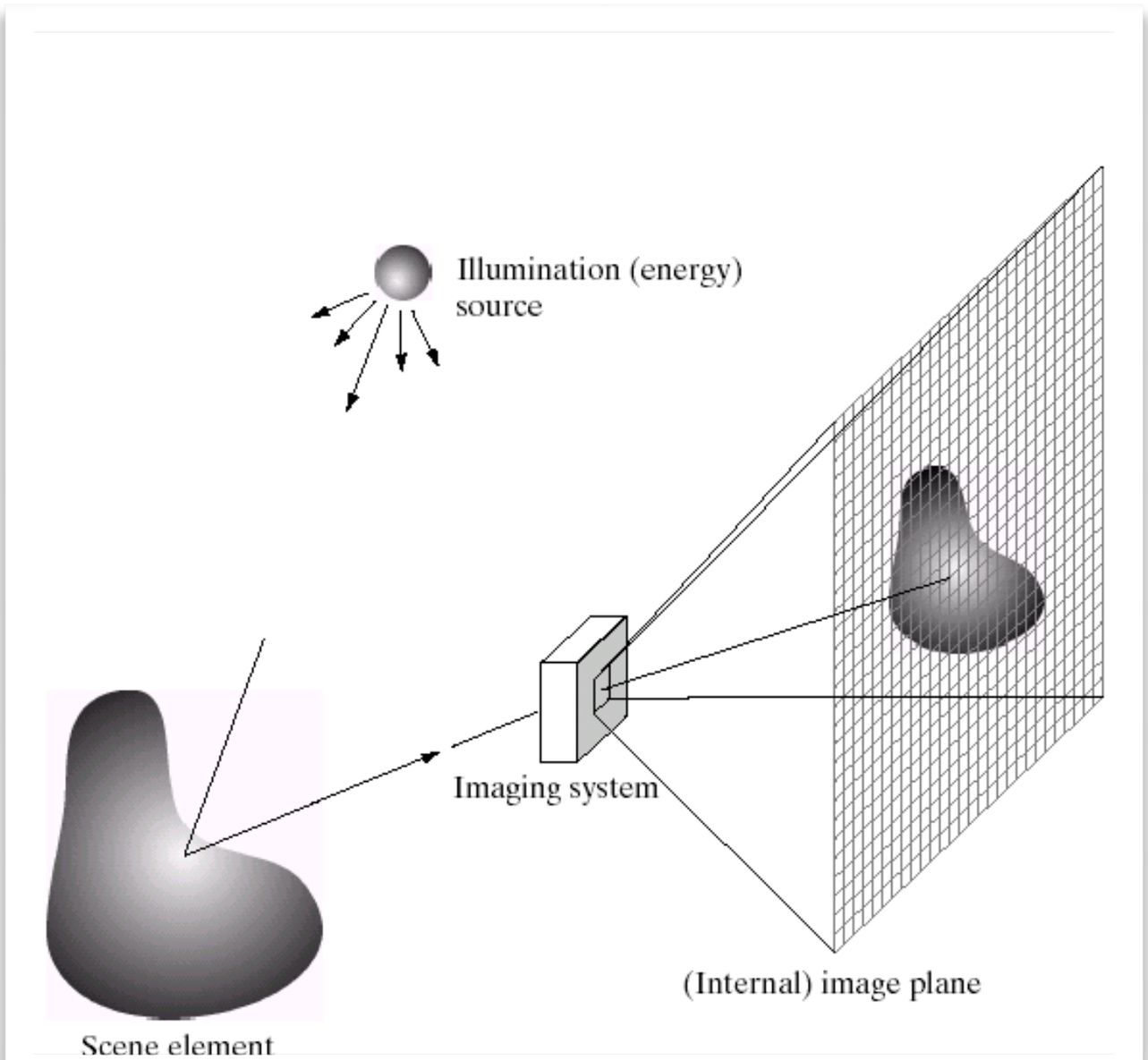
Image formation



Global course recap

Date	Topic	References	Misc
1/16	Introduction		[First class]
1/18	Filters		Special Recitation on HW0; NSH 3305 at 5pm
1/23	Edges		HW0 - Python Intro (not graded)
1/25	Texture		
1/30	Linear algebra review		
2/1	Warping		HW1 - HOG Image Classification
2/6	Alignment	Deva in town!	
2/8	Frequency	Guest lecture	
2/13	Correspondence		
2/15	Descriptors		HW2 - LK Tracking
2/20	Image Formation		
2/22	Cameras		
2/27	Motion		
2/29	Flow		HW3 - Homographies
3/5	NO CLASS [Spring Break]		
3/7	NO CLASS [Spring Break]		
3/12	Two-view		
3/14	Stereo		
3/19	SFM		
3/21	Recognition		HW4 - Reconstruction
