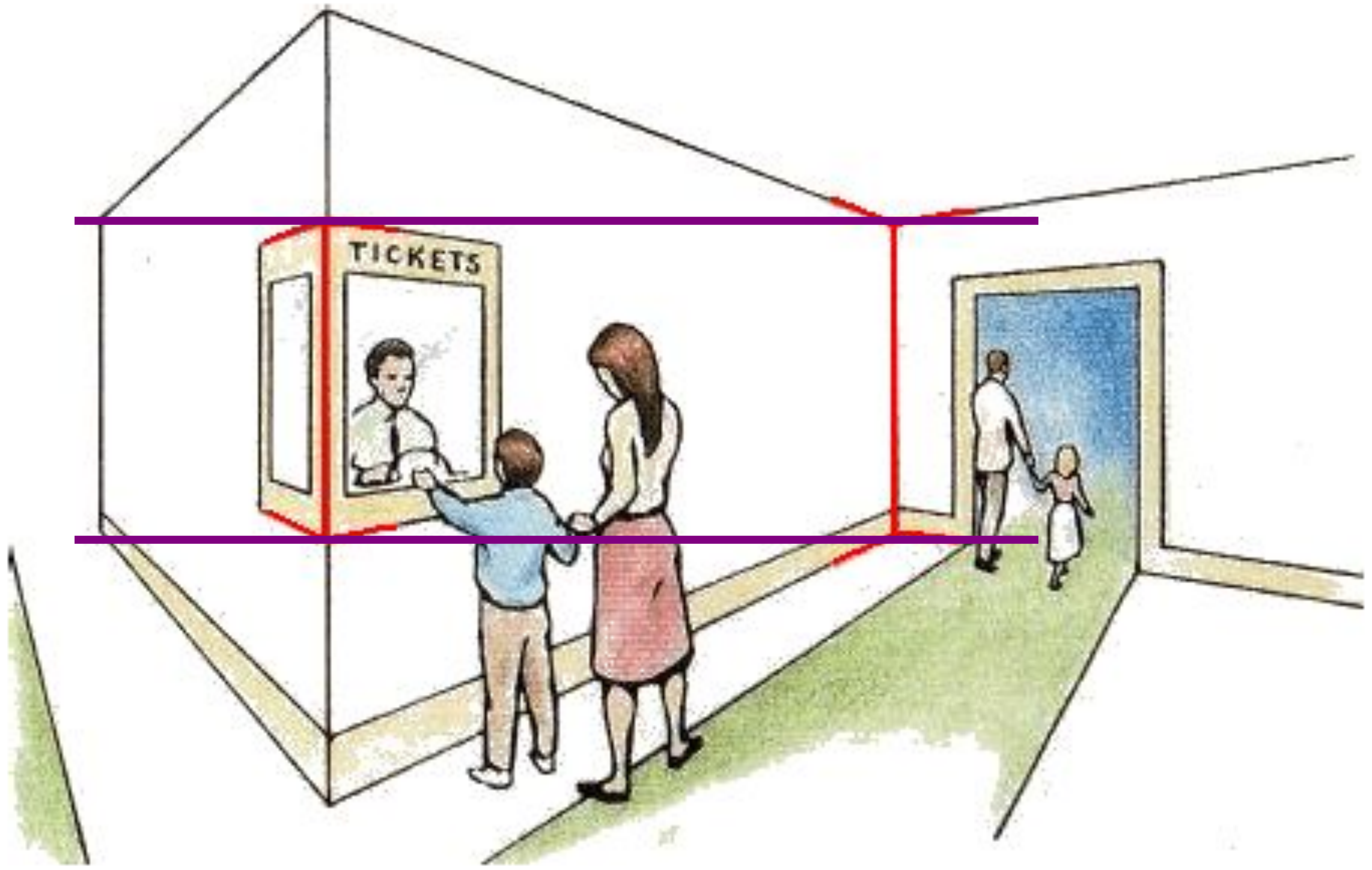
Source: S. Lazebnik

# Müller-Lyer Illusion

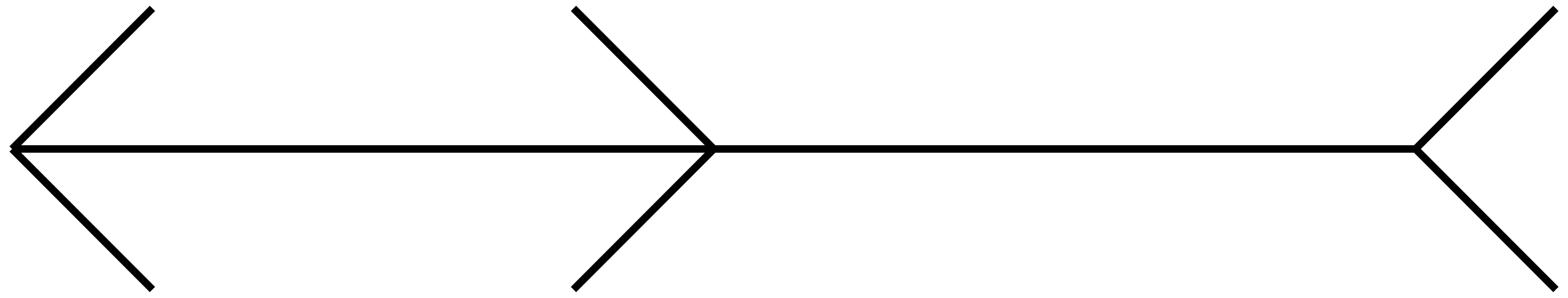


Source: N. Snavely, S. Lazebnik

# Müller-Lyer Illusion



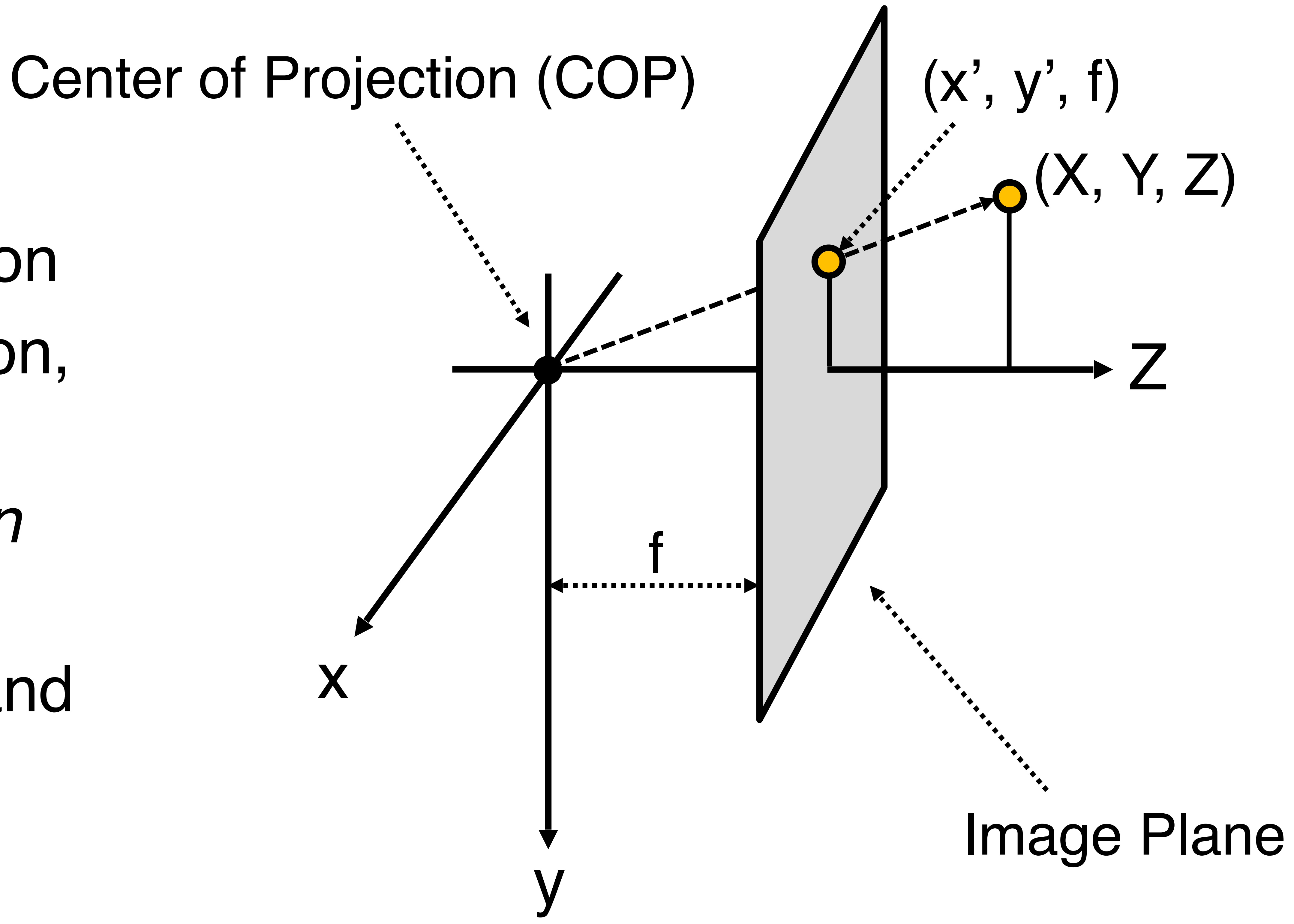

http://www.michaelbach.de/ot/sze_muelue/index.html



Source: N. Snavely, S. Lazebnik

# Modeling projection

- The coordinate system
  - We use the pinhole model as an approximation
  - Put the optical center (aka Center of Projection, or COP) at the origin
  - Put the Image Plane (aka Projection Plane) *in front* of the COP
  - The camera looks down the *positive* z-axis, and the y-axis points down




