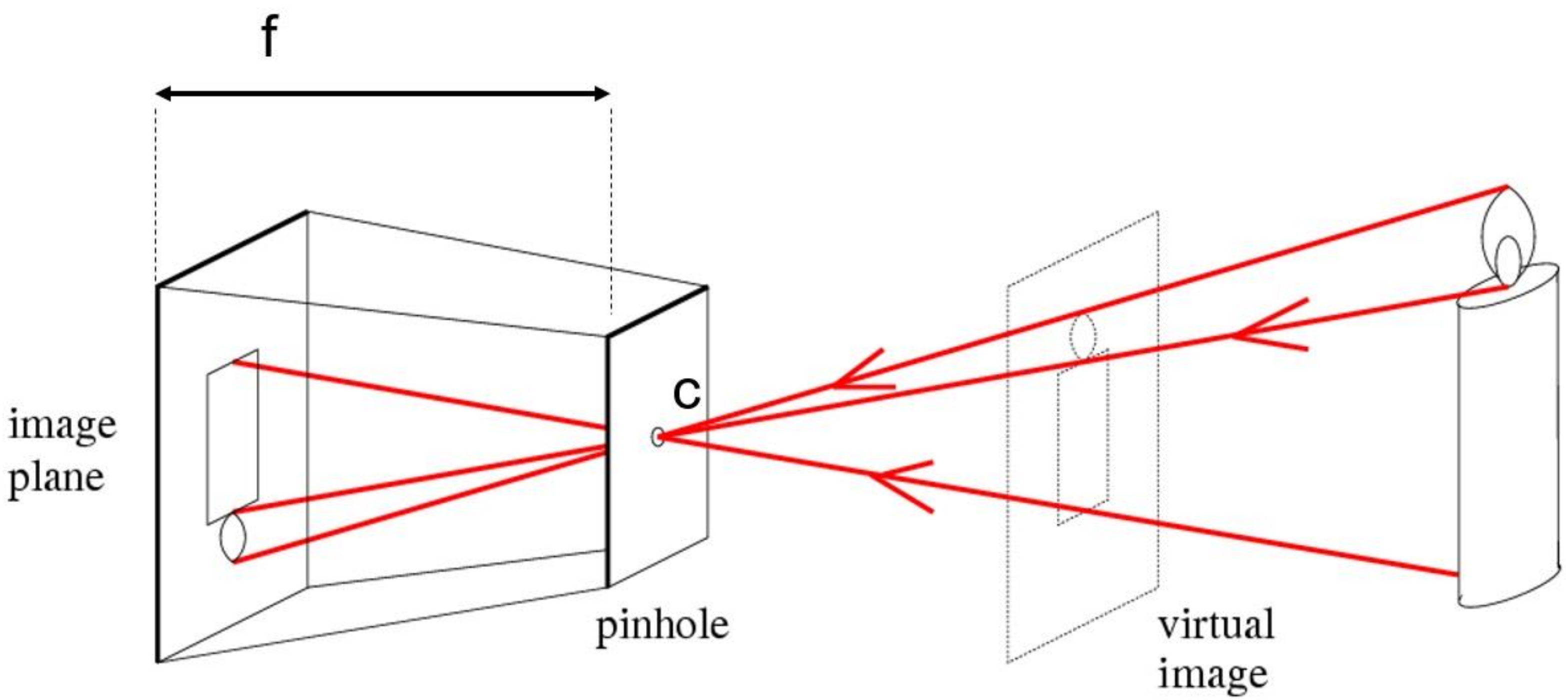
Projective Geometry and Camera Models



Computer Vision

Jia-Bin Huang, Virginia Tech

Many slides from S. Seitz and D. Hoiem

Administrative stuffs

- HWs
  - HW 1 is back
  - HW 2 due 11:59 PM on Oct 3rd.
  - Frequently asked questions for HW 2
  - Partial credits
- Think about your final projects
  - work in groups of 2-4
  - should evince independent effort to learn about a new topic, try something new, or apply to an application of interest
  - Proposals will be due Oct 27

Top edge methods – average F-score

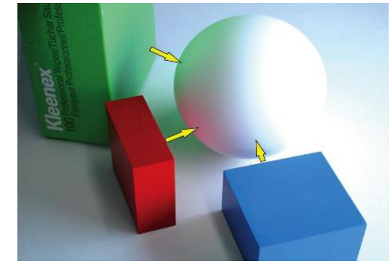
- Subhashree (0.673)
  - Canny for different channels with oriented filters
- Shruti Phadke (0.66)
  - Simple Gradient, colour channel splitting
- Ben Zhao (0.658)
  - Oriented Elongated Gaussian

Excellent HW 1 report:

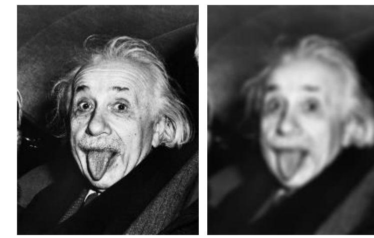
- Badour AlBahar

Review: Interpreting Intensity

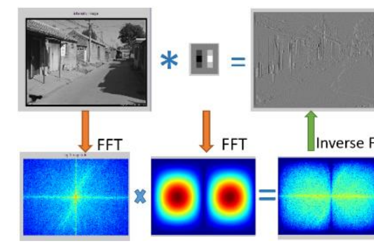
- **Light and color**
  - –What an image records



- **Filtering in spatial domain**
  - Filtering = weighted sum of neighboring pixels
  - Smoothing, sharpening, measuring texture

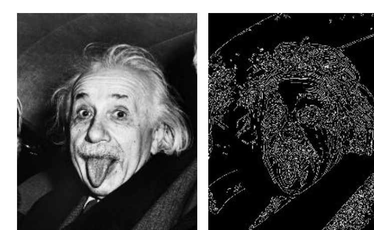


- **Filtering in frequency domain**
  - Filtering = change frequency of the input image
  - Denoising, sampling, image compression



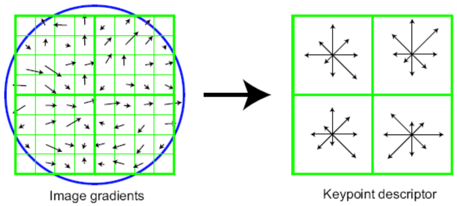
- **Image pyramid and template matching**
  - Filtering = a way to find a template
  - Image pyramids for coarse-to-fine search and multi-scale detection

- **Edge detection**
  - Canny edge = smooth -> derivative -> thin -> threshold -> link
  - Finding straight lines, binary image analysis

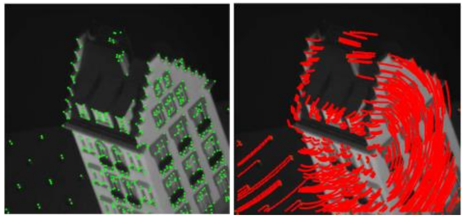


Review: Correspondence and Alignment

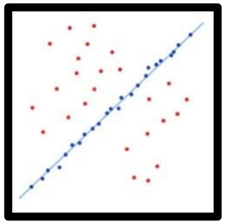
- **Interest points**
  - Find *distinct* and *repeatable* points in images
  - Harris-> corners, DoG -> blobs
  - SIFT -> feature descriptor



- **Feature tracking and optical flow**
  - Find motion of a keypoint/pixel over time
  - Lucas-Kanade:
    - brightness consistency, small motion, spatial coherence
  - Handle large motion:
    - iterative update + pyramid search



- **Fitting and alignment**
  - find the transformation parameters that best align matched points

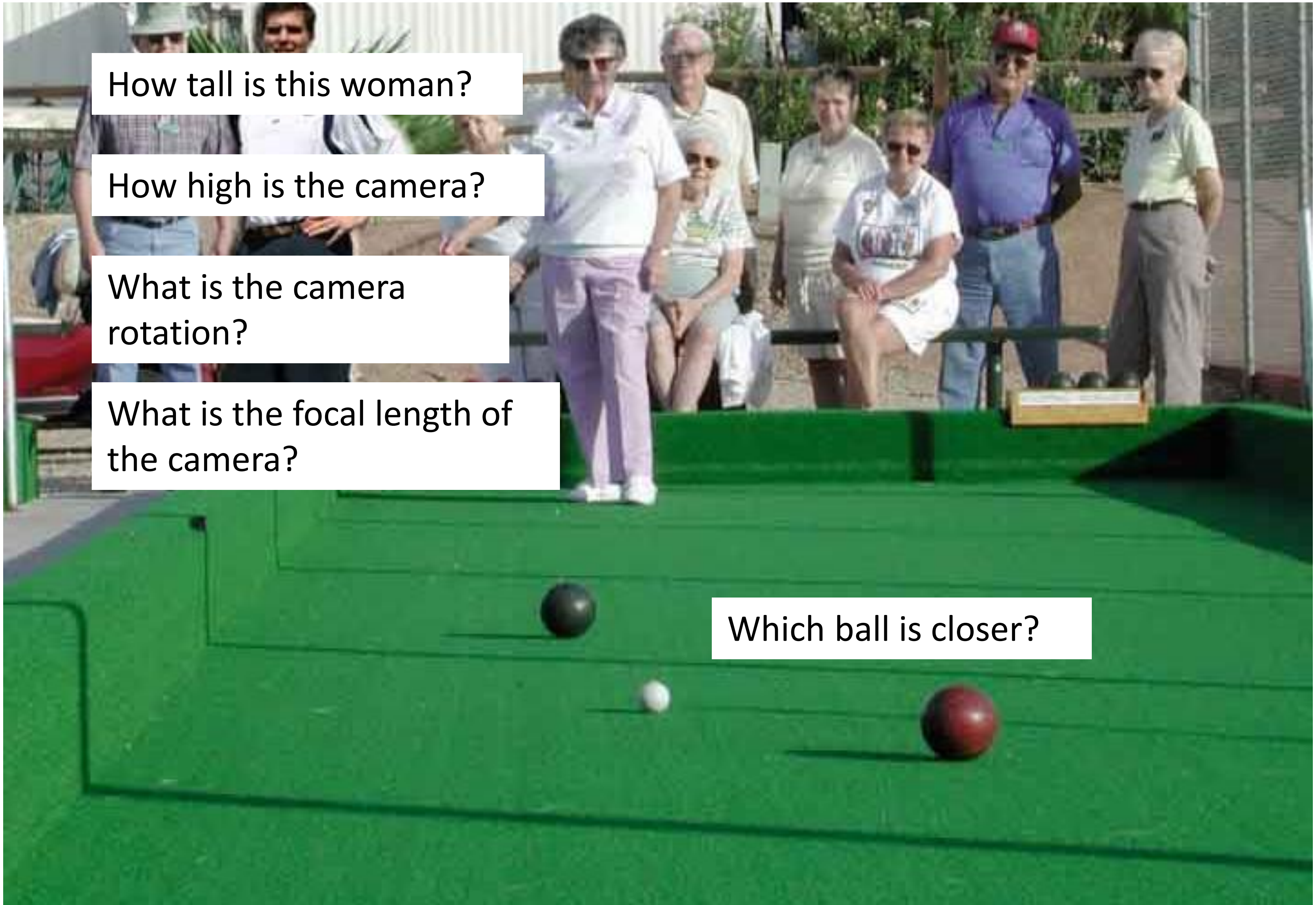


- **Object instance recognition**
  - Keypoint-based object instance recognition and search

Perspective and 3D Geometry

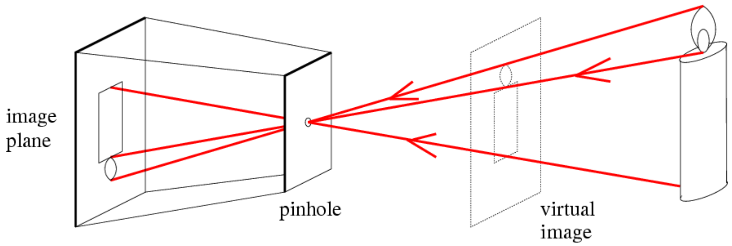
- **Projective geometry and camera models**
  - What’s the mapping between image and world coordiantes?
- **Single view metrology and camera calibration**
  - How can we measure the size of 3D objects in an image?
  - How can we estimate the camera parameters?
- **Photo stitching**
  - What’s the mapping from two images taken without camera translation?
- **Epipolar Geometry and Stereo Vision**
  - What’s the mapping from two images taken with camera translation?
- **Structure from motion**
  - How can we recover 3D points from multiple images?

