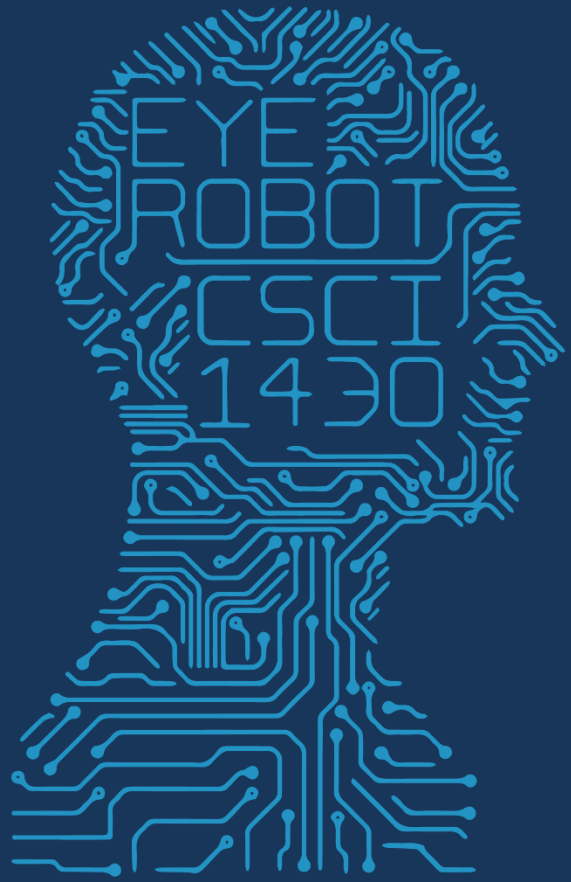
1950

Future Vision



2020

Computer Vision

Cats + mirrors + face filters

[reddit – juicysox]

[Isa Milefchik (1430 HTA Spring 2020)]

[Madelyn Adams (student Spring 2019)]

Zoom protocol

Please:

- Cameras on (it really helps me to see you)
- Real names
- Mics muted
- Raise hands in Zoom for questions, unmute when I call
- I will ask more often for questions

Project 4 – due Friday

- Both parts
  - Written
  - Code

Project 5

- Questions and code due
  Friday April 10th

Final group project

- Groups of four
- Groups of one are discouraged – you need a good reason.
- Group by timezone where possible; use Piazza
- We'll go over possible projects at a later date

Questions

- What else did I miss?

By Suren Manvelyan, http://www.surenmanvelyan.com/gallery/7116

By Suren Manvelyan, http://www.surenmanvelyan.com/gallery/7116

By Suren Manvelyan, http://www.surenmanvelyan.com/gallery/7116

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

Camera obscura: dark room

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

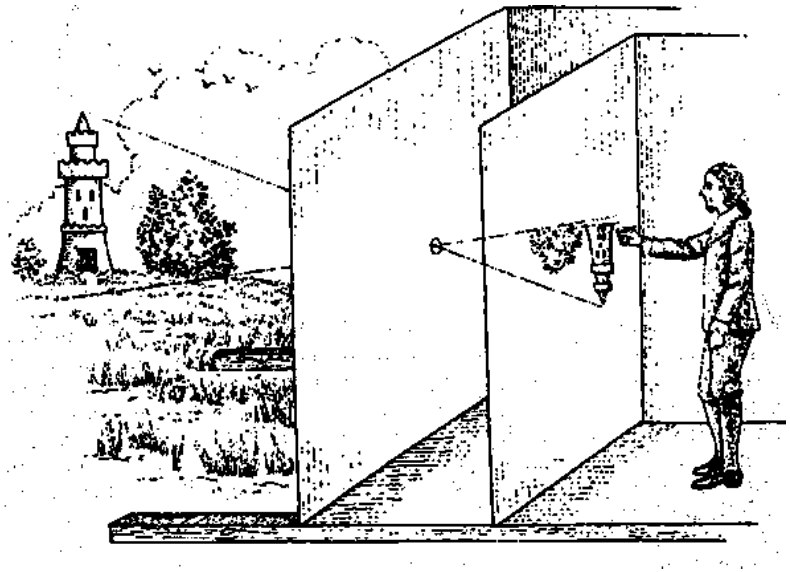
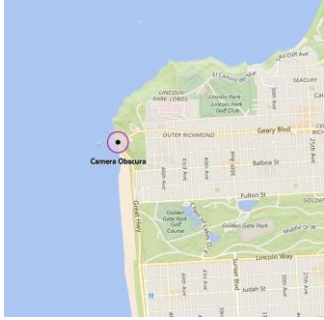


Illustration of Camera Obscura

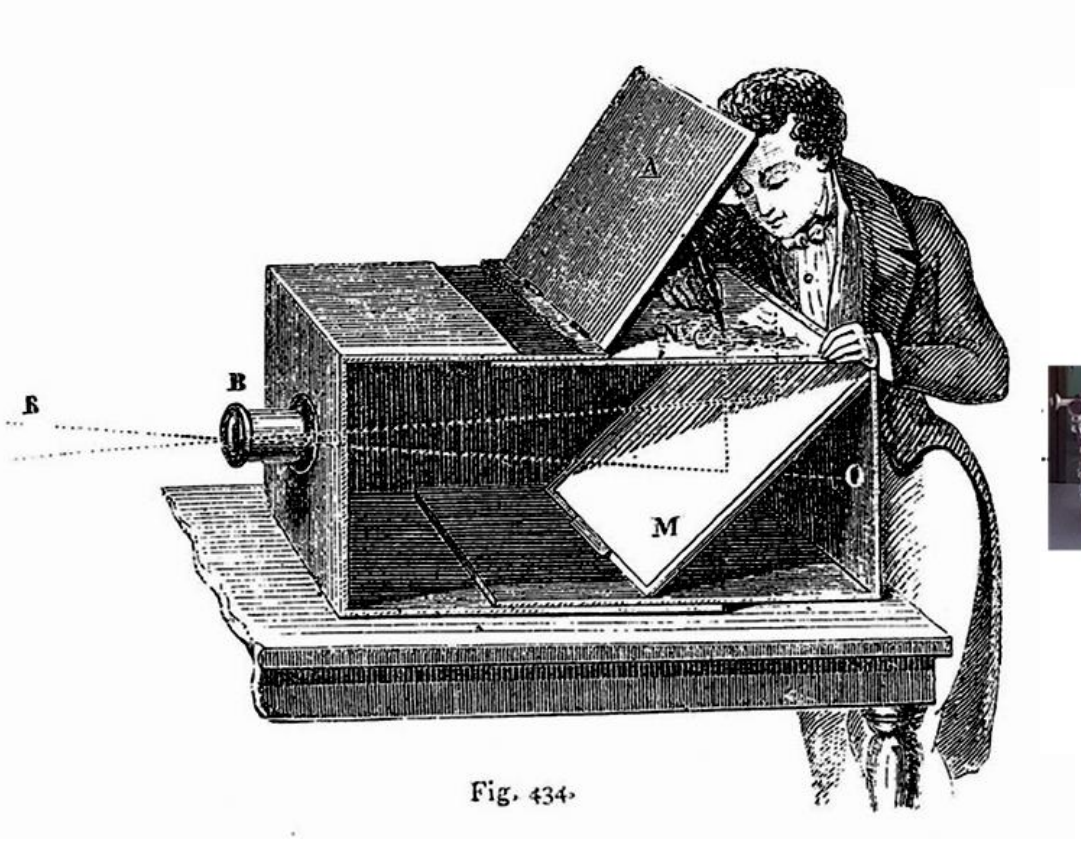
Freestanding camera obscura at UNC Chapel Hill

Photo by Seth Ilys

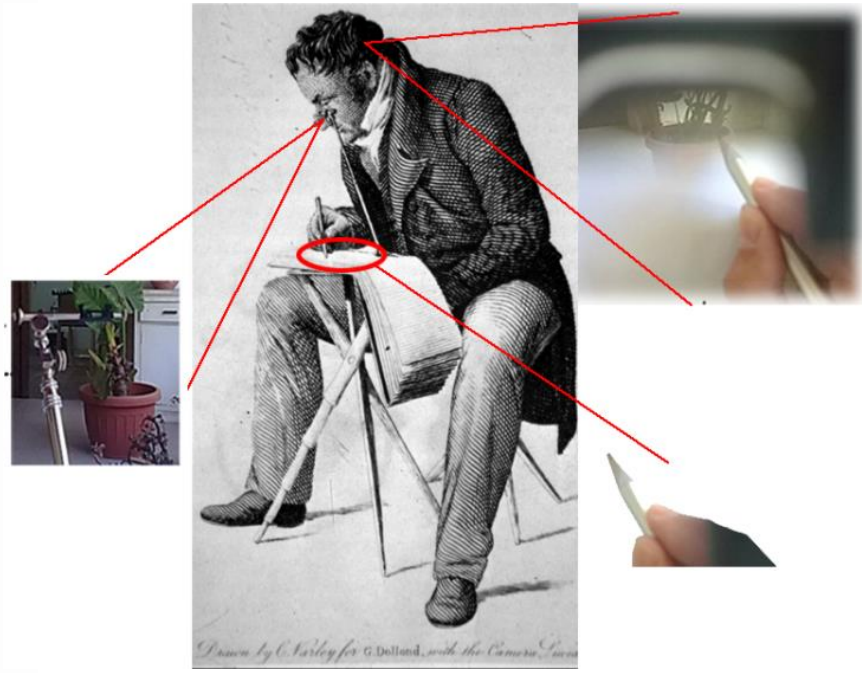
James, San Francisco, Aug. 2017



Camera obscura / lucida used for tracing



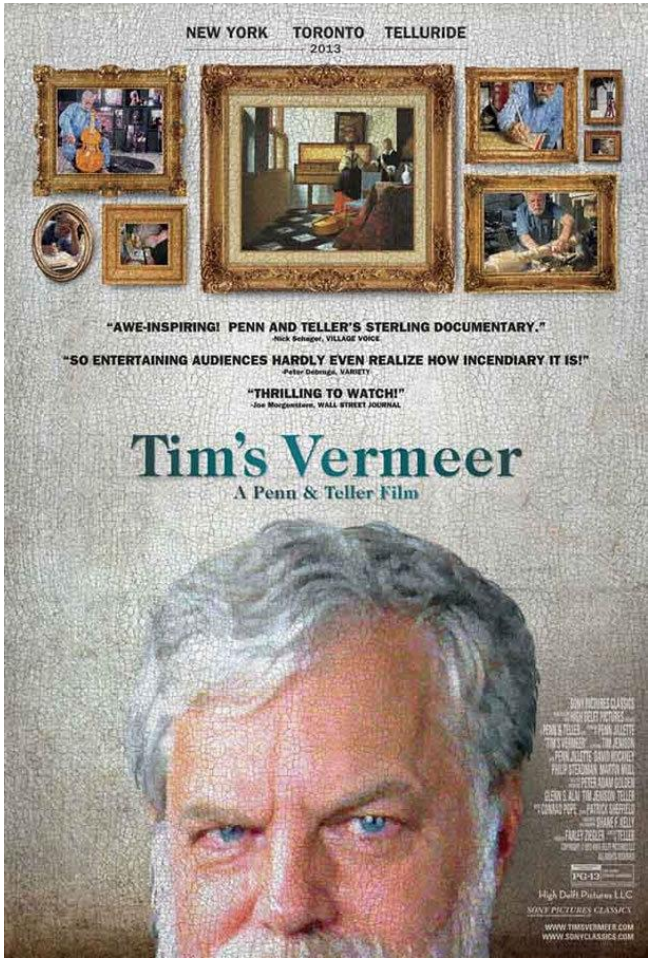
Lens Based Camera Obscura, 1568



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

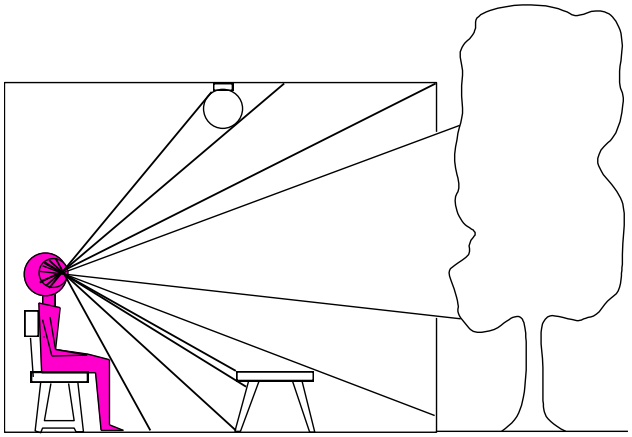
Stored at UT Austin

Niepce later teamed up with Daguerre, who eventually created Daguerrotypes

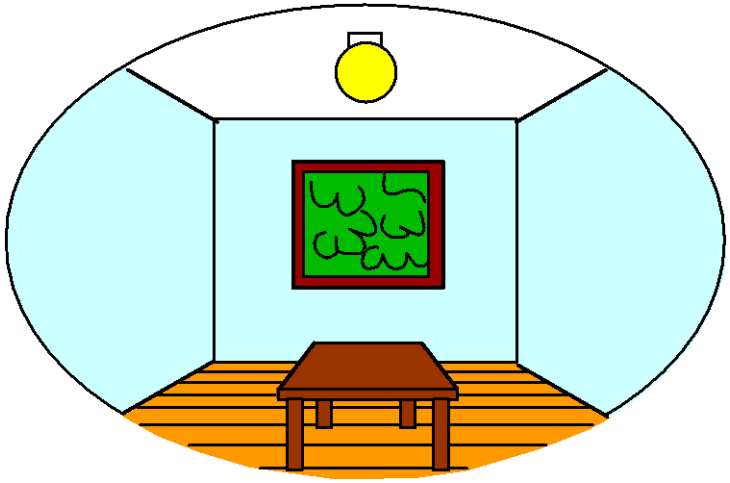
Dimensionality Reduction Machine (3D to 2D)

3D world

2D image



Point of observation

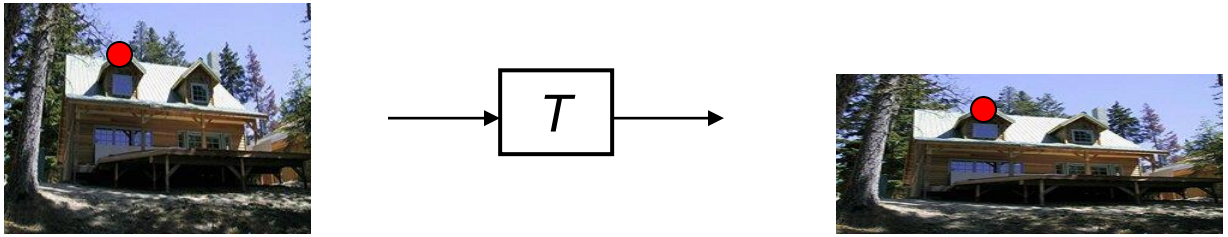


Holbein’s The Ambassadors - 1533

Holbein’s The Ambassadors – Memento Mori

2D IMAGE TRANSFORMS

Parametric (global) transformations



p = (x,y)

p' = (x',y')

Transformation T is a coordinate-changing machine:

$$p' = T(p)$$

What does it mean that *T* is global?

- *T* is the same for any point p

T can be described by just a few numbers (parameters)

For linear transformations, we can represent T as a matrix

$$p' = \mathbf{T}p$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \mathbf{T} \begin{bmatrix} x \\ y \end{bmatrix}$$

Common transformations

Original

Transformed

Translation

Rotation

Scaling

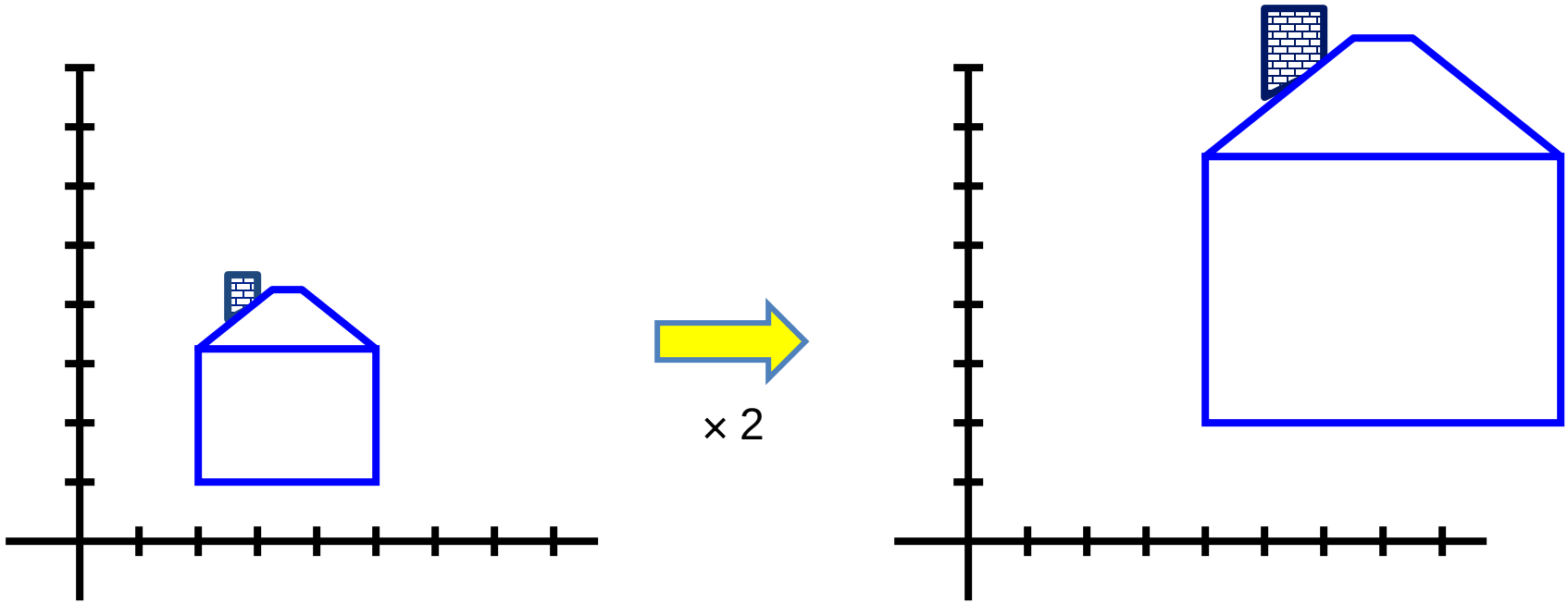
Affine

Perspective

Slide credit (next few slides):
A. Efros and/or S. Seitz

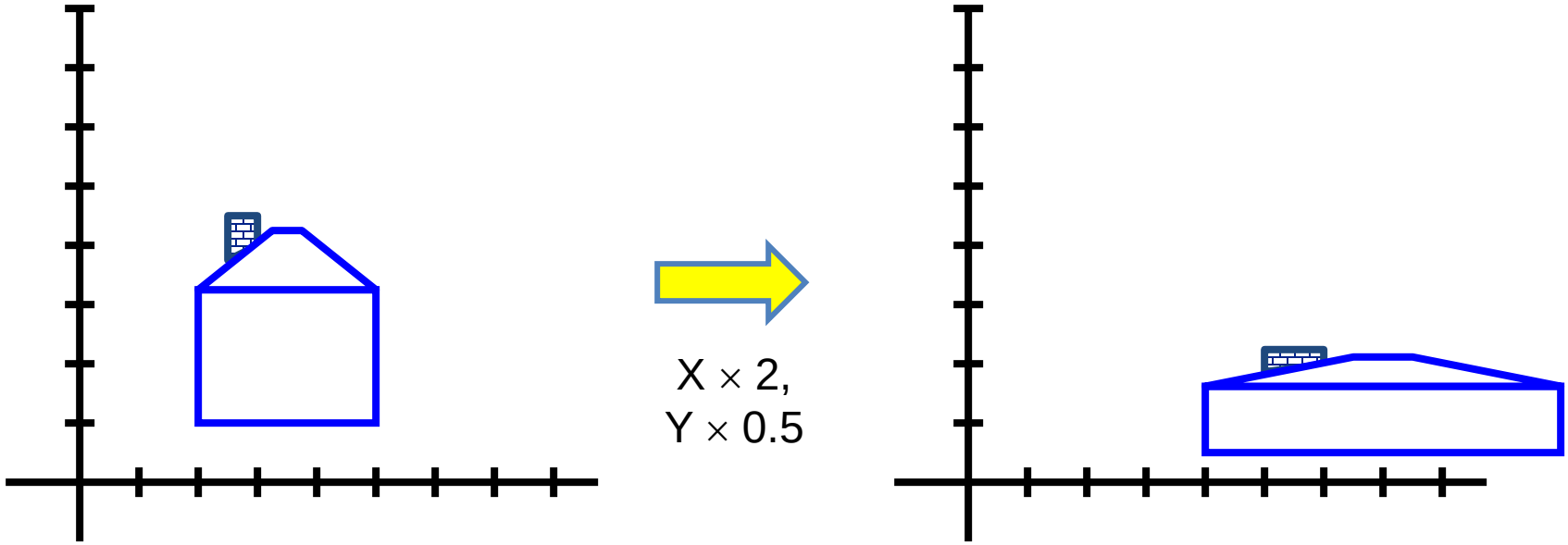
Scaling

- *Scaling* a coordinate means multiplying each of its components by a scalar
- *Uniform scaling* means this scalar is the same for all components:



Scaling

- *Non-uniform scaling*: different scalars per component:



Scaling

- Scaling operation:

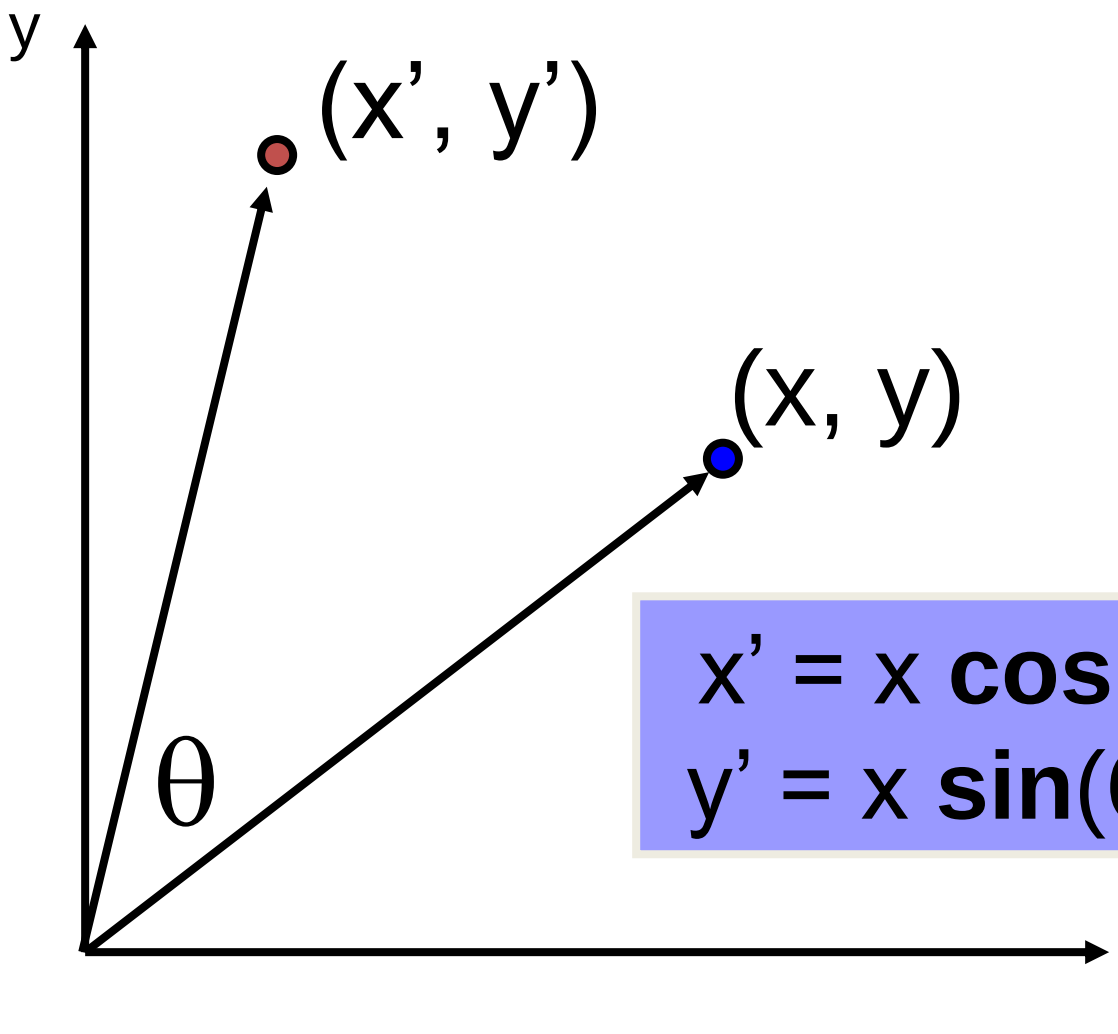
$$x' = ax$$

$$y' = by$$

- Or, in matrix form:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \underbrace{\begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix}}_{\text{scaling matrix } S} \begin{bmatrix} x \\ y \end{bmatrix}$$

2-D Rotation



$$x' = x\cos(\theta) - y\sin(\theta)$$
$$y' = x\sin(\theta) + y\cos(\theta)$$

2-D Rotation

This is easy to capture in matrix form:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \underbrace{\begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}}_{\mathbf{R}} \begin{bmatrix} x \\ y \end{bmatrix}$$

Even though sin(θ) and cos(θ) are nonlinear functions of θ,

- ***x' is a linear combination of x and y***
- ***y' is a linear combination of x and y***

What is the inverse transformation?

