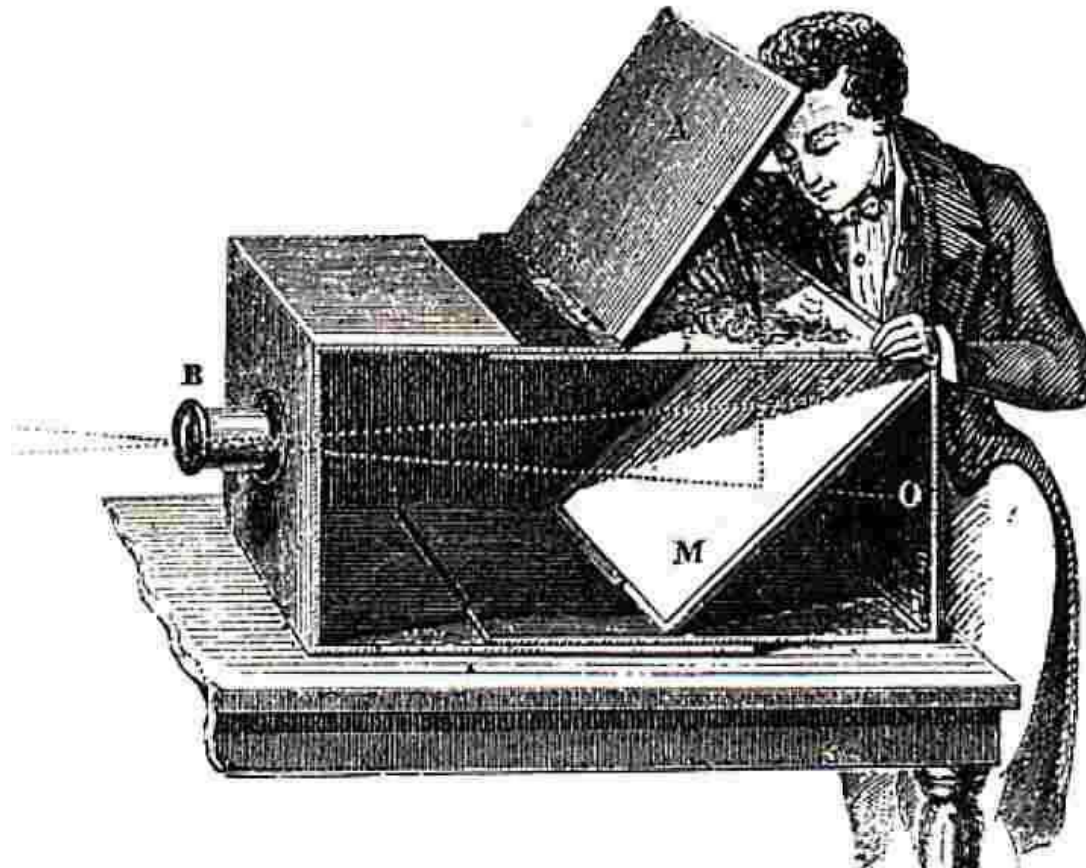
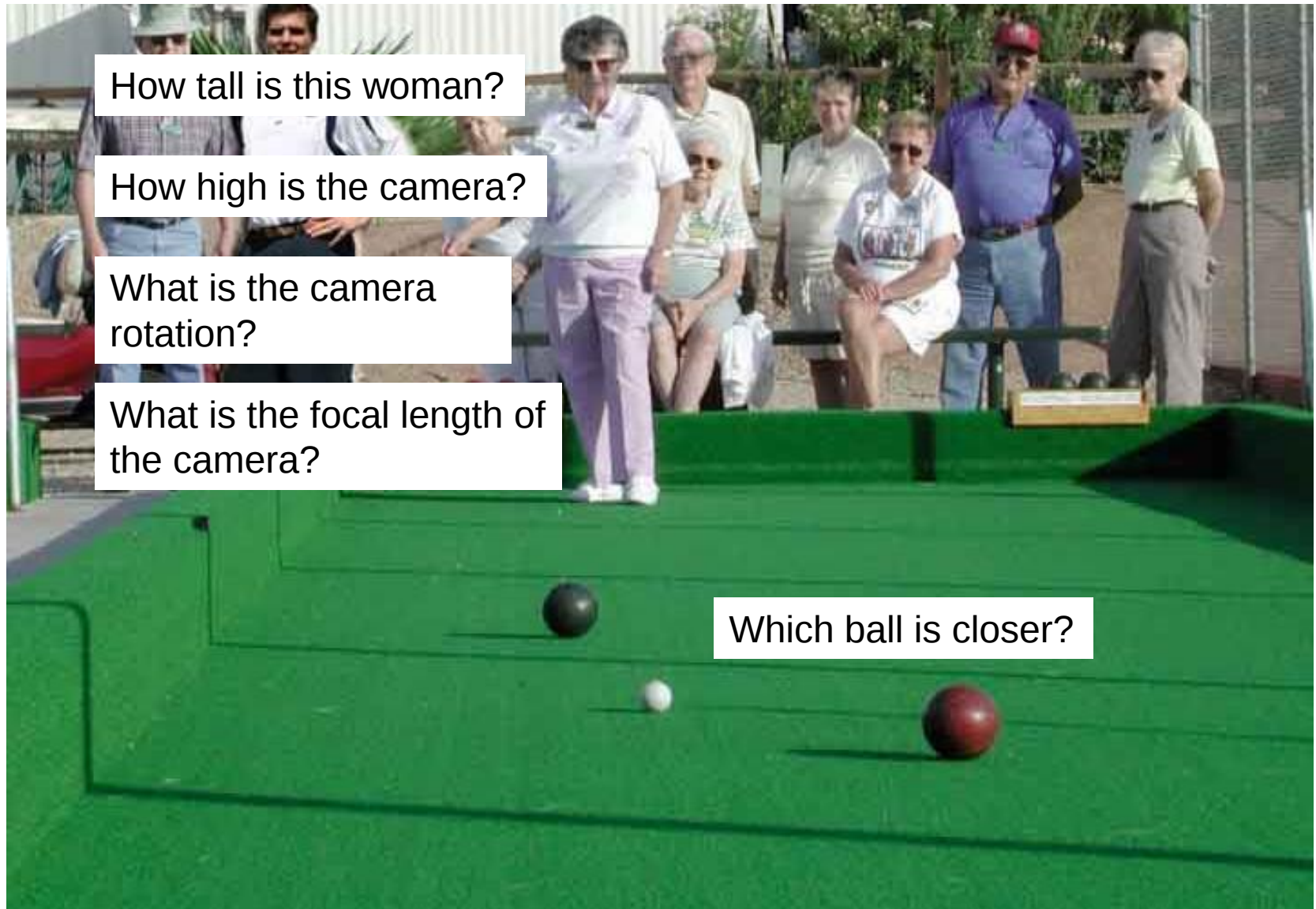


Pinhole Camera Model



Computational Photography
Derek Hoiem, University of Illinois

Next classes: Single-view Geometry

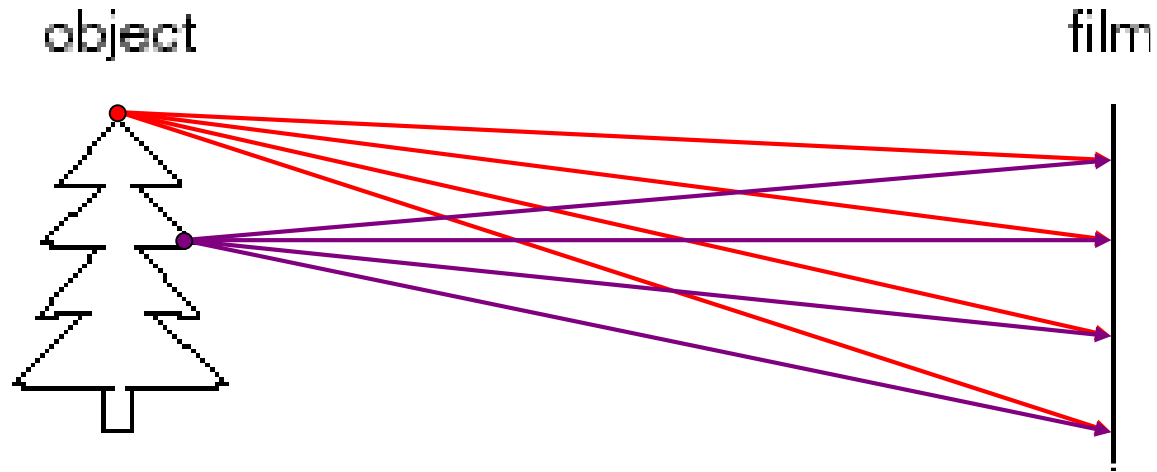


Today's class

Mapping between image and world coordinates

- Pinhole camera model
- Projective geometry
 - Vanishing points and lines
- Projection matrix

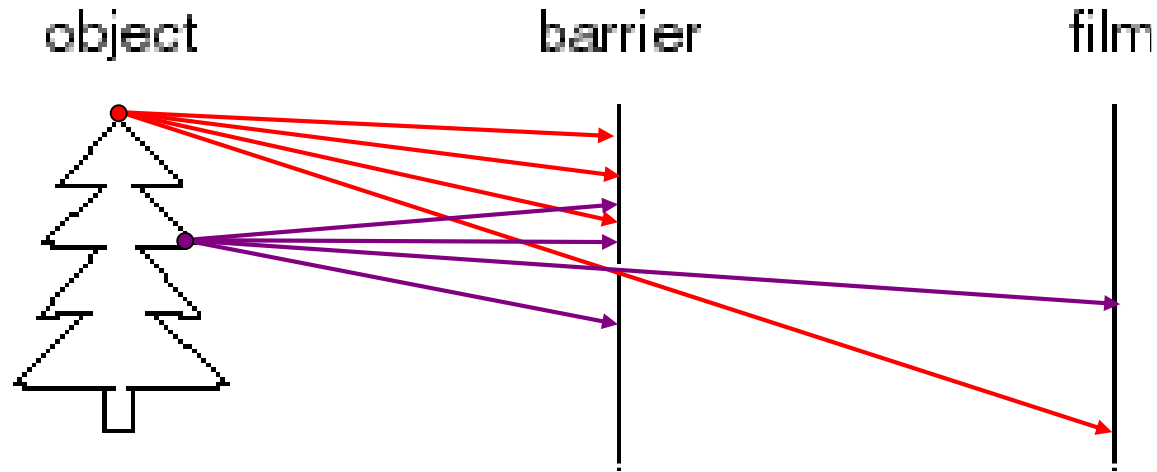
Image formation



Let's design a camera

- Idea 1: put a piece of film in front of an object
- Do we get a reasonable image?