Next two classes: Single-view Geometry

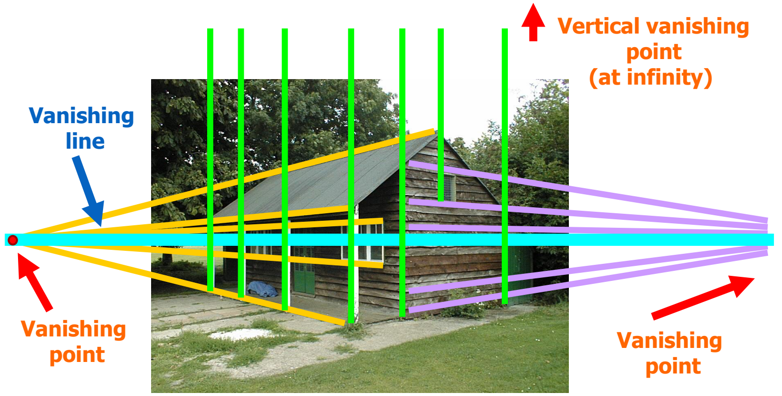


Today's class
Mapping between image and world coordinates

- Pinhole camera model



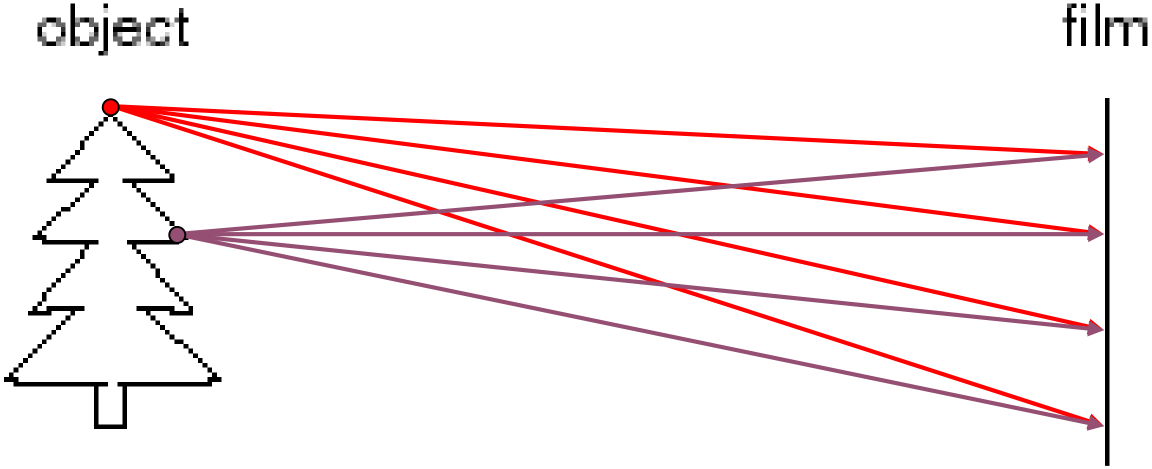
- Projective geometry
    - Vanishing points and lines



- Projection matrix

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

Image formation



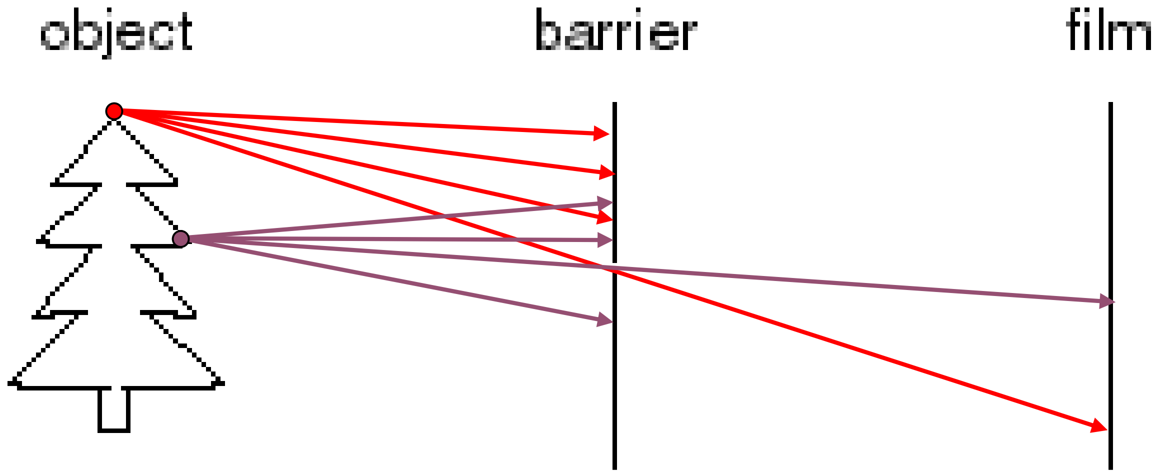
Let’s design a camera

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

–Do we get a reasonable image?

Slide source: Seitz

Pinhole camera

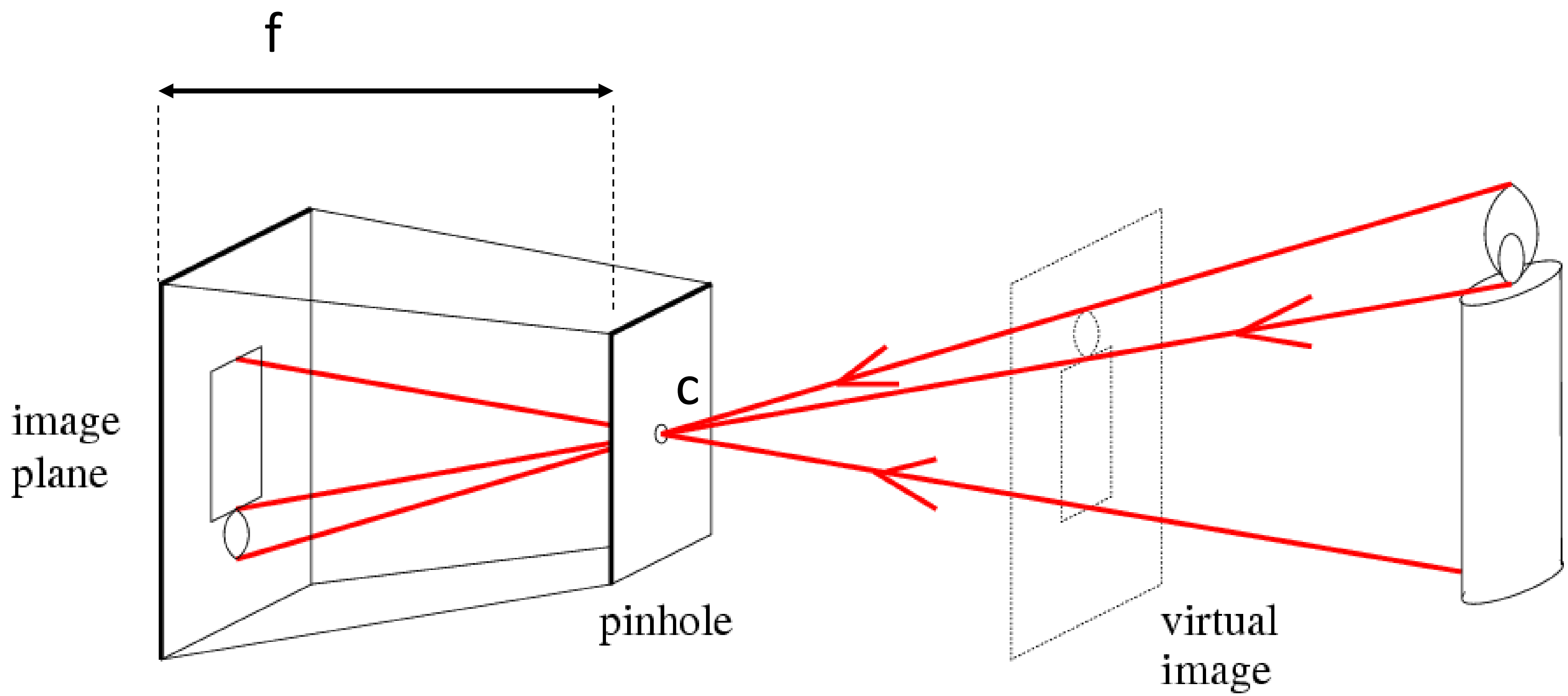


Idea 2: add a barrier to block off most of the rays

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

Slide source: Seitz

Pinhole camera



f = focal length

c = center of the camera

Figure from Forsyth

Camera obscura: the pre-camera

- First idea: Mo-Ti, China (470BC to 390BC)
- First built: Alhazen, Iraq/Egypt (965 to 1039AD)

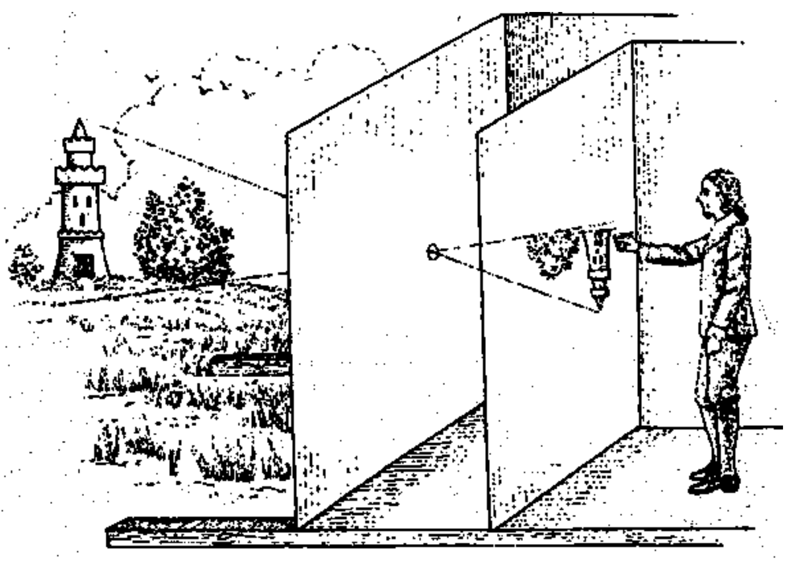
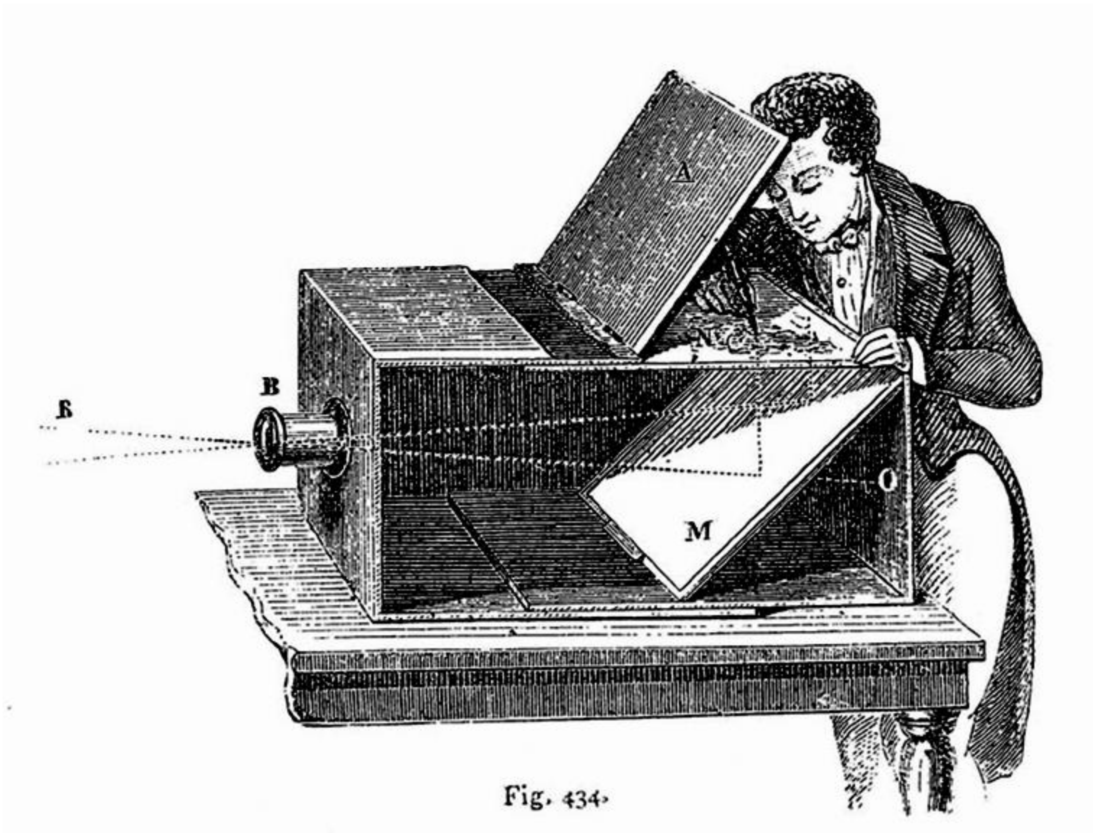