- Rotation by –θ
- For rotation matrices $\mathbf{R}^{-1} = \mathbf{R}^{T}$

Basic 2D transformations

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

Scale

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & \alpha_x \\ \alpha_y & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Shear

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\Theta & -\sin\Theta \\ \sin\Theta & \cos\Theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Rotate

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

Translate

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Affine

Affine is any combination of translation, scale, rotation, and shear

Affine Transformations

Affine transformations are combinations of

- Linear transformations, and
- Translations

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

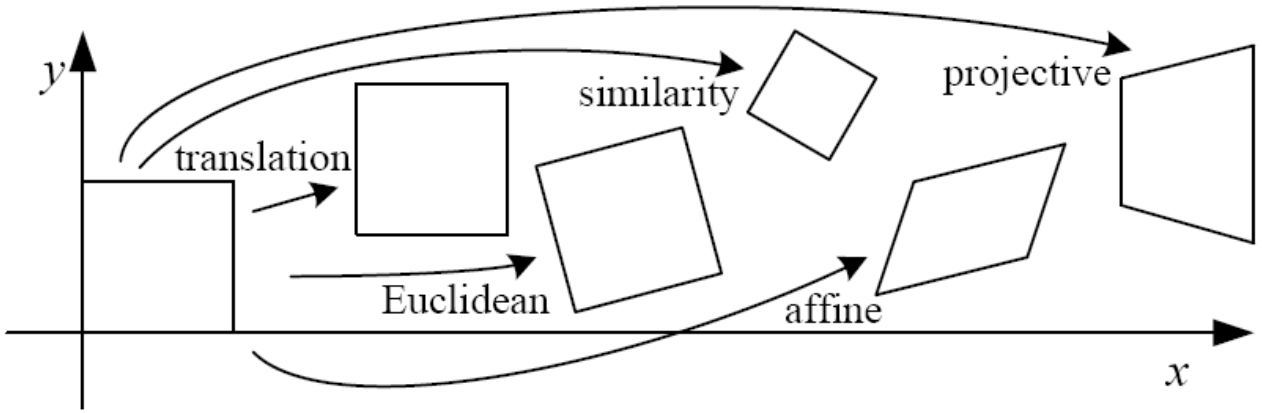
or

Properties of affine transformations:

- Lines map to lines
- Parallel lines remain parallel
- Ratios are preserved
- Closed under composition

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

2D image transformations (reference table)



Name	Matrix	# D.O.F.	Preserves:	Icon
translation	$\begin{bmatrix} \boldsymbol{I} \mid \boldsymbol{t} \end{bmatrix}_{2\times3}$	2	orientation $+\cdots$	
rigid (Euclidean)	$\begin{bmatrix} \boldsymbol{R} \mid \boldsymbol{t} \end{bmatrix}_{2\times3}$	3	lengths $+\cdots$	
similarity	$\begin{bmatrix} s\boldsymbol{R} \mid \boldsymbol{t} \end{bmatrix}_{2\times3}$	4	angles $+\cdots$	
affine	$\begin{bmatrix} \boldsymbol{A} \end{bmatrix}_{2\times3}$	6	parallelism $+\cdots$	
projective	$\begin{bmatrix} \tilde{\boldsymbol{H}} \end{bmatrix}_{3\times3}$	8	straight lines	

'Homography'

Szeliski 2.1

Projective Transformations

Projective transformations are combos of

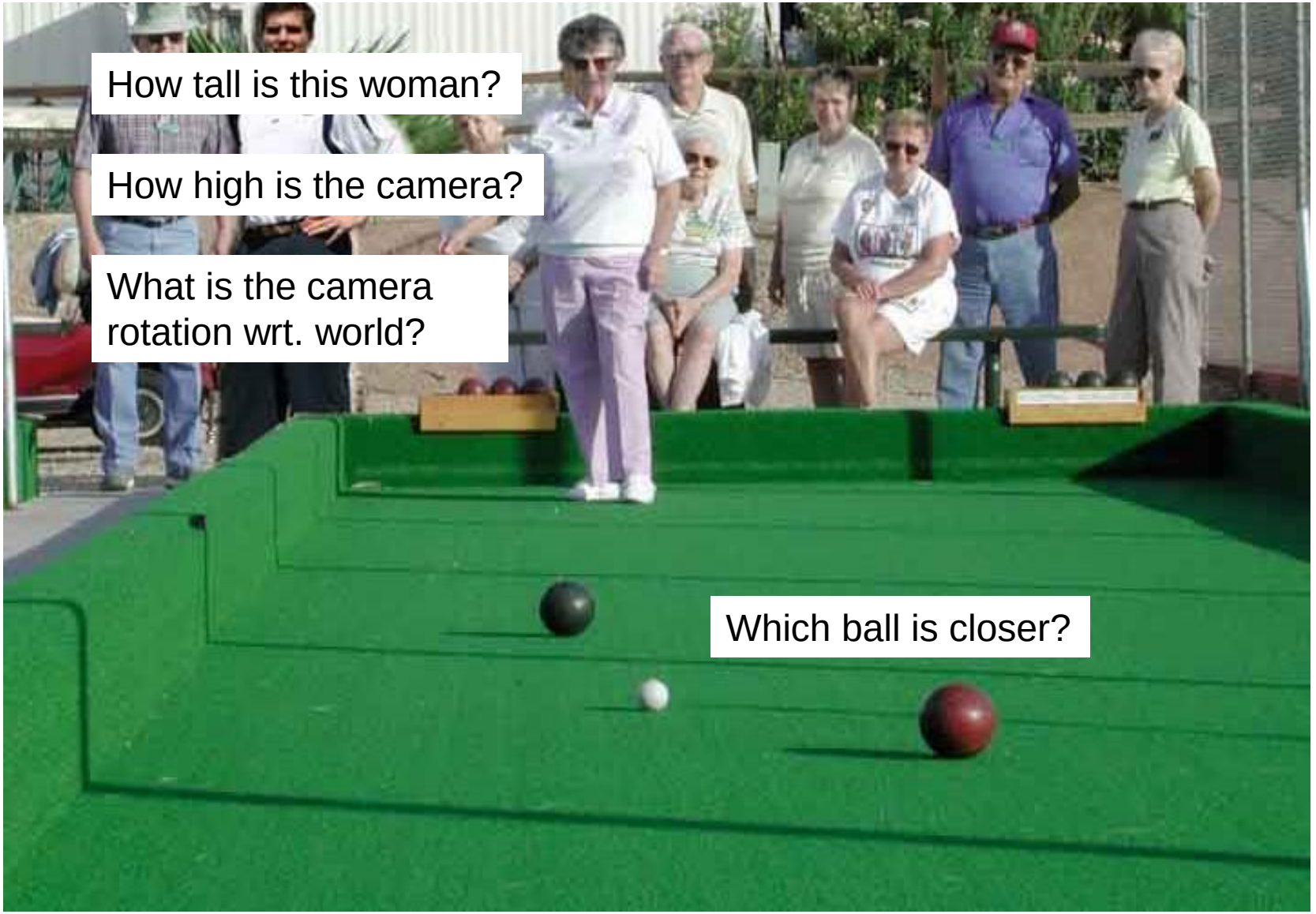
- Affine transformations, and
- Projective warps

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

Properties of projective transformations:

- Lines map to lines
- Parallel lines do not necessarily remain parallel
- Ratios are not preserved
- Closed under composition
- Models change of basis
- Projective matrix is defined up to a scale (8 degrees of freedom)

Cameras and World Geometry



James Hays

Photo Tourism

Exploring photo collections in 3D

Noah Snavely Steven M. Seitz Richard Szeliski

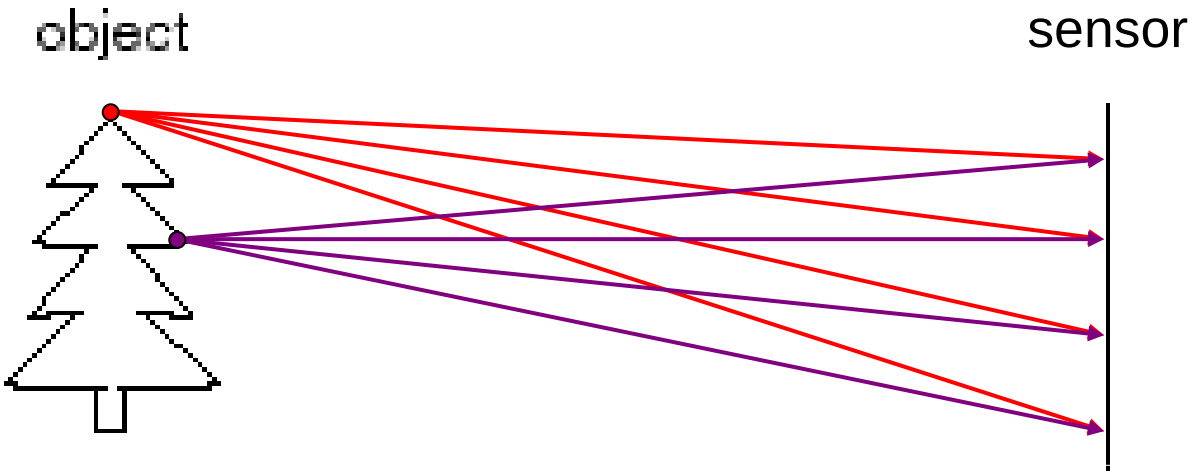
University of Washington *Microsoft Research*

SIGGRAPH 2006

Let’s design a camera

Idea 1: Put a sensor in front of an object

Do we get a reasonable image?

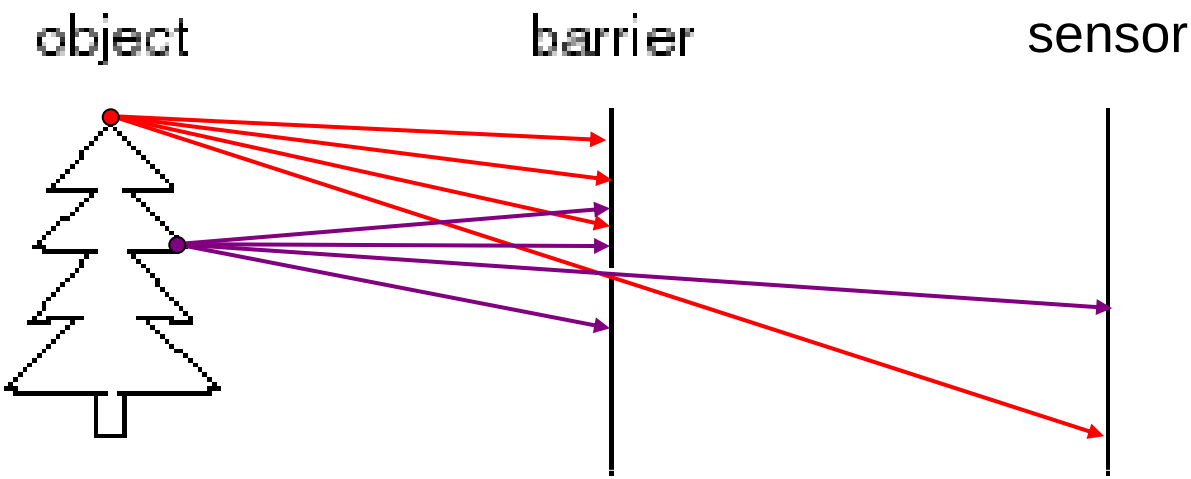


Slide source: Seitz

Let's design a camera

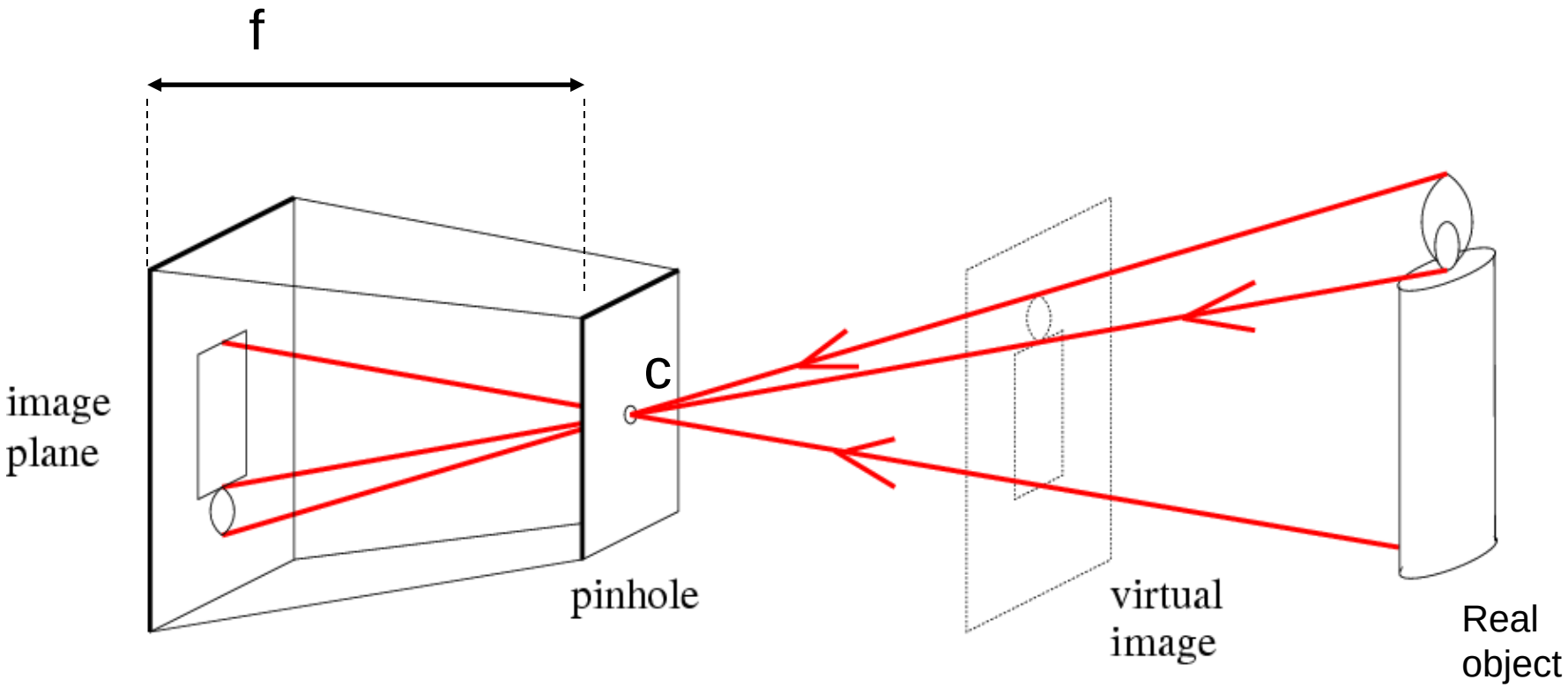
Idea 2: Add a barrier to block most rays

- 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

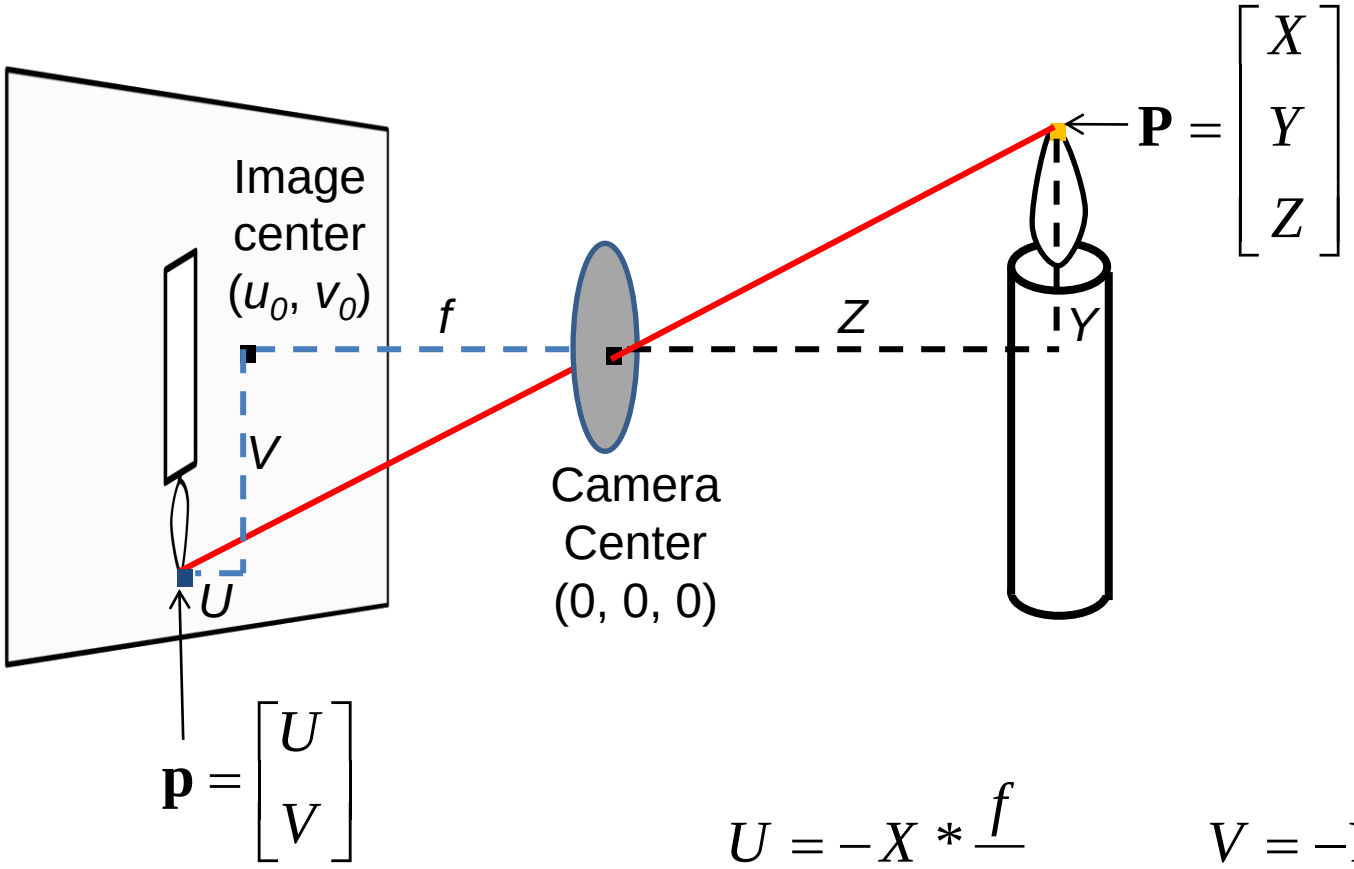


f = Focal length

c = Optical center of the camera

Figure from Forsyth

Projection: world coordinates→image coordinates



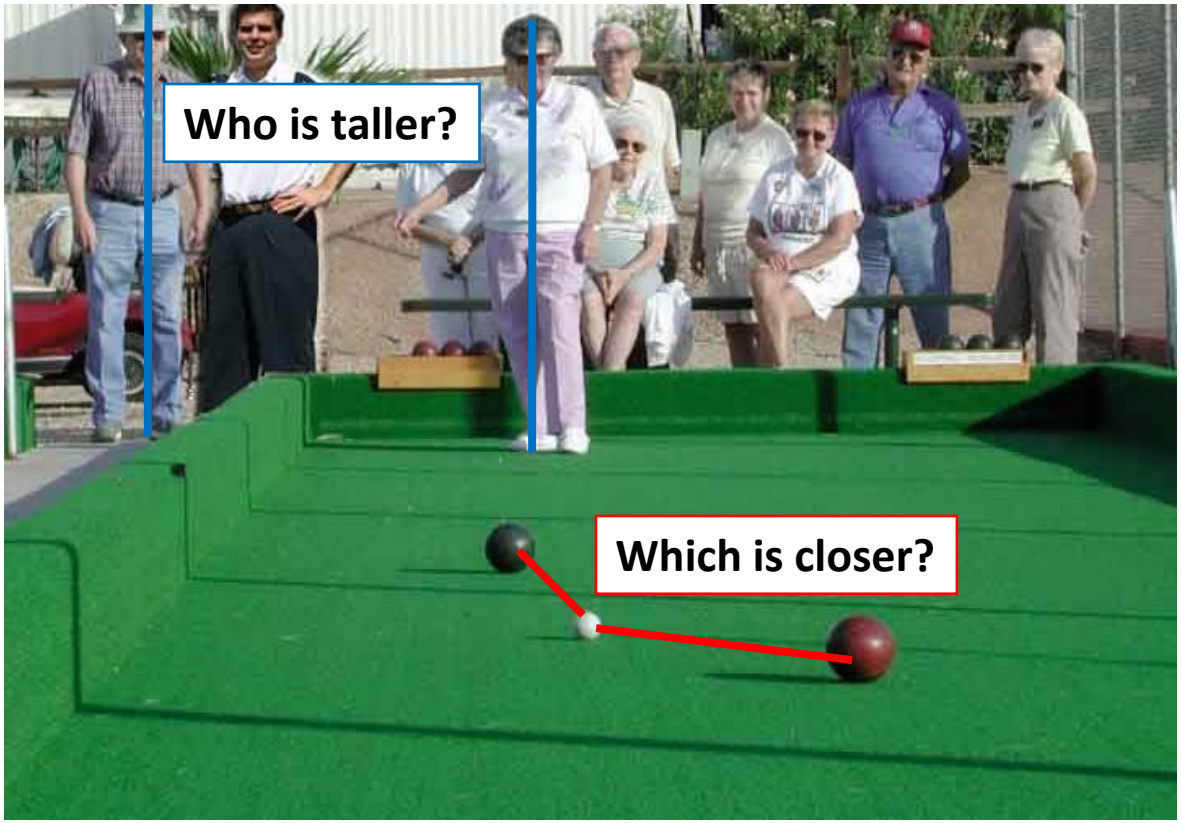
$$U = -X * \frac{f}{Z} \qquad V = -Y * \frac{f}{Z}$$

p = distance from image center

What is the effect if f and Z are equal?

Projective Geometry

Length (and so area) is lost.



Length and area are not preserved

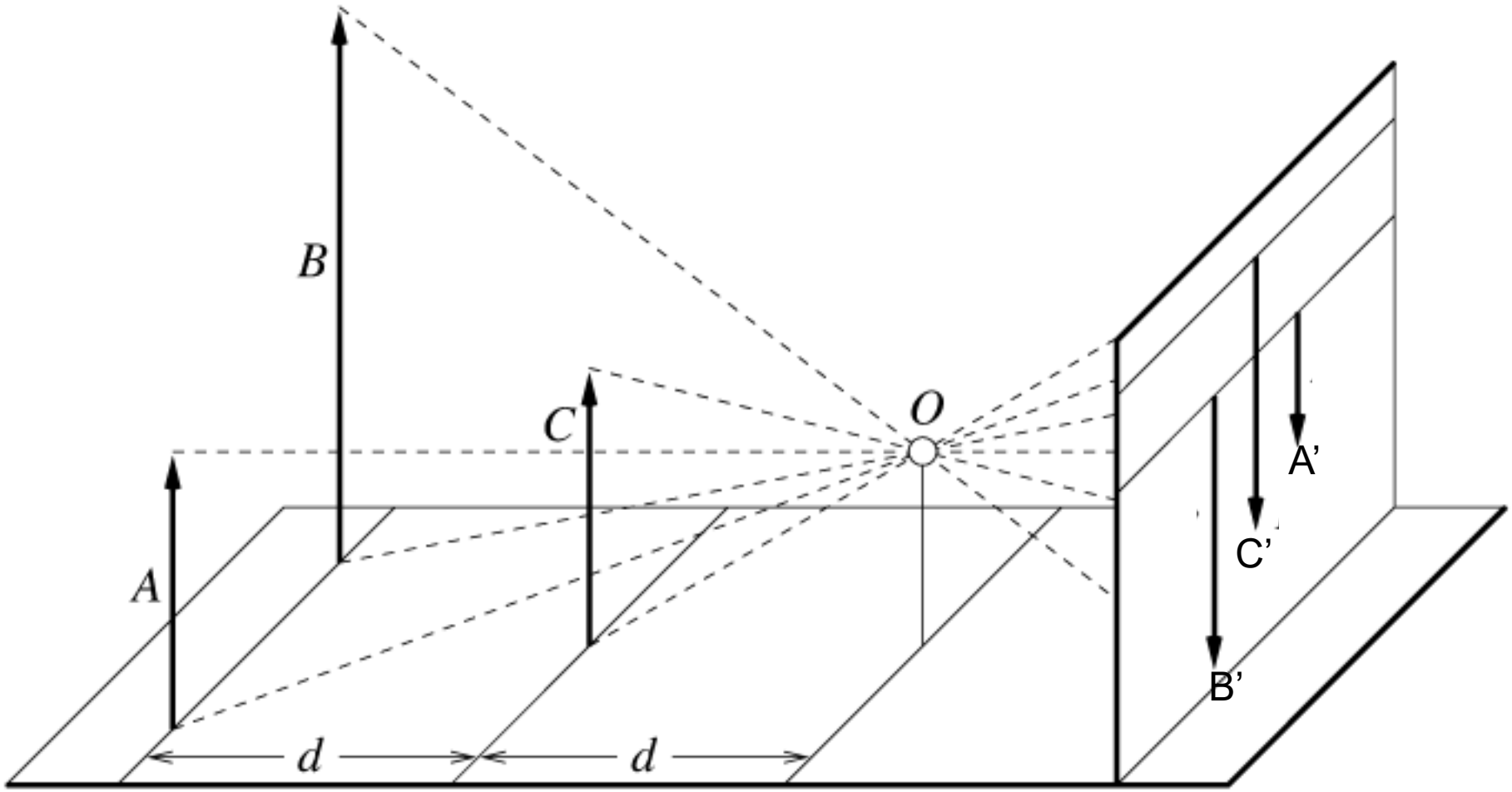
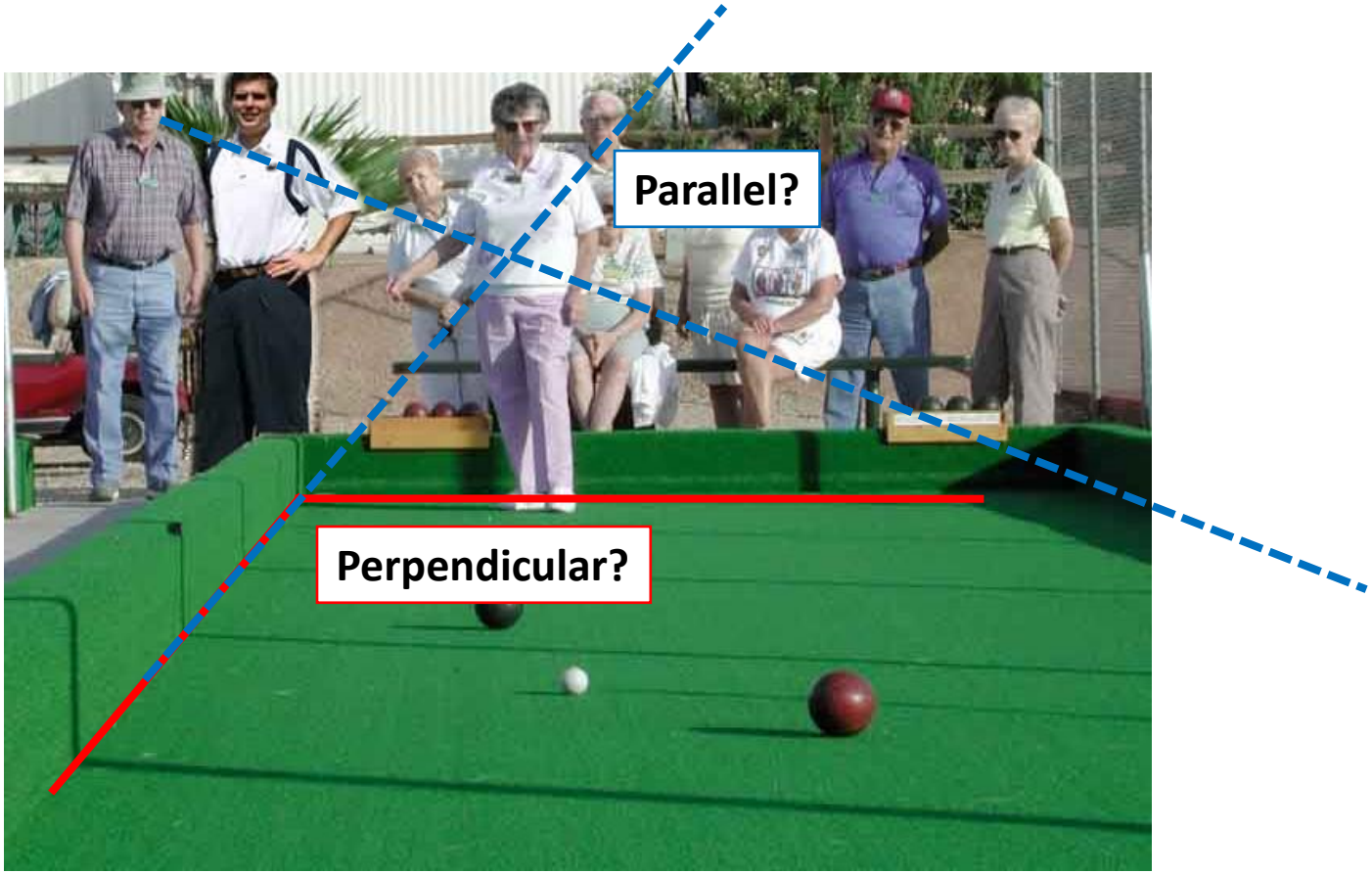


Figure by David Forsyth

Projective Geometry

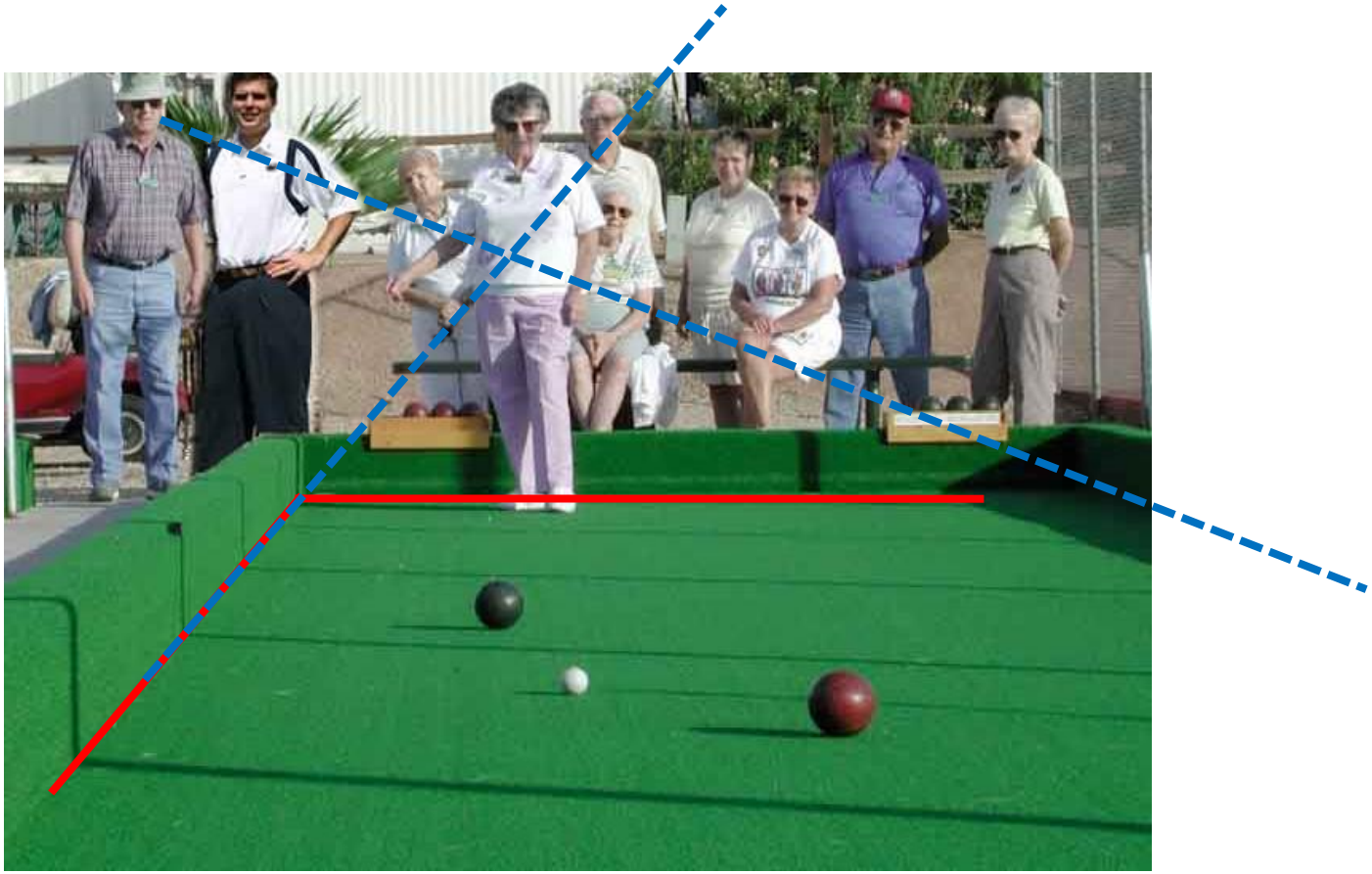
Angles are lost.



Projective Geometry

What is preserved?

- Straight lines are still straight.

