MECS 6616

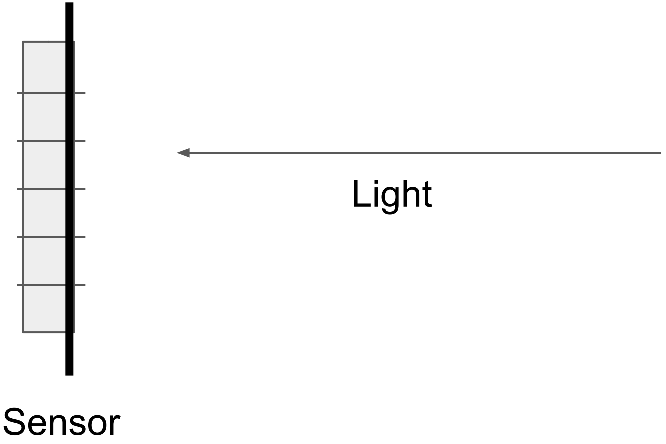
Computer Vision I: Why is Computer Vision Hard?

Spring 2020

Matei Ciocarlie

How do cameras work?

- A pixel measures the amount of light that hits it over a given time period
    - … can almost think of it as “counting photons”...



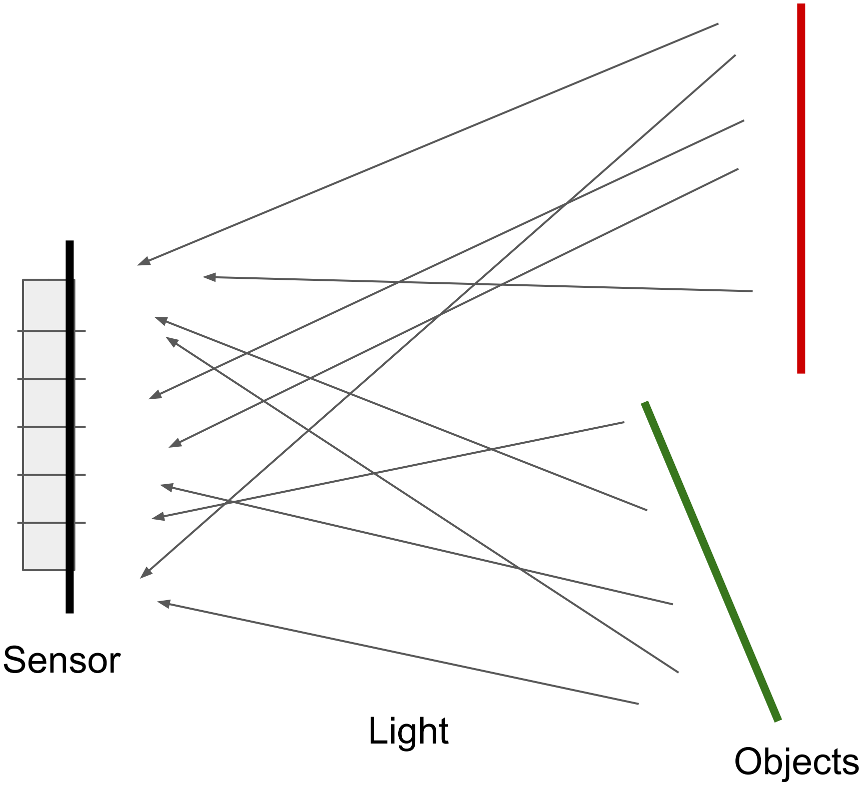
How do cameras work?

- A pixel measures the amount of light that hits it over a given time period
  - … can almost think of it as “counting photons”...
- Then, the result is digitized: **digital images are arrays of numbers**

128	144	120	130	128	165	184
131	152	128	114	180	200	230
122	101	111	190	205	240	255
100	40	42	20	34	50	98
22	30	25	22	72	90	108

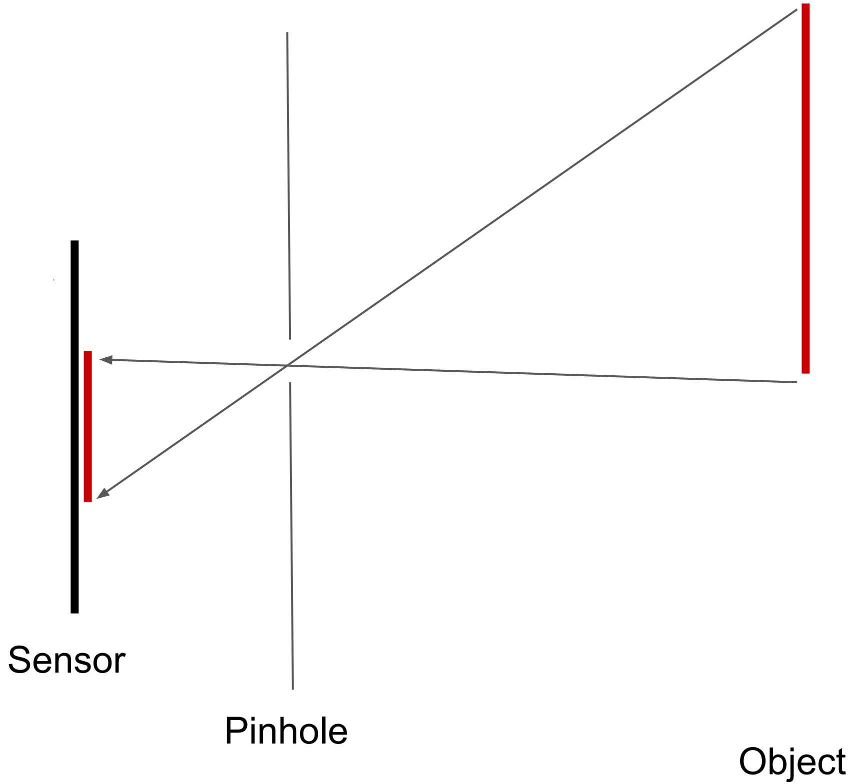
How do cameras work?

