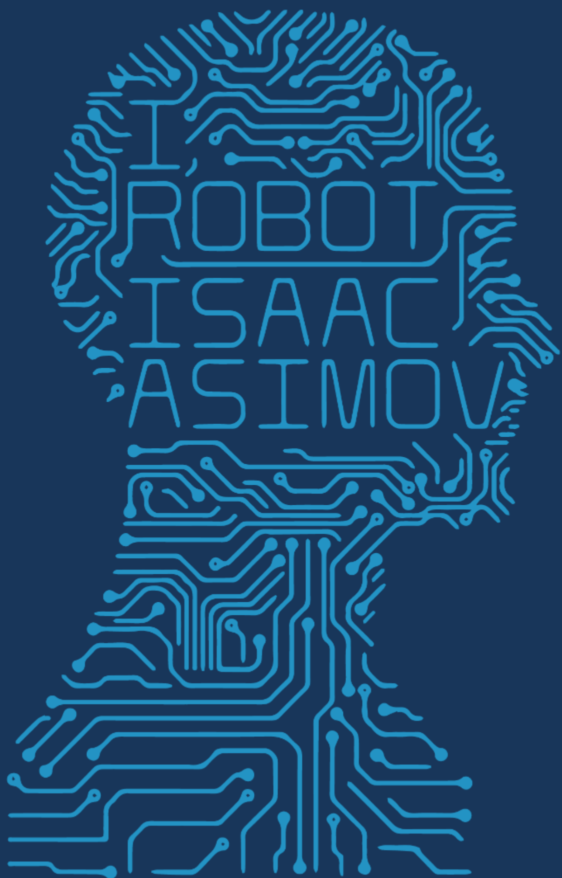
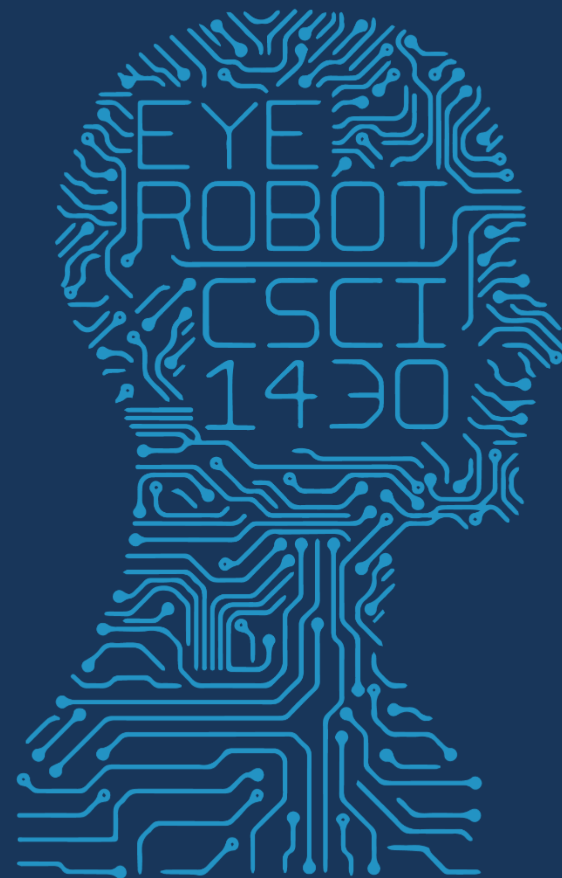



1950

Future Vision




2017 MWF 1pm 368

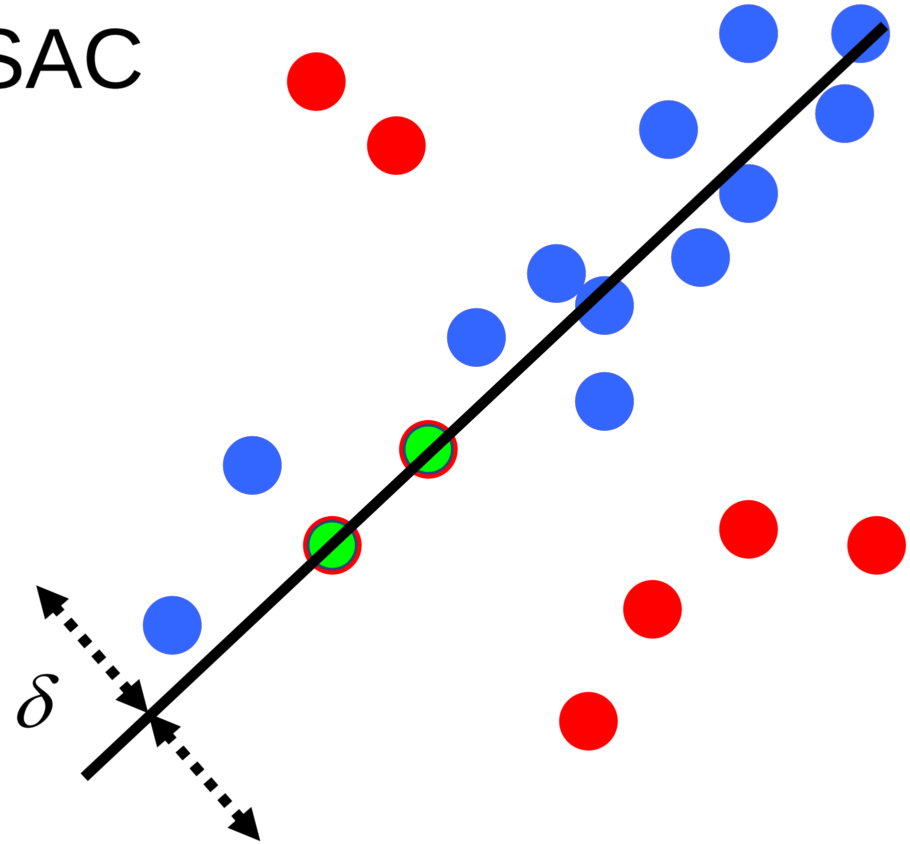
Computer Vision

# Review

- Model fitting
  - Least squares / robust least squares
  - RANSAC
  - Iterative Closest Points

- Models
  - 2D image transformations

# Review: RANSAC



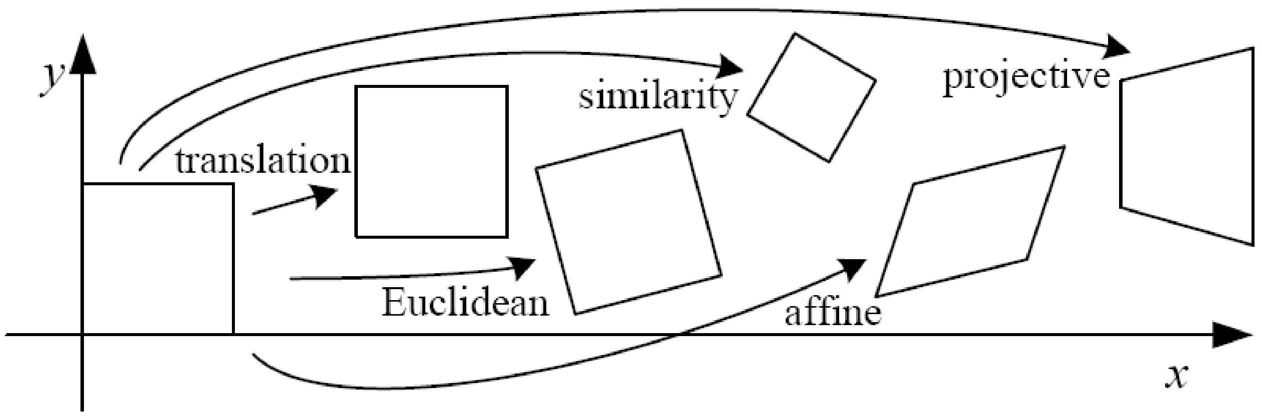

$$N_I = 14$$

Algorithm:

1. **Sample** (randomly) the number of points required to fit the model (#=2)
2. **Solve** for model parameters using samples
3. **Score** by the fraction of inliers within a preset threshold of the model

**Repeat** 1-3 until the best model is found with high confidence

# Review: 2D image transformations




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

Szeliski 2.1

# What if I want to fit multiple models?
# What if my lines are in segments?

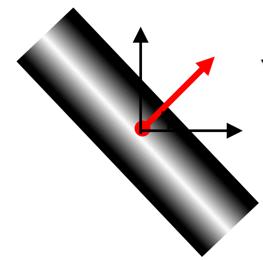

http://ostatic.com/files/images/ss_hough.jpg

# Start with edge detection → Canny

# Edge gradients

- Equation of line: y = m x + b

- Recall: when we detect an edge pixel, we can estimate its gradient m.



$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$$

- With the (x,y) position of the pixel, we can estimate b.

- Thus, each edge pixel (*edgel!*) represents a line.

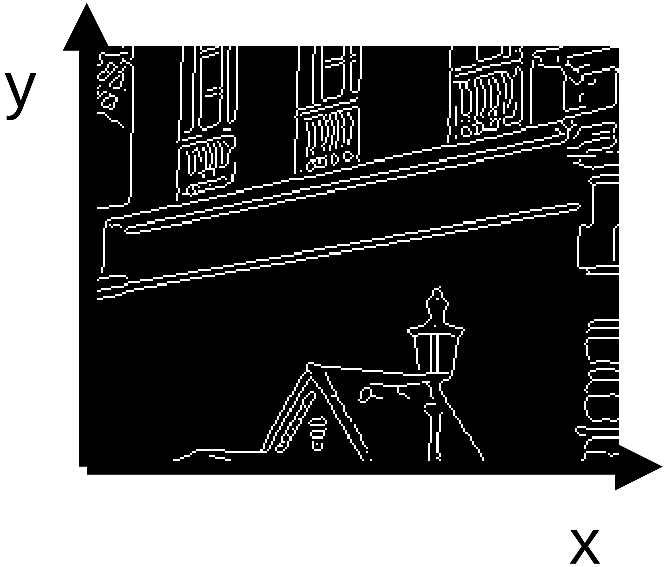
- Hough transform:
  What if each edge pixel voted for the line it might represent?

P.V.C. Hough, *Machine Analysis of Bubble Chamber Pictures,*
Proc. Int. Conf. High Energy Accelerators and Instrumentation, 1959.

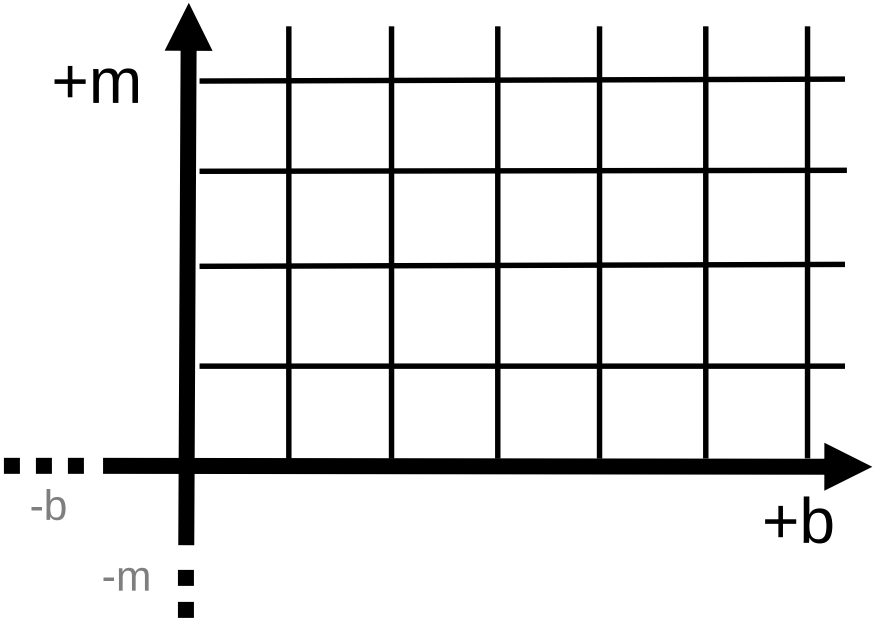
# Hough Transform: Outline

- Create a *grid* of candidate m,b parameter values.
  - Why a grid?
  - m,b are continuous; grid discretizes into hypotheses.

Edge image



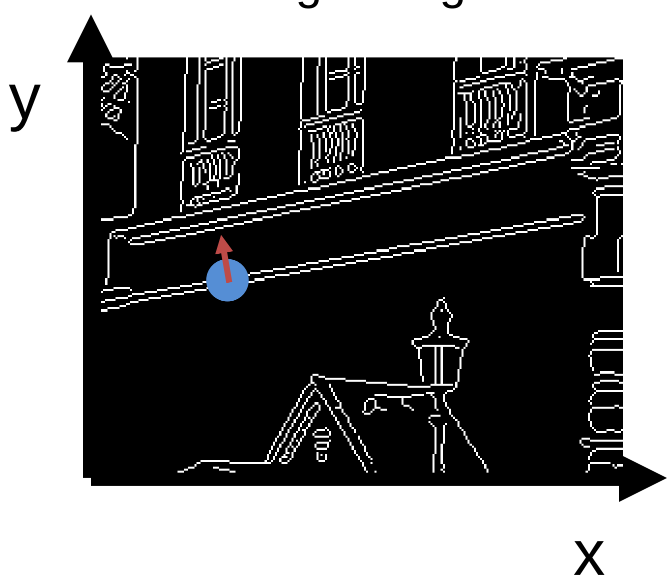

Parameter space (Hough space)



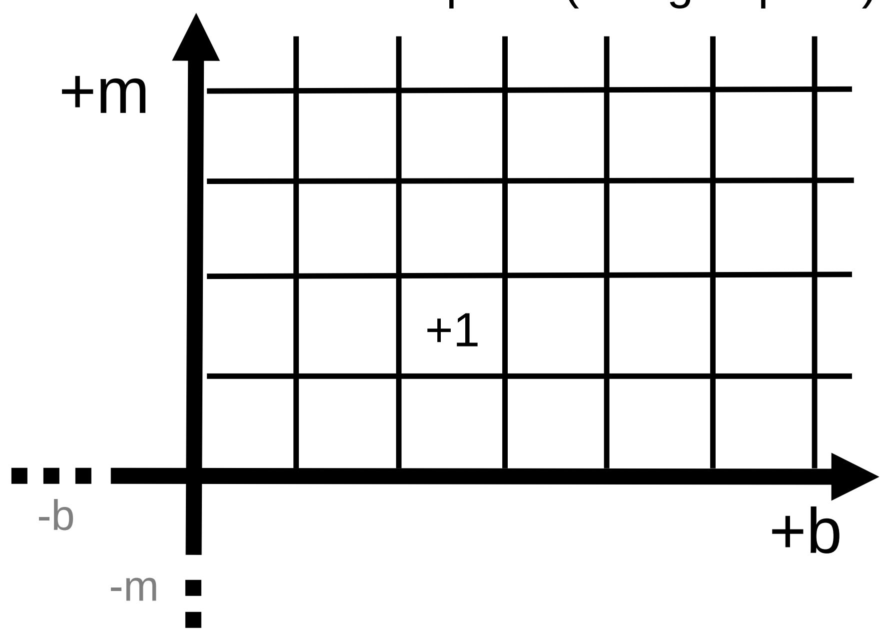
# Hough Transform: Outline

- Create a *grid* of candidate m,b parameter values.
  - Why a grid?
  - m,b are continuous; grid discretizes into hypotheses.
- Each edge pixel votes for a set of parameters, which increments those values in grid.

Edge image



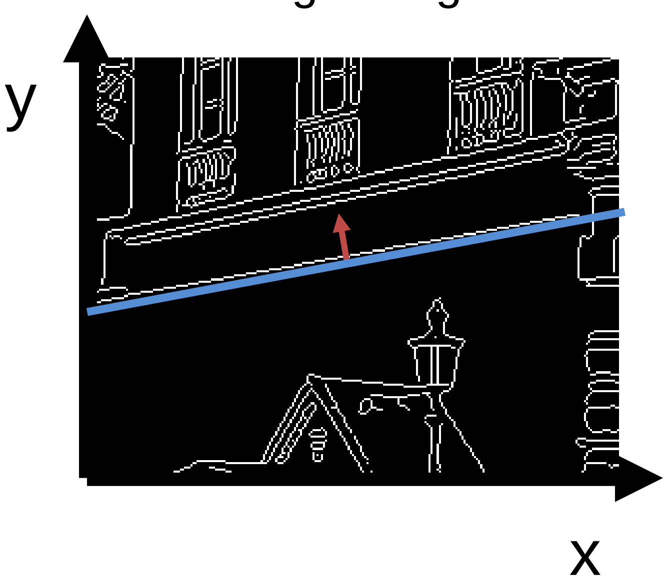

Parameter space (Hough space)



# Hough Transform: Outline

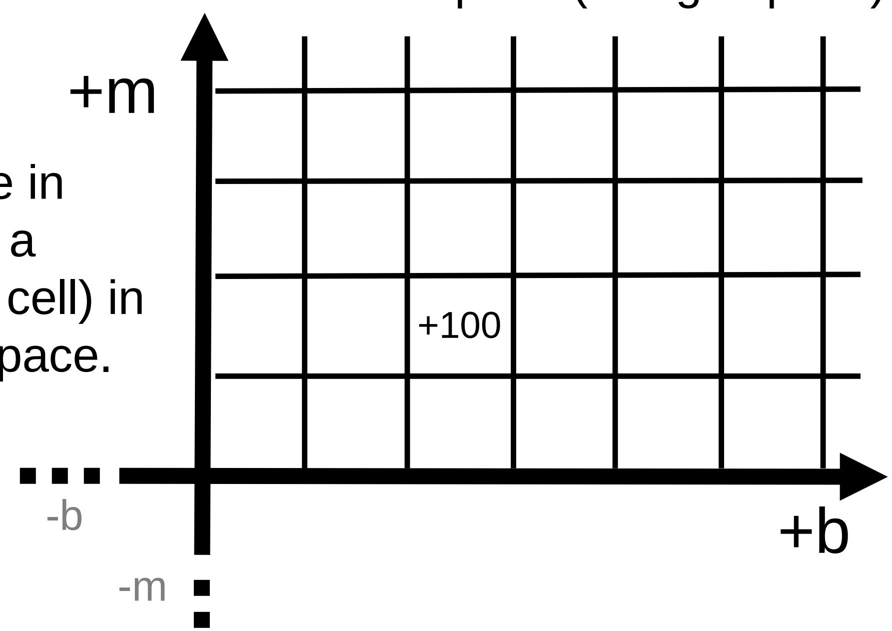
- Create a *grid* of candidate m,b parameter values.
  - Why a grid?
  - m,b are continuous; grid discretizes into hypotheses.
- Each edge pixel votes for a set of parameters, which increments those values in grid.

Edge image




Each line in image is a point (or cell) in Hough space.

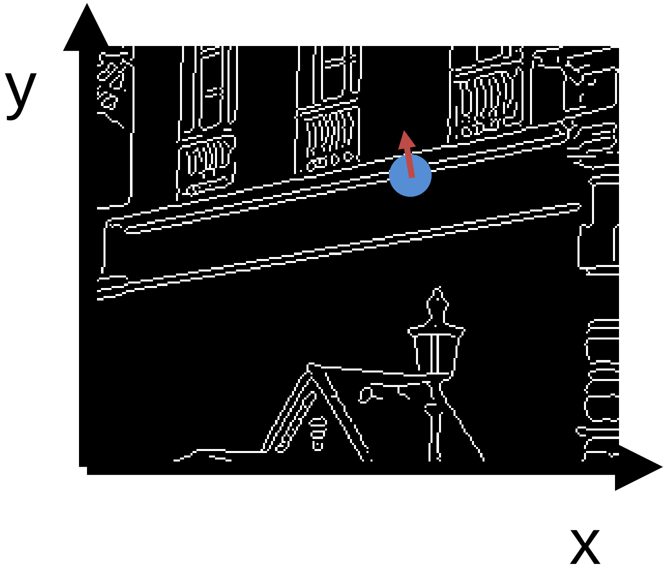
Parameter space (Hough space)



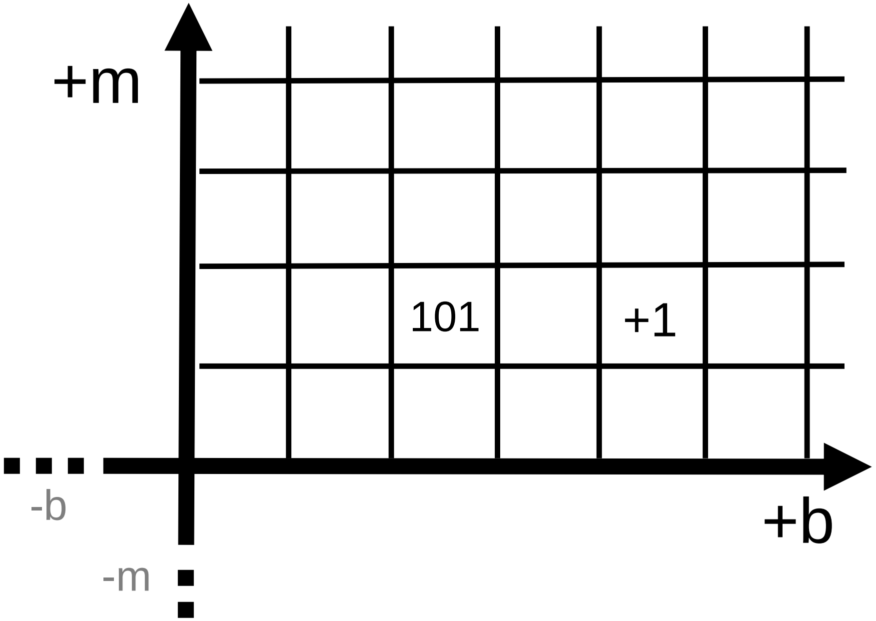
# Hough Transform: Outline

- Create a *grid* of candidate m,b parameter values.
  - Why a grid?
  - m,b are continuous; grid discretizes into hypotheses.
- Each edge pixel votes for a set of parameters, which increments those values in grid.

Edge image



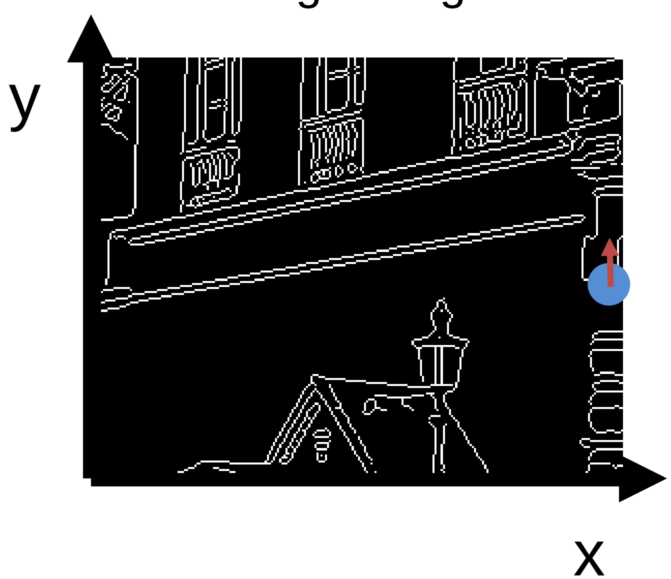

Parameter space (Hough space)



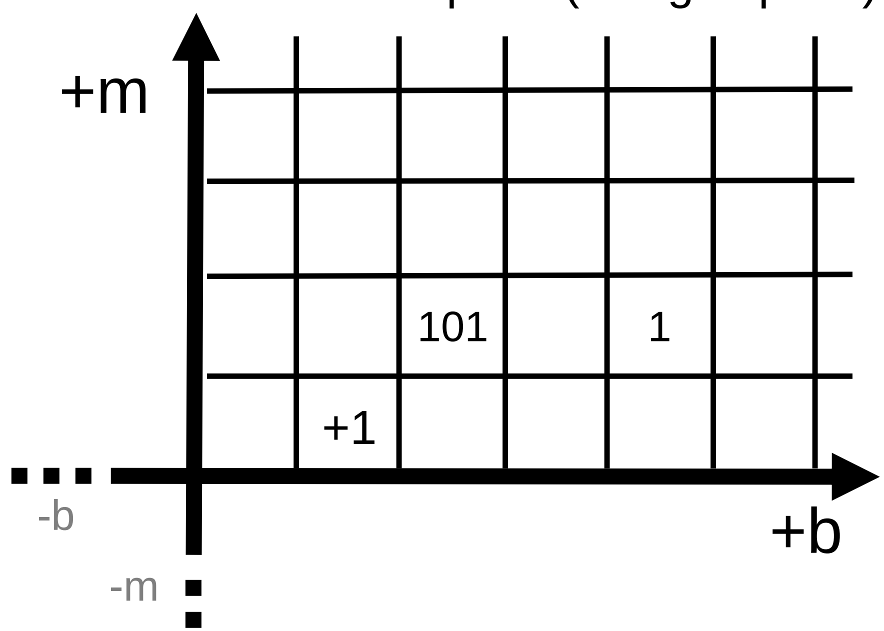
# Hough Transform: Outline

- Create a *grid* of candidate m,b parameter values.
  - Why a grid?
  - m,b are continuous; grid discretizes into hypotheses.
- Each edge pixel votes for a set of parameters, which increments those values in grid.