3/26	Classifiers	Guest Lecture	
3/28	MLPs	Guest Lecture	
4/2	Training Recipes		
4/4	CNNs		
4/9	Interpreting Networks		HW5 - Neural Networks
4/11	NO CLASS [Spring Carnival]		
4/16	Radiometry		
4/18	Color		
4/23	Recent Trends		
4/25	Catchup		HW6 - Photometric Stereo

Preview

From here, I simply slice the video up the same way I would with any existing video and run it through COLMAP and then into nerfstudio, but you can use any radiance field company. It works on the first try, with all camera poses found and trains like any other radiance field. Here is the resulting NeRF from Nerfacto.

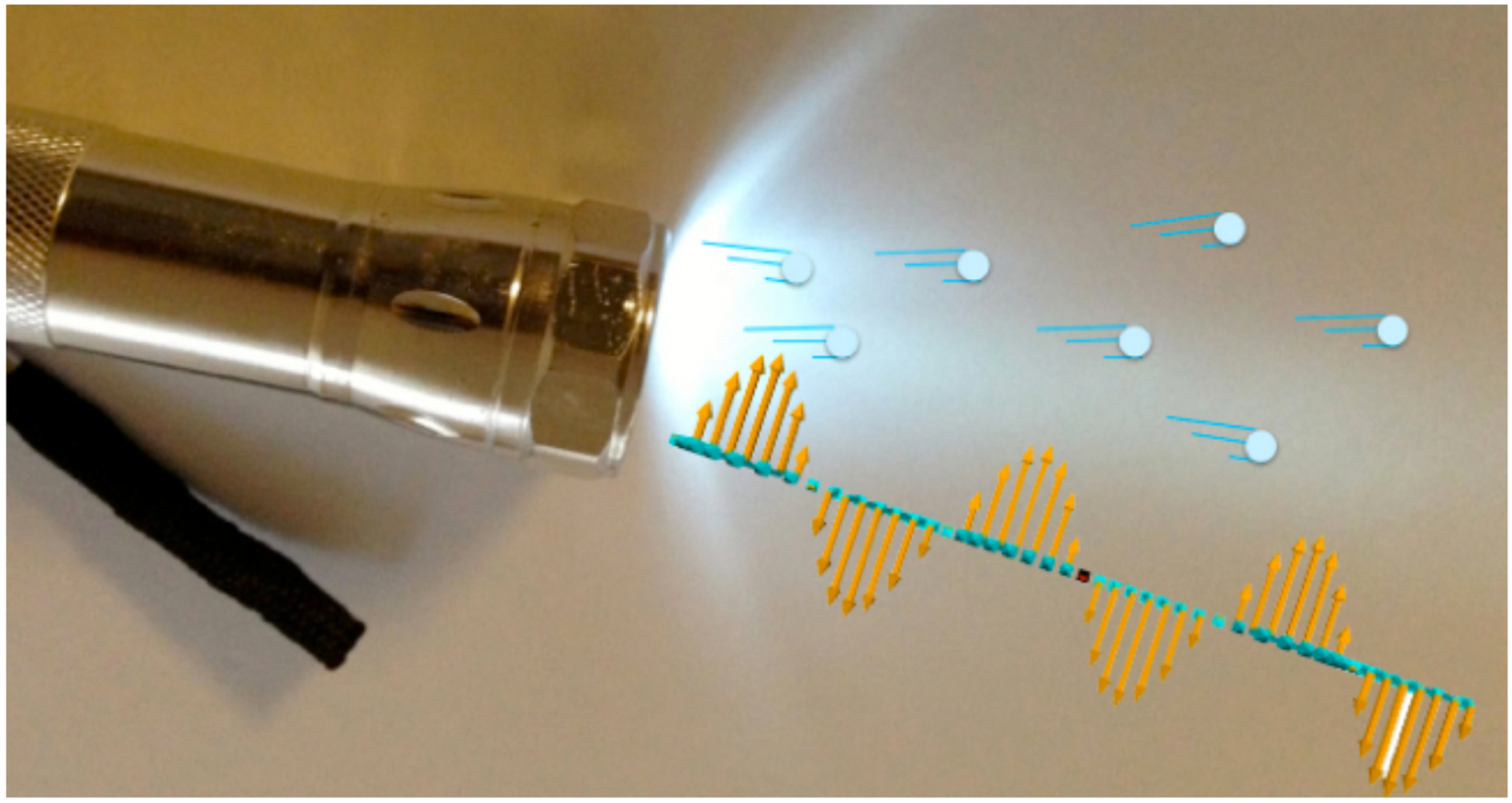
https://radiancefields.com/openai-launches-sora-and-the-world/