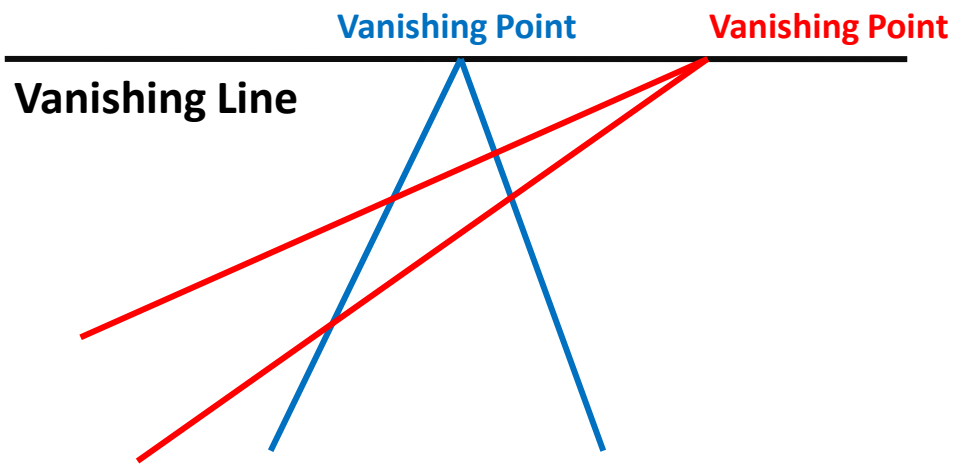


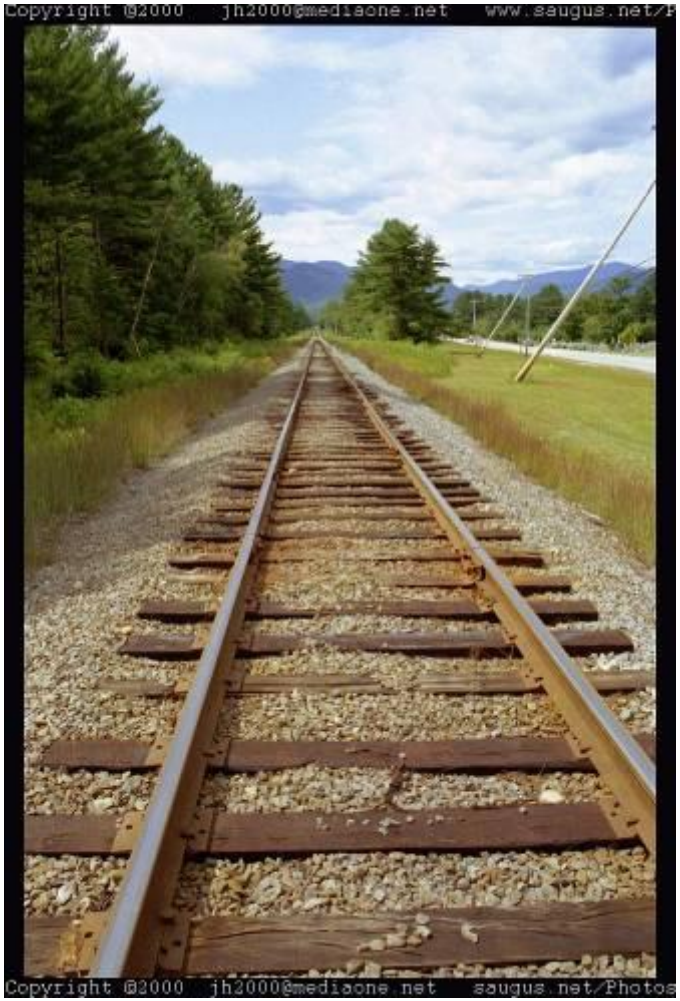
Vanishing points and lines

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

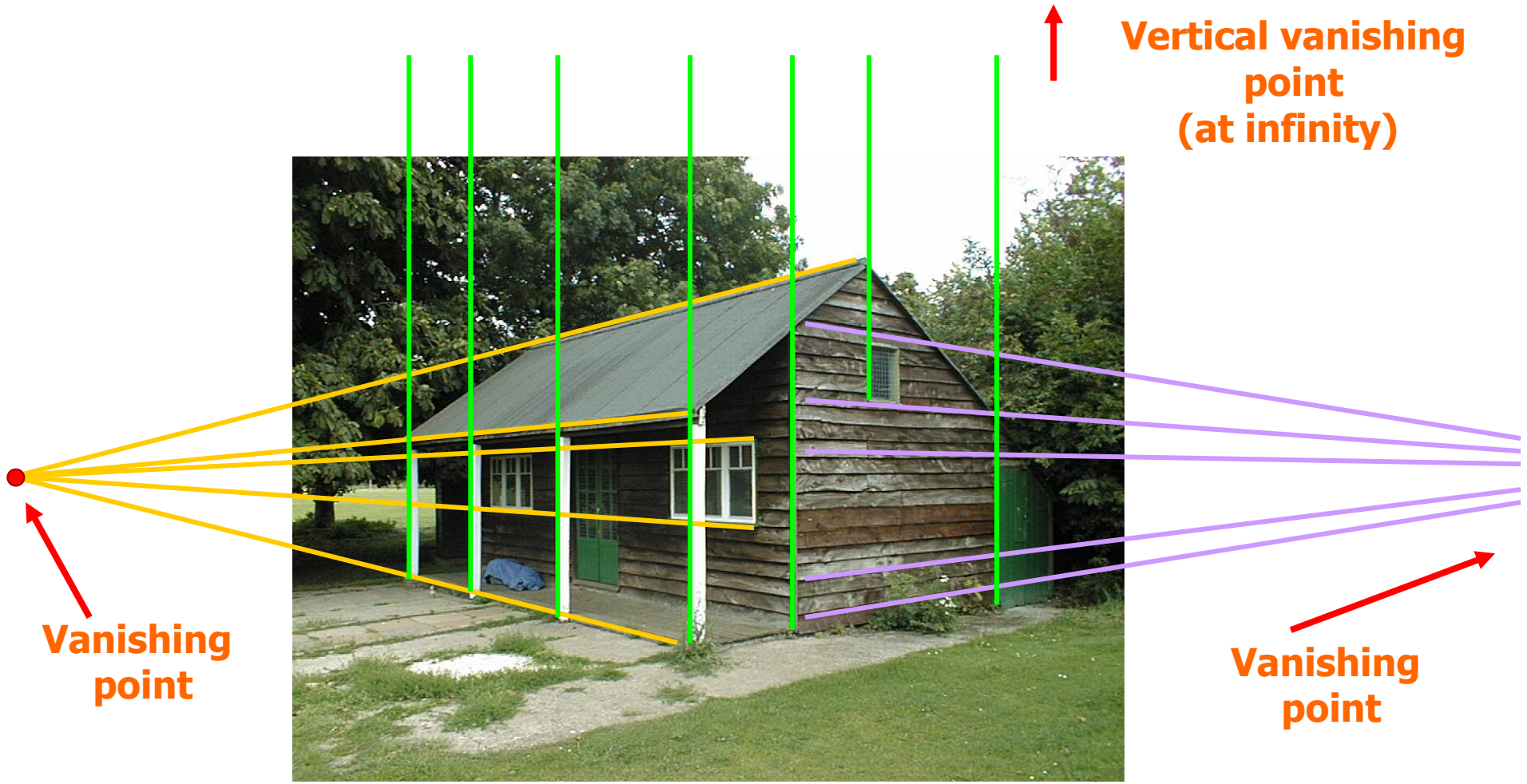
Parallel lines on the same plane in the world converge to vanishing points on a "vanishing line".

E.G., the horizon.



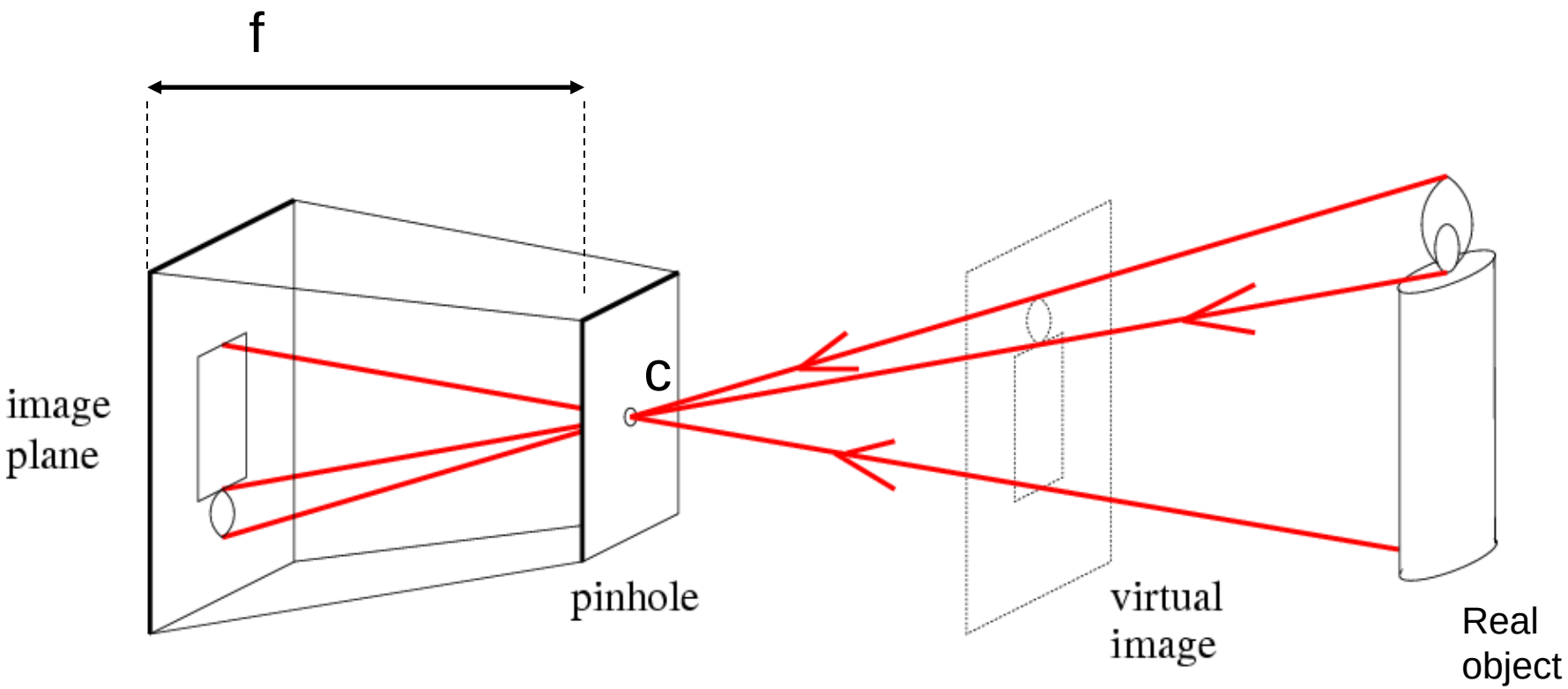


Vanishing points and lines



Slide from Efros, Photo from Criminisi

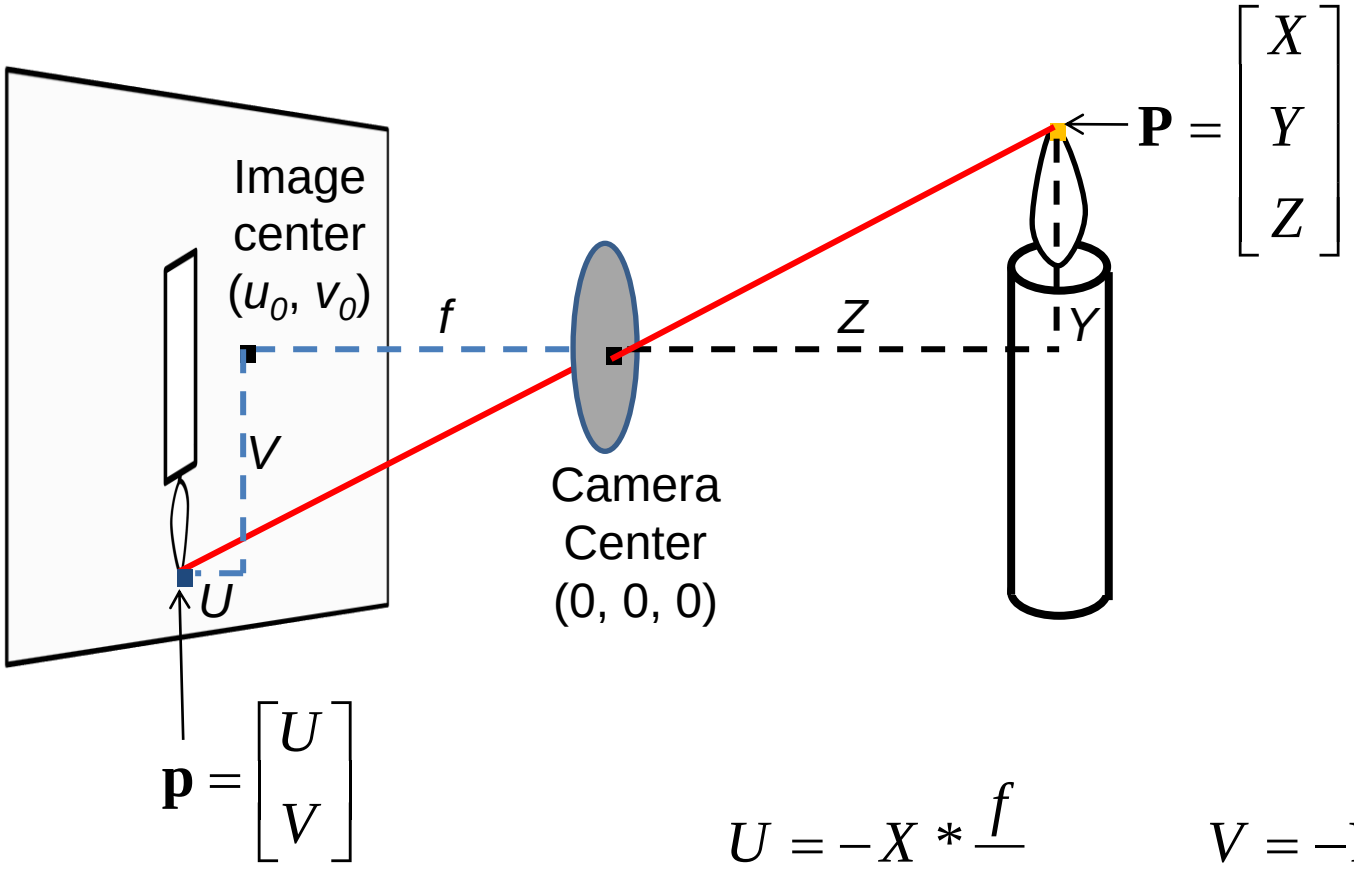
Pinhole camera model



f = Focal length

c = Optical center of the camera

Projection: world coordinates→image coordinates



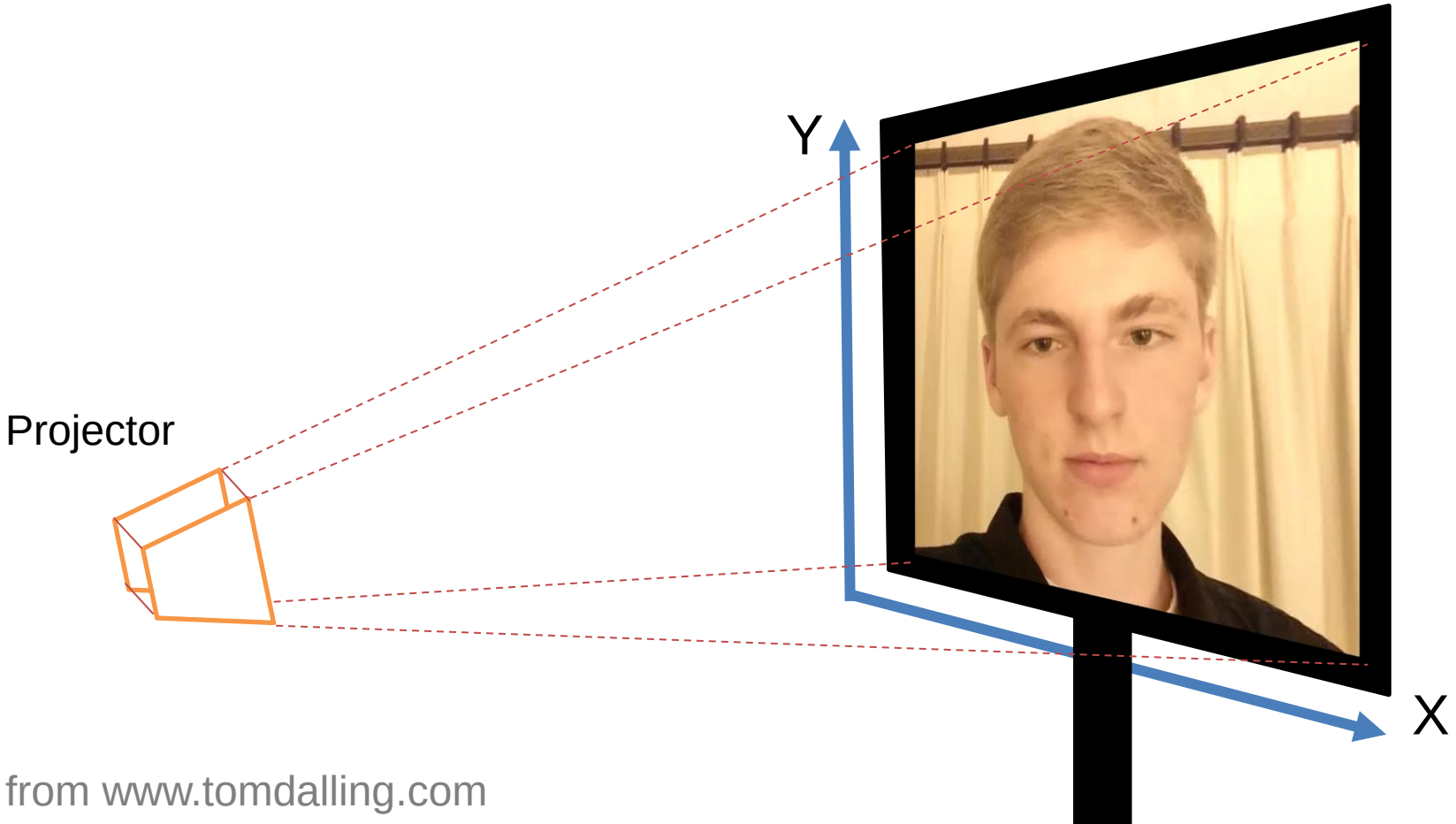
$$U = -X * \frac{f}{Z} \qquad V = -Y * \frac{f}{Z}$$

p = distance from image center

What is the effect if f and Z are equal?

Projective geometry

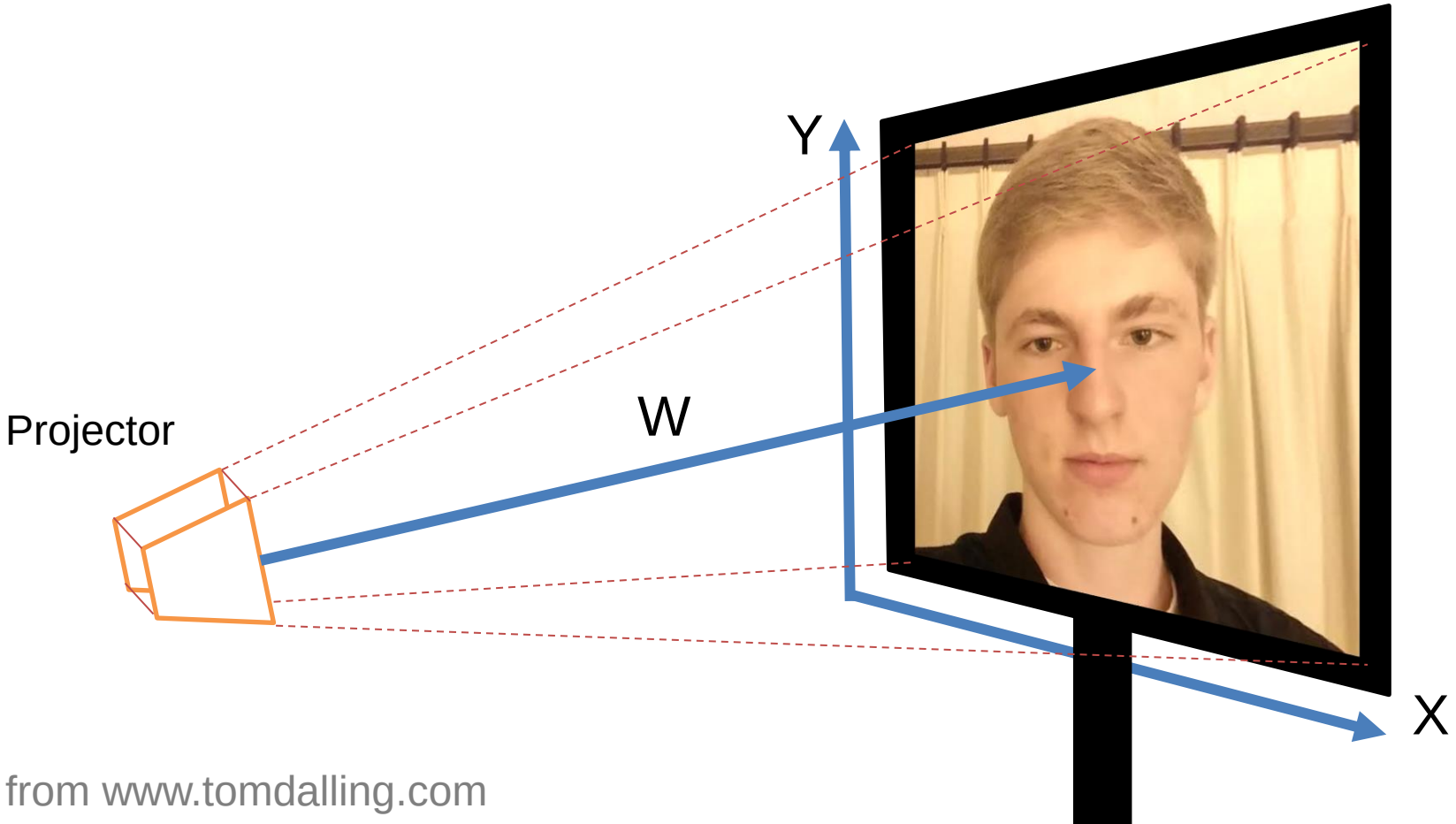
- 2D point in cartesian = (x,y) coordinates
- 2D point in projective = (x,y,w) coordinates



Idea from www.tomdalling.com

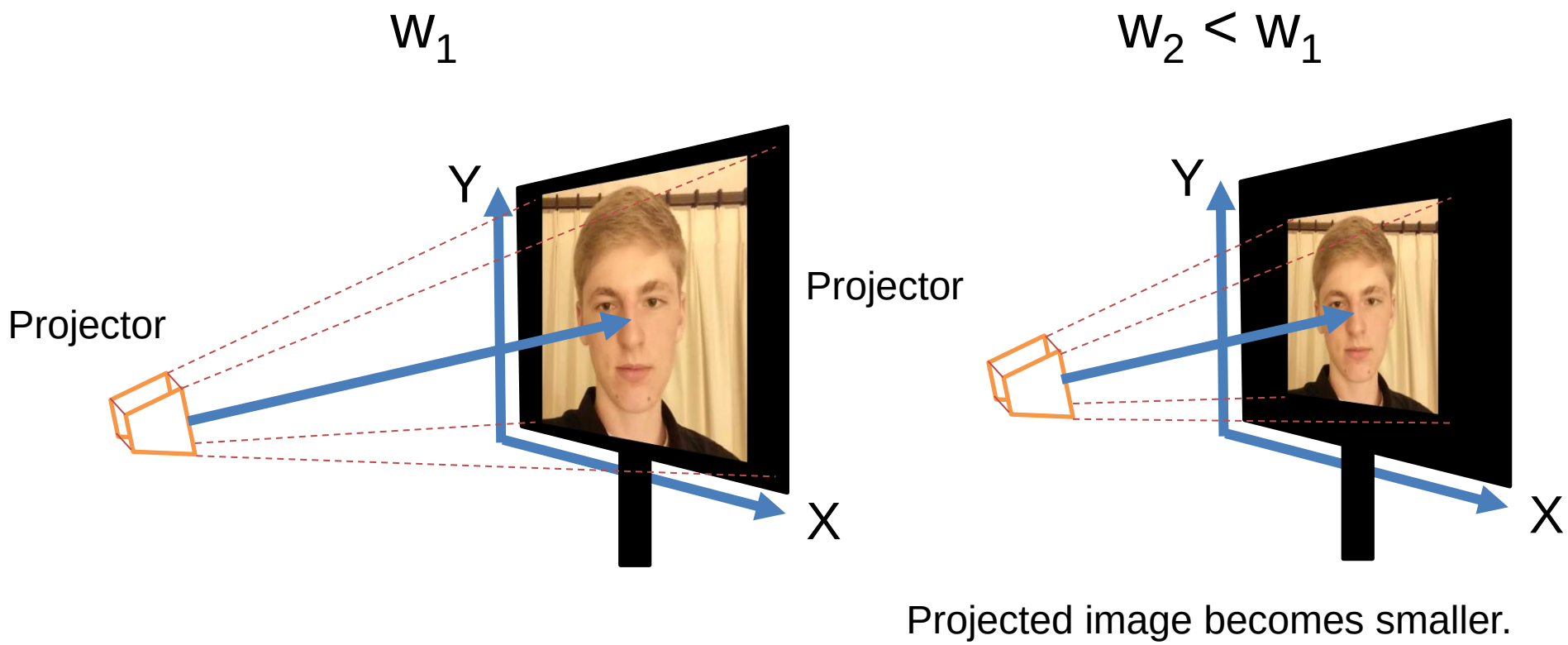
Projective geometry

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



Idea from www.tomdalling.com

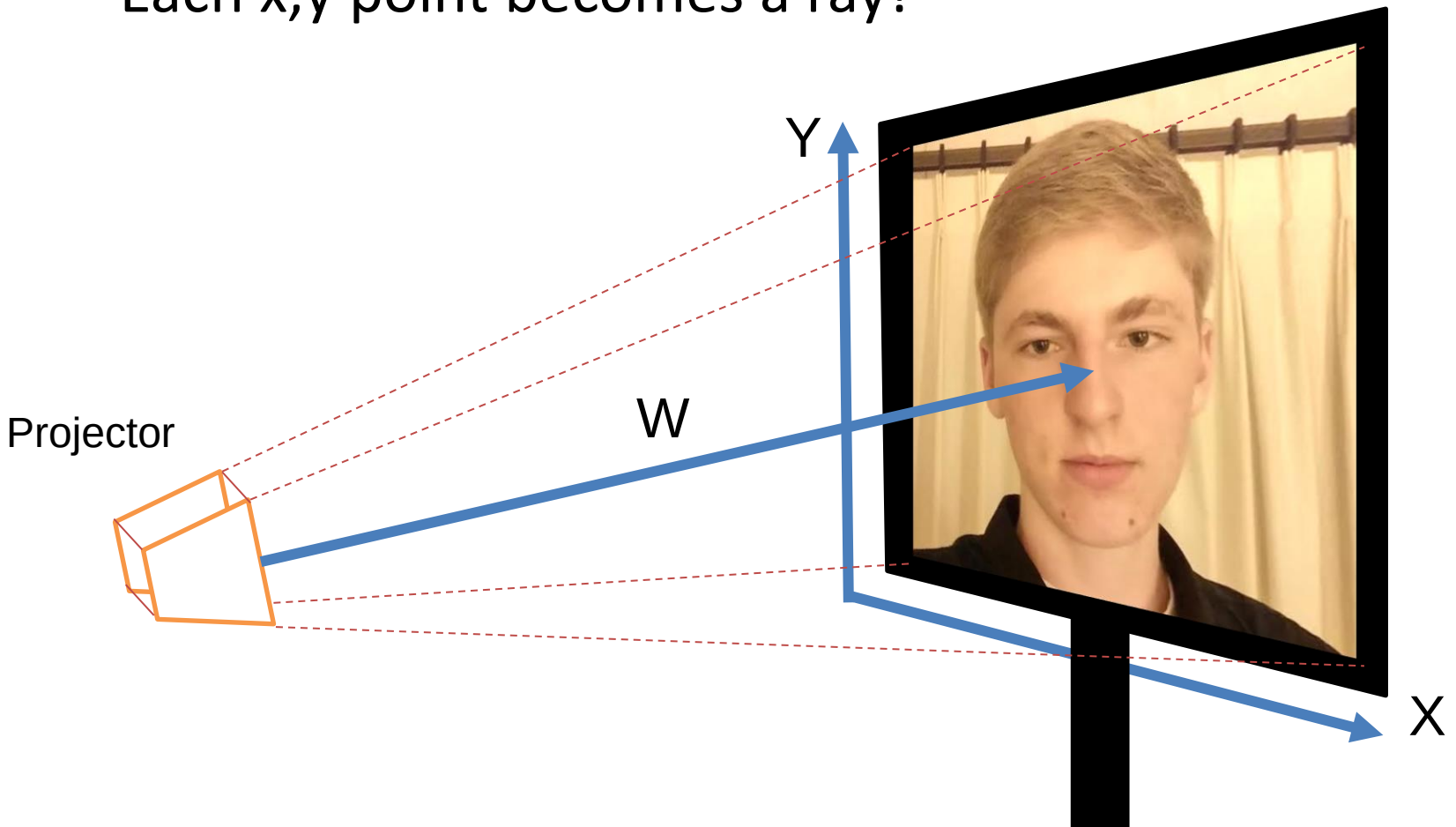
Varying w



Projected image becomes smaller.