Source: N. Snavely

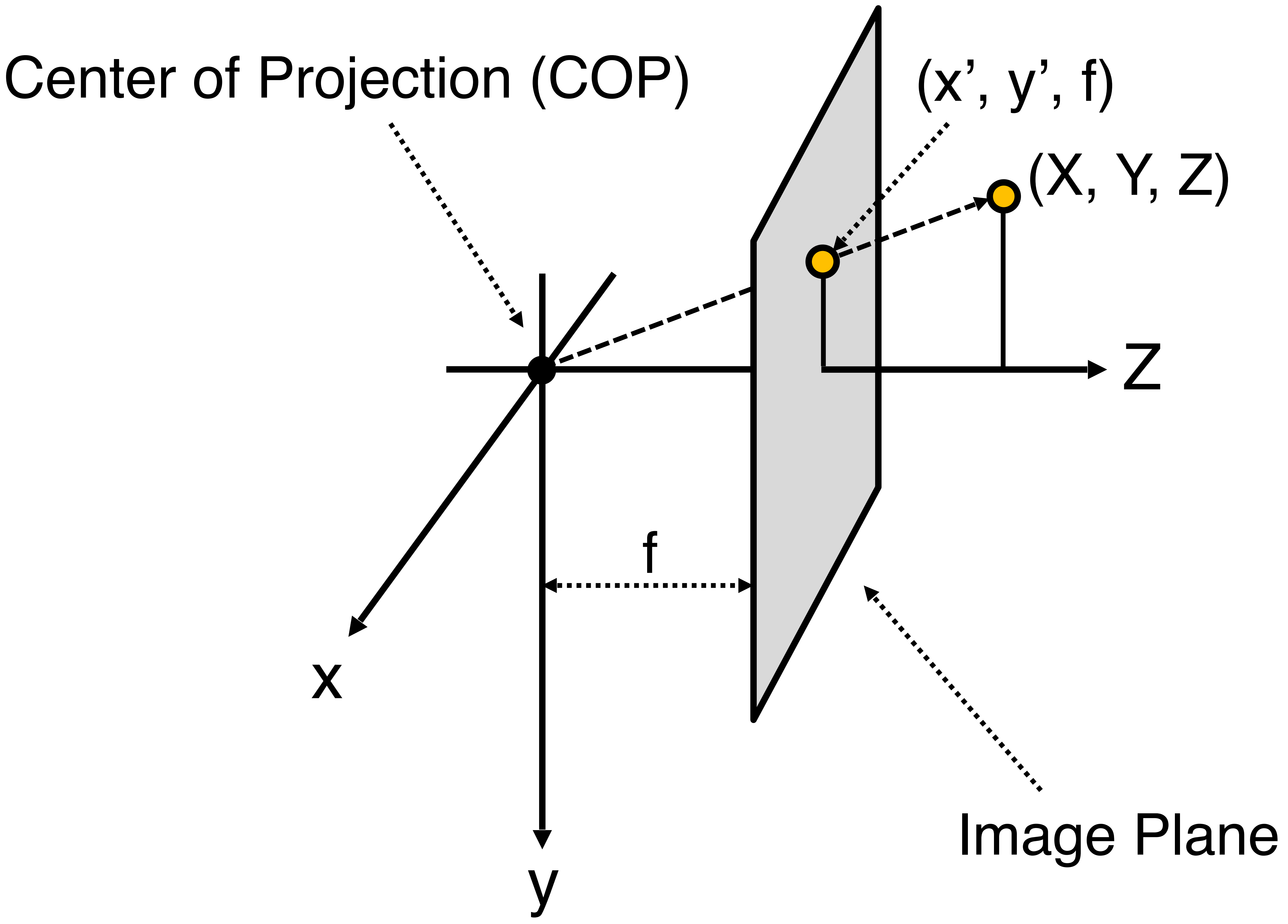
# Modeling projection

- Projection equations
  - Compute intersection with image plane of ray from (X,Y,Z) to COP
  - Derived using similar triangles

$$(x, y, z) \rightarrow (f\frac{x}{z}, f\frac{y}{z}, f)$$

  - We get the projection by throwing out the last coordinate:

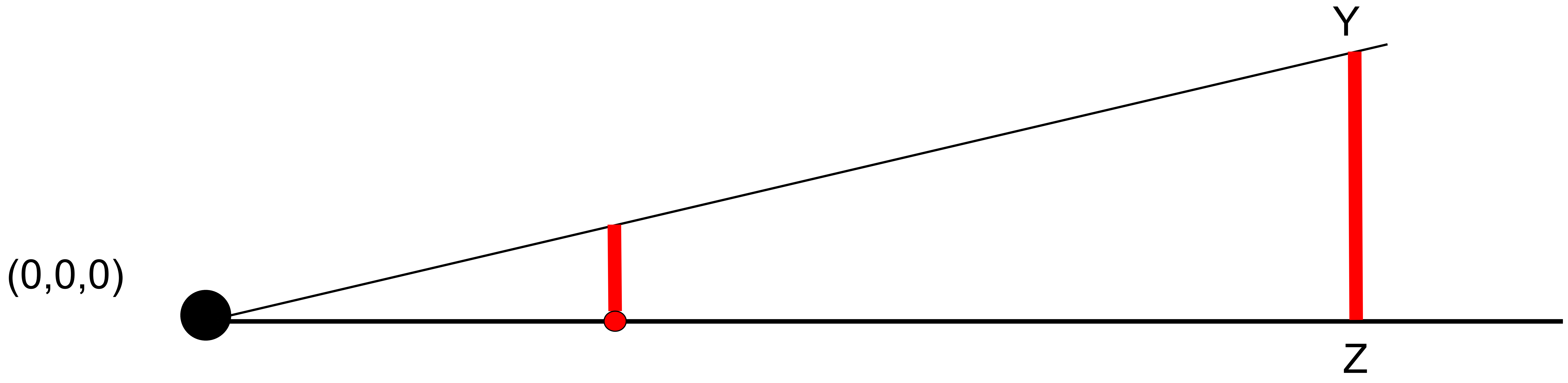
$$(x, y, z) \rightarrow (f\frac{x}{z}, f\frac{y}{z})$$






Source: N. Snavely

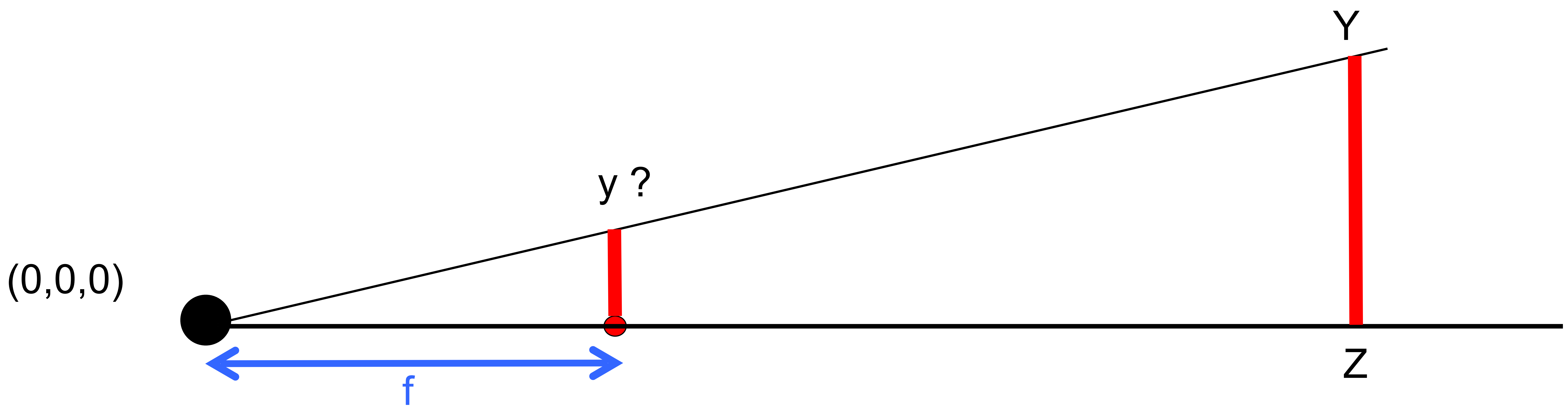
# Perspective projection






Source: Freeman, Torralba, Isola

# Perspective projection




Similar triangles: $y / f = Y / Z$

$$y = f\, Y/Z$$

How can we represent this more compactly?



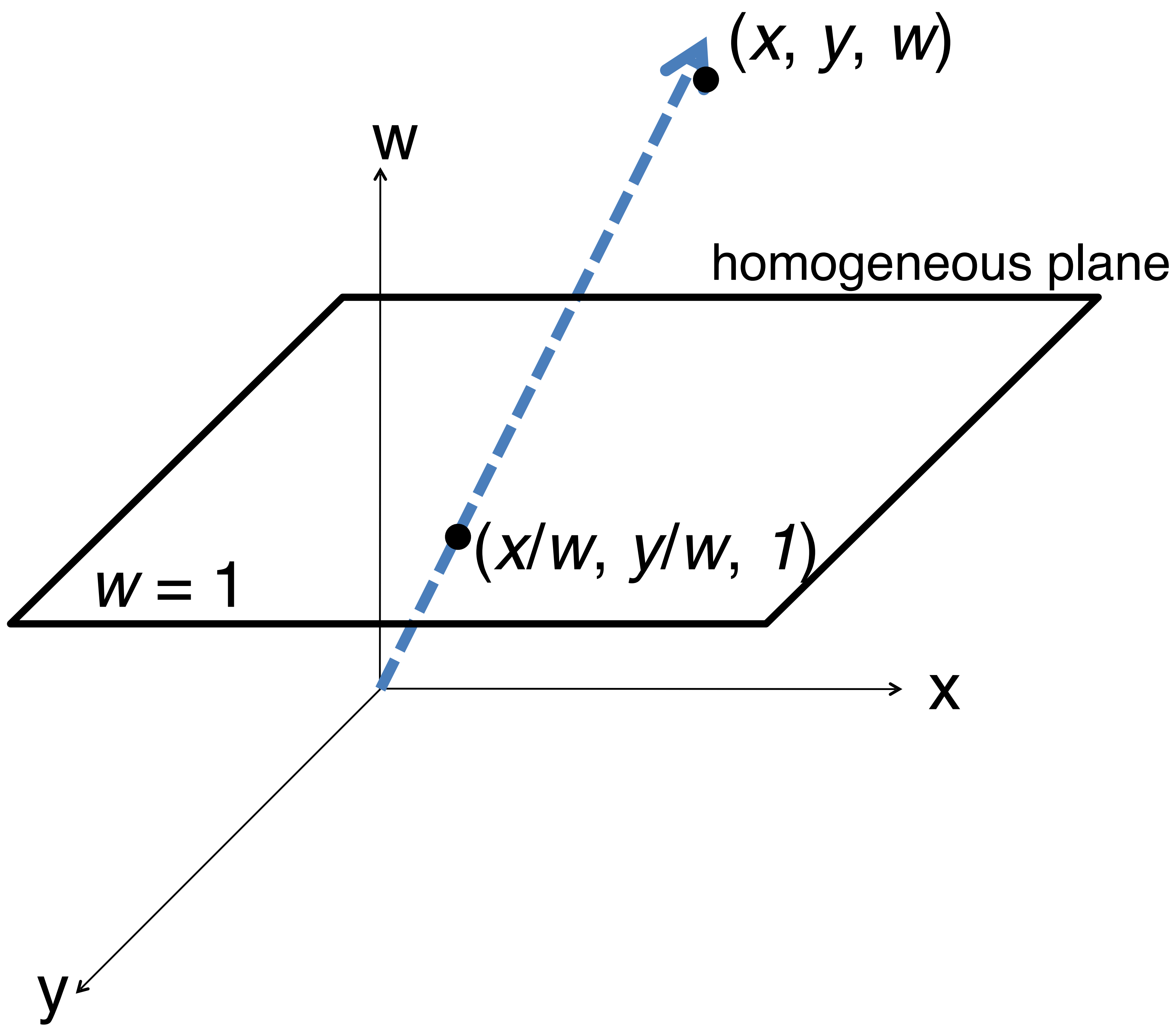
Source: Freeman, Torralba, Isola

# Homogeneous coordinates

Trick: add one more coordinate:

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

homogeneous image coordinates




Converting *from* homogeneous coordinates

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

Source: N. Snavely

# Application: translation with homogeneous coordinates

$$\mathbf{T} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

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

Source: N. Snavely

# Affine transformations

$$\mathbf{T} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

any transformation represented by a 3x3 matrix with last row [ 0 0 1 ] we call an *affine* transformation

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

Source: N. Snavely

# Examples of Affine Transformations

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

Translate

$$\begin{bmatrix} \boldsymbol{x}' \\ \boldsymbol{y}' \\ 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{s_x} & 0 & 0 \\ 0 & \boldsymbol{s_y} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \\ 1 \end{bmatrix}$$

Scale

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

2D *in-plane* rotation

$$\begin{bmatrix} \boldsymbol{x}' \\ \boldsymbol{y}' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & \boldsymbol{sh_x} & 0 \\ \boldsymbol{sh_y} & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \\ 1 \end{bmatrix}$$

Shear

Source: N. Snavely

# Perspective Projection

Projection is a matrix multiply using homogeneous coordinates:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1/f & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ z/f \end{bmatrix} \Rightarrow (f\frac{x}{z}, f\frac{y}{z})$$

divide by third coordinate

This is known as **perspective projection**

- The matrix is the **projection matrix**



Source: N. Snavely

# Perspective Projection

How does scaling the projection matrix change the transformation?

