

# CS5670: Computer Vision

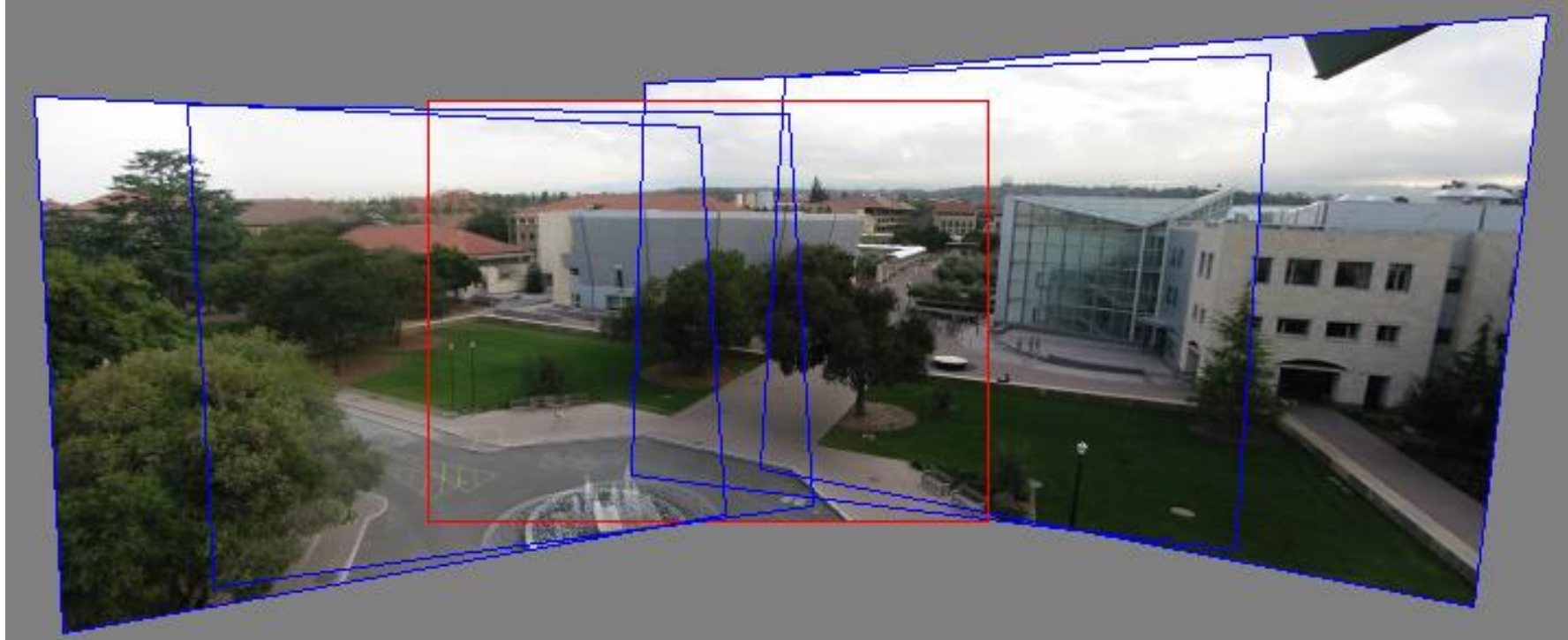
## Cameras



# Announcements

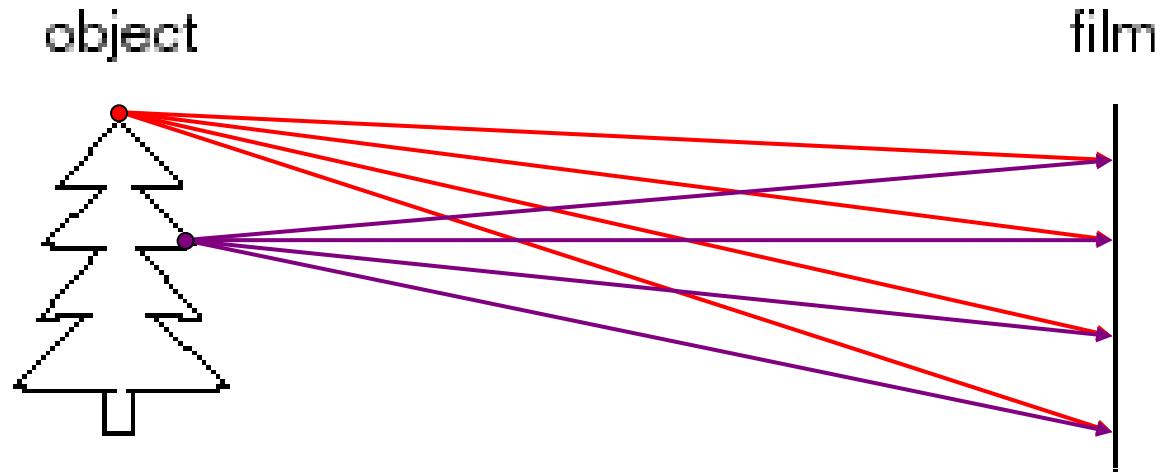
- Take-home midterm handed out at the end of class
  - Due Tuesday, March 8, by 1pm (beginning of class)
- Project 2 due tonight by 8pm
  - Report due Monday by 11:59pm
- Project 3: Panorama Stitching
  - To be released next Tuesday, March 8
  - Tentative due date on Friday, April 2

# Can we use homographies to create a 360 panorama?



- In order to figure this out, we need to learn what a **camera** is

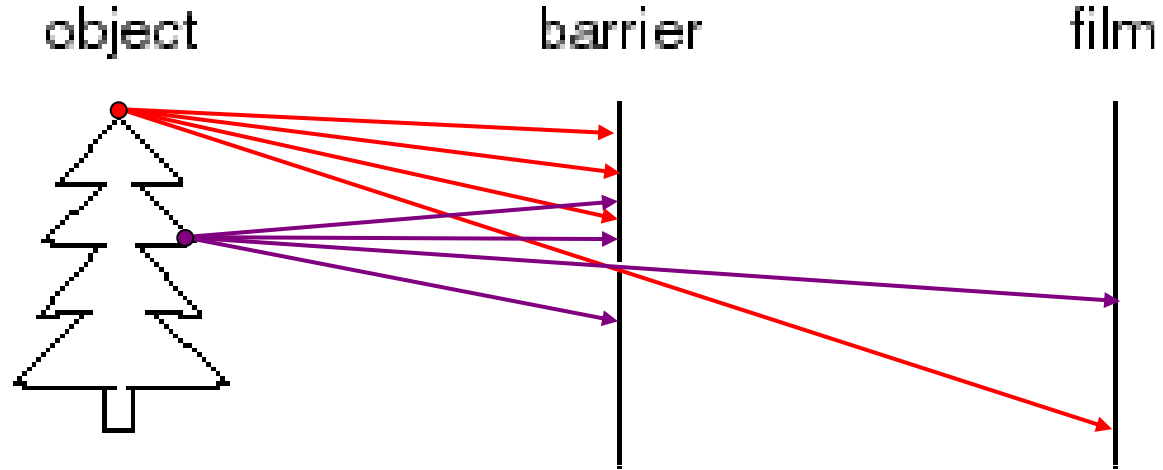
# Image formation



Let's design a camera

- Idea 1: put a piece of film in front of an object
- Do we get a reasonable image?
- No. This is a bad camera.

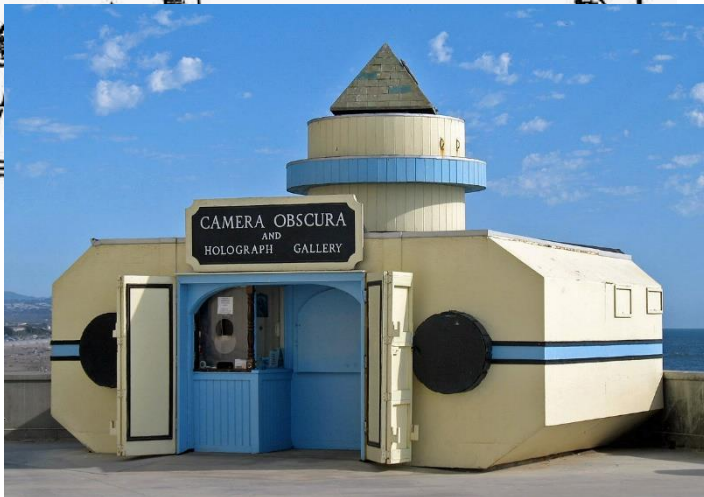
# Pinhole camera



Add a barrier to block off most of the rays

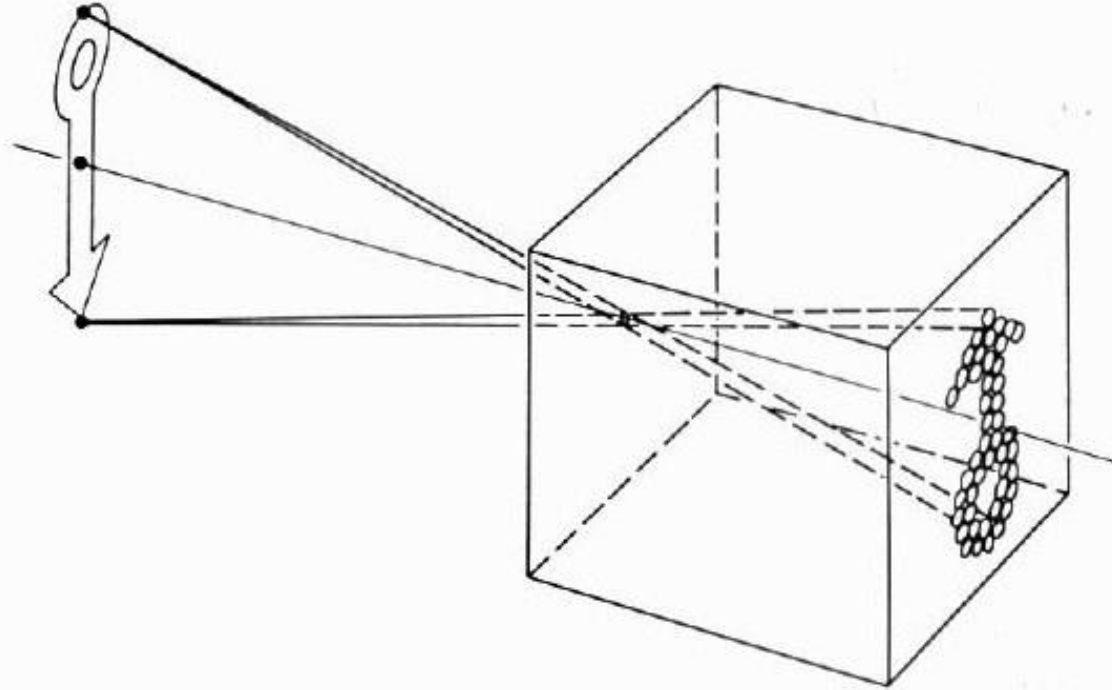
- This reduces blurring
- The opening known as the **aperture**
- How does this transform the image?

# Camera Obscura



- Basic principle known to Mozi (470-390 BC), Aristotle (384-322 BC)
- Drawing aid for artists: described by Leonardo da Vinci (1452-1519)

# Camera Obscura



# Home-made pinhole camera



Why so blurry?