Agenda

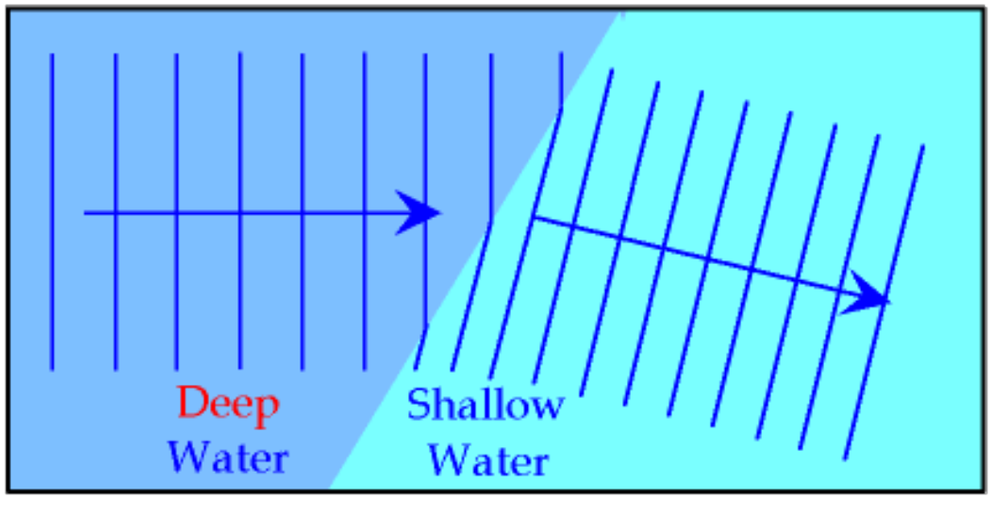
- **Pinhole optics**
  - Perspective projection (vanishing points, horizon, object height)
  - Camera matrices (intrinsics + extrinsics)
  - Homographies (2 views of plane, rotation)

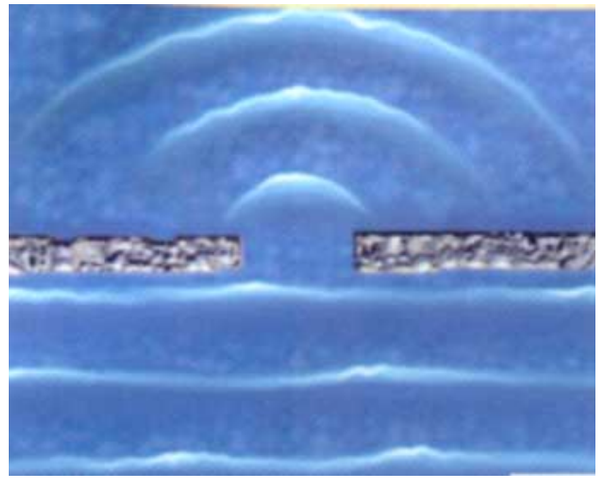
- Camera models
  - Properties of camera matrices (DOF, geometric intuition, pixel2rays)
  - Simplified cameras: orthographic, scaled orthographic, paraperspective, affine
  - Camera calibration (DLT v reprojection error)