Light transport

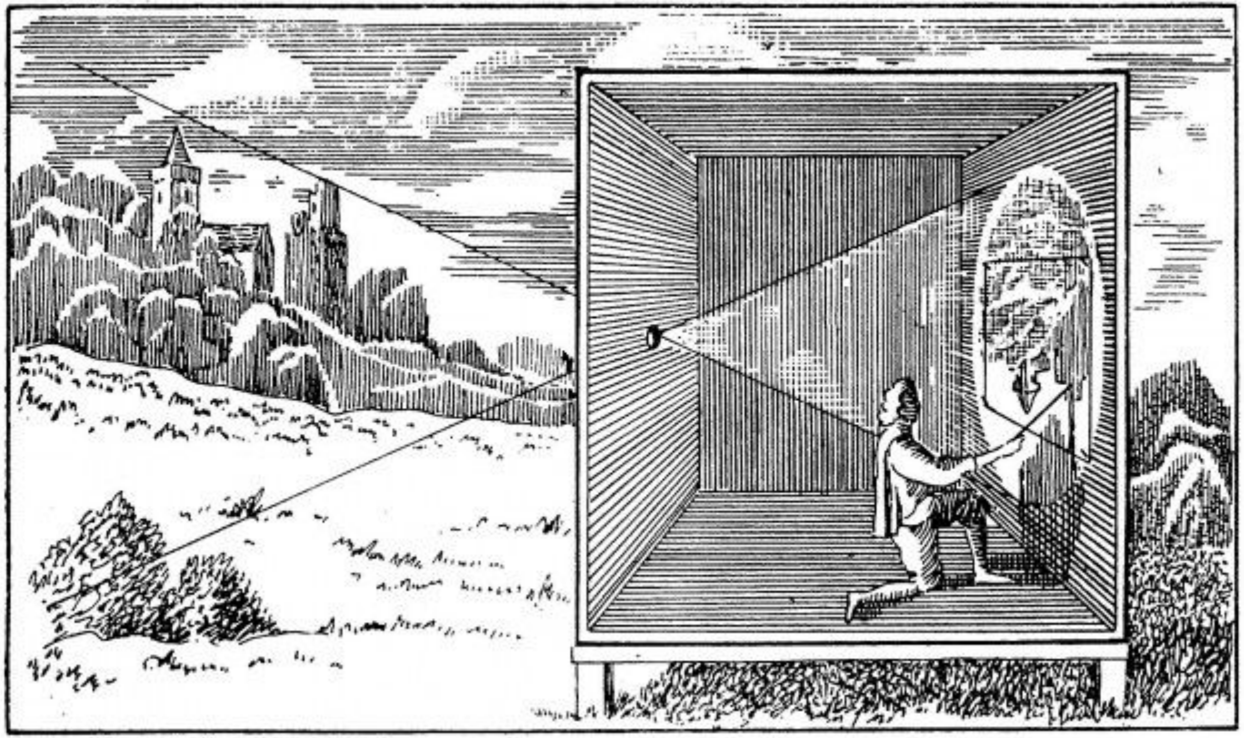


How do cameras work?