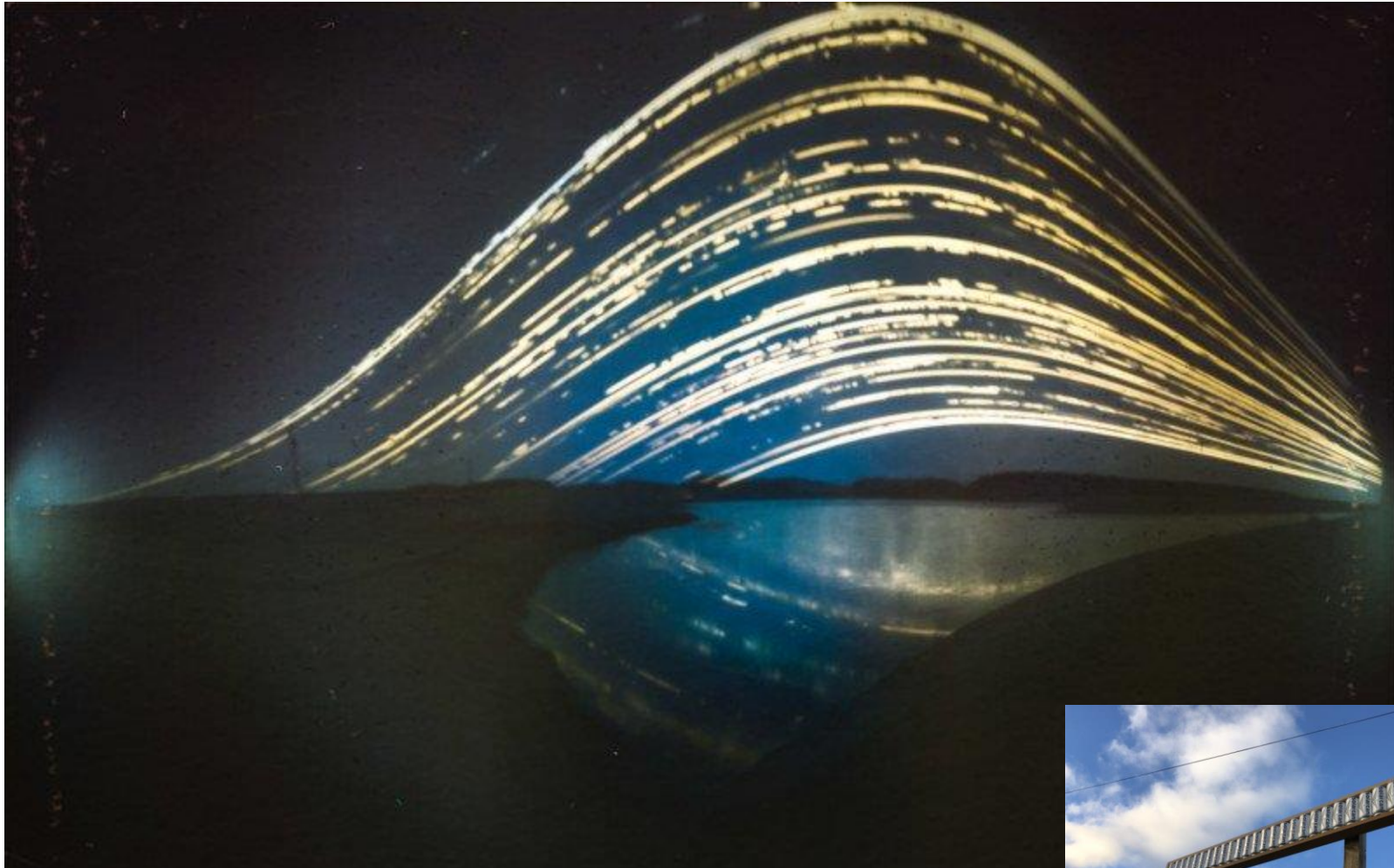




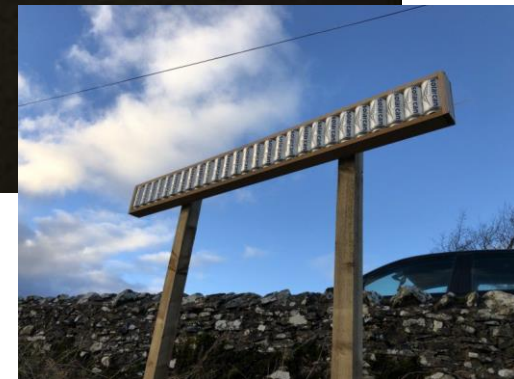
# Pinhole photography



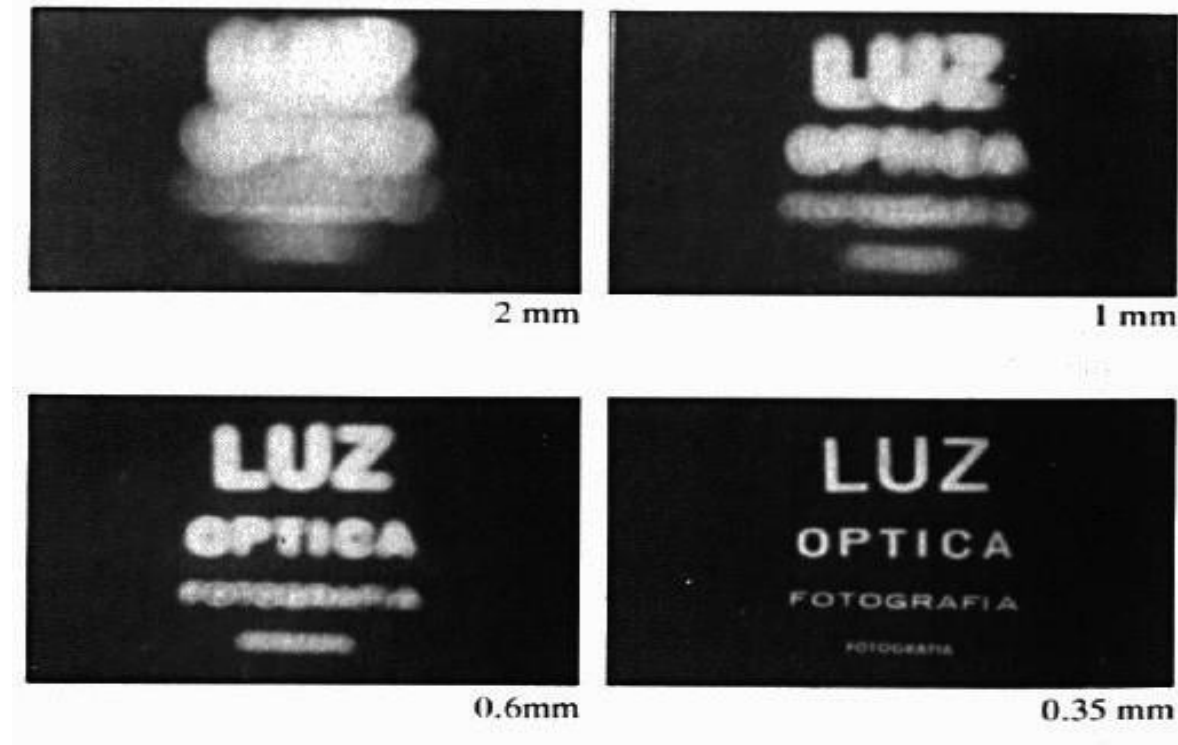
**Justin Quinnell**, The Clifton Suspension Bridge. December 17th 2007 - June 21st 2008  
***6-month exposure***



<https://petapixel.com/2019/07/17/this-is-the-worlds-first-solargraphy-timelapse/>

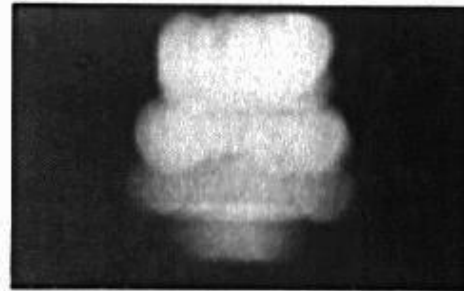


# Shrinking the aperture

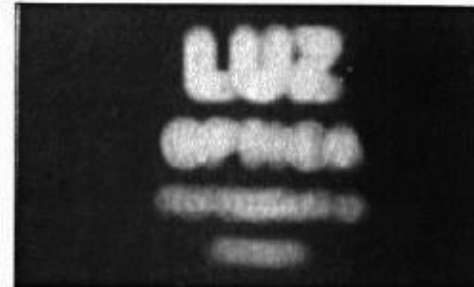


- Why not make the aperture as small as possible?
  - Less light gets through
  - *Diffraction* effects...

# Shrinking the aperture



2 mm



1 mm



0.6 mm



0.35 mm

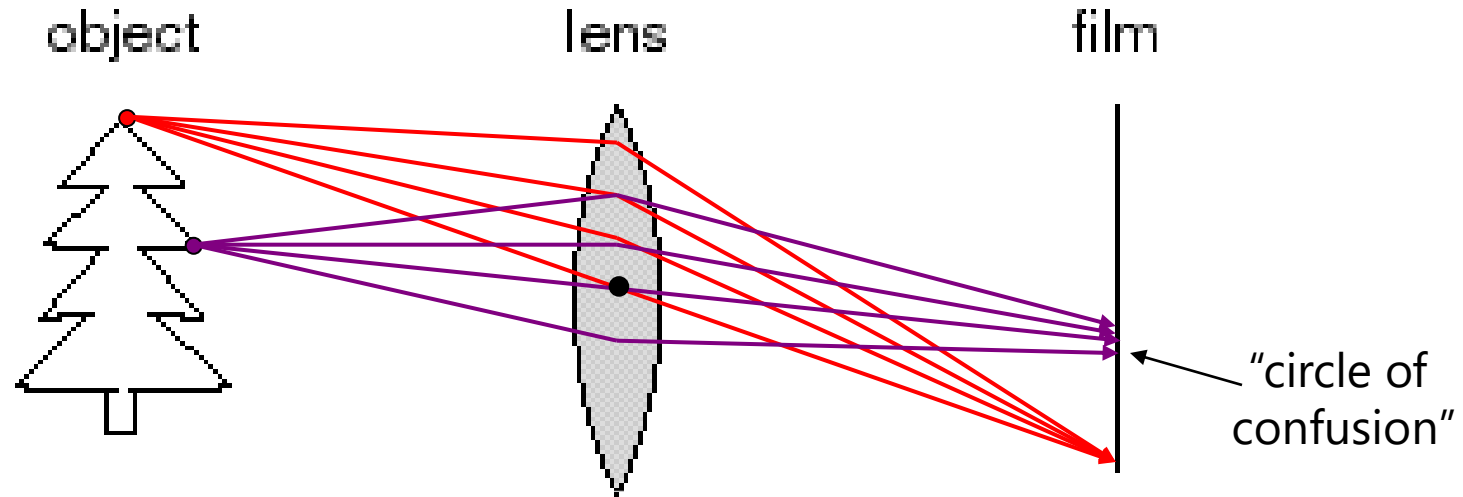


0.15 mm



0.07 mm

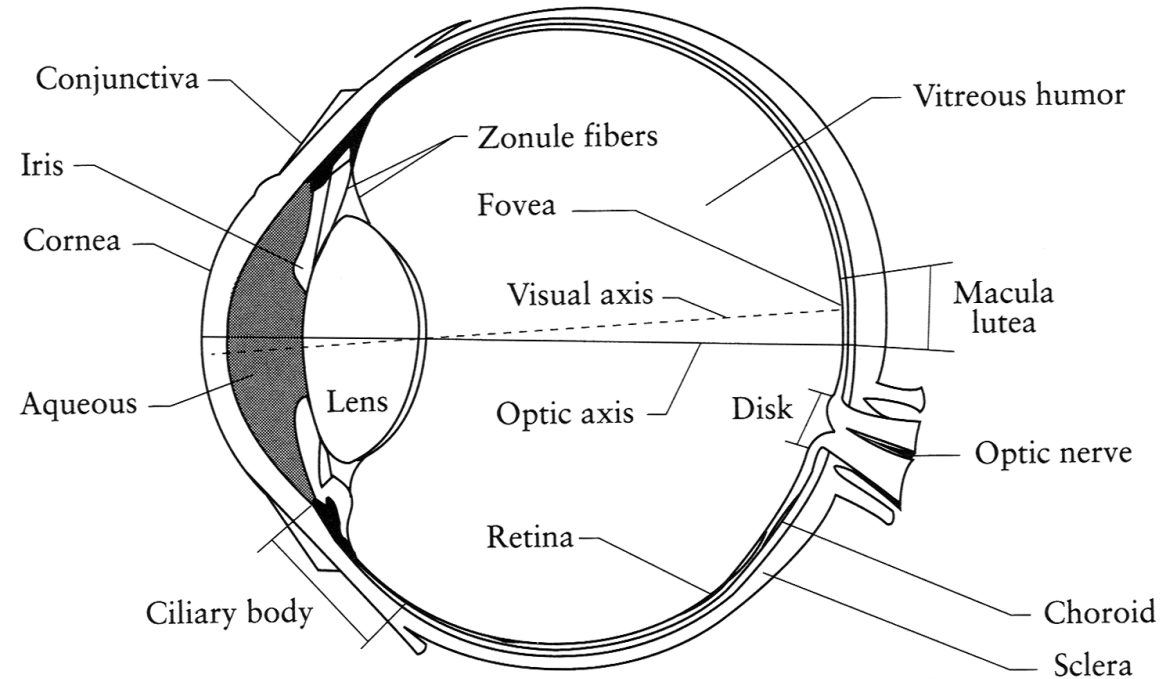
# Adding a lens



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

# The eye

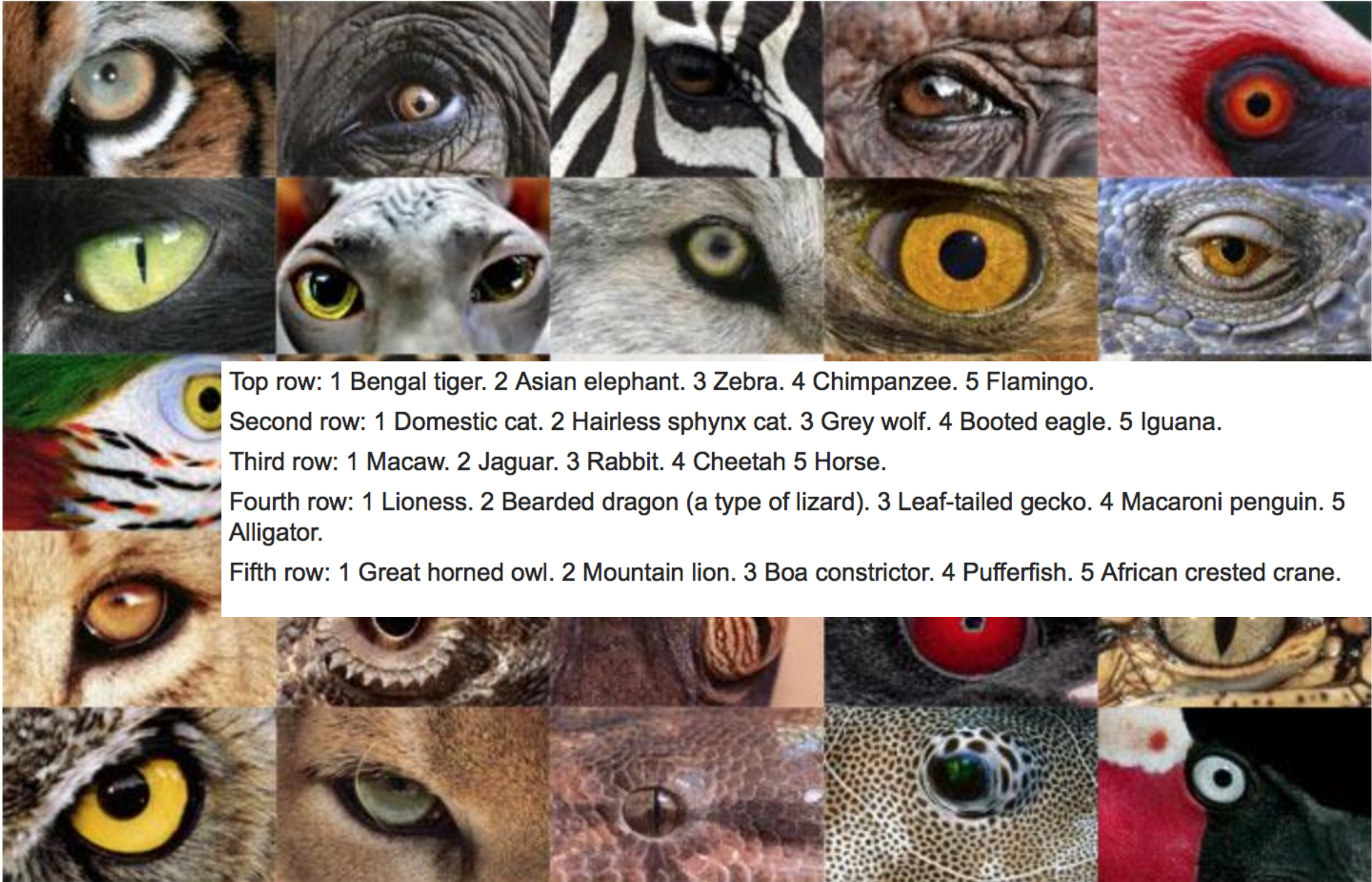


## The human eye is a camera

- **Iris** - colored annulus with radial muscles
- **Pupil** - the hole (aperture) whose size is controlled by the iris
- What's the "film"?
  - photoreceptor cells (rods and cones) in the **retina**