Illustration of Camera Obscura

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

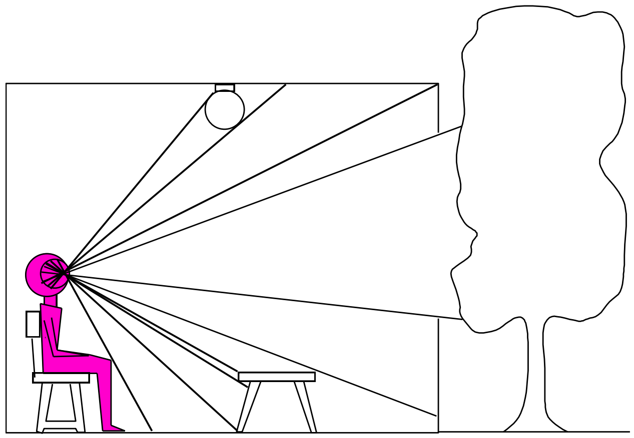
Stored at UT Austin

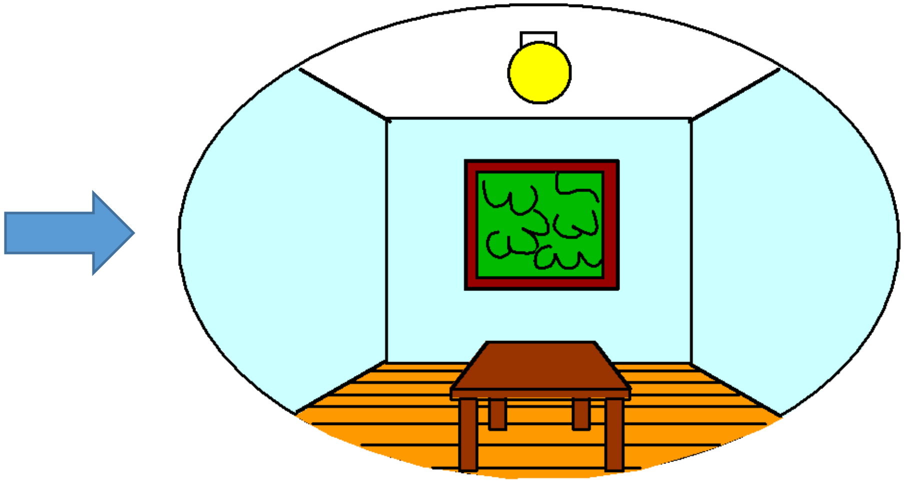
Niepce later teamed up with Daguerre, who eventually created Daguerrotypes

Dimensionality Reduction Machine (3D to 2D)

3D world

2D image





Point of observation

Figures © Stephen E. Palmer, 2002

Slide source: Seitz

Projection can be tricky…

Slide source: Seitz

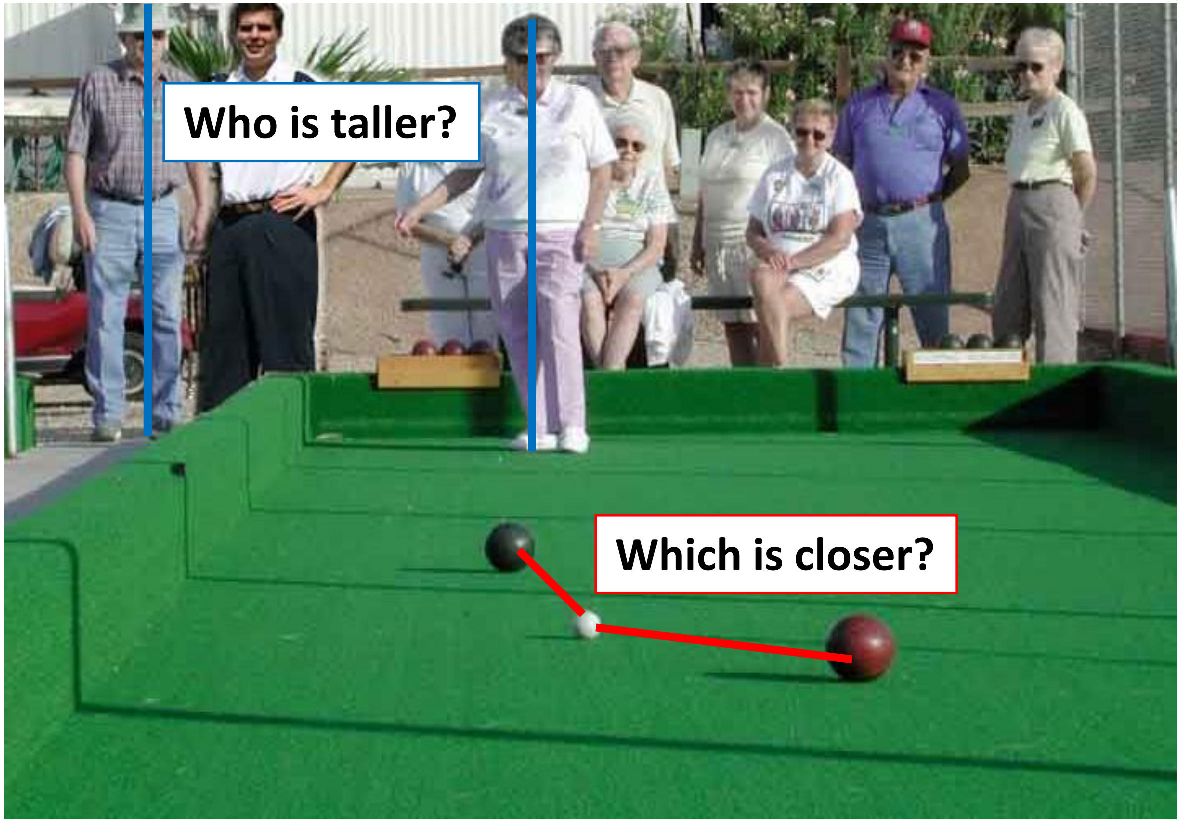
Projection can be tricky…

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

Projective Geometry

What is lost?

- Length



Length is not preserved

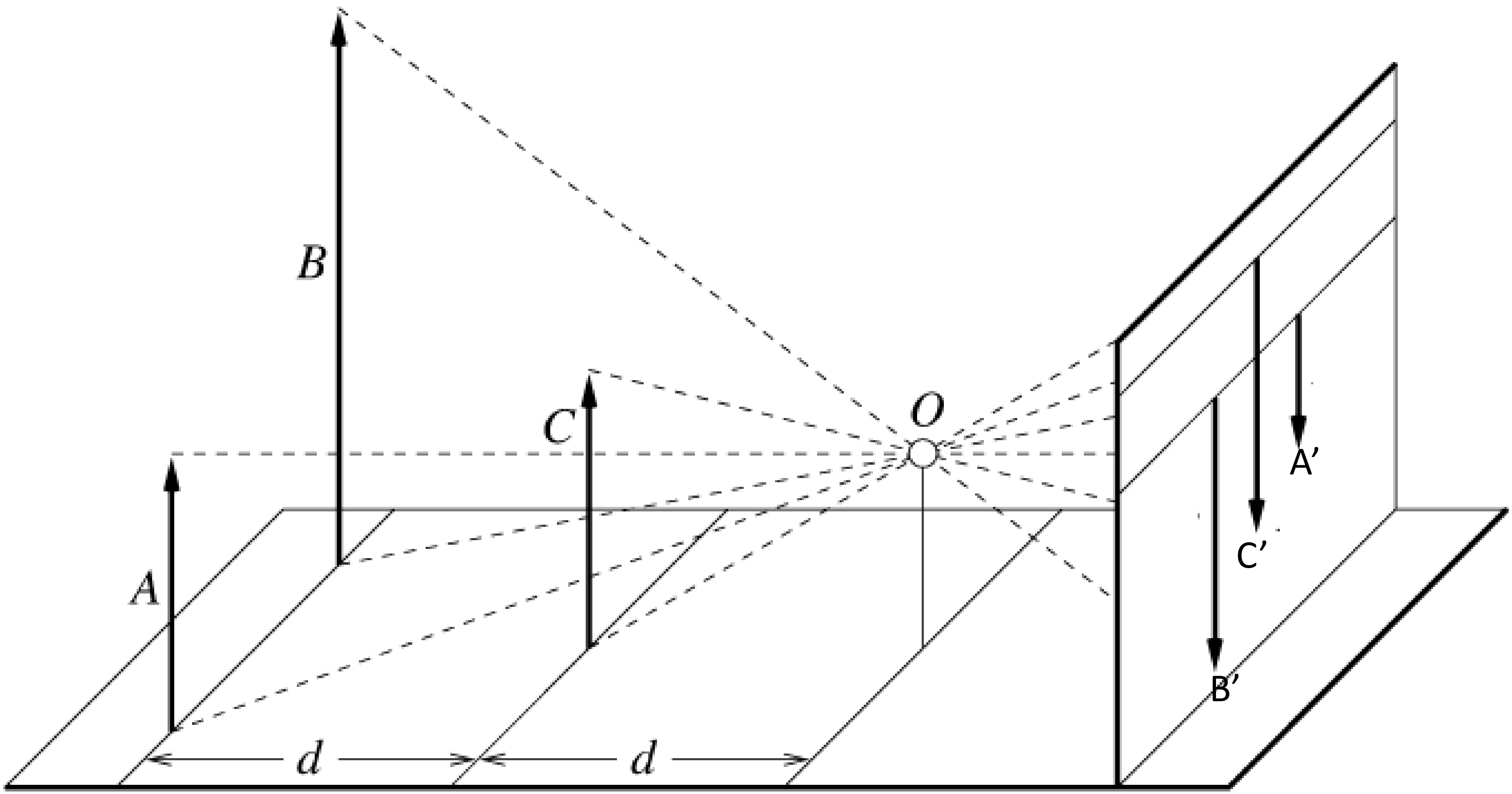
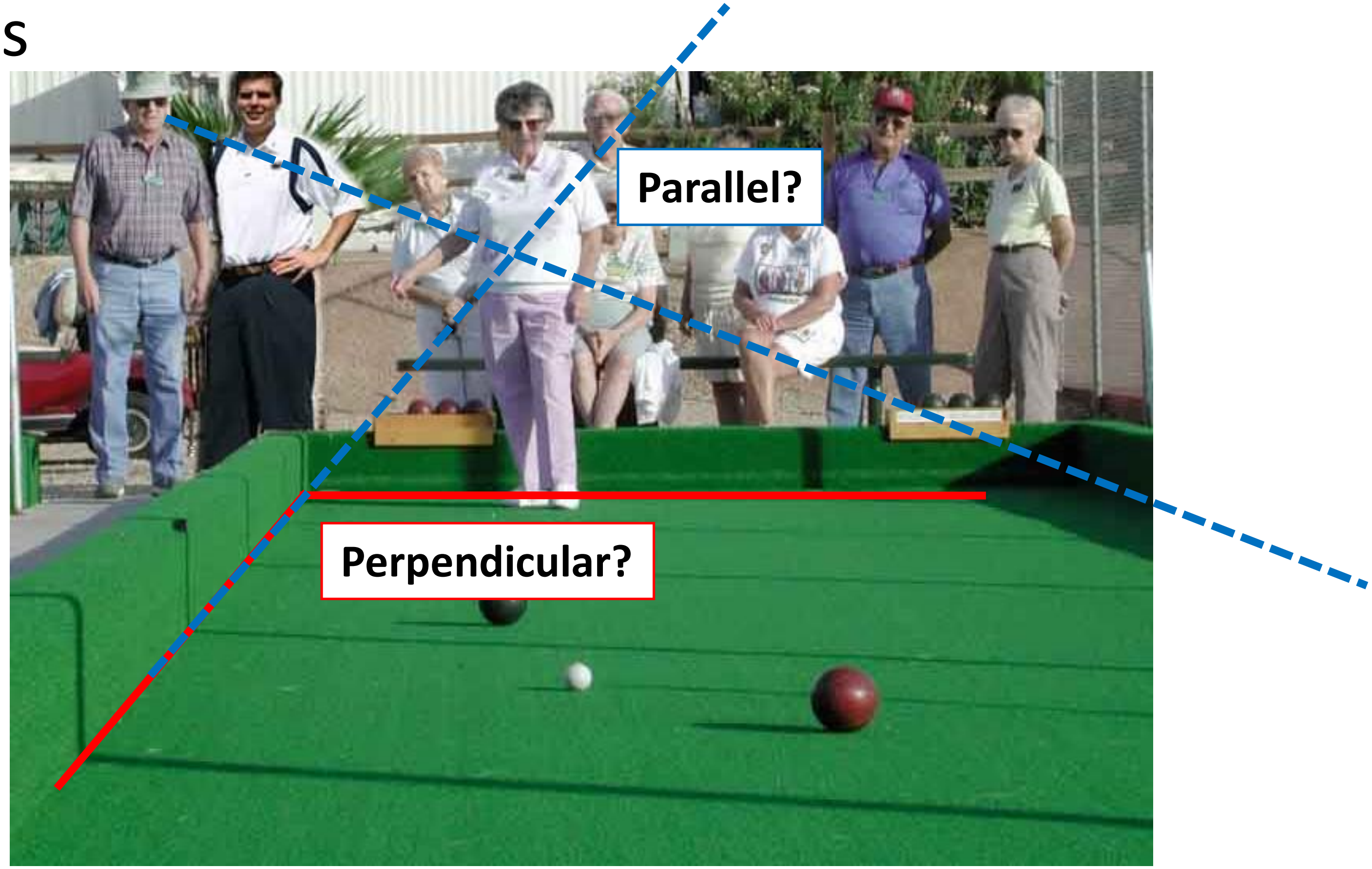


Figure by David Forsyth

Projective Geometry

What is lost?