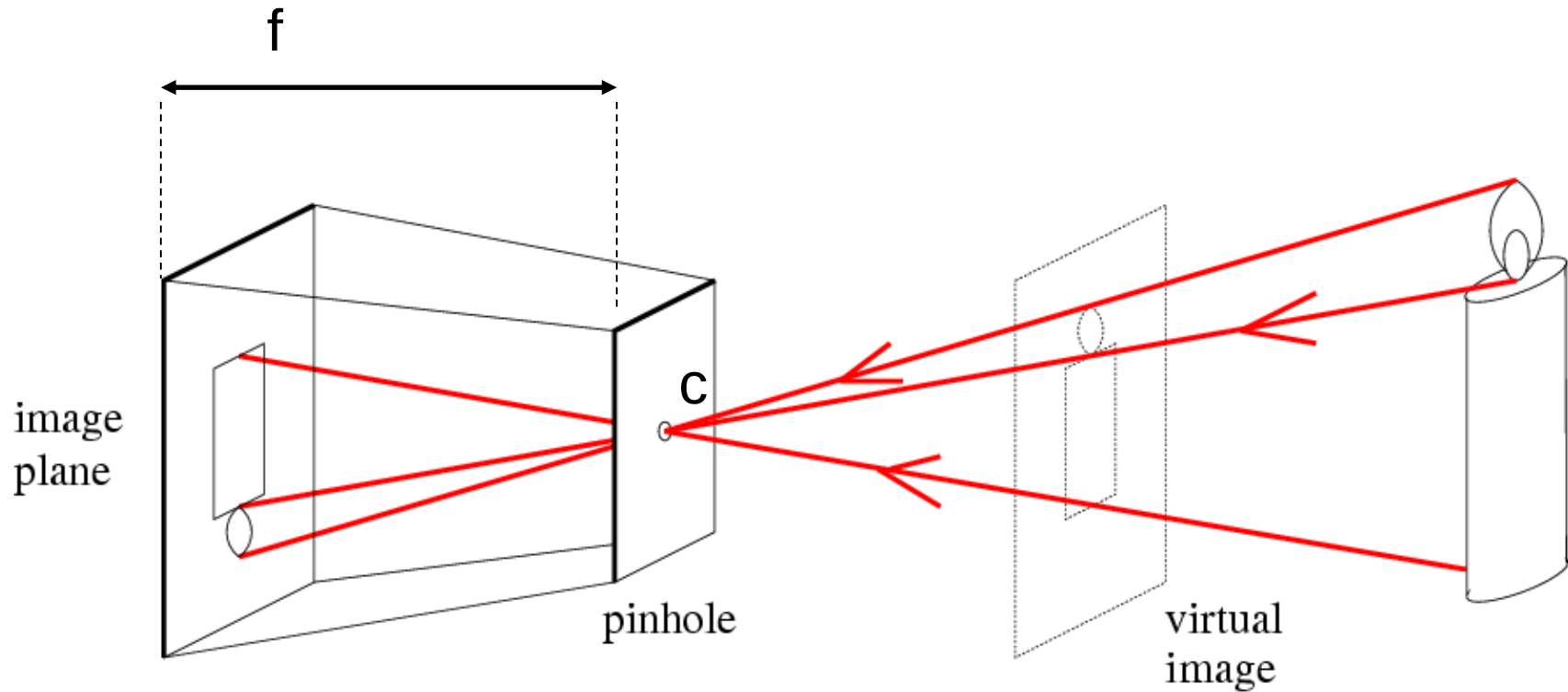
Pinhole camera



Idea 2: add a barrier to block off most of the rays

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

Pinhole camera



f = focal length

c = center of the camera

Camera obscura: the pre-camera

- First idea: Mozi, China (470BC to 390BC)
- First built: Alhacen, Iraq/Egypt (965 to 1039AD)

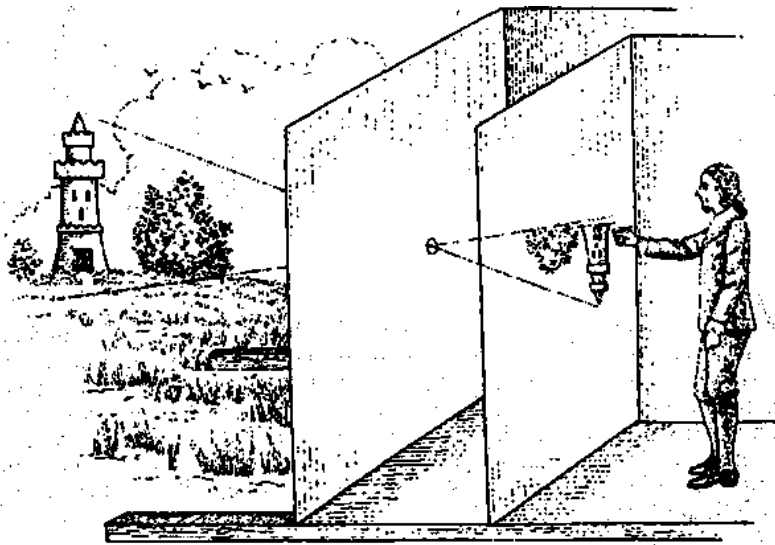


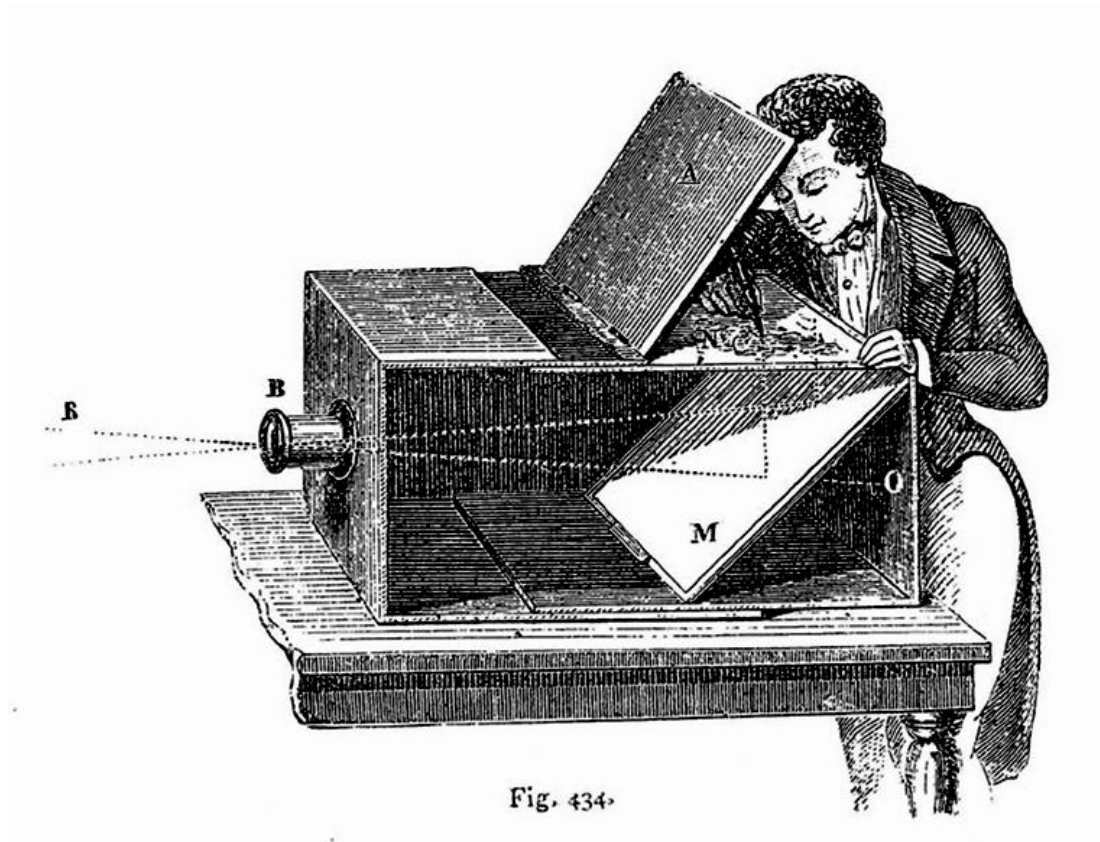
Illustration of Camera Obscura



Freestanding camera obscura at UNC Chapel Hill

Photo by Seth Ilys

Camera Obscura used for Tracing



Lens Based Camera Obscura, 1568

First Photograph

Oldest surviving photograph
– Took 8 hours on pewter plate



Joseph Niepce, 1826

Photograph of the first photograph

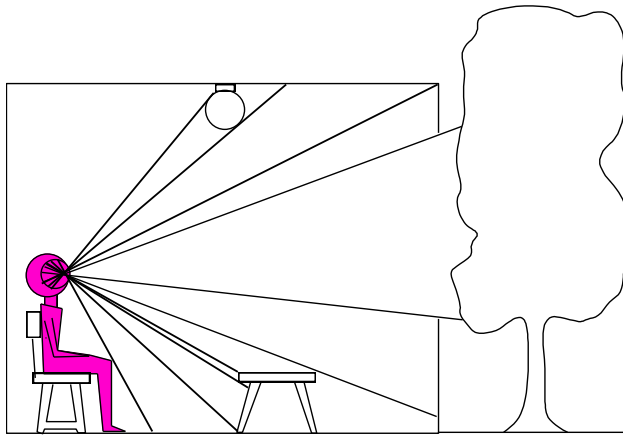


Stored at UT Austin

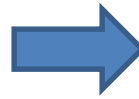
Niepce later teamed up with Daguerre, who eventually created Daguerrotypes

Dimensionality Reduction Machine (3D to 2D)

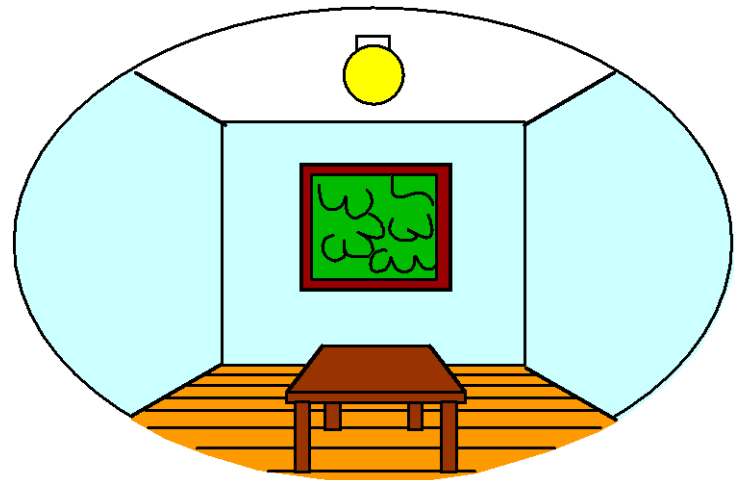
3D world



Point of observation



2D image



Projection can be tricky...