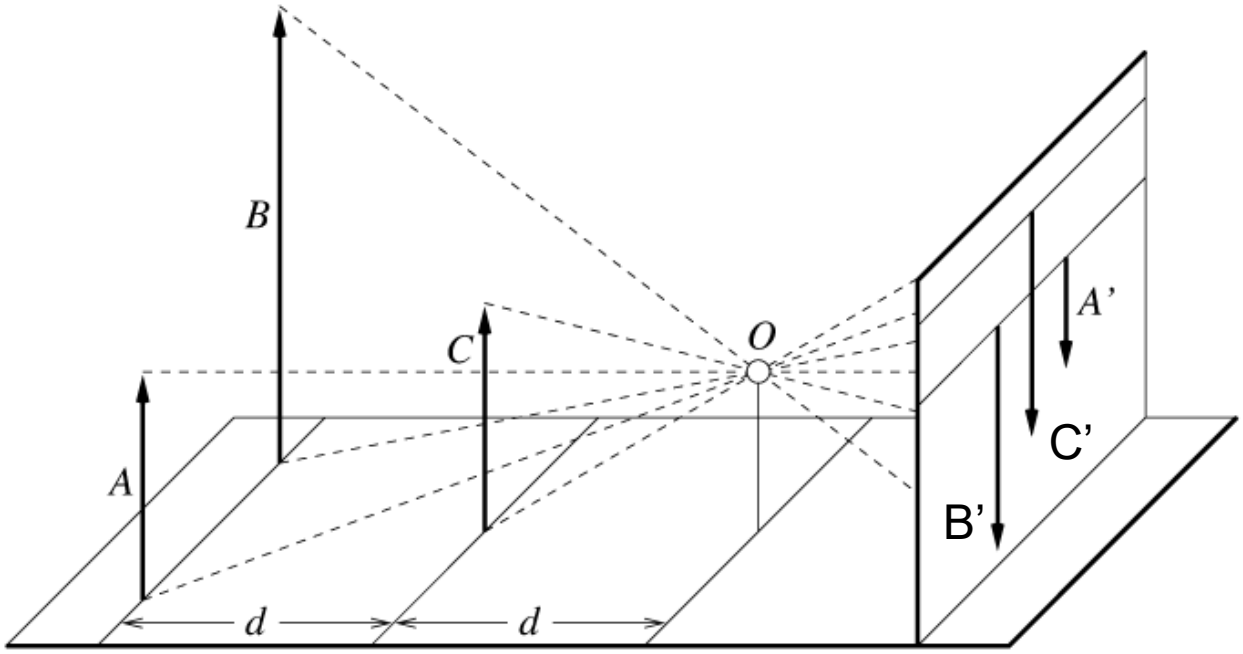
Projective geometry

- 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

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

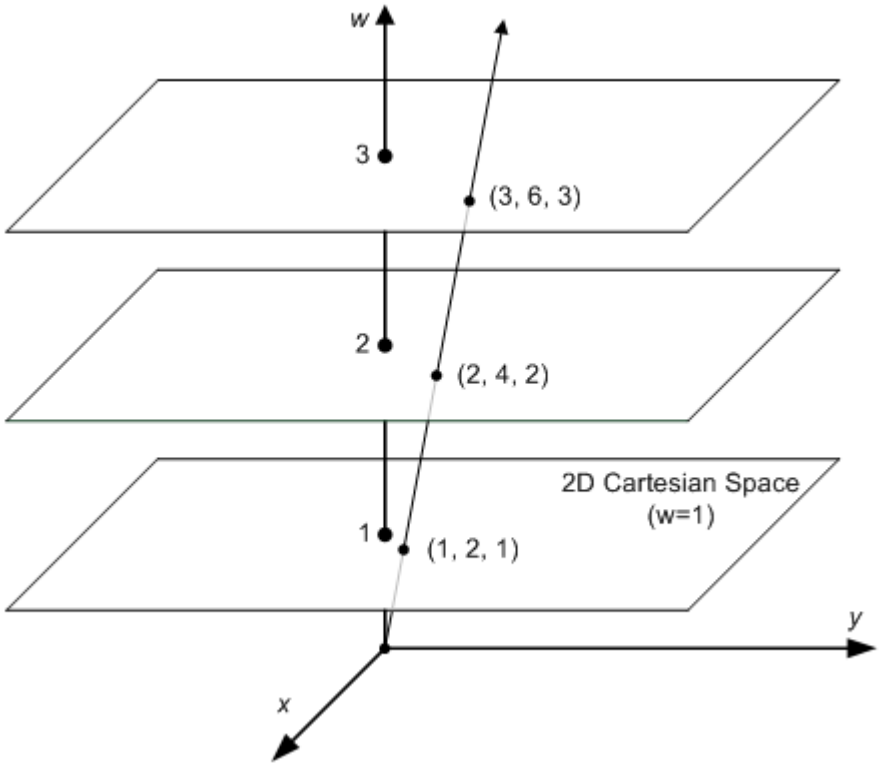
Homogeneous Coordinates

Cartesian Coordinates

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

Homogeneous coordinates

- Projective
- Point becomes a line



To homogeneous

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

From homogeneous

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

Lenses

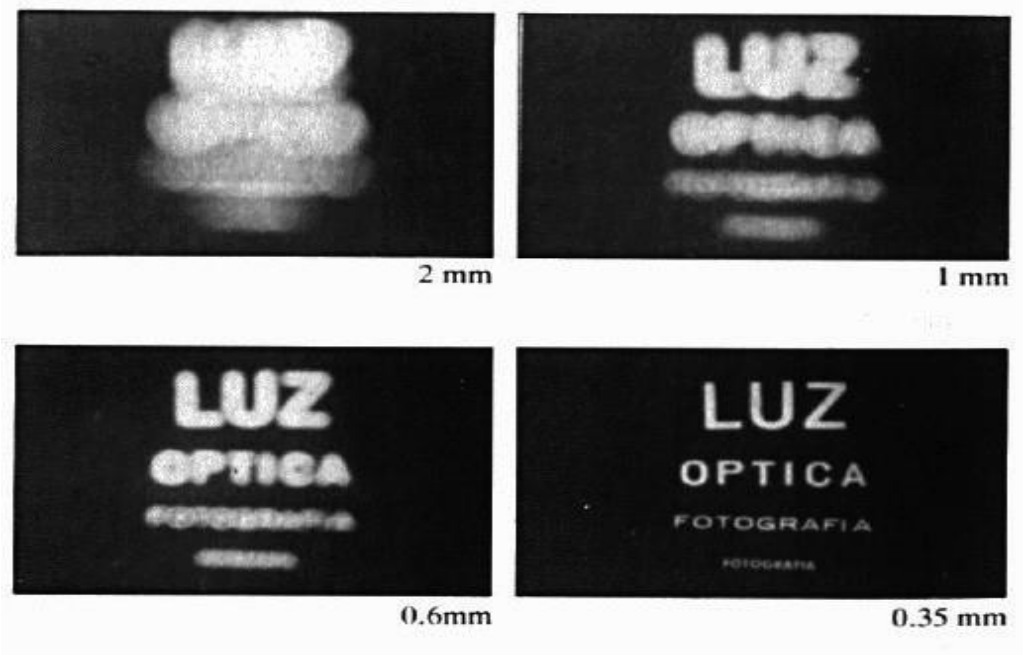
Real cameras aren’t pinhole cameras.

Home-made pinhole camera

Why so blurry?

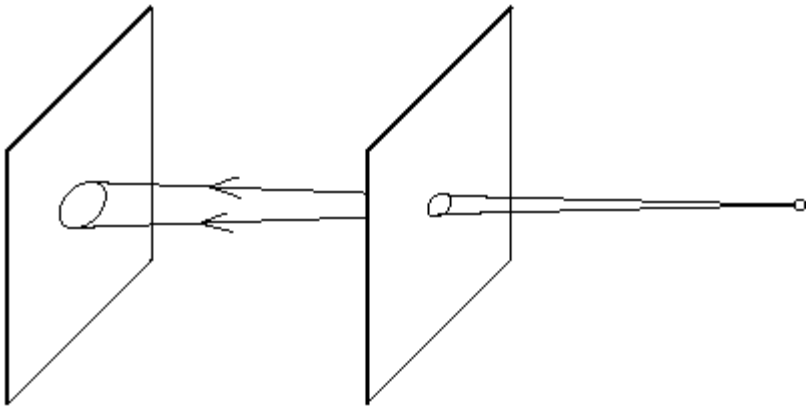
http://www.debevec.org/Pinhole/

Shrinking the aperture



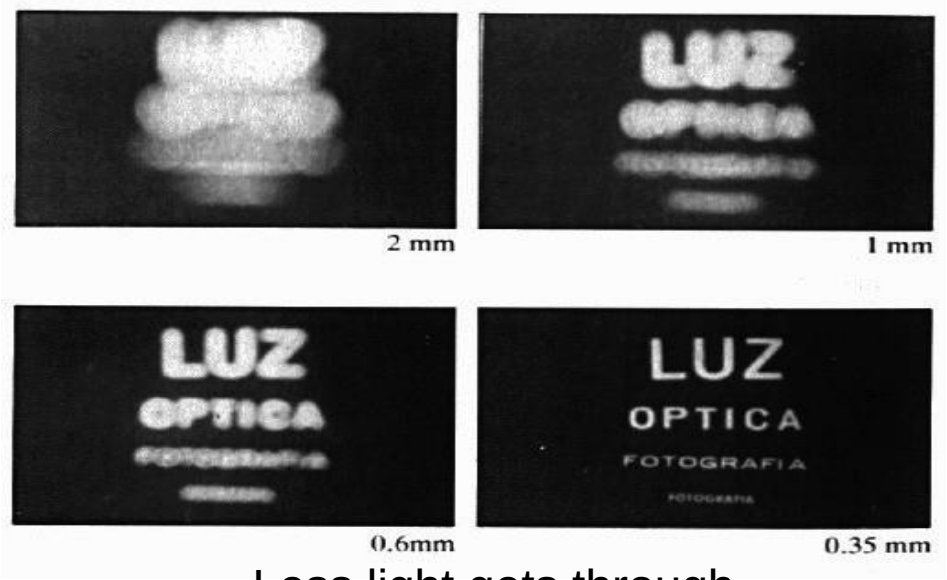
Less light gets through

Integrate over fewer angles



[Steve Seitz]

Shrinking the aperture



Less light gets through

Why not make the aperture as small as possible?

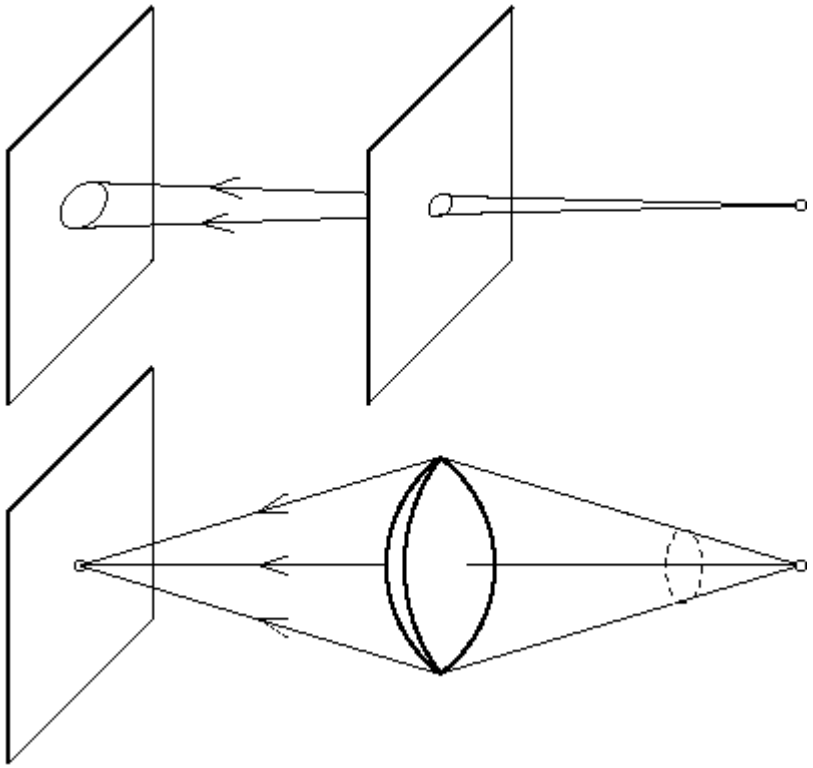
- Less light gets through
- Diffraction effects…

[Steve Seitz]

Shrinking the aperture - diffraction

Light diffracts as wavelength of aperture equals wavelength of light

The reason for lenses

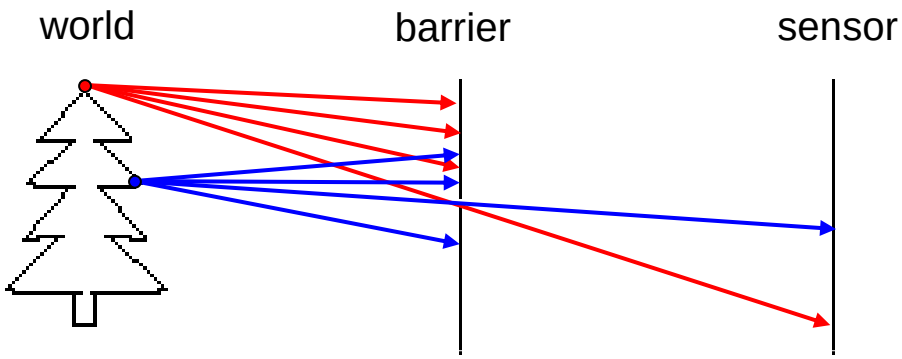


Slide by Steve Seitz

Let's design a camera

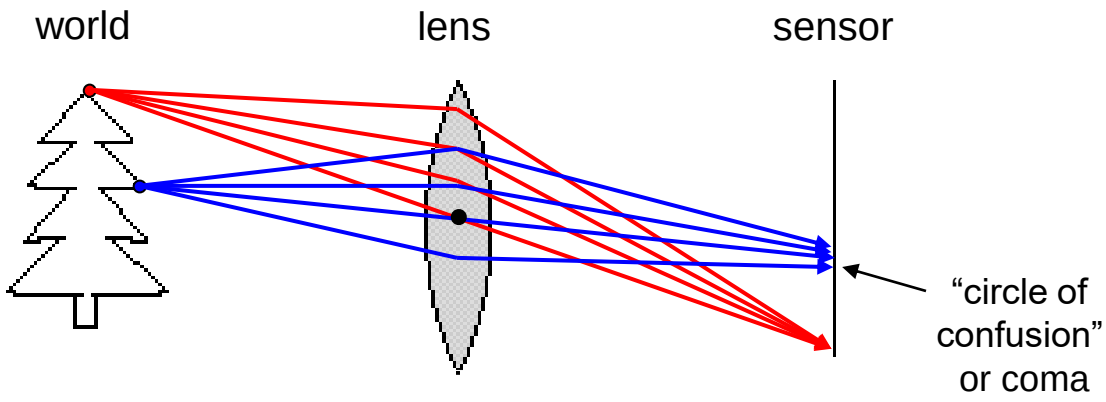
Idea 2: Add a barrier to block most rays

- Pinhole in barrier
- Only sense light from one direction.
  - Reduces blurring.
- In most cameras, this **aperture** can vary in size.



Slide source: Seitz

Focus and Defocus

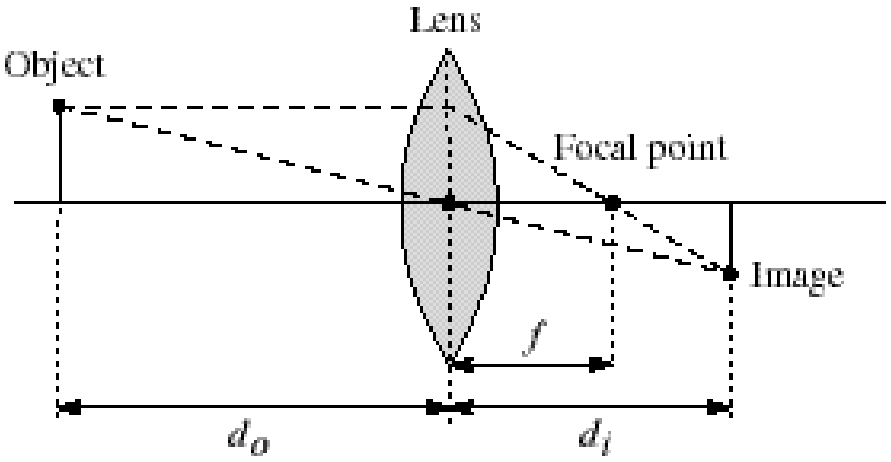


A lens focuses light onto the film

- 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

Slide by Steve Seitz

Thin lenses

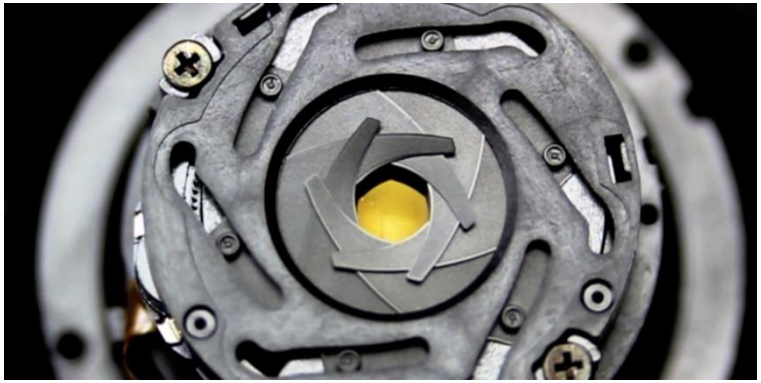


Thin lens equation:

$$\frac{1}{d_o} + \frac{1}{d_i} = \frac{1}{f} \qquad \frac{1}{f} - \frac{1}{d_o} = \frac{1}{d_i}$$

Any object point satisfying this equation is in focus
What is the shape of the focus region?
How can we change the focus region?
Thin lens applet: https://sites.google.com/site/marclevoylectures/applets/operation-of-a-thin-lens
(by Andrew Adams, Nora Willett, Marc Levoy)

Slide by Steve Seitz



Beyond Pinholes: Real apertures

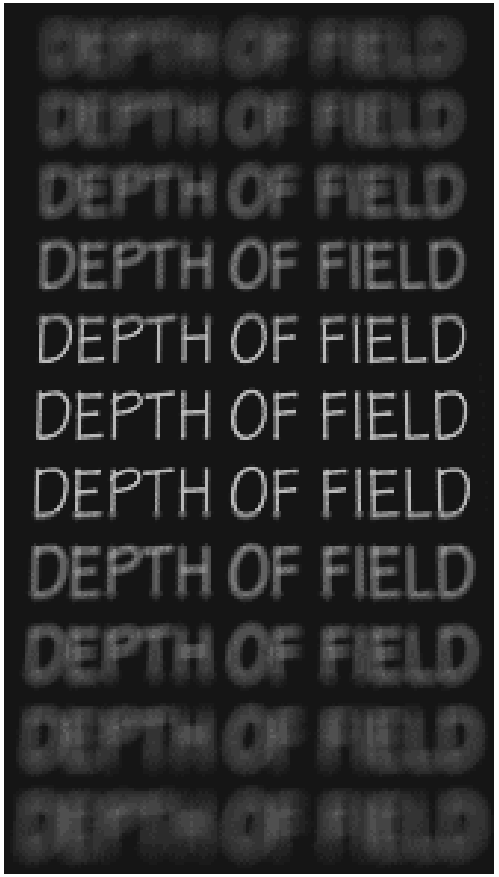
Bokeh:

[Rushif – Wikipedia]

Depth Of Field

Depth of Field

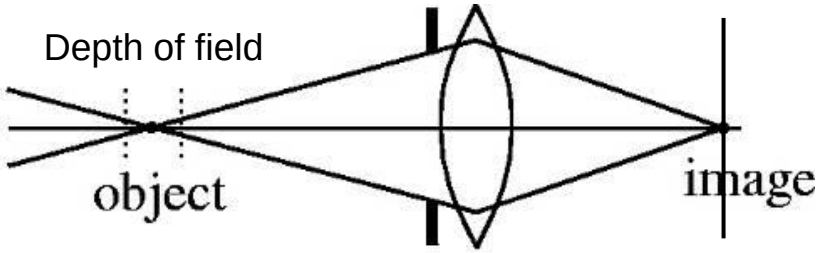
Depth of Field



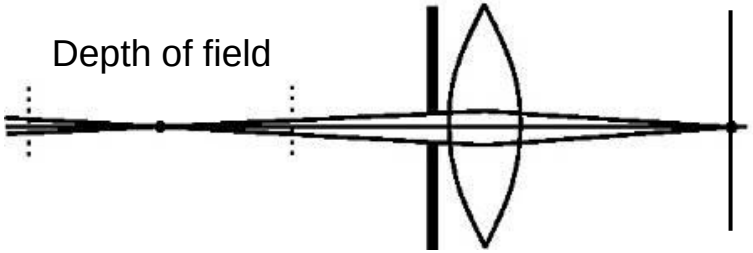
http://www.cambridgeincolour.com/tutorials/depth-of-field.htm

Changing the aperture size affects depth of field

Wide aperture



Narrow aperture



A narrower aperture increases the range in which the object is approximately in focus.

Narrower aperture reduces amount of light – need to increase exposure.

Varying the aperture

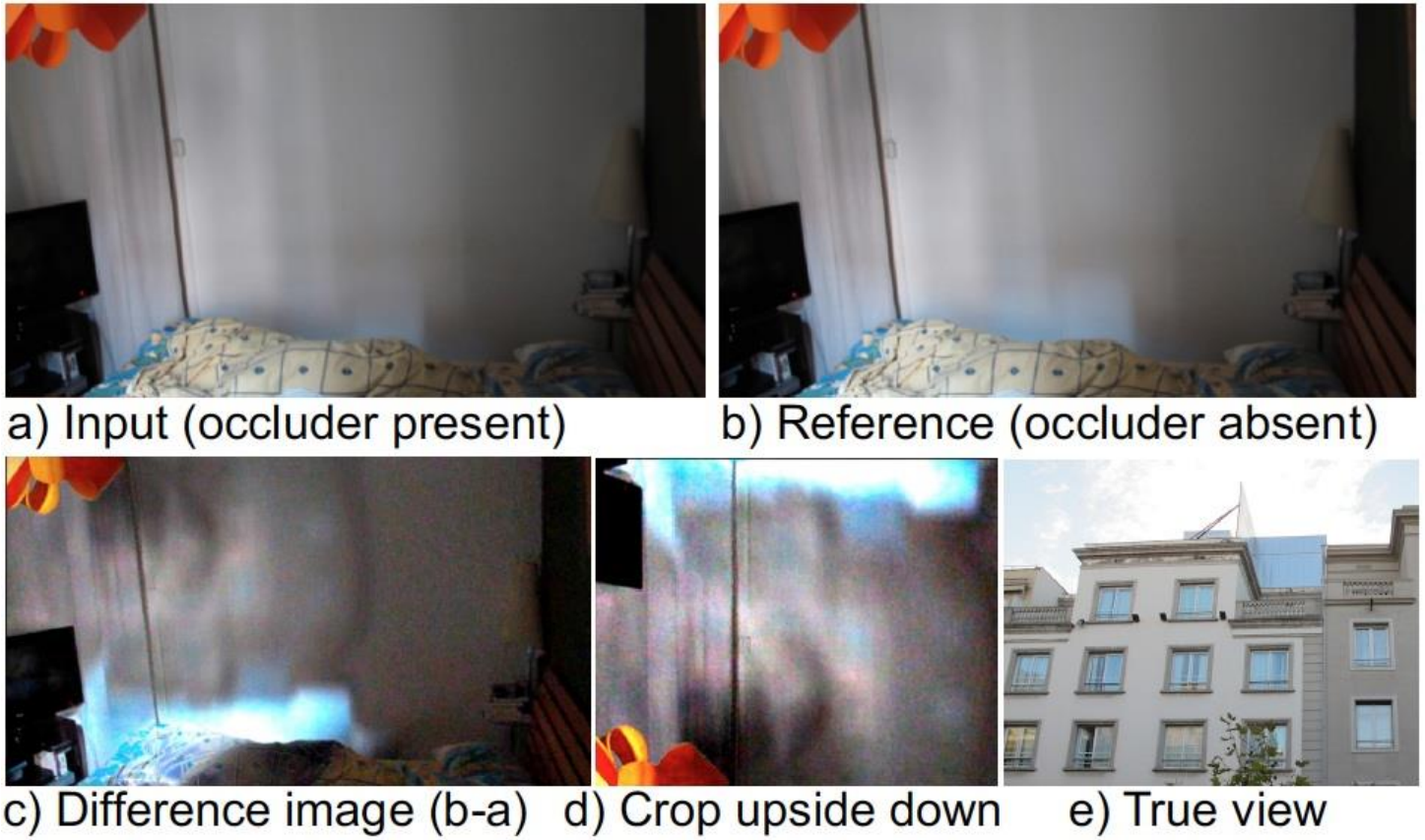
Large aperture = small DOF

Small aperture = large DOF

Accidental Cameras

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

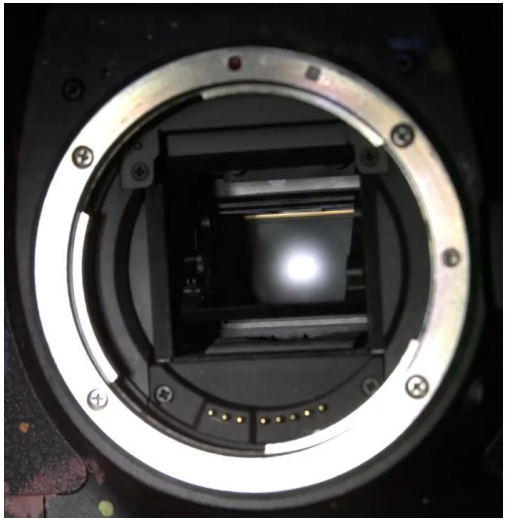
Accidental Cameras

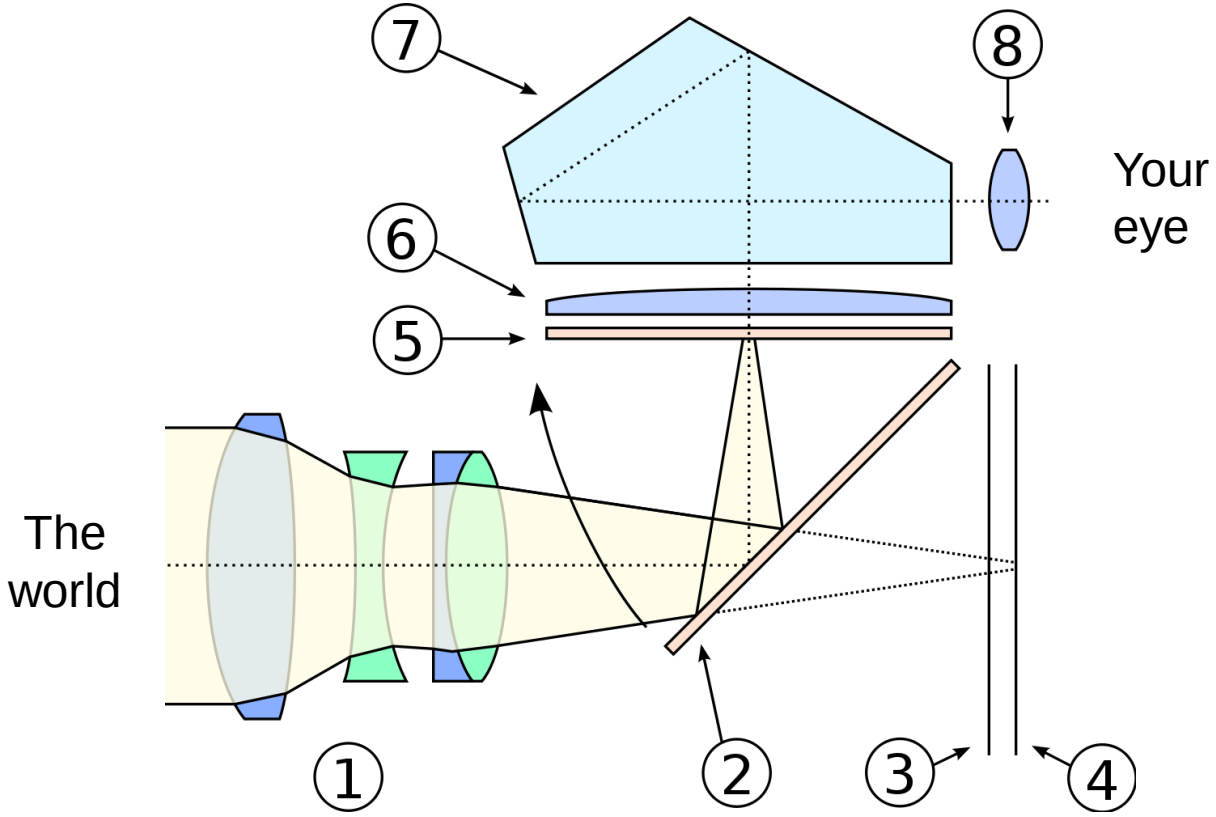


DSLR – Digital Single Lens Reflex Camera

DSLR – Digital Single Lens Reflex Camera

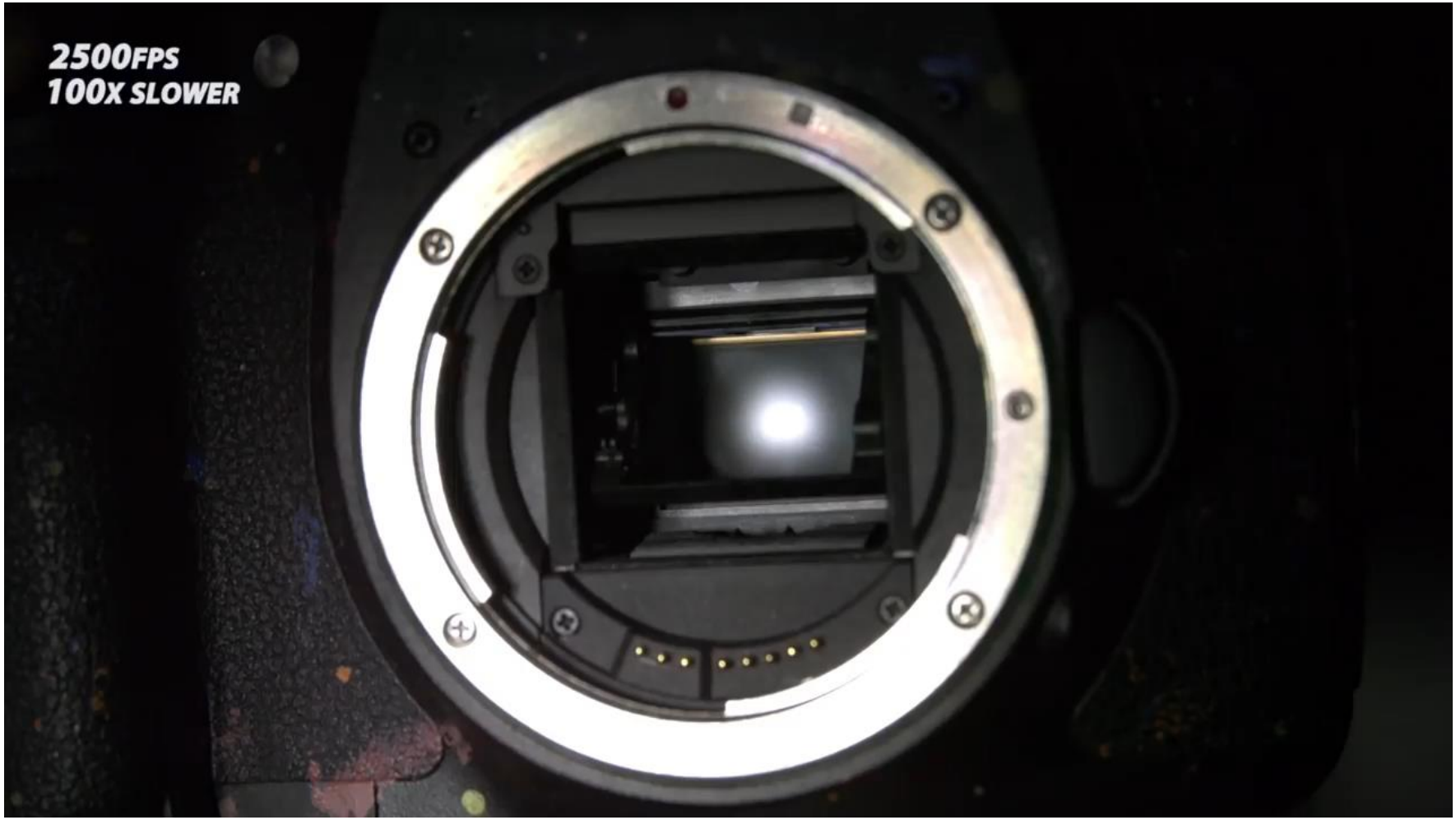
"See what the main lens sees"