<http://www.telegraph.co.uk/news/earth/earthpicturegalleries/7598120/Animal-eyes-quiz-Can-you-work-out-which-creatures-these-are-from-their-eyes.html?image=25>



Top row: 1 Bengal tiger. 2 Asian elephant. 3 Zebra. 4 Chimpanzee. 5 Flamingo.  
Second row: 1 Domestic cat. 2 Hairless sphynx cat. 3 Grey wolf. 4 Booted eagle. 5 Iguana.  
Third row: 1 Macaw. 2 Jaguar. 3 Rabbit. 4 Cheetah 5 Horse.  
Fourth row: 1 Lioness. 2 Bearded dragon (a type of lizard). 3 Leaf-tailed gecko. 4 Macaroni penguin. 5 Alligator.  
Fifth row: 1 Great horned owl. 2 Mountain lion. 3 Boa constrictor. 4 Pufferfish. 5 African crested crane.



# Eyes in nature: eyespots to pinhole camera



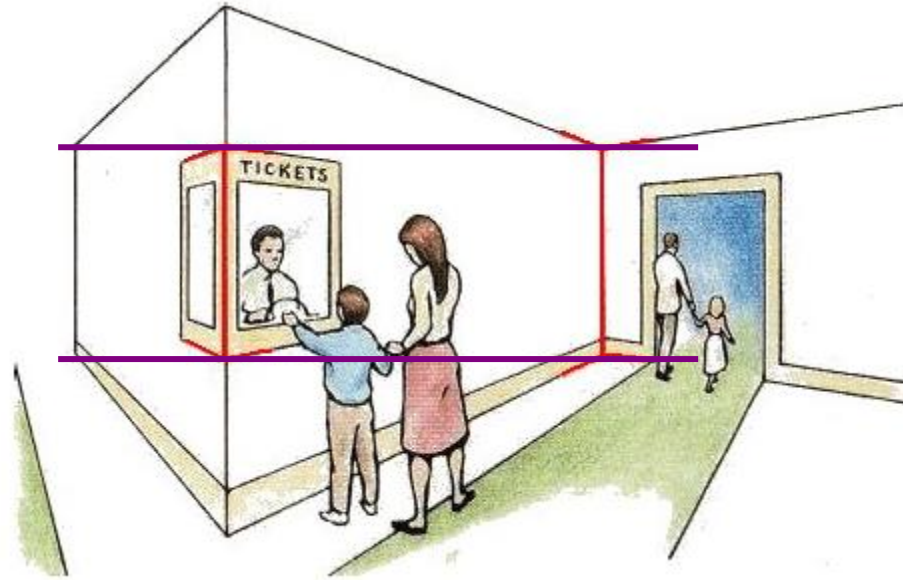
# Projection



# Projection

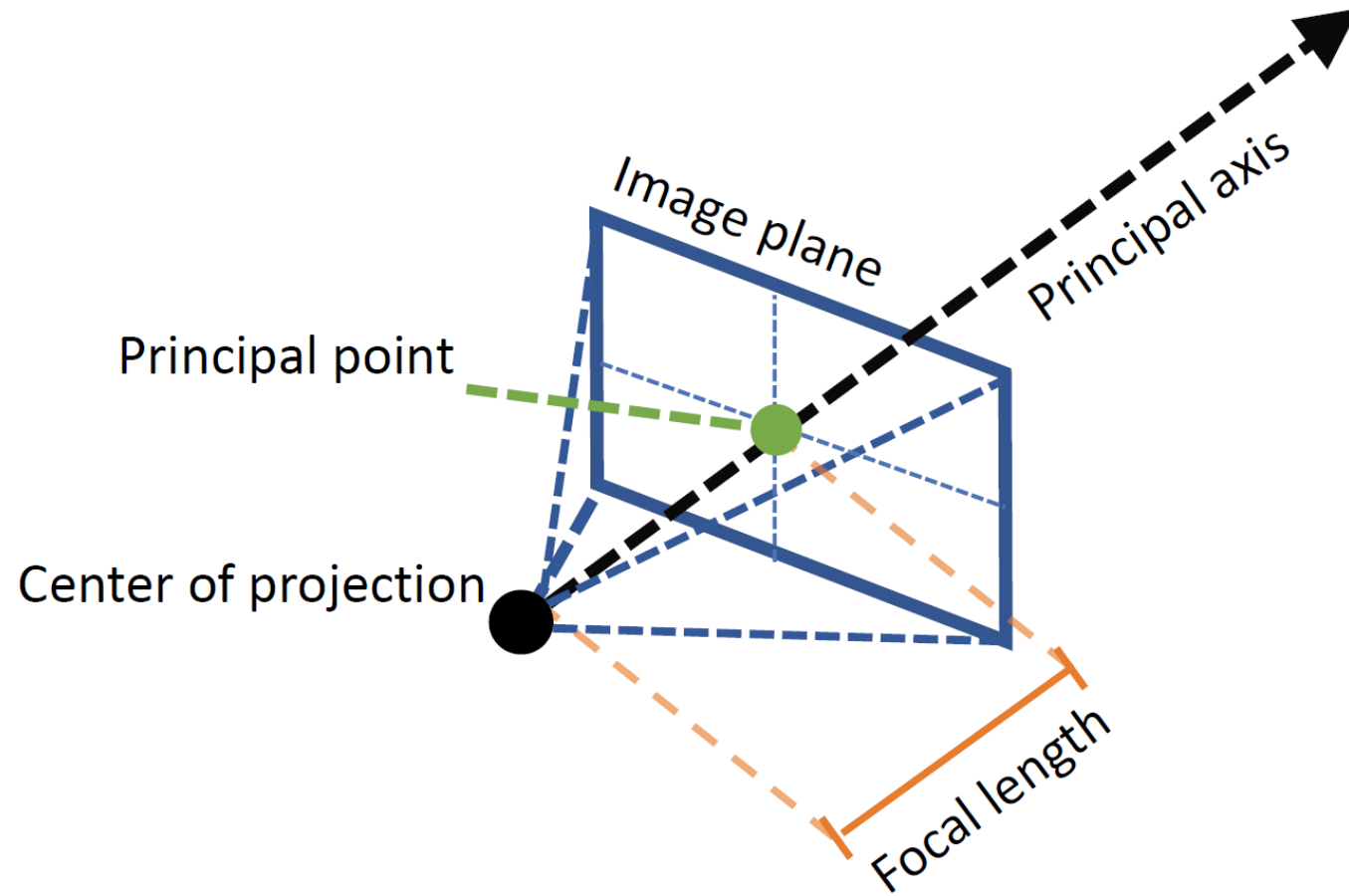


# Müller-Lyer Illusion



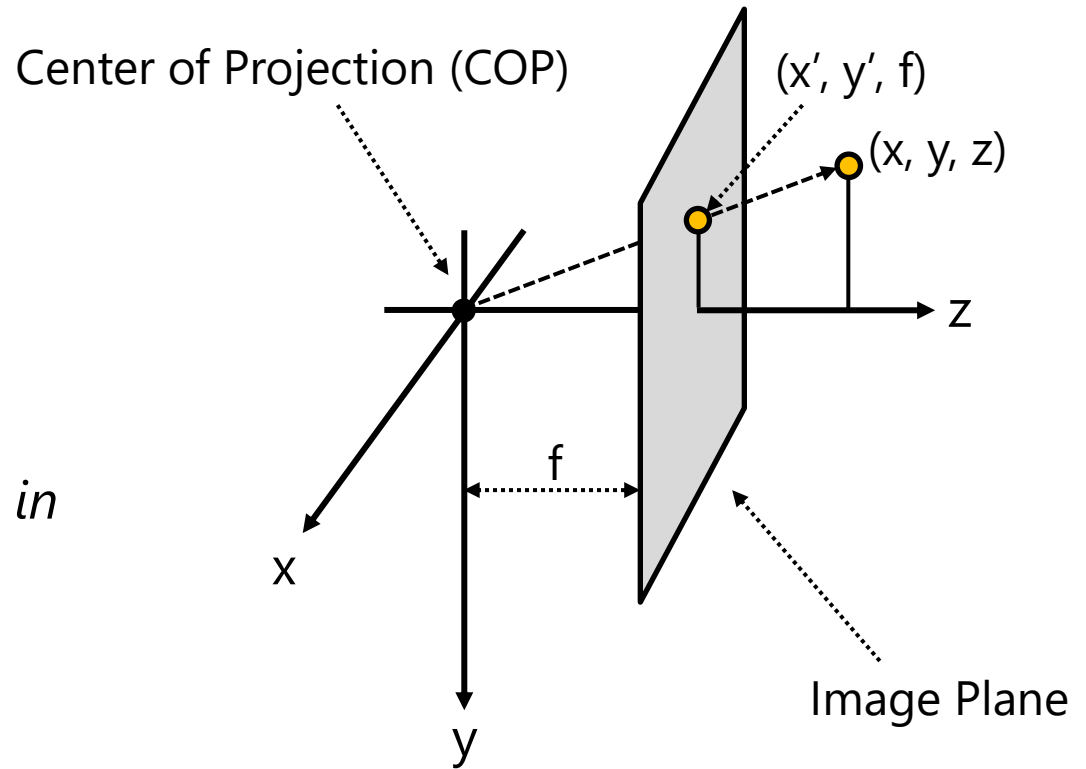
[https://en.wikipedia.org/wiki/Müller-Lyer\\_illusion](https://en.wikipedia.org/wiki/Müller-Lyer_illusion)

# Geometric Model: A Pinhole Camera



# Modeling projection

- The coordinate system
  - We use the pinhole model as an approximation
  - Put the optical center (aka Center of Projection, or COP) at the origin
  - Put the Image Plane (aka Projection Plane) *in front* of the COP (Why)?
  - The camera looks down the *positive*  $z$ -axis, and the  $y$ -axis points down
    - we like this if we want right-handed-coordinates
    - other versions are possible (e.g., OpenGL)



# Modeling projection

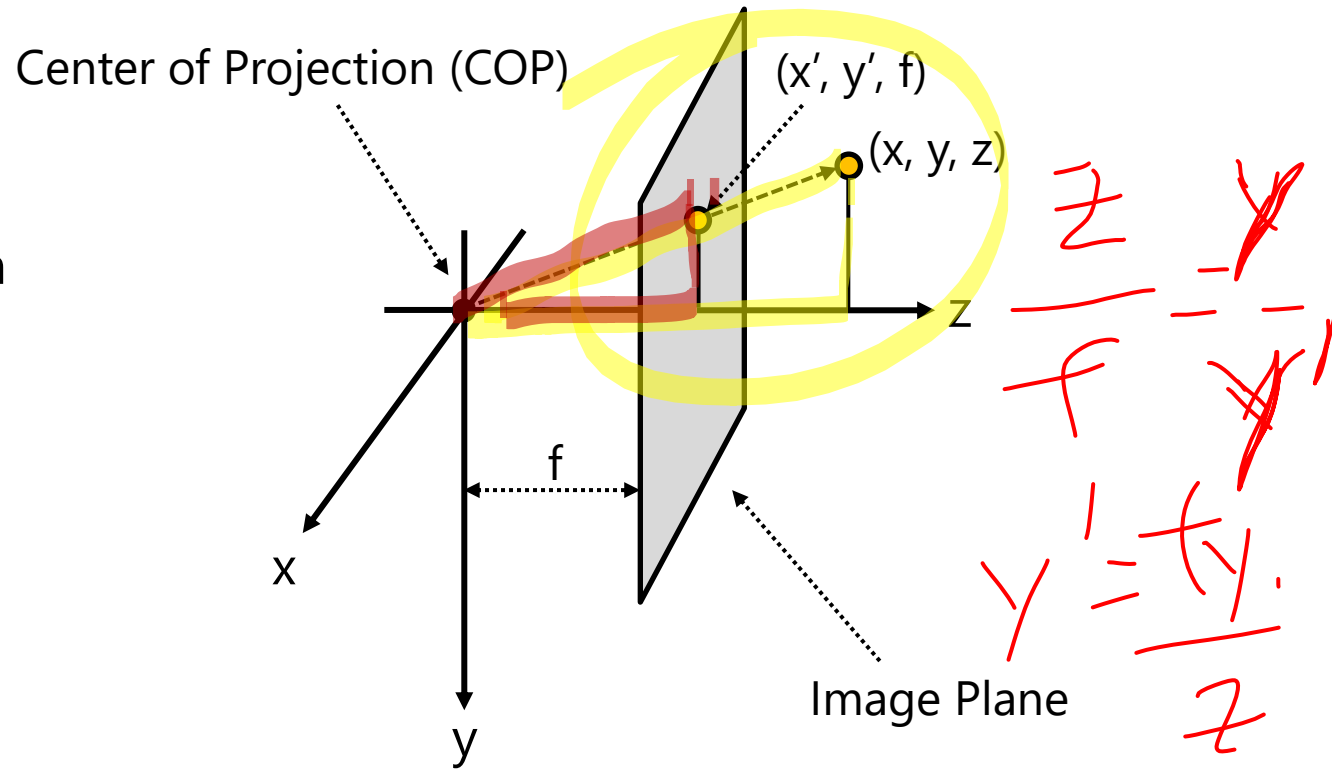
- Projection equations

- Compute intersection with PP of ray from  $(x,y,z)$  to COP
- Derived using similar triangles

$$(x, y, z) \rightarrow \left( f \frac{x}{z}, f \frac{y}{z}, f \right)$$

- We get the projection by throwing out the last coordinate:

$$(x, y, z) \rightarrow \left( f \frac{x}{z}, f \frac{y}{z} \right)$$



# Modeling projection

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

Homogeneous coordinates to the rescue—again!