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1/f & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ z/f \end{bmatrix} \Rightarrow (f\frac{x}{z}, f\frac{y}{z})$$

What if we scale by *f*?

$$\begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} fx \\ fy \\ z \end{bmatrix} \Rightarrow (f\frac{x}{z}, f\frac{y}{z})$$

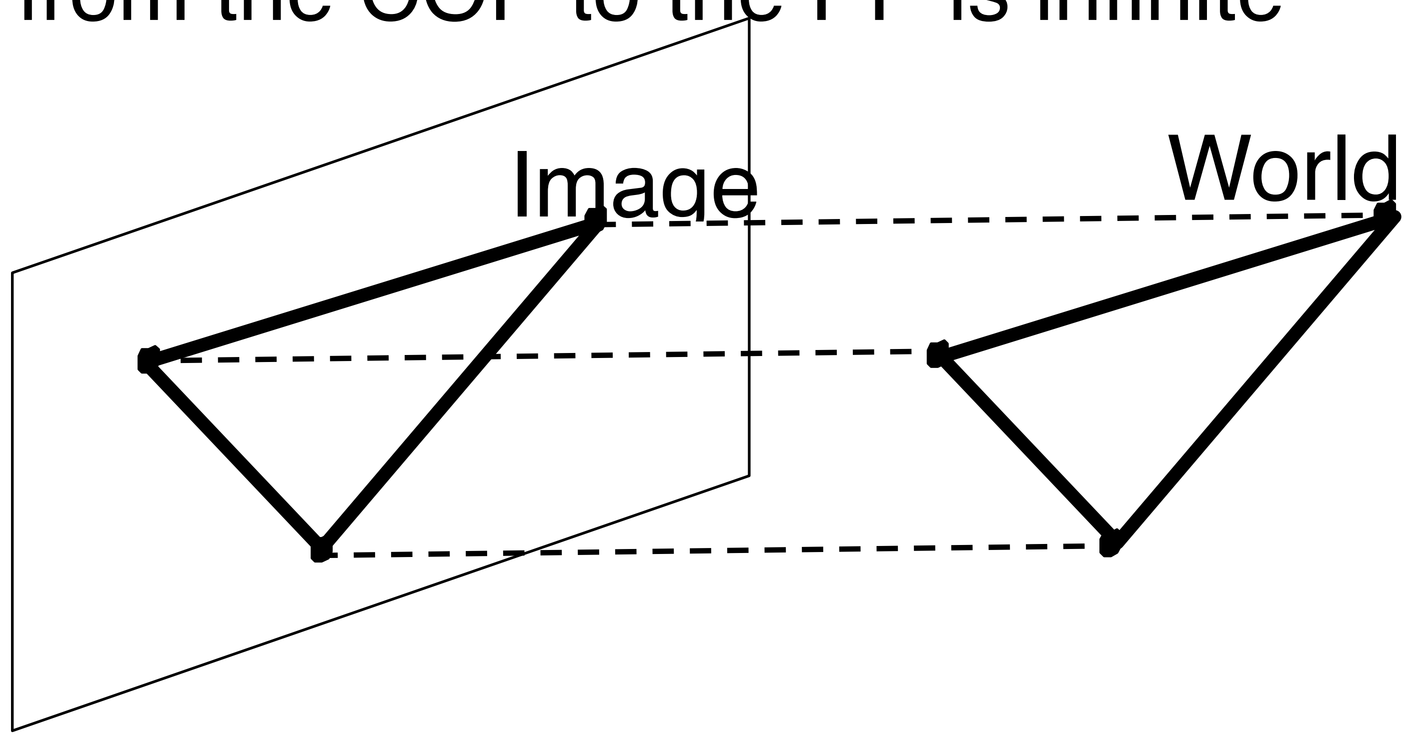
Scaling a projection matrix produces an equivalent projection matrix!



Source: N. Snavely

# Orthographic projection

- Special case of perspective projection
  - Distance from the COP to the PP is infinite



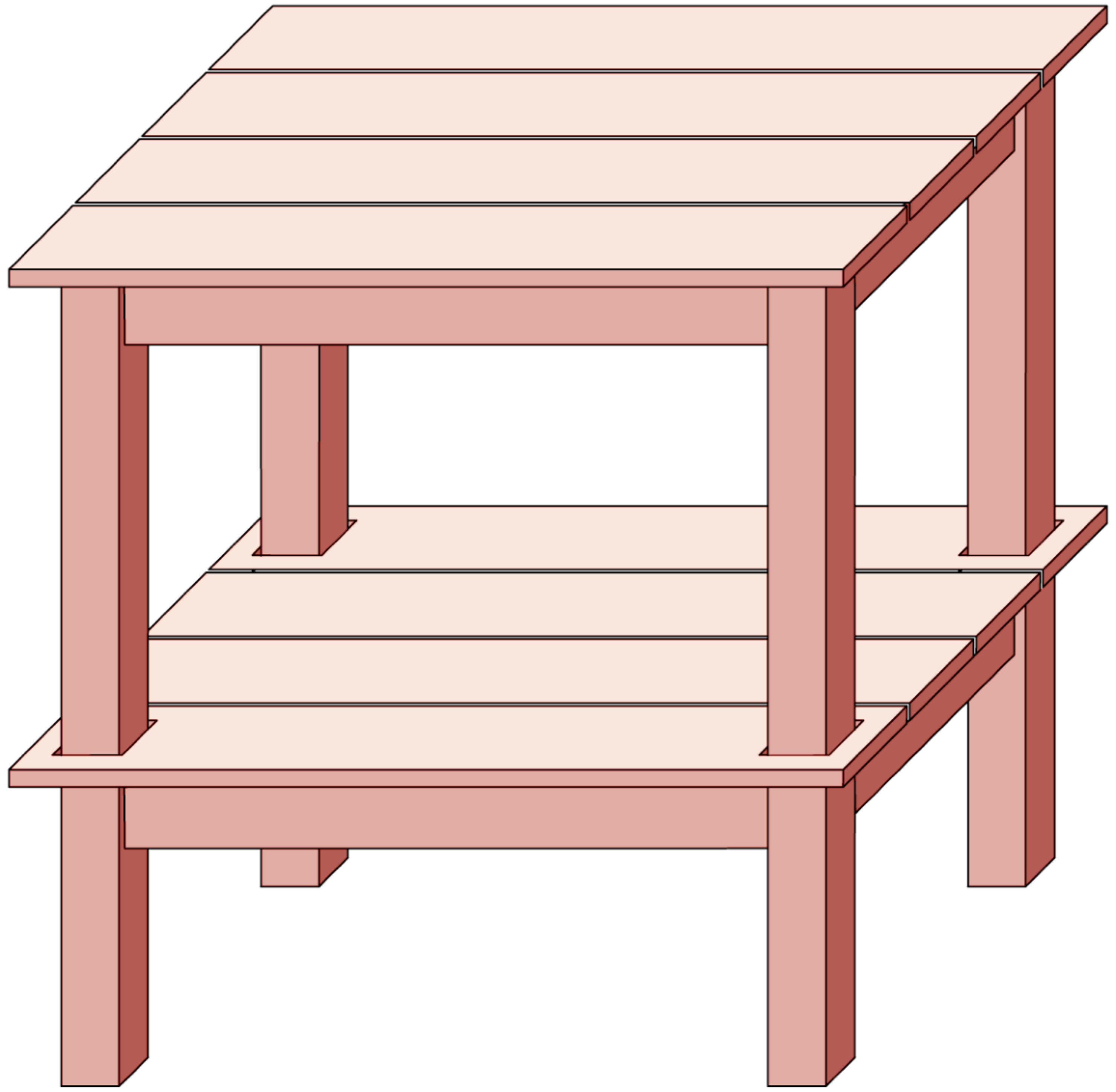

  - Good approximation for telephoto optics
  - Also called “parallel projection”: (x, y, z) → (x, y)
  - What’s the projection matrix?