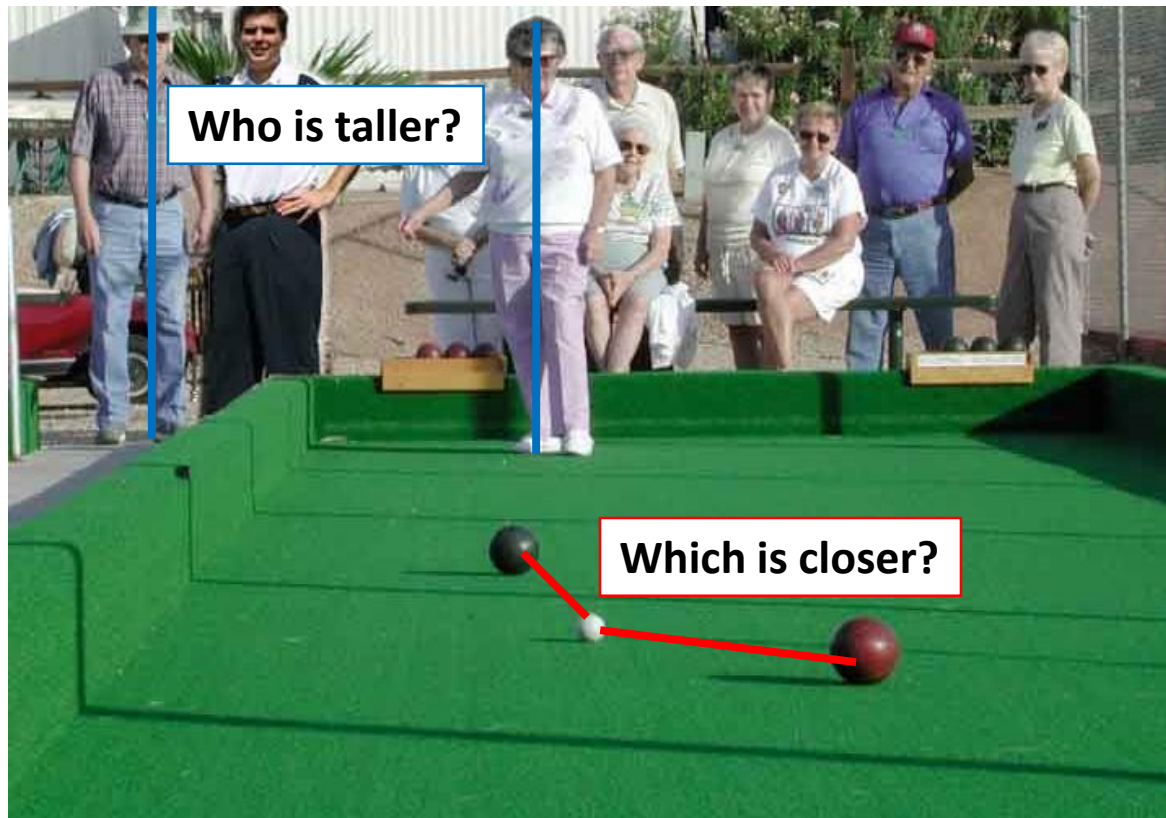
Projection can be tricky...



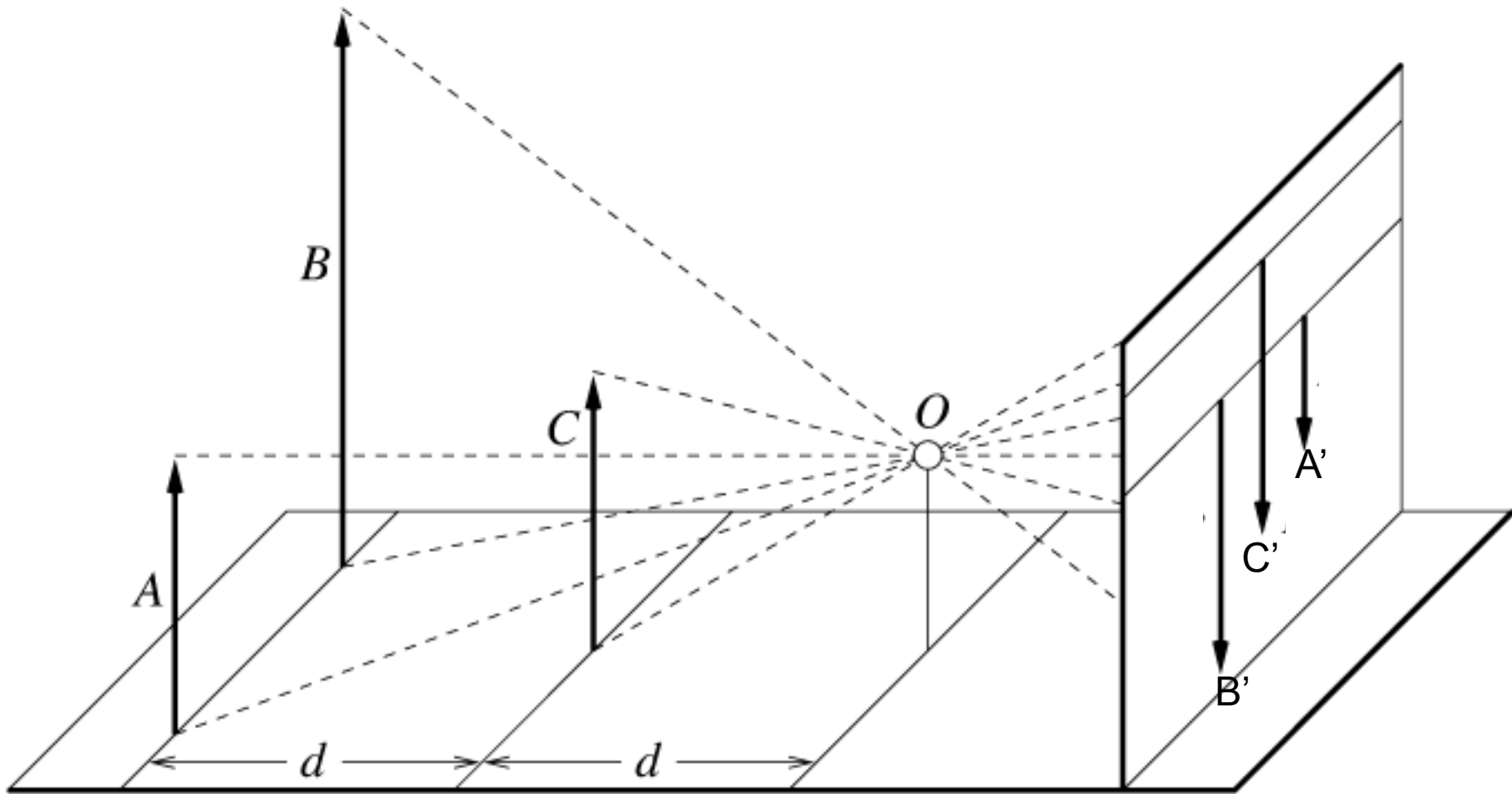
Projective Geometry

What is lost?