Edge image



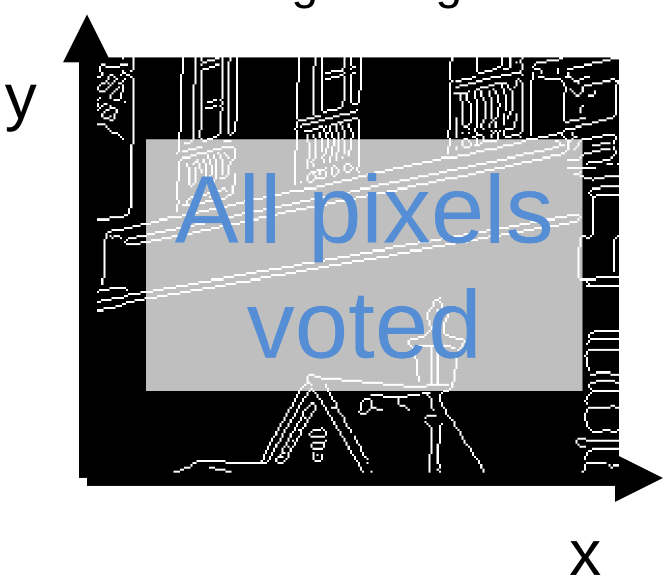

Parameter space (Hough space)



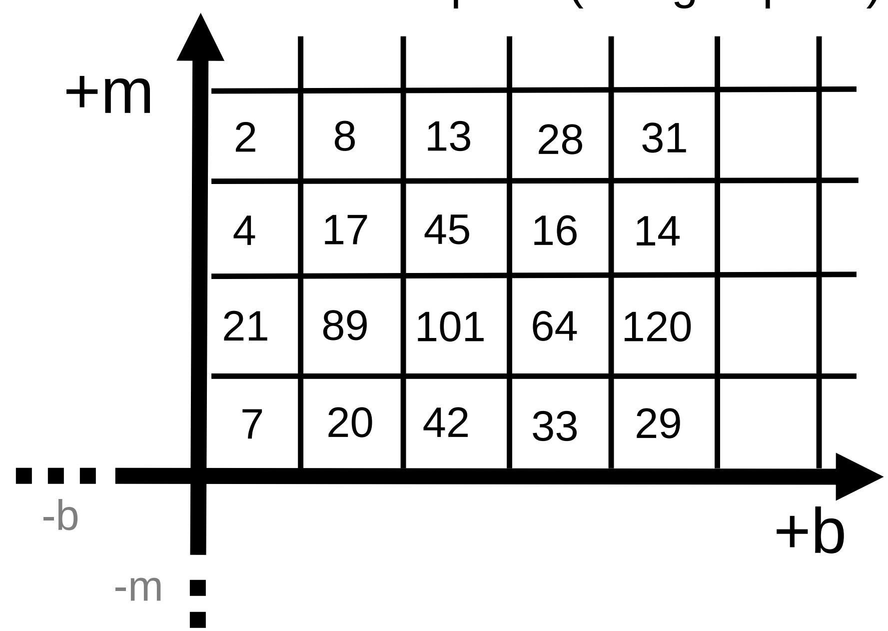
# Hough Transform: Outline

- Create a *grid* of candidate m,b parameter values.
  - Why a grid?
  - m,b are continuous; grid discretizes into hypotheses.
- Each edge pixel votes for a set of parameters, which increments those values in grid.

Edge image



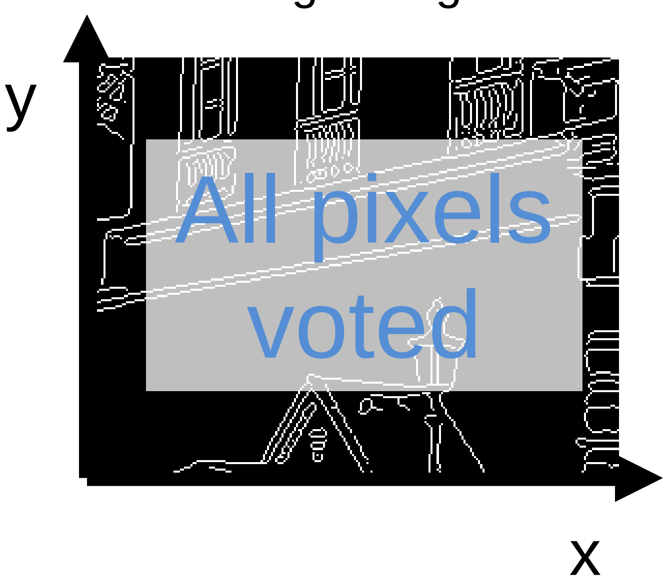

Parameter space (Hough space)



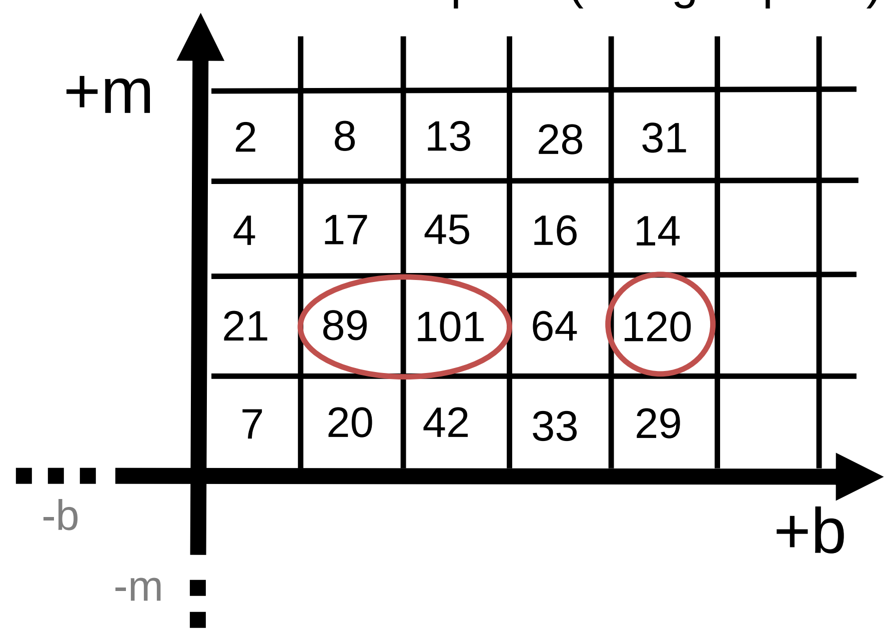
# Hough Transform: Outline

- Create a *grid* of candidate m,b parameter values.
  - Why a grid?
  - m,b are continuous; grid discretizes into hypotheses.
- Each edge pixel votes for a set of parameters, which increments those values in grid.
- Find maxima – our line candidates.

Edge image

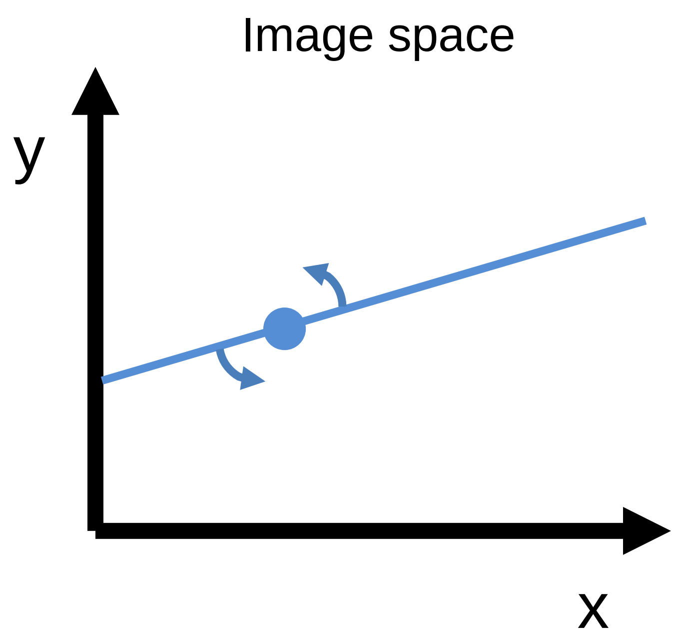


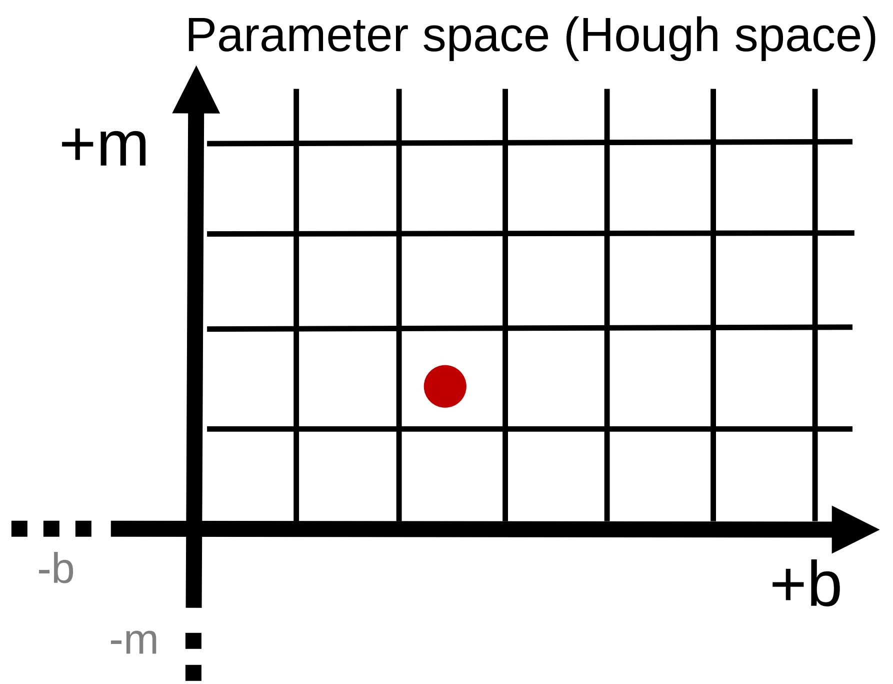

Parameter space (Hough space)



# Hough Transform: Step back

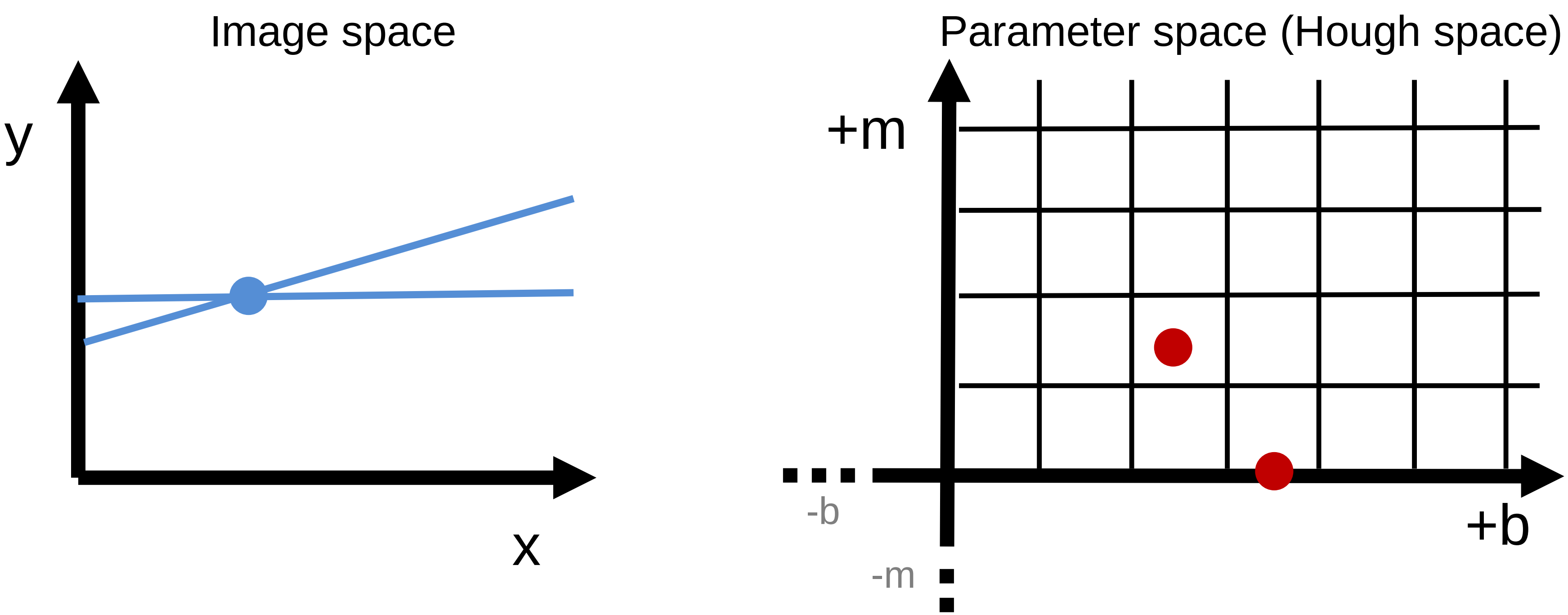
- Hough space represents *all possible lines*.
- With gradient information constriction:
    - Edgel is single point in Hough space.
- Without gradient orientation information?
    - Think-Pair-Share as orientation varies – ramifications!



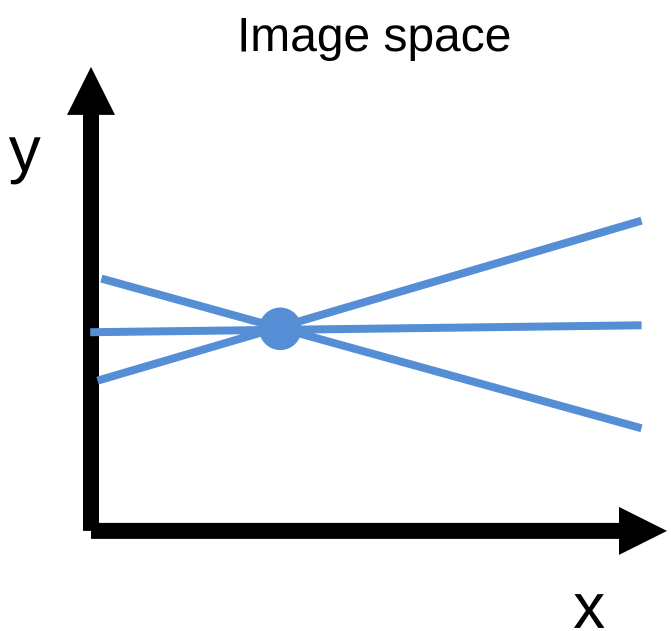
# Hough Transform: Step back

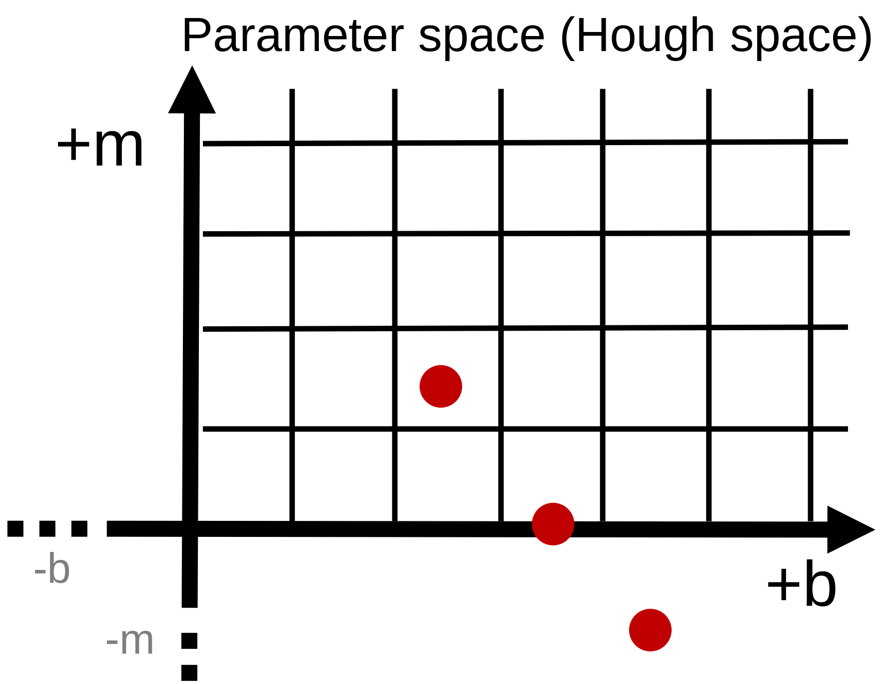
- Hough space represents *all possible lines*.
- With gradient information constriction:
  - Edgel is single point in Hough space.
- Without gradient orientation information?



# Hough Transform: Step back

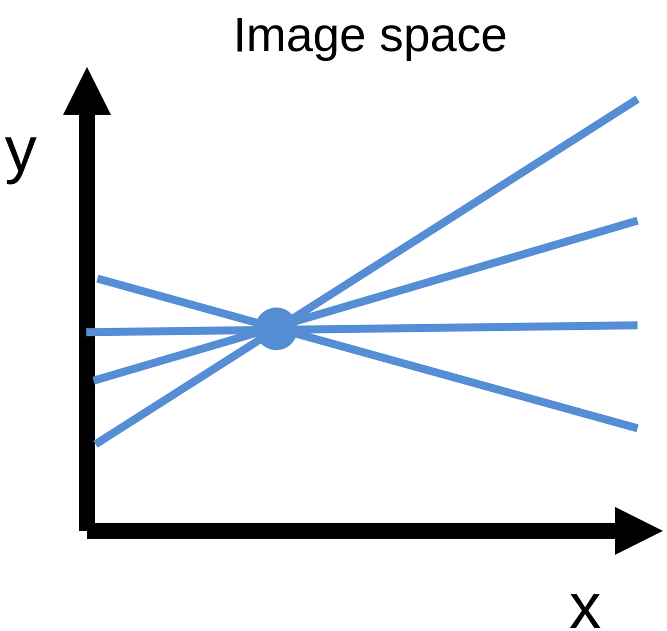
- Hough space represents *all possible lines*.
- With gradient information constriction:
  - Edgel is single point in Hough space.
- Without gradient orientation information?

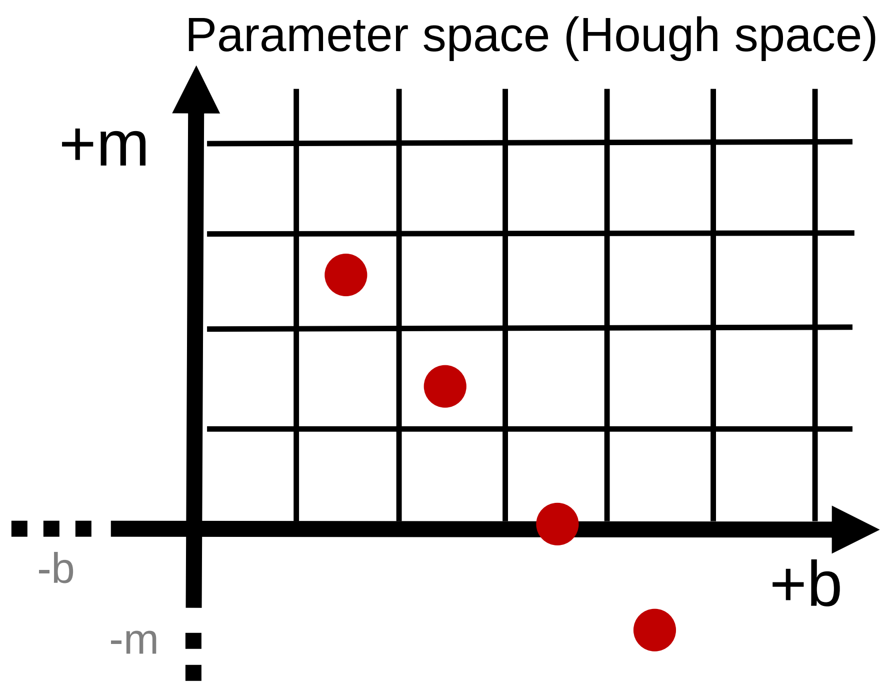


# Hough Transform: Step back

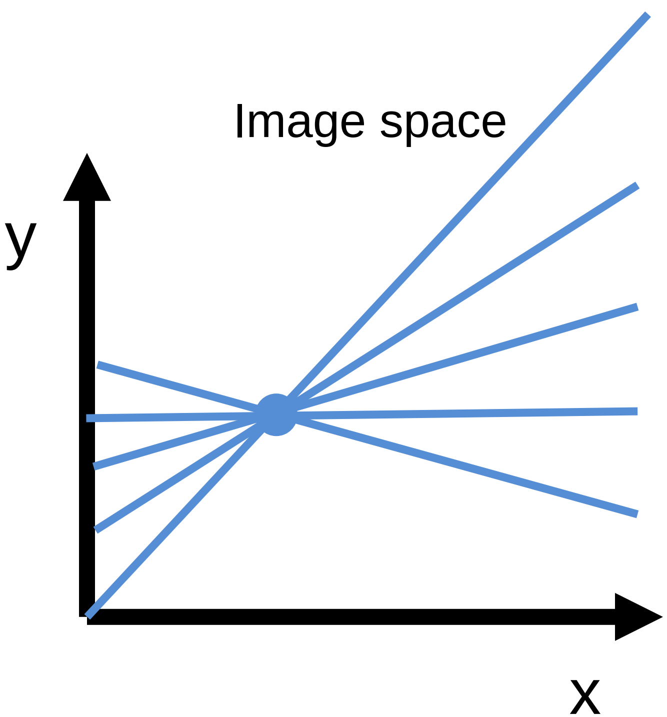
- Hough space represents *all possible lines*.
- With gradient information constriction:
  - Edgel is single point in Hough space.
- Without gradient orientation information?

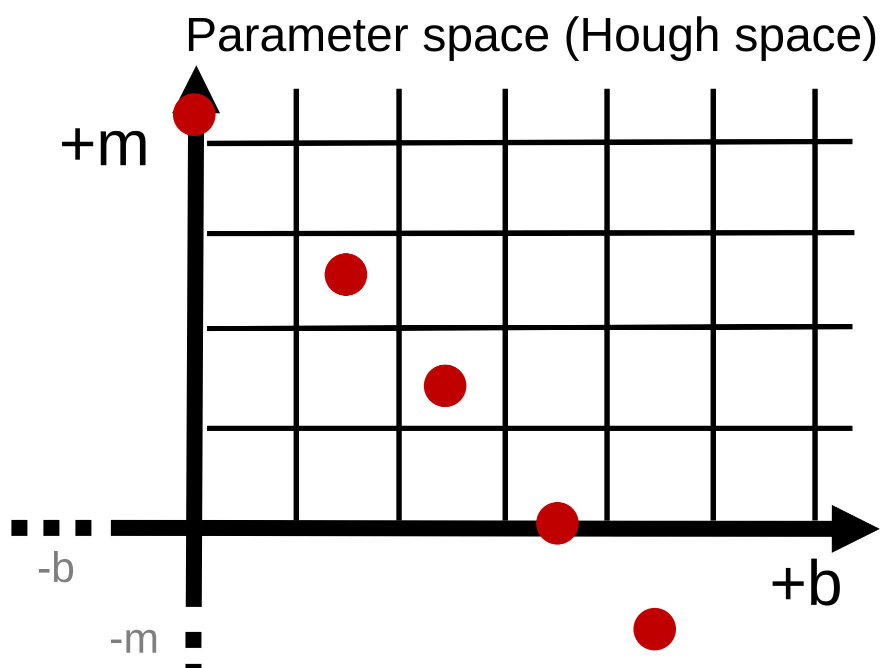


# Hough Transform: Step back

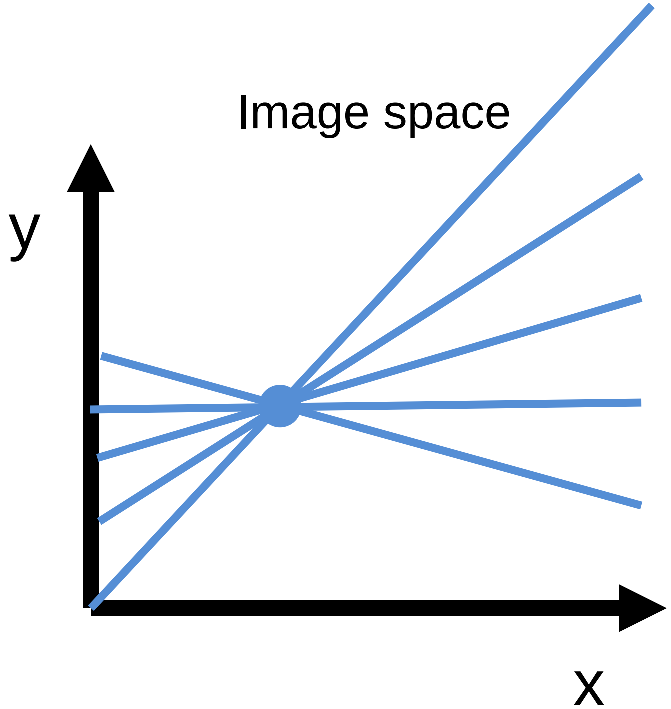
- Hough space represents *all possible lines*.
- With gradient information constriction:
  - Edgel is single point in Hough space.
- Without gradient orientation information?