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \Rightarrow (x, y)$$



Source: N. Snavely

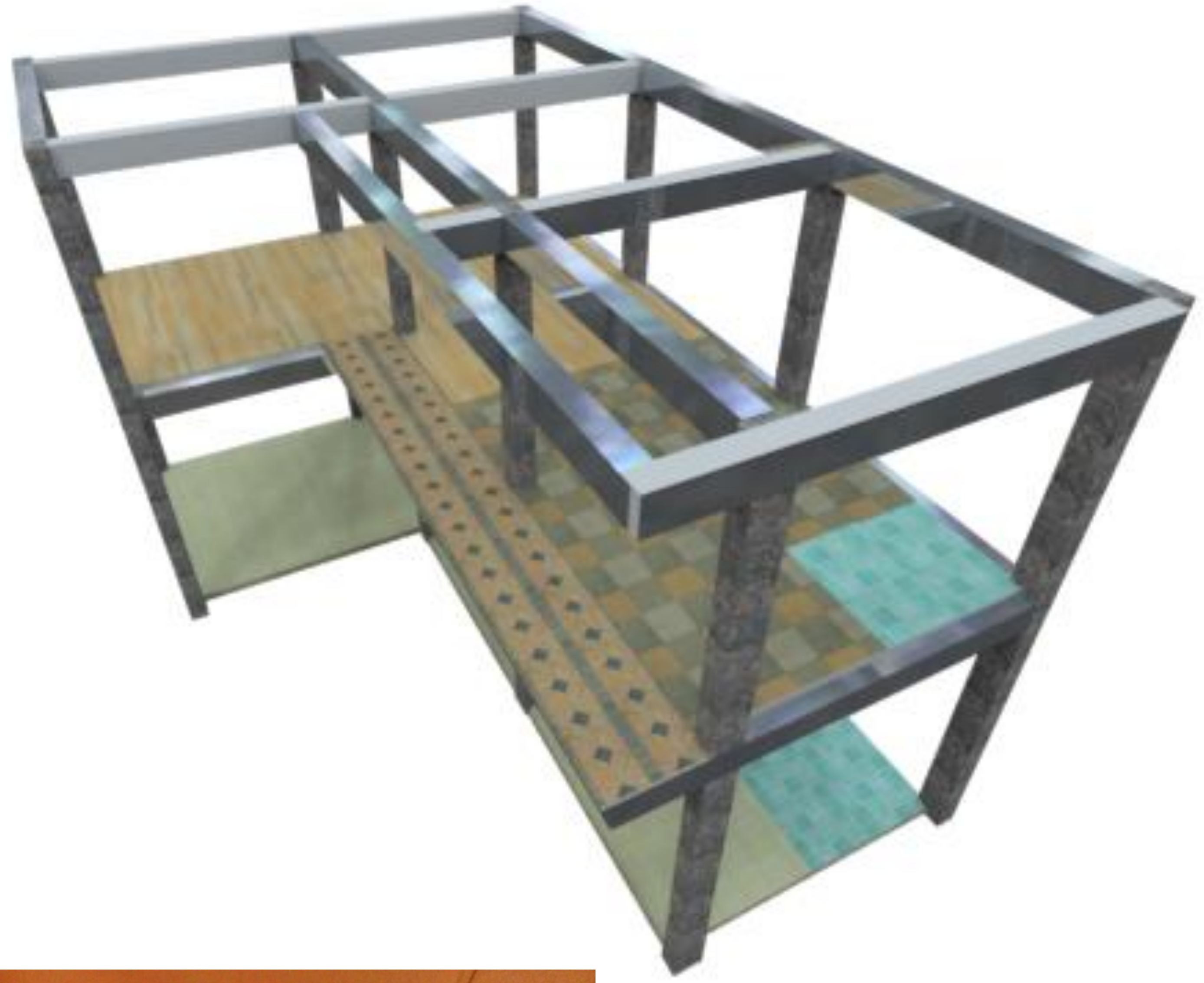
# Orthographic projection






Source: N. Snavely

# Perspective projection



Source: N. Snavely

# Projection properties

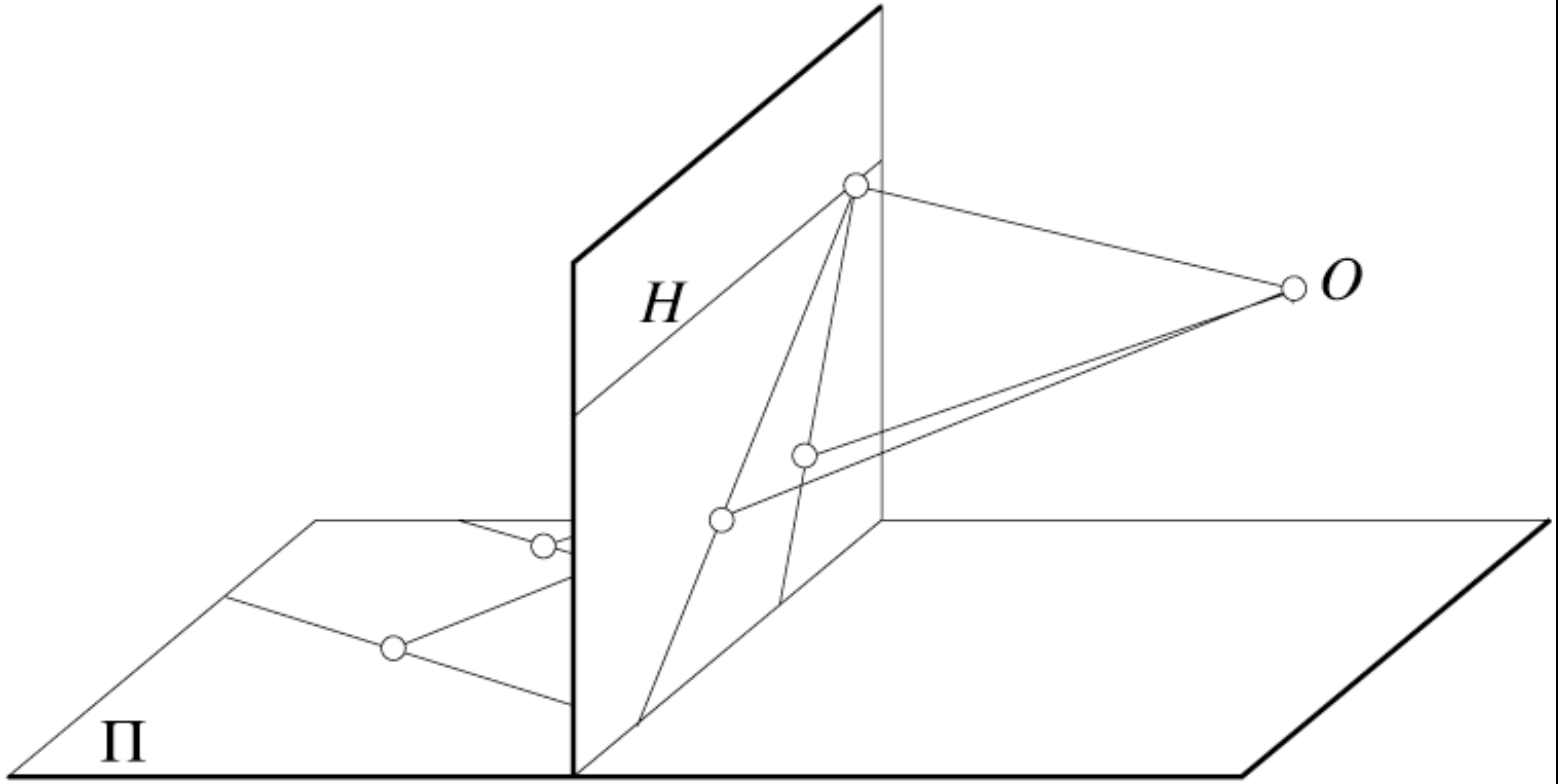
- Many-to-one: any points along same ray map to same point in image
- Points → points
- Lines → lines (collinearity is preserved)
  - But line through focal point projects to a point
- Planes → planes (or half-planes)
  - But plane through focal point projects to line

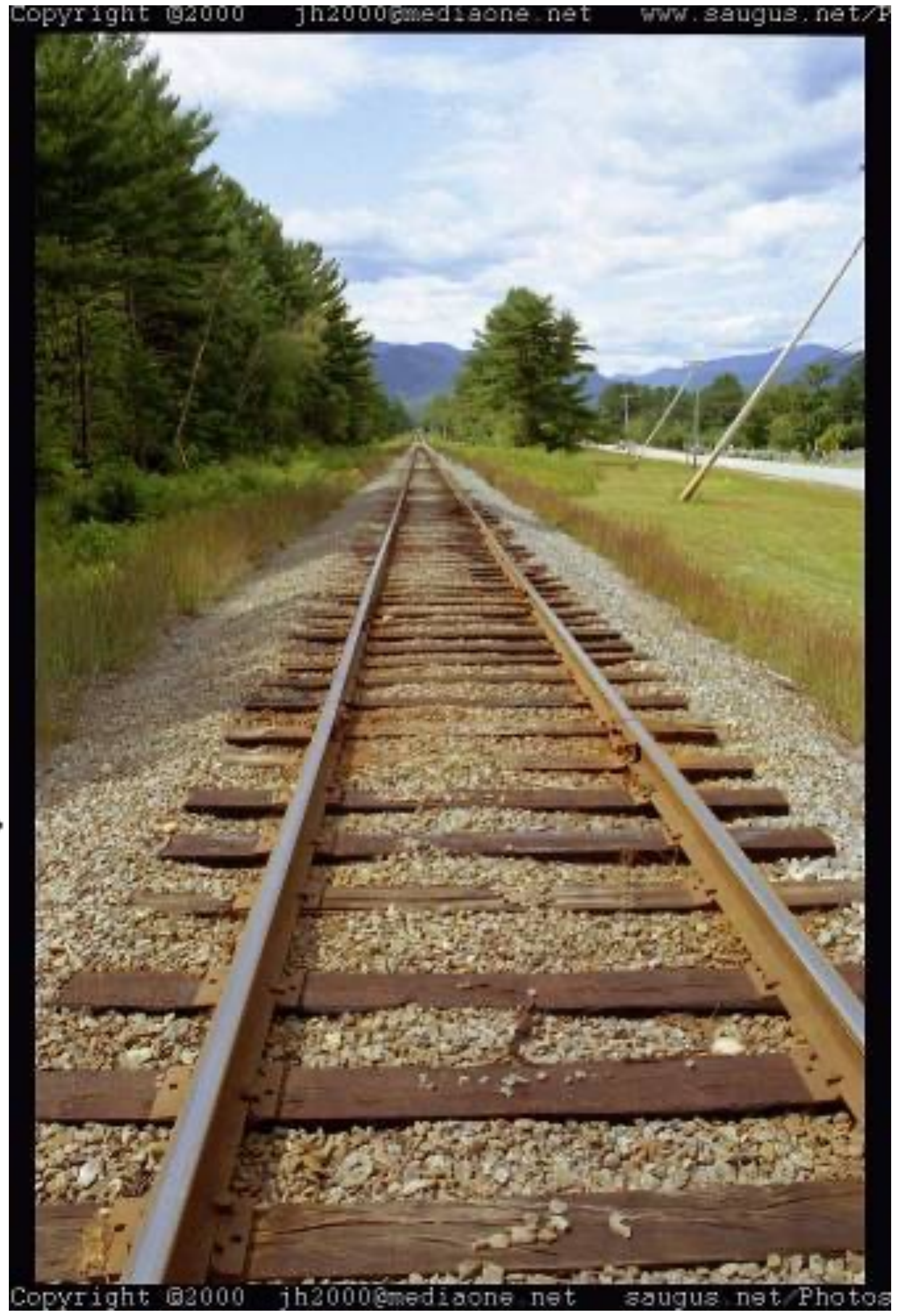


Source: N. Snavely

# Projection properties

- Parallel lines converge at a vanishing point
  - Each direction in space has its own vanishing point
  - But lines parallel to the image plane remain parallel