Light as a wave + particle



Light as a wave (ignore for now)





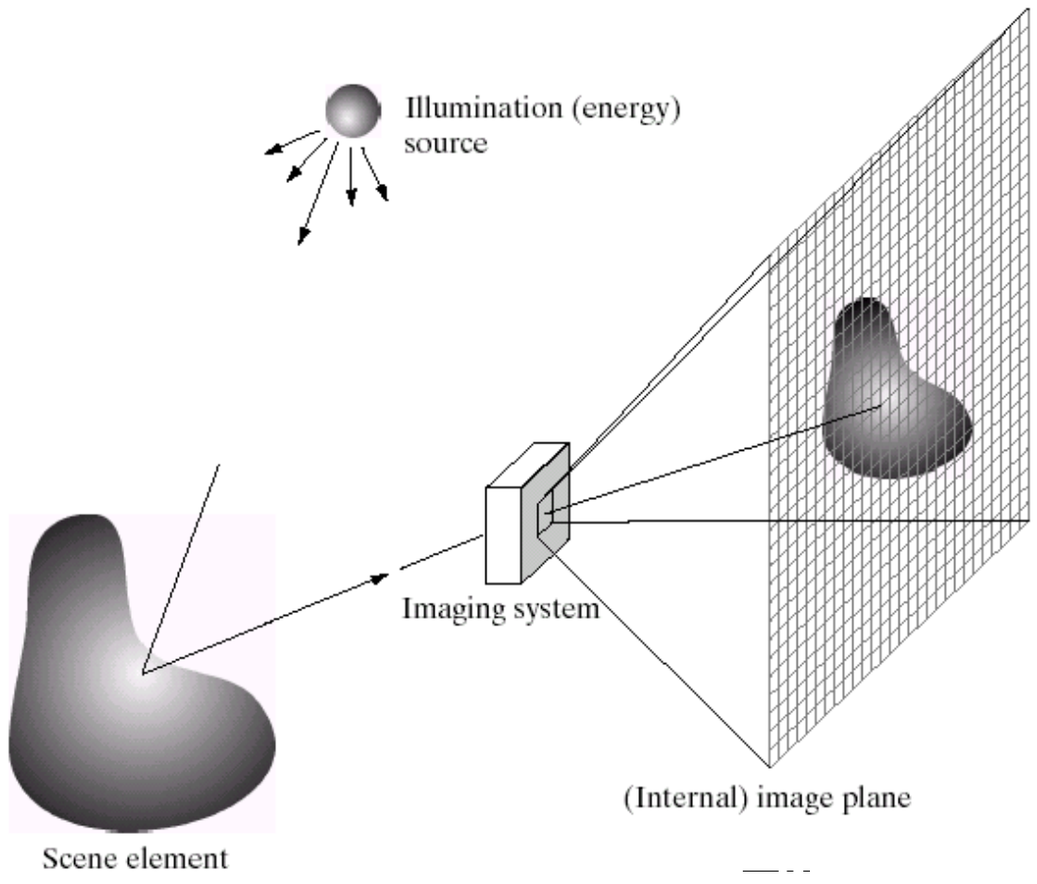
Refraction

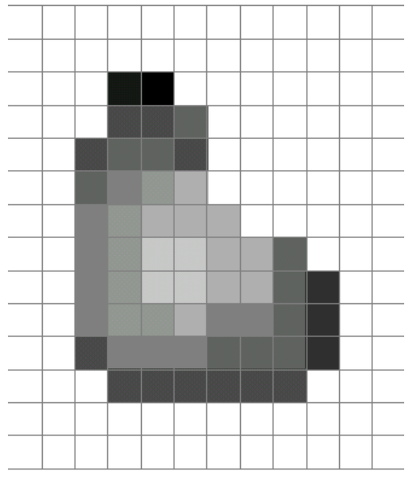
Diffraction

Embracing such phenomena allows you to do cool things like see around corners

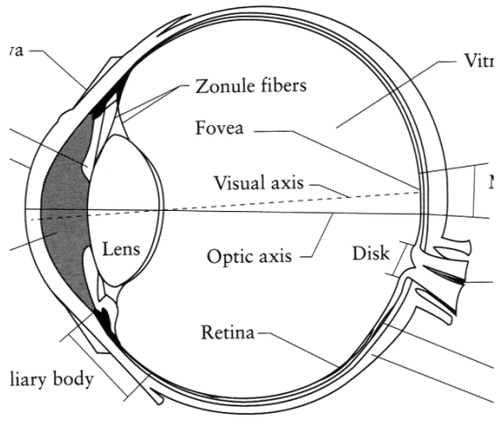
https://imaging.cs.cmu.edu/fermat_paths/