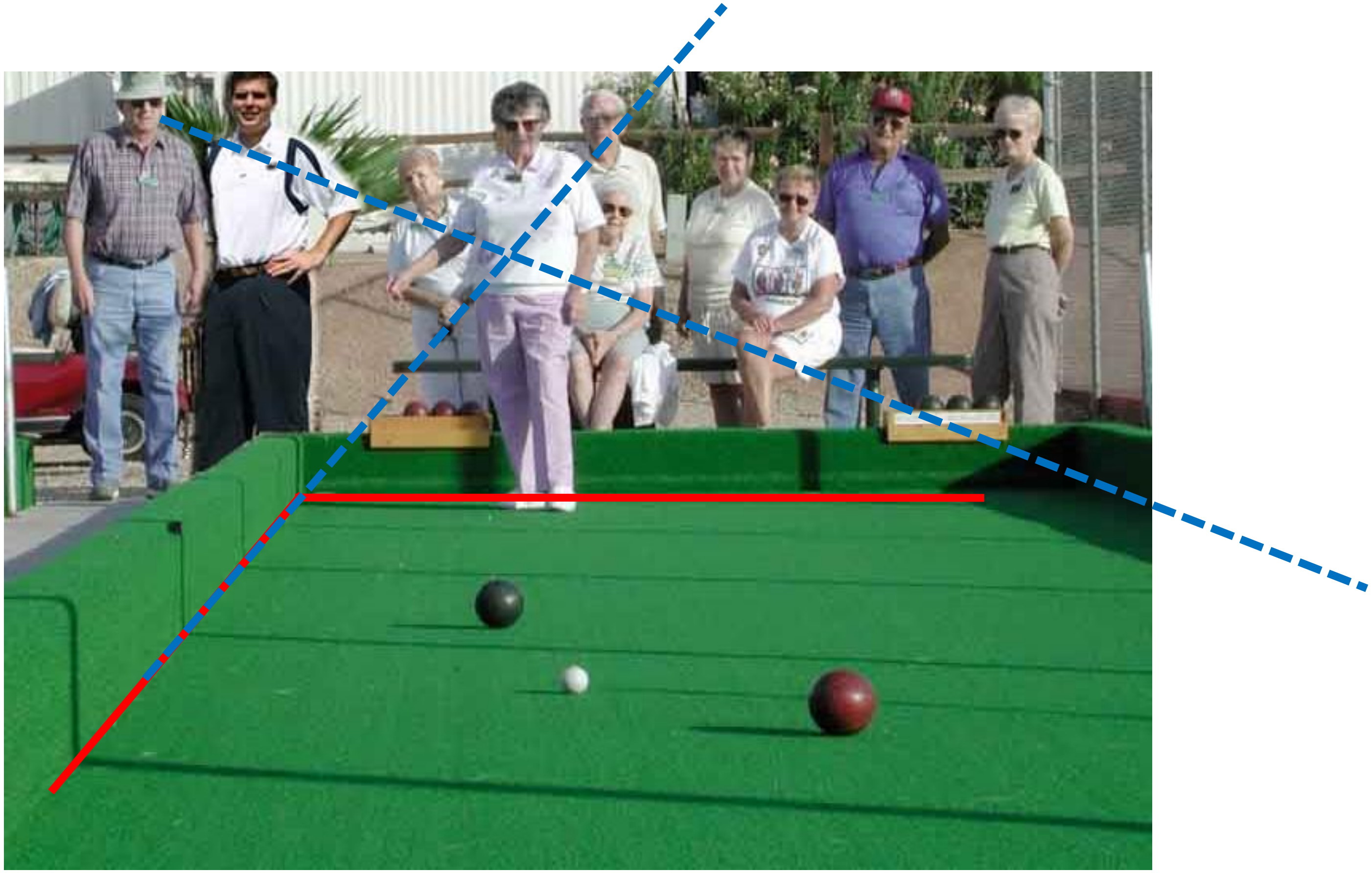
- Length
- Angles



Projective Geometry

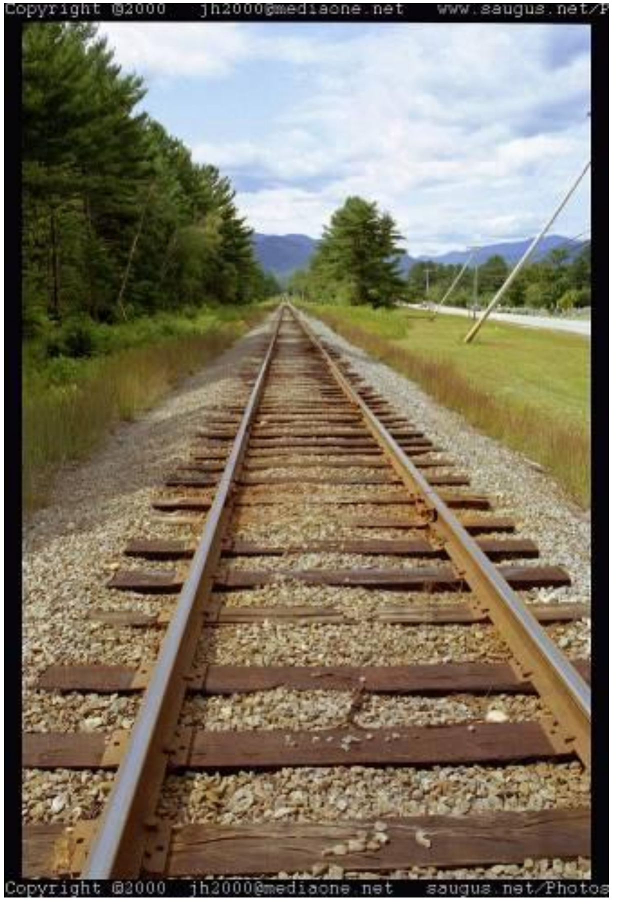
What is preserved?