Pinhole camera



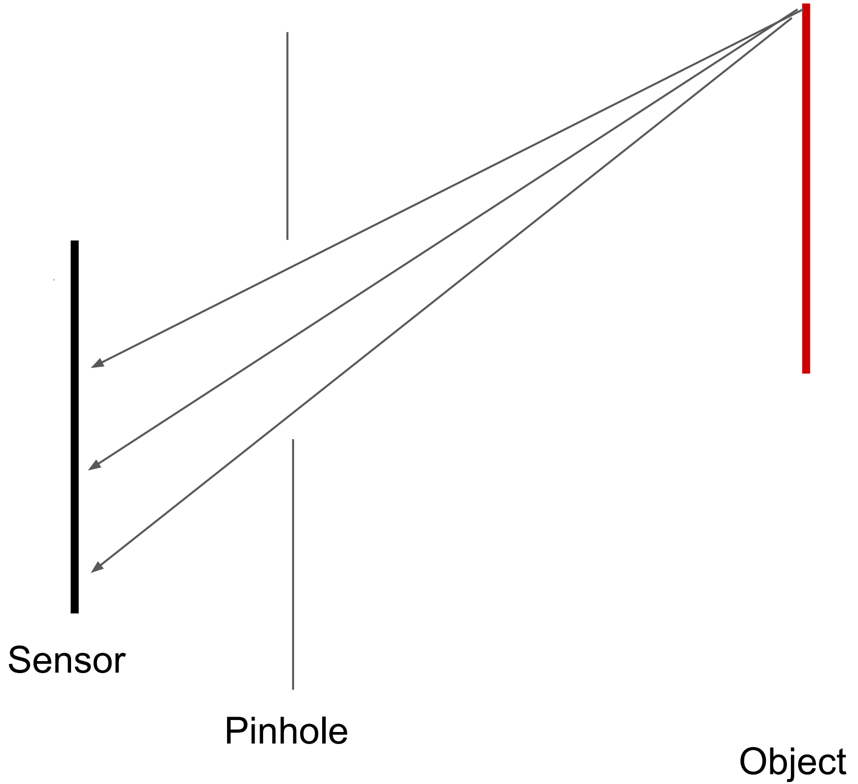
Camera obscura



http://pixsylated.com/blog/pix/Camera-Obscura-diagram-630x375.jpeg

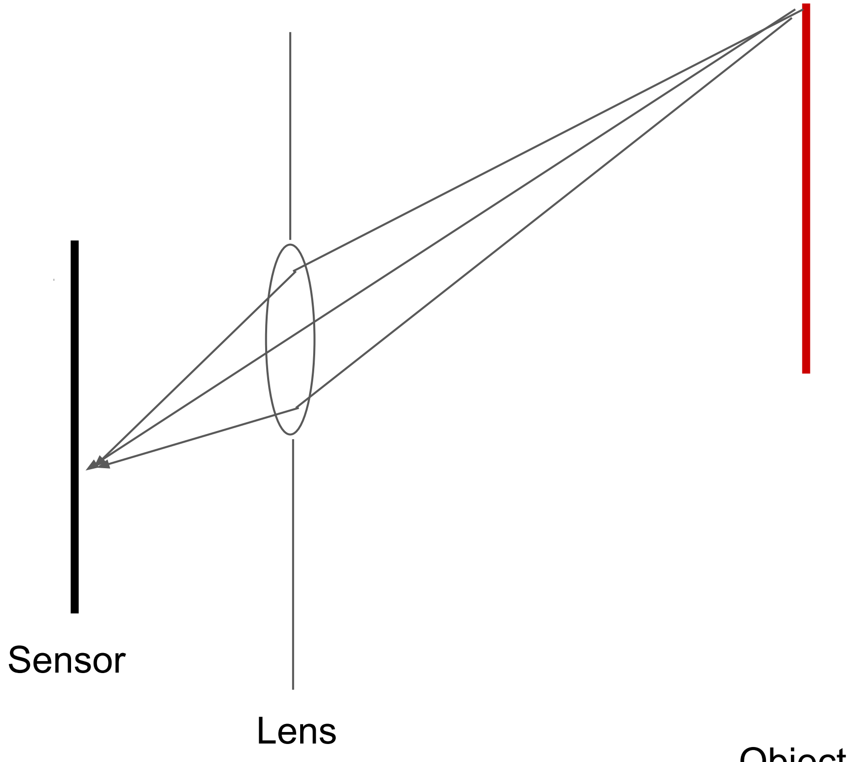
How do cameras work?

Pinhole camera



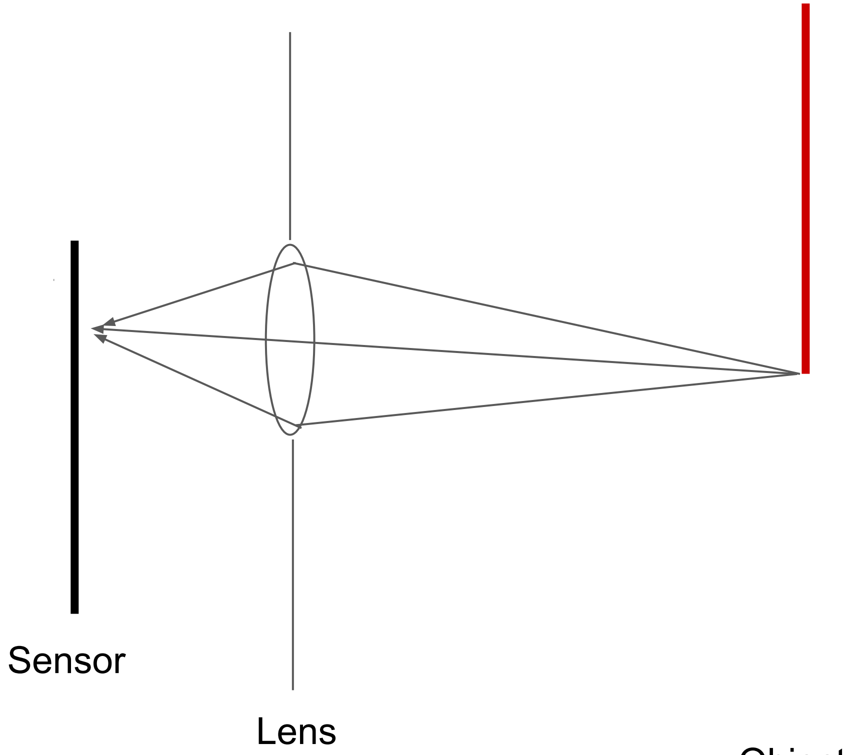
How do cameras work?

Lens camera



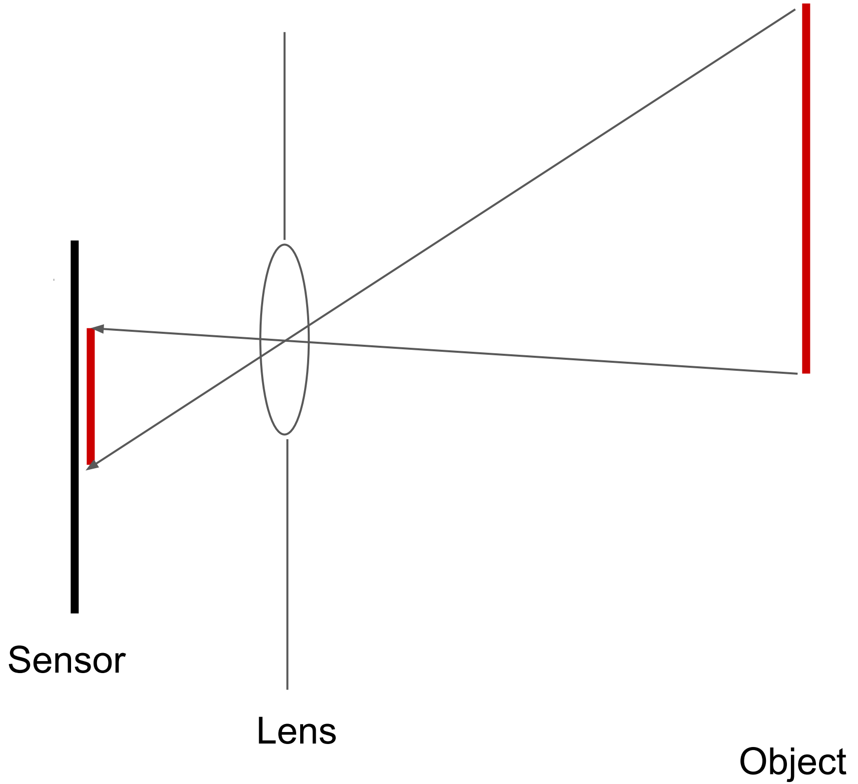
How do cameras work?