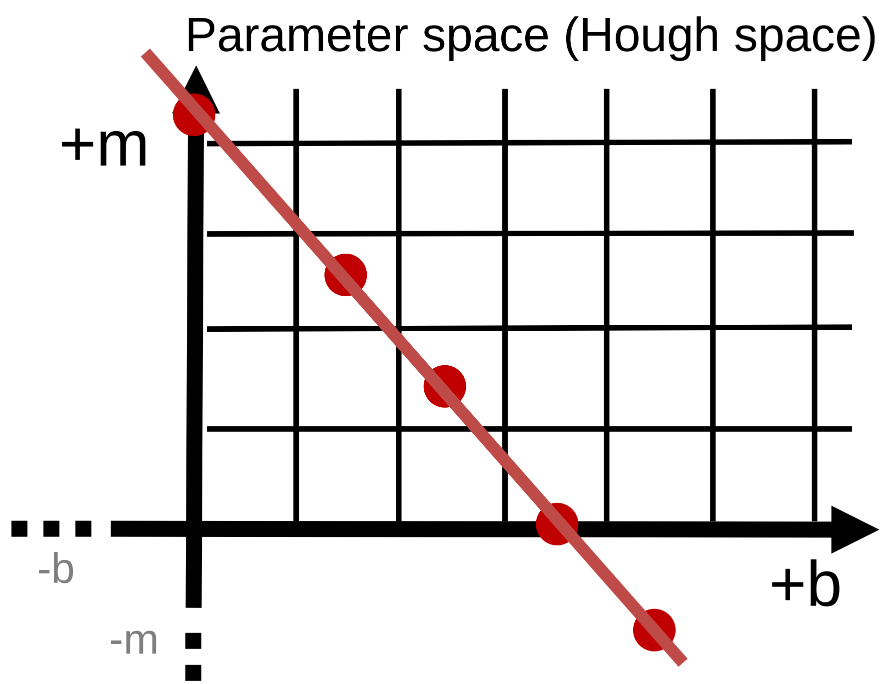


# Hough Transform: Step back

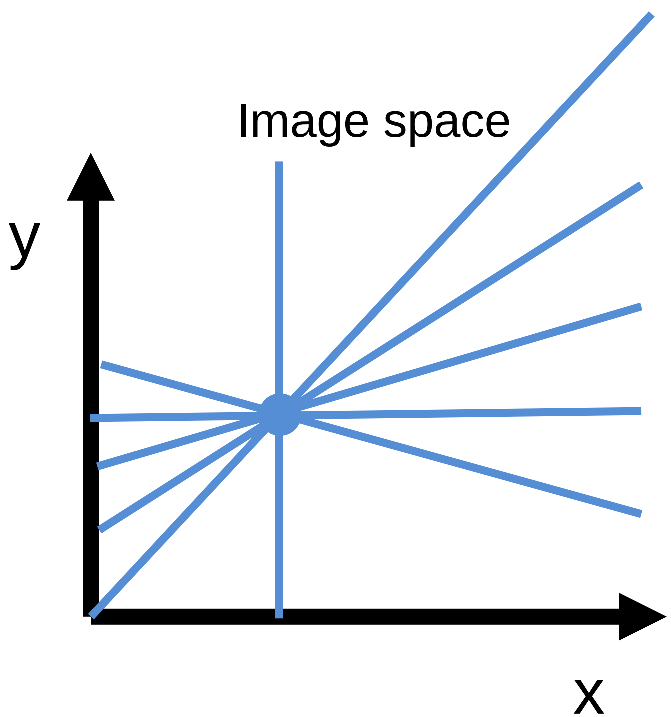
- Hough space represents *all possible lines*.
- With gradient information constriction:
  - Edgel is single point in Hough space.
- Without gradient orientation information?
  - Unoriented point is line is Hough space.



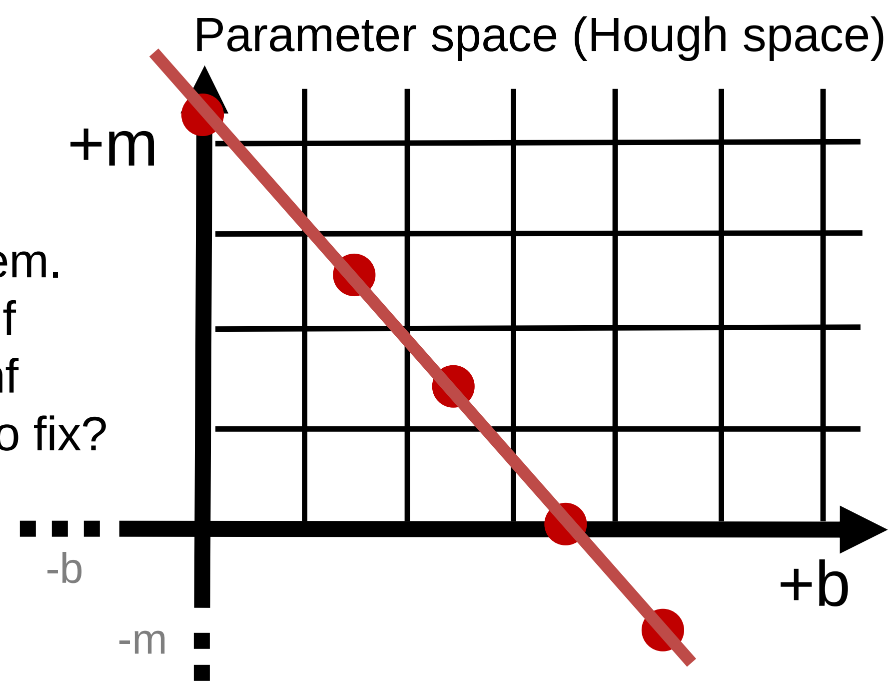



# Hough Transform: Step back

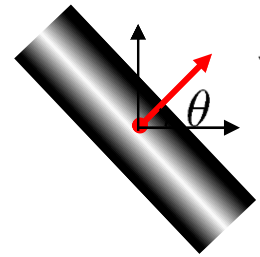
- Hough space represents *all possible lines*.
- With gradient information constriction:
  - Edgel is single point in Hough space.
- Without gradient orientation information?
  - How big is Hough space?




Problem.
m = inf
b = -inf
How to fix?



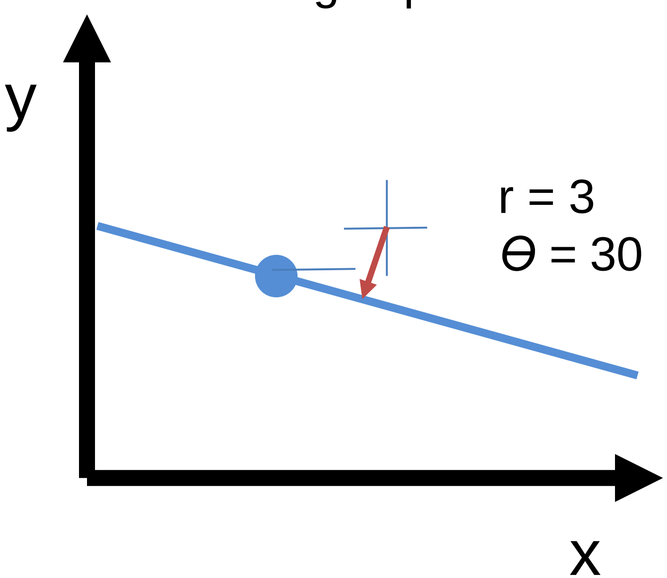
# Hough Transform: Line Normal Form



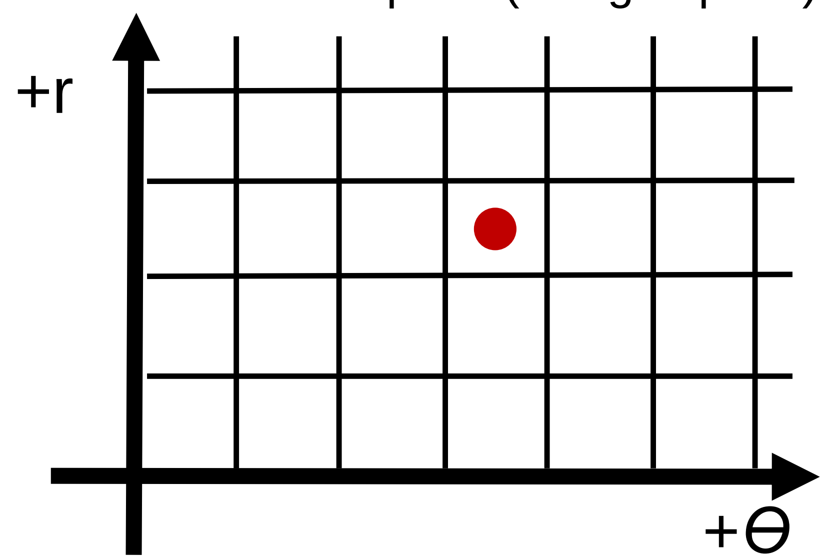

$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$$

- Use $\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right)$
  - Space is 0 to 360
- Use *r* = distance to line from some origin
  - $r_i = x_i \cos\theta_i + y_i \sin\theta_i$
  - Space is $\pm\sqrt{\max\_x^2 + \max\_y^2}$

Image space

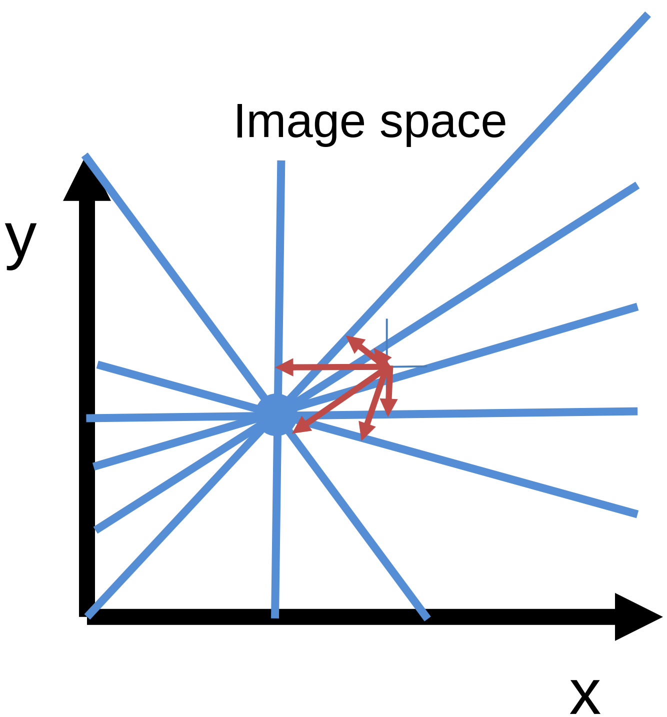


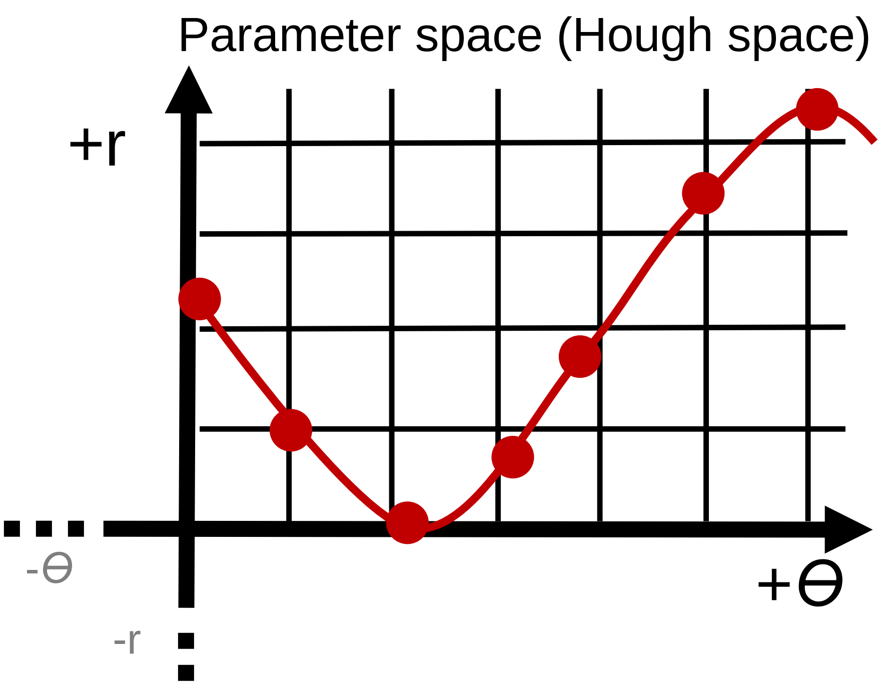

Parameter space (Hough space)



# Hough Transform: Line Normal Form

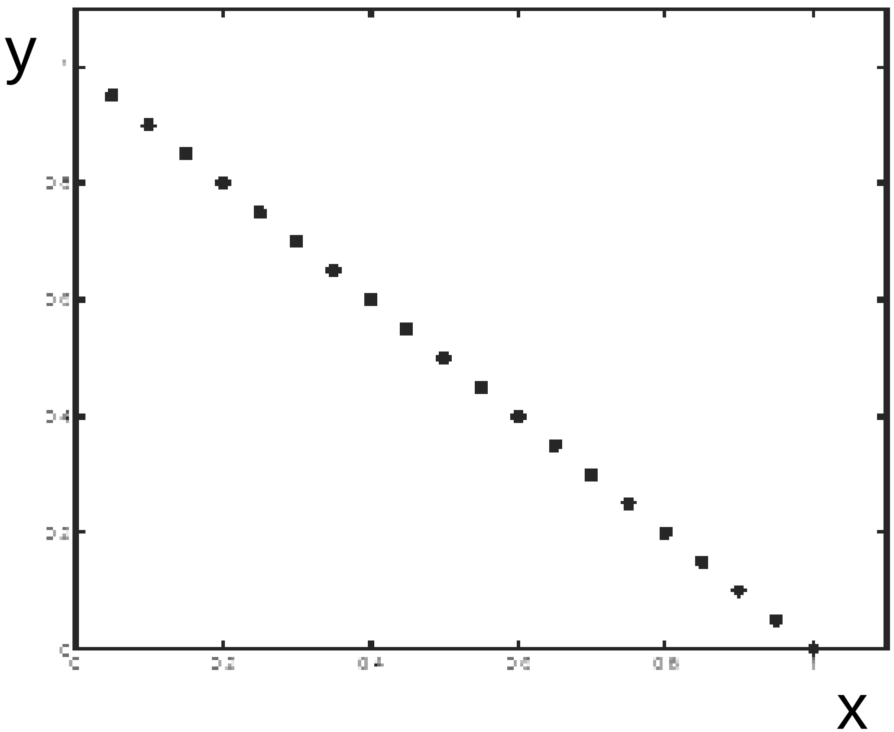
- In this line form, unoriented edge draws a sinusoid in Hough space.



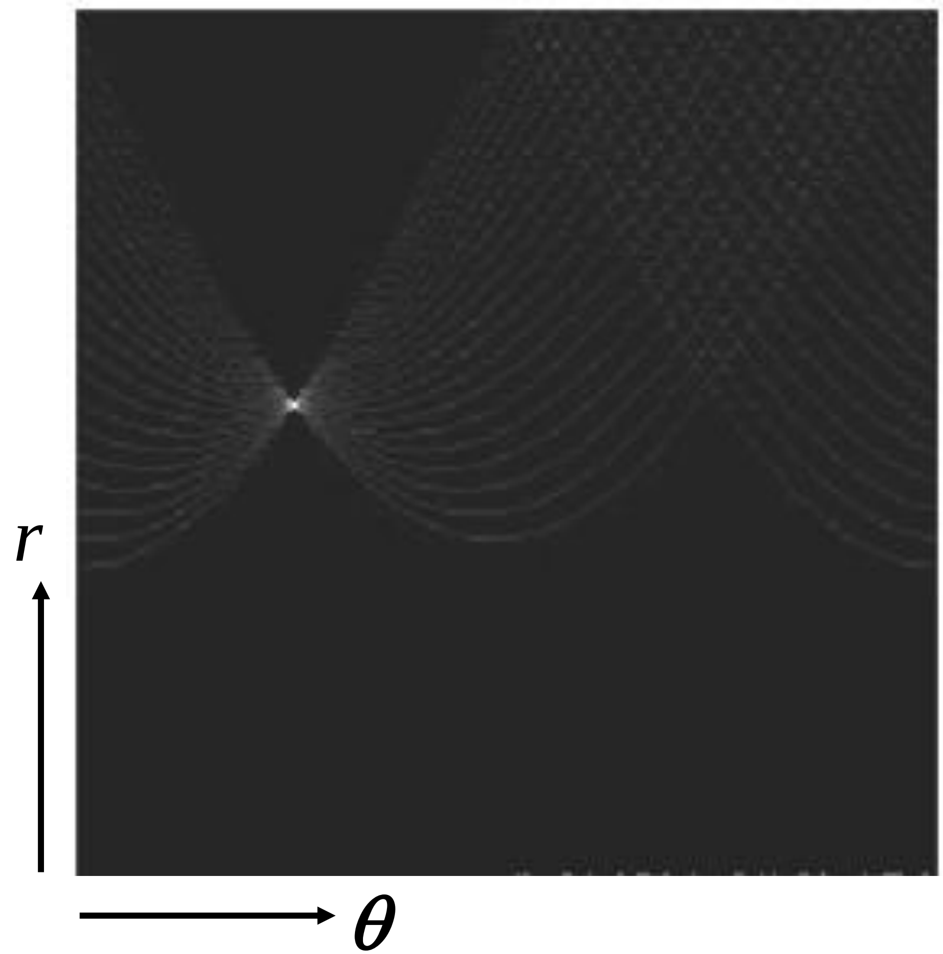



# Hough transform - experiments

Next few images *ignore* edge orientation.
Each point is one sinusoid.