- Straight lines are still straight

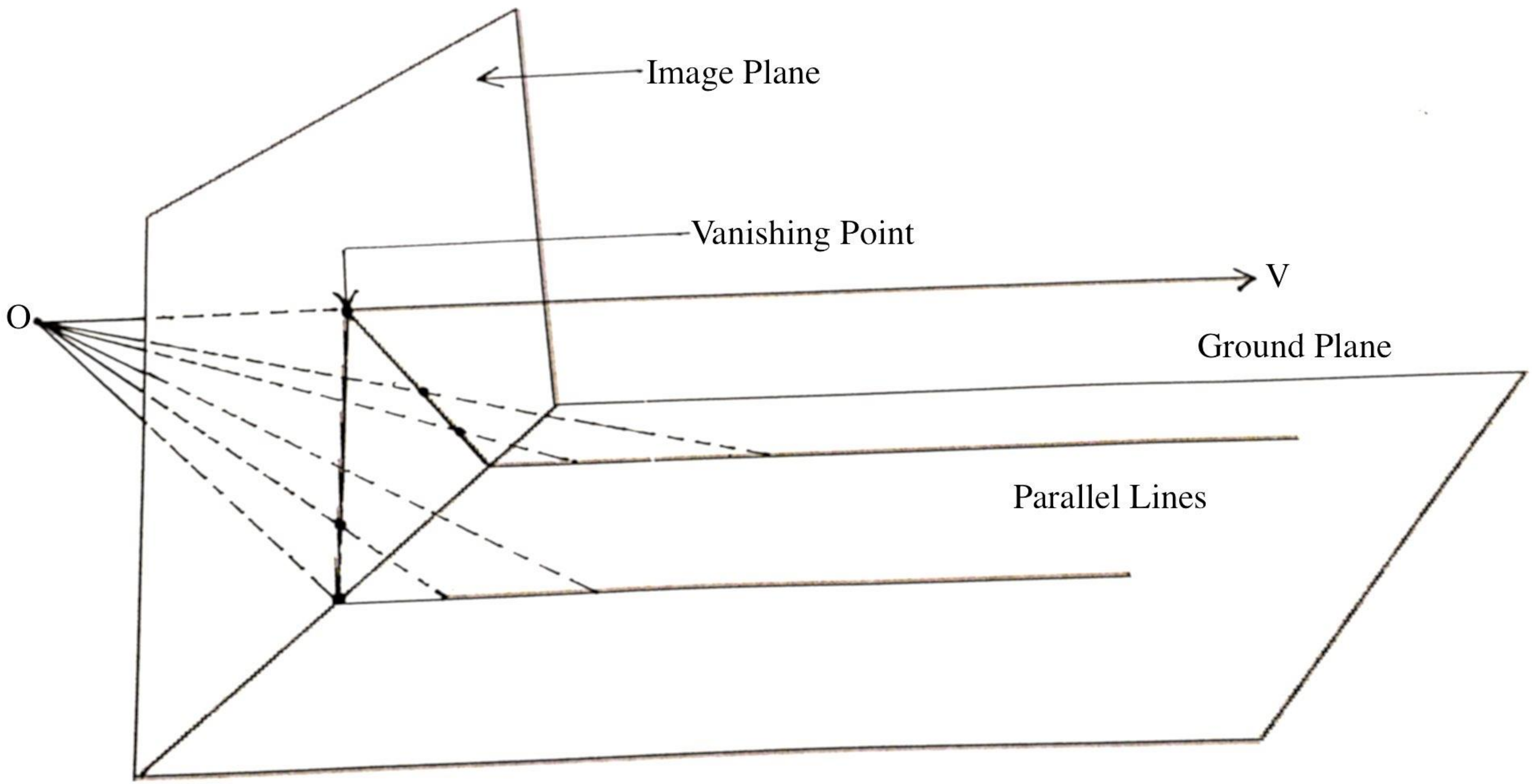


Vanishing points and lines

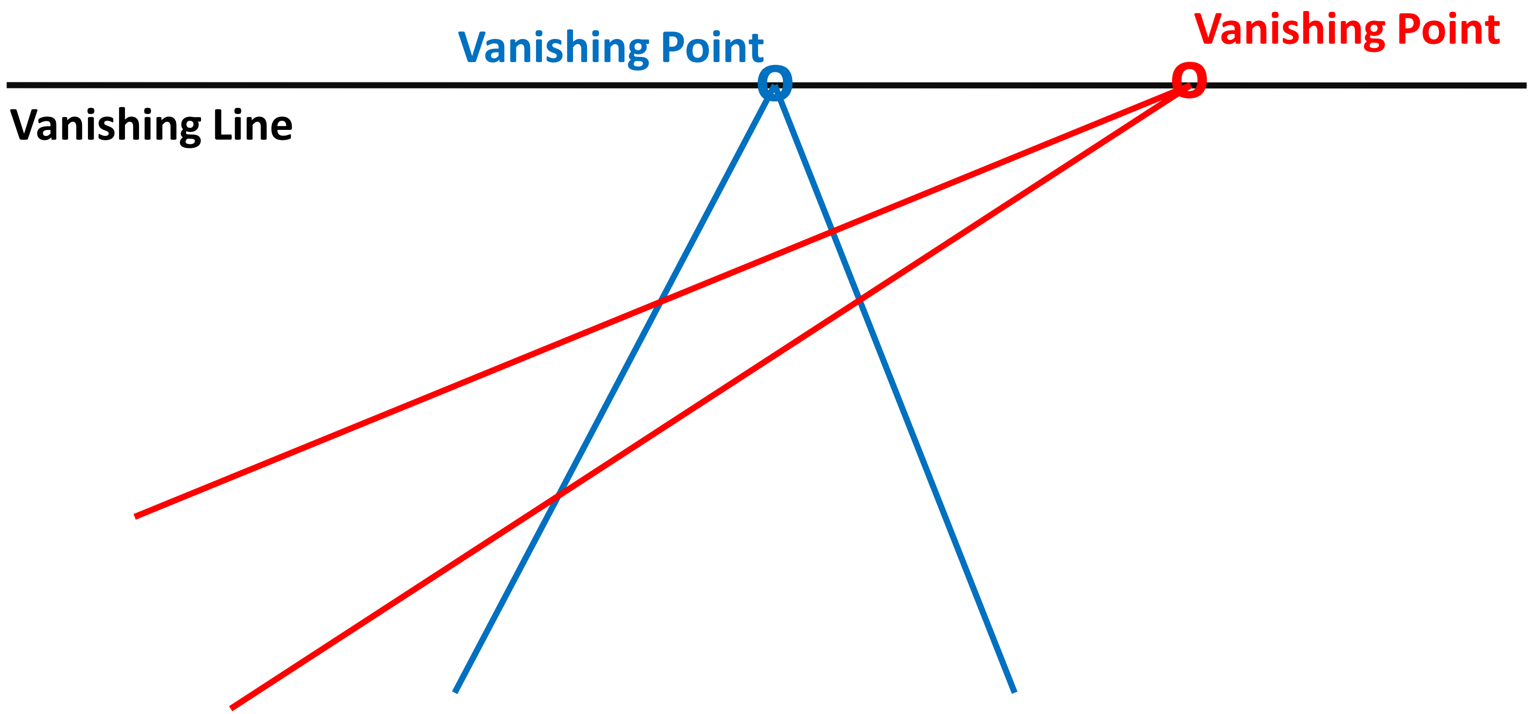
Parallel lines in the world intersect in the image at a "vanishing point"



Vanishing points

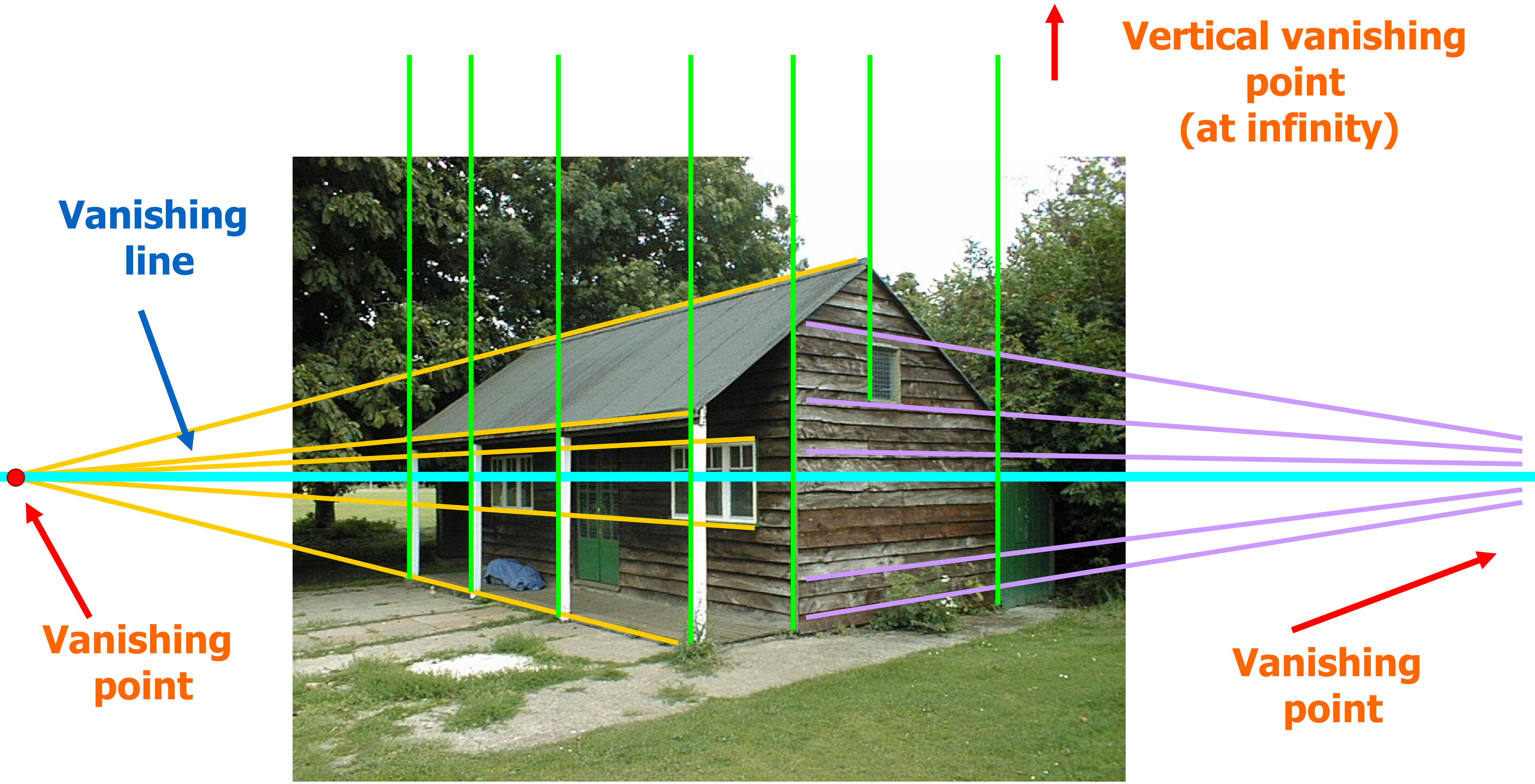


Vanishing points and lines



- The projections of parallel 3D lines intersect at a **vanishing point**
- The projection of parallel 3D planes intersect at a **vanishing line**
- If a set of parallel 3D lines are also parallel to a particular plane, their vanishing point will lie on the vanishing line of the plane
- Not all lines that intersect are parallel
- Vanishing point <-> 3D direction of a line
- Vanishing line <-> 3D orientation of a surface

Vanishing points and lines

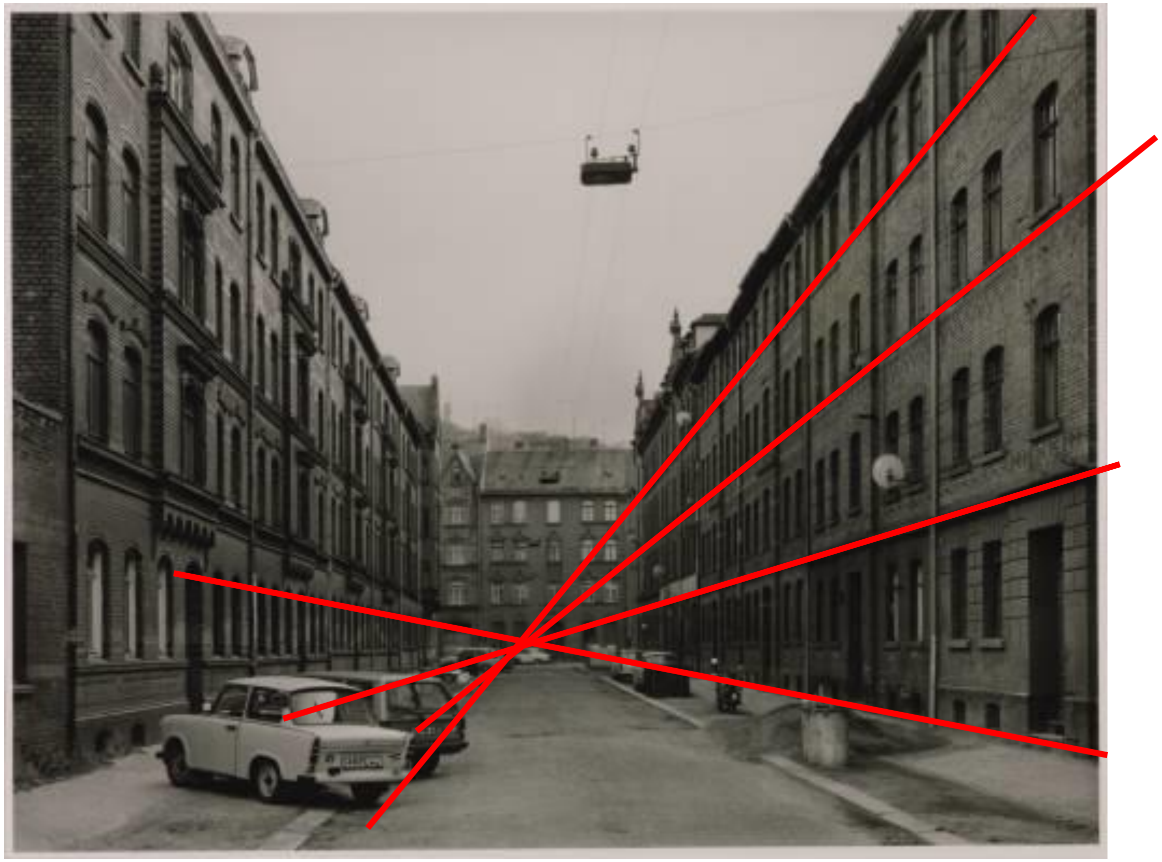


Slide from Efros, Photo from Criminisi

Vanishing points and lines

Photo from online Tate collection

Note on estimating vanishing points



Use multiple lines for better accuracy

… but lines will not intersect at exactly the same point in practice

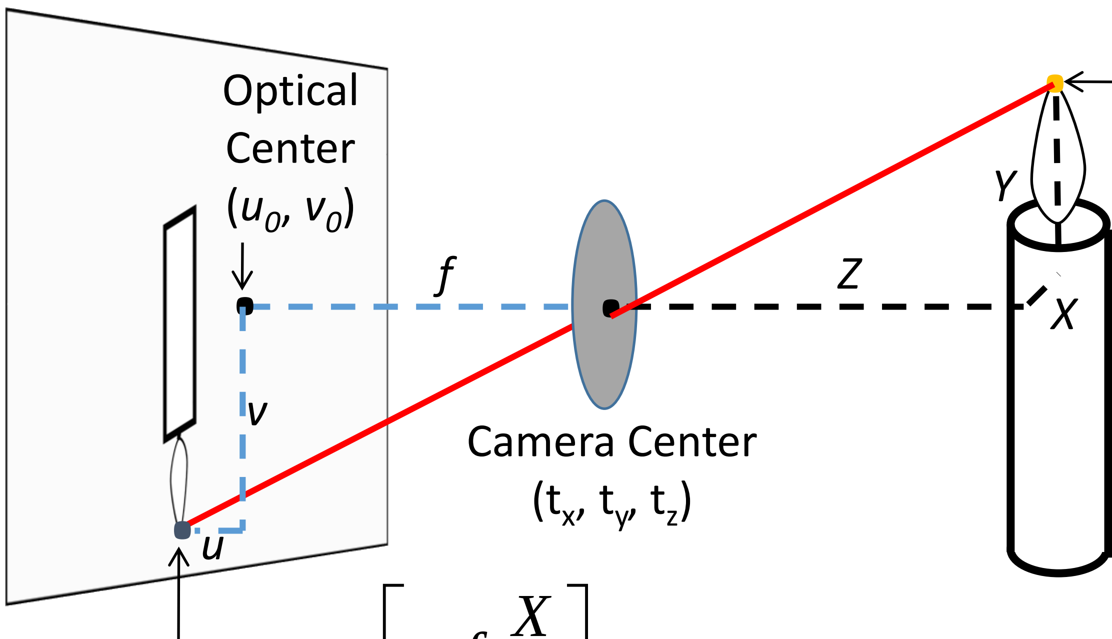
One solution: take mean of intersecting pairs

… bad idea!