- Length



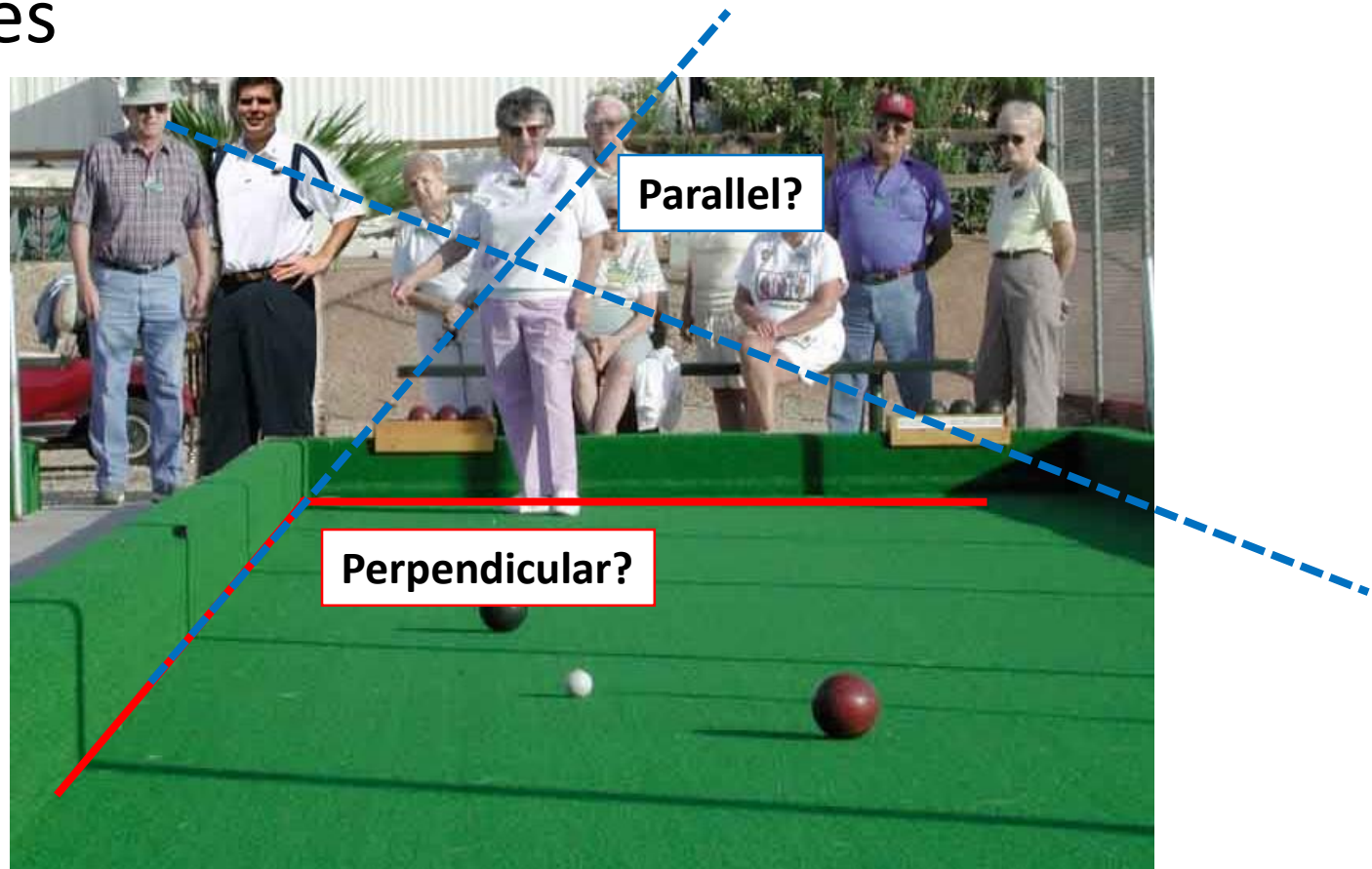
Length is not preserved



Projective Geometry

What is lost?

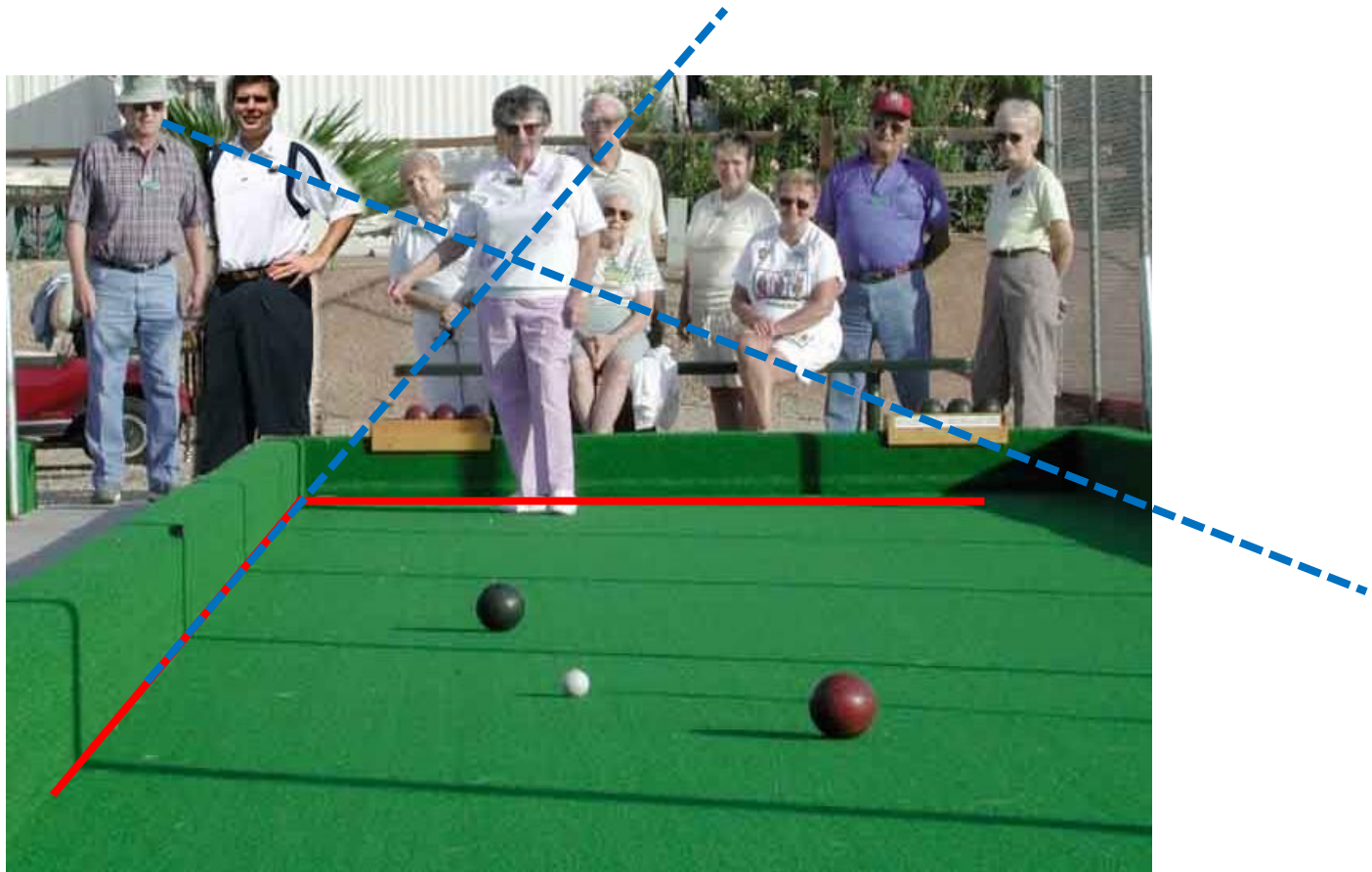
- Length
- Angles



Projective Geometry

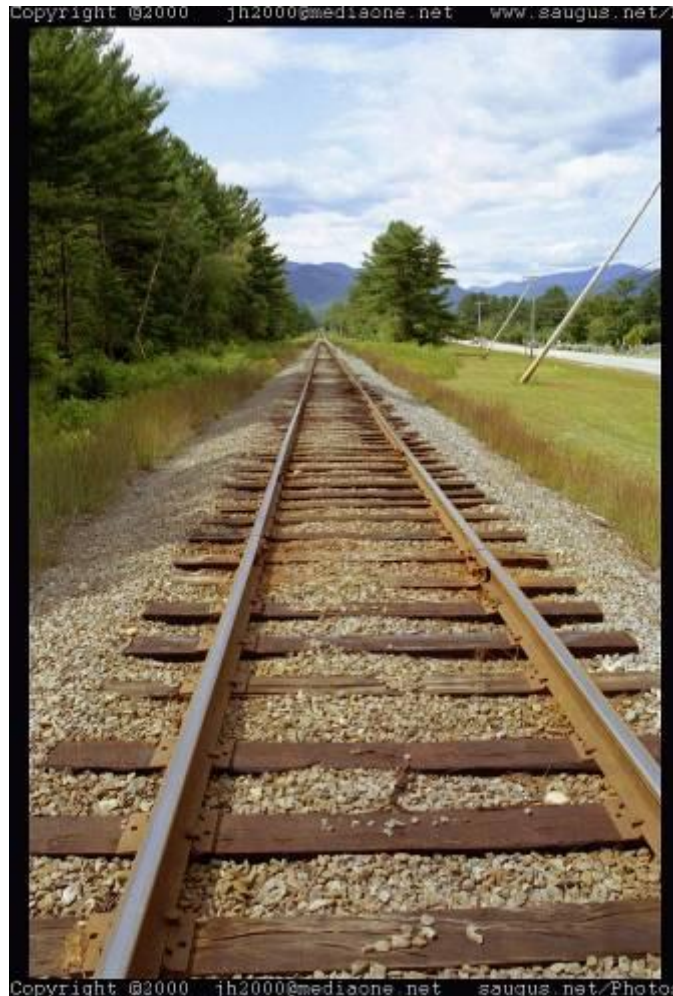
What is preserved?