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

homogeneous image  
coordinates

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

homogeneous scene  
coordinates

Converting *from* homogeneous coordinates

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



# Perspective Projection

Projection is a matrix multiply using homogeneous coordinates:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1/f & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ z/f \end{bmatrix} \Rightarrow \left( f \frac{x}{z}, f \frac{y}{z} \right)$$

divide by third coordinate

This is known as **perspective projection**

- The matrix is the **projection matrix**
- (Can also represent as a 4x4 matrix – OpenGL does something like this)

# Perspective Projection

How does scaling the projection matrix change the transformation?

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1/f & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ z/f \\ 1 \end{bmatrix} \Rightarrow \left( f \frac{x}{z}, f \frac{y}{z} \right)$$

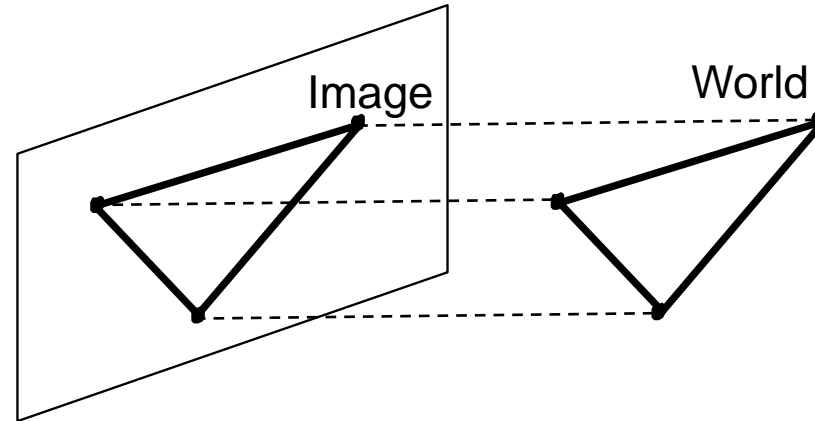
Scale by  $f$ :

$$\begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} fx \\ fy \\ z \\ 1 \end{bmatrix} \Rightarrow \left( f \frac{x}{z}, f \frac{y}{z} \right)$$

Scaling a projection matrix produces an equivalent projection matrix!

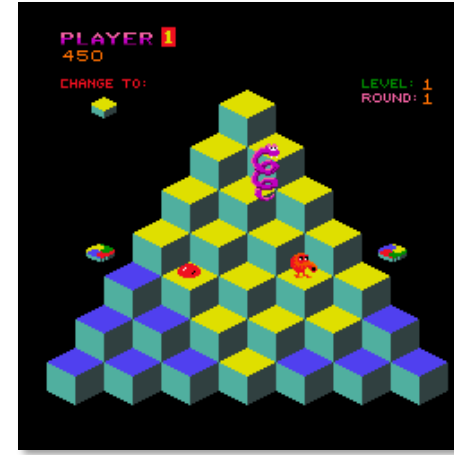
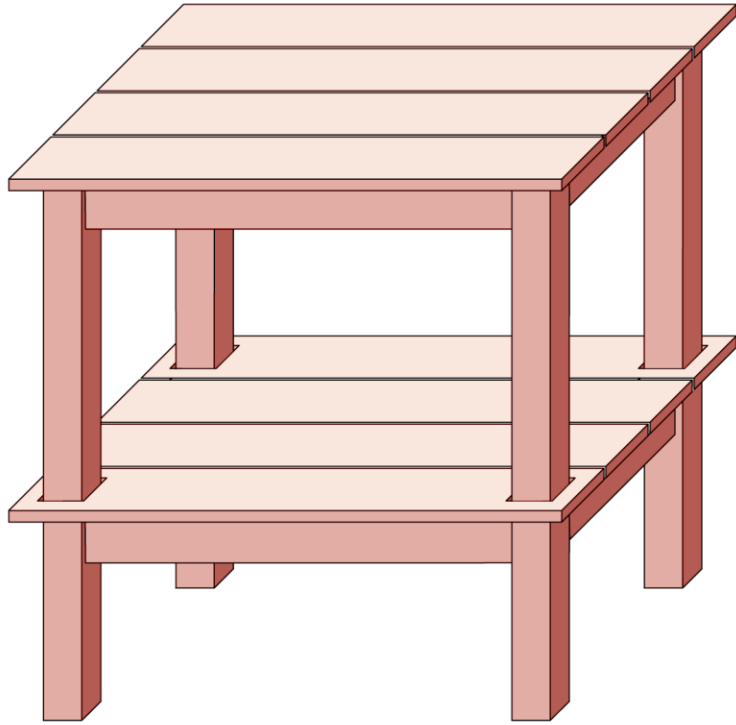
# Orthographic projection

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

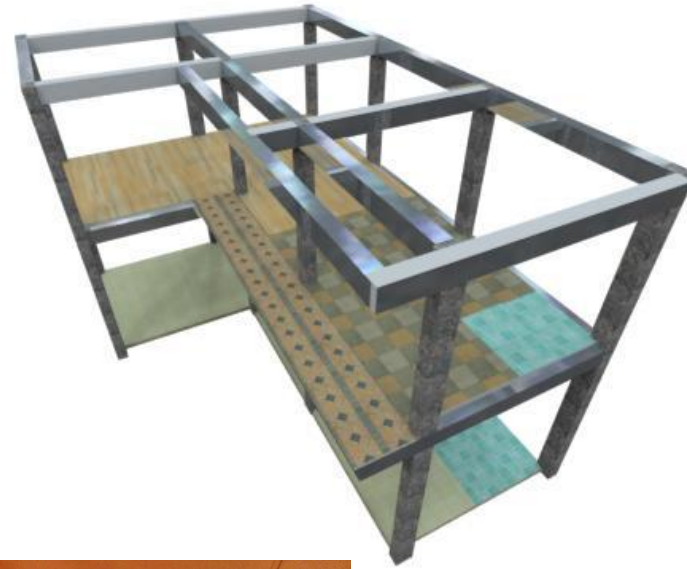


$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \Rightarrow (x, y)$$

# Orthographic projection



# Perspective projection



# Variants of orthographic projection

- Scaled orthographic
  - Also called “weak perspective”

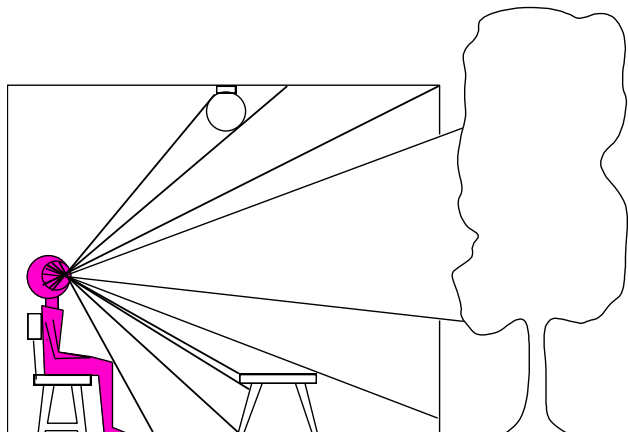
$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1/d \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ 1/d \end{bmatrix} \Rightarrow (dx, dy)$$

- Affine projection
  - Also called “paraperspective”

$$\begin{bmatrix} a & b & c & d \\ e & f & g & h \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

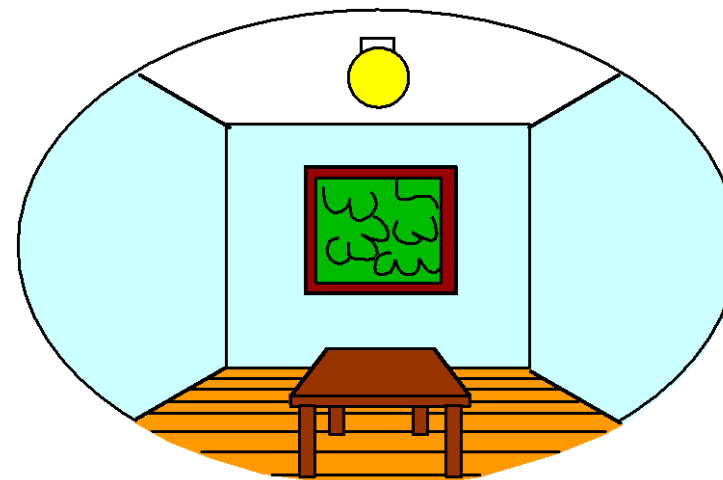
# Dimensionality Reduction Machine (3D to 2D)

*3D world*



Point of observation

*2D image*



What have we lost?

- Angles
- Distances (lengths)

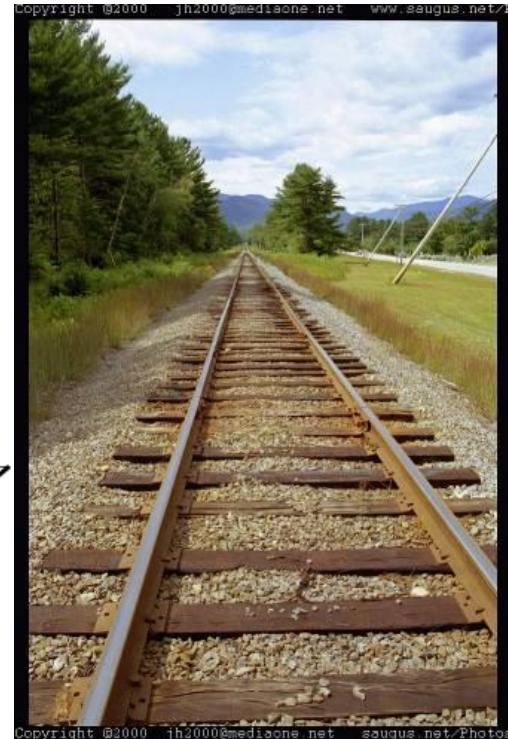
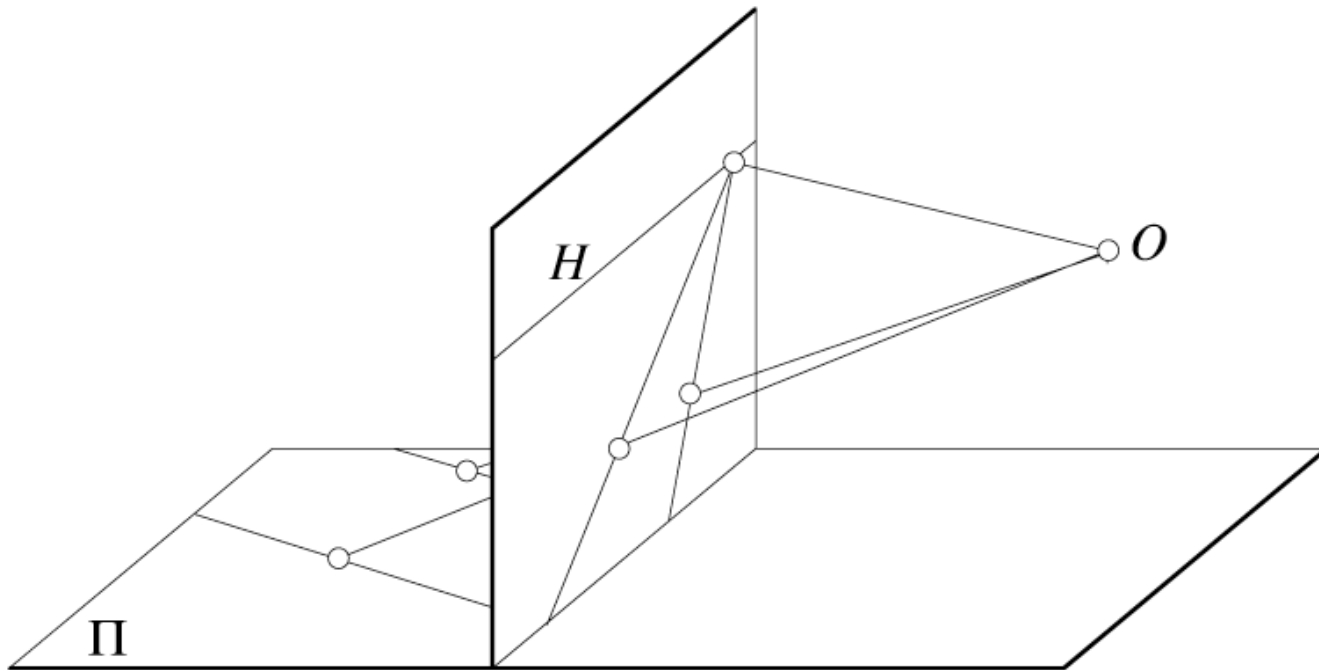
# Projection properties