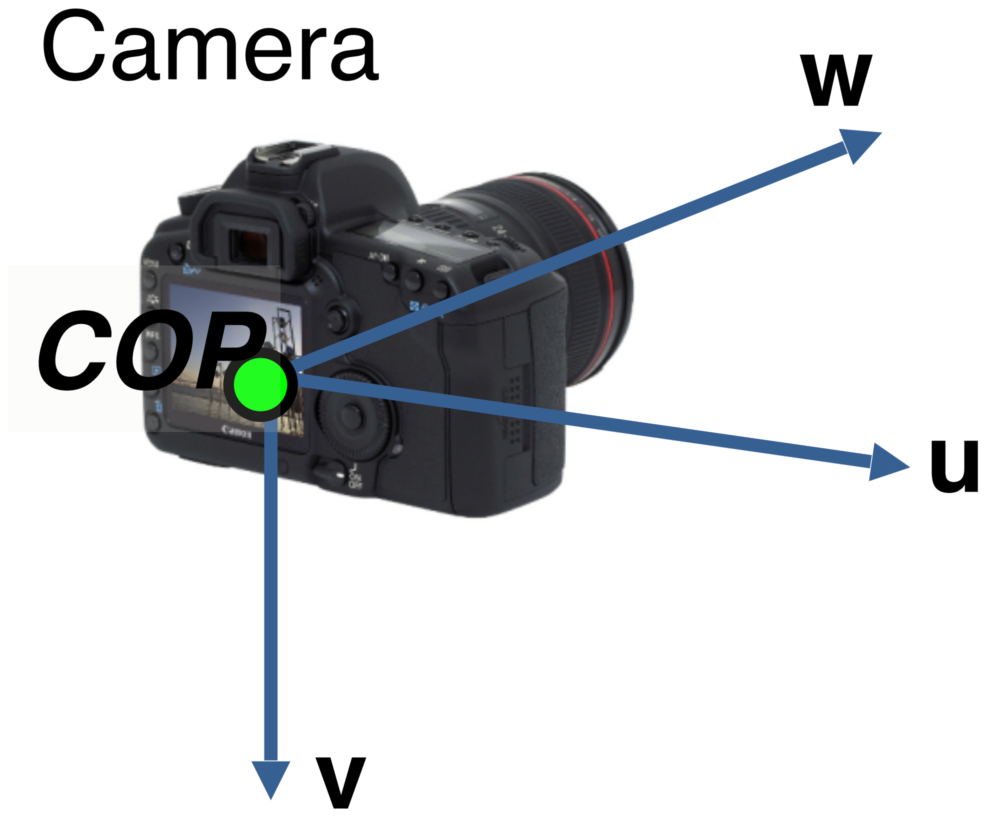


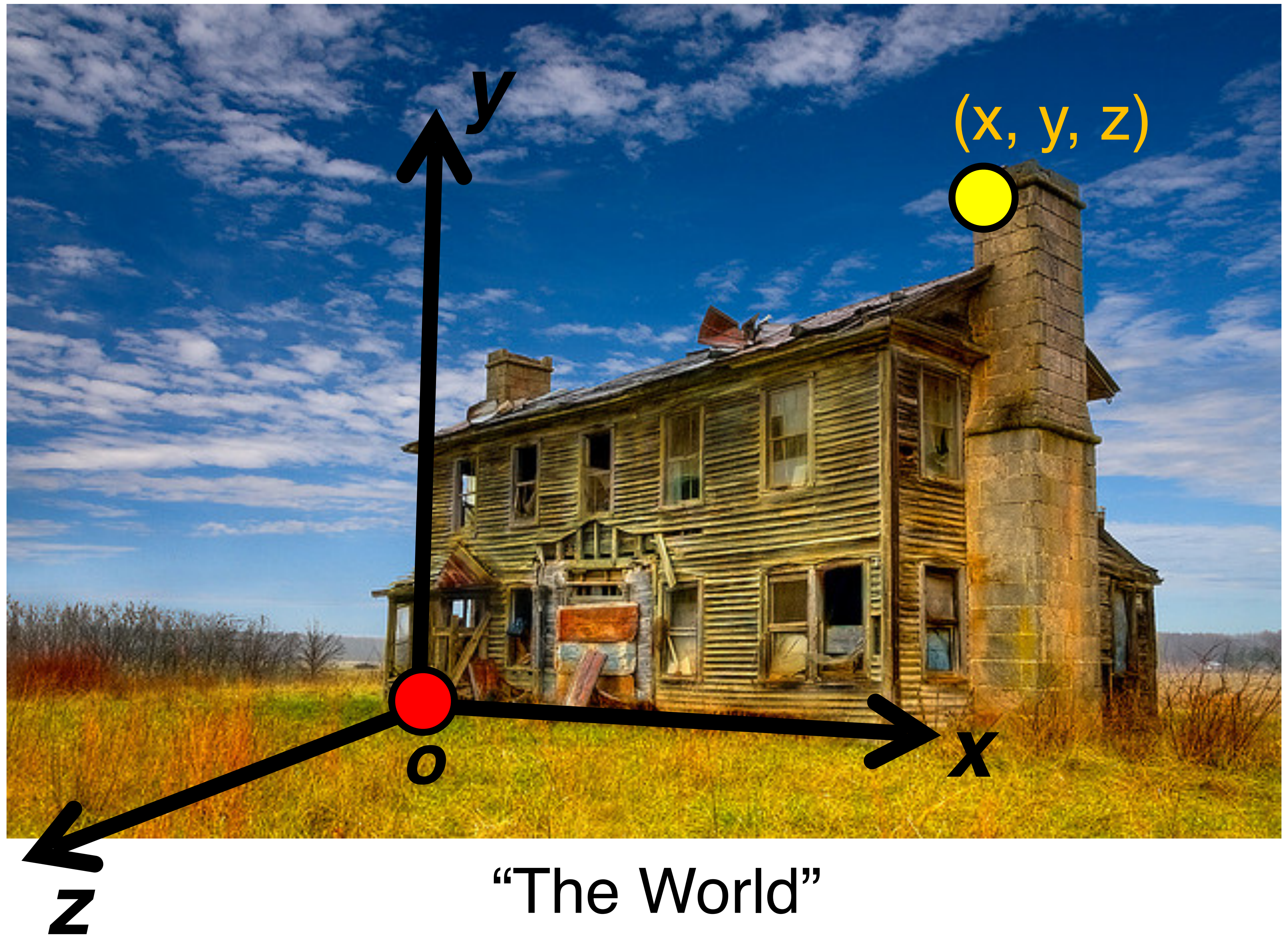



Source: N. Snavely

# Camera parameters

- How can we model the geometry of a camera?




"The World"

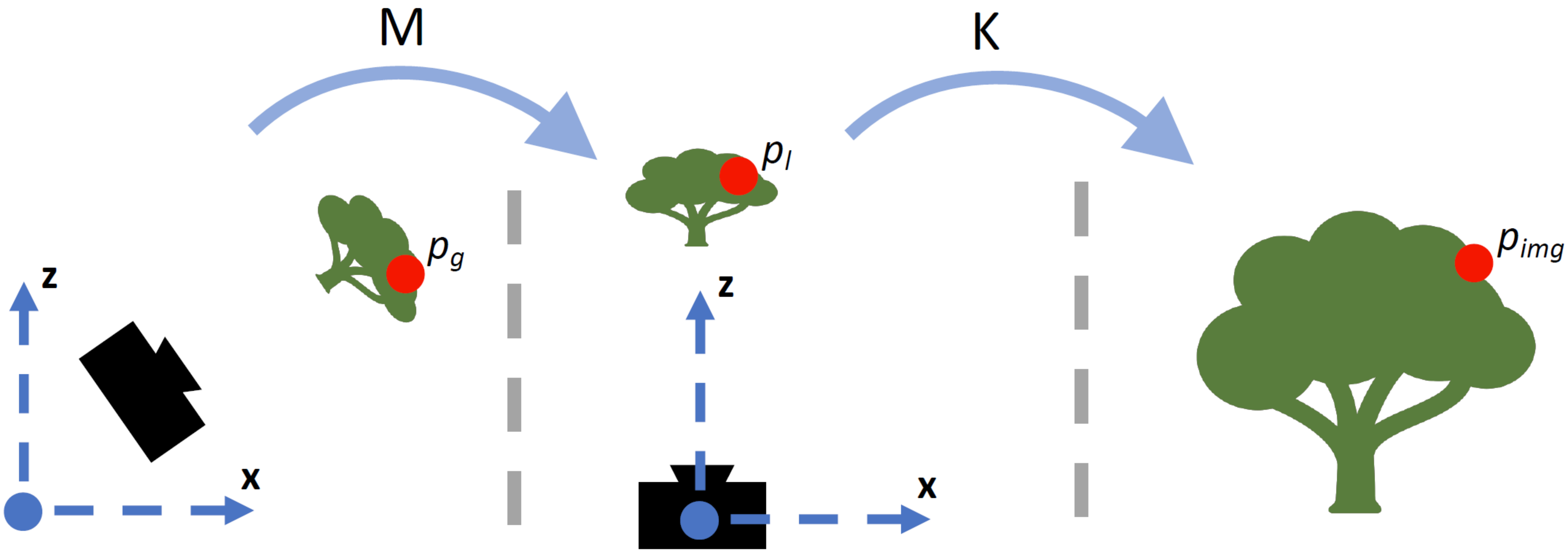
Three important coordinate systems:

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




World coordinates Camera coordinates Image coordinates

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*

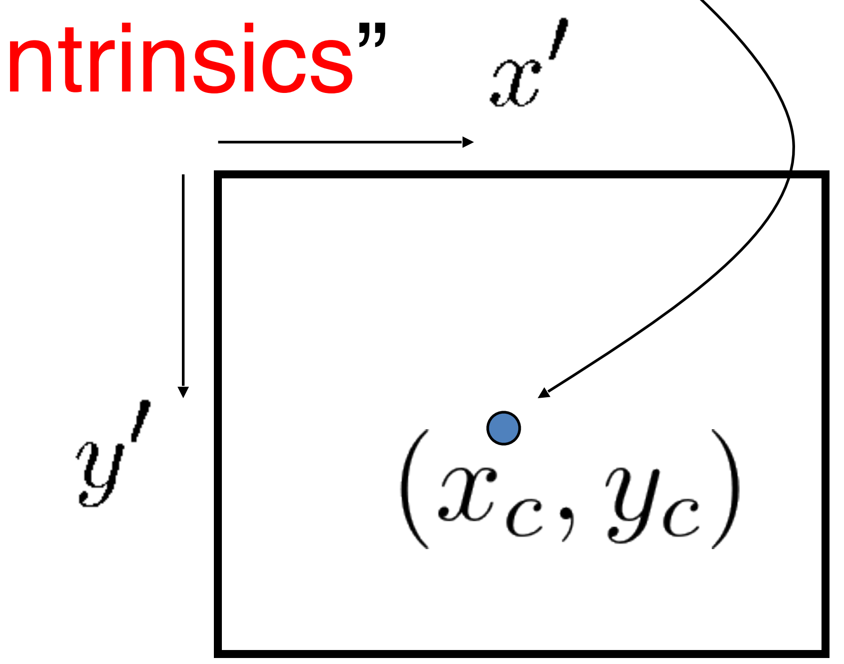


Source: N. Snavely

# Camera parameters

A camera is described by several parameters

- Translation T of the optical center from the origin of world coords
- Rotation R of the image plane
- focal length f, principal point $(c_x, c_y)$, pixel aspect size α
- blue parameters are called "extrinsics," red are "intrinsics"




Projection equation

$$\mathbf{x} = \begin{bmatrix} sx \\ sy \\ s \end{bmatrix} = \begin{bmatrix} * & * & * & * \\ * & * & * & * \\ * & * & * & * \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \mathbf{\Pi X}$$