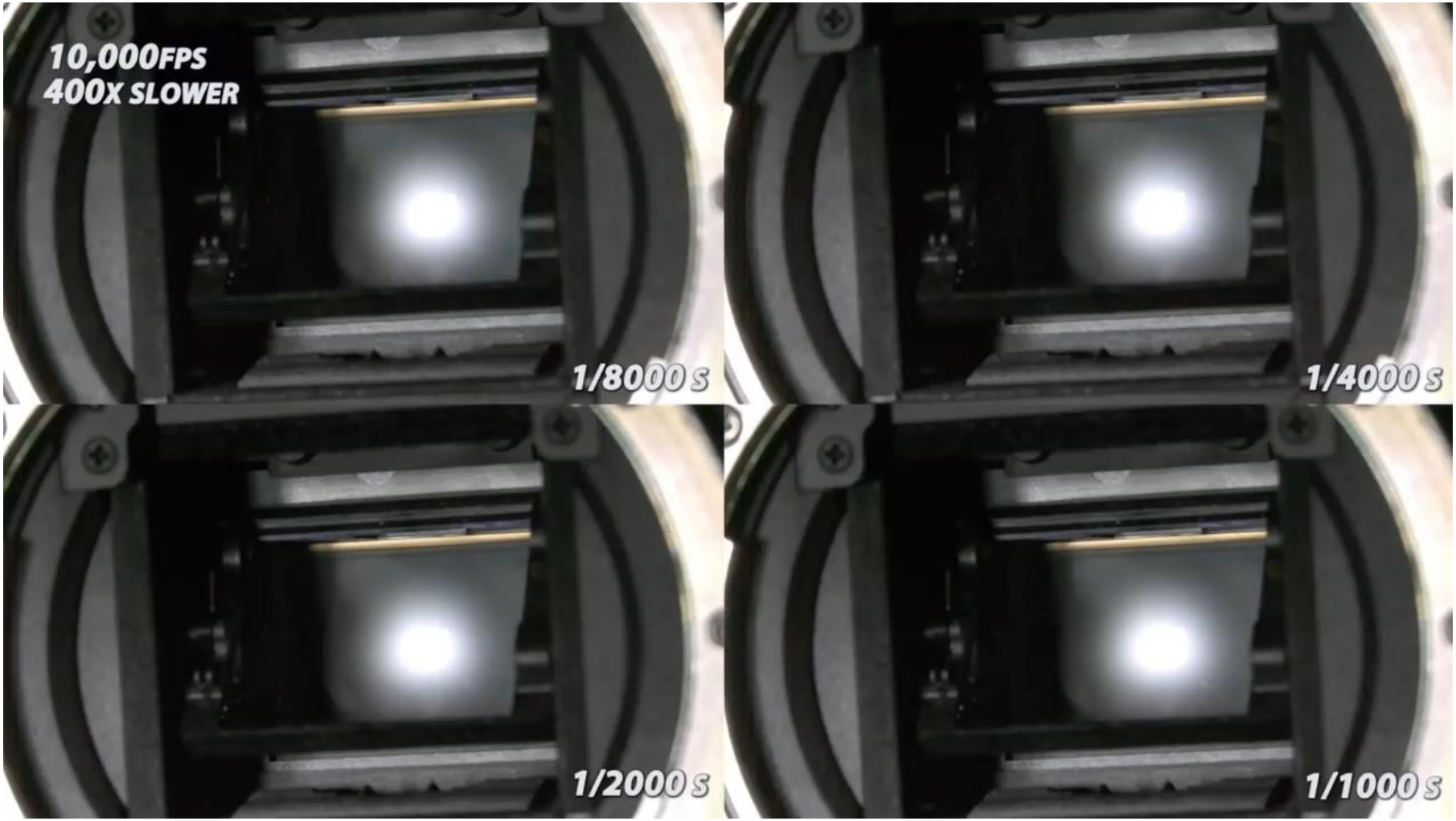
1. Objective (main) lens
2. Mirror
3. Shutter
4. Sensor
5. Mirror in raised position
6. Viewfinder focusing lens
7. Prism
8. Eye prescription lens

Shutters



[The Slo-Mo Guys]

Shutters



[The Slo-Mo Guys]

Shutters

[The Slo-Mo Guys]

Sensors: Rolling shutter vs. global shutter

Many modern cameras have purely digital shutters.

[Reddit – r/educationalgifs – u/Mass1m01973]

Sensor ISO

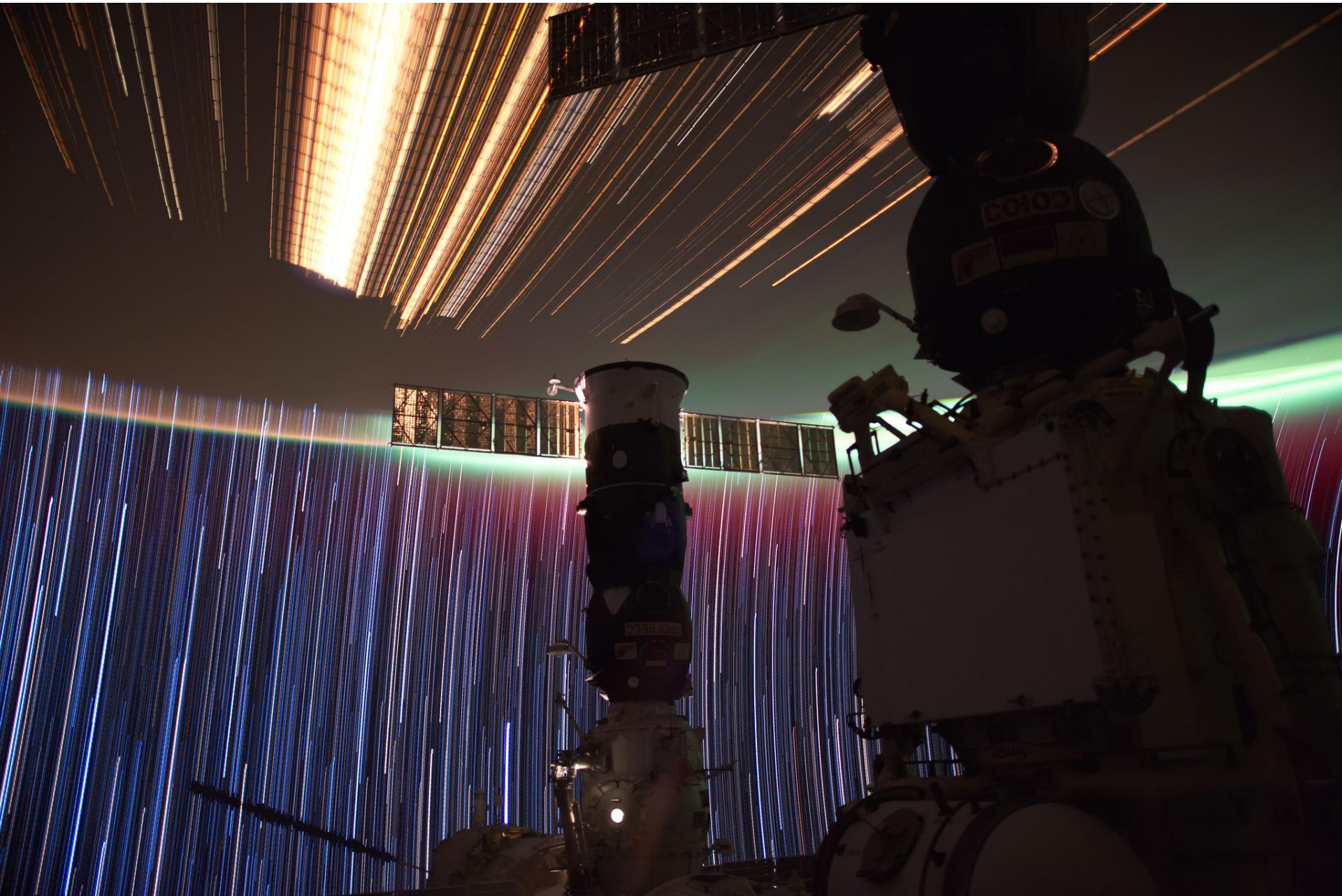
ISO = old film terminology

= sensitivity to light

ISO 200 is twice as sensitive as ISO 100.

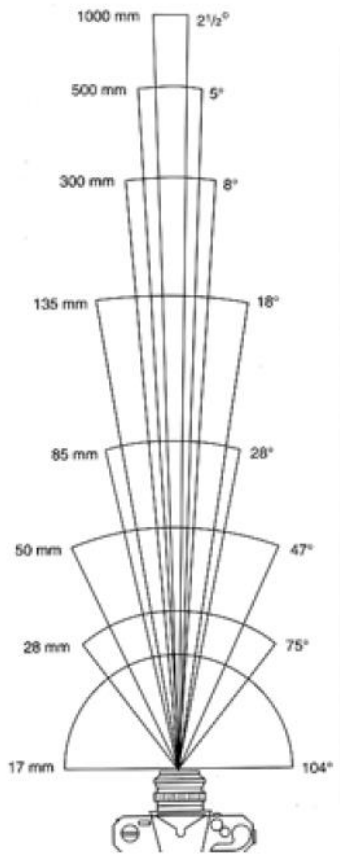
Digital Photography:

ISO = ‘gain’ or amplification of sensor signal



Field of View (Zoom)

Field of View (Zoom)



17mm

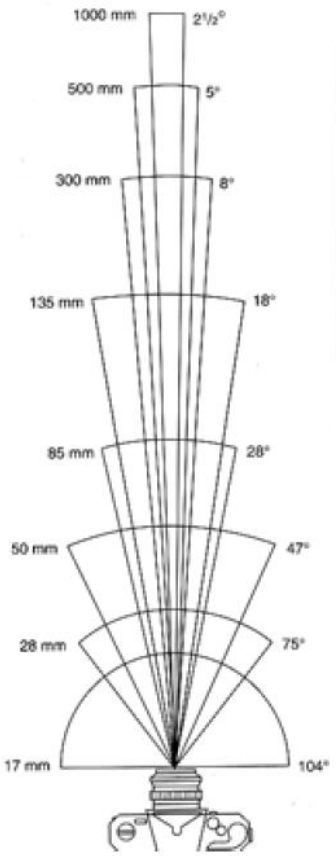
28mm

50mm

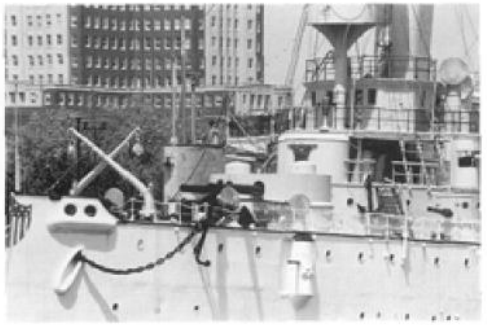
85mm

From London and Upton

Field of View (Zoom) = Cropping



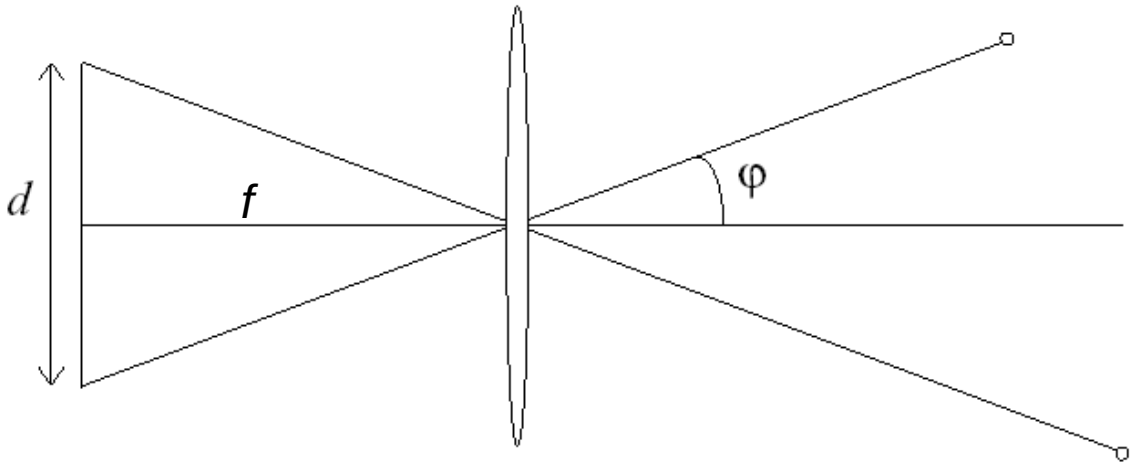
135mm



300mm

From London and Upton

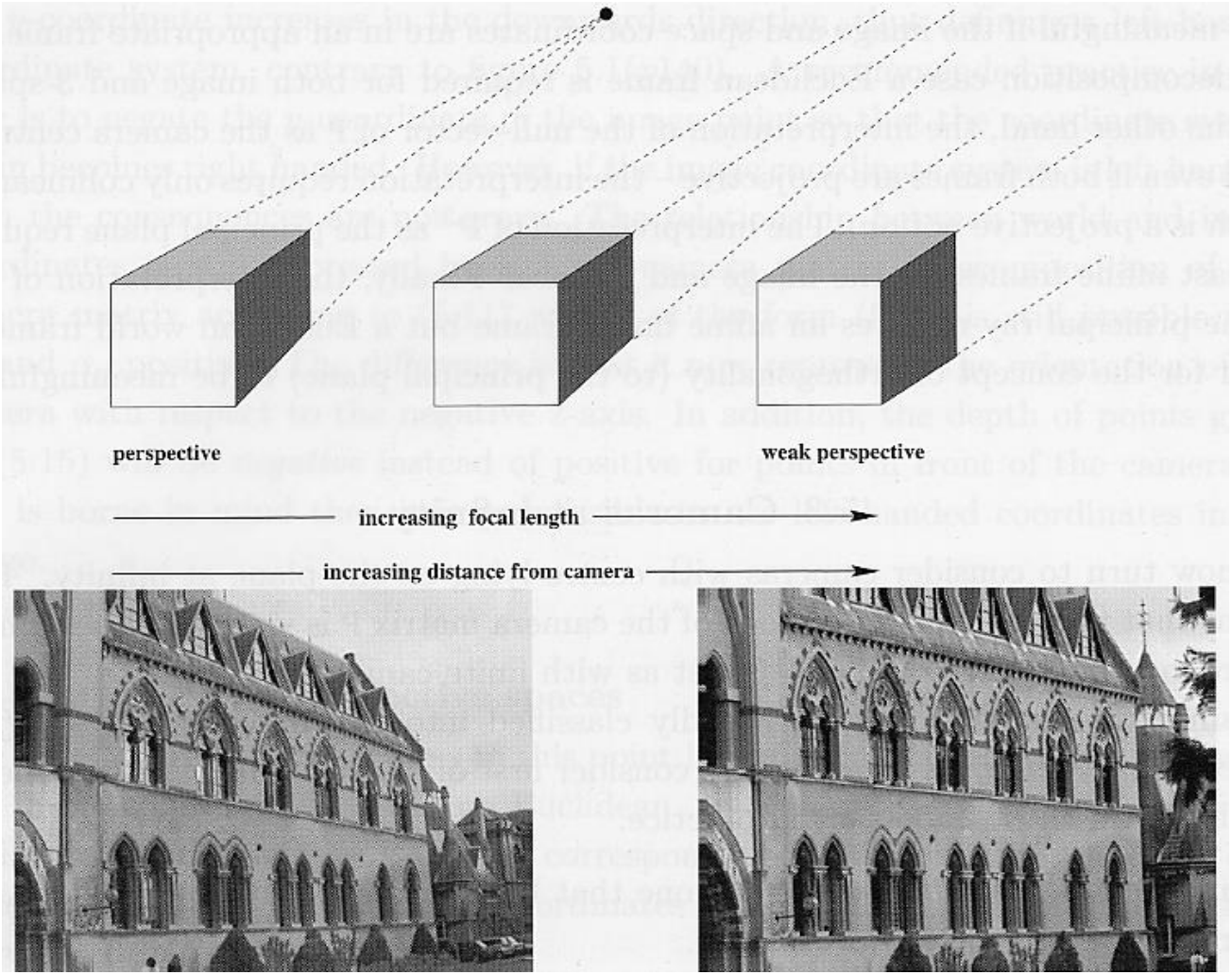
FOV depends of Focal Length



Size of field of view governed by size of the camera retina:

$$\varphi = \tan^{-1}\left(\frac{d}{2f}\right)$$

Smaller FOV = larger Focal Length



From Zisserman & Hartley

Field of View / Focal Length

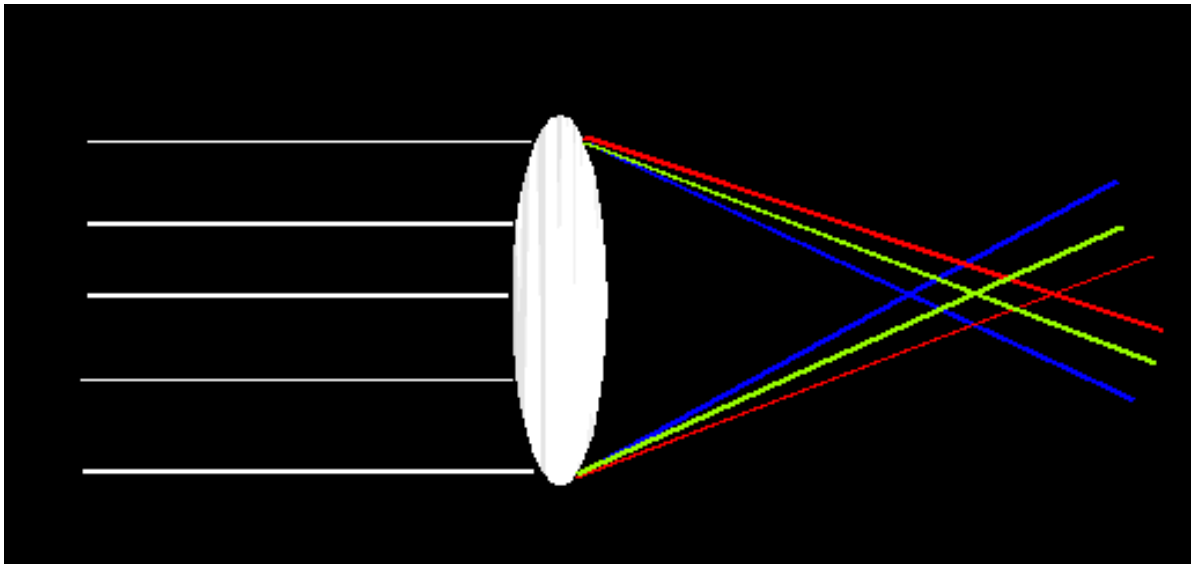
Large FOV, small f
Camera close to car

Small FOV, large f
Camera far from the car

Lens Flaws

Lens Flaws: Chromatic Aberration

- Dispersion: wavelength-dependent refractive index
  - (enables prism to spread white light beam into rainbow)
- Modifies ray-bending and lens focal length: f(λ)



Color fringes near edges of image
Corrections: add ‘doublet’ lens of flint glass, etc.

Chromatic Aberration

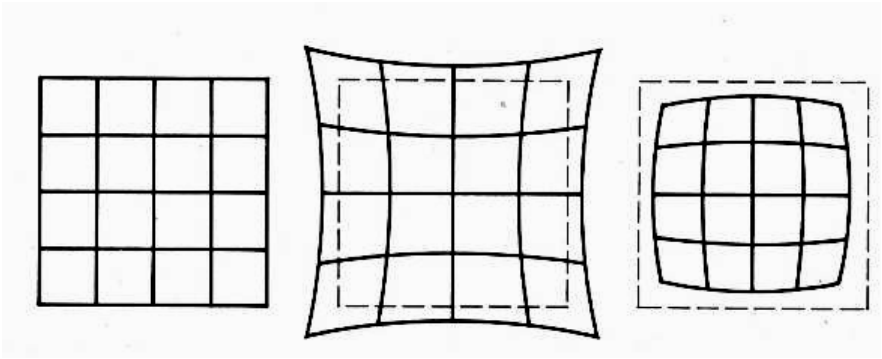
Near Lens Center

Near Lens Outer Edge

Radial Distortion (*e.g.* ‘barrel’ and ‘pin-cushion’)

Straight lines curve around the image center

Radial 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

Corrected Barrel Distortion

Image from Martin Habbecke

Vignetting

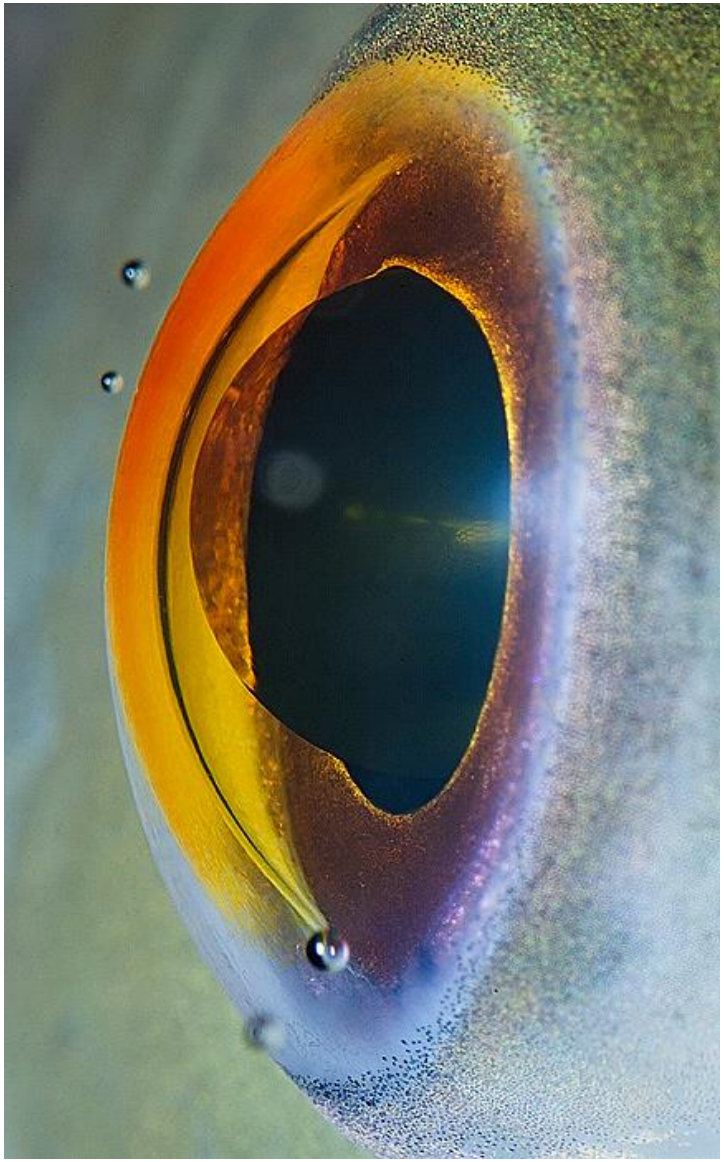
Optical system occludes rays entering at obtuse angles.

Causes darkening at edges.

‘Old mode’ - but WHY?

Computer-aided lens design (optimization) and manufacturing made removing (all) these flaws _much_ easier.

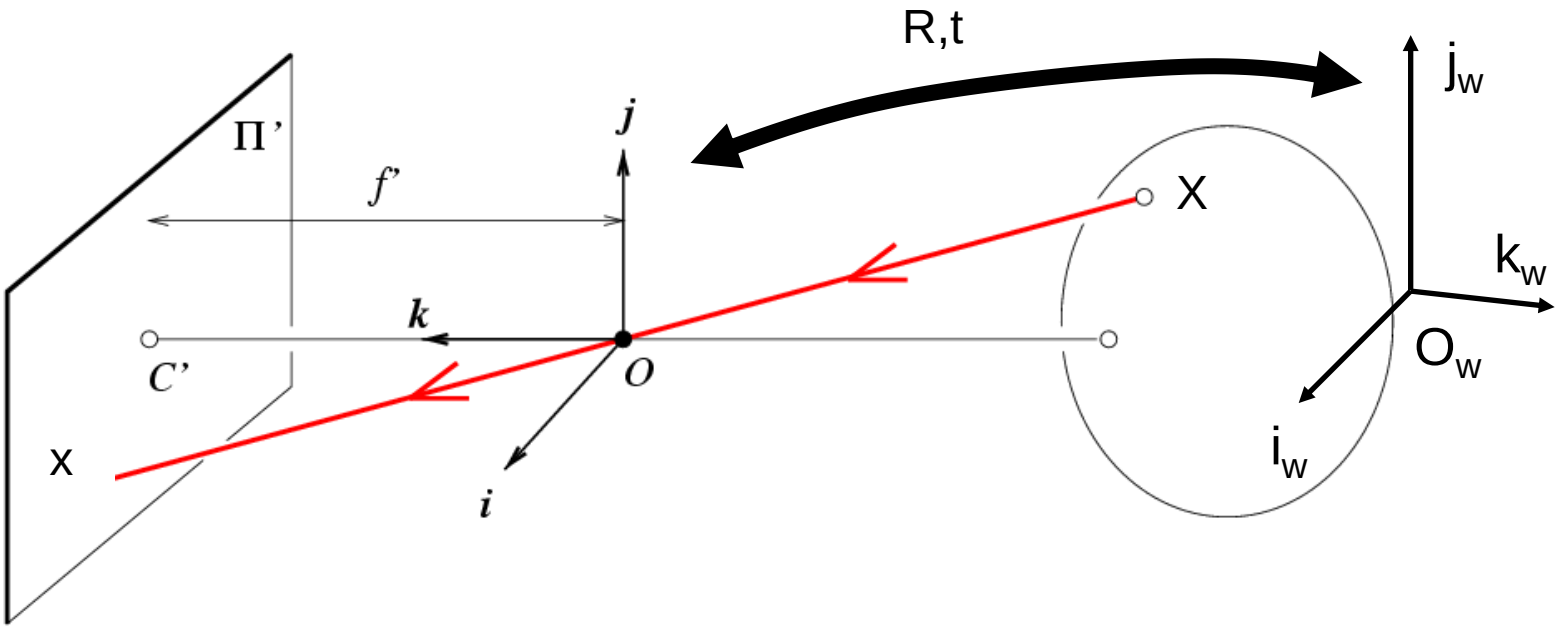
By Suren Manvelyan, http://www.surenmanvelyan.com/gallery/7116



By Suren Manvelyan, http://www.surenmanvelyan.com/gallery/7116

By Suren Manvelyan, http://www.surenmanvelyan.com/gallery/7116

Camera (projection) matrix

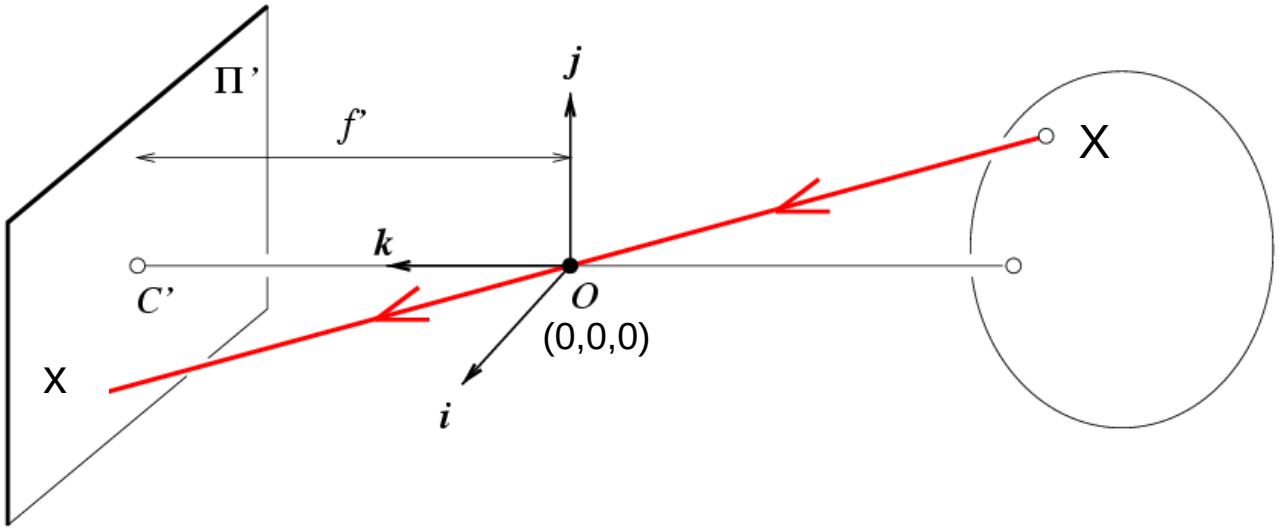


$$\mathbf{x} = \mathbf{K}\begin{bmatrix}\mathbf{R} & \mathbf{t}\end{bmatrix}\mathbf{X}$$