Image formation



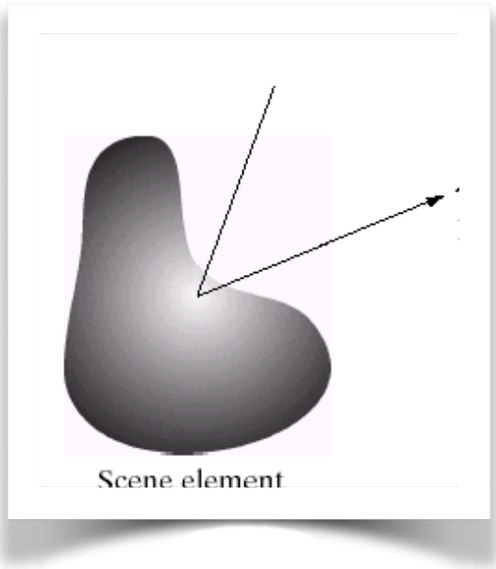


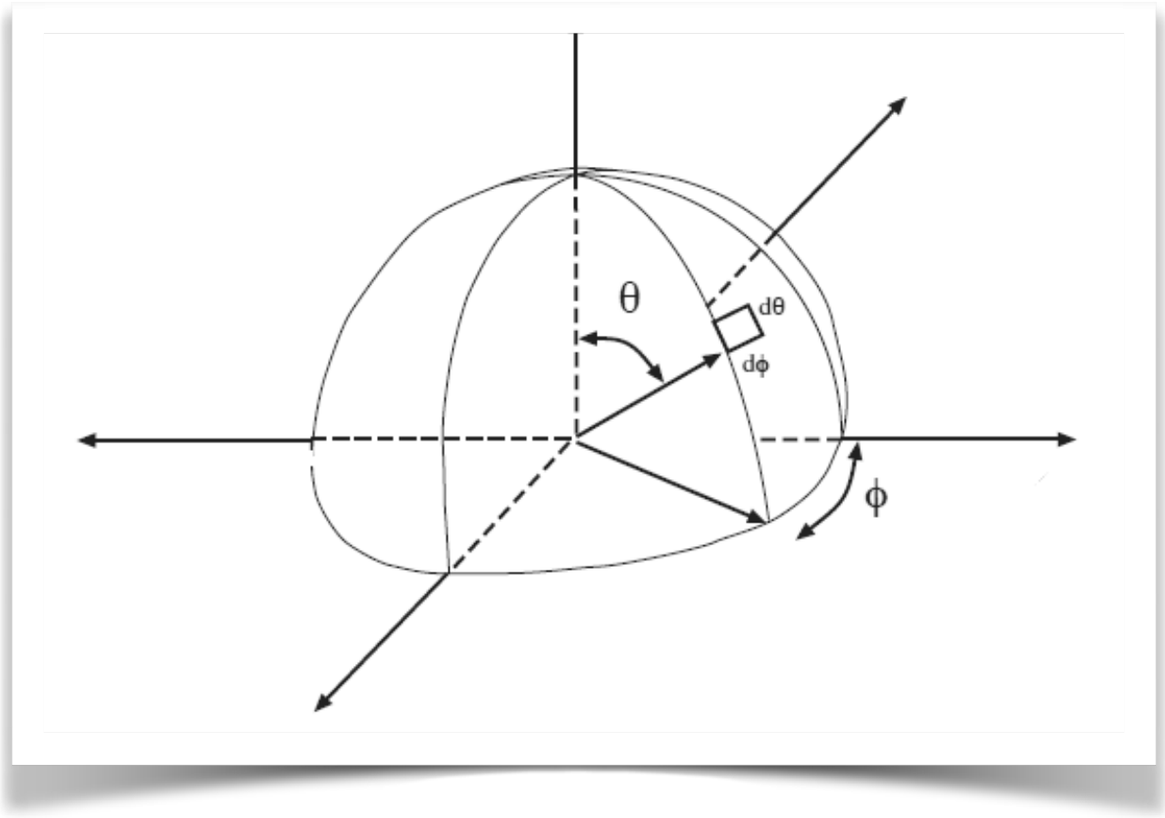
Digital Image



Human eye

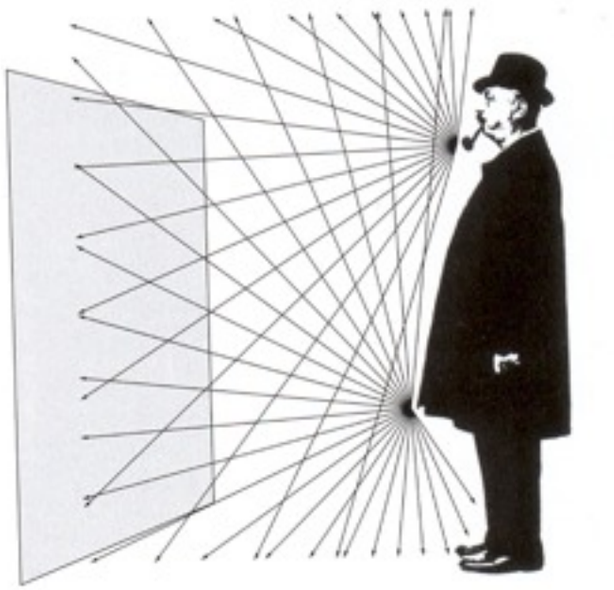