- The projection matrix models the cumulative effect of all parameters
- Useful to decompose into a series of operations

$$\mathbf{\Pi} = \underbrace{\begin{bmatrix} f & s & c_x \\ 0 & \alpha f & c_y \\ 0 & 0 & 1 \end{bmatrix}}_{\text{intrinsics}} \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{\text{projection}} \underbrace{\begin{bmatrix} \mathbf{R}_{3\times3} & \mathbf{0}_{3\times1} \\ \mathbf{0}_{1\times3} & 0 \end{bmatrix}}_{\text{rotation}} \underbrace{\begin{bmatrix} \mathbf{I}_{3\times3} & \mathbf{T}_{3\times1} \\ \mathbf{0}_{1\times3} & 0 \end{bmatrix}}_{\text{translation}}$$

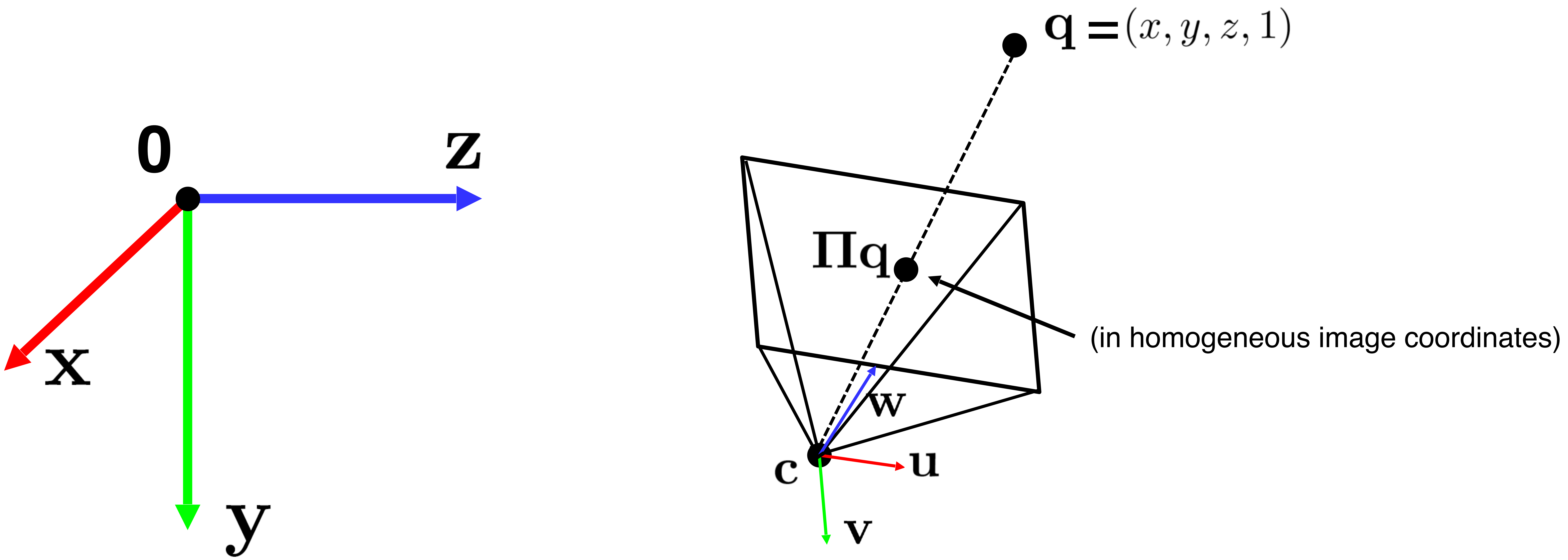
identity matrix

- The definitions of these parameters are **not** completely standardized
  - especially intrinsics—varies from one book to another