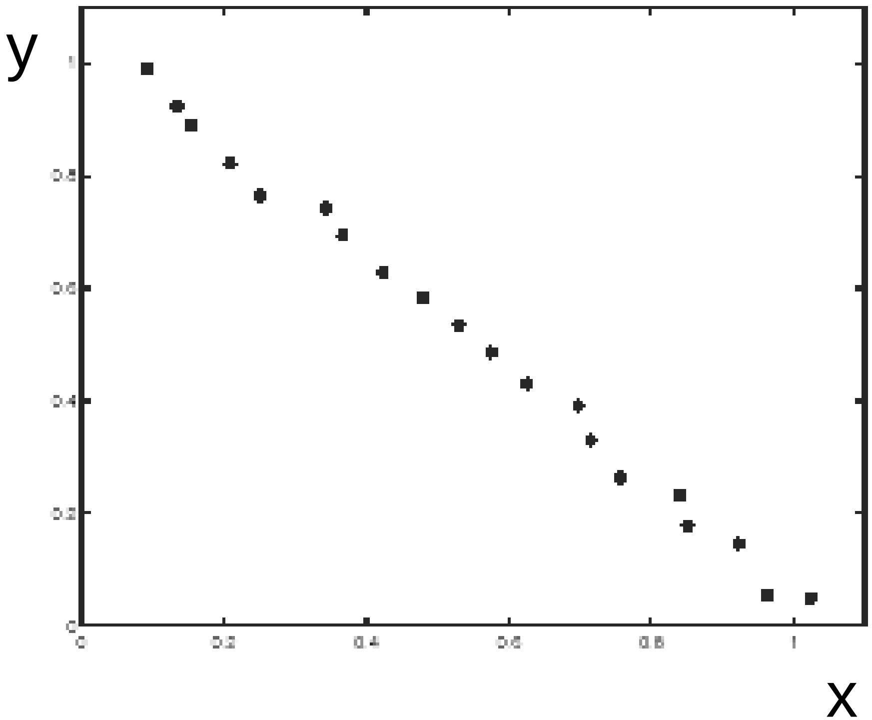

Image



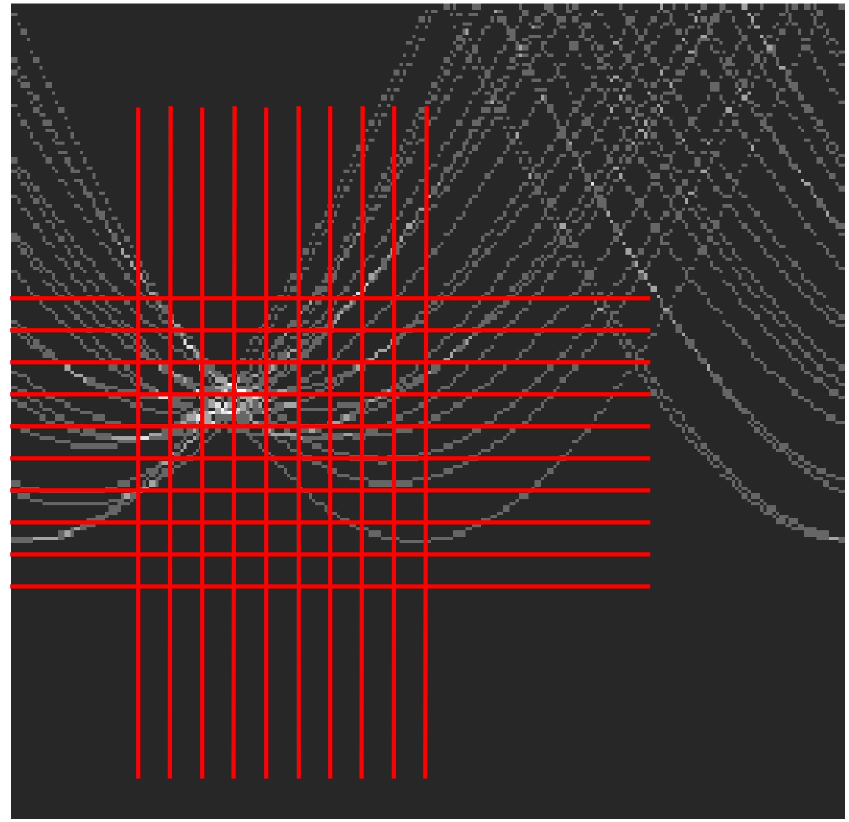

*r*,Θ model parameter histogram

# Hough transform - experiments

Noisy data




Image

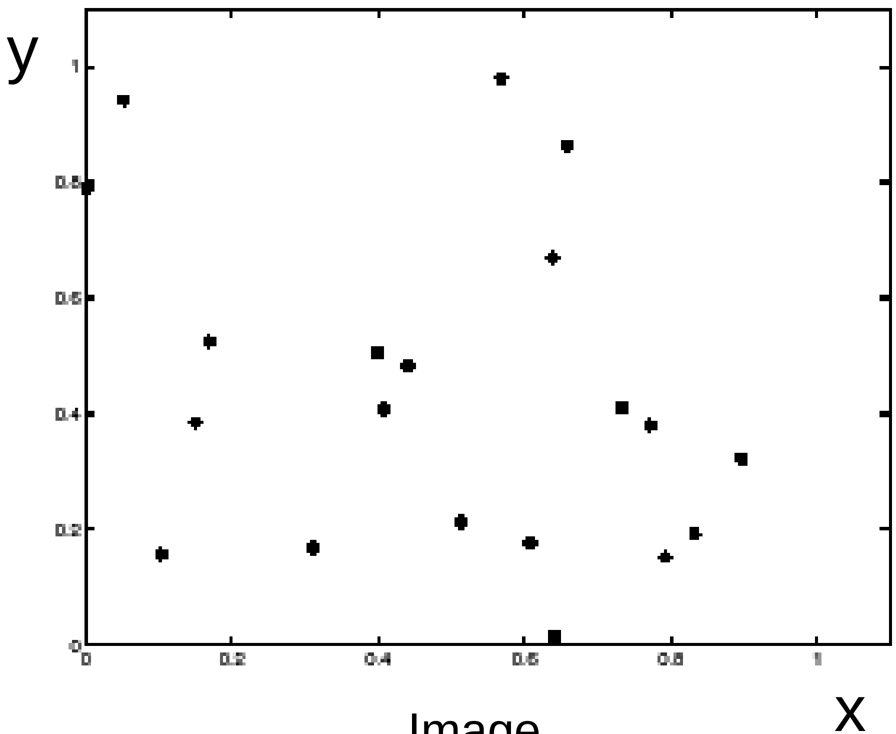


*r*,*Θ* model parameter histogram

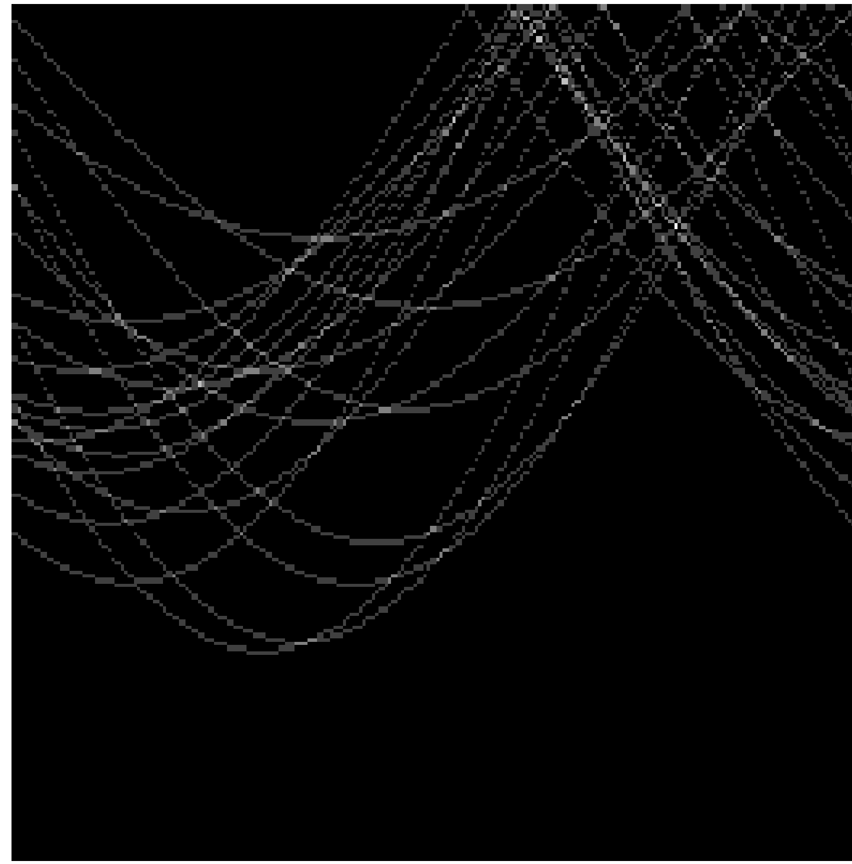
Need to adjust grid size or smooth

Slide from S. Savarese

# Hough transform - experiments




Image

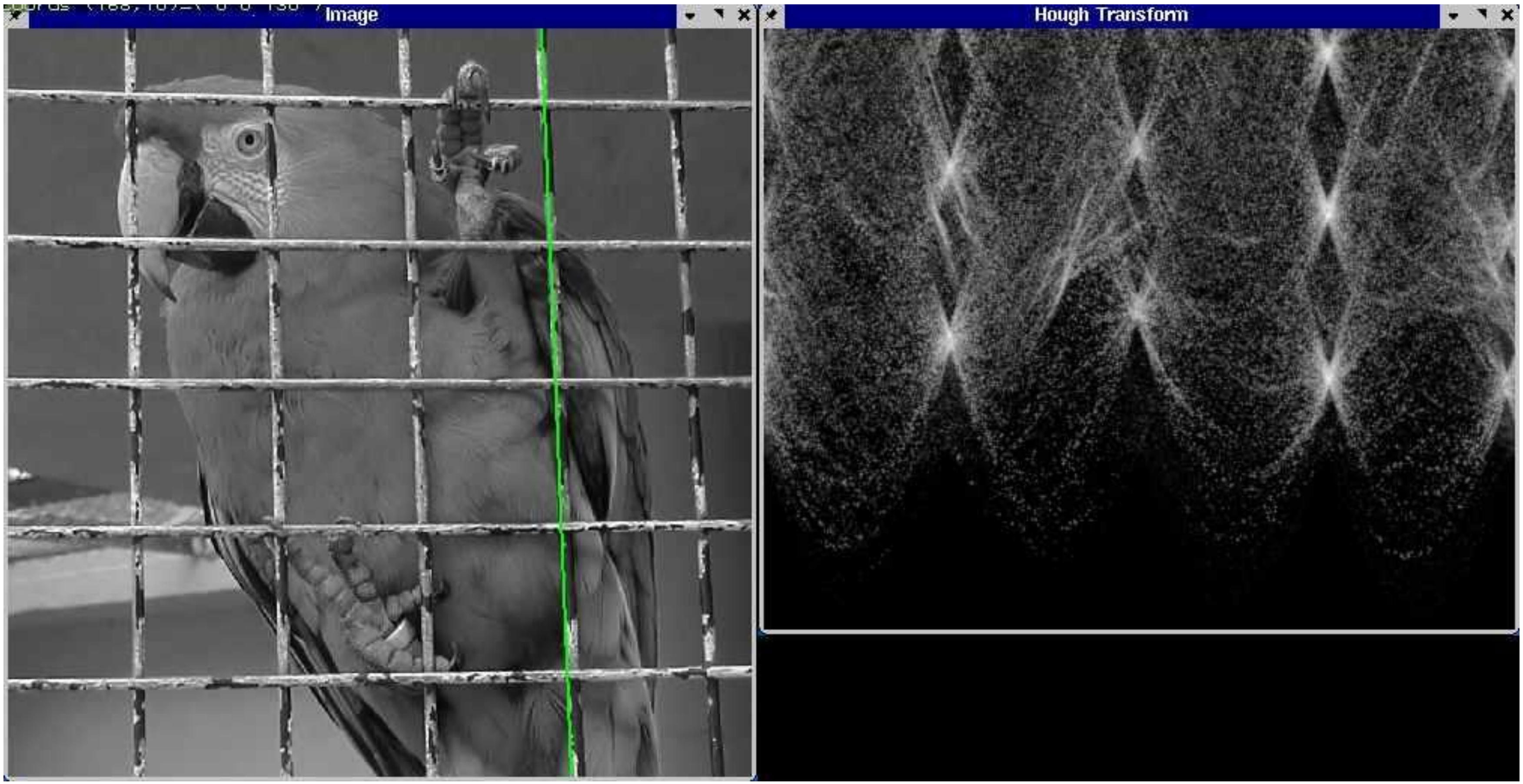


$\rho, \Theta$ model parameter histogram

Issue: spurious peaks due to uniform noise

Slide from S. Savarese

# Hough transform example



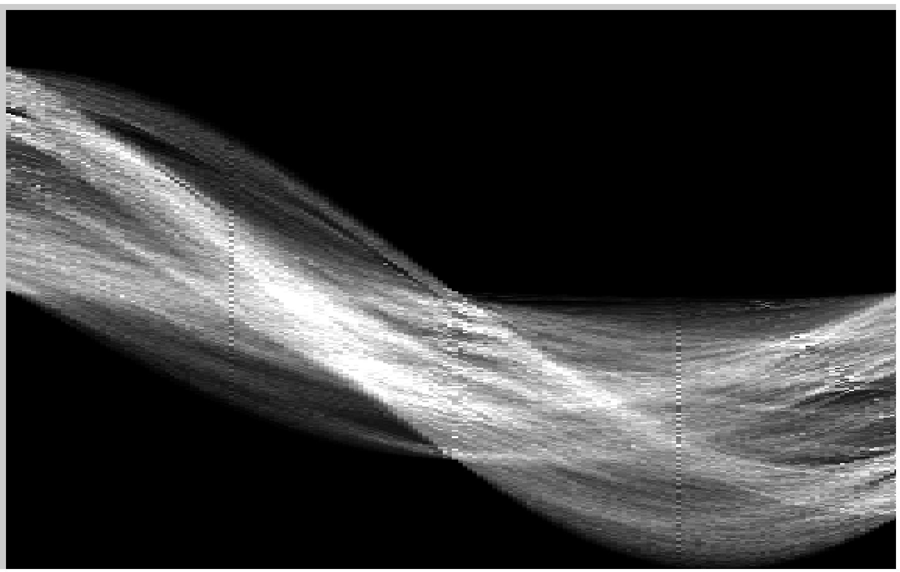

http://ostatic.com/files/images/ss_hough.jpg

# 1. Image → Canny

# 2. Canny → Hough votes

# 3. Hough votes → Edges

Find peaks and post-process.



# Finding lines using Hough transform

- Using known edge orientation to vote for a single line (rather than accumulate over all $\theta$).
- Practical considerations
  - Bin size
  - Smoothing
  - Finding multiple lines
  - Finding line segments
- Can 'fit' line to edgels that 'survive the vote' for more precise estimation.

# Hough transform conclusions

Good

- Robust to outliers: each point votes separately.
- Edge orientation -> fairly efficient (faster than trying all parameter sets).
- Provides multiple model fitting.

Bad

- Some sensitivity to noise
- Bin size trades off between noise tolerance, precision, and speed/memory
  - Can be hard to find sweet spot.
- Not suitable for more than a few parameters
  - Grid size grows exponentially.

Common applications

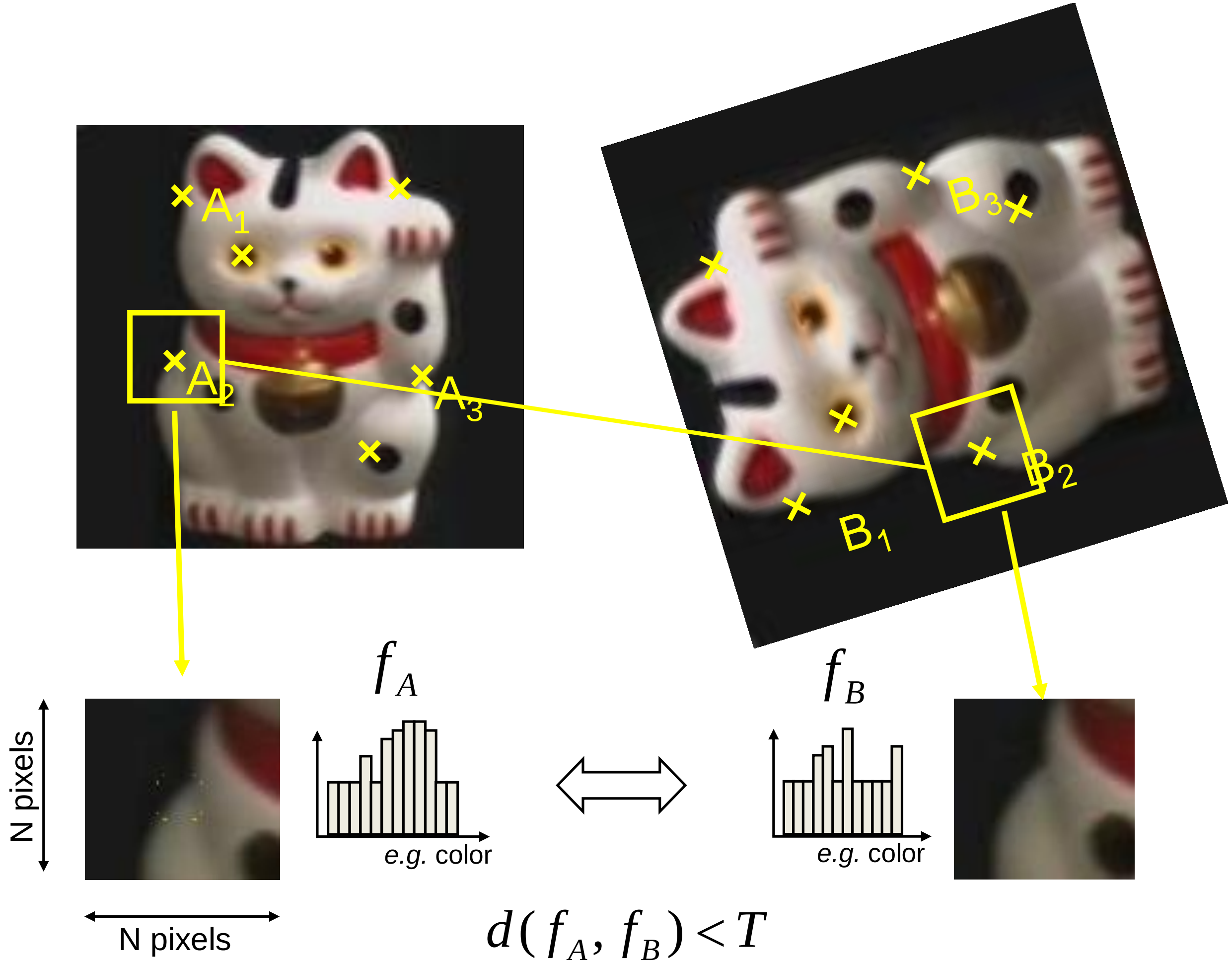
- Line fitting (also circles, ellipses, etc.)
- Object instance recognition (parameters are affine transform)
- Object category recognition  (parameters are position/scale)

James Hays

Computer Vision c. 2007

# FEATURE DETECTION AND MATCHING – DONE.

# Overview of Keypoint Matching

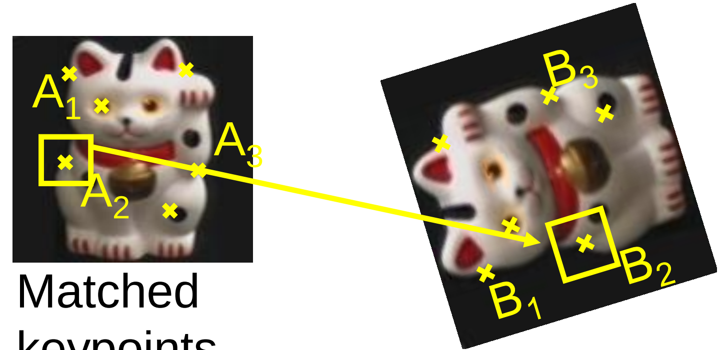



1. **Find a set of distinctive key-points**
2. **Define a region around each keypoint**
3. **Extract and normalize the region content**
4. **Compute a local descriptor from the normalized region**
5. **Match local descriptors**

K. Grauman, B. Leibe

# Object Instance Recognition

1. Match keypoints to object model
2. Solve for affine transformation parameters
3. Score by inliers and choose solutions with score above threshold



Matched keypoints

Affine Parameters

# Inliers

Choose hypothesis with max score above threshold

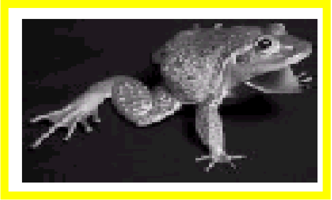
James Hays

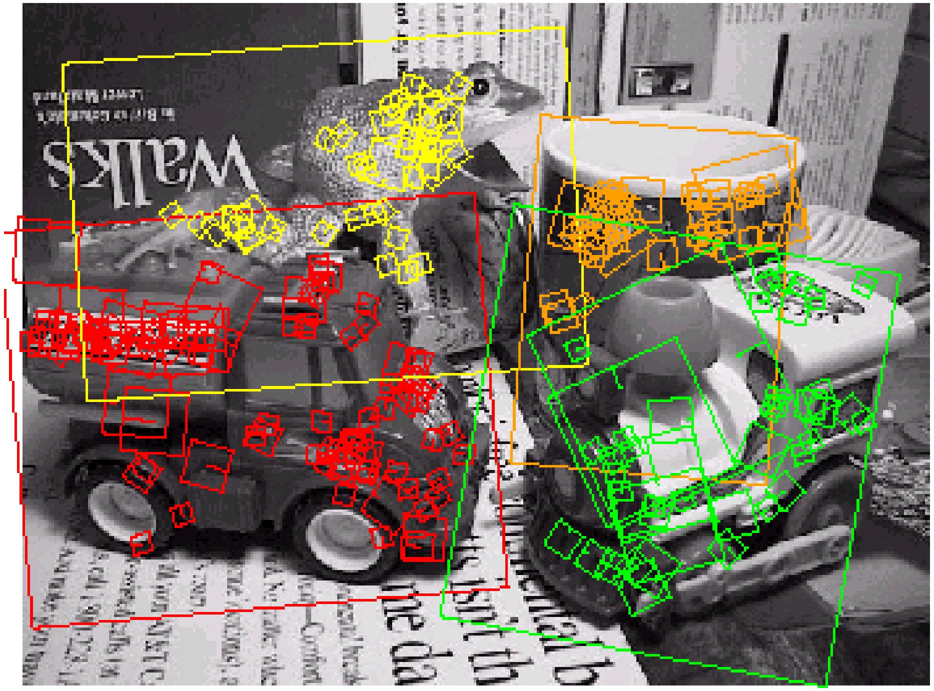
# Finding objects (SIFT, Lowe 2004)

1. Match interest points from input image to database image.
2. Get location/scale/orientation using Hough voting.
   - In database image, each point has known position/scale/orientation wrt. whole object.
   - Matched points vote for the position, scale, and orientation of the entire object.
   - Bins for x, y, scale, orientation
     - Wide bins (0.25 object length in position, 2x scale, 30 degrees orientation)
     - Vote for two closest bin centers in each direction (16 votes total)
3. Geometric verification for each bin with at least 3 keypoints
   - Iterate least squares fit and checking for inliers and outliers
   - *(Advanced)* Compute affine registration to check model fit.
4. Report object if > T inliers (T is typically 3, can be computed to match some probabilistic threshold)

James Hays

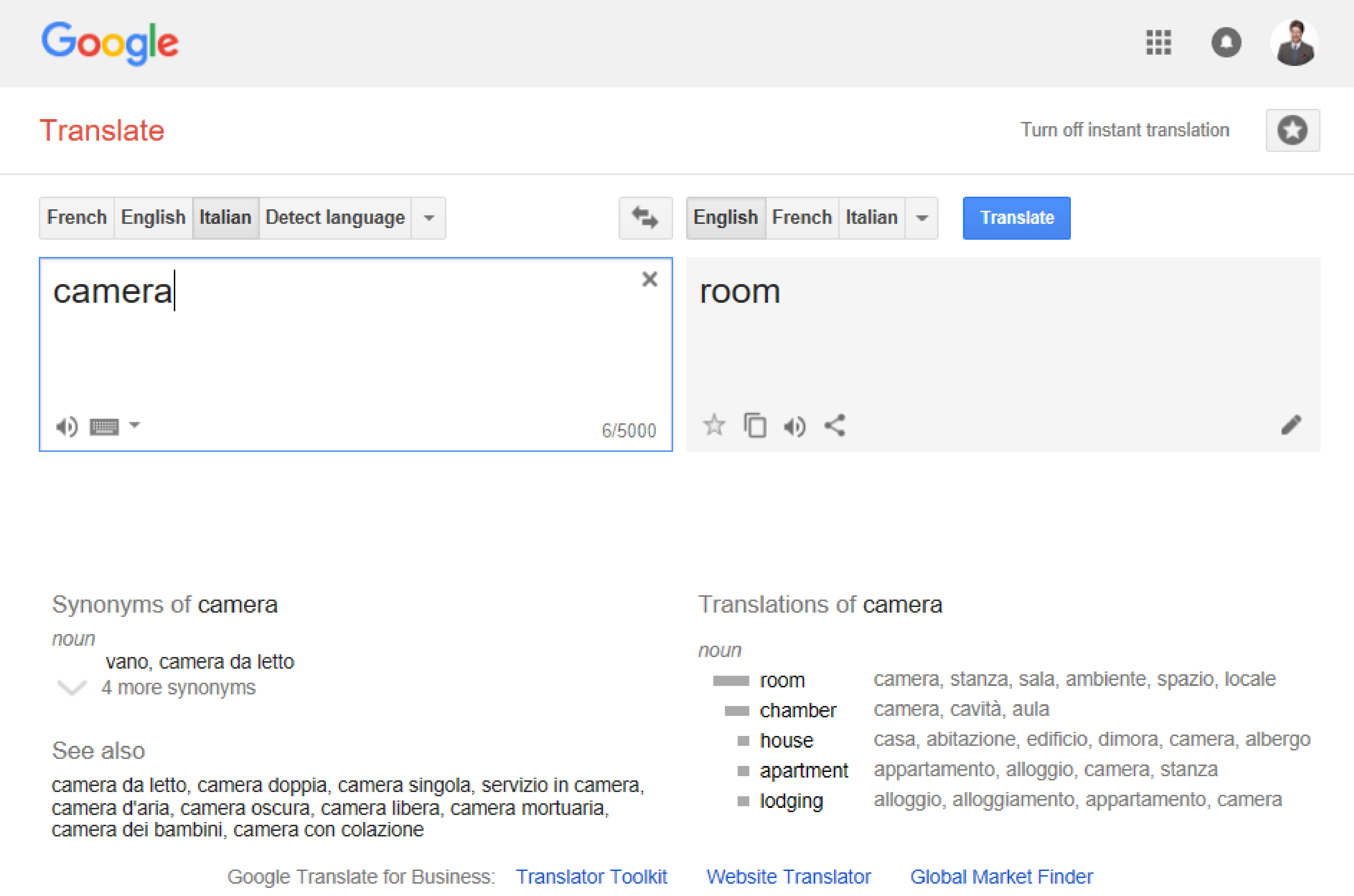
# Examples of recognized objects





# CAMERAS, MULTIPLE VIEWS, AND MOTION

# What is a camera?


Google
Translate
Turn off instant translation
French English Italian Detect language
English French Italian
Translate
camera
6/5000
room
Synonyms of camera
noun
vano, camera da letto
4 more synonyms
See also
camera da letto, camera doppia, camera singola, servizio in camera, camera d'aria, camera oscura, camera libera, camera mortuaria, camera dei bambini, camera con colazione
Translations of camera
noun
room camera, stanza, sala, ambiente, spazio, locale
chamber camera, cavità, aula
house casa, abitazione, edificio, dimora, camera, albergo
apartment appartamento, alloggio, camera, stanza
lodging alloggio, alloggiamento, appartamento, camera
Google Translate for Business: Translator Toolkit Website Translator Global Market Finder

# Camera obscura: dark room

- Known during classical period in China and Greece (e.g., Mo-Ti, China, 470BC to 390BC)

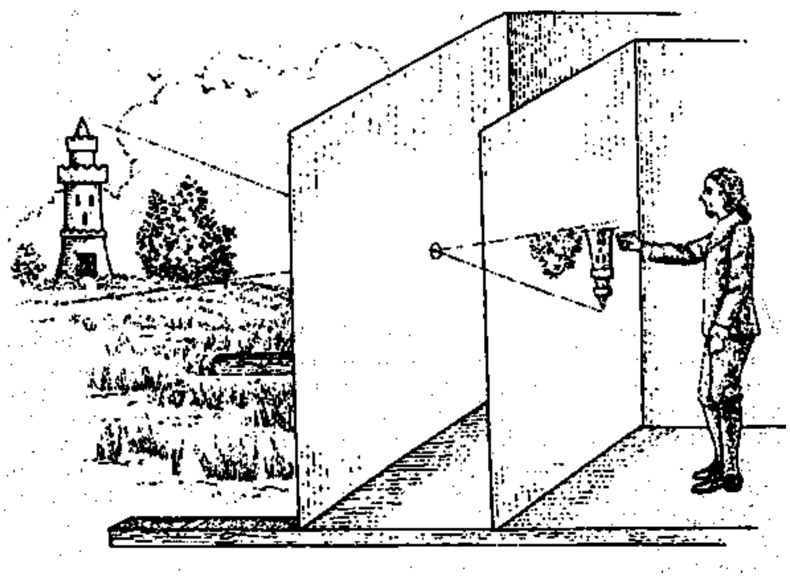
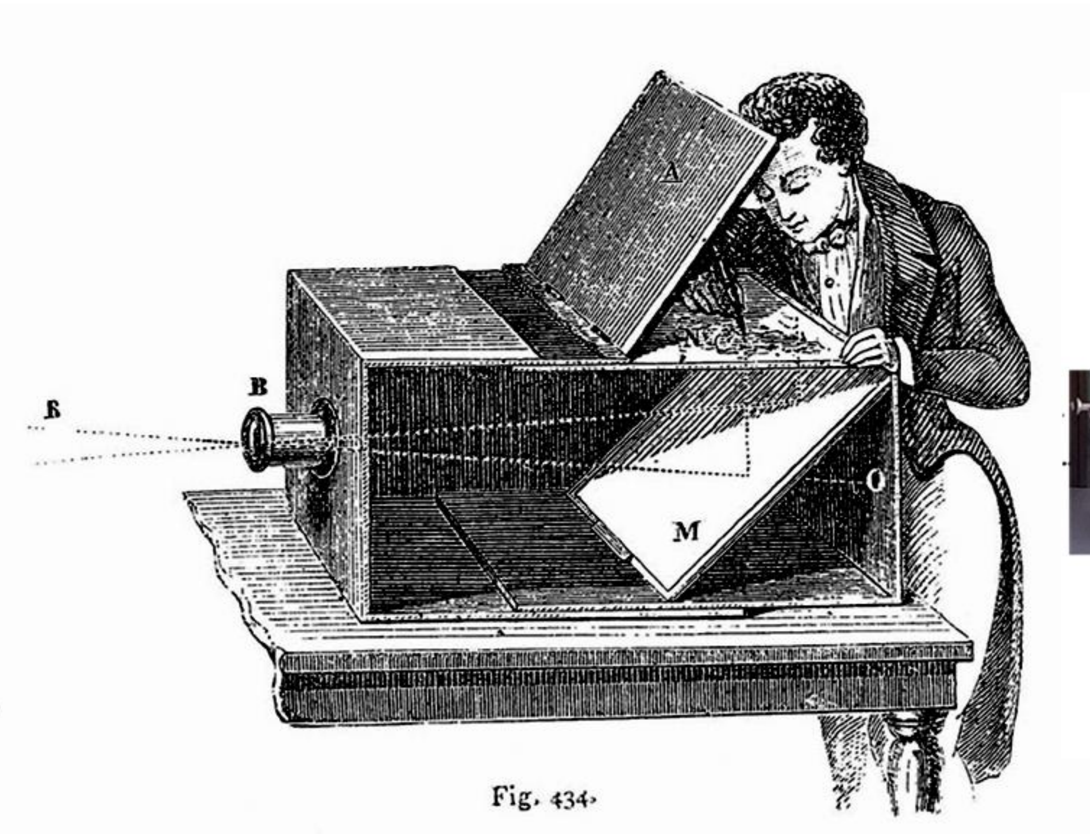


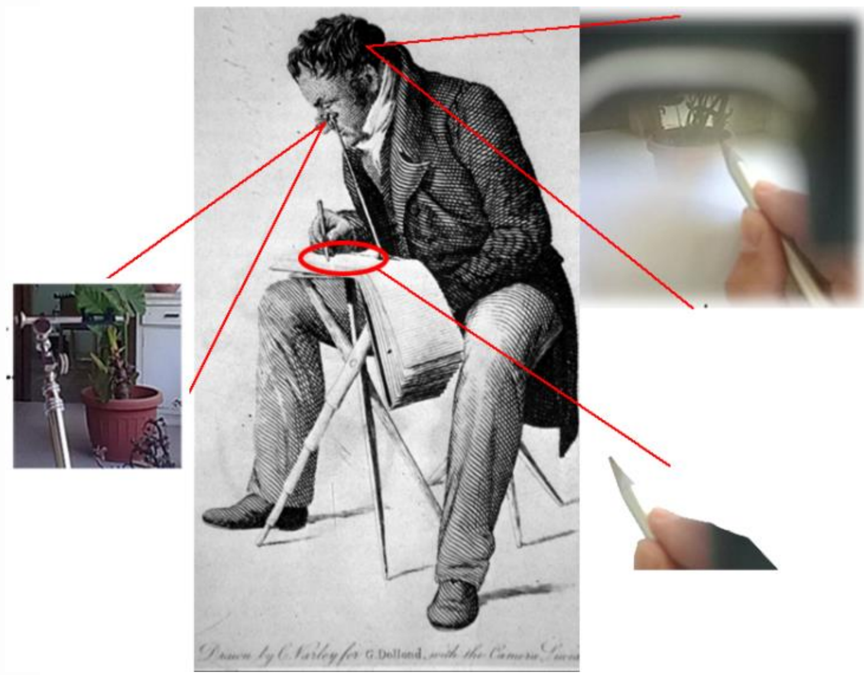
Illustration of Camera Obscura

Freestanding camera obscura at UNC Chapel Hill

Photo by Seth Ilys

# Camera obscura / lucida used for tracing



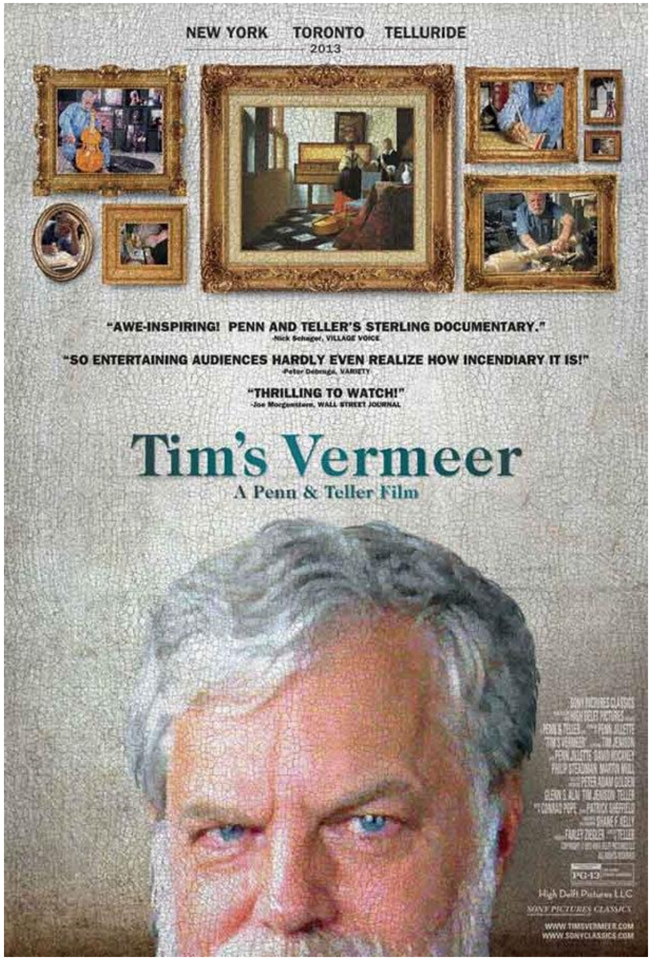



Lens Based Camera Obscura, 1568

Camera lucida

# Tim's Vermeer

Vermeer, The Music Lesson, 1665




Tim Jenison (Lightwave 3D, Video Toaster)