Lens camera



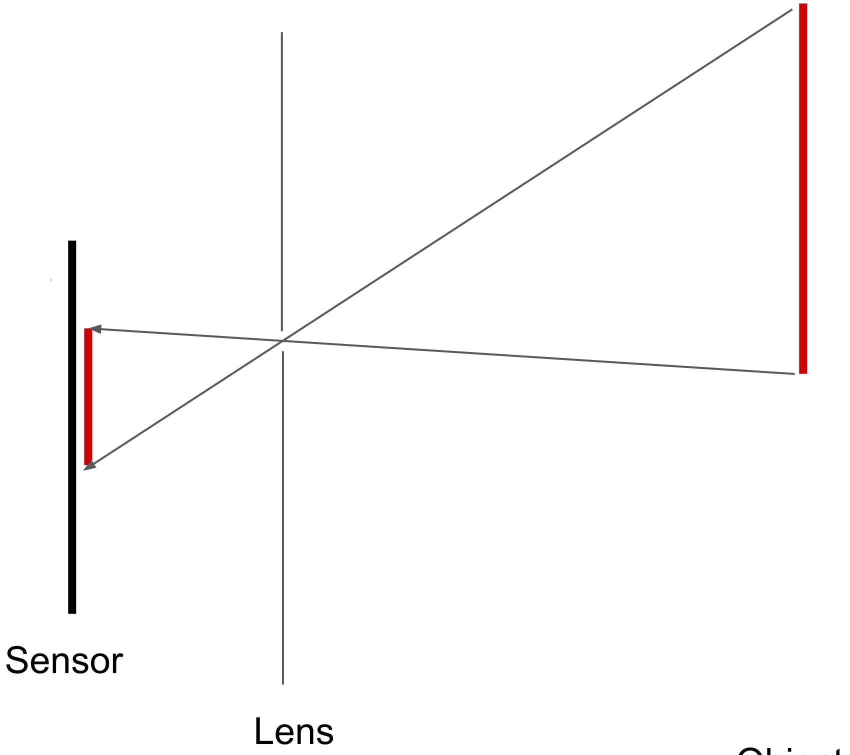
How do cameras work?