Instead, minimize angular differences

Vanishing objects

Projection: world coordinates→image coordinates



$$\mathbf{P} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

$$\mathbf{p} = \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} -f\dfrac{X}{Z} \\ -f\dfrac{Y}{Z} \end{bmatrix}$$

Homogeneous coordinates

- Is this a linear transformation?
  - no—division by z is nonlinear

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

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

Homogeneous Coordinates

Cartesian Coordinates

Point in Cartesian is ray in Homogeneous

Basic geometry in homogeneous coordinates

- Append 1 to pixel coordinate to get homogeneous coordinate

$$p_i = \begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix}$$

- Line equation:

$$line_i^{\top} p = 0$$

$$au + bv + c = 0 \qquad line_i = [a\ b\ c]^{\top}$$

- 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

$$q_{ij} = line_i \times line_j$$

- Three points lies on the same line

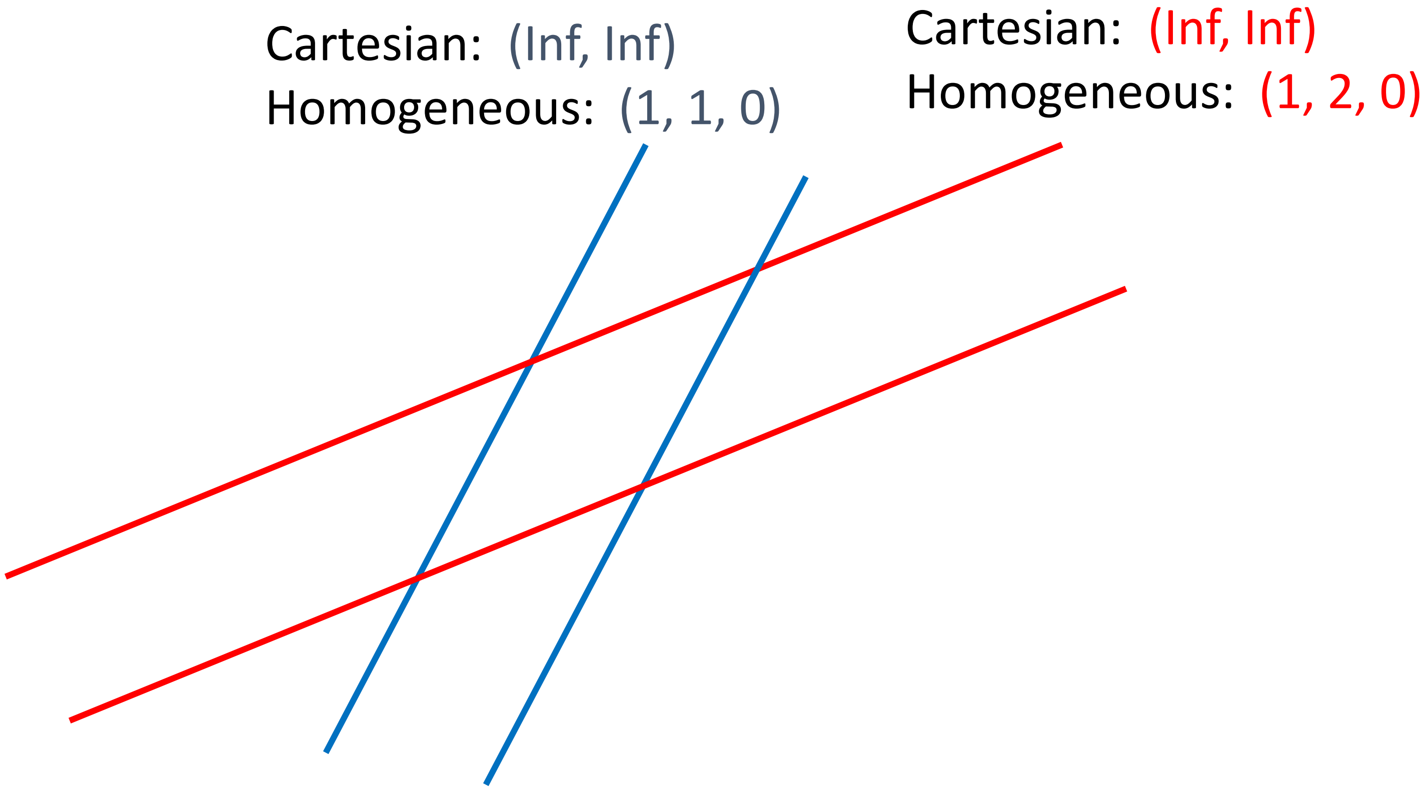
$$p_k^{\top}(p_i \times p_j) = 0$$

- Three lines intersect at the same point

$$line_k^{\top}(line_i \times line_j) = 0$$

Another problem solved by homogeneous coordinates

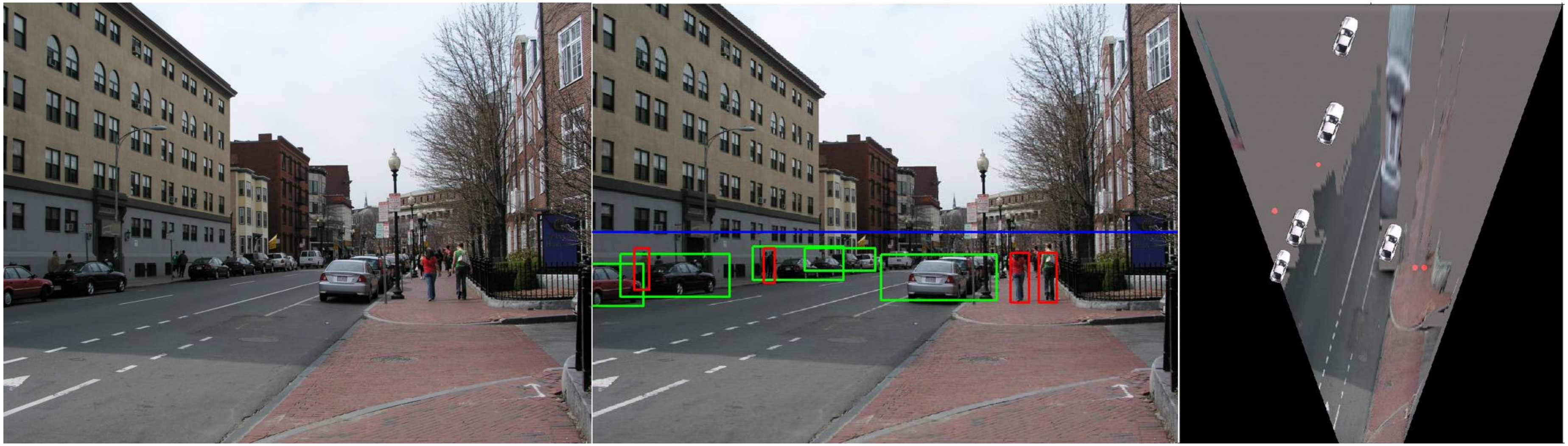
Intersection of parallel lines



Interlude: where can this be useful?

Applications

Object Recognition (CVPR 2006)

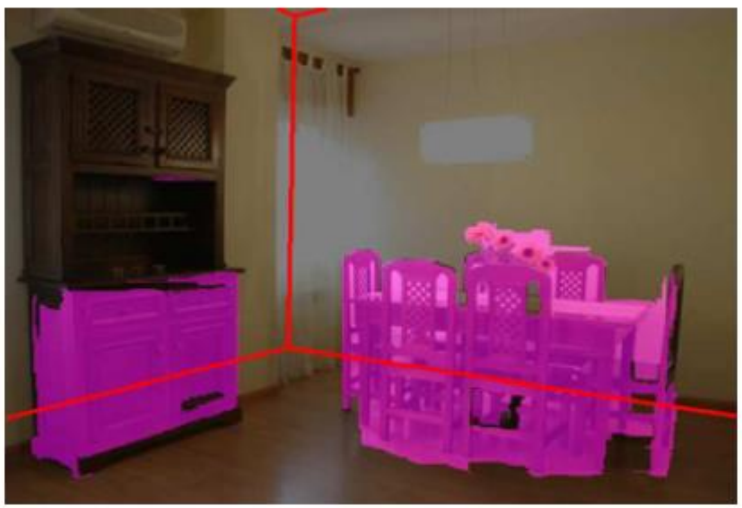


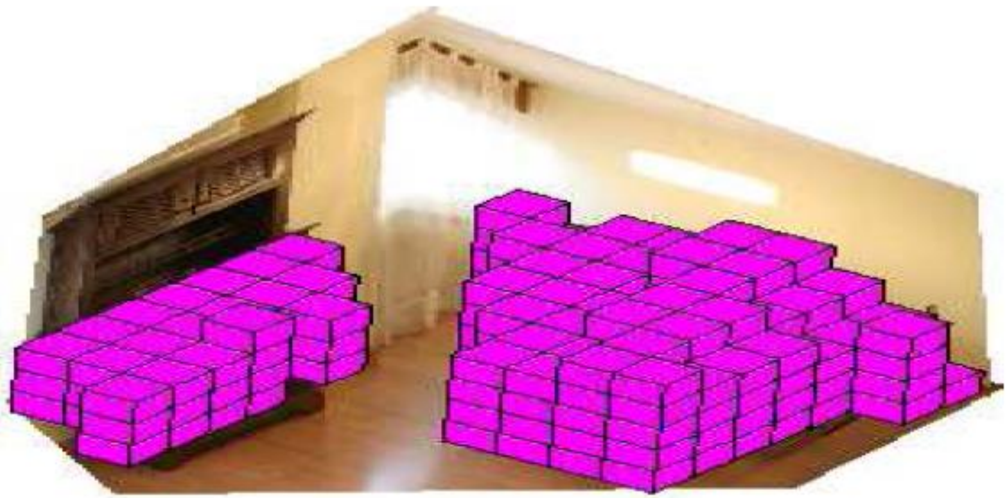
Applications

Single-view reconstruction (SIGGRAPH 2005)

Applications

Getting spatial layout in indoor scenes (ICCV 2009)