# Tim’s Vermeer – video still

# 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

# The Geometry of Image Formation

## Szeliski 2.1, parts of 2.2

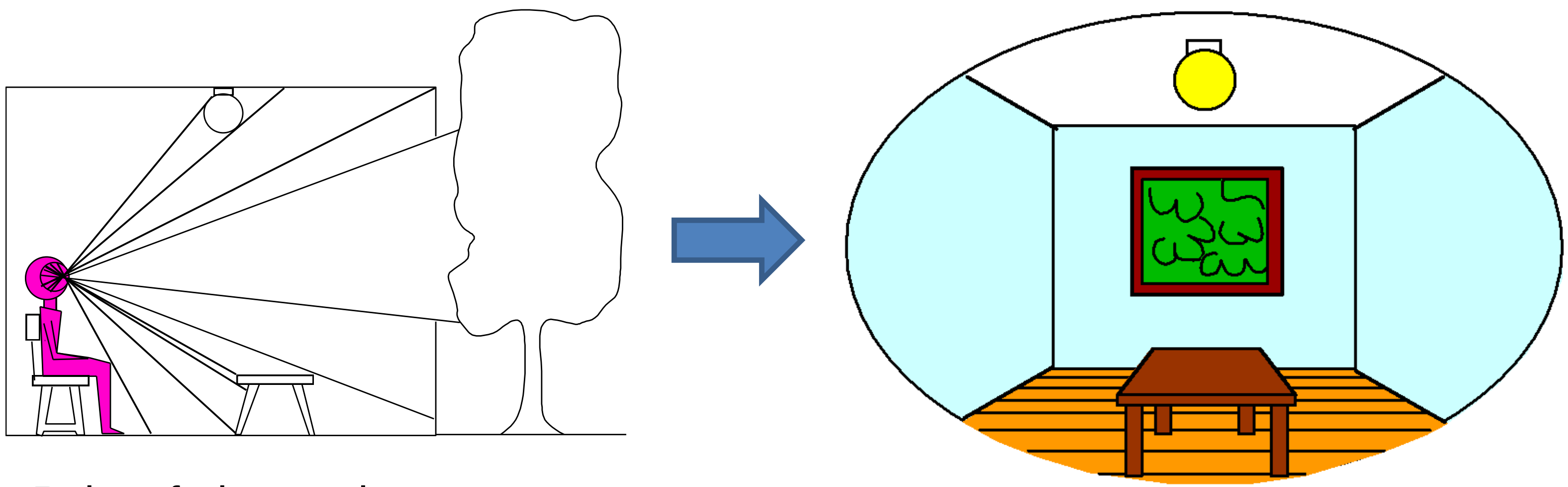
Mapping between image and world coordinates

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

Slides from James Hays, Derek Hoiem, Alexei Efros, Steve Seitz, and David Forsyth

# Dimensionality Reduction Machine (3D to 2D)

*3D world* *2D image*



Point of observation

# Lake Sørvágsvatn in Faroe Islands

100 metres above sea level

# Lake Sørvágsvatn in Faroe Islands

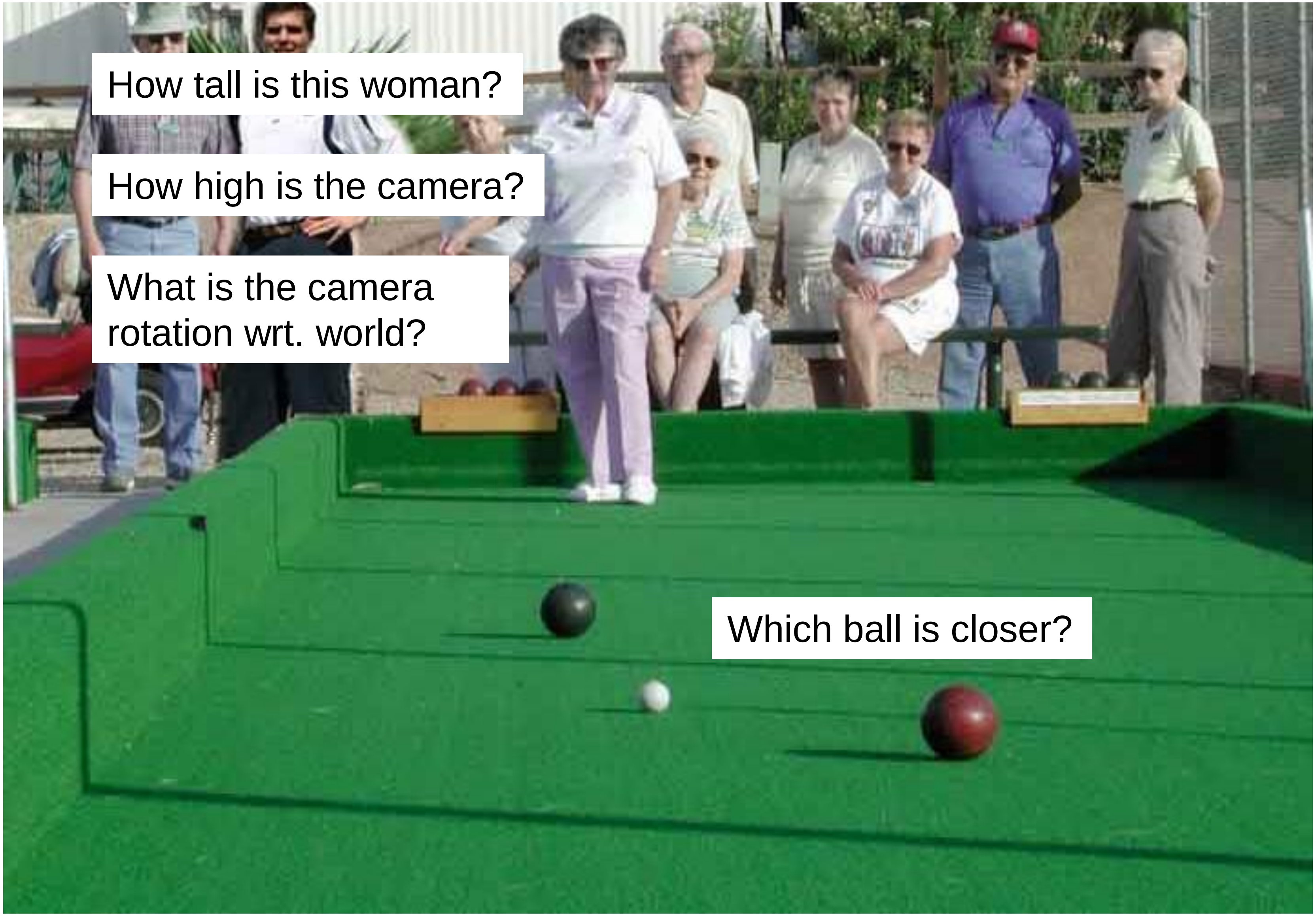
~~100~~ 30 metres above sea level

amusingplanet.com, thanks to Aaron Gokaslan

# Holbein's The Ambassadors - 1533

# Holbein's The Ambassadors – Memento Mori

# Cameras and World Geometry

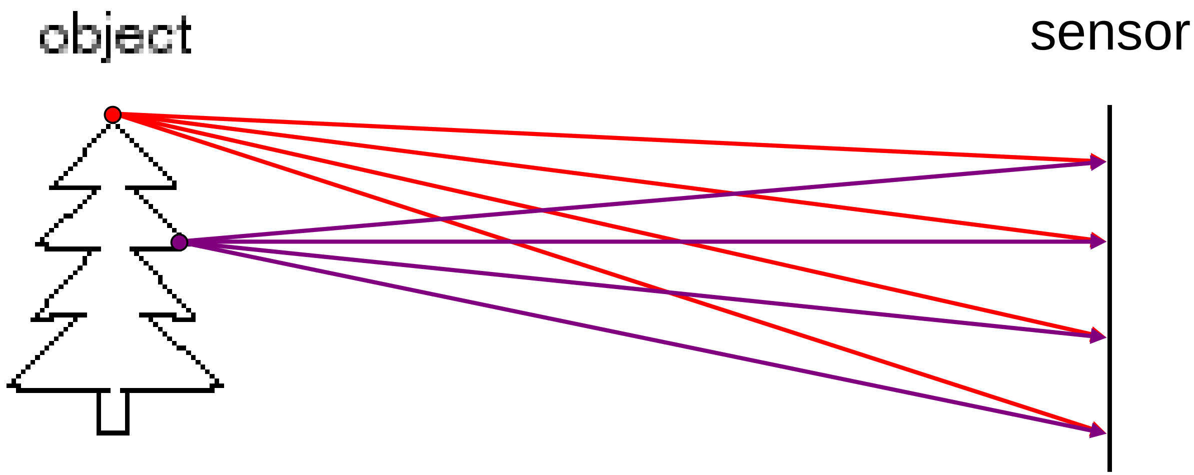



James Hays

# Let’s design a camera

Idea 1: Put a sensor in front of an object

Do we get a reasonable image?




Slide source: Seitz

# Let’s design a camera

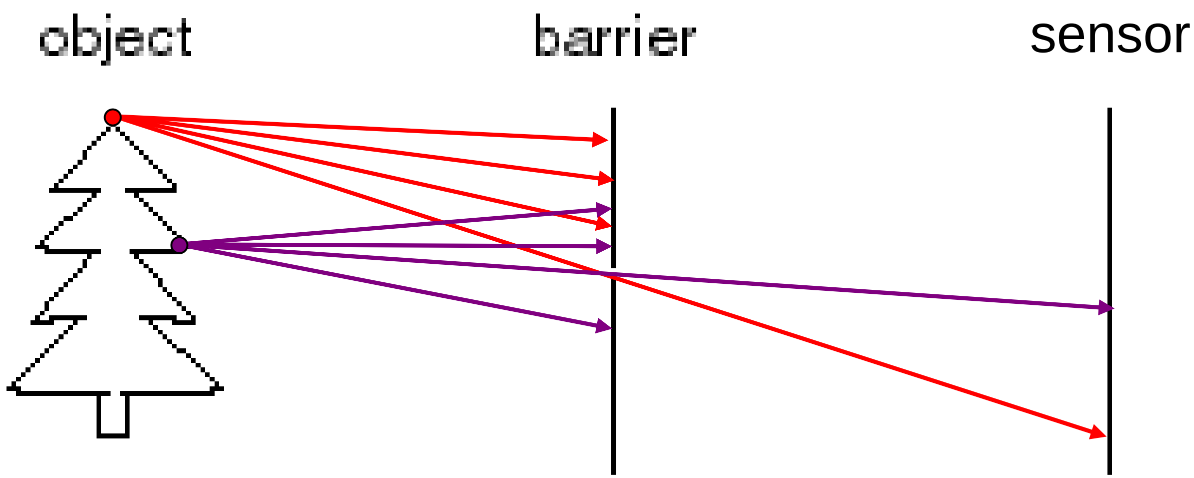
Idea 2: Add a barrier to block most rays

– Pinhole in barrier

– Only sense light from one direction.

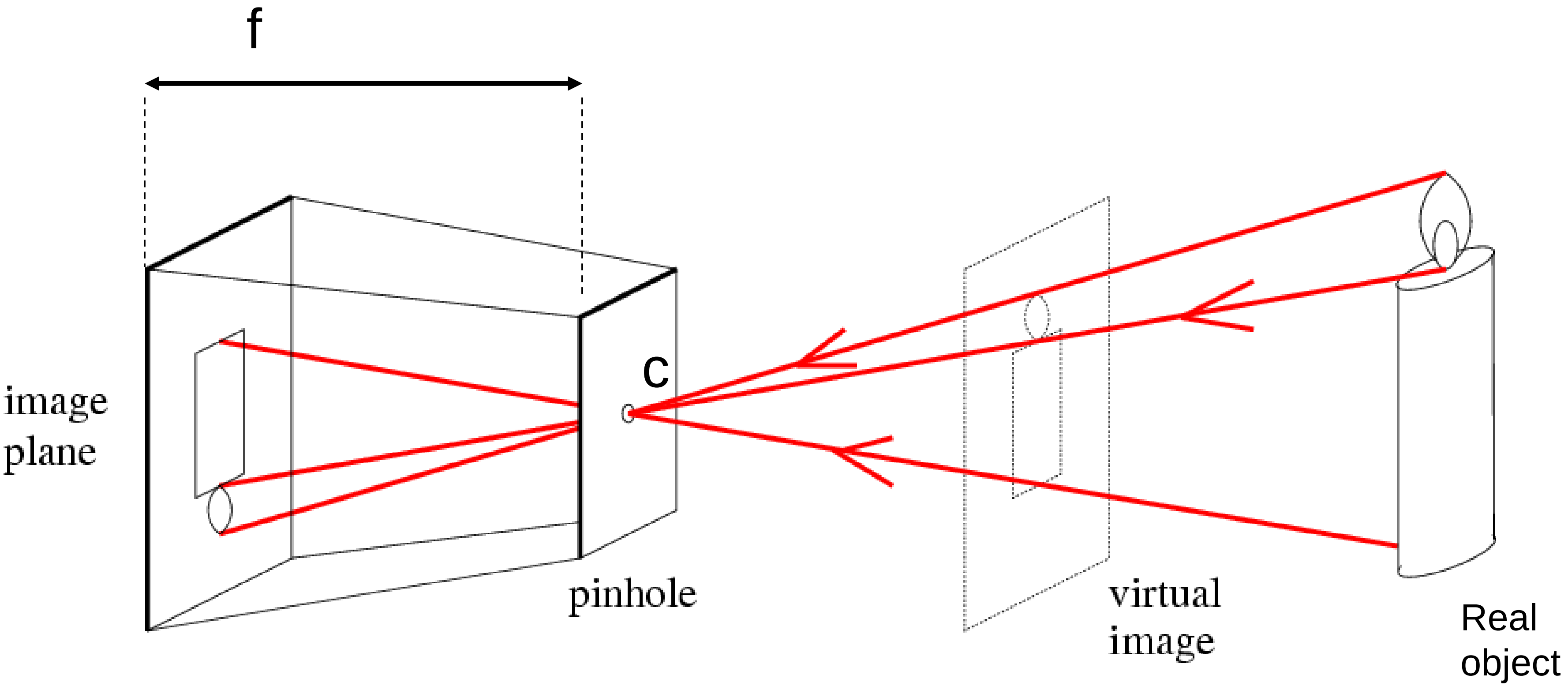
  - Reduces blurring.

– In most cameras, this **aperture** can vary in size.




Slide source: Seitz

# Pinhole camera model

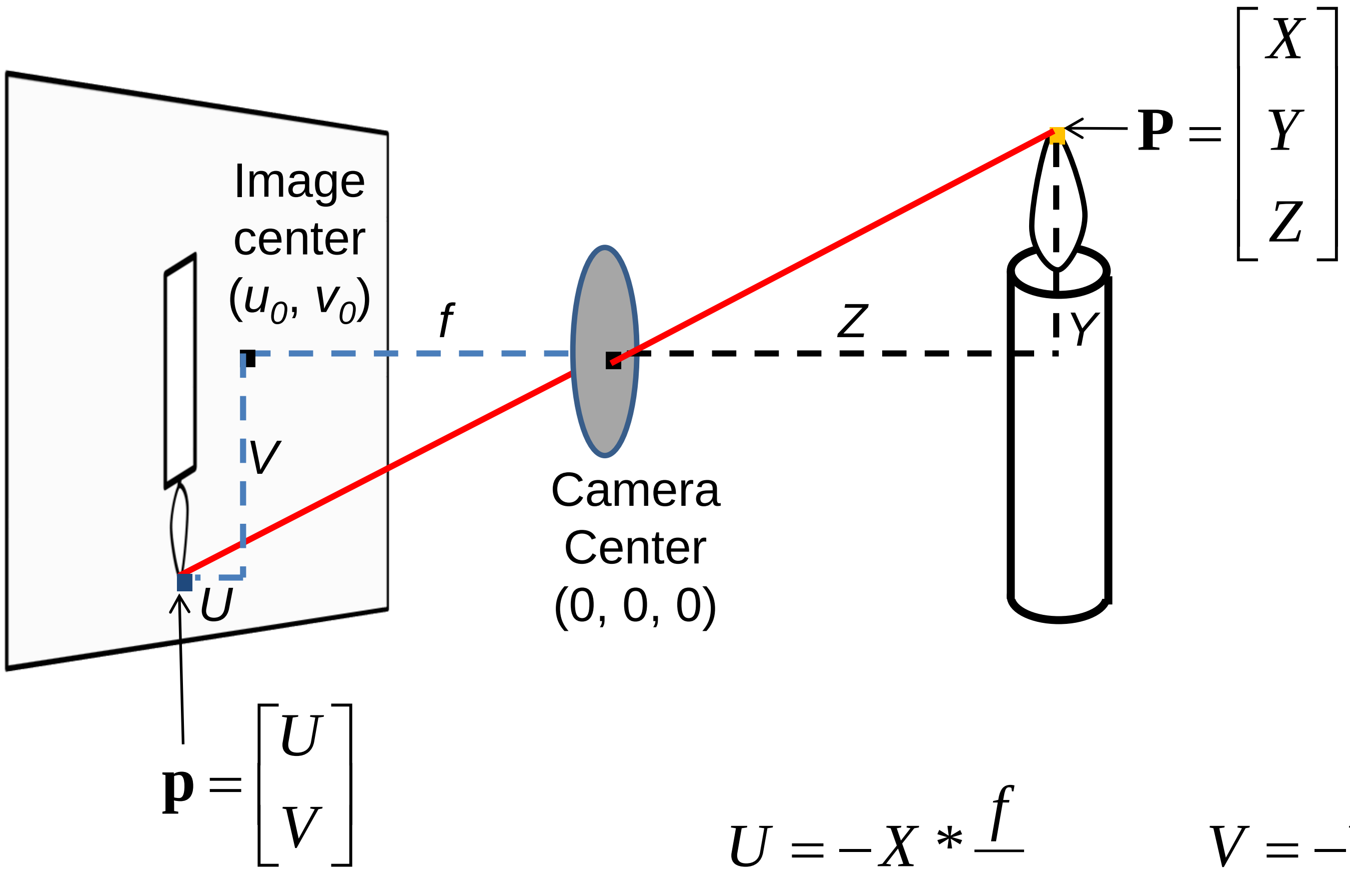



f = Focal length

c = Optical center of the camera

Figure from Forsyth

# Projection: world coordinates→image coordinates



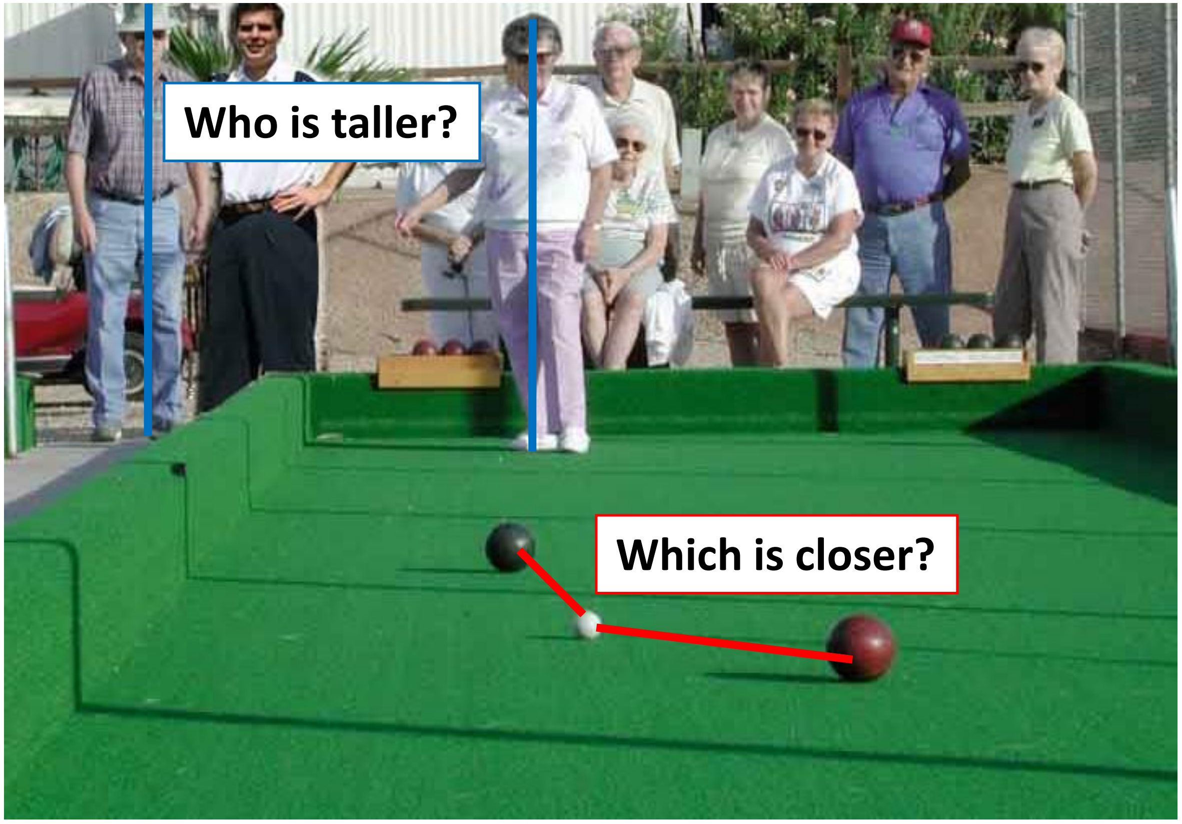

**p** = distance from image center

$$U = -X * \frac{f}{Z} \qquad V = -Y * \frac{f}{Z}$$

What is the effect if f and Z are equal?

# Projective Geometry

Length (and so area) is lost.