Pixel brightness

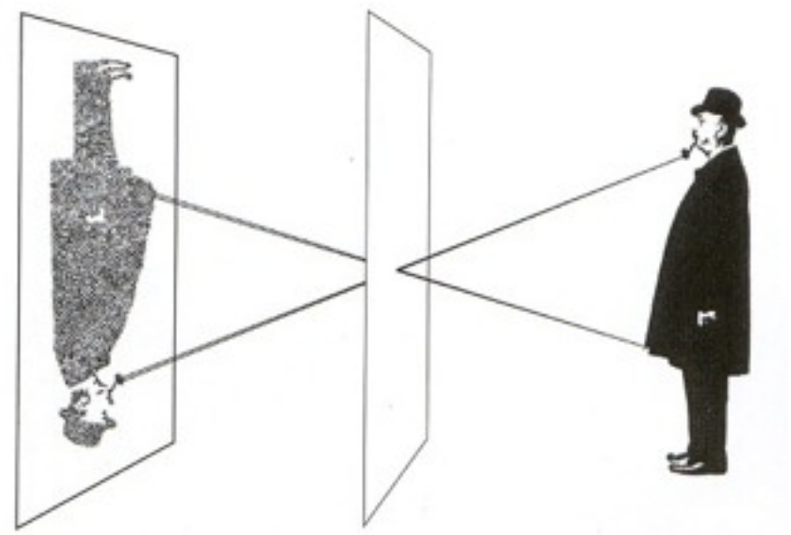




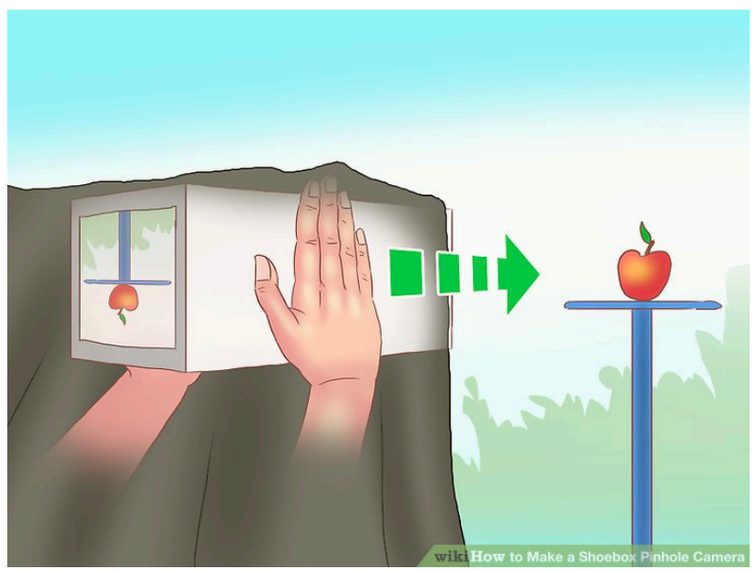
(More on “light as physics” at end of semseter)