- Many-to-one: any points along same ray map to same point in image
- Points  $\rightarrow$  points
- Lines  $\rightarrow$  lines (collinearity is preserved)
  - But line through focal point projects to a point
- Planes  $\rightarrow$  planes (or half-planes)
  - But plane through focal point projects to line



# Projection properties

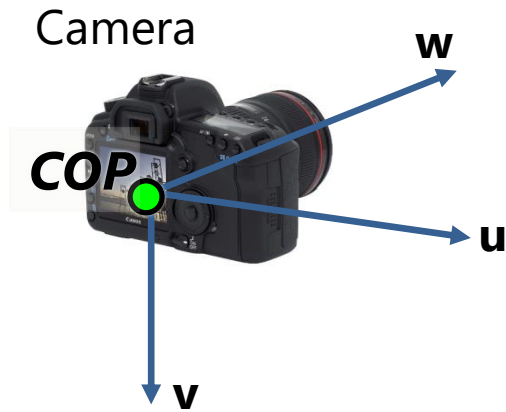
- Parallel lines converge at a vanishing point
  - Each direction in space has its own vanishing point
  - But parallel lines parallel to the image plane remain parallel



**Questions?**

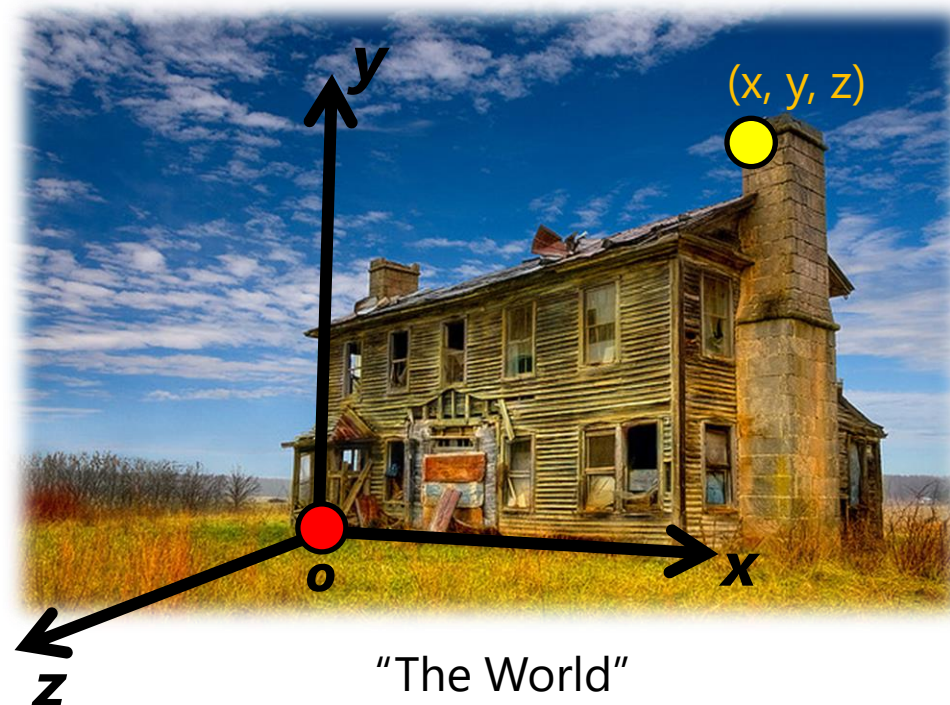
# Camera parameters

- How can we model the geometry of a camera?



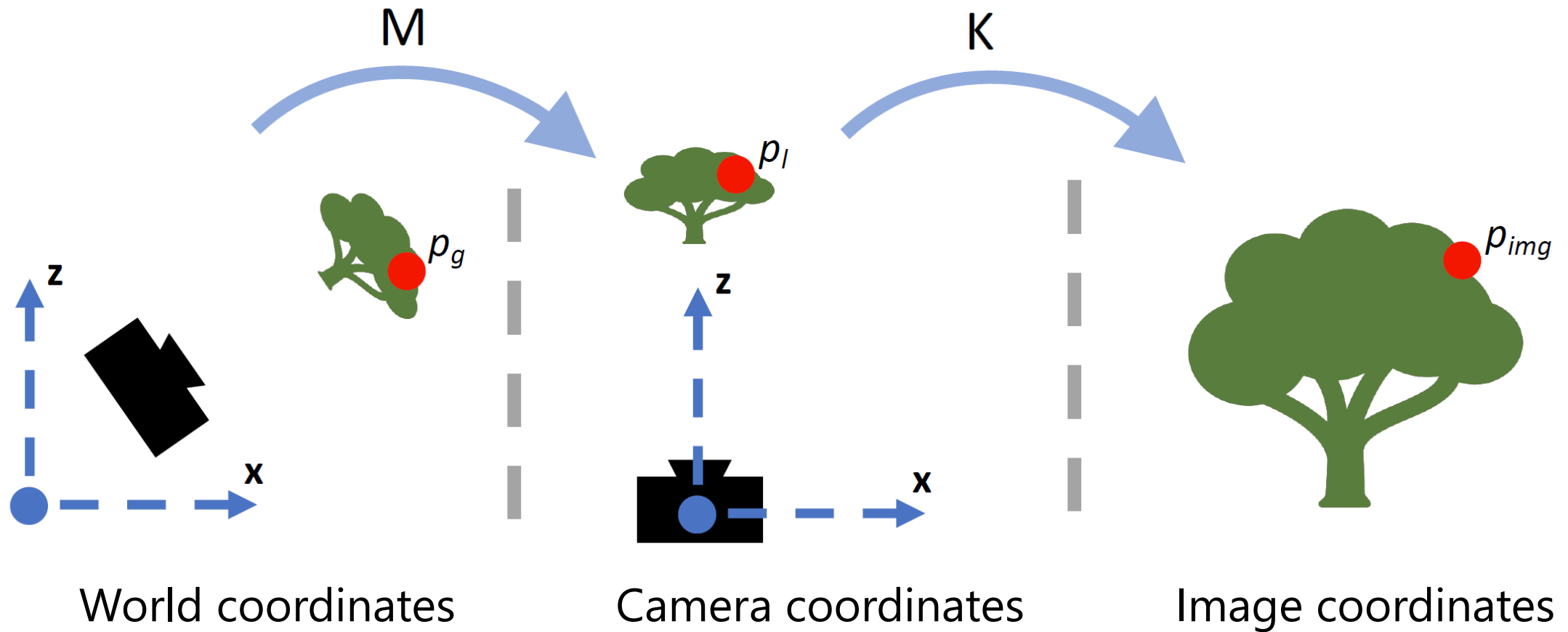
Three important coordinate systems:

1. *World* coordinates
2. *Camera* coordinates
3. *Image* coordinates



How do we project a given world point  $(x, y, z)$  to an image point?

# Coordinate frames



# Camera parameters

To project a point  $(x, y, z)$  in *world* coordinates into a camera

- First transform  $(x, y, z)$  into *camera* coordinates
- Need to know
  - Camera position (in world coordinates)
  - Camera orientation (in world coordinates)
- Then project into the image plane to get *image (pixel) coordinates*
  - Need to know camera *intrinsic*s

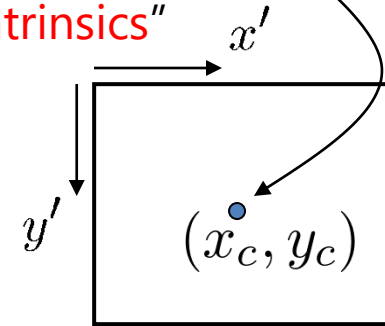
# Camera parameters

A camera is described by several parameters

- Translation  $\mathbf{T}$  of the optical center from the origin of world coords
- Rotation  $\mathbf{R}$  of the image plane
- focal length  $f$ , principal point  $(c_x, c_y)$ , pixel aspect size  $\alpha$
- blue parameters are called "extrinsics," red are "intrinsics"

Projection equation

$$\mathbf{x} = \begin{bmatrix} sx \\ sy \\ s \end{bmatrix} = \begin{bmatrix} * & * & * & * \\ * & * & * & * \\ * & * & * & * \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \mathbf{\Pi} \mathbf{X}$$



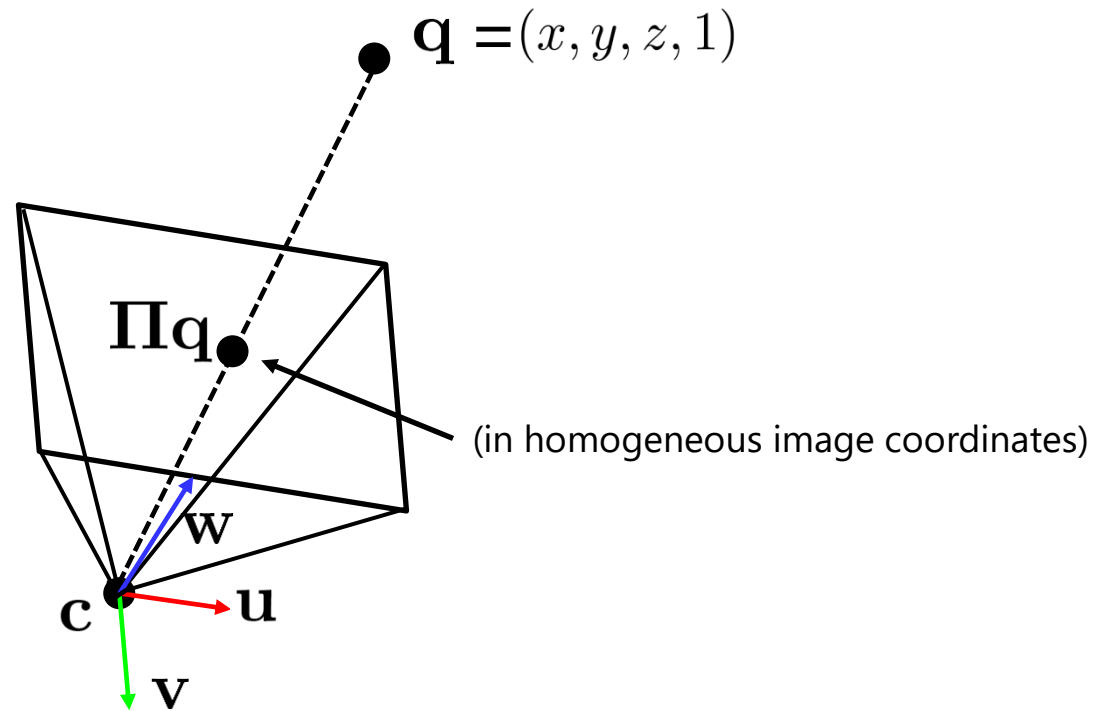
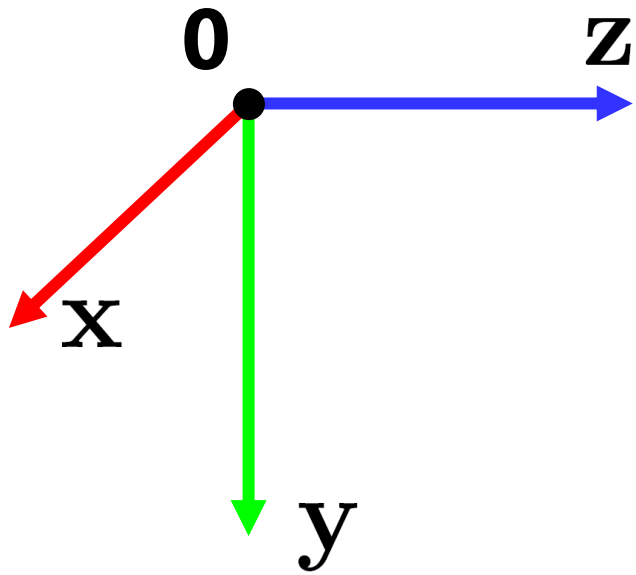
- The projection matrix models the cumulative effect of all parameters
- Useful to decompose into a series of operations

$$\mathbf{\Pi} = \underbrace{\begin{bmatrix} f & s & c_x \\ 0 & \alpha f & c_y \\ 0 & 0 & 1 \end{bmatrix}}_{\text{intrinsics}} \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{\text{projection}} \underbrace{\begin{bmatrix} \mathbf{R}_{3 \times 3} & \mathbf{0}_{3 \times 1} \\ \mathbf{0}_{1 \times 3} & 0 \end{bmatrix}}_{\text{rotation}} \underbrace{\begin{bmatrix} \mathbf{I}_{3 \times 3} & \mathbf{T}_{3 \times 1} \\ \mathbf{0}_{1 \times 3} & 0 \end{bmatrix}}_{\text{translation}}$$

identity matrix

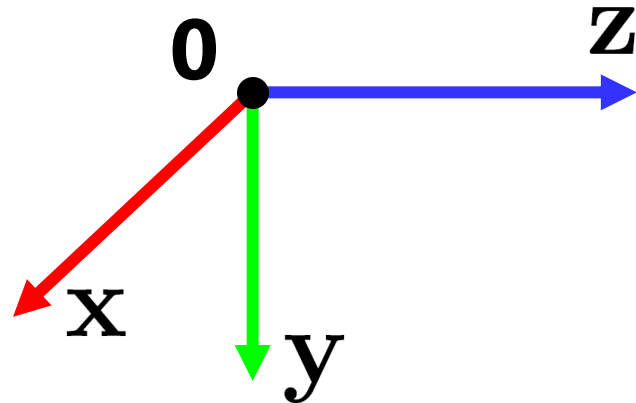
- The definitions of these parameters are **not** completely standardized
  - especially intrinsics—varies from one book to another