# Length and area are not preserved

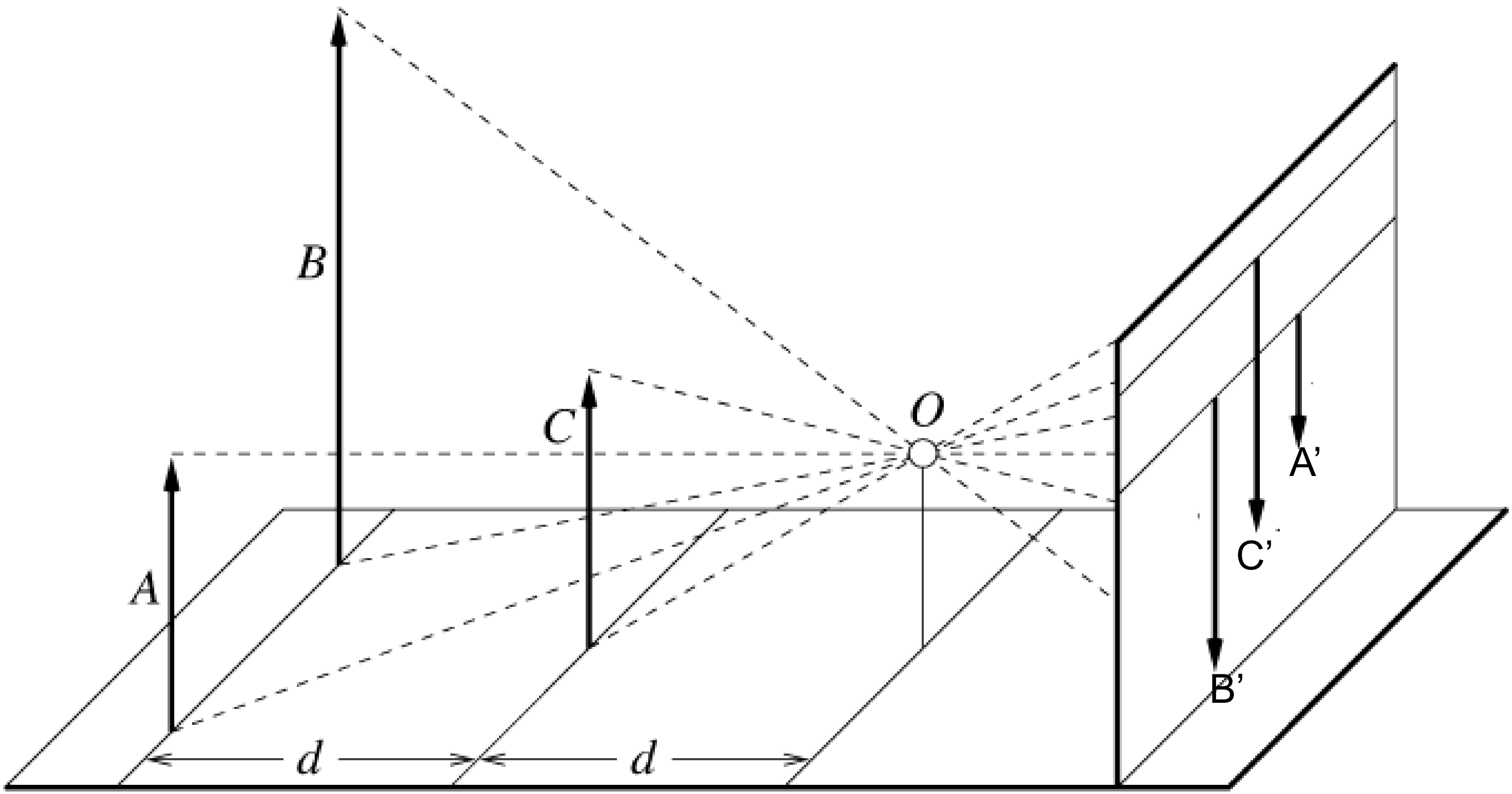
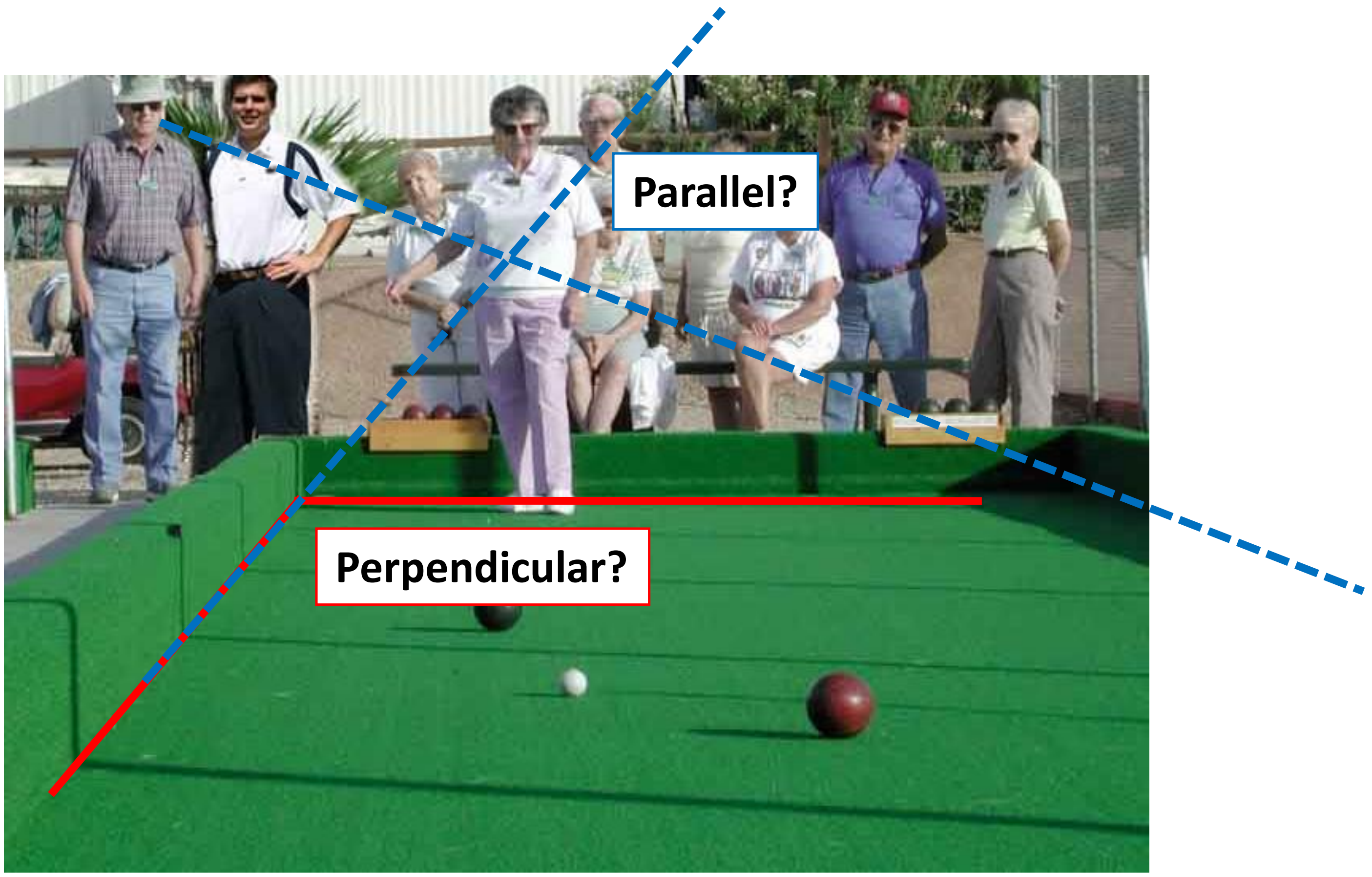



Figure by David Forsyth

# Projective Geometry

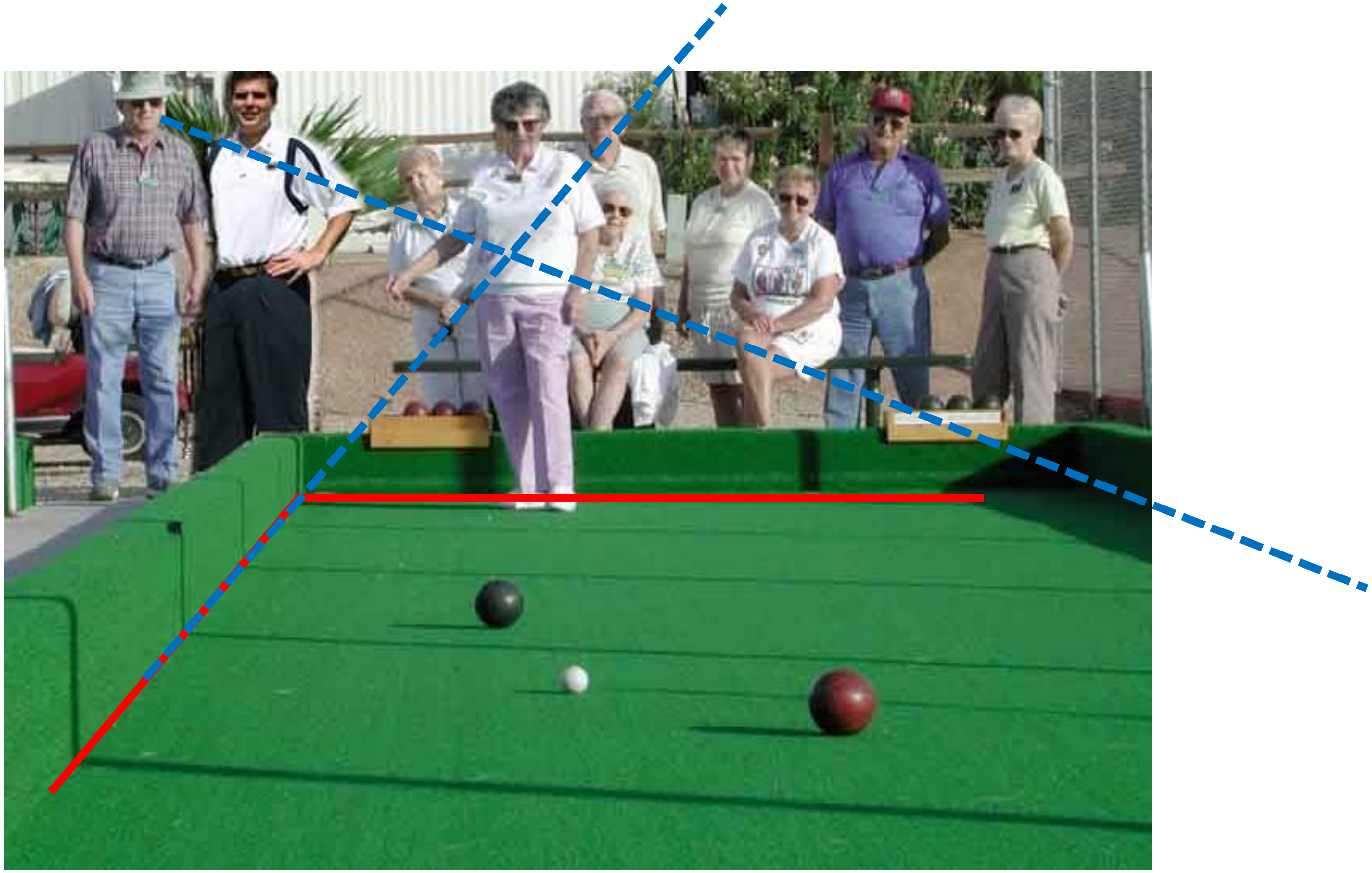
Angles are lost.



# Projective Geometry

## What is preserved?

- Straight lines are still straight.

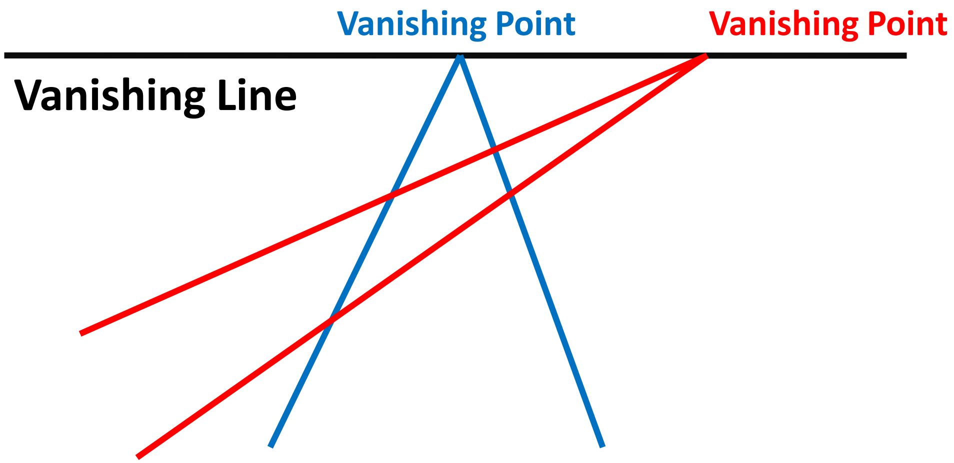


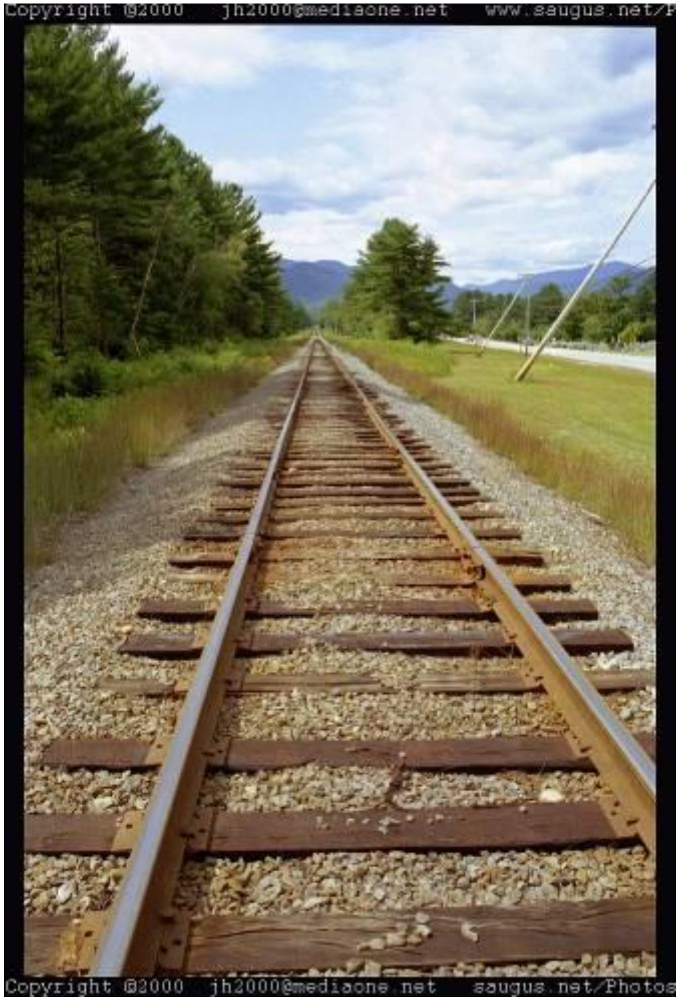
# Vanishing points and lines

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

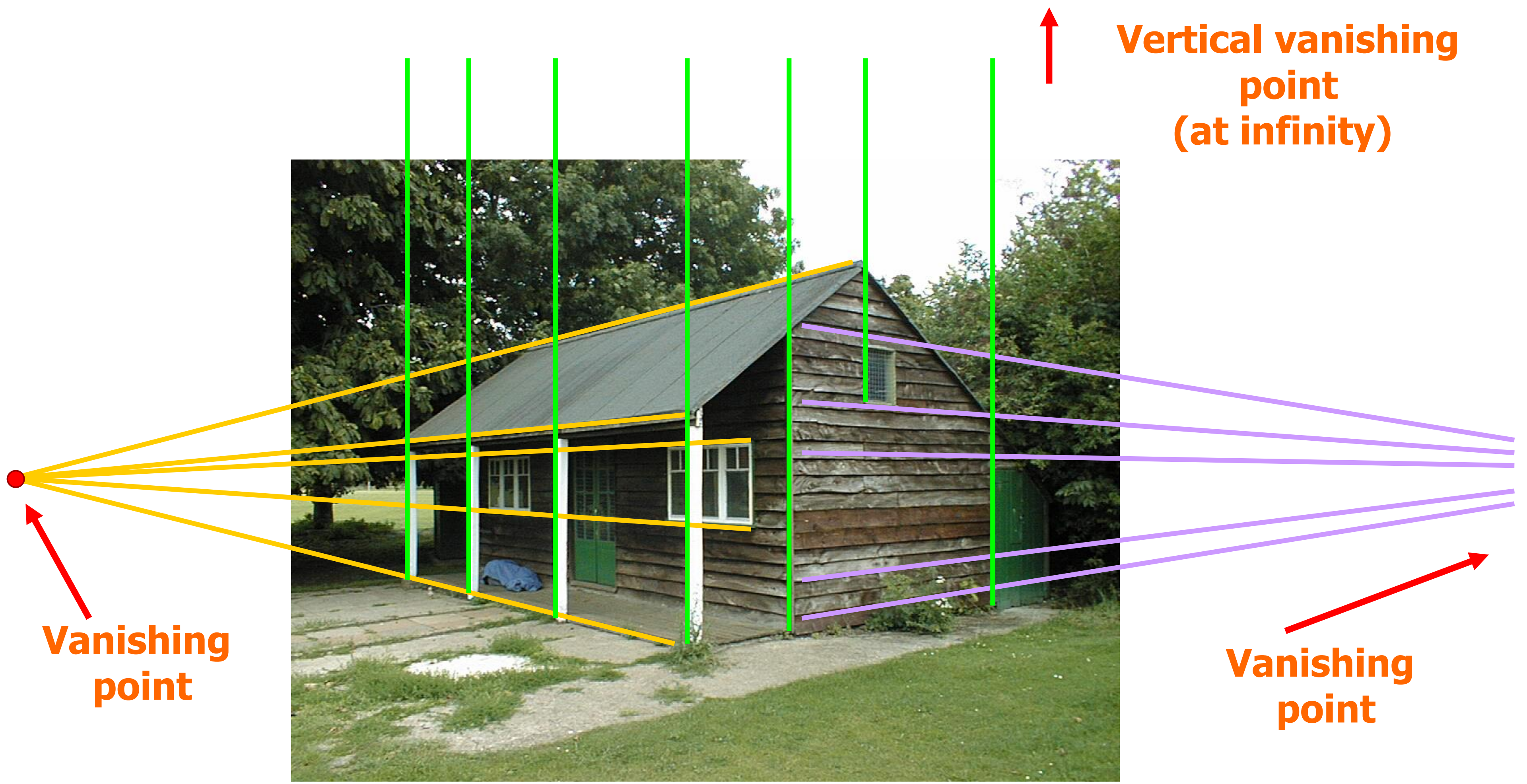
Parallel lines on the same plane in the world converge to vanishing points on a “vanishing line”.

E.G., the horizon.



# Vanishing points and lines




Slide from Efros, Photo from Criminisi

# Photo Tourism

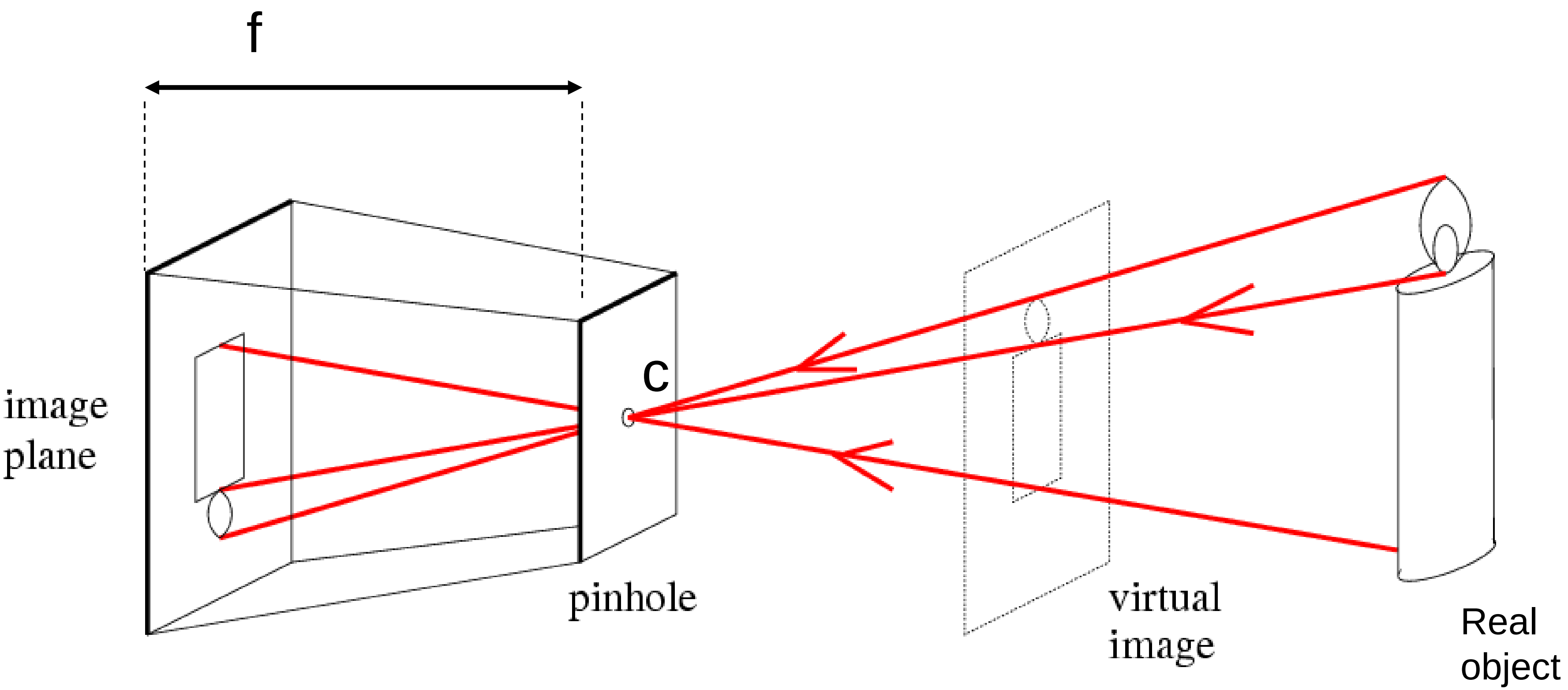
## Exploring photo collections in 3D

Noah Snavely    Steven M. Seitz    Richard Szeliski

*University of Washington*    *Microsoft Research*

SIGGRAPH 2006

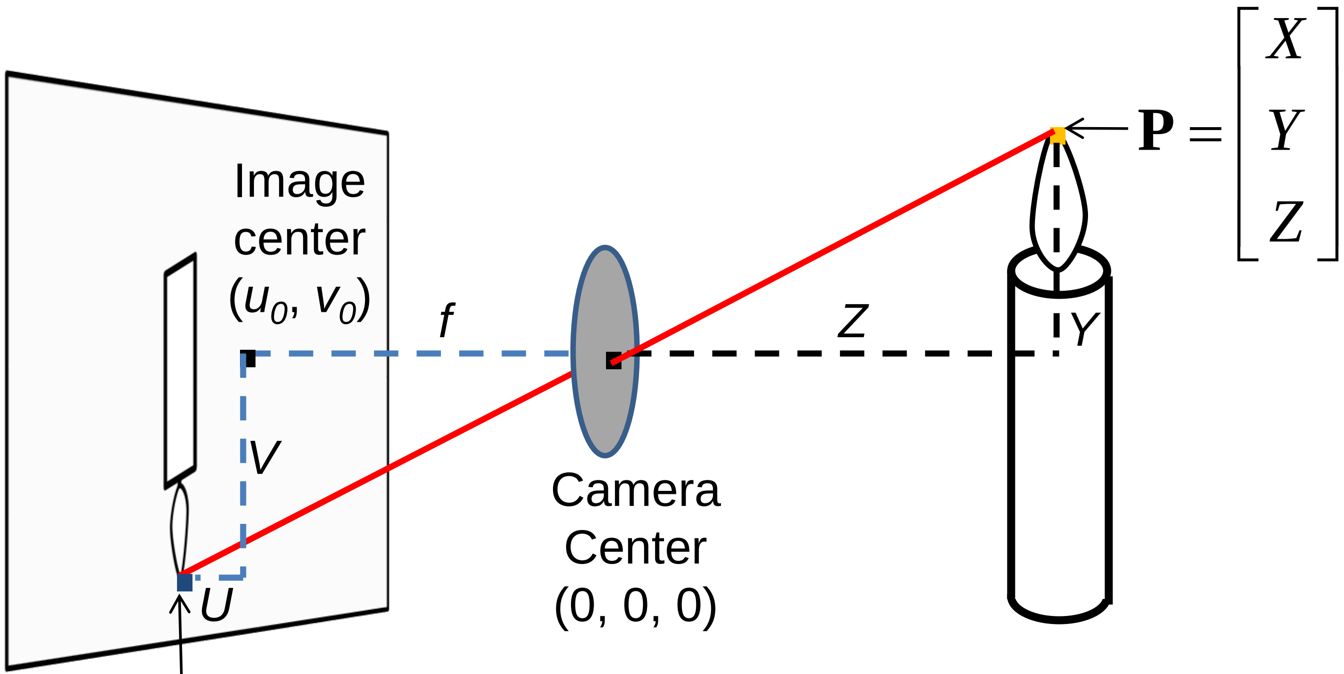
# Pinhole camera model




f = Focal length
c = Optical center of the camera

Forsyth

# Projection: world coordinates→image coordinates




$$\mathbf{p} = \begin{bmatrix} U \\ V \end{bmatrix}$$

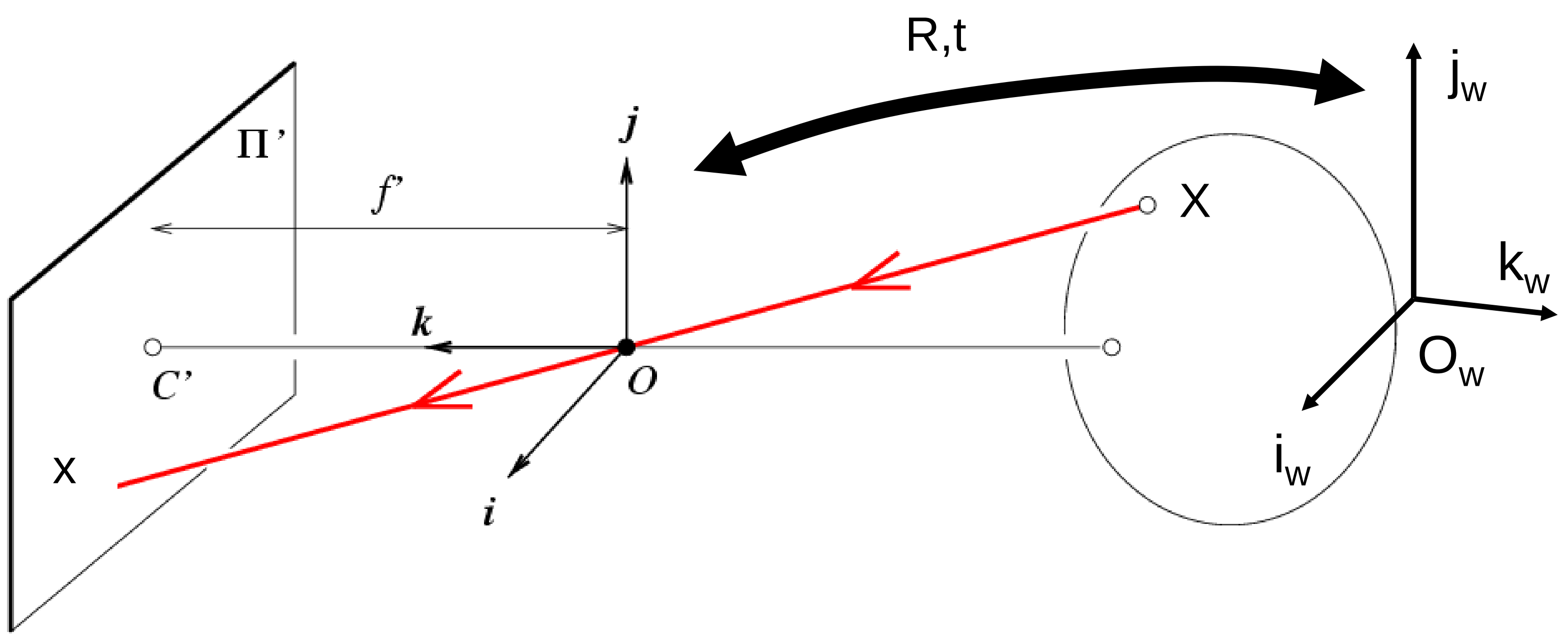
**p** = distance from image center

$$U = -X * \frac{f}{Z} \qquad V = -Y * \frac{f}{Z}$$

What is the effect if f and Z are equal?