Pinhole optics





Limit the light landing on a sensor “wall” using a pinhole

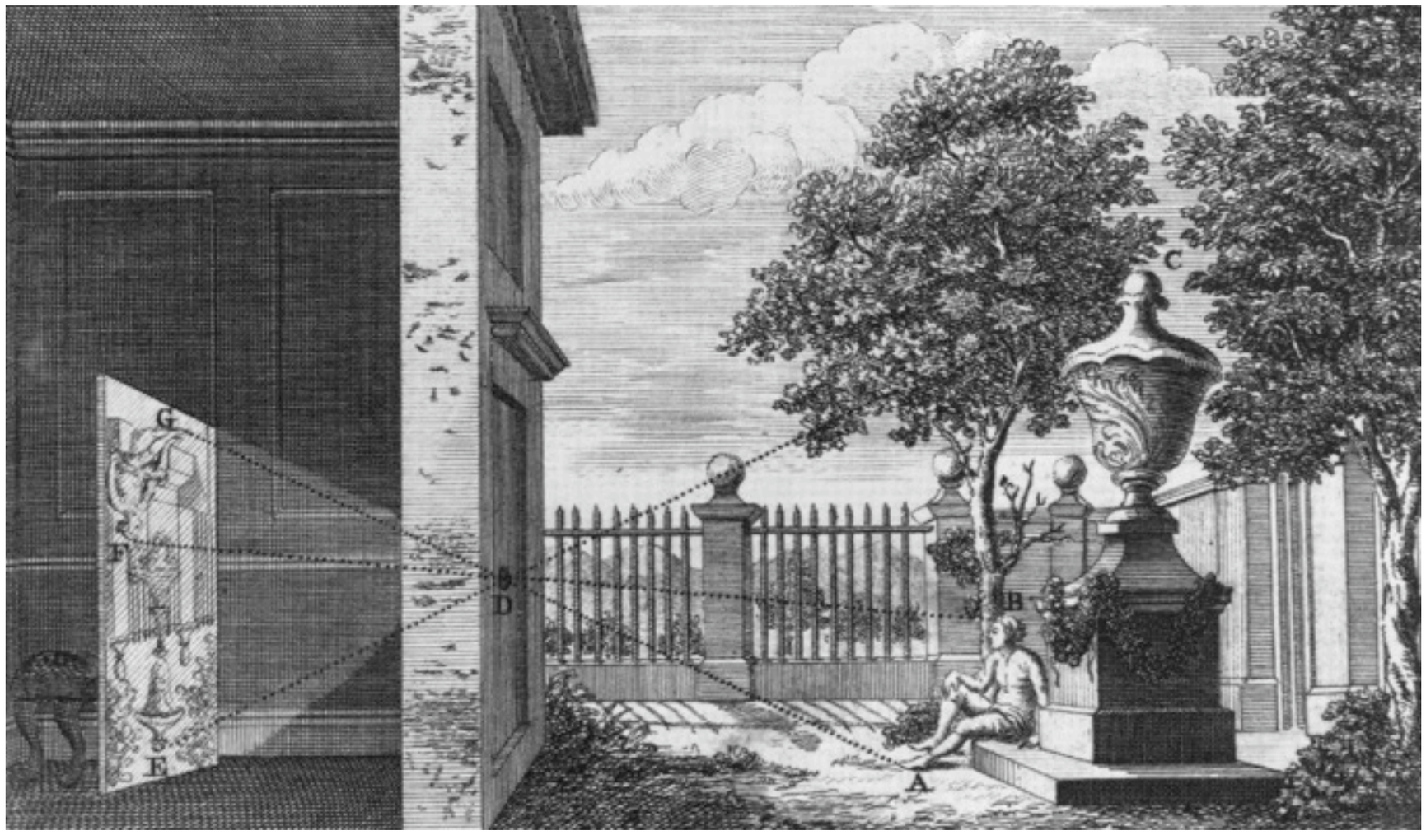


https://www.instructables.com/Shoebox-Camera/

World’s largest photograph

El Toro Marine Corps, Irvine CA 2006

Camera Obscura



So why don’t we see such images in our daily lives?