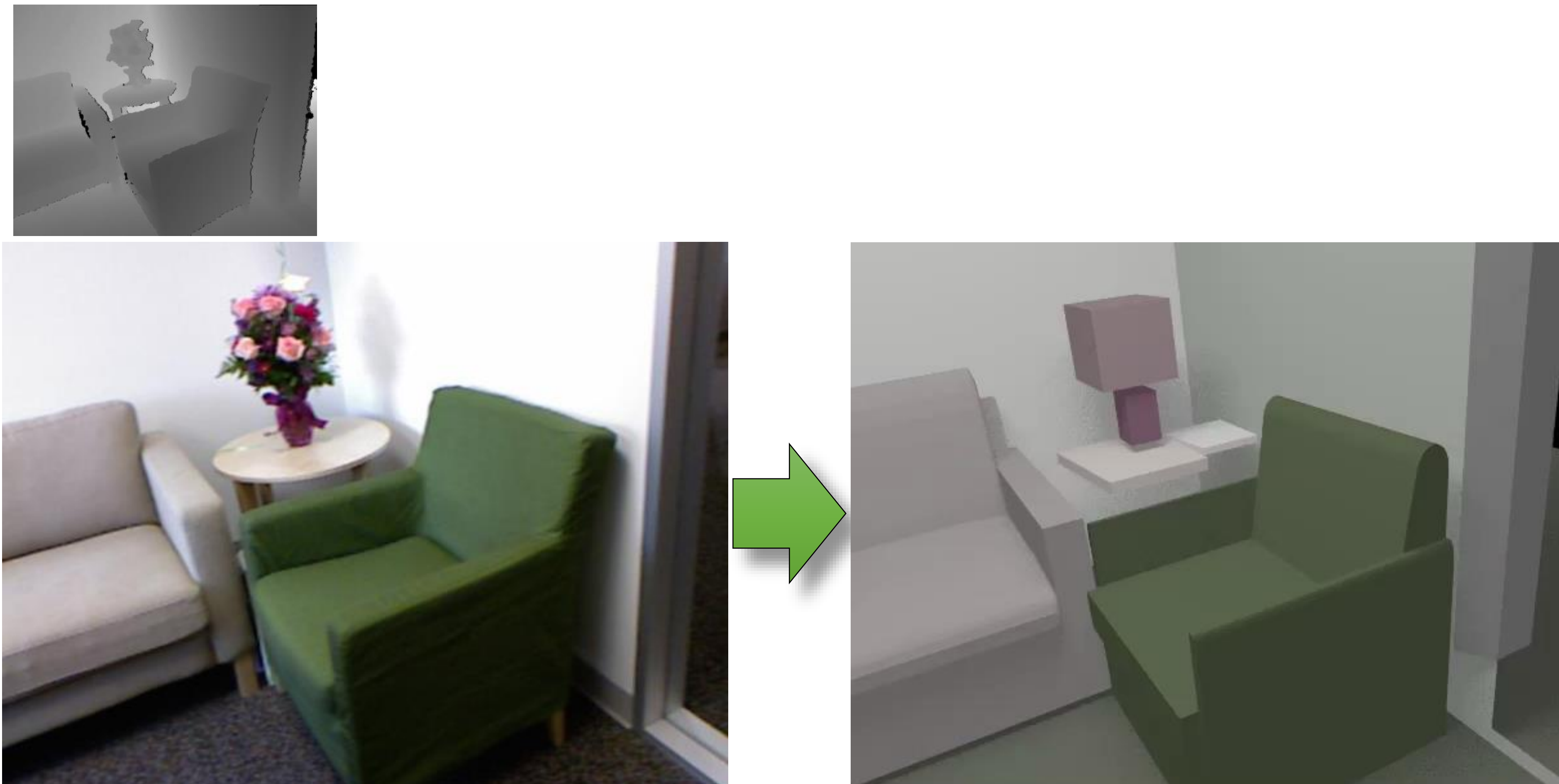


Applications

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

Applications

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



Perspective Projection Matrix

- Projection is a matrix multiplication using homogeneous coordinates

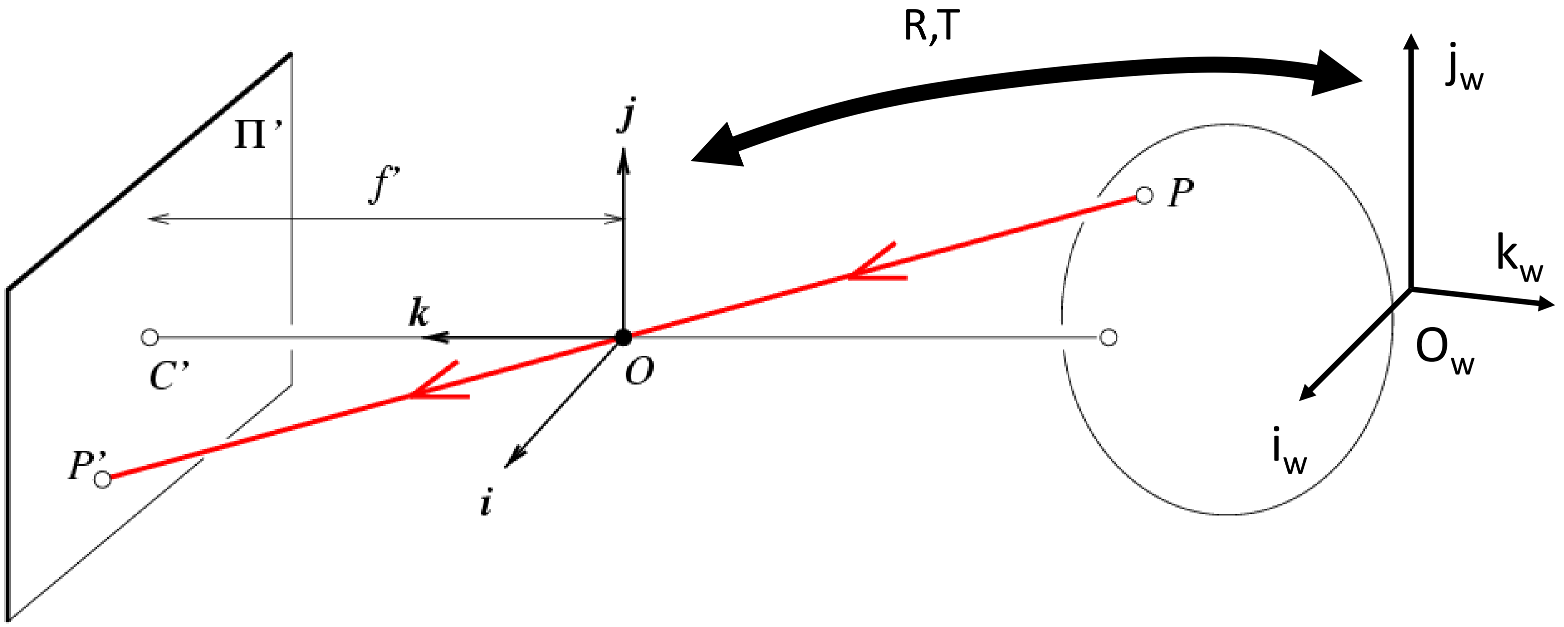
$$\begin{bmatrix} & & & \\ & & & \\ & & & \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} \\ \\ \\ \end{bmatrix} \Rightarrow \left(f\,\frac{x}{z}, f\,\frac{y}{z} \right)$$

divide by the third coordinate

In practice: lots of coordinate transformations…

$$\begin{pmatrix} \text{2D} \\ \text{point} \\ \text{(3x1)} \end{pmatrix} = \begin{pmatrix} \text{Camera to} \\ \text{pixel coord.} \\ \text{trans. matrix} \\ \text{(3x3)} \end{pmatrix} \begin{pmatrix} \text{Perspective} \\ \text{projection matrix} \\ \text{(3x4)} \end{pmatrix} \begin{pmatrix} \text{World to} \\ \text{camera coord.} \\ \text{trans. matrix} \\ \text{(4x4)} \end{pmatrix} \begin{pmatrix} \text{3D} \\ \text{point} \\ \text{(4x1)} \end{pmatrix}$$

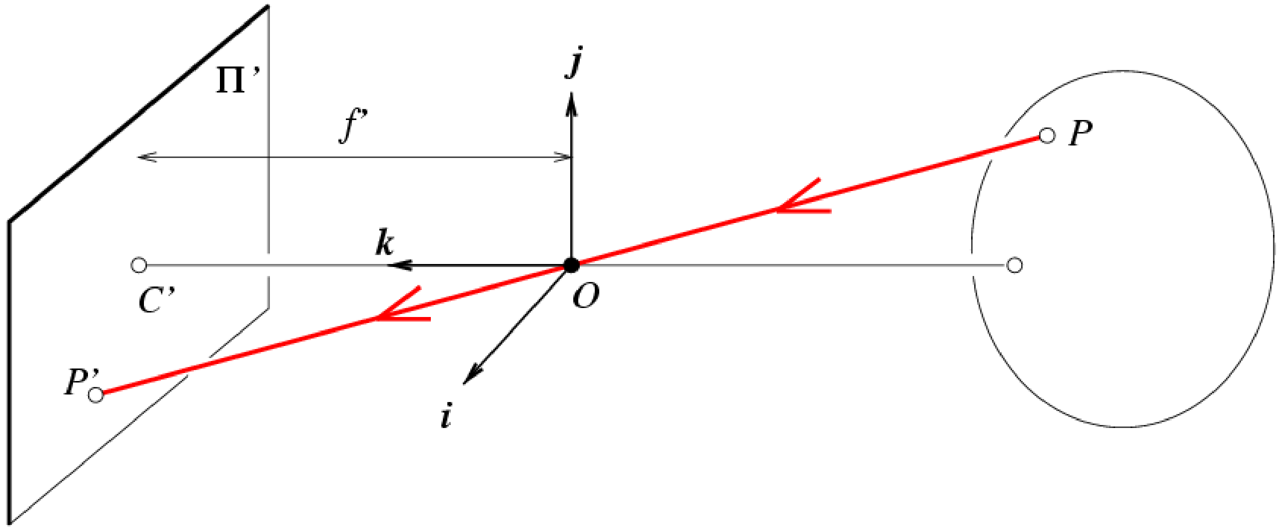
Projection matrix



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

x: Image Coordinates: (u,v,1)
K: Intrinsic Matrix (3x3)
R: Rotation (3x3)
t: Translation (3x1)
X: World Coordinates: (X,Y,Z,1)