Extrinsic Matrix

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

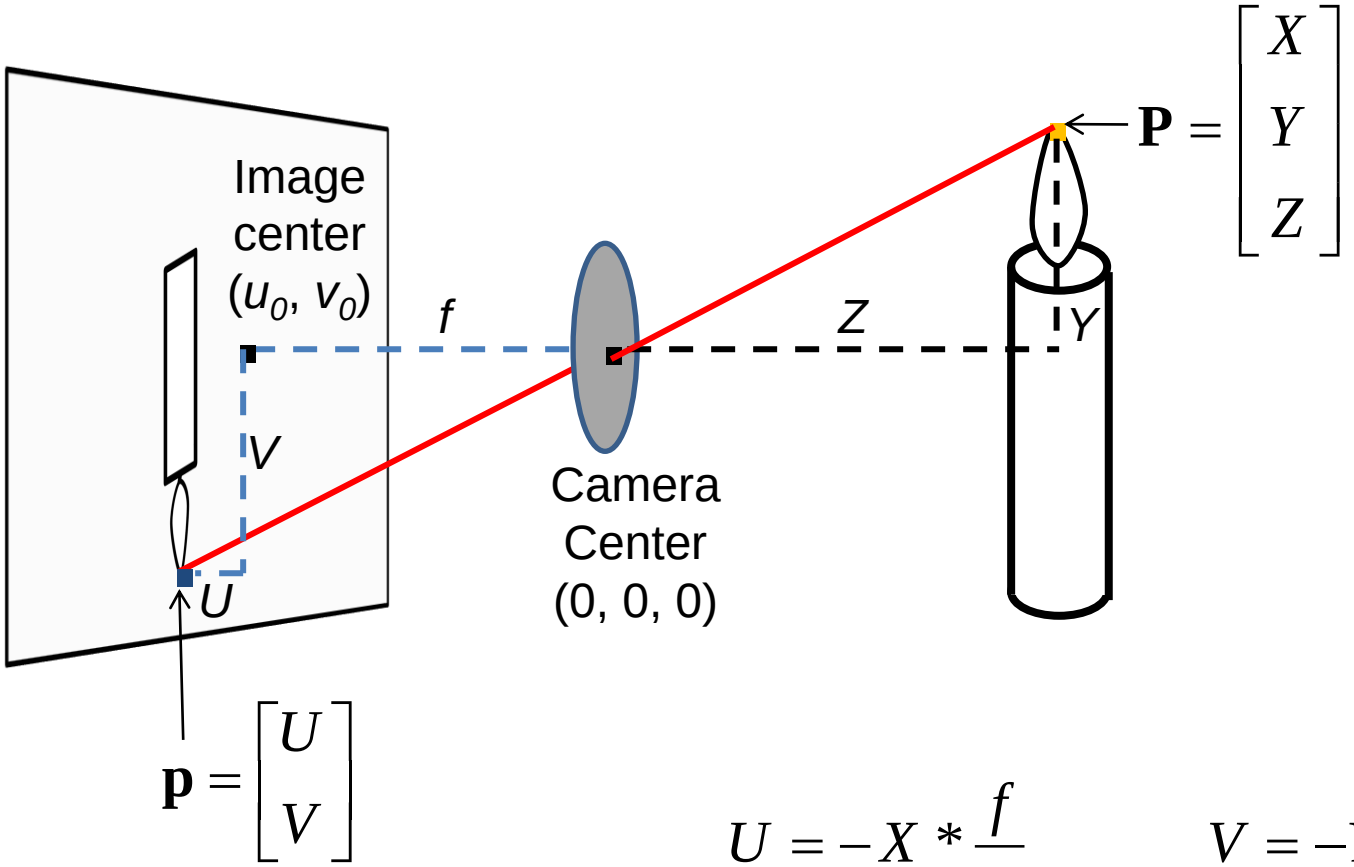
Extrinsic Assumptions

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

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

Slide Credit: Savarese

Projection: world coordinates→image coordinates



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

p = distance from image center

Remove assumption: known optical center

Intrinsic Assumptions
- Unit aspect ratio
- No skew

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

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

K

Remove assumption: equal aspect ratio

Intrinsic Assumptions
- No skew

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

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

Remove assumption: non-skewed pixels

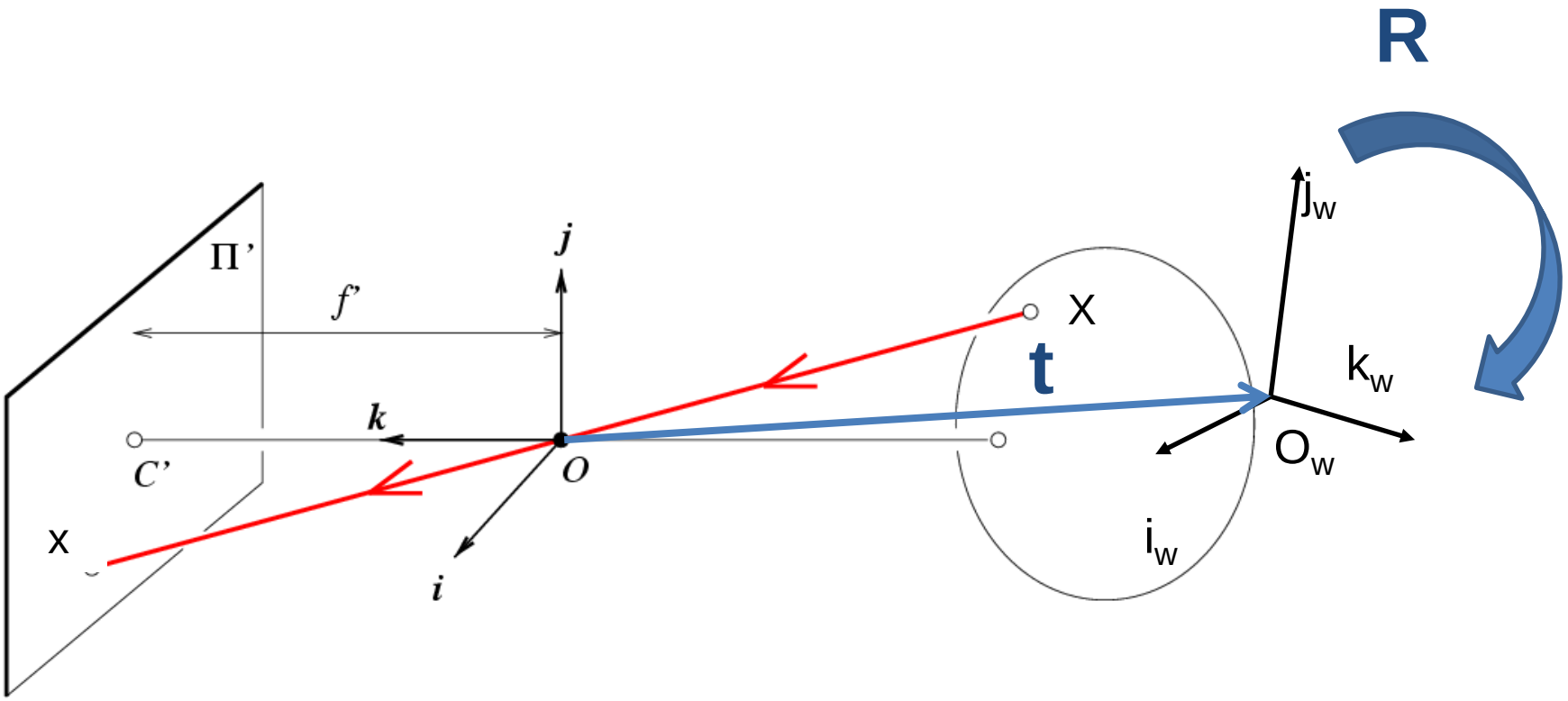
Intrinsic Assumptions

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

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

Note: different books use different notation for parameters

Oriented and Translated Camera



Allow camera translation

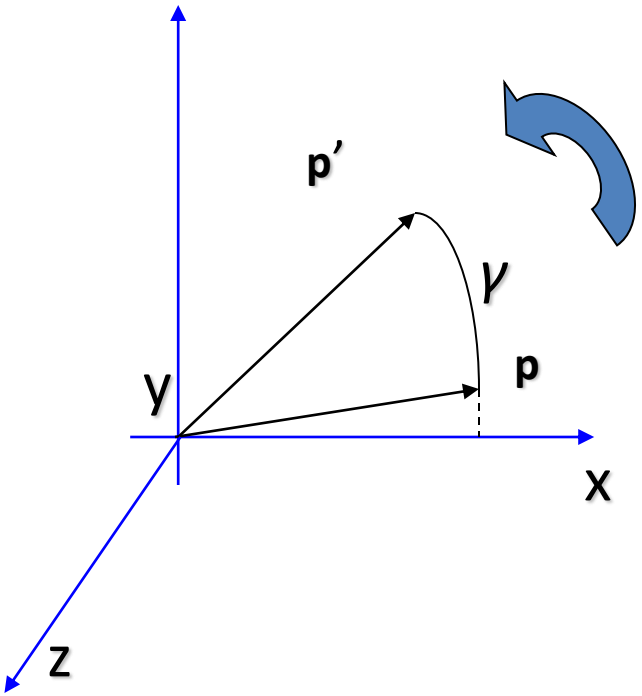
Intrinsic Assumptions

Extrinsic Assumptions
- No rotation

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

3D Rotation of Points

Rotation around the coordinate axes, counter-clockwise:



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

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

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

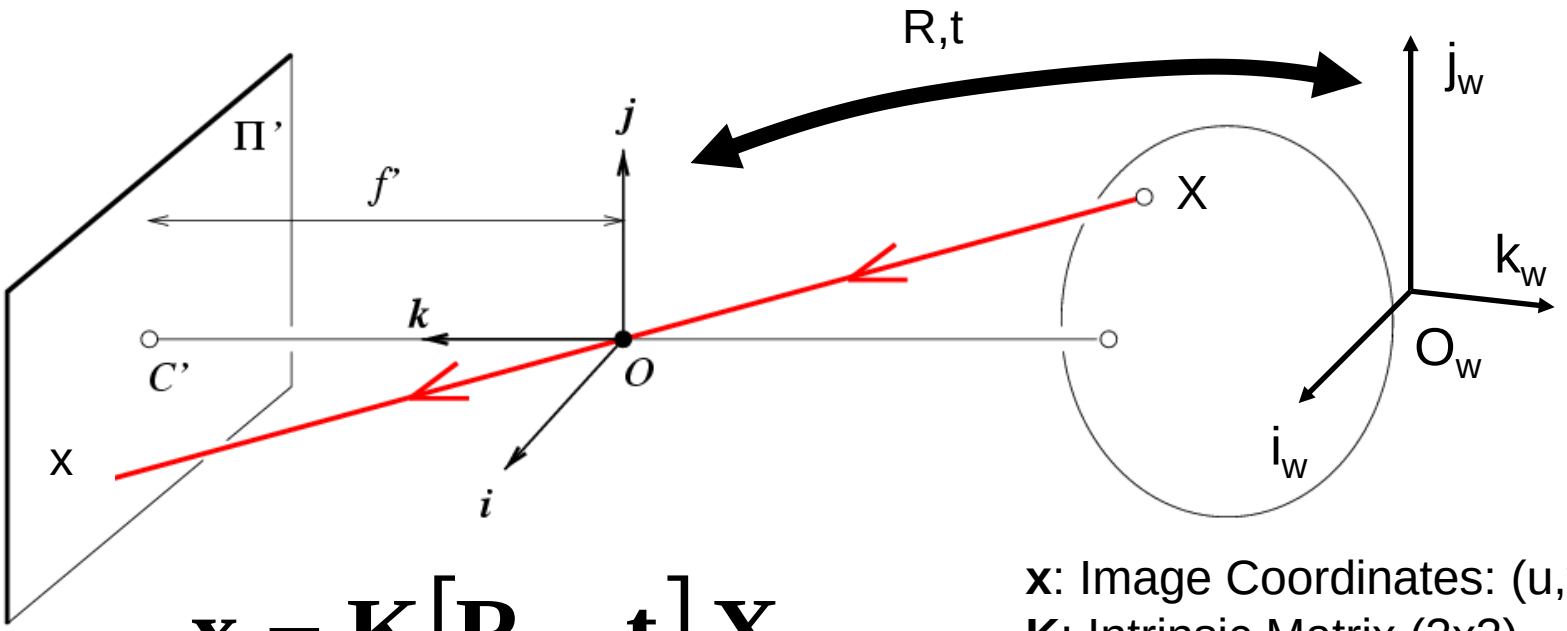
Allow camera rotation

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

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

Slide Credit: Savarese

Camera (projection) matrix



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

Extrinsic Matrix

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

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

Demo – Kyle Simek

“Dissecting the Camera Matrix”

Three-part blog series

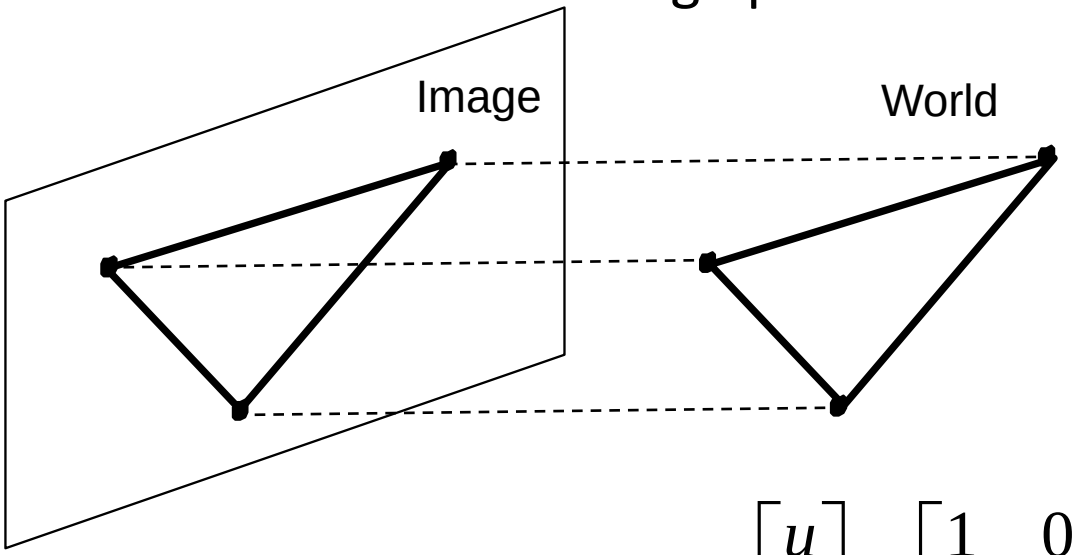
- http://ksimek.github.io/2012/08/14/decompose/
- http://ksimek.github.io/2012/08/22/extrinsic/
- http://ksimek.github.io/2013/08/13/intrinsic/

“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



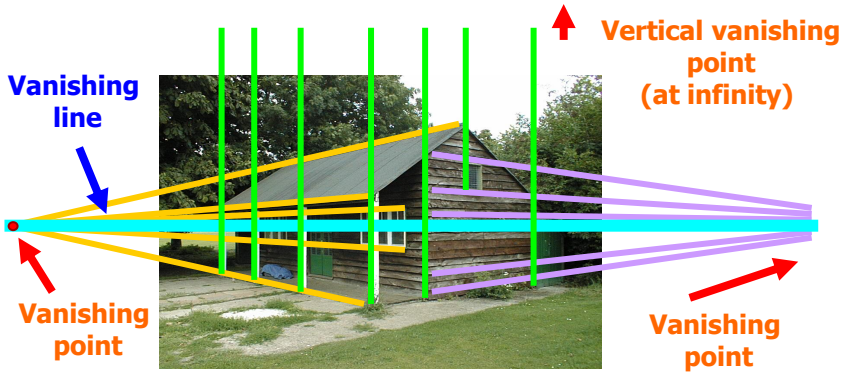
  - Also called “parallel projection”
  - What’s the projection matrix?

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

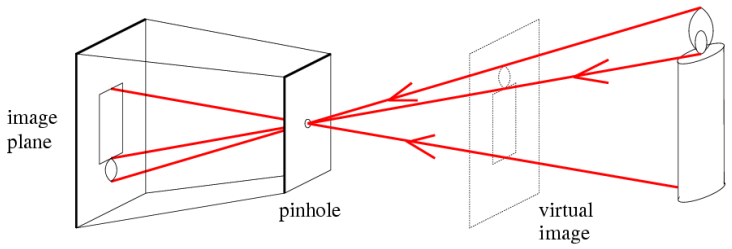
Slide by Steve Seitz

Things to remember

Projection loses length, area, angle, but straight lines remain straight.



Pinhole camera model and camera projection matrix.



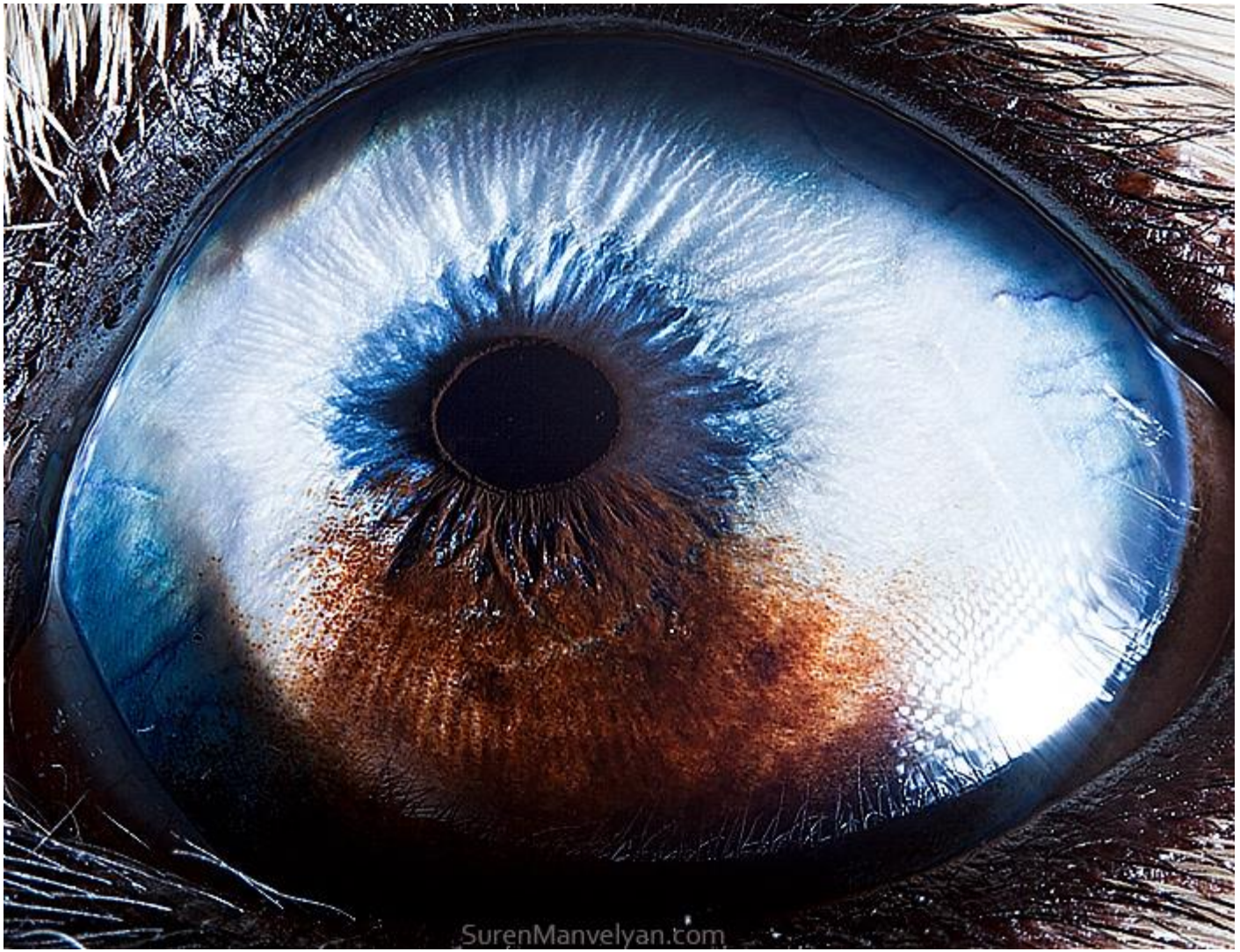
$$\mathbf{x} = \mathbf{K}\begin{bmatrix}\mathbf{R} & \mathbf{t}\end{bmatrix}\mathbf{X}$$

Homogeneous coordinates.

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

By Suren Manvelyan, http://www.surenmanvelyan.com/gallery/7116

By Suren Manvelyan, http://www.surenmanvelyan.com/gallery/7116



By Suren Manvelyan, http://www.surenmanvelyan.com/gallery/7116

[Grad TA Eleanor Tursman’s dog – Blueberry]

Heterochromia iridum

From Wikipedia, the free encyclopedia

Not to be confused with Heterochromatin or Dichromatic (disambiguation).

In anatomy, **heterochromia** (ancient Greek: ἕτερος, *héteros*, different + χρώμα, *chróma*, color[1]) is a difference in coloration, usually of the iris but also of hair or skin. Heterochromia is a result of the relative excess or lack of melanin (a pigment). It may be inherited, or caused by genetic mosaicism, chimerism, disease, or injury.[2]

Heterochromia of the eye (***heterochromia iridis*** or ***heterochromia iridum***) is of three kinds. In *complete heterochromia*, one iris is a different color from the other. In *sectoral heterochromia*, part of one iris is a different color from its remainder and finally in "central heterochromia" there are spikes of different colours radiating from the pupil.

Heterochromia

Complete heterochromia in human eyes: one brown and one green/hazel

Classification and external resources

Specialty	ophthalmology
ICD-10	Q13.2, H20.8, L67.1
ICD-9-CM	364.53
OMIM	142500
DiseasesDB	31289

How to calibrate the camera? (also called “camera resectioning”)

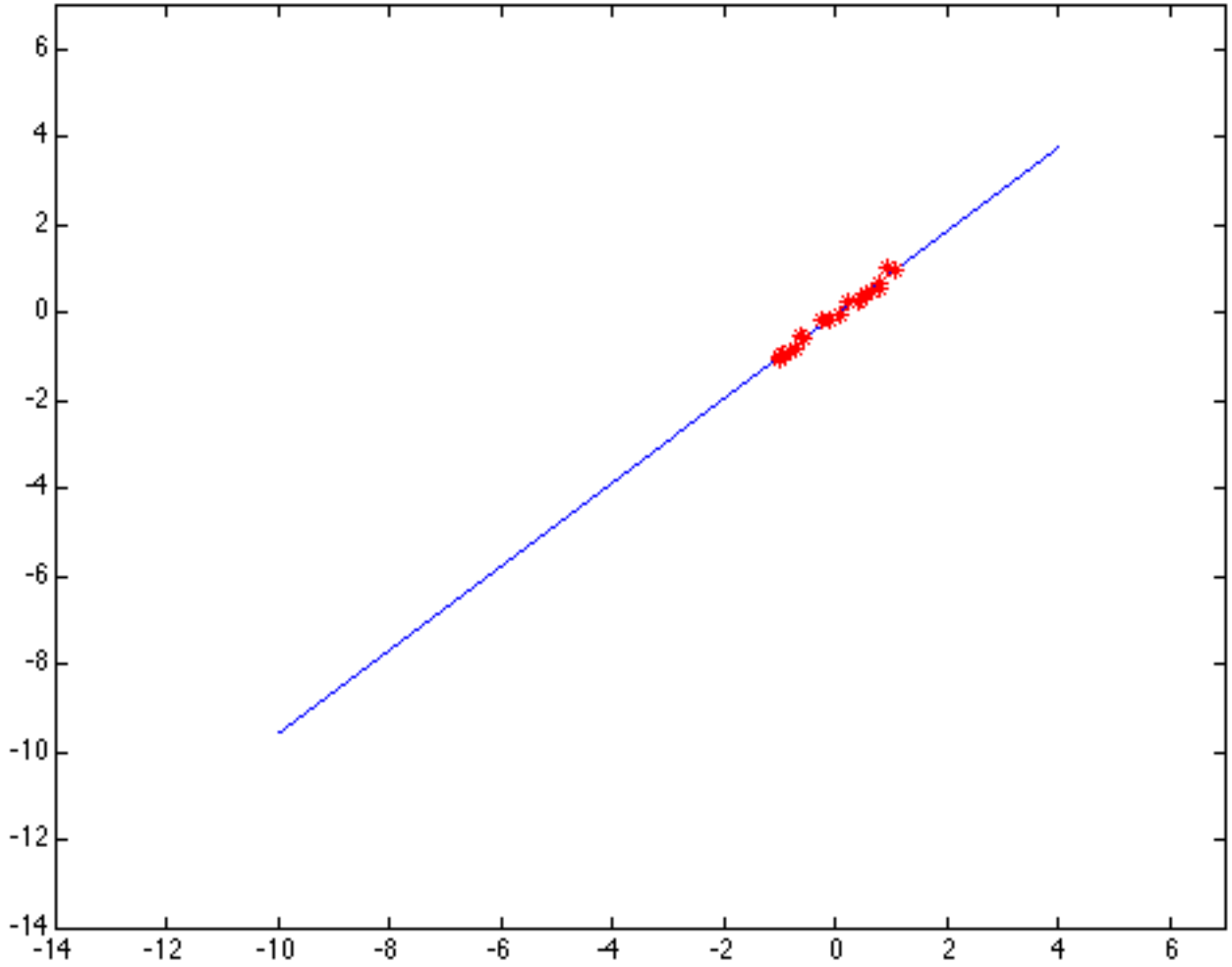
$$\mathbf{x} = \mathbf{K}\begin{bmatrix}\mathbf{R} & \mathbf{t}\end{bmatrix}\mathbf{X}$$

$$\mathbf{x} = \mathbf{M}\mathbf{X}$$

$$\begin{bmatrix} wu \\ wv \\ w \end{bmatrix} = \begin{bmatrix} * & * & * & * \\ * & * & * & * \\ * & * & * & * \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

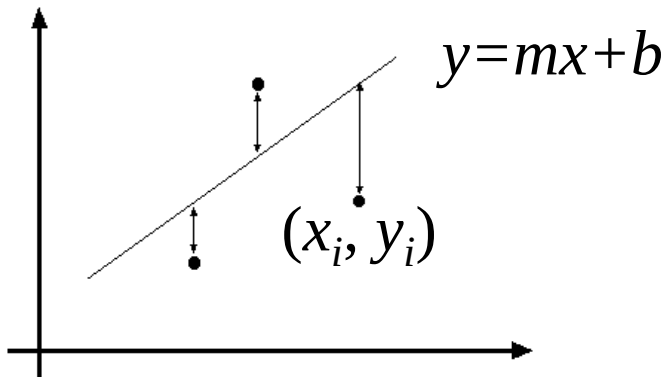
Linear least-squares regression!