Lens camera

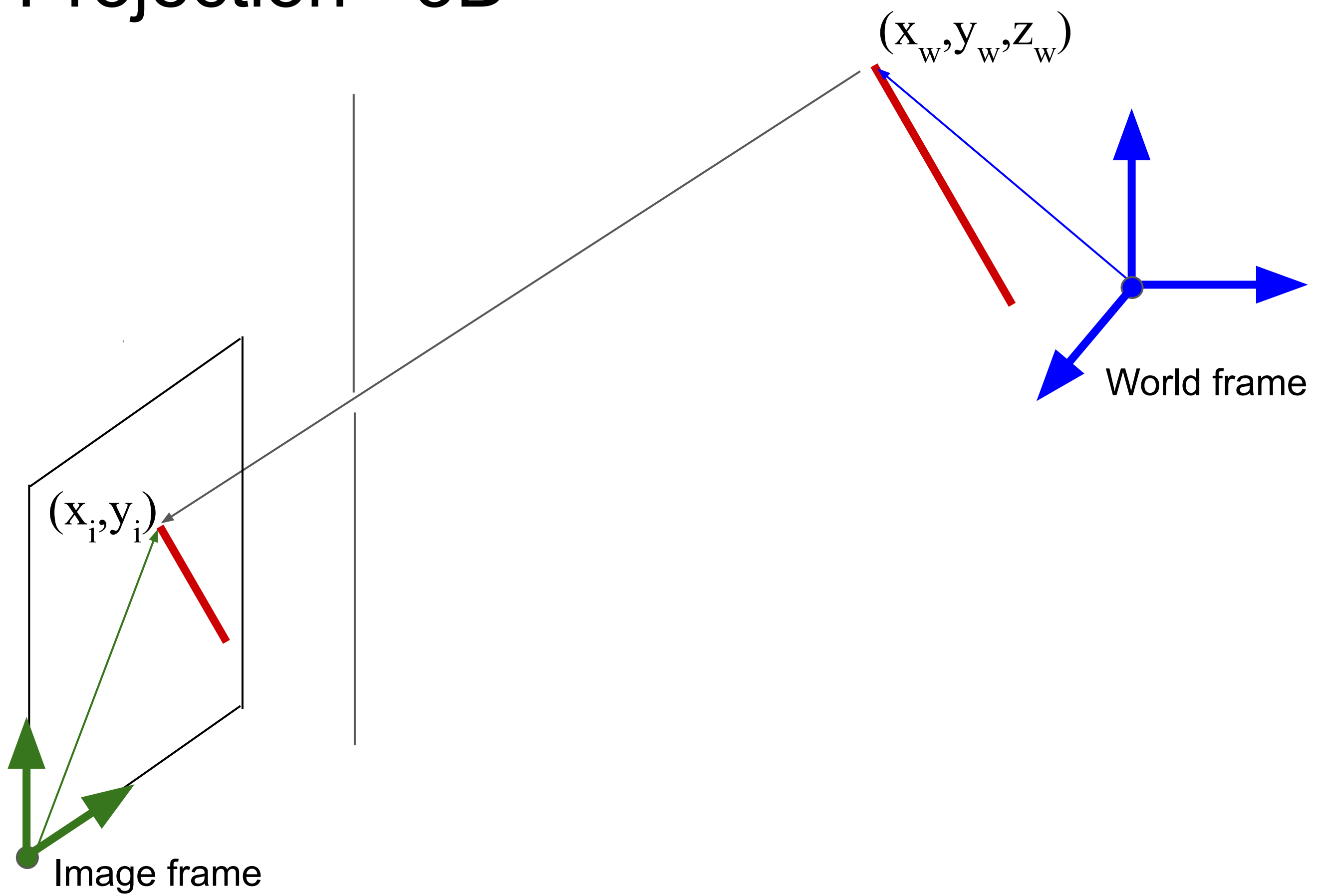


Lens cameras

- Lenses gather light and focus it on image plane
- Distance from lens to image plane is critical for sharp images
- However, for simplicity, in geometrical reasoning one can often assume simpler pinhole model
- Introduces **Perspective Projection**



Perspective Projection - 3D



Perspective Projection

A few properties:

- lines always project to lines
- parallel lines appear to be intersecting (unless parallel to image plane)
    - all the lines in a parallel set appear to meet at the same **vanishing point**
- objects further away appear smaller
- angles are generally **not** preserved

Camera Matrix

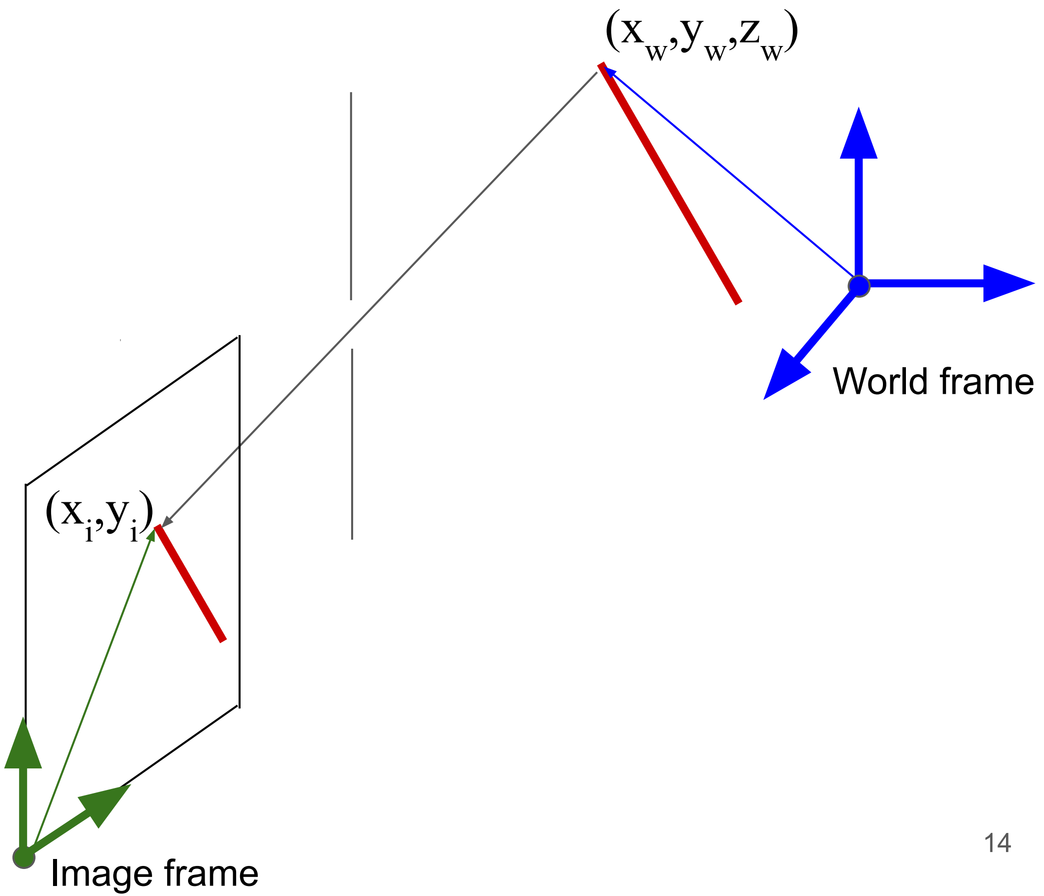
World point $\mathbf{x}_\mathrm{w} = [x_\mathrm{w}, y_\mathrm{w}, z_\mathrm{w}, 1]^\mathrm{T}$

Image point $\mathbf{x}_\mathrm{c} = [x_\mathrm{i}, y_\mathrm{i}, 1]^\mathrm{T}$

Camera matrix $\mathbf{P} \subset \Re^{3\mathrm{x}4}$:

$$\mathbf{x}_\mathrm{c} = \mathbf{P}\ \mathbf{x}_\mathrm{w}$$

… for any point in scene.