Slide Credit: Savarese

# Camera (projection) matrix

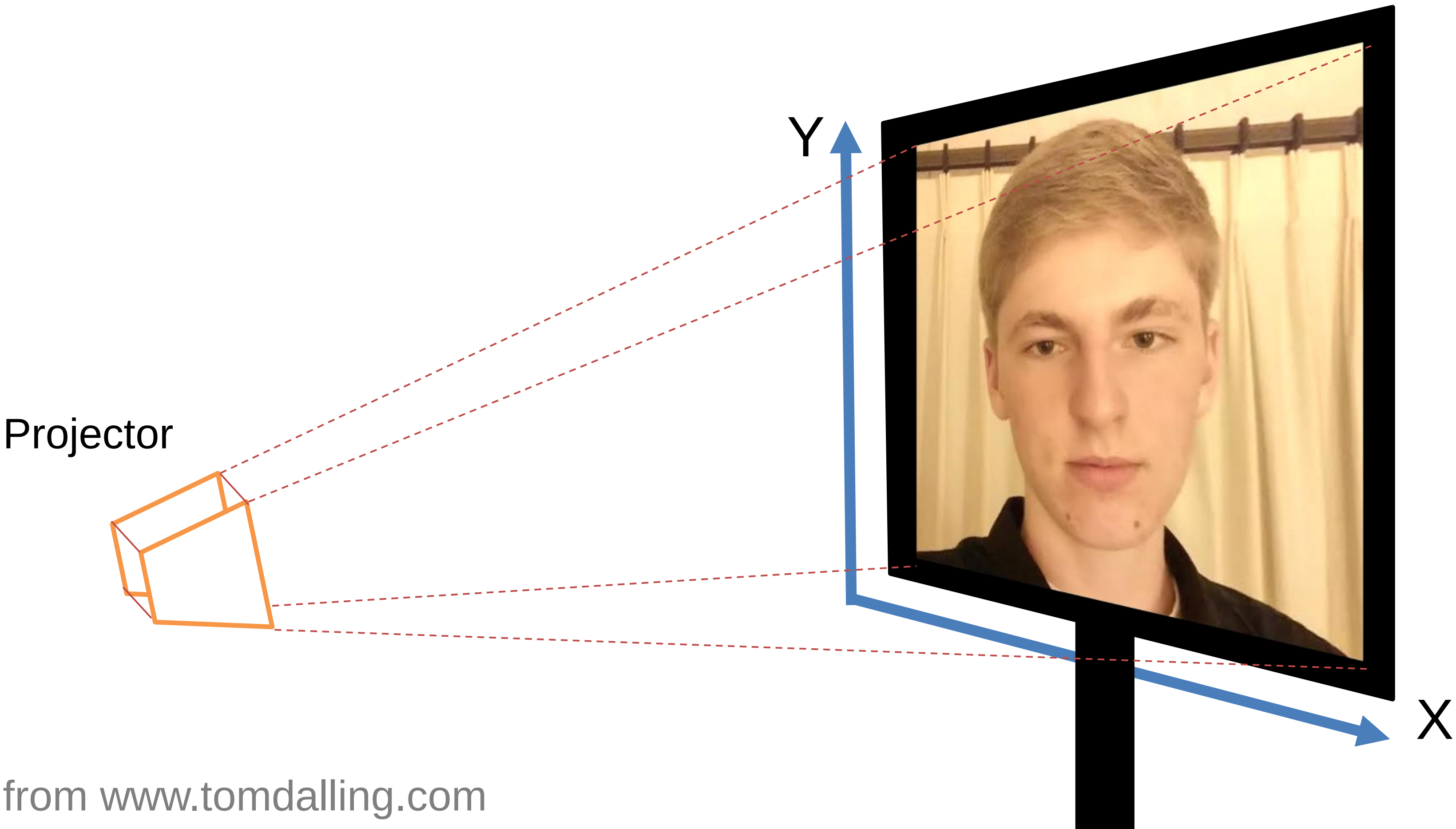



$$\mathbf{x} = \mathbf{K}\underbrace{[\mathbf{R} \quad \mathbf{t}]}_{\text{Extrinsic Matrix}}\mathbf{X}$$

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

# Projective geometry

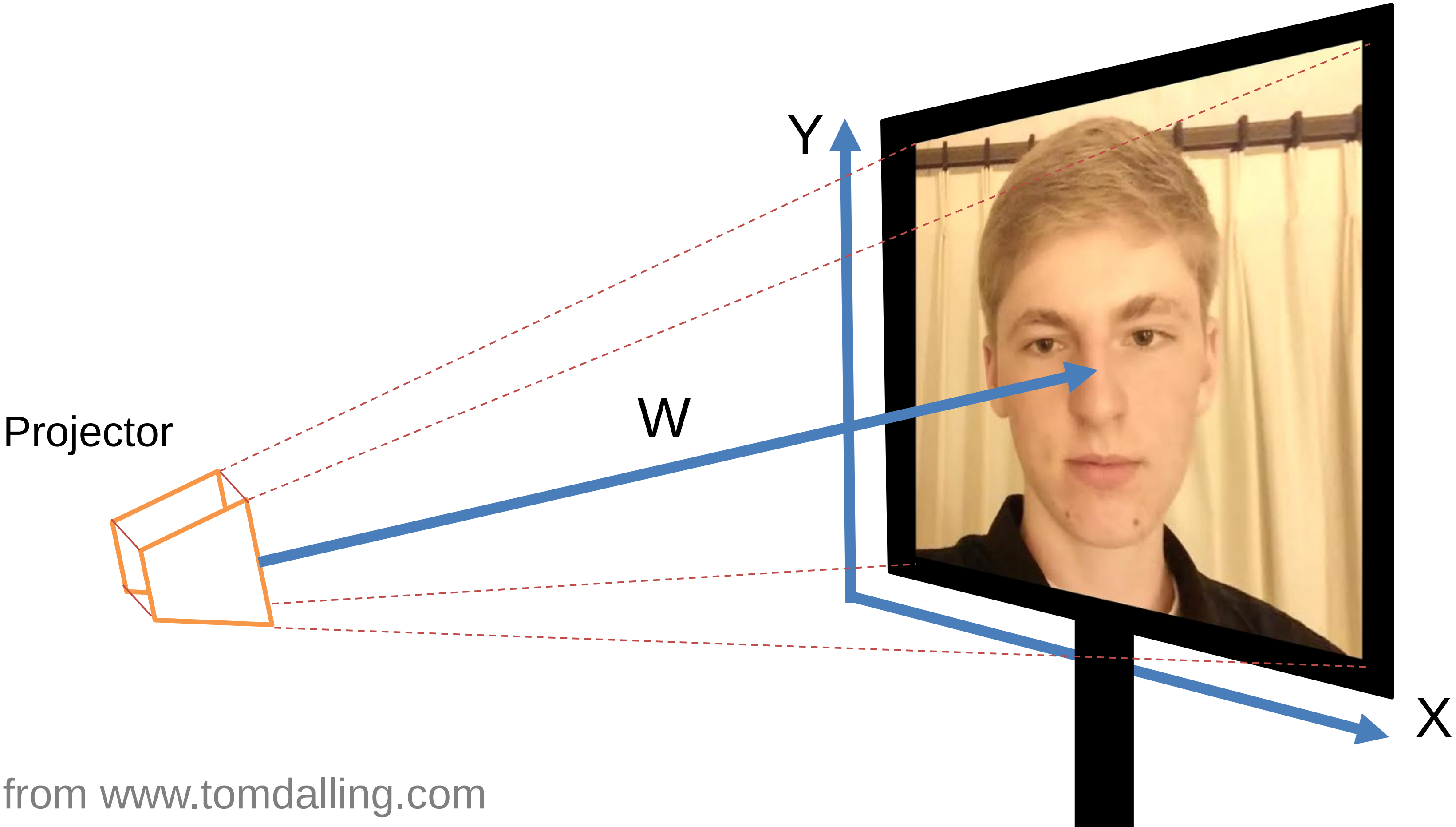
- 2D point in cartesian = (x,y) coordinates
- 2D point in projective = (x,y,w) coordinates




Idea from www.tomdalling.com

# Projective geometry

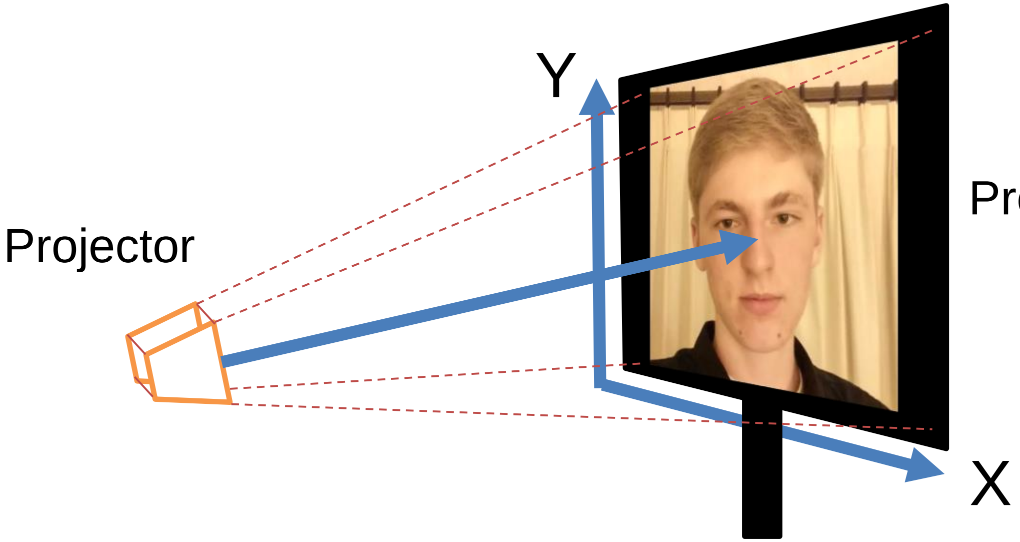
- 2D point in cartesian = (x,y) coordinates
- 2D point in projective = (x,y,w) coordinates



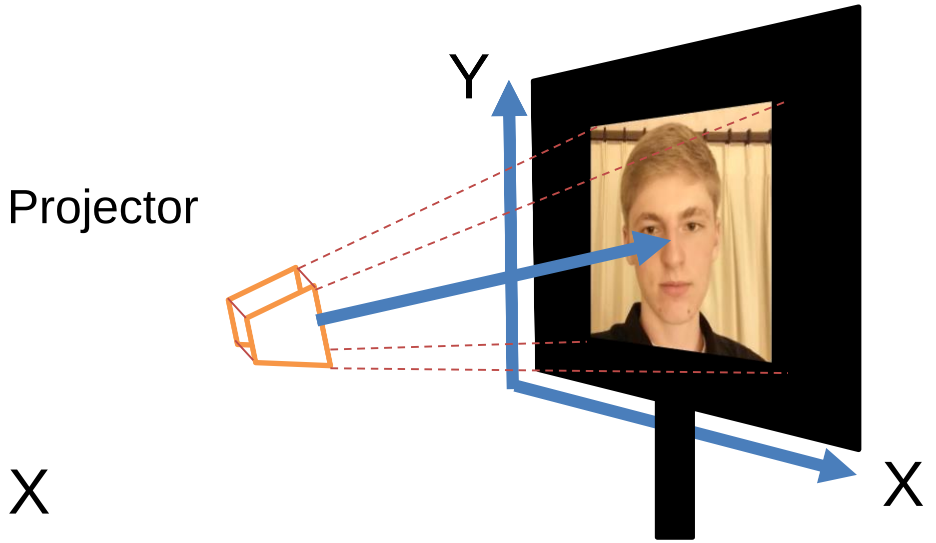

Idea from www.tomdalling.com

# Varying w

$w_1$



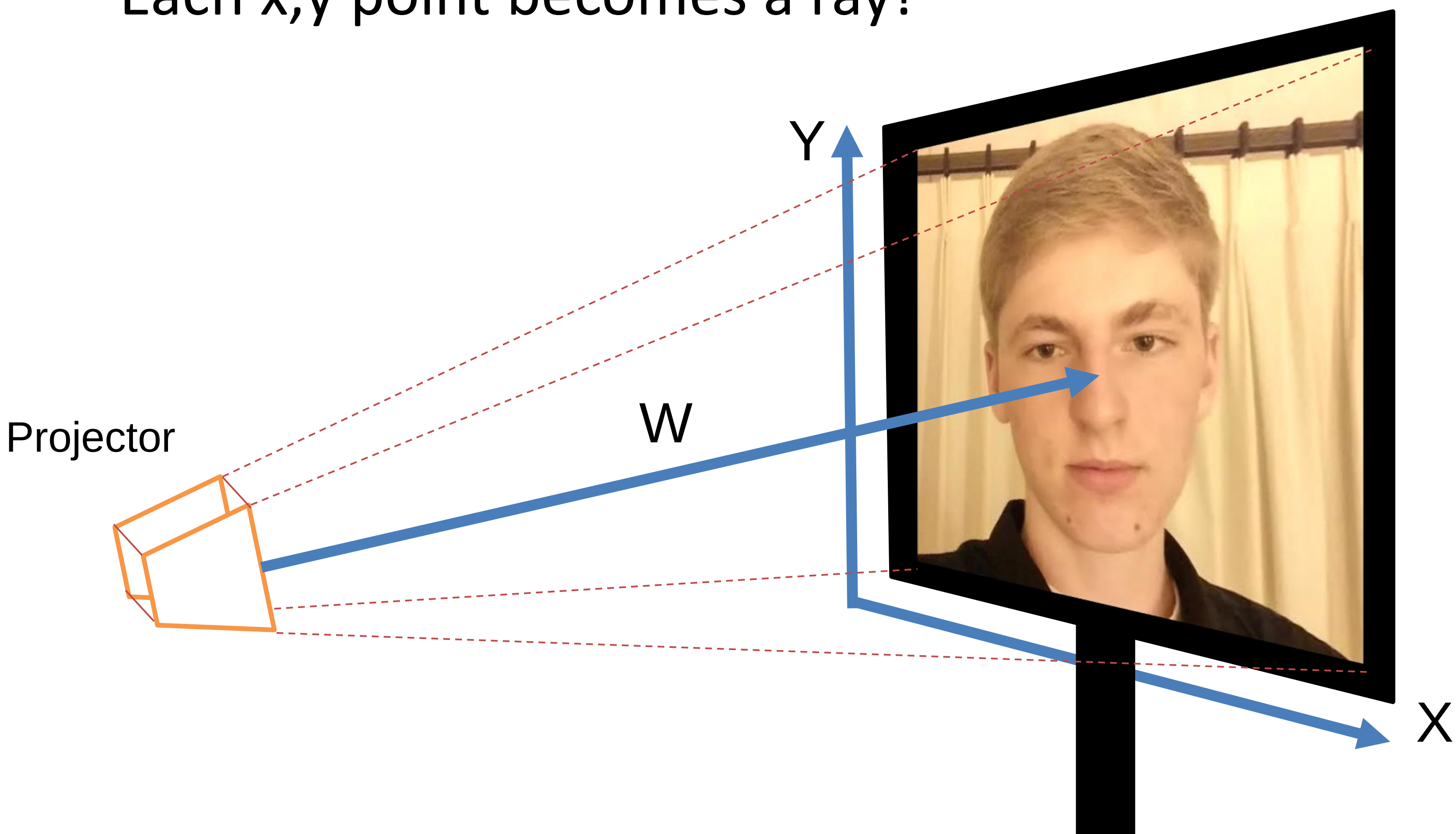

$w_2 < w_1$




Projected image becomes smaller.

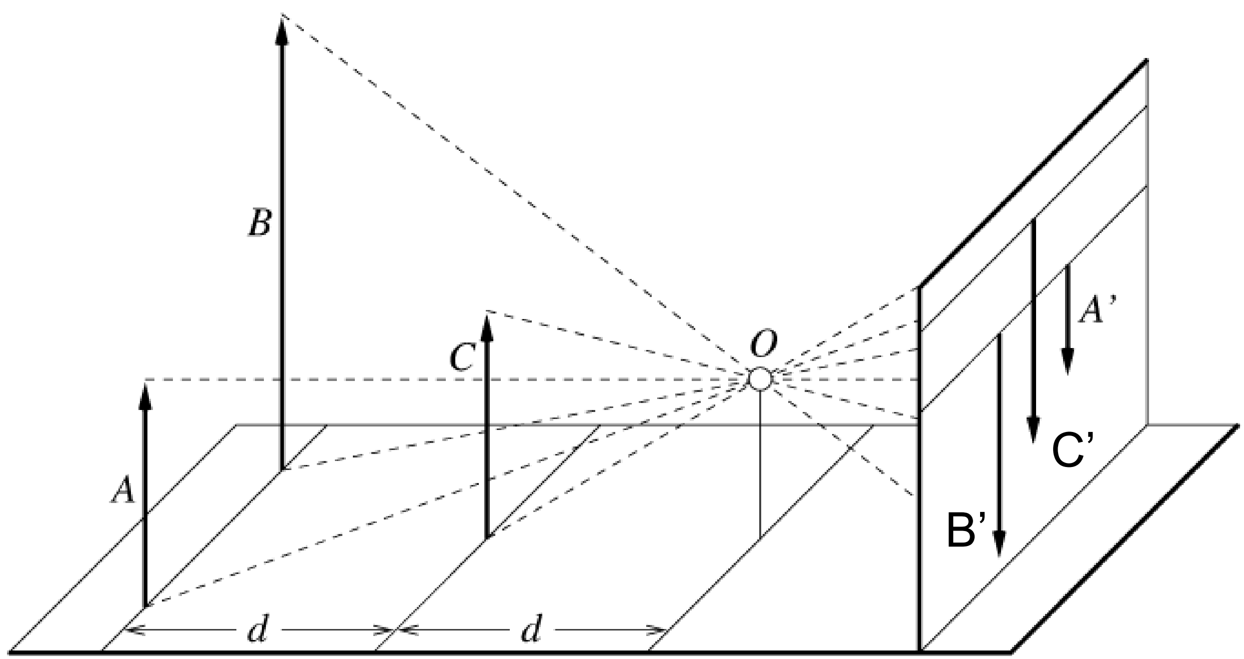
# Projective geometry

- 2D point in projective = (x,y,w) coordinates
  - w defines the scale of the projected image.
  - Each x,y point becomes a ray!



# Projective geometry

- In 3D, point (x,y,z) becomes (x,y,z,w)
- Perspective is w varying with z:
  - Objects far away are appear smaller



# Homogeneous coordinates

Converting *to* homogeneous coordinates

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

2D (image) coordinates

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

3D (scene) coordinates

Converting *from* homogeneous coordinates

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

2D (image) coordinates

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

3D (scene) coordinates

# Homogeneous coordinates

## Scale invariance in projection space

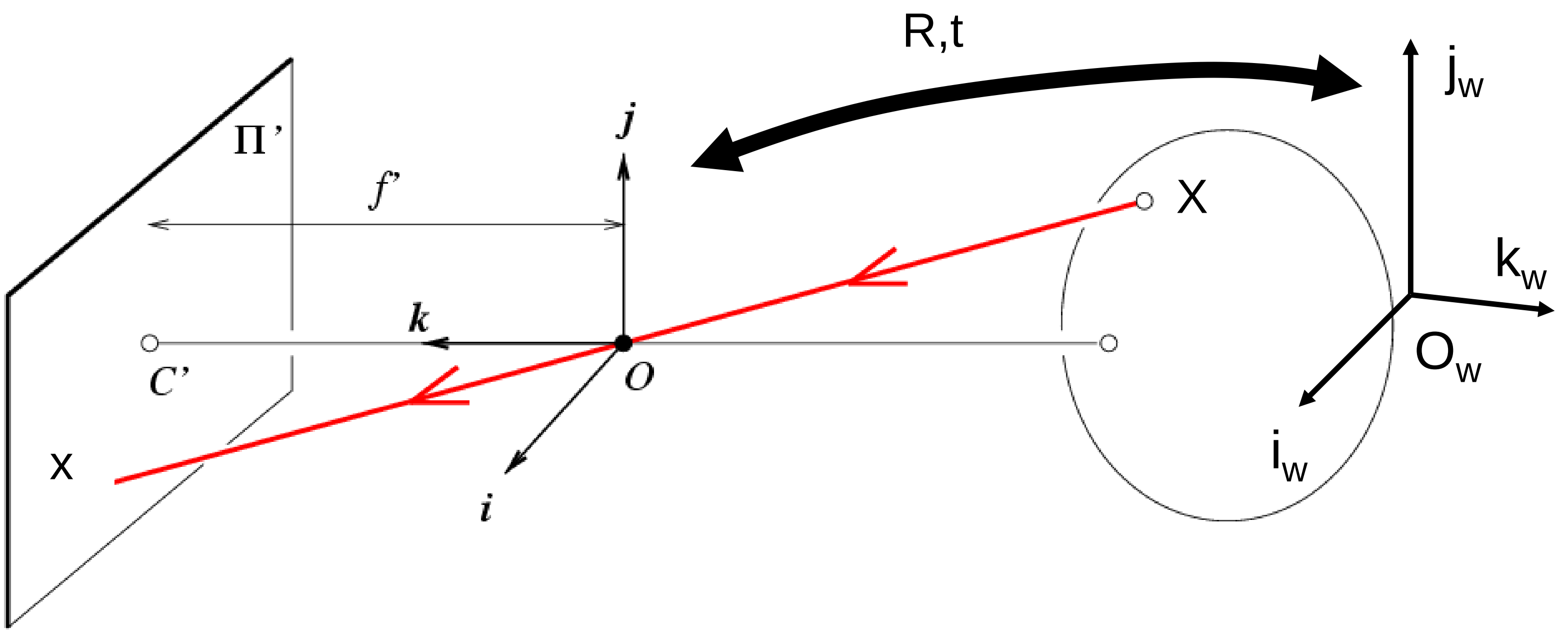
$$k\begin{bmatrix} x \\ y \\ w \end{bmatrix} = \begin{bmatrix} kx \\ ky \\ kw \end{bmatrix} \Rightarrow \begin{bmatrix} \frac{kx}{kw} \\ \frac{ky}{kw} \end{bmatrix} = \begin{bmatrix} \frac{x}{w} \\ \frac{y}{w} \end{bmatrix}$$

Homogeneous Coordinates                Cartesian Coordinates

E.G., we can uniformly scale the projective space, and it will still produce the same image -> *scale ambiguity*

Slide Credit: Savarese

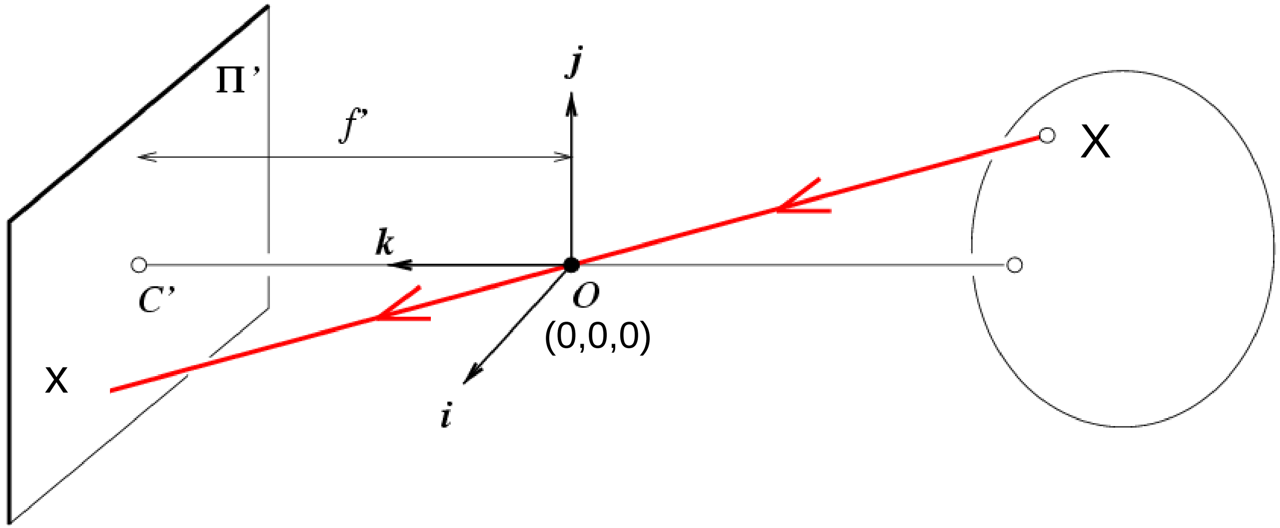
# Camera (projection) matrix




$$\mathbf{x} = \mathbf{K}\underbrace{\begin{bmatrix}\mathbf{R} & \mathbf{t}\end{bmatrix}}_{\text{Extrinsic Matrix}}\mathbf{X}$$