Source: N. Snavely

# Projection matrix

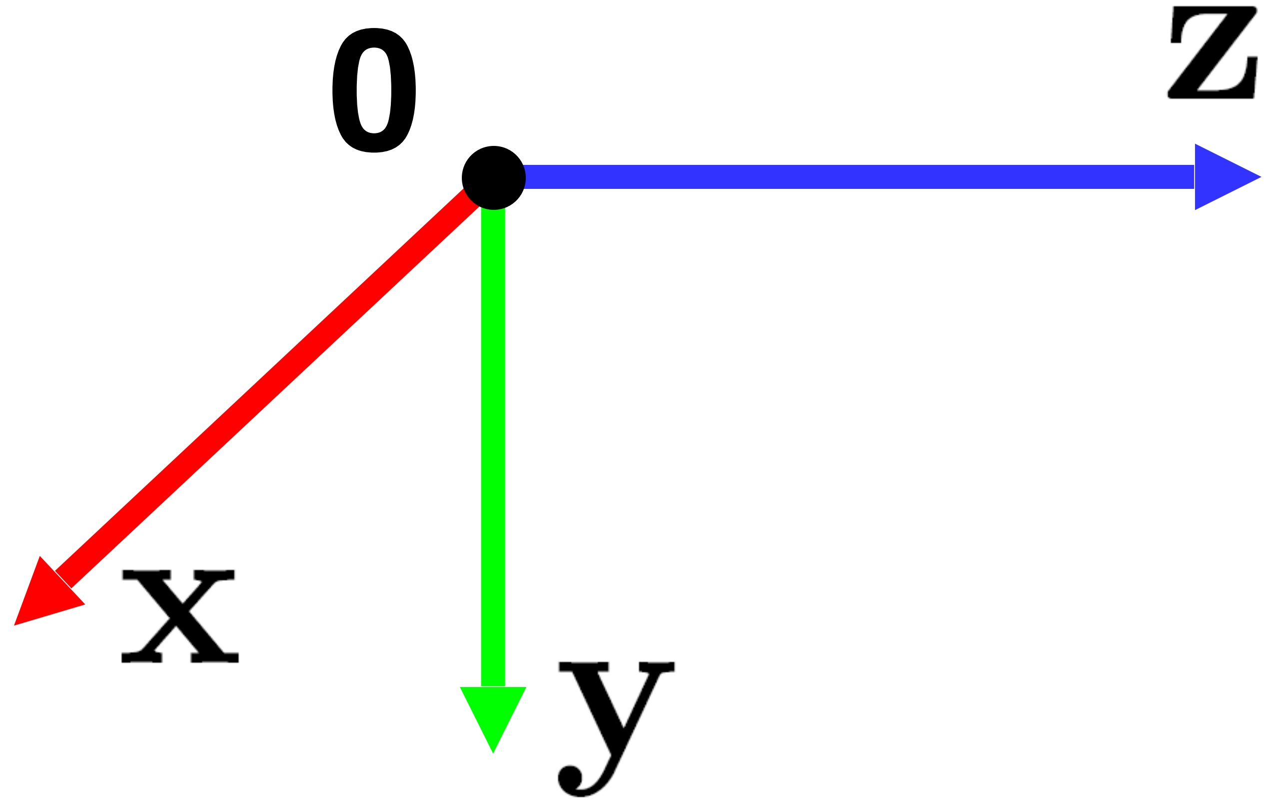




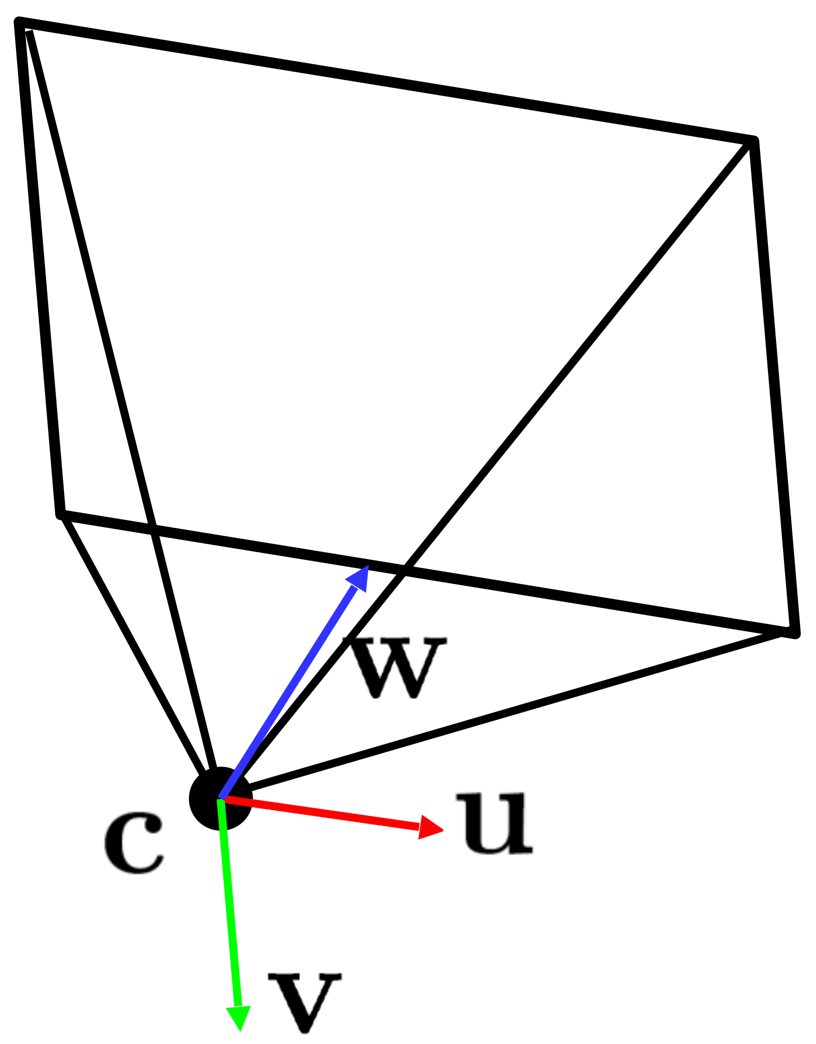

Source: N. Snavely

# 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**

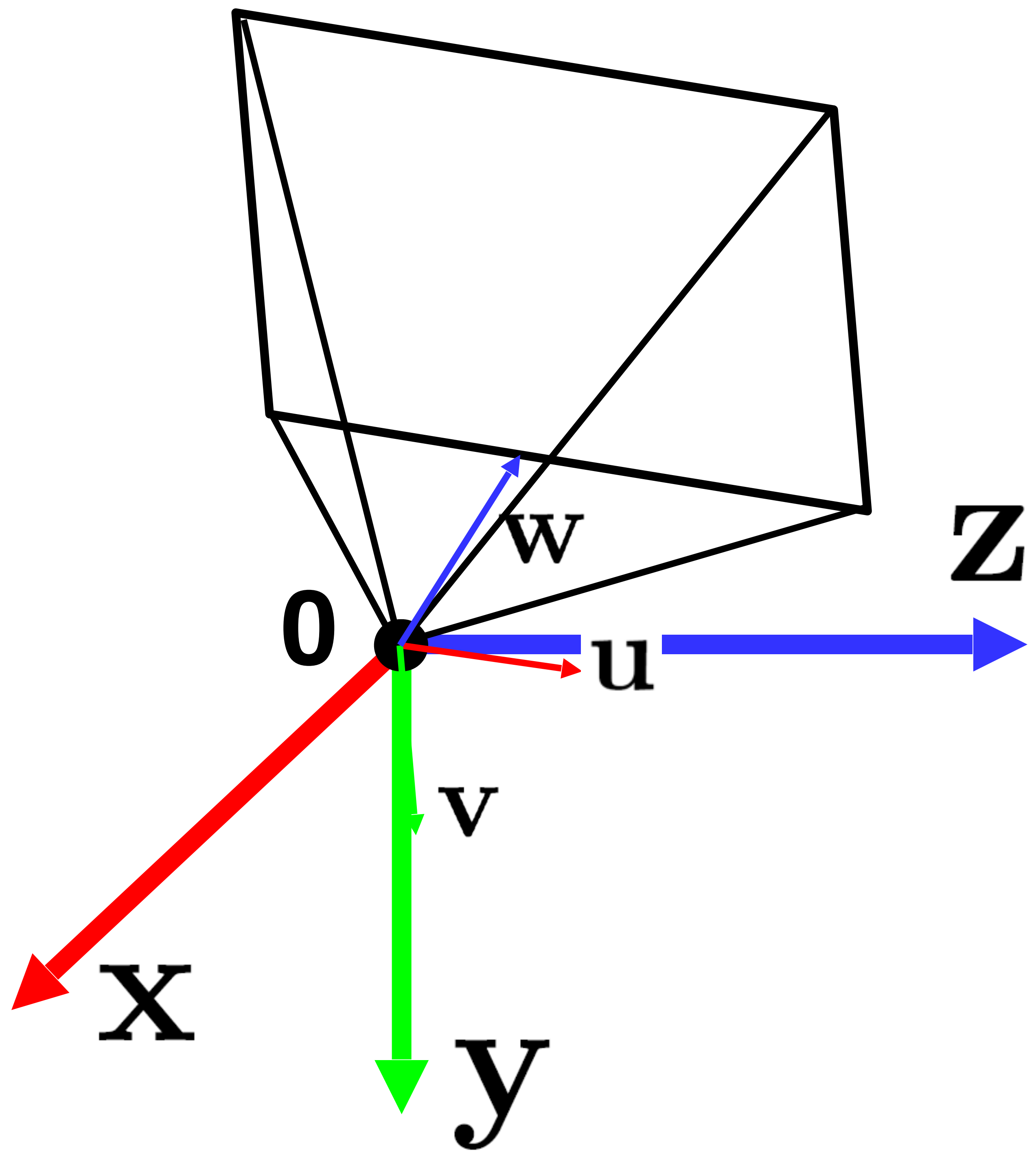





Source: N. Snavely

# 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 **-c**

How do we represent translation as a matrix multiplication?

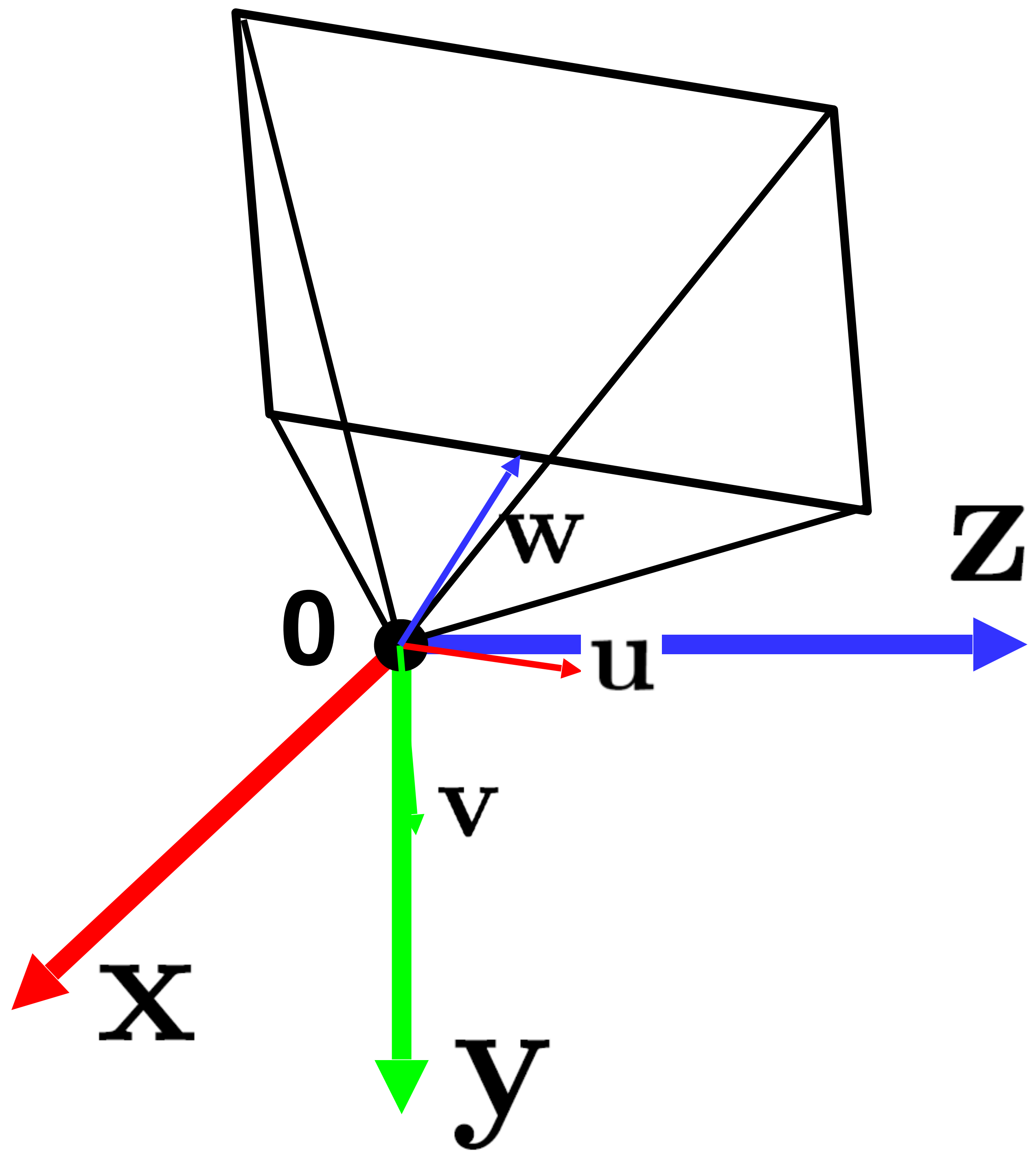
$$\mathbf{T} = \begin{bmatrix} \mathbf{I}_{3\times 3} & -\mathbf{c} \\ 0 \quad 0 \quad 0 & 1 \end{bmatrix}$$



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**

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

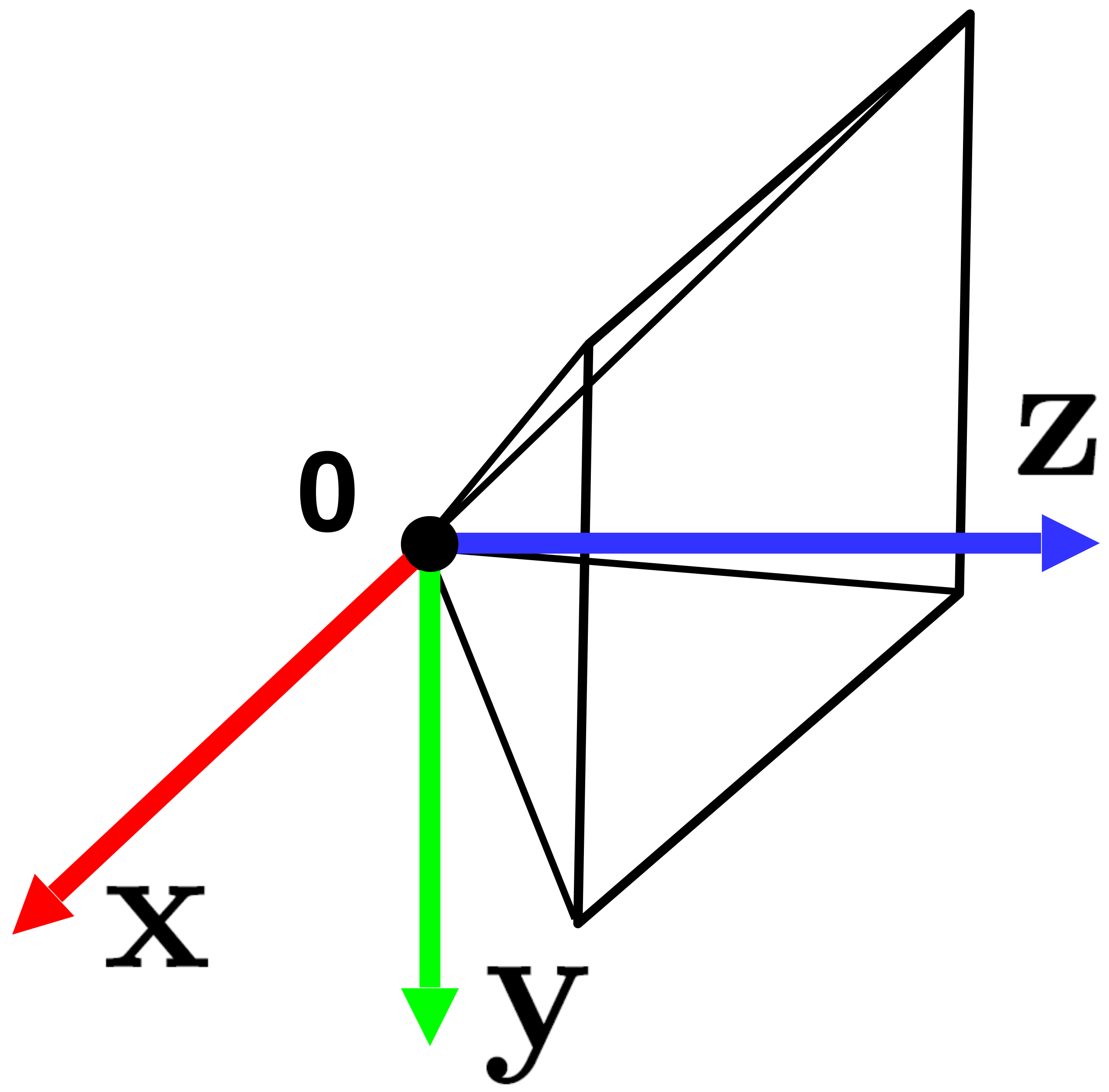
3x3 rotation matrix



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

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

$\mathbf{K}$ (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)

$\alpha$: **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

# 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

# Projection matrix

$$\mathbf{\Pi} = \mathbf{K} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{R} & \begin{matrix} 0 \\ 0 \\ 0 \end{matrix} \\ 0\ 0\ 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{I}_{3\times3} & -\mathbf{c} \\ 0\ 0\ 0 & 1 \end{bmatrix}$$

intrinsics

projection rotation translation

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).