Camera Matrix

- Can be computed if enough correspondences are known:

$$\mathbf{x}_c^{\,j} = \mathbf{P}\,\mathbf{x}_w^{\,j}$$

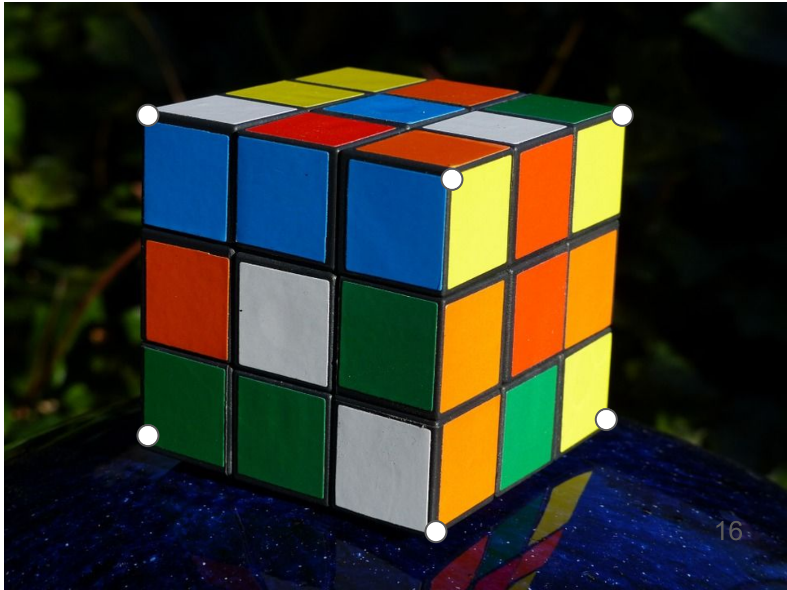
- In theory, at least 6 points are needed
- In practice, use more to reduce error

Camera Matrix

- Can be computed if enough correspondences are known:

$$\mathbf{x}_c^{\,j} = \mathbf{P}\, \mathbf{x}_w^{\,j}$$

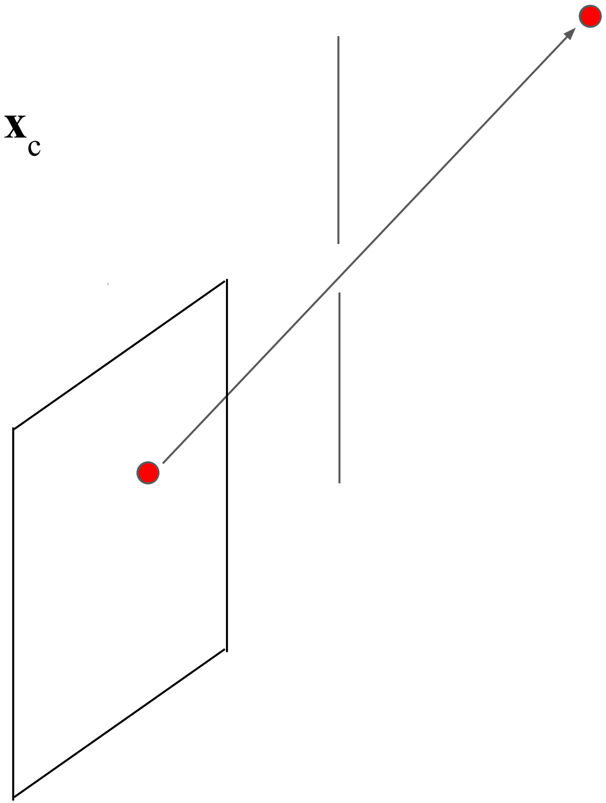
- In theory, at least 6 points are needed
- In practice, use more to reduce error



Calibration problem: compute the matrix $\mathbf{P}$ for a given camera.

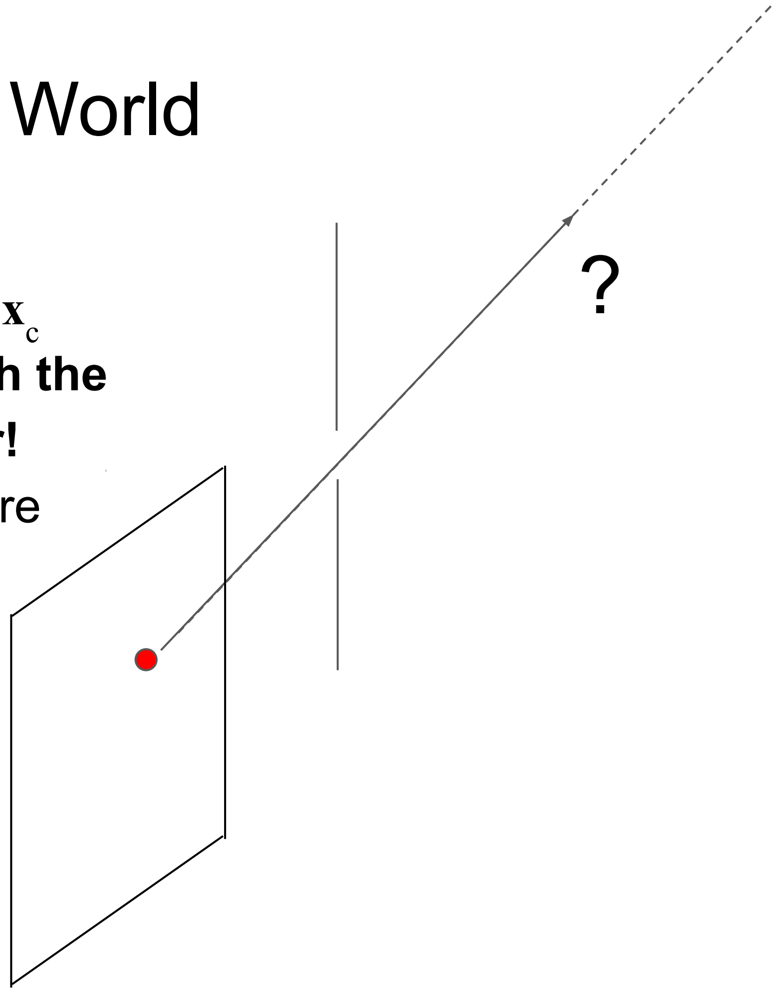
Projecting Back into the World

- Into the image: $\mathbf{x}_c = \mathbf{P}\,\mathbf{x}_w$
- Into the world: compute $\mathbf{x}_w$ given $\mathbf{x}_c$