Projection matrix



Intrinsic Assumptions

- Unit aspect ratio
- Optical center at (0,0)
- No skew

Extrinsic Assumptions

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

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

K

Slide Credit: Saverese

Remove assumption: known optical center

Intrinsic Assumptions
- Unit aspect ratio
- No skew

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

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

Remove assumption: square pixels

Intrinsic Assumptions
- No skew

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

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

Remove assumption: non-skewed pixels

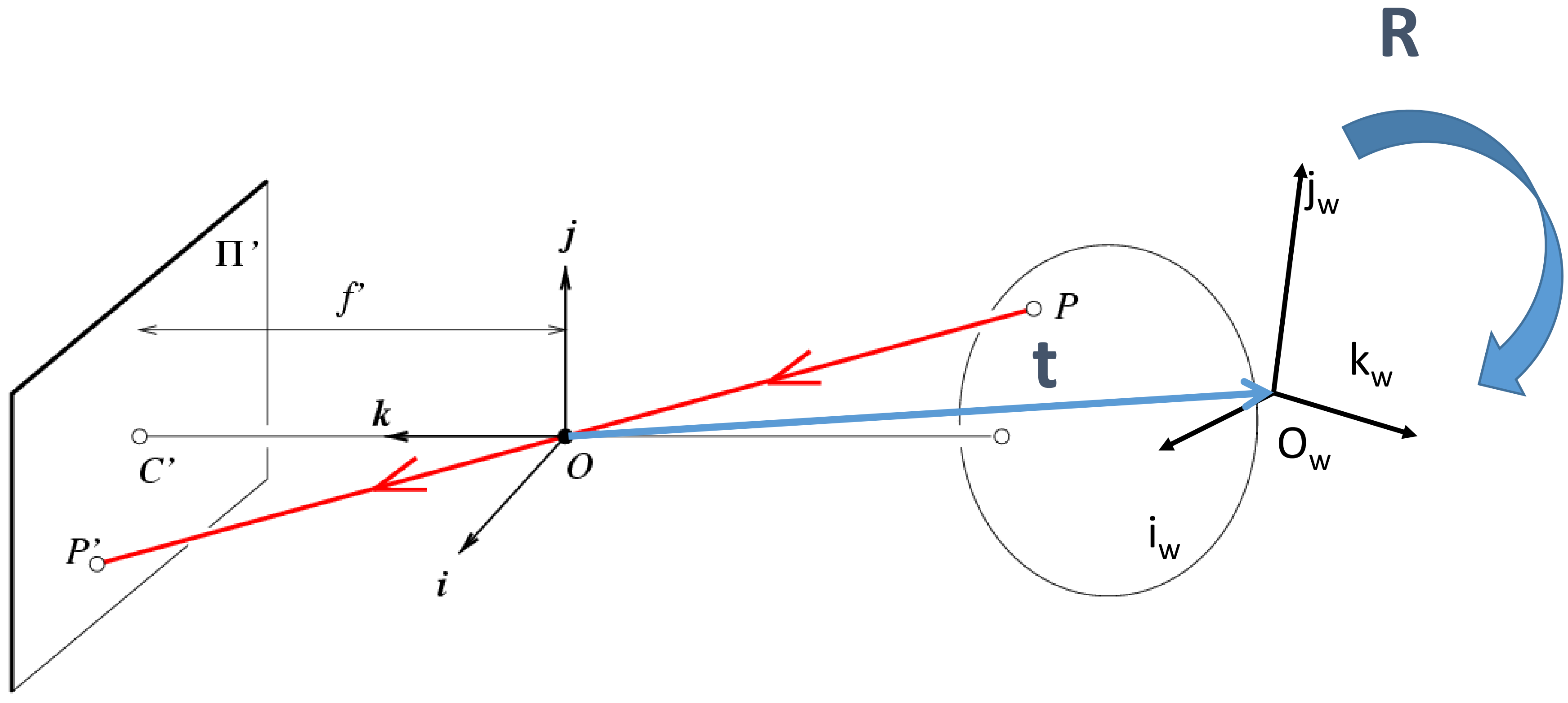
Intrinsic Assumptions

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

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

Note: different books use different notation for parameters

Oriented and Translated Camera



Allow camera translation

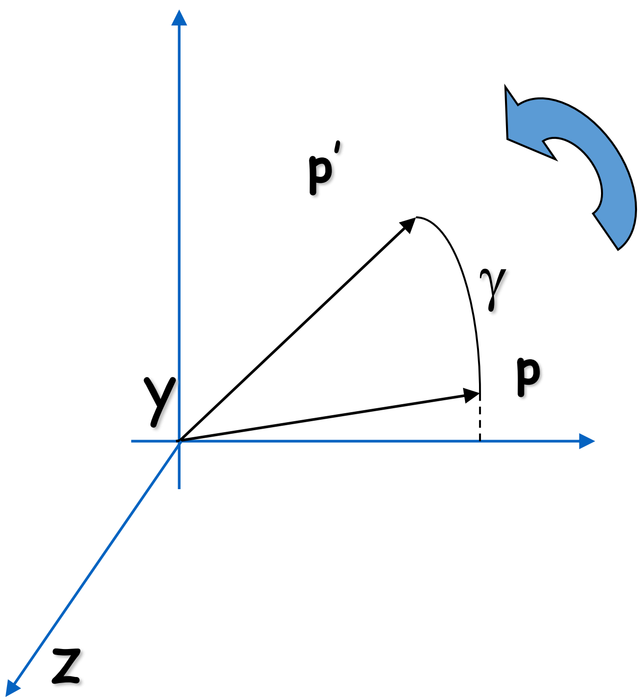
Intrinsic Assumptions

Extrinsic Assumptions
- No rotation

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

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

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

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

Degrees of freedom

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

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

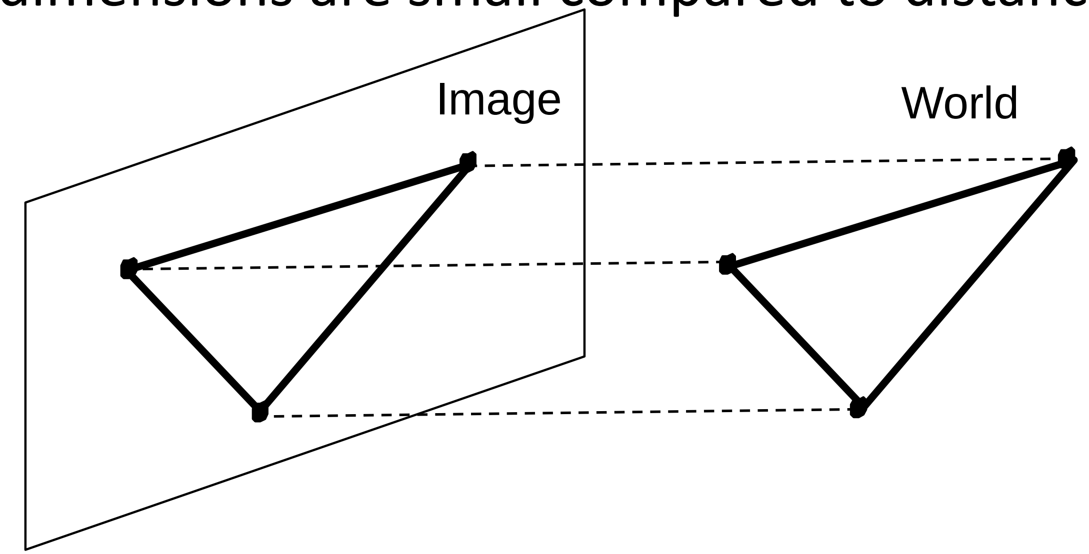
Vanishing Point = Projection from Infinity

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

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

Scaled Orthographic Projection

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

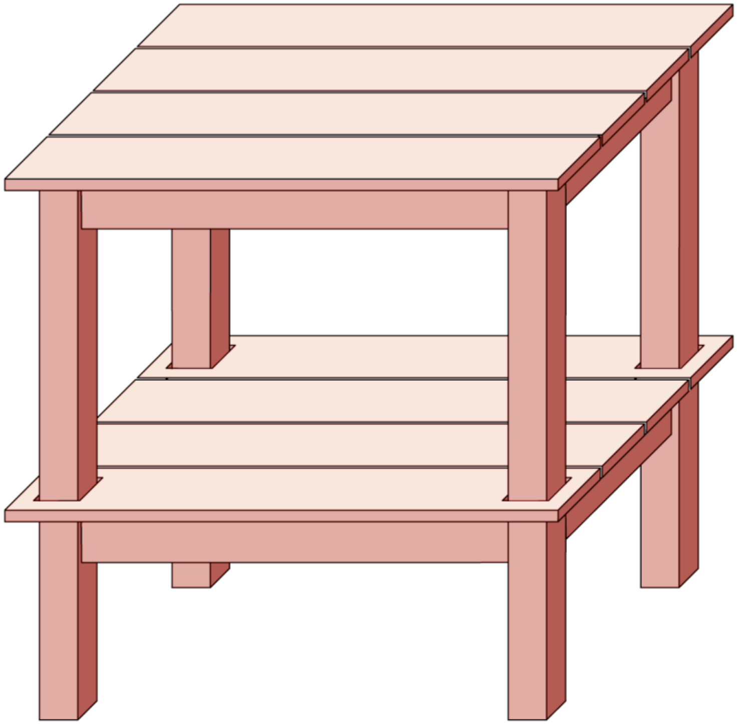


  - Also called "weak perspective"

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

Slide by Steve Seitz

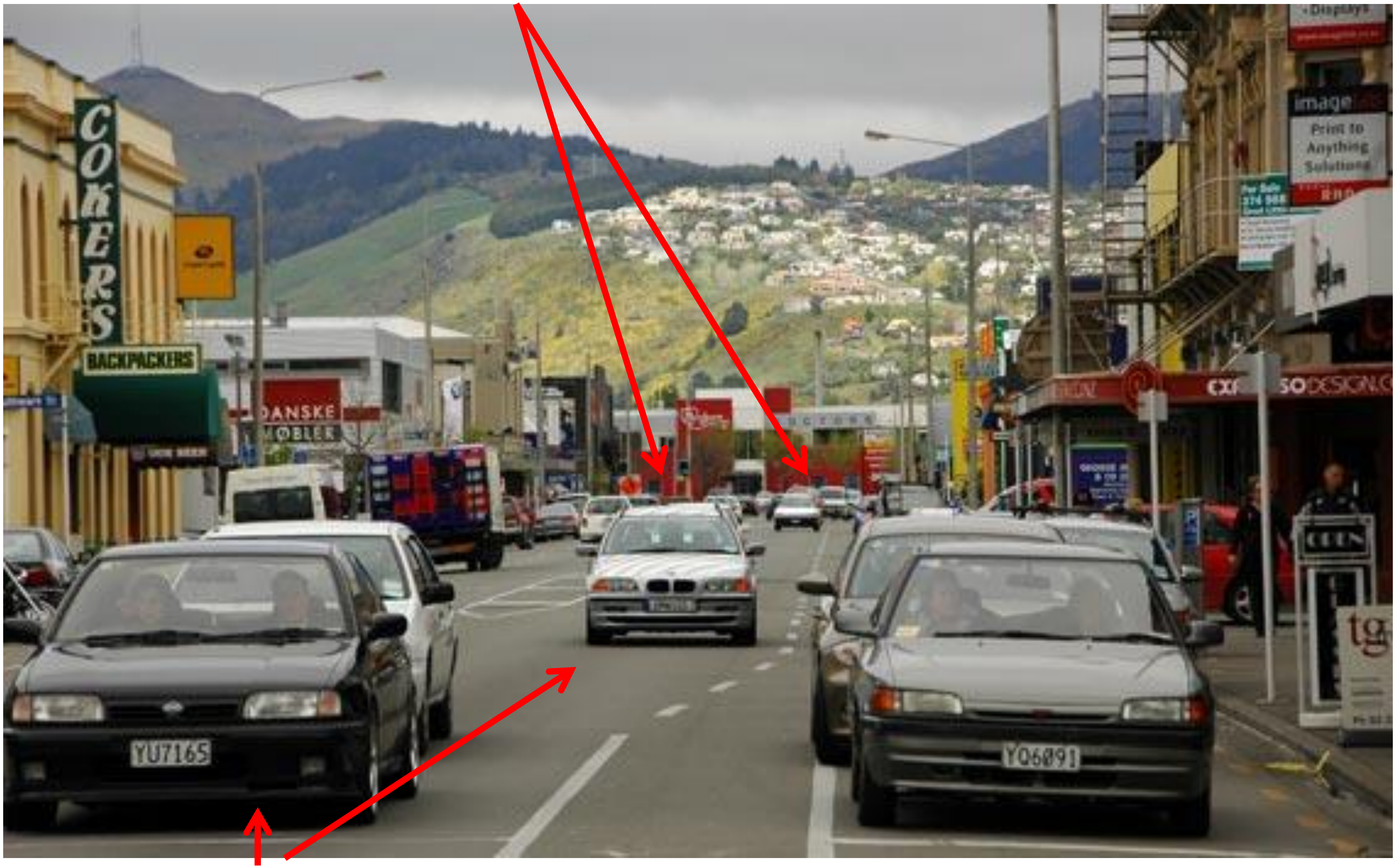
Orthographic Projection - Examples



Orthographic Projection - Examples

Applications in object detection

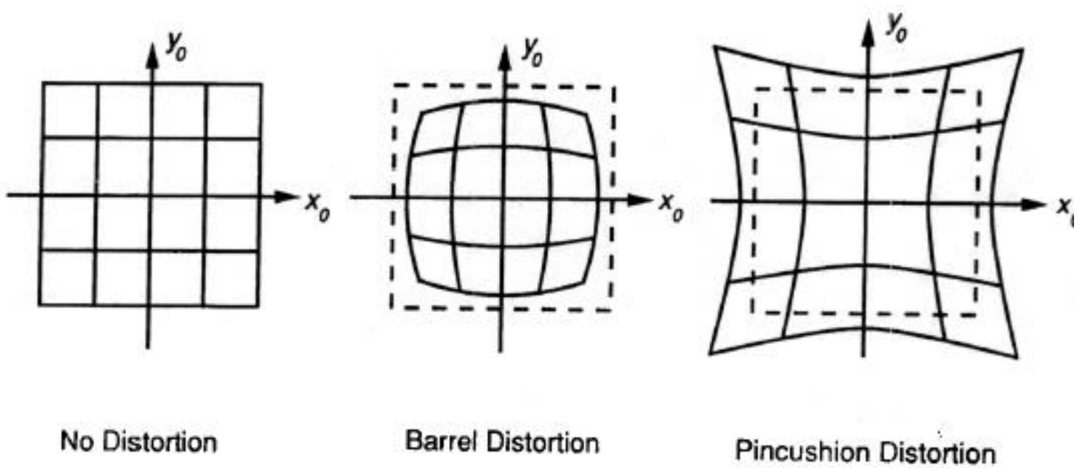
Far field: object appearance doesn't change as objects translate



Near field: object appearance changes as objects translate

Beyond Pinholes: Radial Distortion

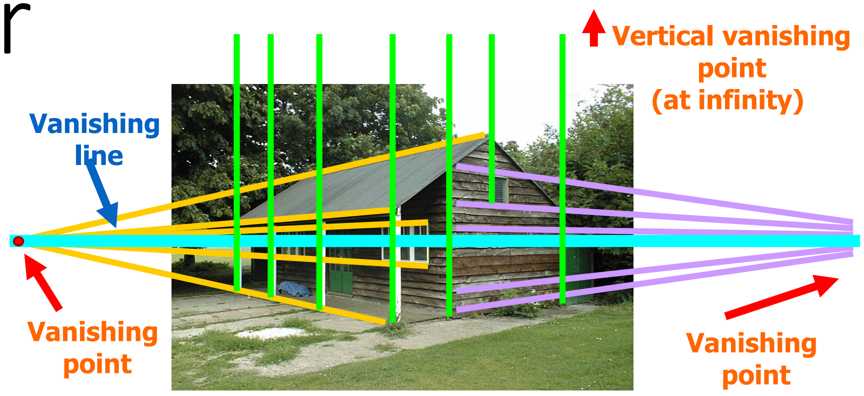
- Common in wide-angle lenses or for special applications (e.g., security)
- Creates non-linear terms in projection
- Usually handled by through solving for non-linear terms and then correcting image



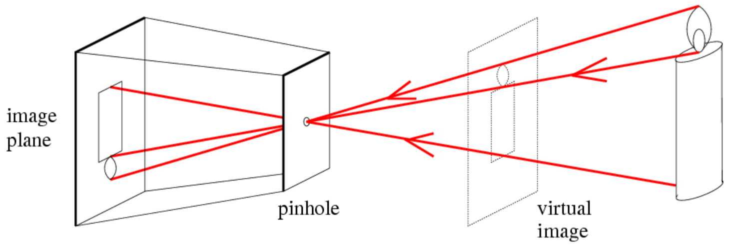
Corrected Barrel Distortion

Image from Martin Habbecke

Things to remember



- Vanishing points and vanishing lines



- Pinhole camera model and camera projection matrix

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

- Homogeneous coordinates

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

Next class
Applications of camera model and projective geometry

- Recovering the camera intrinsic and extrinsic parameters from an image

- Measuring size in the world

- Projecting from one plane to another