# Projection matrix

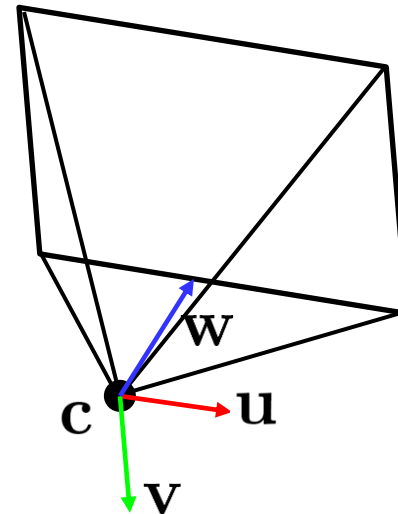


# Extrinsics

- How do we get the camera to "canonical form"?
  - Canonical form: Center of projection at the origin, x-axis points right, y-axis points down, z-axis points forwards



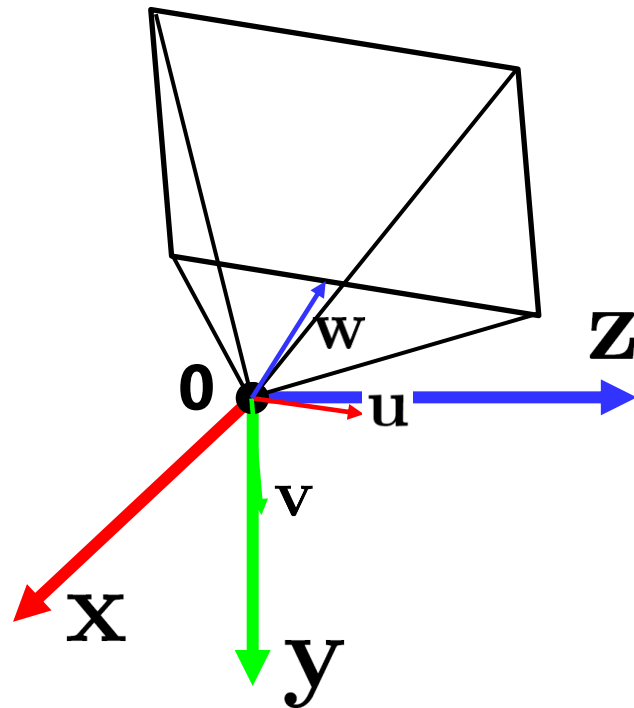
Step 1: Translate by  $-\mathbf{c}$





# Extrinsics

- How do we get the camera to “canonical form”?
  - Canonical form: Center of projection at the origin, x-axis points right, y-axis points down, z-axis points forwards



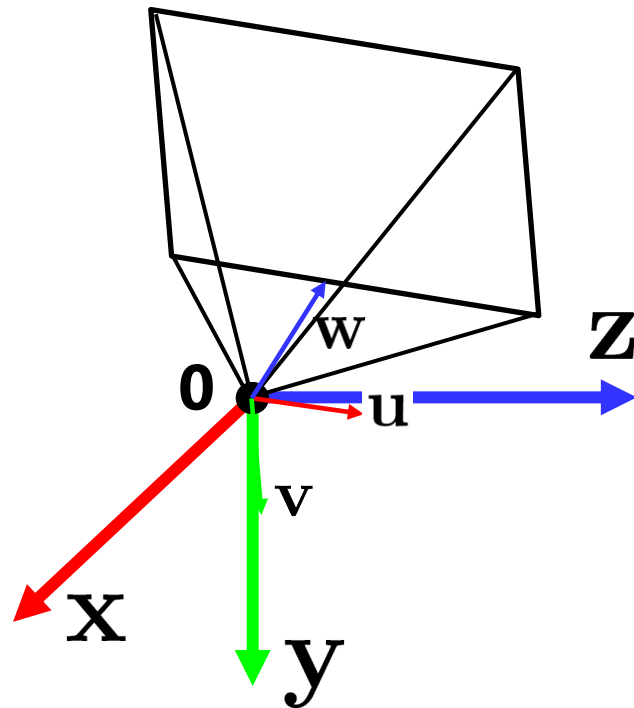
Step 1: Translate by  $-\mathbf{c}$

How do we represent translation as a matrix multiplication?

$$\mathbf{T} = \begin{bmatrix} \mathbf{I}_{3 \times 3} & -\mathbf{c} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

# Extrinsics

- How do we get the camera to “canonical form”?
  - Canonical form: Center of projection at the origin, x-axis points right, y-axis points down, z-axis points forwards



Step 1: Translate by  $-\mathbf{c}$

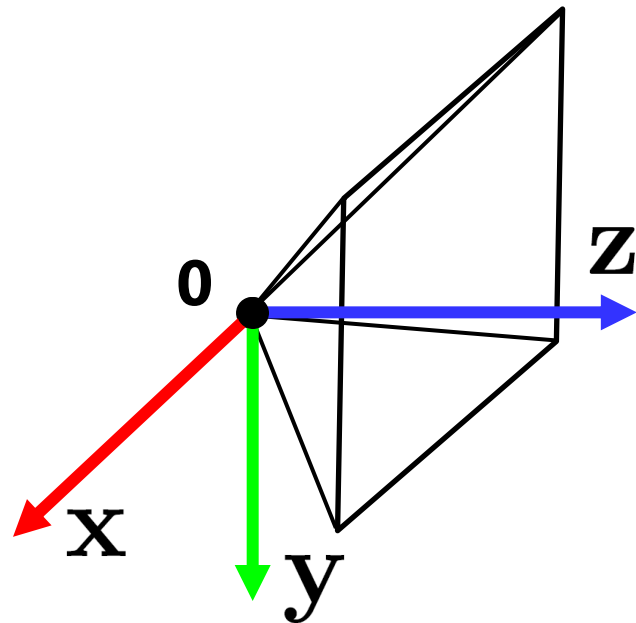
Step 2: Rotate by  $\mathbf{R}$

$\mathbf{R} = \begin{bmatrix} \mathbf{u}^T \\ \mathbf{v}^T \\ \mathbf{w}^T \end{bmatrix}$

3x3 rotation matrix

# Extrinsics

- How do we get the camera to "canonical form"?
  - Canonical form: Center of projection at the origin, x-axis points right, y-axis points down, z-axis points forwards



Step 1: Translate by  $-\mathbf{c}$

Step 2: Rotate by  $\mathbf{R}$

$$\mathbf{R} = \begin{bmatrix} \mathbf{u}^T \\ \mathbf{v}^T \\ \mathbf{w}^T \end{bmatrix}$$

(with extra row/column of [0 0 0 1])

# Perspective projection

$$\underbrace{\begin{bmatrix} f & 0 & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{bmatrix}}_{\mathbf{K}} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

**K**  
(intrinsics)

(converts from 3D rays in camera coordinate system to pixel coordinates)

in general,  $\mathbf{K} = \begin{bmatrix} f & s & c_x \\ 0 & \alpha f & c_y \\ 0 & 0 & 1 \end{bmatrix}$  (upper triangular matrix)

$\alpha$  : **aspect ratio** (1 unless pixels are not square)

$s$  : **skew** (0 unless pixels are shaped like rhombi/parallelograms)

$(c_x, c_y)$  : **principal point** ((w/2,h/2) unless optical axis doesn't intersect projection plane at image center)

# Typical intrinsics matrix

$$\mathbf{K} = \begin{bmatrix} f & 0 & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

- **2D affine transform** corresponding to a scale by  $f$  (focal length) and a translation by  $(c_x, c_y)$  (principal point)
- Maps 3D rays to 2D pixels

# Focal length

- Can think of as "zoom"



24mm



50mm



200mm



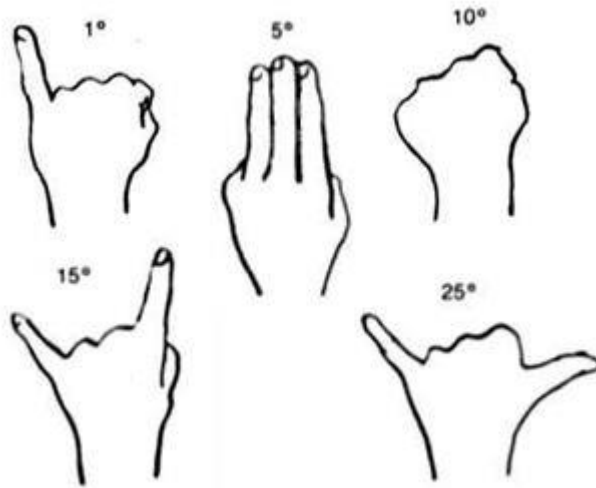
800mm



- Also related to *field of view*

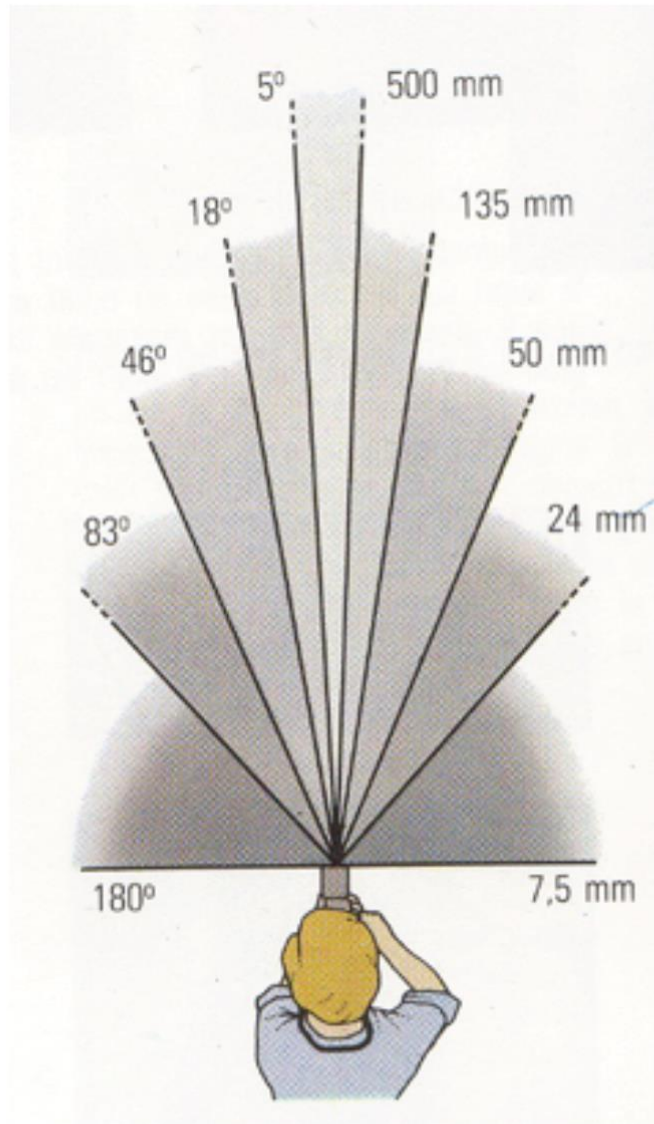
# Field of view

APS-C Crop Body Measurement Table



Lens	After 1.62 Multiplier	APS-C Sensor (1.62 lens multiplier) Canon 60D, 7D, 70D, T3i, T4i	Hand Positions
18mm	29.16mm	Three hands wide at full arms length.	Shows Lens on Canon APS-C Sensor = 29.16mm Frame Width = Three Hands Wide
28mm	45.36mm	Slightly less than two hands wide at full arms length.	Shows Lens on Canon APS-C Sensor = 45.36mm Frame Width = Slightly less than Two Hands Wide
35mm	56.7mm	One hand + width of one fist at full arms length.	Shows Lens on Canon APS-C Sensor = 56.7mm Frame Width = One Hand + Width of One Fist
50mm	81mm	One hand wide + width of thumb at full arms length.	Shows Lens on Canon APS-C Sensor = 81mm Frame Width = One Hand + Width of Thumb
55mm	89.1mm	Slightly less than one hand wide at full arms length.	Shows Lens on Canon APS-C Sensor = 89.1mm Frame Width = Slightly less than One Hand Wide
85mm	137.7mm	Inside edge of thumb to tip of forefinger wide with hand in "L" shape, thumb up.	Shows Lens on Canon APS-C Sensor = 137.7mm Frame Width = Inside Edge of Thumb to Tip of Forefinger Wide