Simple example: Fitting a line



Least squares fitting – line model

- Data: $(x_1, y_1), \ldots, (x_n, y_n)$
- Line equation: $y_i = m x_i + b$
- Find (m, b) to minimize



$$E = \sum_{i=1}^{n} (y_i - mx_i - b)^2$$

$$E = \sum_{i=1}^{n} \left(\begin{bmatrix} x_i & 1 \end{bmatrix} \begin{bmatrix} m \\ b \end{bmatrix} - y_i \right)^2 = \left\| \begin{bmatrix} x_1 & 1 \\ \vdots & \vdots \\ x_n & 1 \end{bmatrix} \begin{bmatrix} m \\ b \end{bmatrix} - \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} \right\|^2 = \|\mathbf{Ap} - \mathbf{y}\|^2$$

$$= \mathbf{y}^T\mathbf{y} - 2(\mathbf{Ap})^T\mathbf{y} + (\mathbf{Ap})^T(\mathbf{Ap})$$

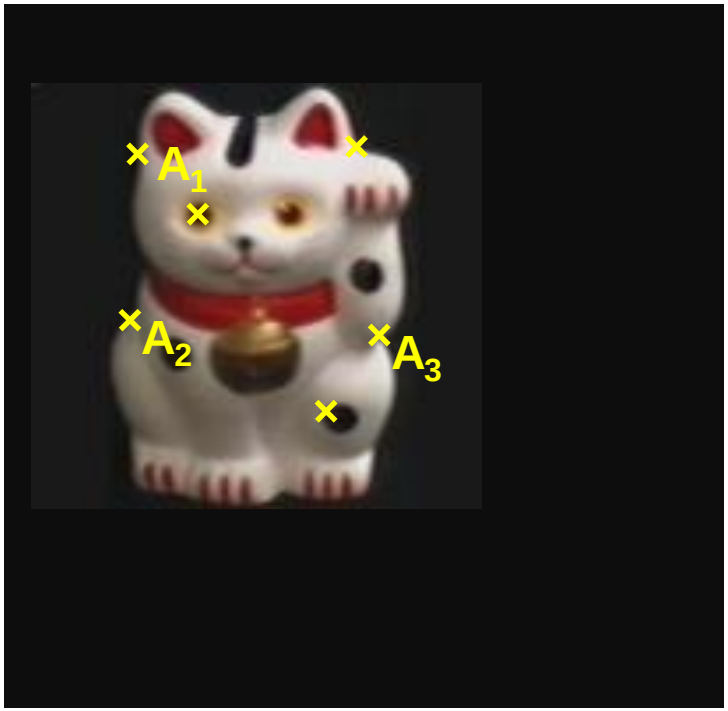
$$\frac{dE}{dp} = 2\mathbf{A}^T\mathbf{Ap} - 2\mathbf{A}^T\mathbf{y} = 0$$

```
Matlab: p = A \ y;
Python:
p = np.linalg.lstsq(A,y)[0]
```

$$\mathbf{A}^T\mathbf{Ap} = \mathbf{A}^T\mathbf{y} \Rightarrow \mathbf{p} = \left(\mathbf{A}^T\mathbf{A}\right)^{-1}\mathbf{A}^T\mathbf{y}$$ (Closed form solution)

Modified from S. Lazebnik

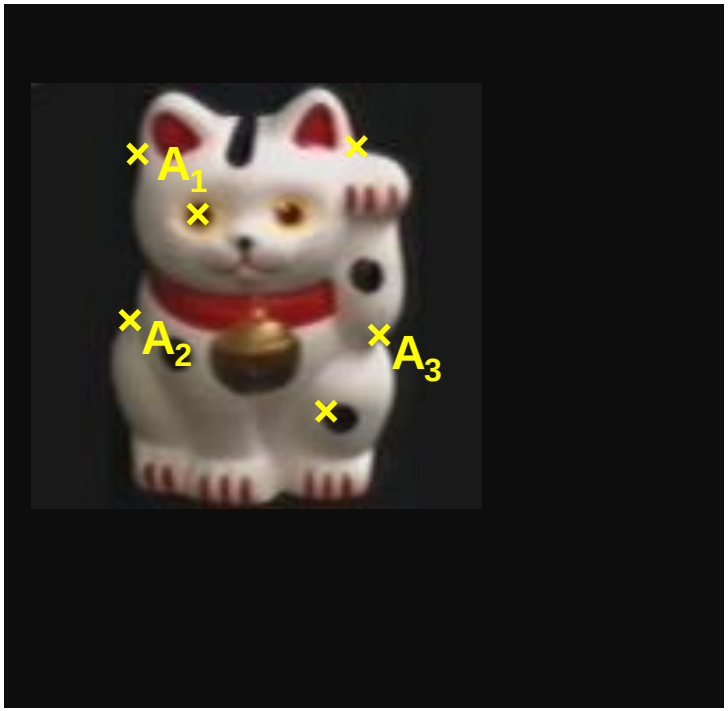
Example: solving for translation



Given matched points in {A} and {B}, estimate the translation of the object

$$\begin{bmatrix} x_i^B \\ y_i^B \end{bmatrix} = \begin{bmatrix} x_i^A \\ y_i^A \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

Example: solving for translation



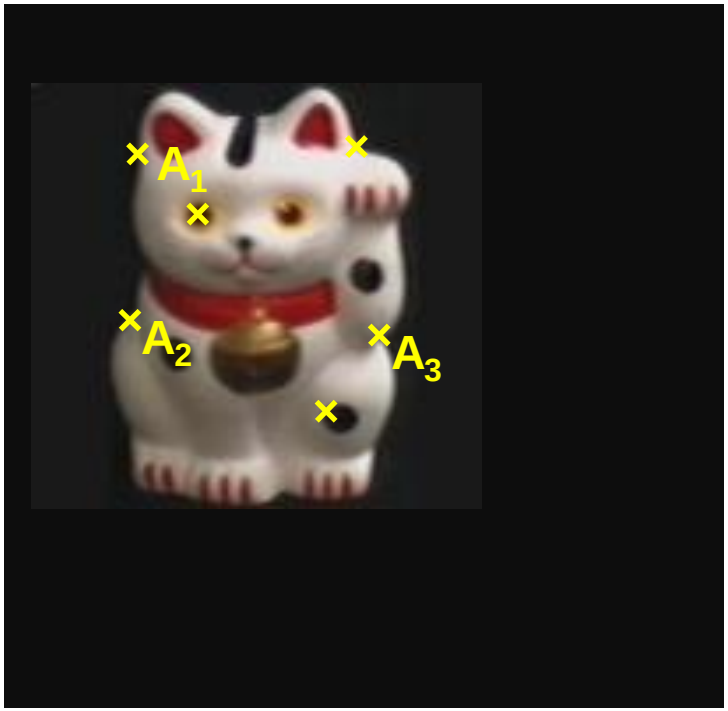
(t_x, t_y)

$$\begin{bmatrix} x_i^B \\ y_i^B \end{bmatrix} = \begin{bmatrix} x_i^A \\ y_i^A \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

Least squares setup

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ \vdots & \vdots \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} t_x \\ t_y \end{bmatrix} = \begin{bmatrix} x_1^B - x_1^A \\ y_1^B - y_1^A \\ \vdots \\ x_n^B - x_n^A \\ y_n^B - y_n^A \end{bmatrix}$$

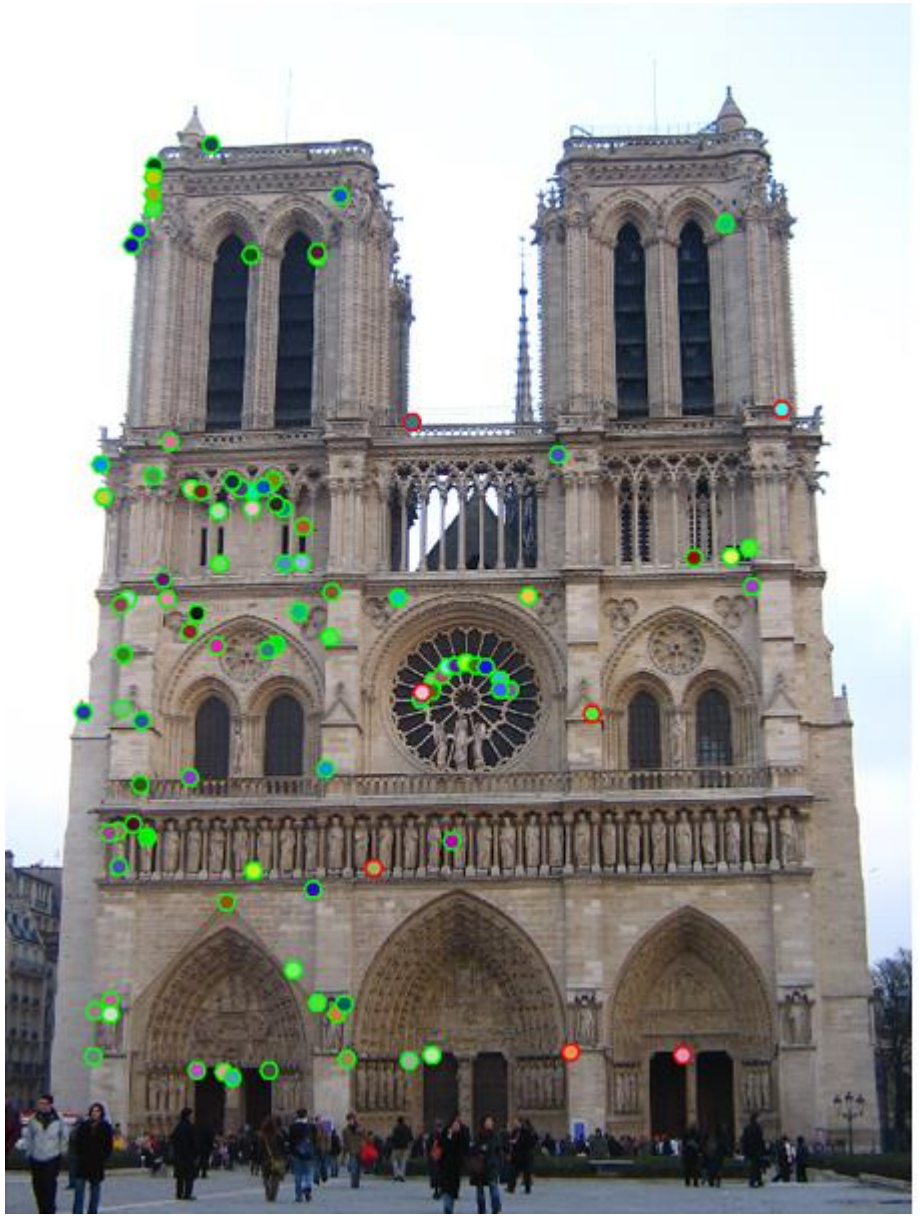
Example: discovering rot/trans/scale

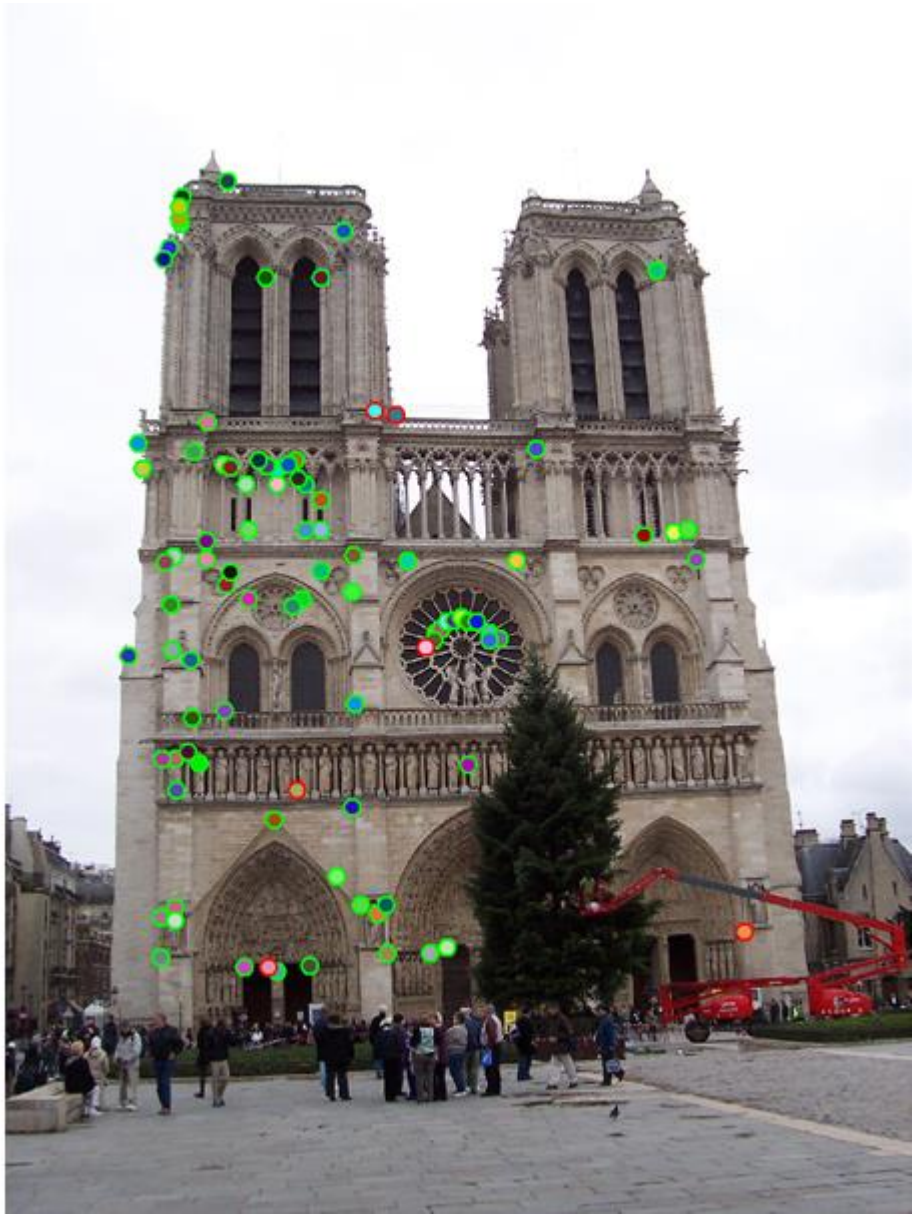


Given matched points in {A} and {B}, estimate the transformation matrix

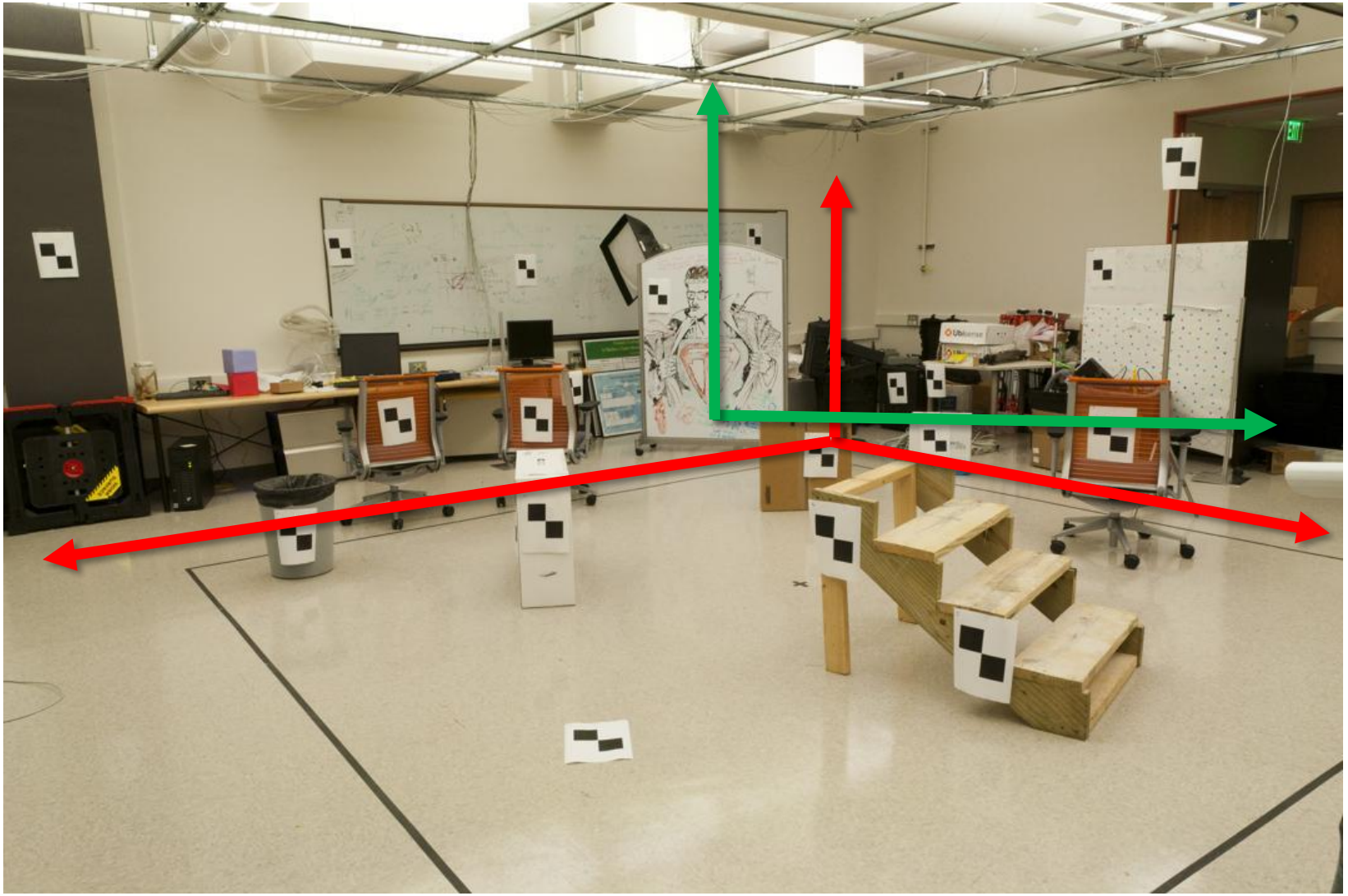
$$\begin{bmatrix} x_i^B \\ y_i^B \end{bmatrix} = \mathrm{T}\begin{bmatrix} x_i^A \\ y_i^A \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix} \qquad \mathrm{T} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

Are these transformations enough?





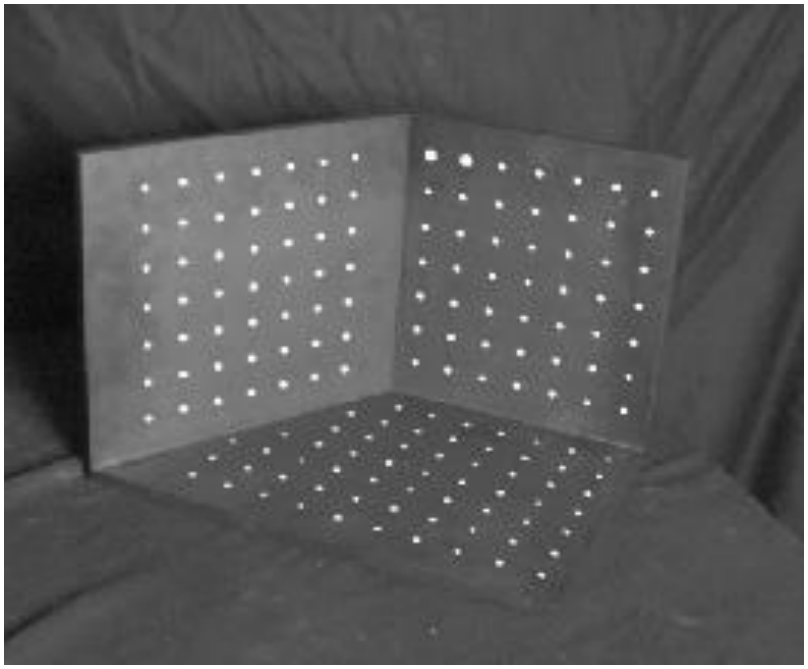
World vs Camera coordinates



Calibrating the Camera

Use an scene with **known** geometry

- Correspond image points to 3d points
- Get least squares solution (or non-linear solution)



Known 2d image coords

Known 3d world locations

$$\begin{bmatrix} su \\ sv \\ s \end{bmatrix} = \mathbf{M} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

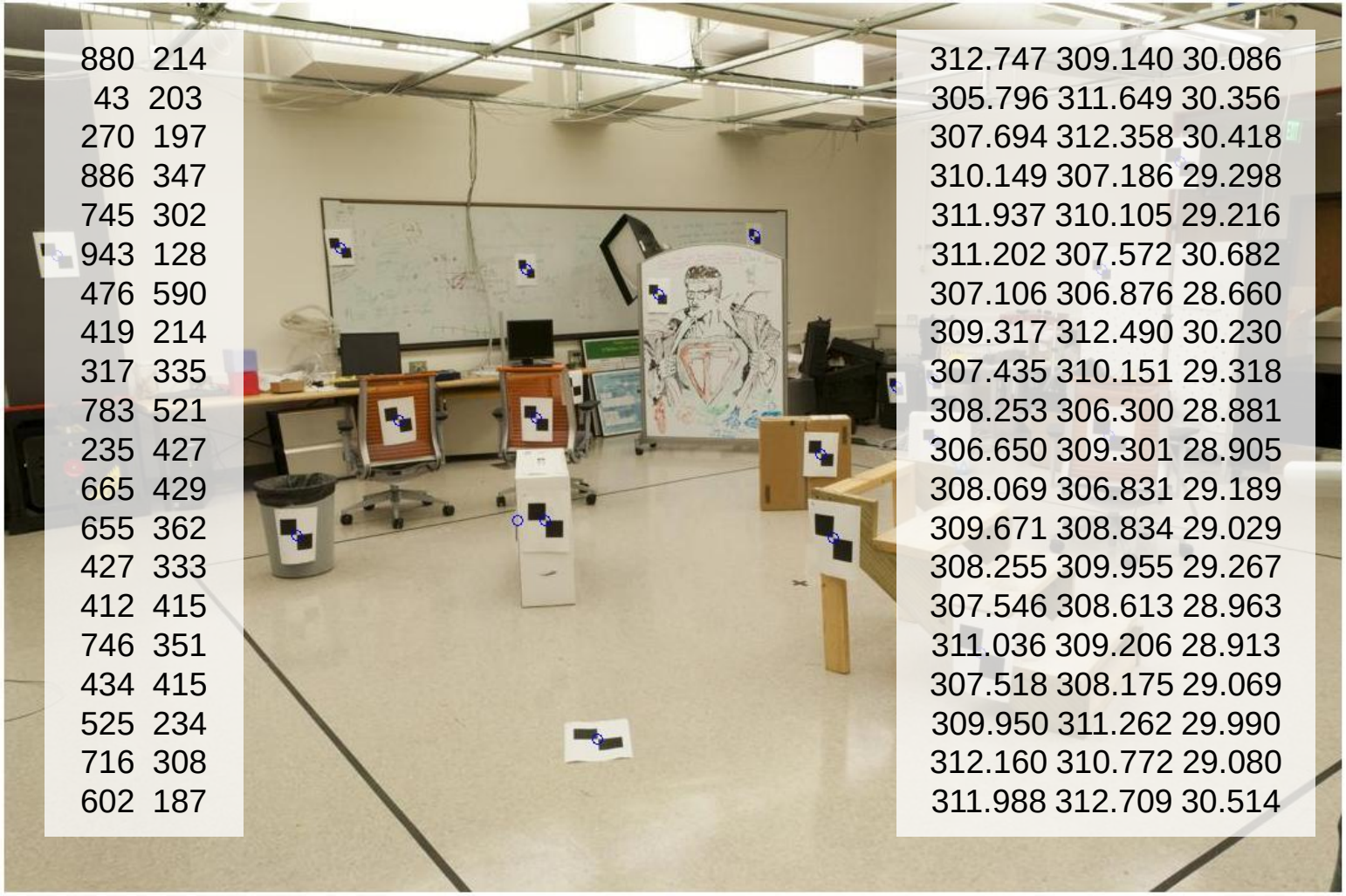
$$\mathbf{M} = \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \end{bmatrix}$$

Unknown Camera Parameters

How do we calibrate a camera?

Known 2d image coords

Known 3d world locations

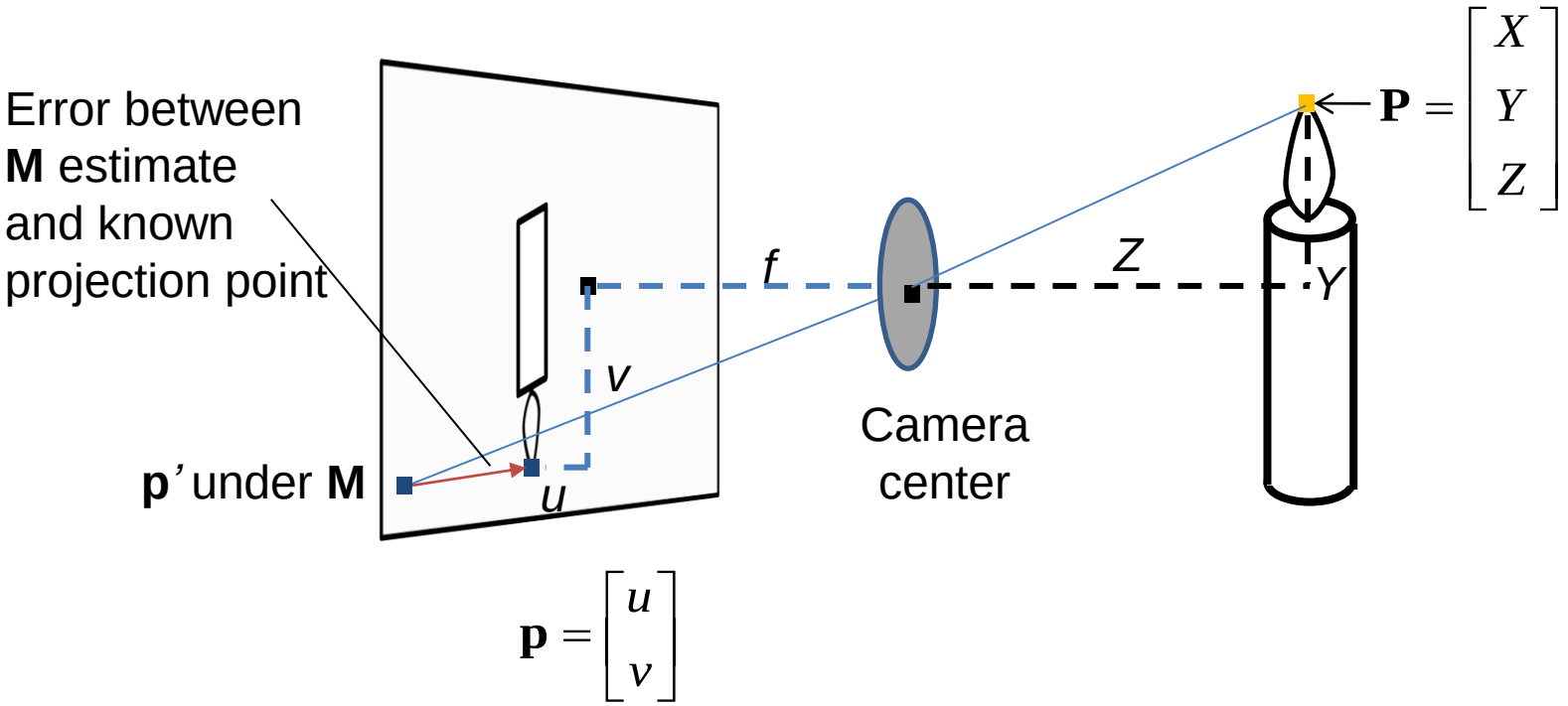


880	214
43	203
270	197
886	347
745	302
943	128
476	590
419	214
317	335
783	521
235	427
665	429
655	362
427	333
412	415
746	351
434	415
525	234
716	308
602	187

312.747	309.140	30.086
305.796	311.649	30.356
307.694	312.358	30.418
310.149	307.186	29.298
311.937	310.105	29.216
311.202	307.572	30.682
307.106	306.876	28.660
309.317	312.490	30.230
307.435	310.151	29.318
308.253	306.300	28.881
306.650	309.301	28.905
308.069	306.831	29.189
309.671	308.834	29.029
308.255	309.955	29.267
307.546	308.613	28.963
311.036	309.206	28.913
307.518	308.175	29.069
309.950	311.262	29.990
312.160	310.772	29.080
311.988	312.709	30.514

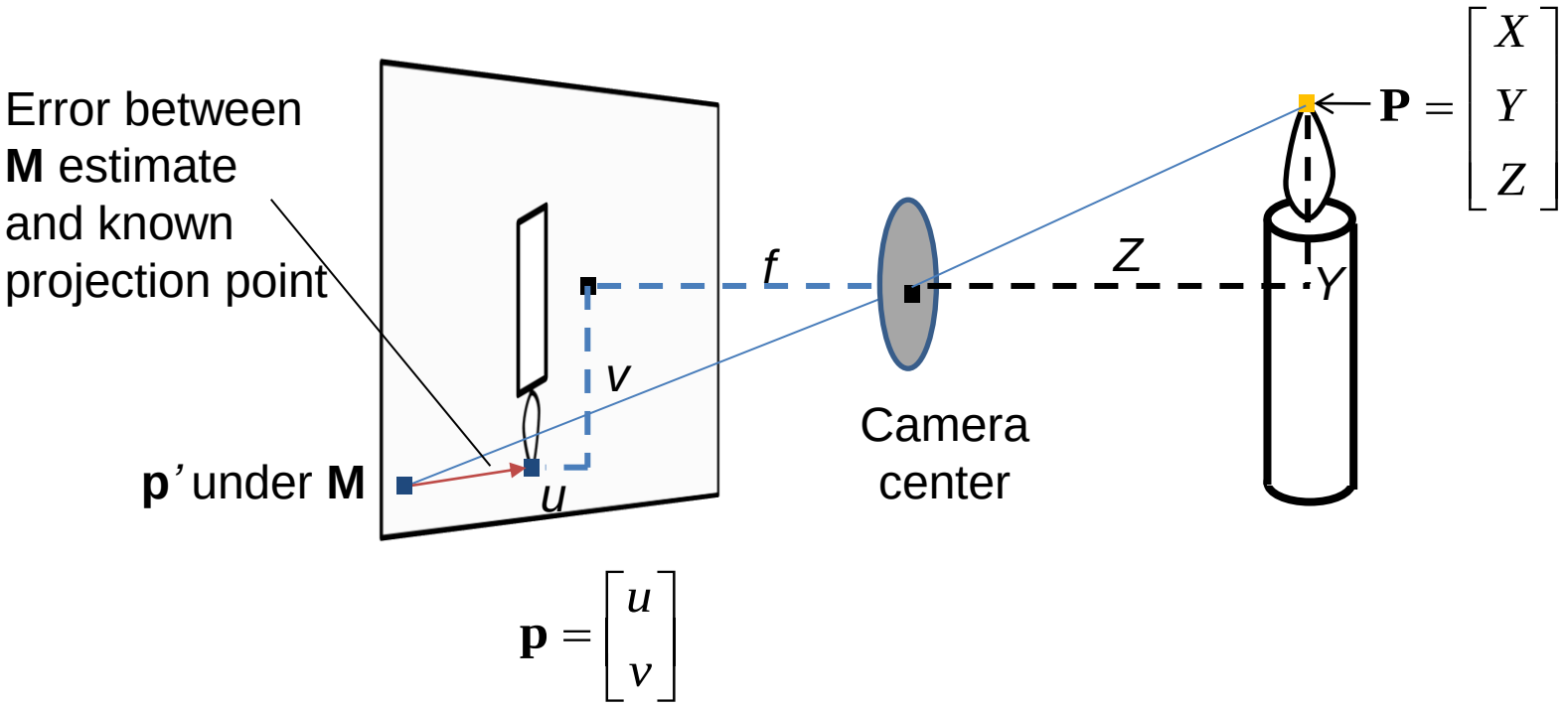
What is least squares doing?

- Given 3D point evidence, find best **M** which minimizes error between estimate (**p**') and known corresponding 2D points (**p**).



What is least squares doing?