Projecting Back into the World

- Into the image: $\mathbf{x}_c = \mathbf{P}\,\mathbf{x}_w$
- Into the world: compute $\mathbf{x}_w$ given $\mathbf{x}_c$
  - **can not be fully solved with the information we have so far!**
  - we just know $\mathbf{x}_w$ is somewhere along **a ray**



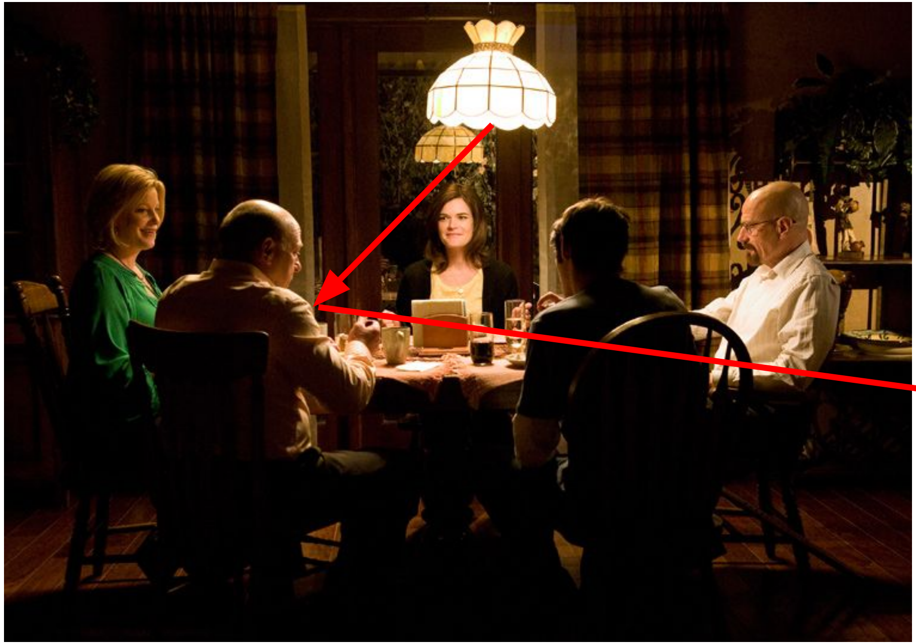
Intensity and Color

Recall once more: images are arrays of numbers!

128	144	120	130	128	165	184
131	152	128	114	180	200	230
122	101	111	190	205	240	255
100	40	42	20	34	50	98
22	30	25	22	72	90	108

Intensity and Color

Light leaves the **source**, bounces of the **object**, and is captured by the **sensor**



128	144	120	130	128	165	184
131	152	128	114	180	200	230
122	101	111	190	205	240	255
100	40	42	20	34	50	98
22	30	25	22	72	90	108

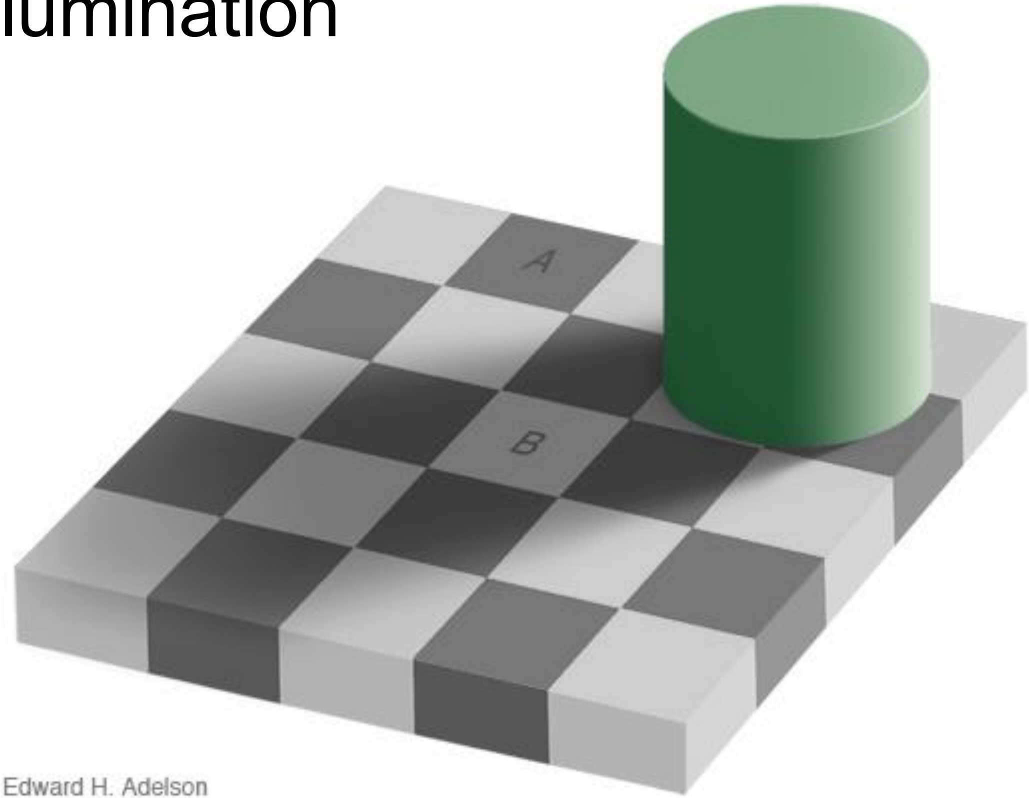
Intensity and Color

Light leaves the **source**, bounces of the **object**, and is captured by the **sensor**