# Focal length in practice



24mm



50mm

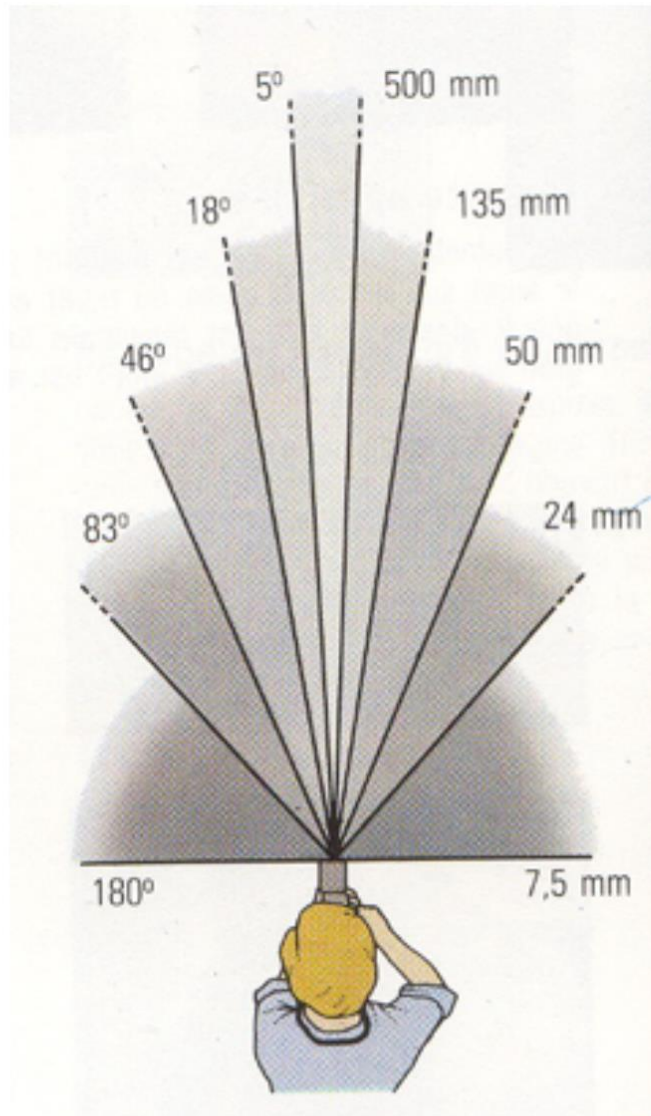


135mm

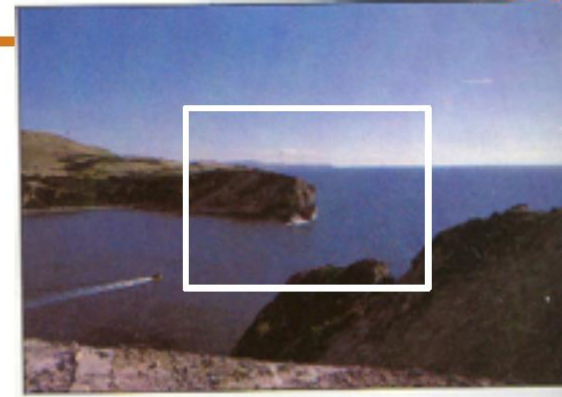




# Focal length = cropping



24mm



50mm

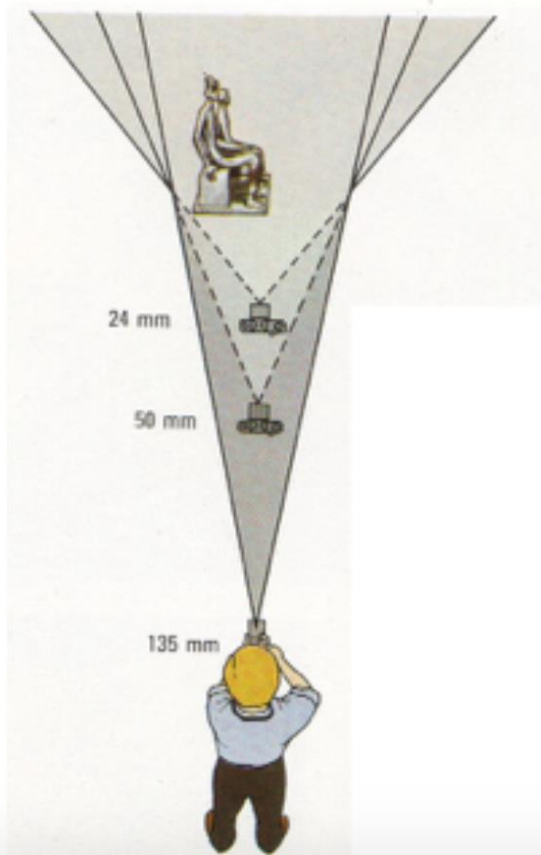


135mm



# Focal length vs. viewpoint

- **Telephoto makes it easier to select background (a small change in viewpoint is a big change in background).**



Grand-angle 24 mm



Normal 50 mm



Longue focale 135 mm



Fredo Durand



Wide angle



Standard



Telephoto



<http://petapixel.com/2013/01/11/how-focal-length-affects-your-subjects-apparent-weight-as-seen-with-a-cat/>

# Projection matrix

$$\mathbf{\Pi} = \mathbf{K} \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{\text{projection}} \underbrace{\begin{bmatrix} \mathbf{R} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{\text{rotation}} \underbrace{\begin{bmatrix} \mathbf{I}_{3 \times 3} & -\mathbf{c} \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{\text{translation}}$$

The  $\mathbf{K}$  matrix converts 3D rays in the camera's coordinate system to 2D image points in image (pixel) coordinates.

This part converts 3D points in world coordinates to 3D rays in the camera's coordinate system. There are 6 parameters represented (3 for position/translation, 3 for rotation).

# Projection matrix

$$\mathbf{\Pi} = \mathbf{K} \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{\text{projection}} \underbrace{\begin{bmatrix} \mathbf{R} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{\text{rotation}} \underbrace{\begin{bmatrix} \mathbf{I}_{3 \times 3} & -\mathbf{c} \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{\text{translation}}$$

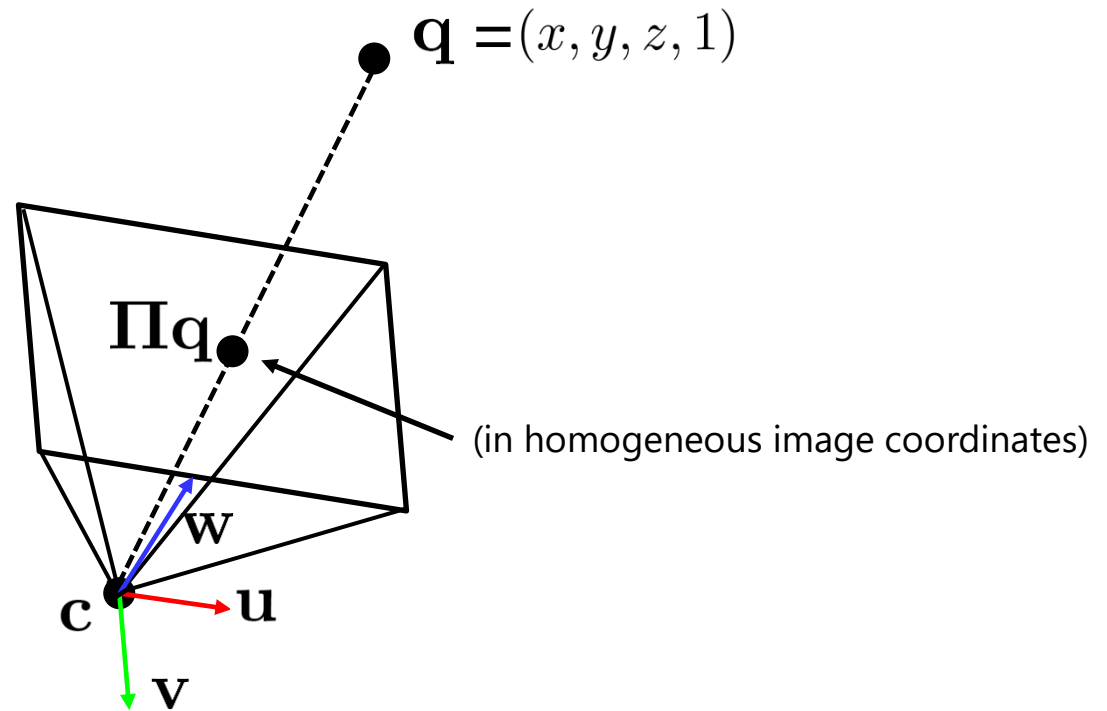
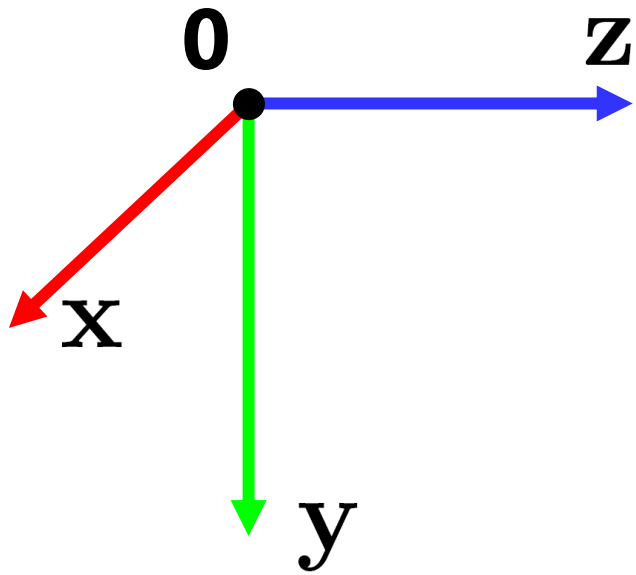
$$\left[ \mathbf{R} \mid \underbrace{-\mathbf{R}\mathbf{c}} \right]$$



( $\mathbf{t}$  in book's notation)

$$\mathbf{\Pi} = \mathbf{K} \left[ \mathbf{R} \mid -\mathbf{R}\mathbf{c} \right]$$

# Projection matrix





**Questions?**

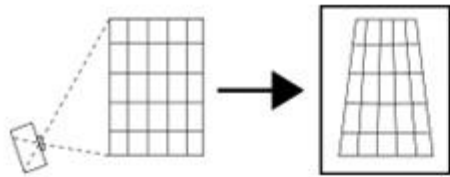
# Perspective distortion

- Problem for architectural photography: converging verticals

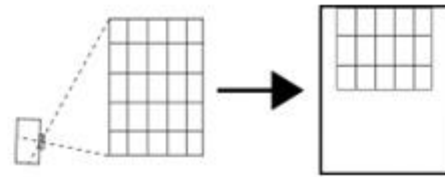


# Perspective distortion

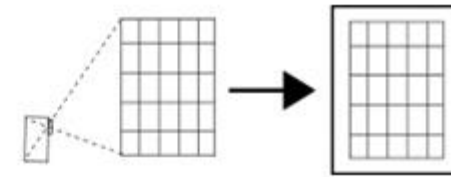
- Problem for architectural photography: converging verticals



Tilting the camera upwards results in converging verticals

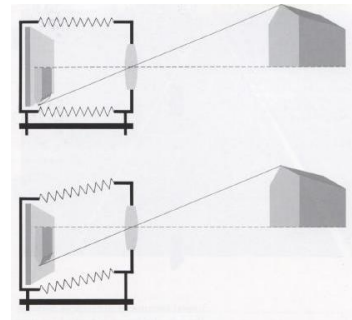
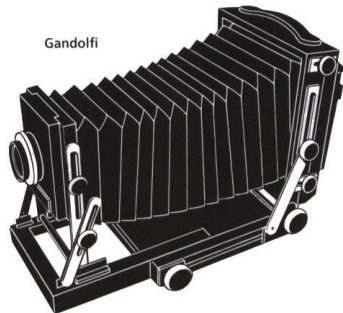


Keeping the camera level, with an ordinary lens, captures only the bottom portion of the building



Shifting the lens upwards results in a picture of the entire subject

- Solution: view camera (lens shifted w.r.t. film)



[http://en.wikipedia.org/wiki/Perspective\\_correction\\_lens](http://en.wikipedia.org/wiki/Perspective_correction_lens)

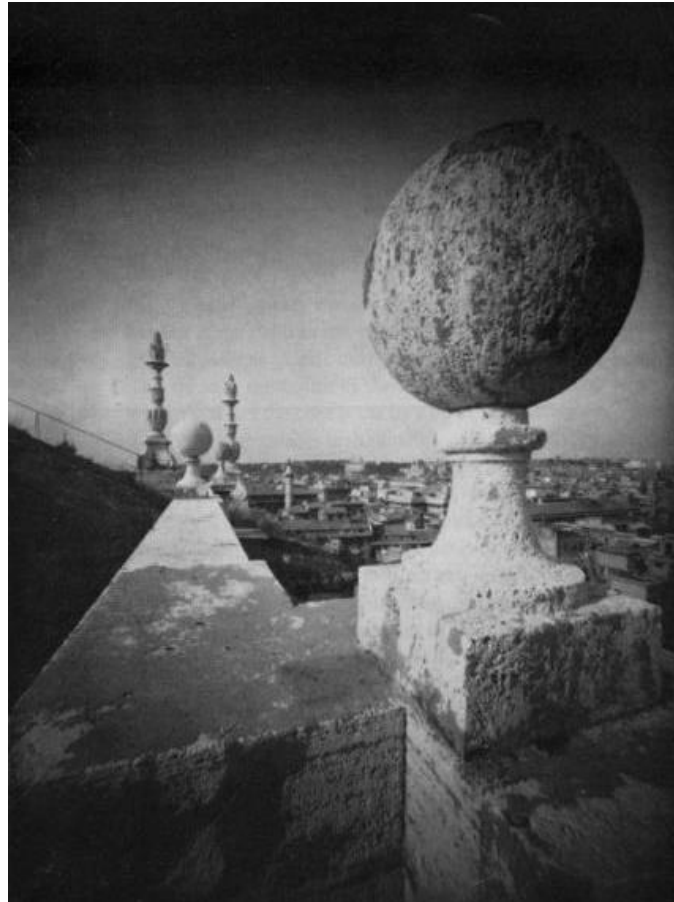
# Perspective distortion

- Problem for architectural photography: converging verticals
- Result:



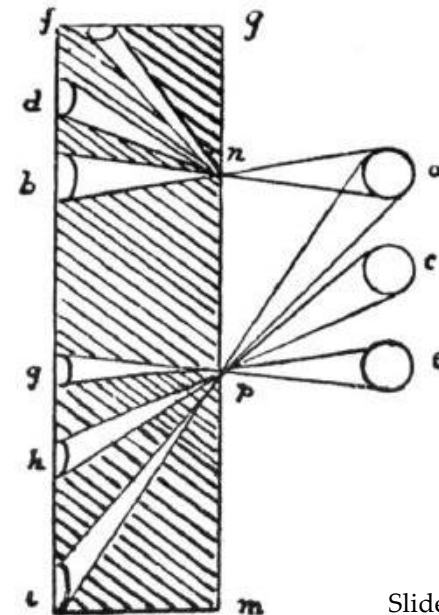
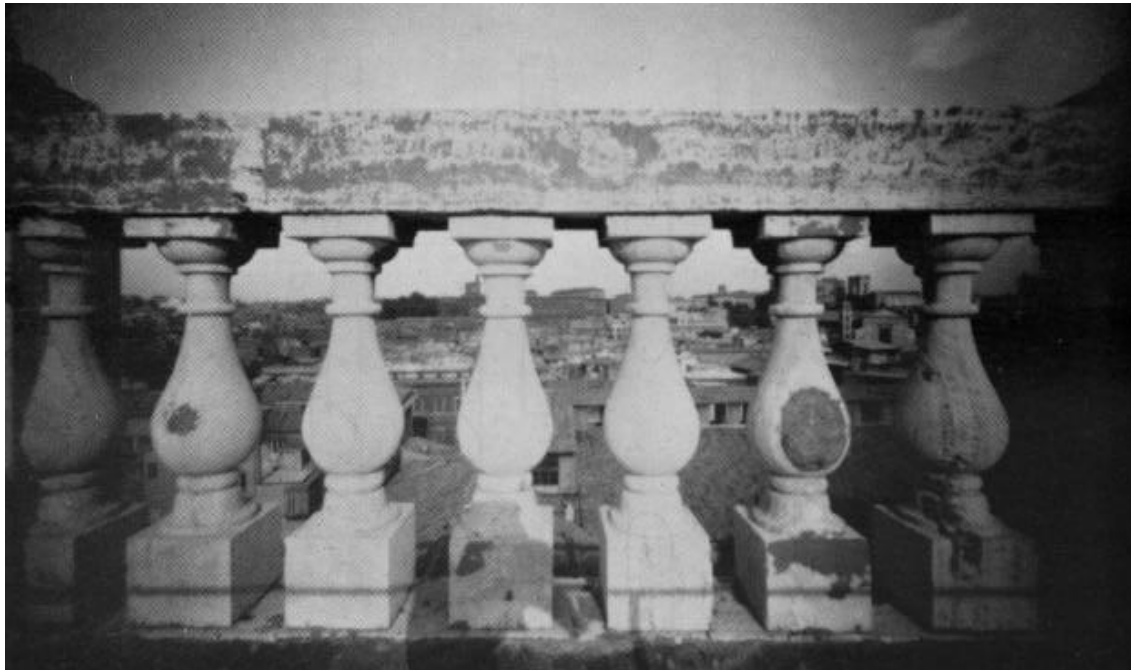
# Perspective distortion

- What does a sphere project to?



# Perspective distortion

- The exterior columns appear bigger
- The distortion is not due to lens flaws
- Problem pointed out by Da Vinci





<https://aaronhertzmamm.com/2022/02/28/how-does-perspective-work.html>

# Perspective distortion: People





# Distortion-Free Wide-Angle Portraits on Camera Phones



(a) A wide-angle photo with distortions on subjects' faces.

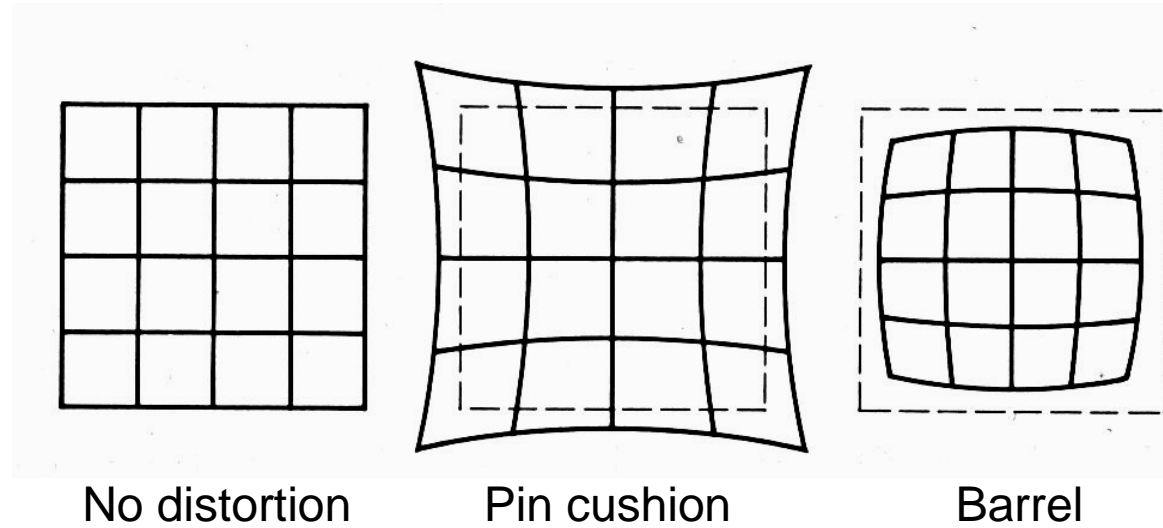


(b) Distortion-free photo by our method.

YiChang Shih, Wei-Sheng Lai, and Chia-Kai Liang, Distortion-Free Wide-Angle Portraits on Camera Phones, SIGGRAPH 2019

[https://people.csail.mit.edu/yichangshih/wide\\_angle\\_portrait/](https://people.csail.mit.edu/yichangshih/wide_angle_portrait/)

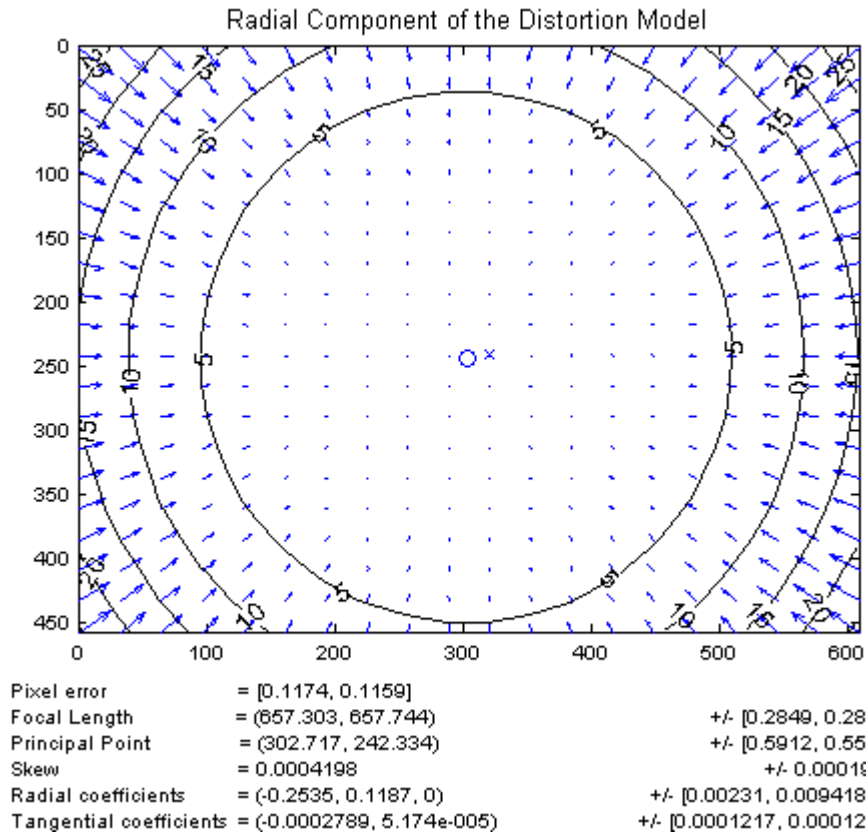
# Distortion



- Radial distortion of the image
  - Caused by imperfect lenses
  - Deviations are most noticeable for rays that pass through the edge of the lens



# Radial distortion



- Arrows show motion of projected points relative to an ideal (distortion-free lens)

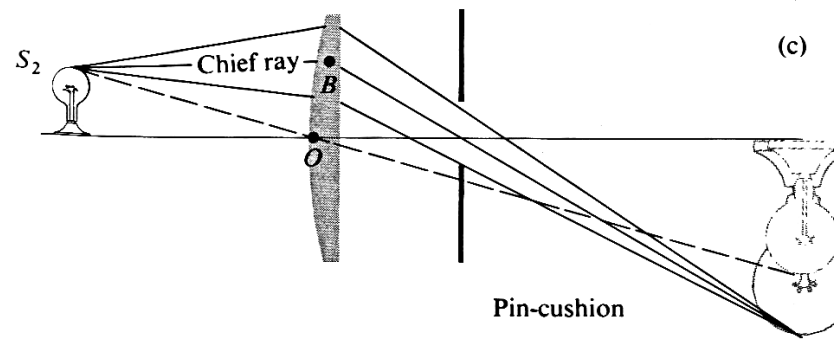
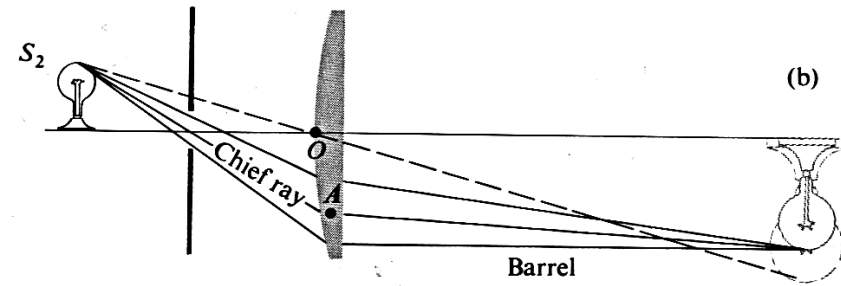
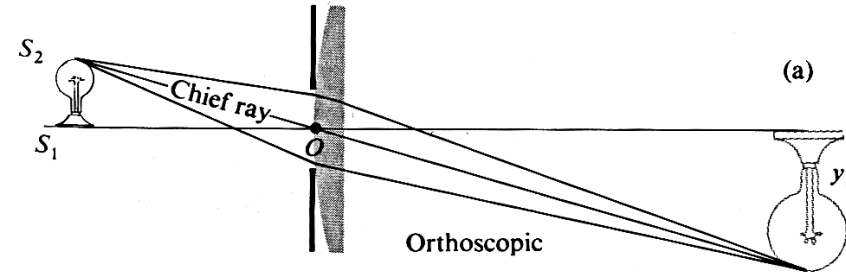
[Image credit: J. Bouguet [http://www.vision.caltech.edu/bouguetj/calib\\_doc/htmls/example.html](http://www.vision.caltech.edu/bouguetj/calib_doc/htmls/example.html)]

# Correcting radial distortion



from [Helmut Dersch](#)

# Distortion



# Modeling distortion

Project  $(\hat{x}, \hat{y}, \hat{z})$   
to "normalized"  
image coordinates

$$x'_n = \hat{x} / \hat{z}$$
$$y'_n = \hat{y} / \hat{z}$$

Apply radial distortion

$$r^2 = x'^2_n + y'^2_n$$
$$x'_d = x'_n (1 + \kappa_1 r^2 + \kappa_2 r^4)$$
$$y'_d = y'_n (1 + \kappa_1 r^2 + \kappa_2 r^4)$$

Apply focal length  
translate image center

$$x' = f x'_d + x_c$$
$$y' = f y'_d + y_c$$

- To model lens distortion
  - Use above projection operation instead of standard projection matrix multiplication

# Other types of projection

- Lots of intriguing variants...
- (I'll just mention a few fun ones)



# 360 degree field of view...



- Basic approach
  - Take a photo of a parabolic mirror with an orthographic lens (Nayar)
  - Or buy one a lens from a variety of omniscam manufacturers...
    - See <http://www.cis.upenn.edu/~kostas/omni.html>

# Tilt-shift

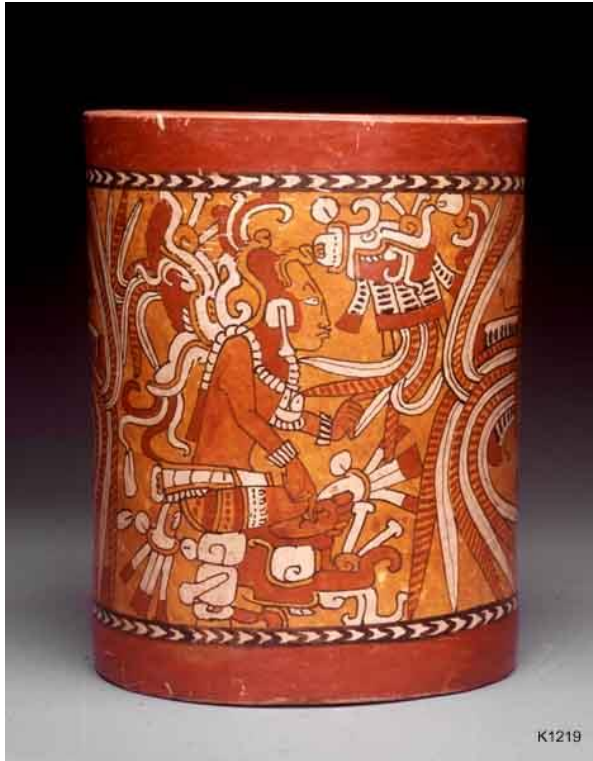


[http://www.northlight-images.co.uk/article\\_pages/tilt\\_and\\_shift\\_ts-e.html](http://www.northlight-images.co.uk/article_pages/tilt_and_shift_ts-e.html)



Tilt-shift images from [Olivo Barbieri](#)  
and Photoshop [imitations](#)

# Rotating sensor (or object)



Rollout Photographs © Justin Kerr

<http://research.famsi.org/kerrmaya.html>

Also known as "cyclographs", "peripheral images"

**Questions?**