- Straight lines are still straight

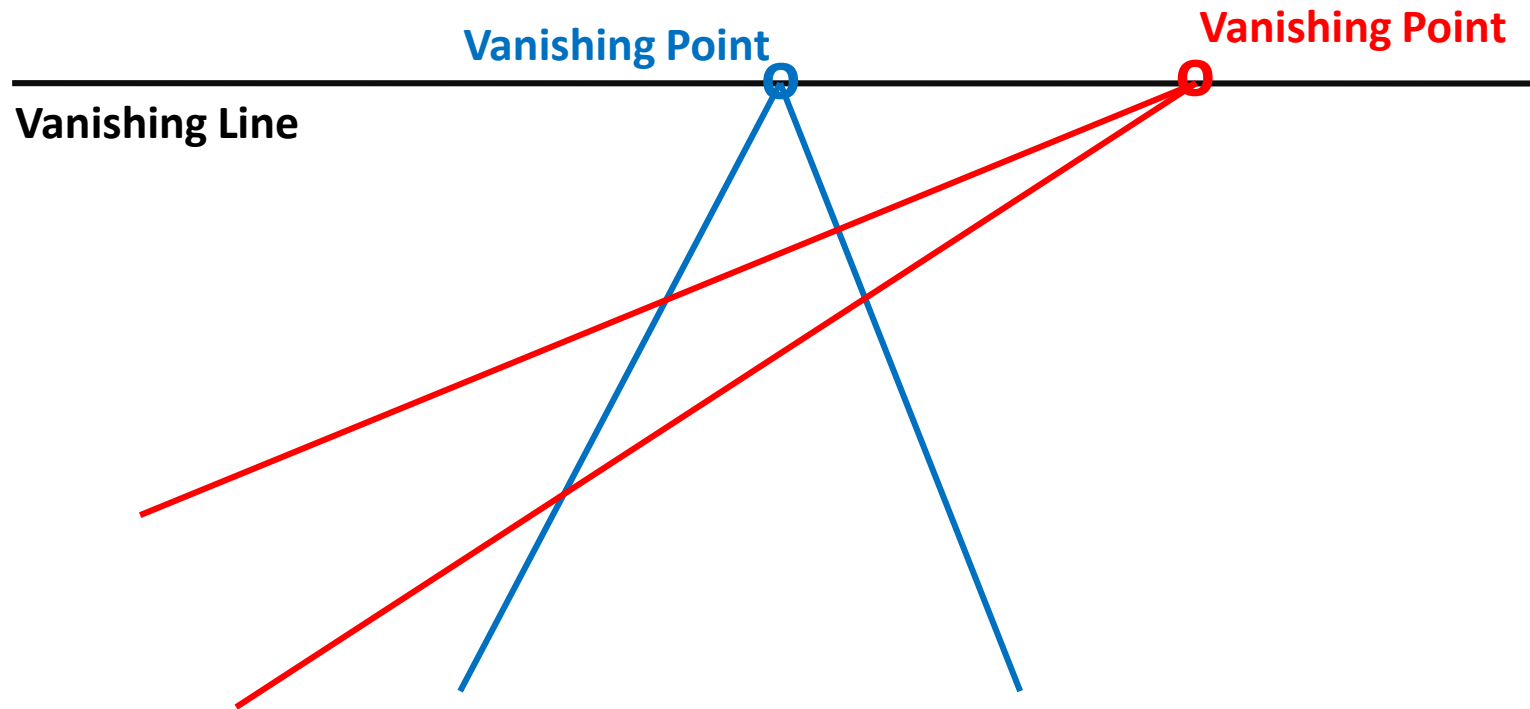


Vanishing points and lines

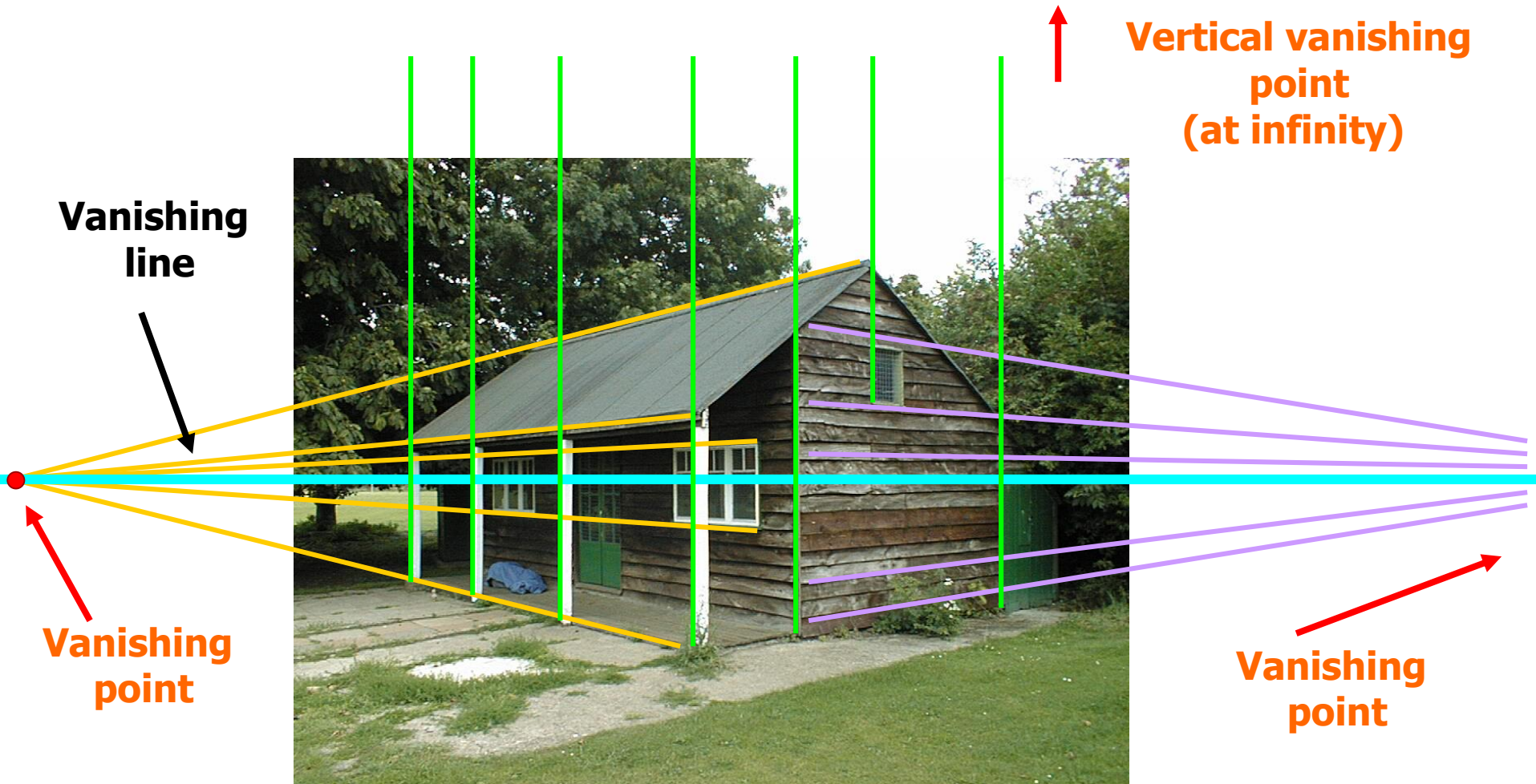
Parallel lines in the world intersect in the image at a “vanishing point”



Vanishing points and lines



Vanishing points and lines



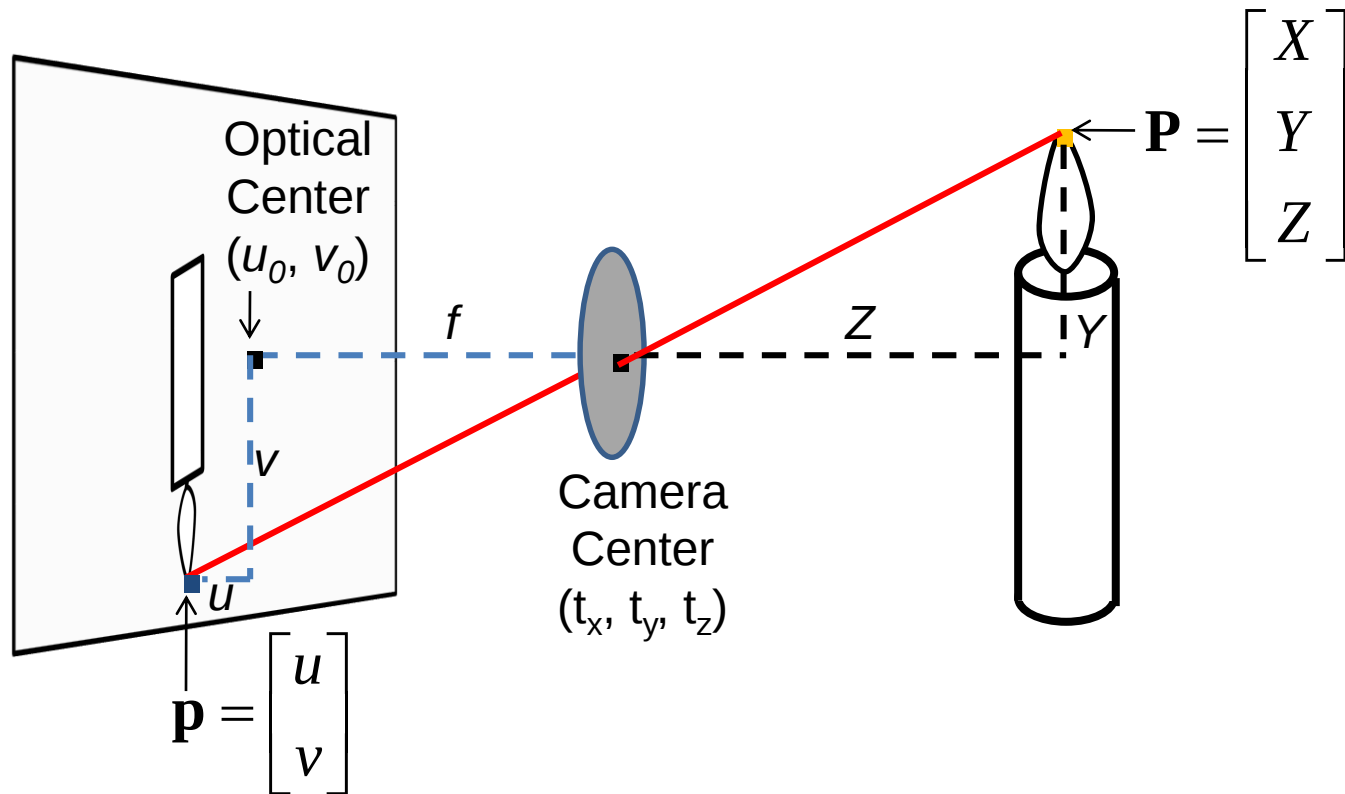
Vanishing points and lines



Vanishing objects



Projection: world coordinates $\rightarrow$ image coordinates



Homogeneous coordinates

Conversion

Converting to *homogeneous* coordinates

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

homogeneous image
coordinates

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

homogeneous scene
coordinates

Converting *from* homogeneous coordinates

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

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

Homogeneous coordinates

Invariant to scaling

$$k \begin{bmatrix} x \\ y \\ w \end{bmatrix} = \begin{bmatrix} kx \\ ky \\ kw \end{bmatrix} \Rightarrow \begin{bmatrix} \frac{kx}{kw} \\ \frac{ky}{kw} \\ \frac{kw}{kw} \end{bmatrix} = \begin{bmatrix} \frac{x}{w} \\ \frac{y}{w} \\ 1 \end{bmatrix}$$