- Best **M** occurs when **p'** = **p**, or when **p'** – **p** = 0
- Form these equations from all point evidence
- Solve for model via closed-form regression



Unknown Camera Parameters

Known 2d image coords

$$\begin{bmatrix} su \\ sv \\ s \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Known 3d locations

First, work out where X,Y,Z projects to under candidate **M.**

$$su = m_{11}X + m_{12}Y + m_{13}Z + m_{14}$$

$$sv = m_{21}X + m_{22}Y + m_{23}Z + m_{24}$$

$$s = m_{31}X + m_{32}Y + m_{33}Z + m_{34}$$

Two equations per 3D point correspondence

$$u = \frac{m_{11}X + m_{12}Y + m_{13}Z + m_{14}}{m_{31}X + m_{32}Y + m_{33}Z + m_{34}}$$

$$v = \frac{m_{21}X + m_{22}Y + m_{23}Z + m_{24}}{m_{31}X + m_{32}Y + m_{33}Z + m_{34}}$$

Unknown Camera Parameters

Known 2d image coords

$$\begin{bmatrix} su \\ sv \\ s \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Known 3d locations

Next, rearrange into form where all **M** coefficients are individually stated in terms of X,Y,Z,u,v.
-> Allows us to form lsq matrix.

$$u = \frac{m_{11}X + m_{12}Y + m_{13}Z + m_{14}}{m_{31}X + m_{32}Y + m_{33}Z + m_{34}}$$

$$v = \frac{m_{21}X + m_{22}Y + m_{23}Z + m_{24}}{m_{31}X + m_{32}Y + m_{33}Z + m_{34}}$$

$$(m_{31}X + m_{32}Y + m_{33}Z + m_{34})u = m_{11}X + m_{12}Y + m_{13}Z + m_{14}$$

$$(m_{31}X + m_{32}Y + m_{33}Z + m_{34})v = m_{21}X + m_{22}Y + m_{23}Z + m_{24}$$

$$m_{31}uX + m_{32}uY + m_{33}uZ + m_{34}u = m_{11}X + m_{12}Y + m_{13}Z + m_{14}$$

$$m_{31}vX + m_{32}vY + m_{33}vZ + m_{34}v = m_{21}X + m_{22}Y + m_{23}Z + m_{24}$$

Unknown Camera Parameters

Known 2d image coords

$$\begin{bmatrix} su \\ sv \\ s \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Known 3d locations

Next, rearrange into form where all **M** coefficients are individually stated in terms of X,Y,Z,u,v.
-> Allows us to form lsq matrix.

$$m_{31}uX + m_{32}uY + m_{33}uZ + m_{34}u = m_{11}X + m_{12}Y + m_{13}Z + m_{14}$$
$$m_{31}vX + m_{32}vY + m_{33}vZ + m_{34}v = m_{21}X + m_{22}Y + m_{23}Z + m_{24}$$

$$0 = m_{11}X + m_{12}Y + m_{13}Z + m_{14} - m_{31}uX - m_{32}uY - m_{33}uZ - m_{34}u$$
$$0 = m_{21}X + m_{22}Y + m_{23}Z + m_{24} - m_{31}vX - m_{32}vY - m_{33}vZ - m_{34}v$$

Unknown Camera Parameters

Known 2d image coords

$$\begin{bmatrix} su \\ sv \\ s \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Known 3d locations

- Finally, solve for m's entries using linear least squares
- Method 1 – **Ax**=**b** form

$$\underbrace{\begin{bmatrix} X_1 & Y_1 & Z_1 & 1 & 0 & 0 & 0 & 0 & -u_1X_1 & -u_1Y_1 & -u_1Z_1 \\ 0 & 0 & 0 & 0 & X_1 & Y_1 & Z_1 & 1 & -v_1X_1 & -v_1Y_1 & -v_1Z_1 \\ & & & & & & \vdots & & & & \\ X_n & Y_n & Z_n & 1 & 0 & 0 & 0 & 0 & -u_nX_n & -u_nY_n & -u_nZ_n \\ 0 & 0 & 0 & 0 & X_n & Y_n & Z_n & 1 & -v_nX_n & -v_nY_n & -v_nZ_n \end{bmatrix}}_{\text{A}} \underbrace{\begin{bmatrix} m_{11} \\ m_{12} \\ m_{13} \\ m_{14} \\ m_{21} \\ m_{22} \\ m_{23} \\ m_{24} \\ m_{31} \\ m_{32} \\ m_{33} \end{bmatrix}}_{\text{x}} = \underbrace{\begin{bmatrix} u_1 \\ v_1 \\ \vdots \\ u_n \\ v_n \end{bmatrix}}_{\text{b}}$$

MATLAB:

```
M = A\b;
M = [M;1];
M = reshape(M,[],3)';
```

Python Numpy:

```
M = np.linalg.lstsq(A,b)[0];
M = np.append(M,1)
M = np.reshape(M, (3,4))
```

Note: Must reshape M afterwards!

Unknown Camera Parameters

Known 2d image coords

$$\begin{bmatrix} su \\ sv \\ s \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Known 3d locations

- Or, solve for m's entries using total linear least-squares
- Method 2 – **Ax=0** form
  - Find non-trivial solution (not A=0)

$$\underbrace{\begin{bmatrix} X_1 & Y_1 & Z_1 & 1 & 0 & 0 & 0 & 0 & -u_1X_1 & -u_1Y_1 & -u_1Z_1 & -u_1 \\ 0 & 0 & 0 & 0 & X_1 & Y_1 & Z_1 & 1 & -v_1X_1 & -v_1Y_1 & -v_1Z_1 & -v_1 \\ & & & & & & \vdots & & & & & \\ X_n & Y_n & Z_n & 1 & 0 & 0 & 0 & 0 & -u_nX_n & -u_nY_n & -u_nZ_n & -u_n \\ 0 & 0 & 0 & 0 & X_n & Y_n & Z_n & 1 & -v_nX_n & -v_nY_n & -v_nZ_n & -v_n \end{bmatrix}}_{\text{A}} \underbrace{\begin{bmatrix} m_{11} \\ m_{12} \\ m_{13} \\ m_{14} \\ m_{21} \\ m_{22} \\ m_{23} \\ m_{24} \\ m_{31} \\ m_{32} \\ m_{33} \\ m_{34} \end{bmatrix}}_{\text{x}} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}$$

MATLAB:

```
[U, S, V] = svd(A);
M = V(:,end);
M = reshape(M,[],3)';
```

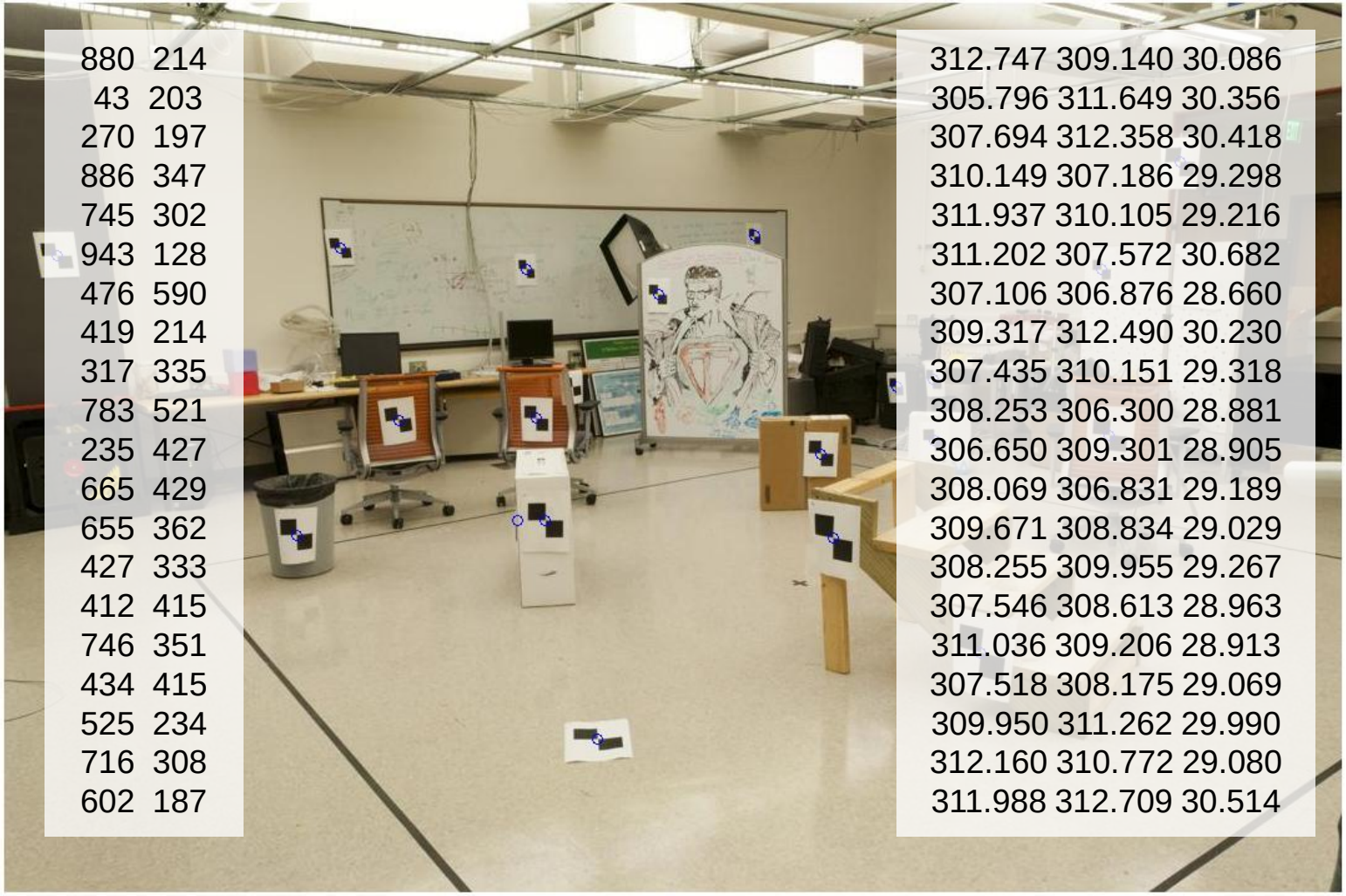
Python Numpy:

```
U, S, Vh = np.linalg.svd(a)
# V = Vh.T
M = Vh[-1,:]
M = np.reshape(M, (3,4))
```

How do we calibrate a camera?

Known 2d image coords

Known 3d world locations



880	214
43	203
270	197
886	347
745	302
943	128
476	590
419	214
317	335
783	521
235	427
665	429
655	362
427	333
412	415
746	351
434	415
525	234
716	308
602	187

312.747	309.140	30.086
305.796	311.649	30.356
307.694	312.358	30.418
310.149	307.186	29.298
311.937	310.105	29.216
311.202	307.572	30.682
307.106	306.876	28.660
309.317	312.490	30.230
307.435	310.151	29.318
308.253	306.300	28.881
306.650	309.301	28.905
308.069	306.831	29.189
309.671	308.834	29.029
308.255	309.955	29.267
307.546	308.613	28.963
311.036	309.206	28.913
307.518	308.175	29.069
309.950	311.262	29.990
312.160	310.772	29.080
311.988	312.709	30.514

Known 2d image coords

Known 3d world locations

1st point

	(u_1, v_1)		(X_1, Y_1, Z_1)
880	214	312.747 309.140 30.086	
43	203	305.796 311.649 30.356	
270	197	307.694 312.358 30.418	
886	347	310.149 307.186 29.298	
745	302	311.937 310.105 29.216	
943	128	311.202 307.572 30.682	
476	590	307.106 306.876 28.660	
419	214	309.317 312.490 30.230	
317	335	307.435 310.151 29.318	
…		…..	

Projection error defined by two equations – one for *u* and one for *v*

$$\begin{bmatrix} 312.747 & 309.140 & 30.086 & 1 & 0 & 0 & 0 & 0 & -880\times 312.747 & -880\times 309.140 & -880\times 30.086 & -880 \\ 0 & 0 & 0 & 0 & 312.747 & 309.140 & 30.086 & 1 & -214\times 312.747 & -214\times 309.140 & -214\times 30.086 & -214 \\ & & & & & & \vdots & & & & & \\ X_n & Y_n & Z_n & 1 & 0 & 0 & 0 & 0 & -u_n X_n & -u_n Y_n & -u_n Z_n & -u_n \\ 0 & 0 & 0 & 0 & X_n & Y_n & Z_n & 1 & -v_n X_n & -v_n Y_n & -v_n Z_n & -v_n \end{bmatrix} \begin{bmatrix} m_{11} \\ m_{12} \\ m_{13} \\ m_{14} \\ m_{21} \\ m_{22} \\ m_{23} \\ m_{24} \\ m_{31} \\ m_{32} \\ m_{33} \\ m_{34} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}$$

Known 2d image coords

Known 3d world locations

2nd point

Projection error defined by two equations – one for *u* and one for *v*

$$\begin{bmatrix} 312.747 & 309.140 & 30.086 & 1 & 0 & 0 & 0 & 0 & -880\times312.747 & -880\times309.140 & -880\times30.086 & -880 \\ 0 & 0 & 0 & 0 & 312.747 & 309.140 & 30.086 & 1 & -214\times312.747 & -214\times309.140 & -214\times30.086 & -214 \\ 305.796 & 311.649 & 30.356 & 1 & 0 & 0 & 0 & 0 & -43\times305.796 & -43\times311.649 & -43\times30.356 & -43 \\ 0 & 0 & 0 & 0 & 305.796 & 311.649 & 30.356 & 1 & -203\times305.796 & -203\times311.649 & -43\times30.356 & -203 \\ & & & & & & \vdots & & & & & \\ X_n & Y_n & Z_n & 1_n & 0 & 0 & 0 & 0 & -u_nX_n & -u_nY_n & -u_nZ_n & -u_n \\ 0 & 0 & 0 & 0 & X_n & Y_n & Z_n & 1 & -v_nX_n & -v_nY_n & -v_nZ_n & -v_n \end{bmatrix} \begin{bmatrix} m_{11} \\ m_{12} \\ m_{13} \\ m_{14} \\ m_{21} \\ m_{22} \\ m_{23} \\ m_{24} \\ m_{31} \\ m_{32} \\ m_{33} \\ m_{34} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}$$

How many points do I need to fit the model?

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

Degrees of freedom? 5 6

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

Think 3:

- Rotation around x
- Rotation around y
- Rotation around z

How many points do I need to fit the model?

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

Degrees of freedom? 5 6

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

M is 3x4, so 12 unknowns, but projective scale ambiguity – 11 deg. freedom.
One equation per unknown -> 5 1/2 point correspondences determines a solution (e.g., either u or v).

More than 5 1/2 point correspondences -> overdetermined, many solutions to **M**.
Least squares is finding the solution that best satisfies the overdetermined system.

Why use more than 6? Robustness to error in feature points.

Calibration with linear method

- Advantages
  - Easy to formulate and solve
  - Provides initialization for non-linear methods

- Disadvantages
  - Doesn’t directly give you camera parameters
  - Doesn’t model radial distortion
  - Can’t impose constraints, such as known focal length

- Non-linear methods are preferred
  - Define error as difference between projected points and measured points
  - Minimize error using Newton’s method or other non-linear optimization

Can we factorize M back to K [R | T]?

- Yes! We can directly solve for the individual entries of K [R | T].

$\mathbf{a}_n$ = nth column of A

Extracting camera parameters

$$\frac{M}{\rho} = \begin{pmatrix} \alpha \boldsymbol{r}_1^T - \alpha \cot\theta \boldsymbol{r}_2^T + u_0 \boldsymbol{r}_3^T & \alpha t_x - \alpha \cot\theta t_y + u_0 t_z \\ \frac{\beta}{\sin\theta} \boldsymbol{r}_2^T + v_0 \boldsymbol{r}_3^T & \frac{\beta}{\sin\theta} t_y + v_0 t_z \\ \boldsymbol{r}_3^T & t_z \end{pmatrix} = \mathrm{K}\begin{bmatrix} \mathrm{R} & \mathrm{T} \end{bmatrix}$$

A $\quad$ $\mathbf{b}$

$$\mathrm{K} = \begin{bmatrix} \alpha & -\alpha\cot\theta & u_o \\ 0 & \frac{\beta}{\sin\theta} & v_o \\ 0 & 0 & 1 \end{bmatrix}$$

Box 1

$$\mathrm{A} = \begin{bmatrix} \mathbf{a}_1^{\mathrm{T}} \\ \mathbf{a}_2^{\mathrm{T}} \\ \mathbf{a}_3^{\mathrm{T}} \end{bmatrix} \qquad \mathrm{b} = \begin{bmatrix} \mathrm{b}_1 \\ \mathrm{b}_2 \\ \mathrm{b}_3 \end{bmatrix}$$

Estimated values

Intrinsic

$$\rho = \frac{\pm 1}{|\mathbf{a}_3|} \qquad u_o = \rho^2(\mathbf{a}_1 \cdot \mathbf{a}_3) \qquad \mathrm{v}_o = \rho^2(\mathbf{a}_2 \cdot \mathbf{a}_3)$$

$$\cos\theta = \frac{(\mathbf{a}_1 \times \mathbf{a}_3)\cdot(\mathbf{a}_2 \times \mathbf{a}_3)}{|\mathbf{a}_1 \times \mathbf{a}_3| \cdot |\mathbf{a}_2 \times \mathbf{a}_3|}$$

Extracting camera parameters

$$\frac{\mathcal{M}}{\rho} = \begin{pmatrix} \alpha \boldsymbol{r}_1^T - \alpha \cot\theta \boldsymbol{r}_2^T + u_0 \boldsymbol{r}_3^T & \alpha t_x - \alpha \cot\theta t_y + u_0 t_z \\ \frac{\beta}{\sin\theta} \boldsymbol{r}_2^T + v_0 \boldsymbol{r}_3^T & \frac{\beta}{\sin\theta} t_y + v_0 t_z \\ \boldsymbol{r}_3^T & t_z \end{pmatrix} = \mathrm{K}\begin{bmatrix}\mathrm{R} & \mathrm{T}\end{bmatrix}$$

A **b**

$$\mathrm{A} = \begin{bmatrix} \mathbf{a}_1^{\mathrm{T}} \\ \mathbf{a}_2^{\mathrm{T}} \\ \mathbf{a}_3^{\mathrm{T}} \end{bmatrix} \qquad \mathrm{b} = \begin{bmatrix} \mathrm{b}_1 \\ \mathrm{b}_2 \\ \mathrm{b}_3 \end{bmatrix}$$

Estimated values

Intrinsic

$$\alpha = \rho^2 |\mathbf{a}_1 \times \mathbf{a}_3| \sin\theta$$

$$\beta = \rho^2 |\mathbf{a}_2 \times \mathbf{a}_3| \sin\theta$$

Extracting camera parameters

$$\frac{\mathcal{M}}{\rho} = \begin{pmatrix} \alpha \boldsymbol{r}_1^T - \alpha \cot\theta \boldsymbol{r}_2^T + u_0 \boldsymbol{r}_3^T & \alpha t_x - \alpha \cot\theta t_y + u_0 t_z \\ \frac{\beta}{\sin\theta} \boldsymbol{r}_2^T + v_0 \boldsymbol{r}_3^T & \frac{\beta}{\sin\theta} t_y + v_0 t_z \\ \boldsymbol{r}_3^T & t_z \end{pmatrix} = \mathrm{K}\begin{bmatrix}\mathrm{R} & \mathrm{T}\end{bmatrix}$$

A b

$$\mathrm{A} = \begin{bmatrix} \mathbf{a}_1^{\mathrm{T}} \\ \mathbf{a}_2^{\mathrm{T}} \\ \mathbf{a}_3^{\mathrm{T}} \end{bmatrix} \qquad \mathrm{b} = \begin{bmatrix} \mathrm{b}_1 \\ \mathrm{b}_2 \\ \mathrm{b}_3 \end{bmatrix}$$

Estimated values

Extrinsic

$$\mathbf{r}_1 = \frac{(\mathbf{a}_2 \times \mathbf{a}_3)}{|\mathbf{a}_2 \times \mathbf{a}_3|} \qquad \mathbf{r}_3 = \frac{\pm \mathbf{a}_3}{|\mathbf{a}_3|}$$

$$\mathbf{r}_2 = \mathbf{r}_3 \times \mathbf{r}_1 \qquad \mathrm{T} = \rho\, \mathrm{K}^{-1}\mathbf{b}$$

Can we factorize M back to K [R | T]?

- Yes! We can also use *RQ* factorization (not QR)
  - R in RQ is not rotation matrix R; crossed names!

R (upper triangular or 'Right' triangular) is K

Q (orthogonal basis) is R

T, the last column of [R | T], is inv(K) * last column of M.

  - But you need to do a bit of post-processing to make sure that the matrices are valid. See http://ksimek.github.io/2012/08/14/decompose/

Recovering the camera center

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

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

t

This is not the camera center C.
It is –RC, as the point is rotated before t_x, t_y, and t_z are added

So we need
$-R^{-1} K^{-1} m_4$ to get C.

$$\begin{bmatrix}su\\sv\\s\end{bmatrix} = \begin{bmatrix}* & * & * & *\\* & * & * & *\\* & * & * & *\end{bmatrix}\begin{bmatrix}X\\Y\\Z\\1\end{bmatrix}$$

m_4

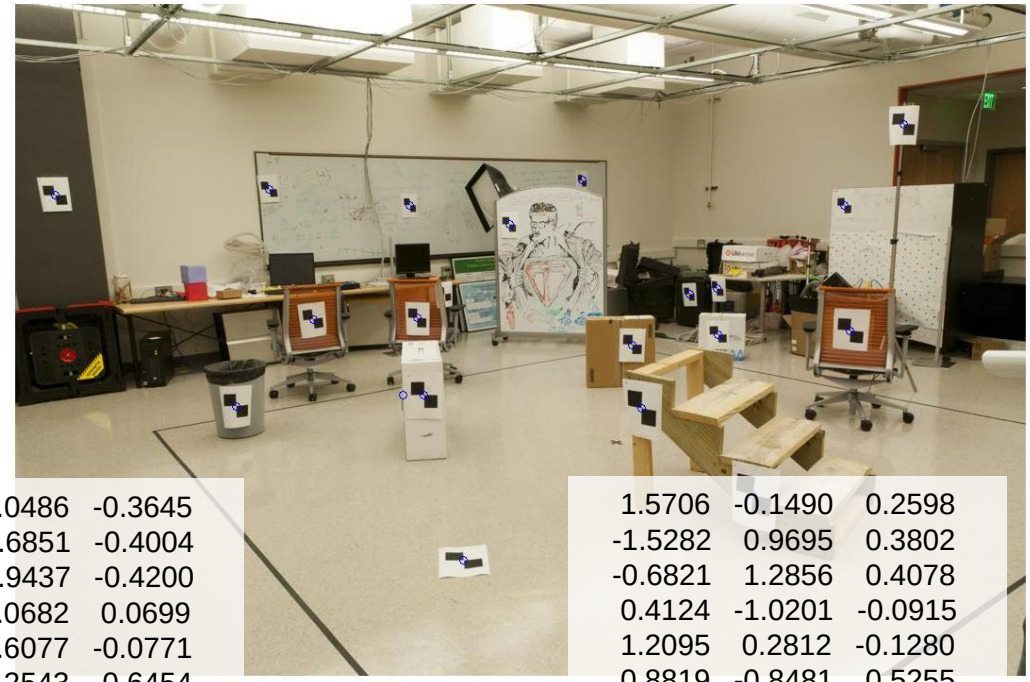
Q

This is **t** × K

So $K^{-1} m_4$ is **t**

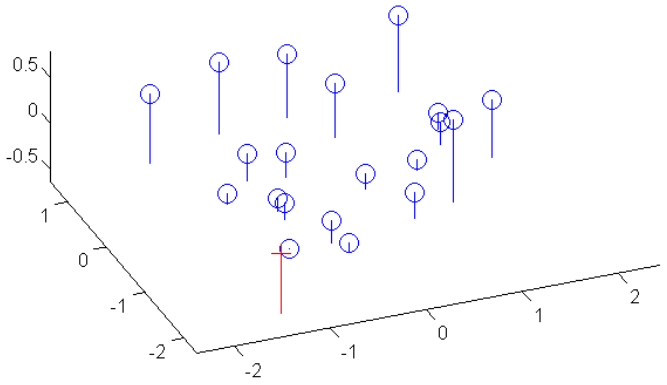
Q is K × R.
So we just need $-Q^{-1} m_4$

Estimate of camera center

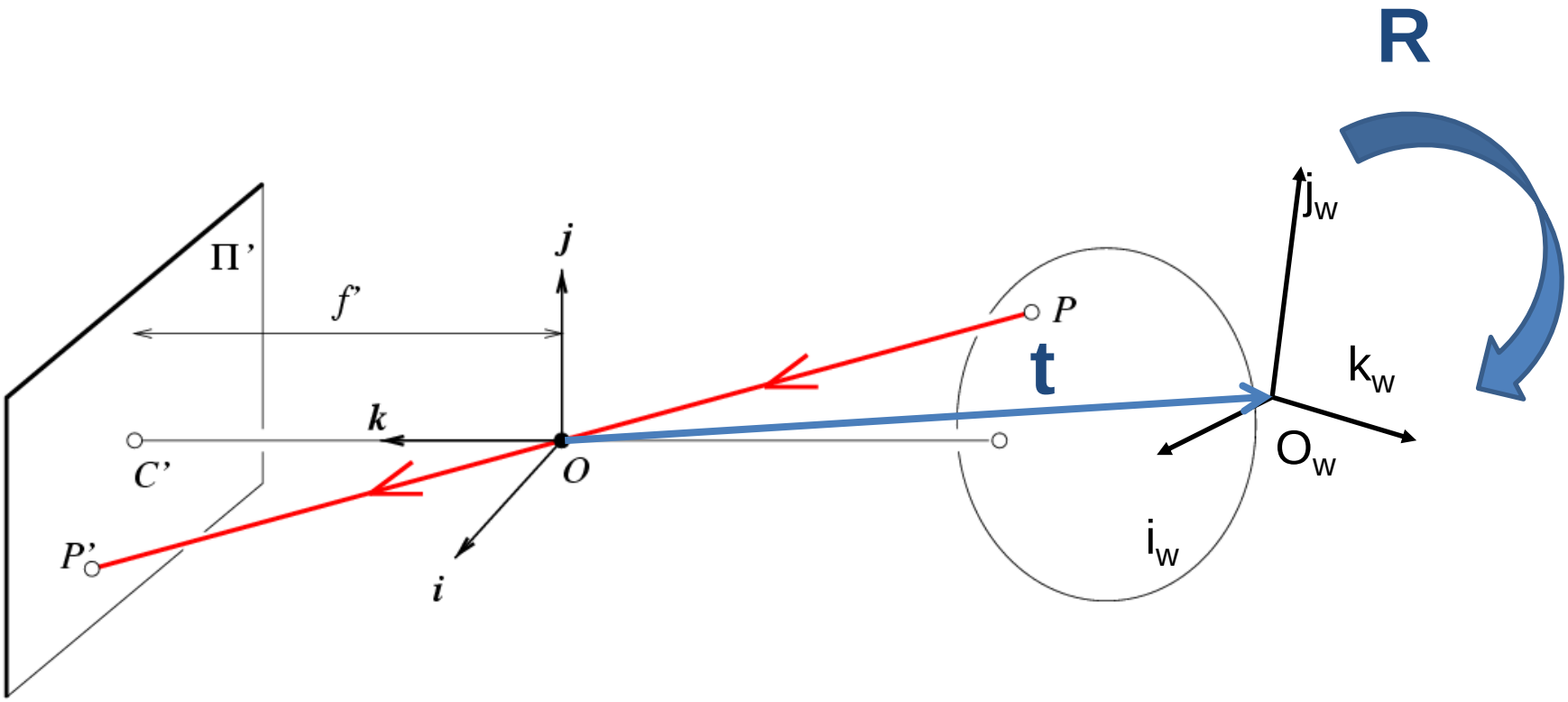


1.0486	-0.3645
-1.6851	-0.4004
-0.9437	-0.4200
1.0682	0.0699
0.6077	-0.0771
1.2543	-0.6454
-0.2709	0.8635
-0.4571	-0.3645
-0.7902	0.0307
0.7318	0.6382
-1.0580	0.3312
0.3464	0.3377
0.3137	0.1189
-0.4310	0.0242
-0.4799	0.2920
0.6109	0.0830
-0.4081	0.2920
-0.1109	-0.2992
0.5129	-0.0575
0.1406	-0.4527

1.5706	-0.1490	0.2598
-1.5282	0.9695	0.3802
-0.6821	1.2856	0.4078
0.4124	-1.0201	-0.0915
1.2095	0.2812	-0.1280
0.8819	-0.8481	0.5255
-0.9442	-1.1583	-0.3759
0.0415	1.3445	0.3240
-0.7975	0.3017	-0.0826
-0.4329	-1.4151	-0.2774
-1.1475	-0.0772	-0.2667
-0.5149	-1.1784	-0.1401
0.1993	-0.2854	-0.2114
-0.4320	0.2143	-0.1053
-0.7481	-0.3840	-0.2408
0.8078	-0.1196	-0.2631
-0.7605	-0.5792	-0.1936
0.3237	0.7970	0.2170
1.3089	0.5786	-0.1887
1.2323	1.4421	0.4506



Oriented and Translated Camera



Great! So now I have K and Rt

Marker-based augmented reality.

- Define marker with known real-world geometry
- Find points on marker in image
- Estimate camera pose from corresponding 2D/3D points
- Render graphics from virtual camera with same pose

Great! So now I have K and Rt

Building block for detailed reconstruction.

Goal: reconstruct depth.

So far: we have ‘calibrated’ one camera.

Or, potentially two…

Summary

- Projections
  - Rotation, translation, affine, perspective
- Cameras
  - Lenses, apertures, sensors
- Pinhole camera model and camera matrix
  - Intrinsic matrix
  - Extrinsic matrix
- Recovering camera matrix using least squares regression

ONE DIFFICULT EXAMPLE…

Erik Johansson – The Architect