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



Source: N. Snavely

# Projection matrix

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

$$\left[ \begin{array}{c|c} \mathbf{R} & \underbrace{-\mathbf{Rc}} \end{array} \right]$$

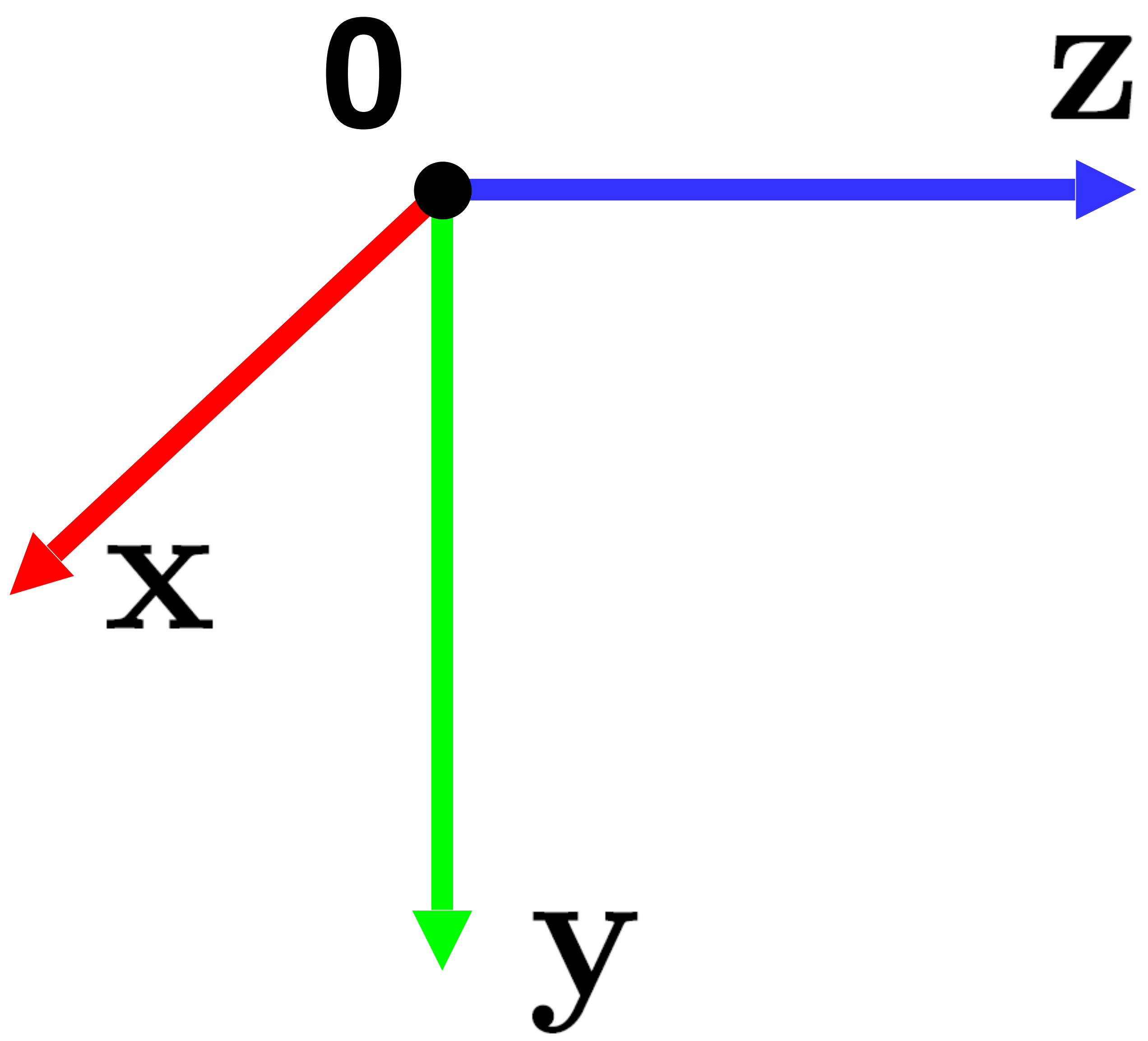
(sometimes called **t**)

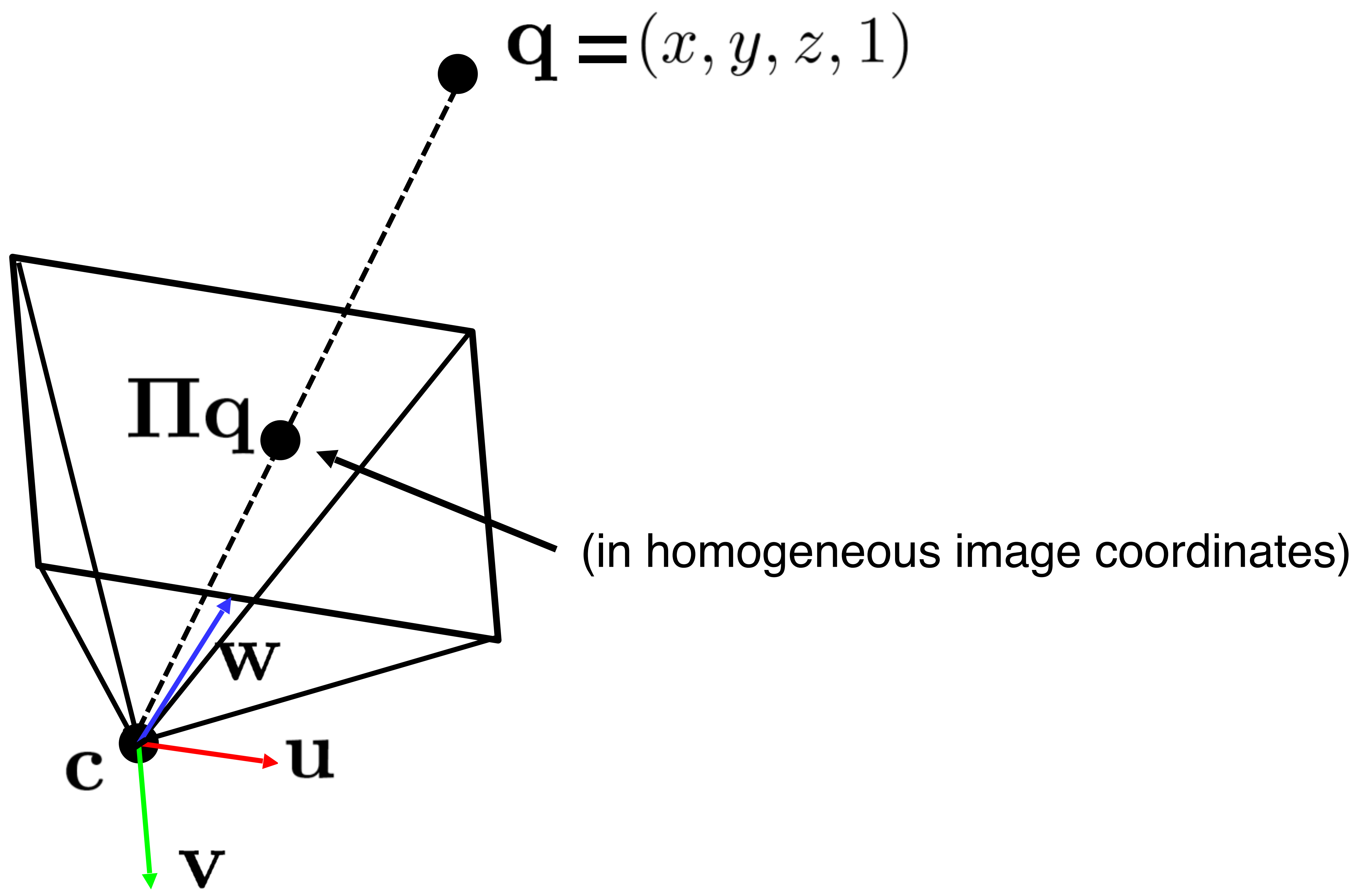
$$\mathbf{\Pi} = \mathbf{K} \left[ \begin{array}{c|c} \mathbf{R} & -\mathbf{Rc} \end{array} \right]$$



Source: N. Snavely

# Projection matrix

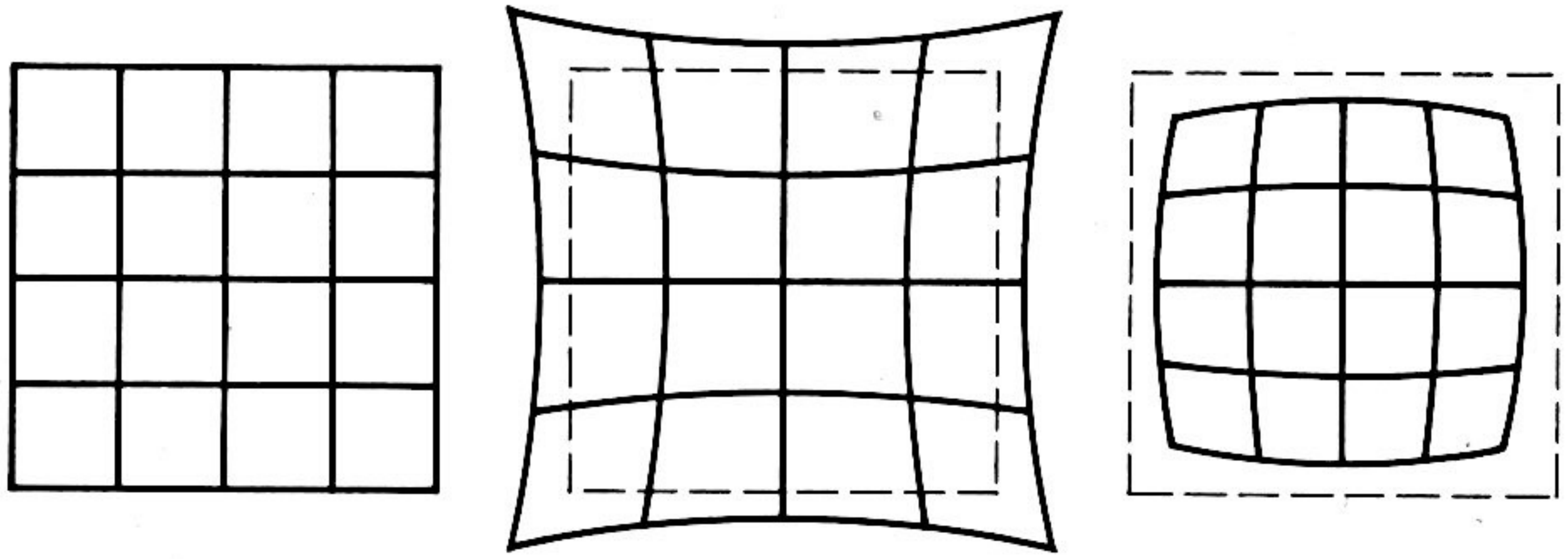





Source: N. Snavely

# 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

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

Apply radial distortion

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

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

**Next class:** More geometry!