Homogeneous
Coordinates

Cartesian
Coordinates

Point in Cartesian is ray in Homogeneous

Basic geometry in homogeneous coordinates

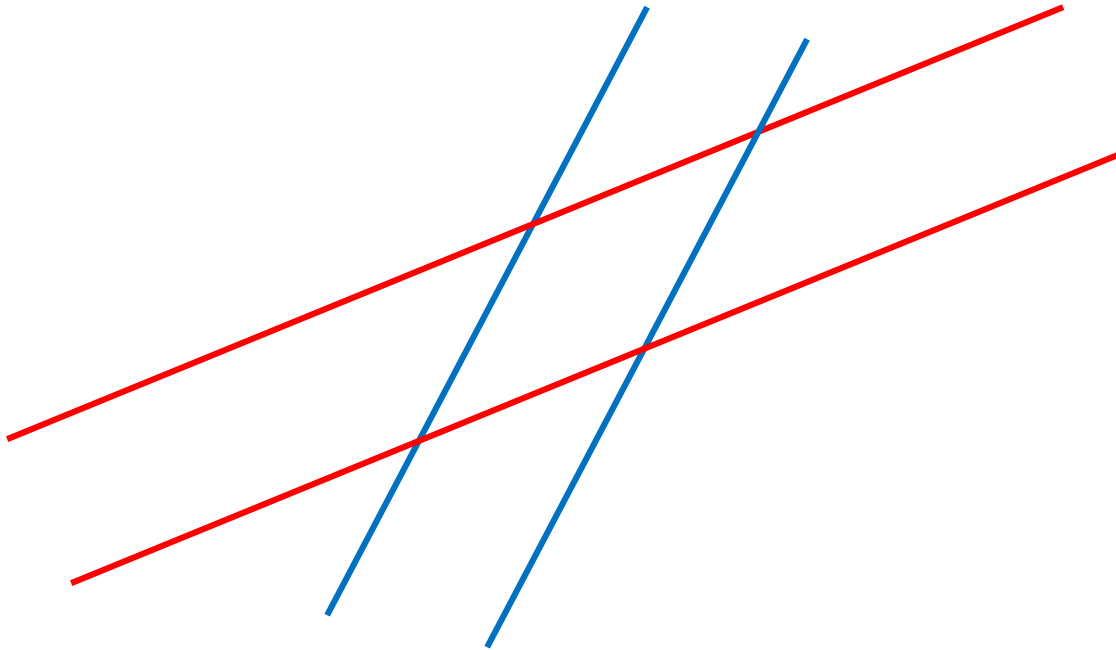
- Line equation: $ax + by + c = 0$ $line_i = \begin{bmatrix} a_i \\ b_i \\ c_i \end{bmatrix}$
- Append 1 to pixel coordinate to get homogeneous coordinate $p_i = \begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix}$
- Line given by cross product of two points $line_{ij} = p_i \times p_j$
- Intersection of two lines given by cross product of the lines $q_{ij} = line_i \times line_j$

Another problem solved by homogeneous coordinates

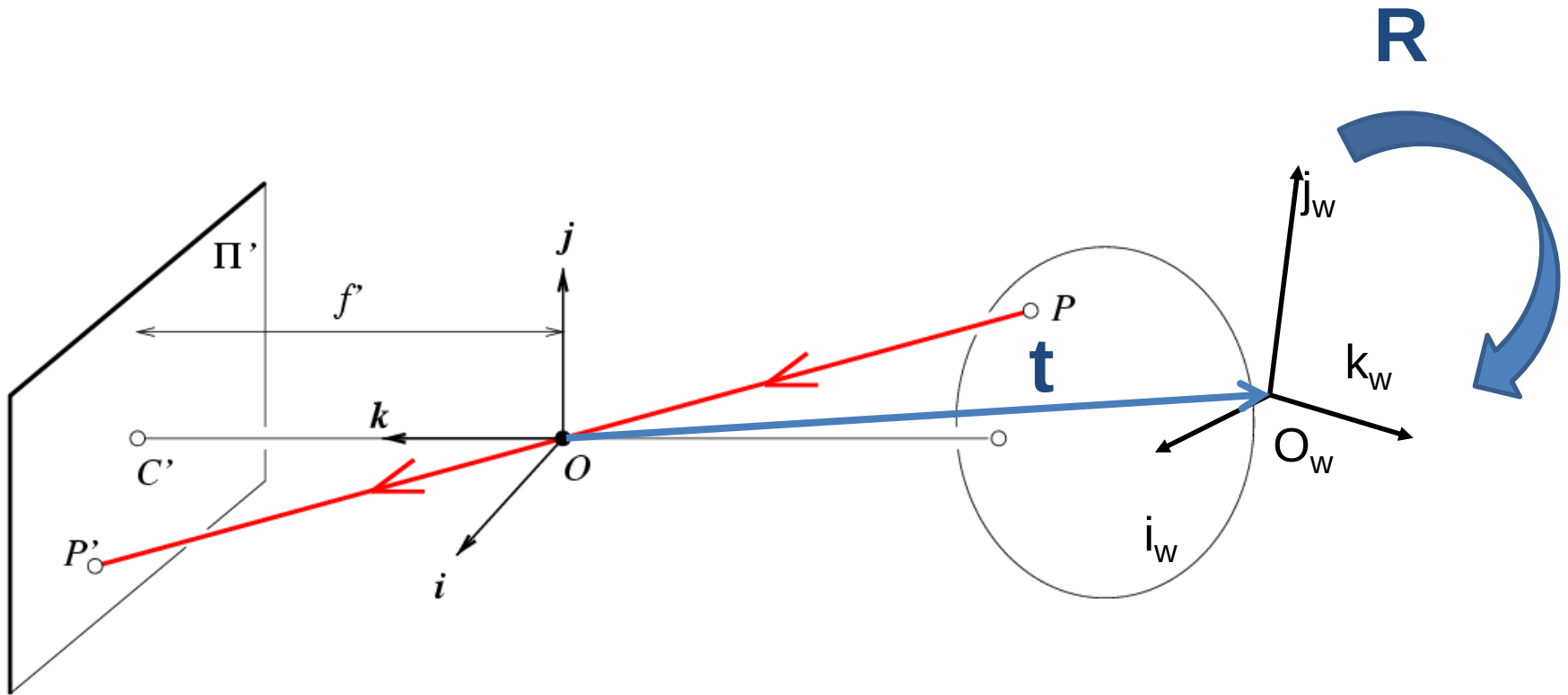
Intersection of parallel lines

Cartesian: (Inf, Inf)
Homogeneous: $(1, 1, 0)$

Cartesian: (Inf, Inf)
Homogeneous: $(1, 2, 0)$



Pinhole Camera Model



$$\mathbf{x} = \mathbf{K} \begin{bmatrix} \mathbf{R} & \mathbf{t} \end{bmatrix} \mathbf{X}$$

$\mathbf{x}$: Image Coordinates: $(u, v, 1)$

$\mathbf{K}$: Intrinsic Matrix (3x3)

$\mathbf{R}$: Rotation (3x3)

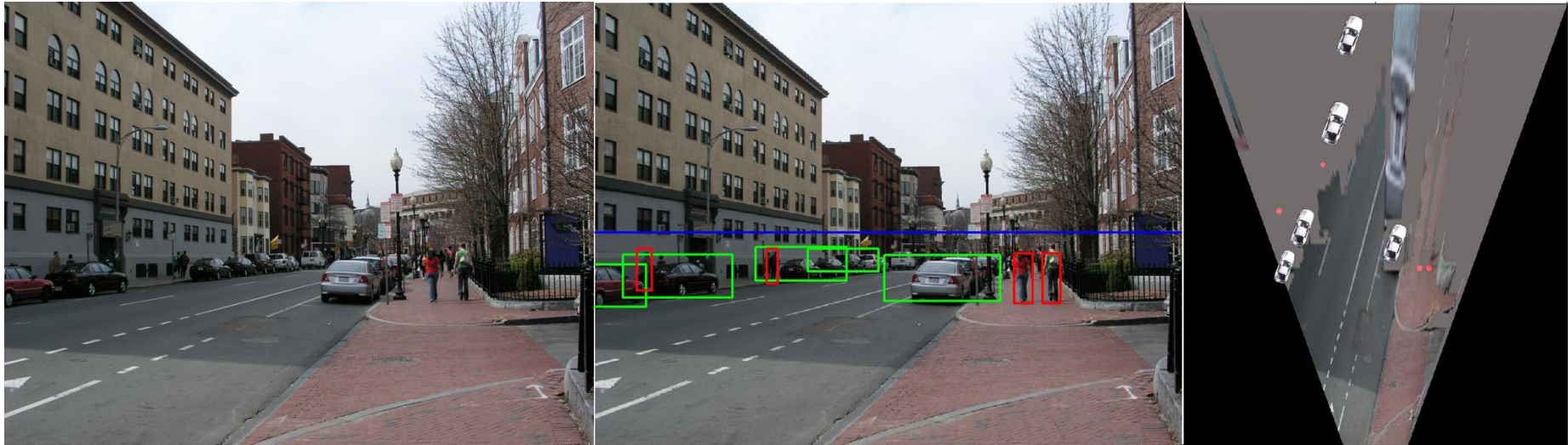
$\mathbf{t}$: Translation (3x1)

$\mathbf{X}$: World Coordinates: $(X, Y, Z, 1)$

Interlude: when have I used this stuff?

When have I used this stuff?

Object Recognition (CVPR 2006)



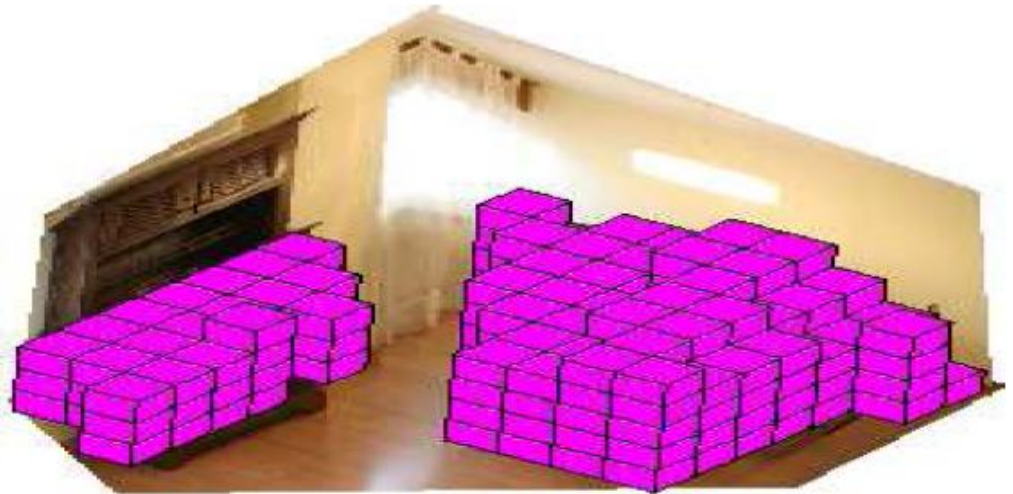
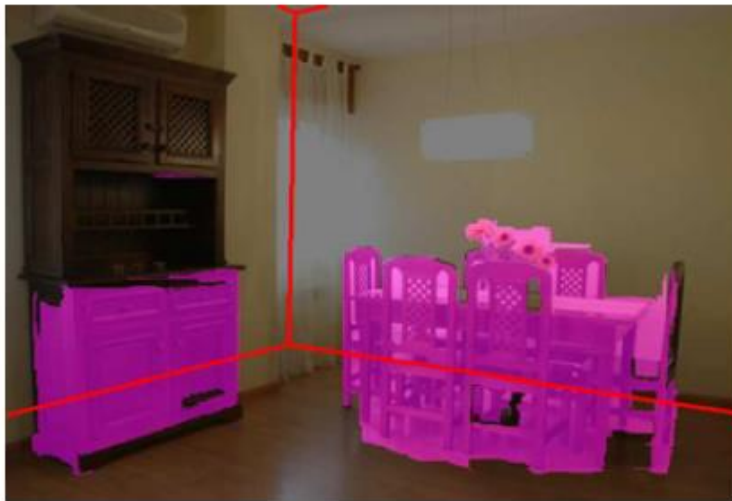
When have I used this stuff?