- what the sensor sees depends on the characteristics of both the light source and the object that is captured (without accounting for multiple bounces…)
- main light source characteristic: spectral distribution
- main object characteristics:
    - geometry
    - albedo (color)
    - “shiny” vs. “dull”

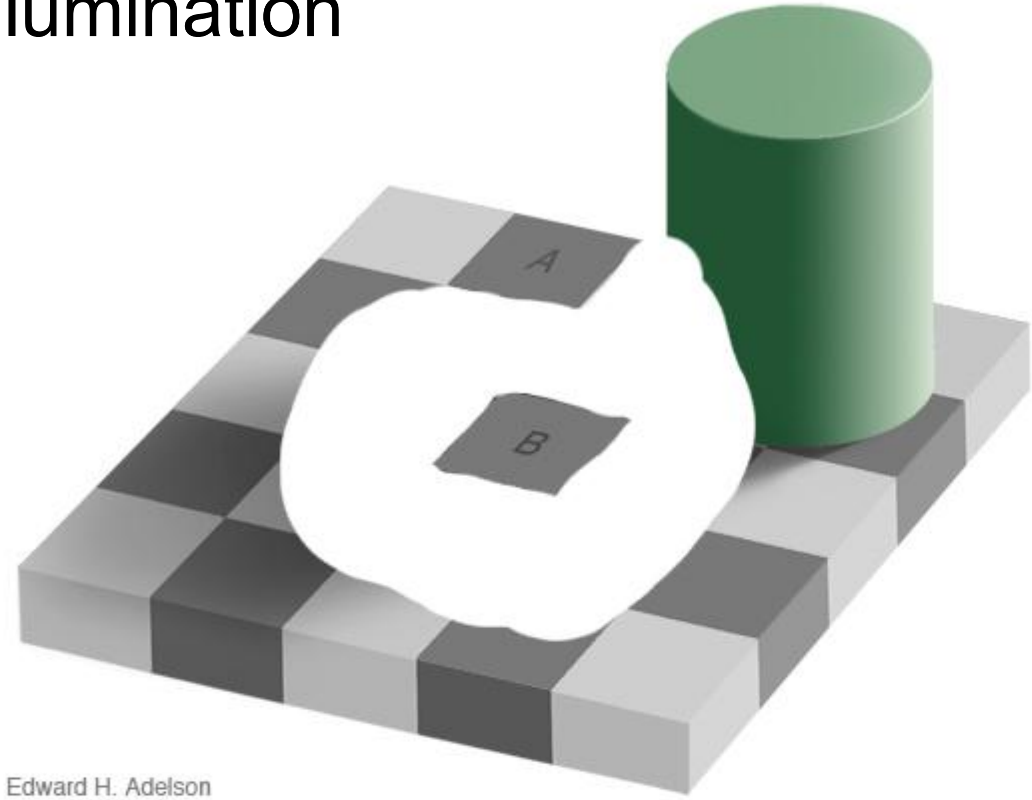
In general, it is **impossible to compute all of these factors** based only on the recording from the sensor, without additional information about the scene!

Color vs. Illumination

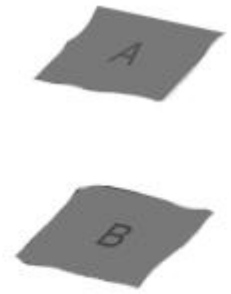


Edward H. Adelson

Color vs. Illumination



Color vs. Illumination



Recap

- "Color" (number) of a pixel depends on:
    - light source: spectral distribution
    - object(s) light bounces of:
        - geometry
        - shininess
        - color
    - camera parameters:
        - position relative to scene
        - projection matrix
        - focus
- Generally impossible to disambiguate!

128	144	120	130	128	165	184
131	152	128	114	180	200	230
122	101	111	190	205	240	255
100	40	42	20	34	50	98
22	30	25	22	72	90	108

Object Detection / Segmentation

- Does this image contain a coffee mug?
- Draw a bounding box (as tight as possible) around the mug in the image
- Which pixels in the image belong to mug?

Object Detection / Segmentation

- Does this image contain a coffee mug?
- Draw a bounding box (as tight as possible) around the mug in the image
- Which pixels in the image belong to mug?

What type of learning problems are these?

Object Detection / Segmentation

- Does this image contain a coffee mug?
- Draw a bounding box (as tight as possible) around the mug in the image
- Which pixels in the image belong to mug?

- Types of detection/segmentation problems:
  - Instance level (one specific mug)
  - Class level (any mug)

What can change for each of these?

Regression and Classification on Images

128	144	120	130	128	165	184
131	152	128	114	180	200	230
122	101	111	190	205	240	255
100	40	42	20	34	50	98
22	30	25	22	72	90	108

128
131
122
100
22
144
152
101
40
30
120
128
111
42

Regressor / Classifier

Image class

Bounding box location

Per pixel class

Regression and Classification on Images

1,000 x 1,000 Pixels

1M pixel / channel

3M pixel / image

$\mathbf{x}_i \subset \Re^{3,000,000}$

$N = ?$

Regressor / Classifier

Image class

Bounding box location

Per pixel class

Regression and Classification on Images

1,000 x 1,000 Pixels

1M pixel / channel

3M pixel / image

$\mathbf{x}_i \subset \Re^{3,000,000}$

$N = ?$

Regressor / Classifier

Image class

Bounding box location

Per pixel class