**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**

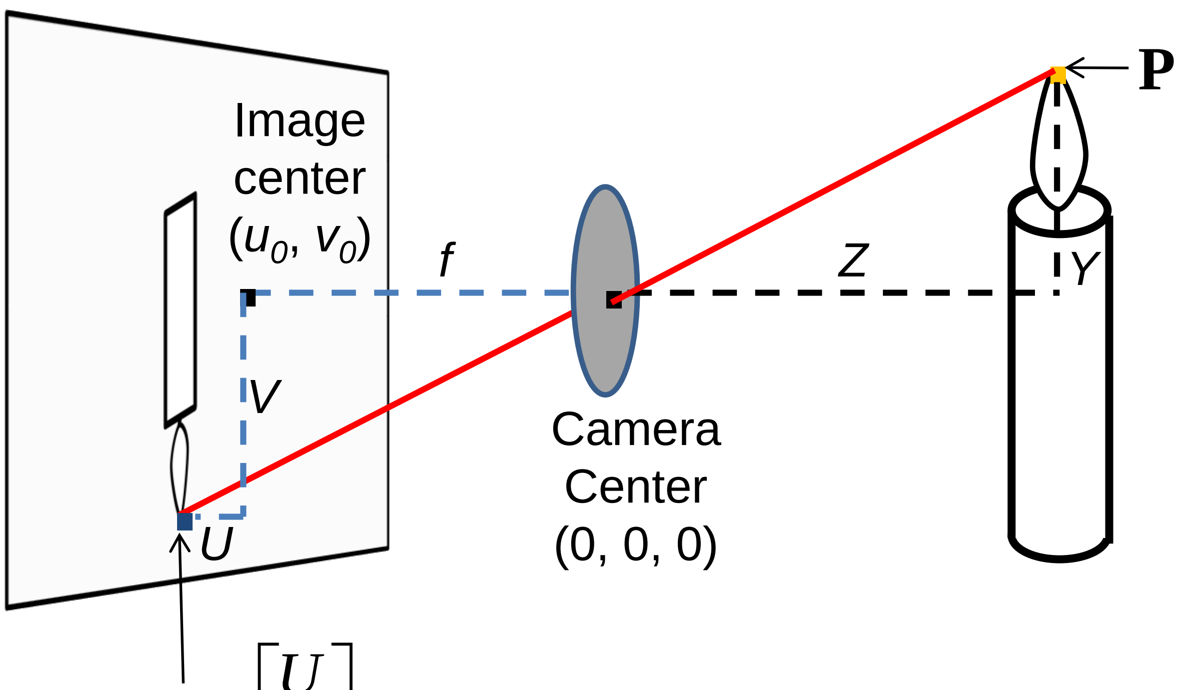
- No rotation
- Camera at (0,0,0)

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

**K**

Slide Credit: Savarese

# Projection: world coordinates→image coordinates




$$\mathbf{p} = \begin{bmatrix} U \\ V \end{bmatrix}$$

**p** = distance from image center

$$U = -X * \frac{f}{Z} \qquad V = -Y * \frac{f}{Z}$$

# Remove assumption: known optical center

Intrinsic Assumptions

- Unit aspect ratio
- No skew

Extrinsic Assumptions

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

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

James Hays

# Remove assumption: equal aspect ratio

Intrinsic Assumptions
- No skew

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

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

# Remove assumption: non-skewed pixels

Intrinsic Assumptions

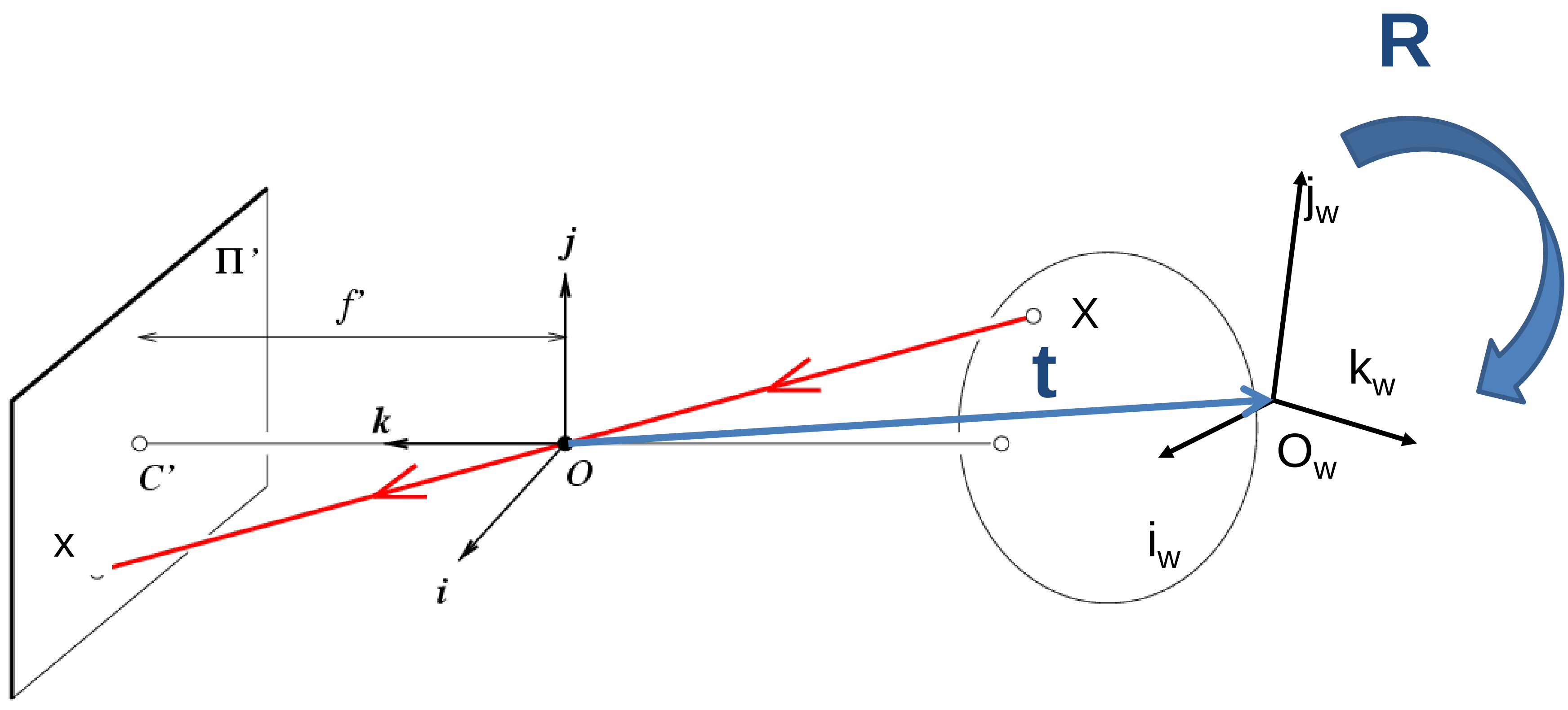
Extrinsic Assumptions
- No rotation
- Camera at (0,0,0)

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

Note: different books use different notation for parameters

James Hays

# Oriented and Translated Camera




James Hays

# Allow camera translation

Intrinsic Assumptions

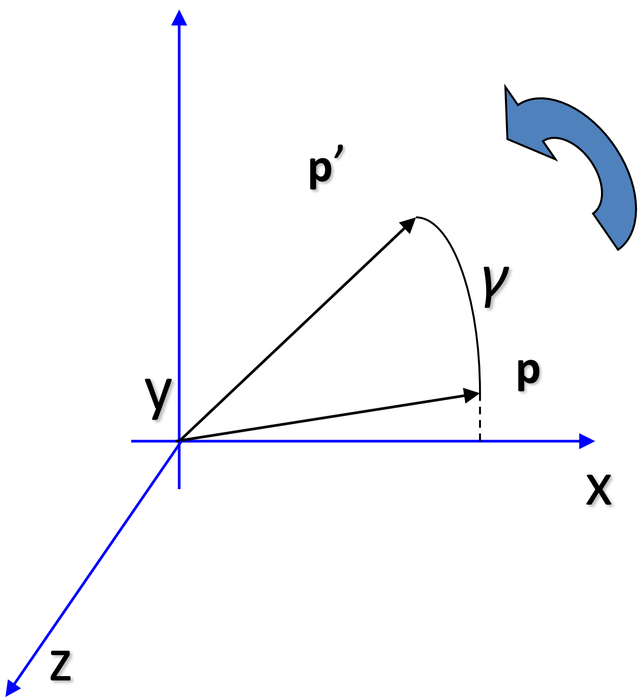
Extrinsic Assumptions
- No rotation

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

James Hays

# 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

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

$$\Downarrow$$

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

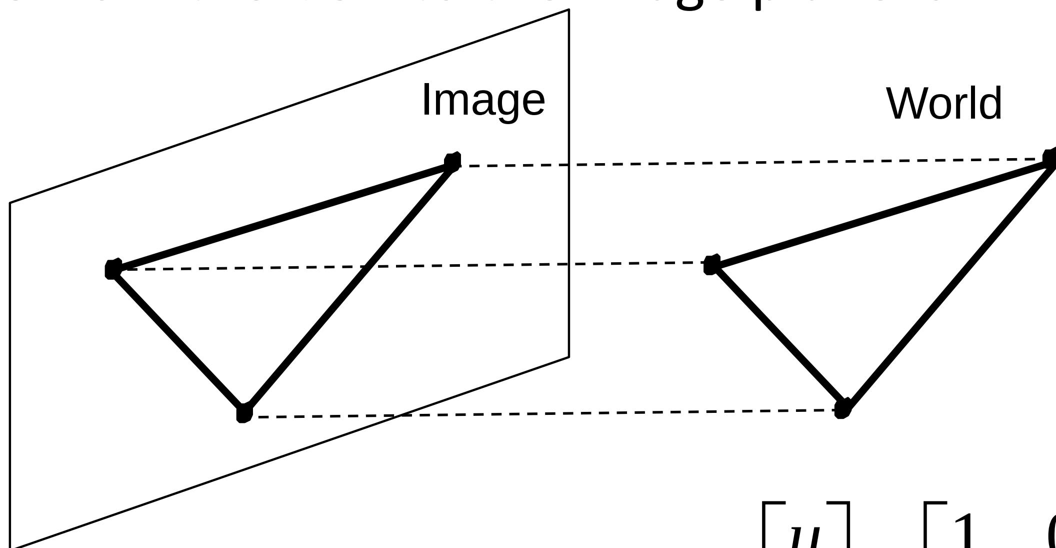
# Demo – Kyle Simek

- "Dissecting the Camera Matrix"
- Three-part blog series
- http://ksimek.github.io/2012/08/14/decompose/
- http://ksimek.github.io/2012/08/22/extrinsic/
- http://ksimek.github.io/2013/08/13/intrinsic/

- "Perspective toy"
- http://ksimek.github.io/perspective_camera_toy.html

# Orthographic Projection

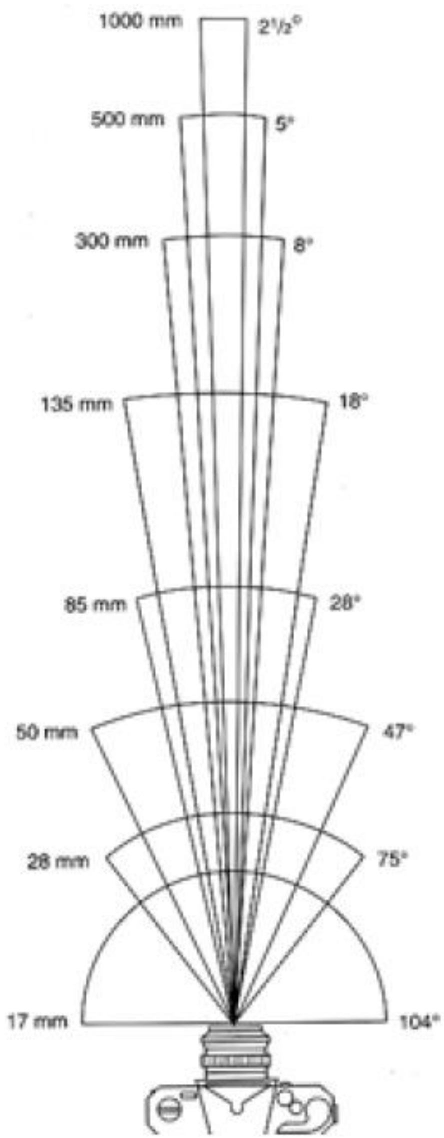
- Special case of perspective projection
  - Distance from the COP to the image plane is infinite

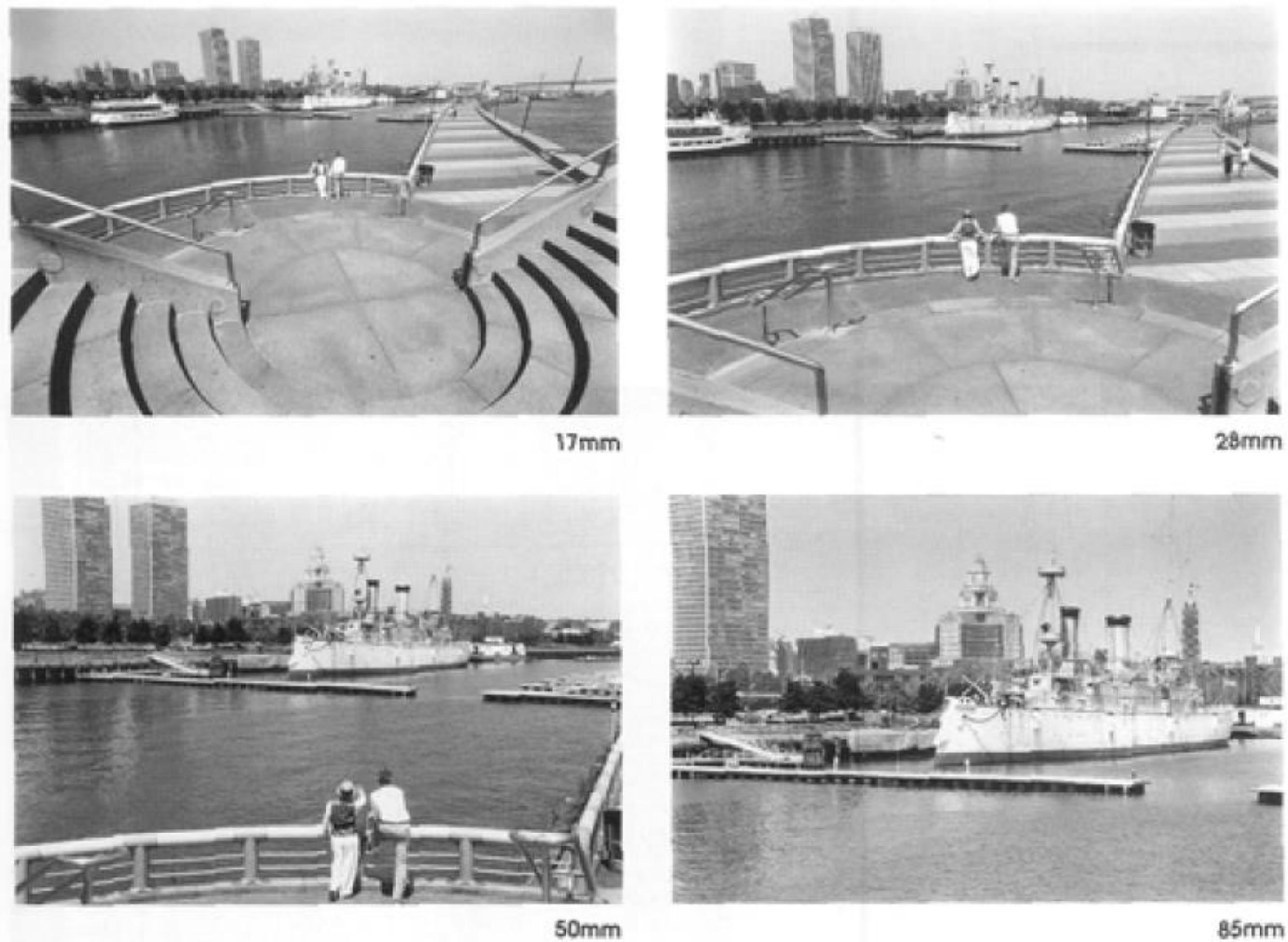



  - Also called "parallel projection"
  - What's the projection matrix?

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

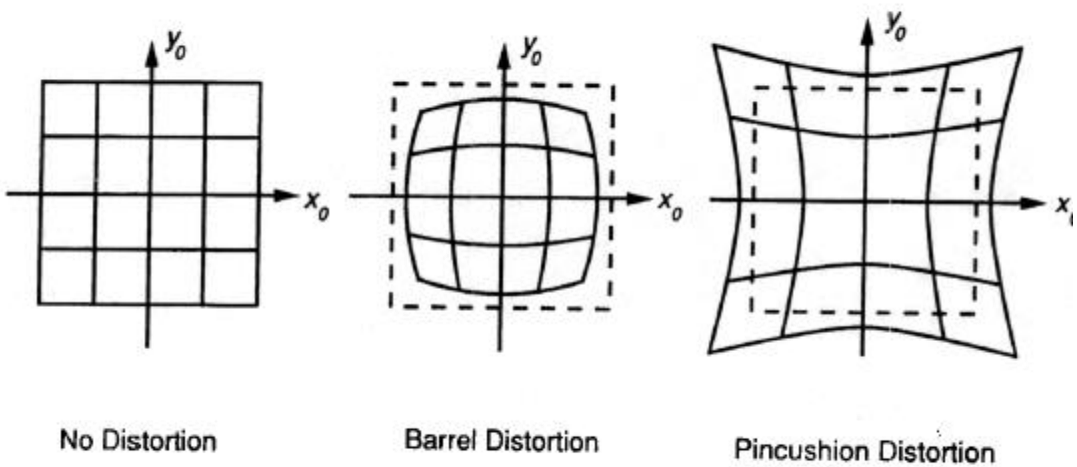
# Field of View (Zoom, focal length)




**From London and Upton**

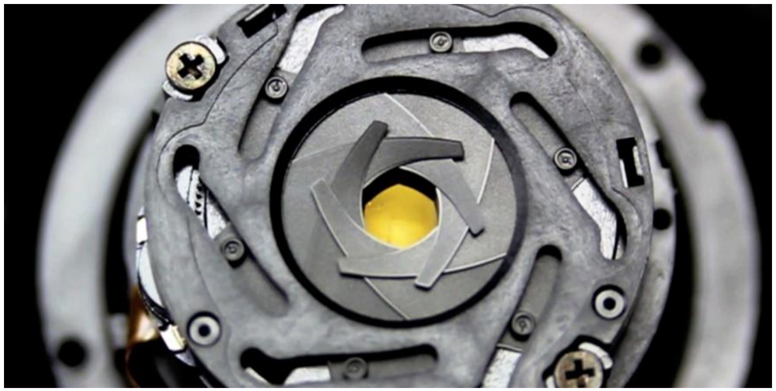
# Beyond Pinholes: Radial Distortion

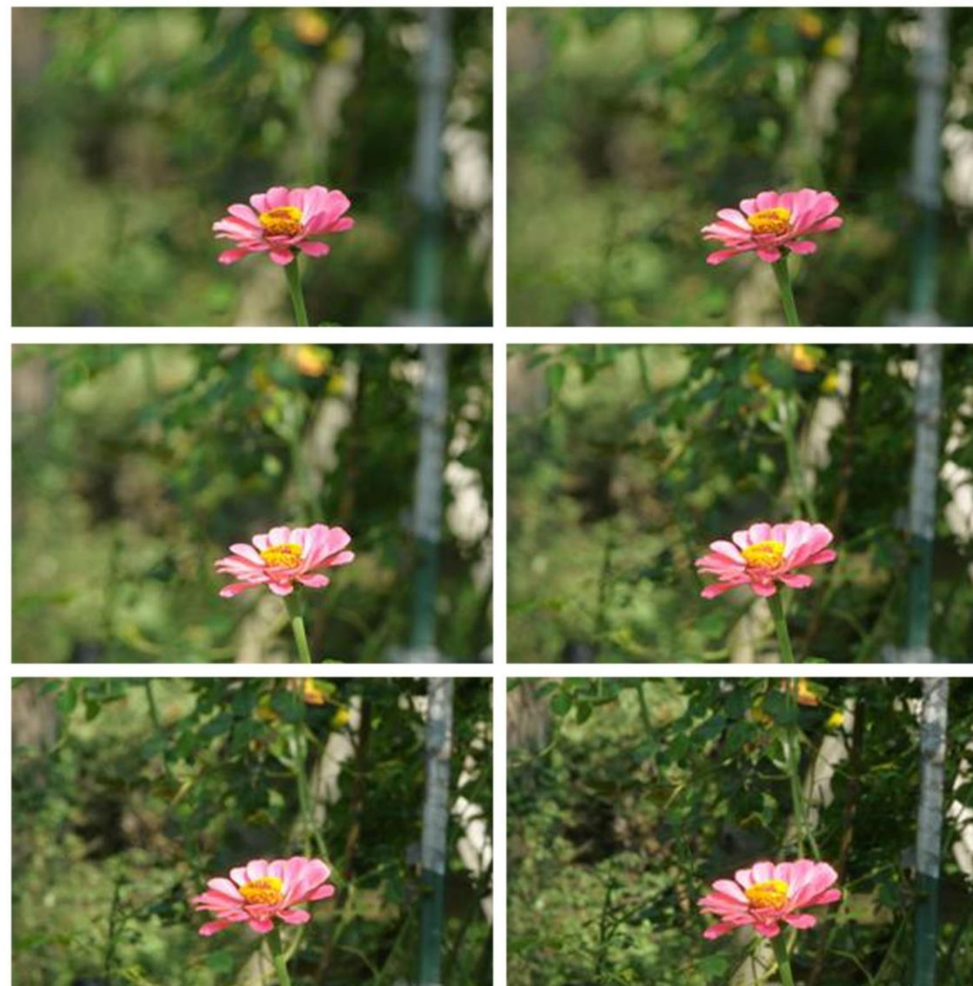



Corrected Barrel Distortion

Image from Martin Habbecke

# Beyond Pinholes: Real apertures

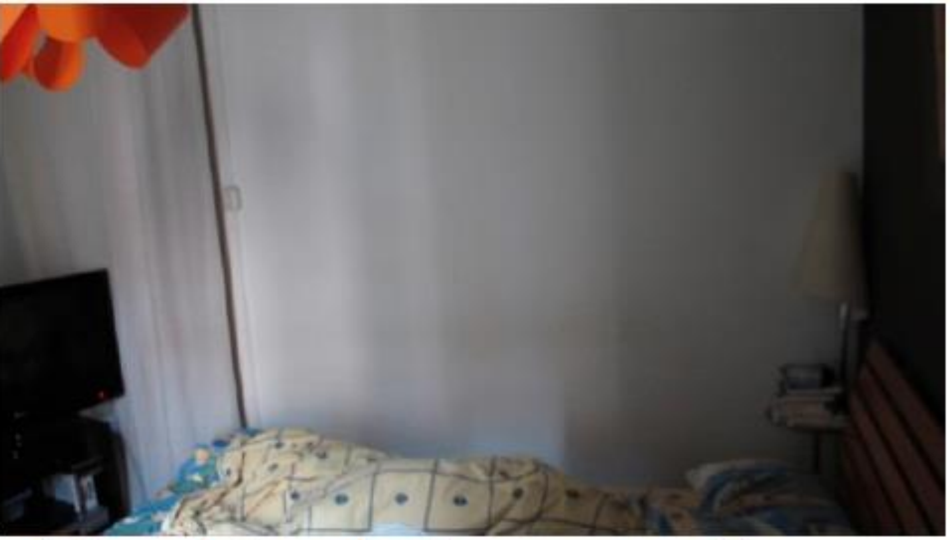




# Accidental Cameras

Accidental Pinhole and Pinspeck Cameras
Revealing the scene outside the picture.
Antonio Torralba, William T. Freeman

# Accidental Cameras



a) Input (occluder present)

b) Reference (occluder absent)

c) Difference image (b-a)

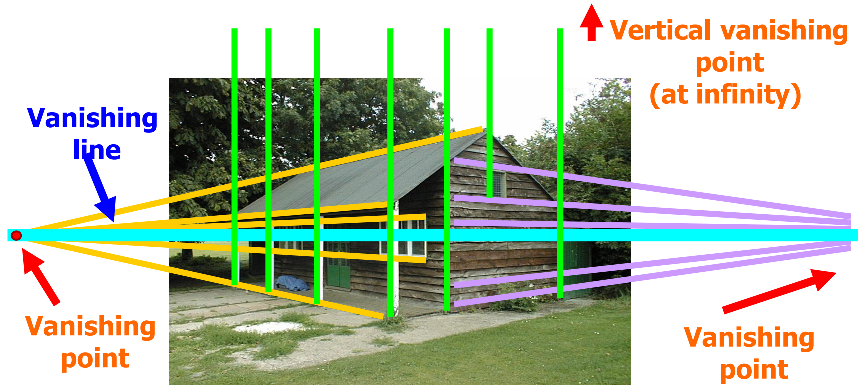
d) Crop upside down

e) True view

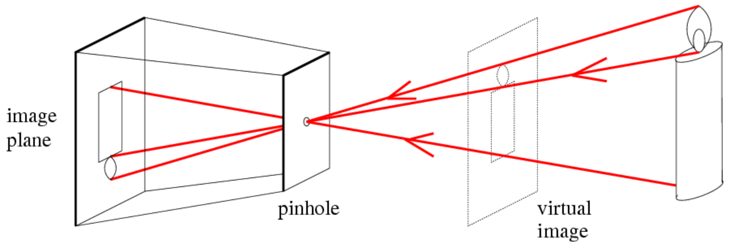
James Hays

# 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

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

James Hays

# IS THIS ENOUGH?

Erik Johansson – The Architect