Single-view reconstruction (SIGGRAPH 2005)



When have I used this stuff?

Getting spatial layout in indoor scenes (ICCV 2009)



When have I used this stuff?

Inserting photographed objects into images
(SIGGRAPH 2007)



Original



Created

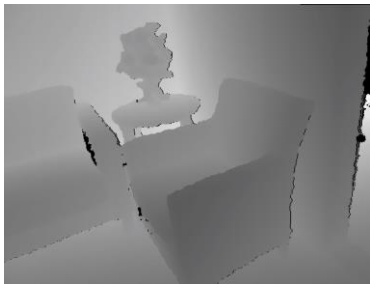
When have I used this stuff?

Inserting synthetic objects into images: <http://vimeo.com/28962540>

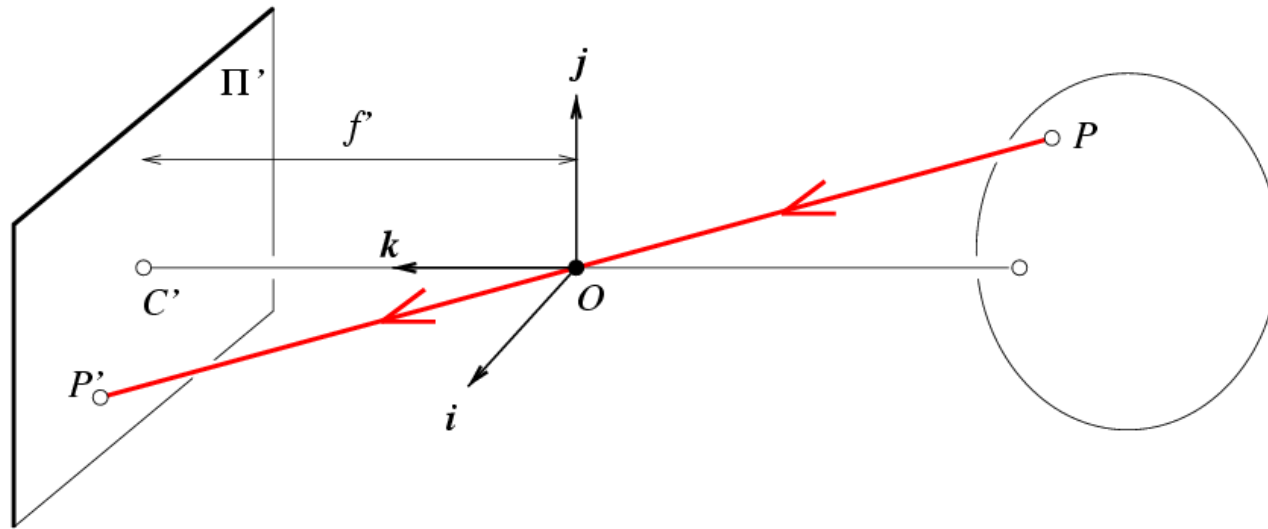


When have I used this stuff?

Creating detailed and complete 3D scene models from a single view (ongoing)



Projection matrix



Intrinsic Assumptions

- Unit aspect ratio
- Principal point at $(0,0)$
- No skew

Extrinsic Assumptions

- No rotation
- Camera at $(0,0,0)$

$$\mathbf{x} = \mathbf{K} \begin{bmatrix} \mathbf{I} & \mathbf{0} \end{bmatrix} \mathbf{X} \Rightarrow {}^w \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

The matrix $\mathbf{K}$ is highlighted with a red dashed box and a red arrow pointing to it from the label $\mathbf{K}$.

Remove assumption: known optical center

Intrinsic Assumptions

- Unit aspect ratio
- No skew

Extrinsic Assumptions

- No rotation
- Camera at (0,0,0)

$$\mathbf{x} = \mathbf{K} \begin{bmatrix} \mathbf{I} & \mathbf{0} \end{bmatrix} \mathbf{X} \Rightarrow w \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & u_0 \\ 0 & f & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Remove assumption: square pixels

Intrinsic Assumptions

- No skew

Extrinsic Assumptions

- No rotation
- Camera at (0,0,0)

$$\mathbf{x} = \mathbf{K} \begin{bmatrix} \mathbf{I} & \mathbf{0} \end{bmatrix} \mathbf{X} \Rightarrow w \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha & 0 & u_0 & 0 \\ 0 & \beta & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Remove assumption: non-skewed pixels

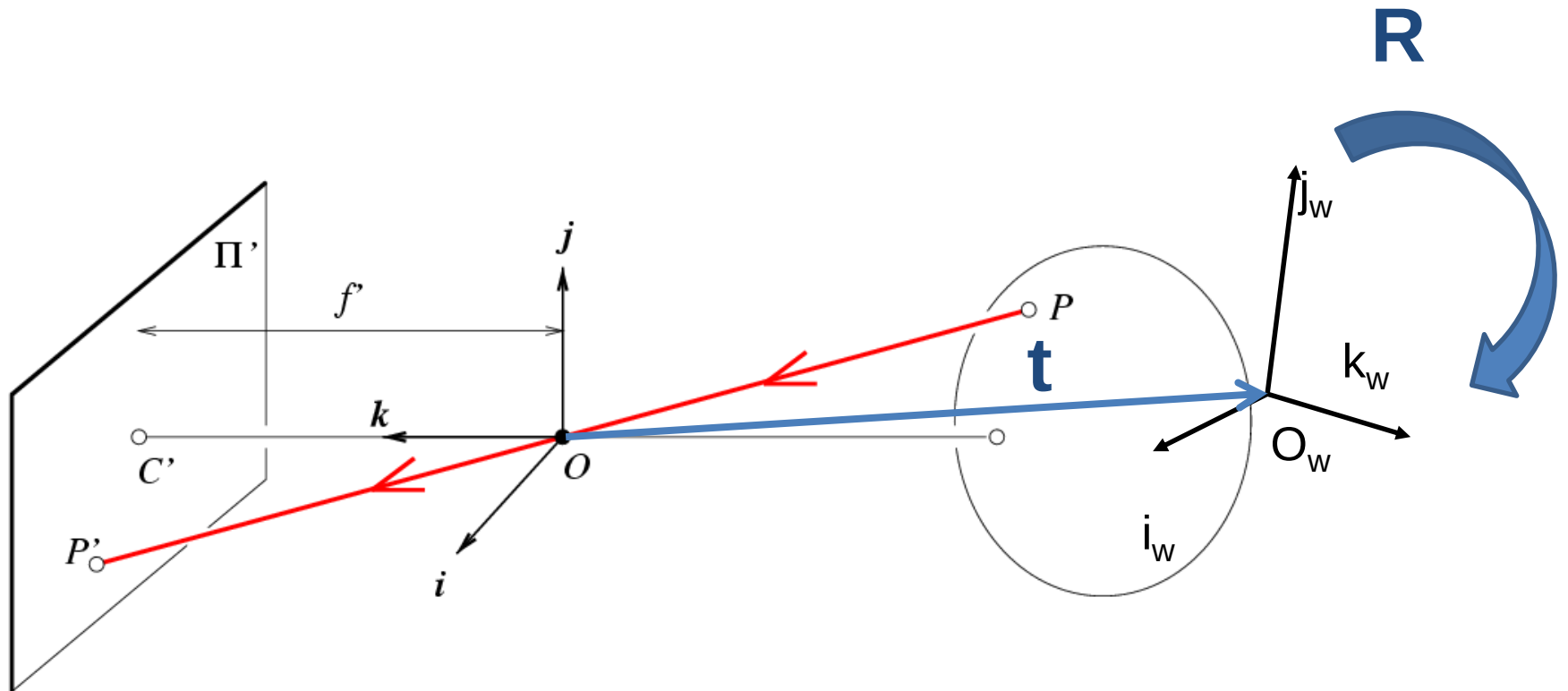
Intrinsic Assumptions Extrinsic Assumptions

- No rotation
- Camera at (0,0,0)

$$\mathbf{x} = \mathbf{K} \begin{bmatrix} \mathbf{I} & \mathbf{0} \end{bmatrix} \mathbf{X} \Rightarrow w \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha & s & u_0 & 0 \\ 0 & \beta & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Note: different books use different notation for parameters

Oriented and Translated Camera



Allow camera translation

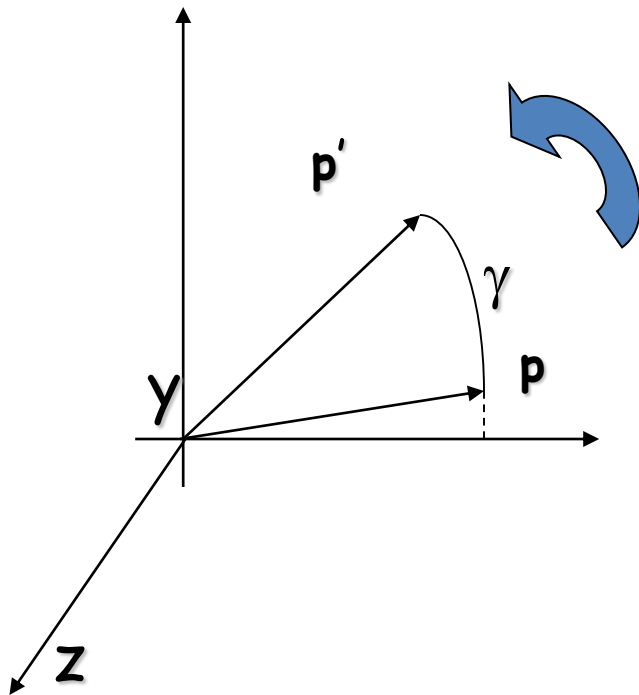
Intrinsic Assumptions Extrinsic Assumptions

- No rotation

$$\mathbf{x} = \mathbf{K} \begin{bmatrix} \mathbf{I} & \mathbf{t} \end{bmatrix} \mathbf{X} \Rightarrow w \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha & 0 & u_0 \\ 0 & \beta & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

3D Rotation of Points

Rotation around the coordinate axes, **counter-clockwise**:



$$R_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \alpha & -\sin \alpha \\ 0 & \sin \alpha & \cos \alpha \end{bmatrix}$$

$$R_y(\beta) = \begin{bmatrix} \cos \beta & 0 & \sin \beta \\ 0 & 1 & 0 \\ -\sin \beta & 0 & \cos \beta \end{bmatrix}$$

$$R_z(\gamma) = \begin{bmatrix} \cos \gamma & -\sin \gamma & 0 \\ \sin \gamma & \cos \gamma & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Allow camera rotation

$$\mathbf{x} = \mathbf{K}[\mathbf{R} \quad \mathbf{t}] \mathbf{X}$$



$$w \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha & s & u_0 \\ 0 & \beta & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Degrees of freedom

$$\mathbf{x} = \mathbf{K} \begin{bmatrix} \mathbf{R} & \mathbf{t} \end{bmatrix} \mathbf{X}$$



$${}^w \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \overset{5}{\begin{bmatrix} \alpha & s & u_0 \\ 0 & \beta & v_0 \\ 0 & 0 & 1 \end{bmatrix}} \overset{6}{\begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix}} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

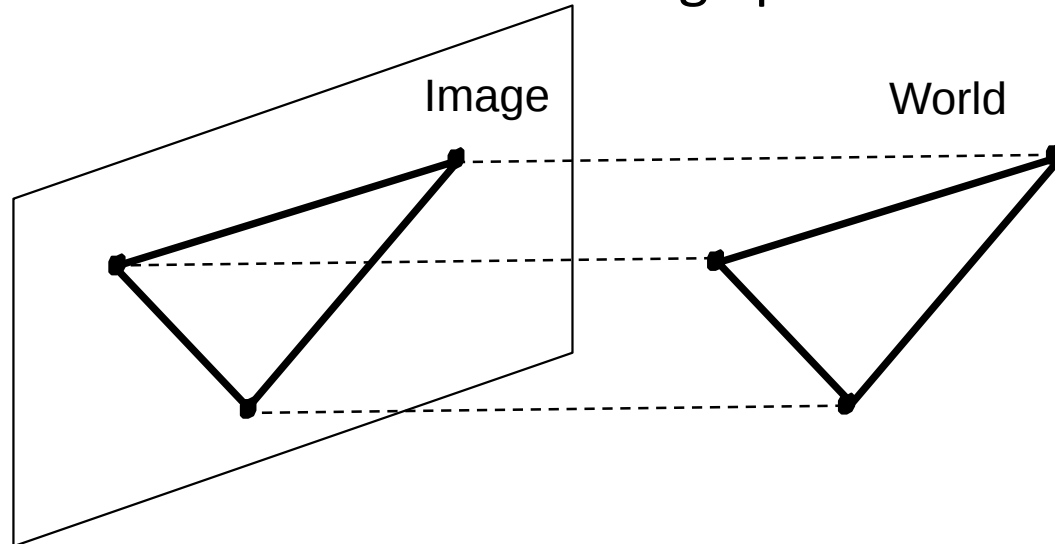
Vanishing Point = Projection from Infinity

$$\mathbf{p} = \mathbf{K}[\mathbf{R} \quad \mathbf{t}] \begin{bmatrix} x \\ y \\ z \\ 0 \end{bmatrix} \Rightarrow \mathbf{p} = \mathbf{K}\mathbf{R} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \Rightarrow \mathbf{p} = \mathbf{K} \begin{bmatrix} x_R \\ y_R \\ z_R \end{bmatrix}$$

$$w \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & u_0 \\ 0 & f & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_R \\ y_R \\ z_R \end{bmatrix} \Rightarrow \begin{aligned} u &= \frac{fx_R}{z_R} + u_0 \\ v &= \frac{fy_R}{z_R} + v_0 \end{aligned}$$

Orthographic Projection

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

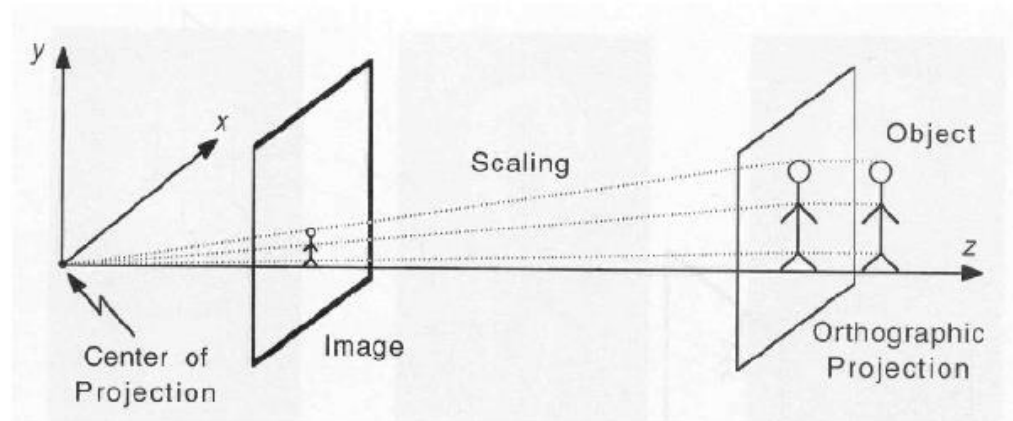


- Also called “parallel projection”
- What’s the projection matrix?

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Scaled Orthographic Projection

- Special case of perspective projection
 - Object dimensions are small compared to distance to camera



- Also called “weak perspective”
- What’s the projection matrix?

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 0 & s \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

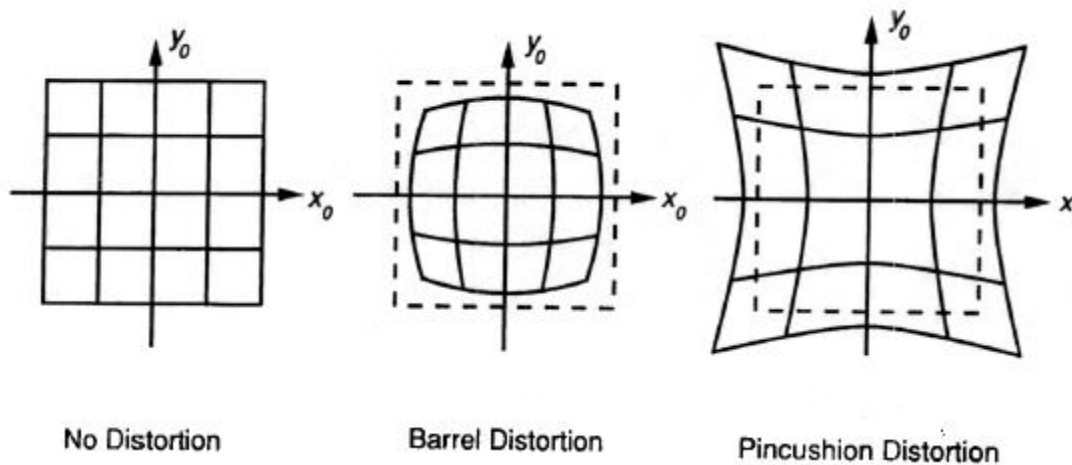
Take-home question

Suppose we have two 3D cubes on the ground facing the viewer, one near, one far.

1. What would they look like in perspective?
2. What would they look like in weak perspective?



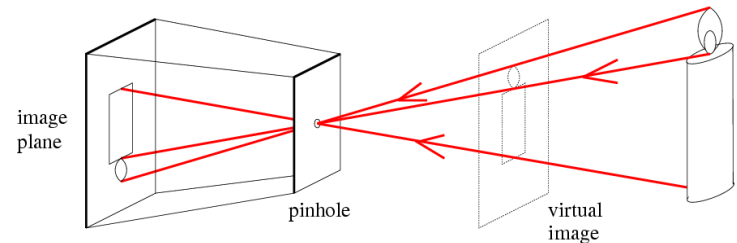
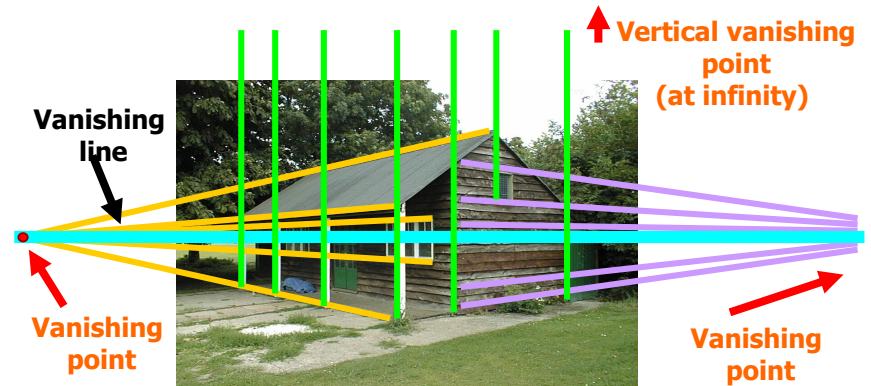
Beyond Pinholes: Radial Distortion



Corrected Barrel Distortion

Things to remember

- Vanishing points and vanishing lines
- Pinhole camera model and camera projection matrix



$$\mathbf{x} = \mathbf{K}[\mathbf{R} \quad \mathbf{t}] \mathbf{X}$$

Next classes

- Tues: single-view metrology
 - Measuring 3D distances from the image
- Thurs: single-view 3D reconstruction