Accidental pinholes

what's the dark stuff?

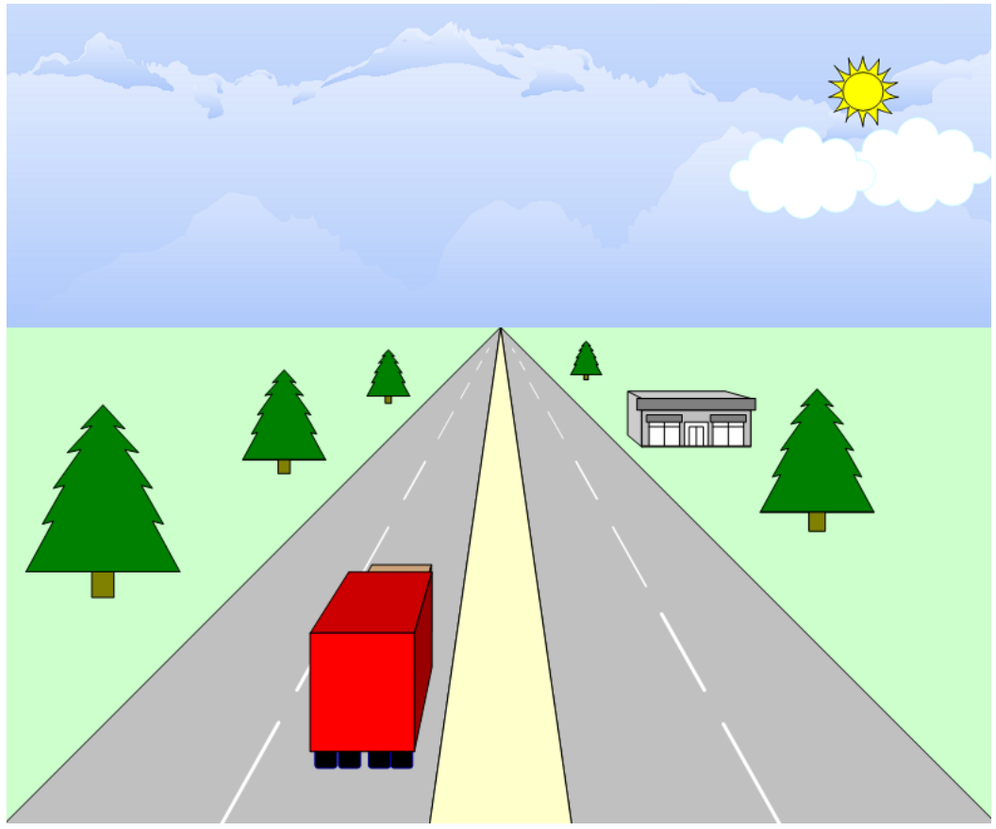
(the view from Antonio Torralba's hotel room)

Accidental pinhole and pinspeck cameras: revealing the scene outside the picture

CVPR 2012

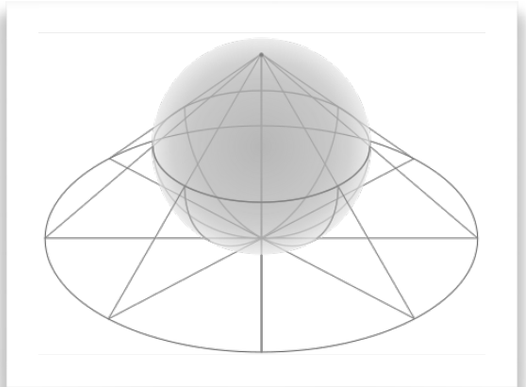
Antonio Torralba, William T. Freeman
Computer Science and Artificial Intelligence Laboratory (CSAIL)
MIT
torralba@mit.edu, billf@mit.edu

Perspective projection



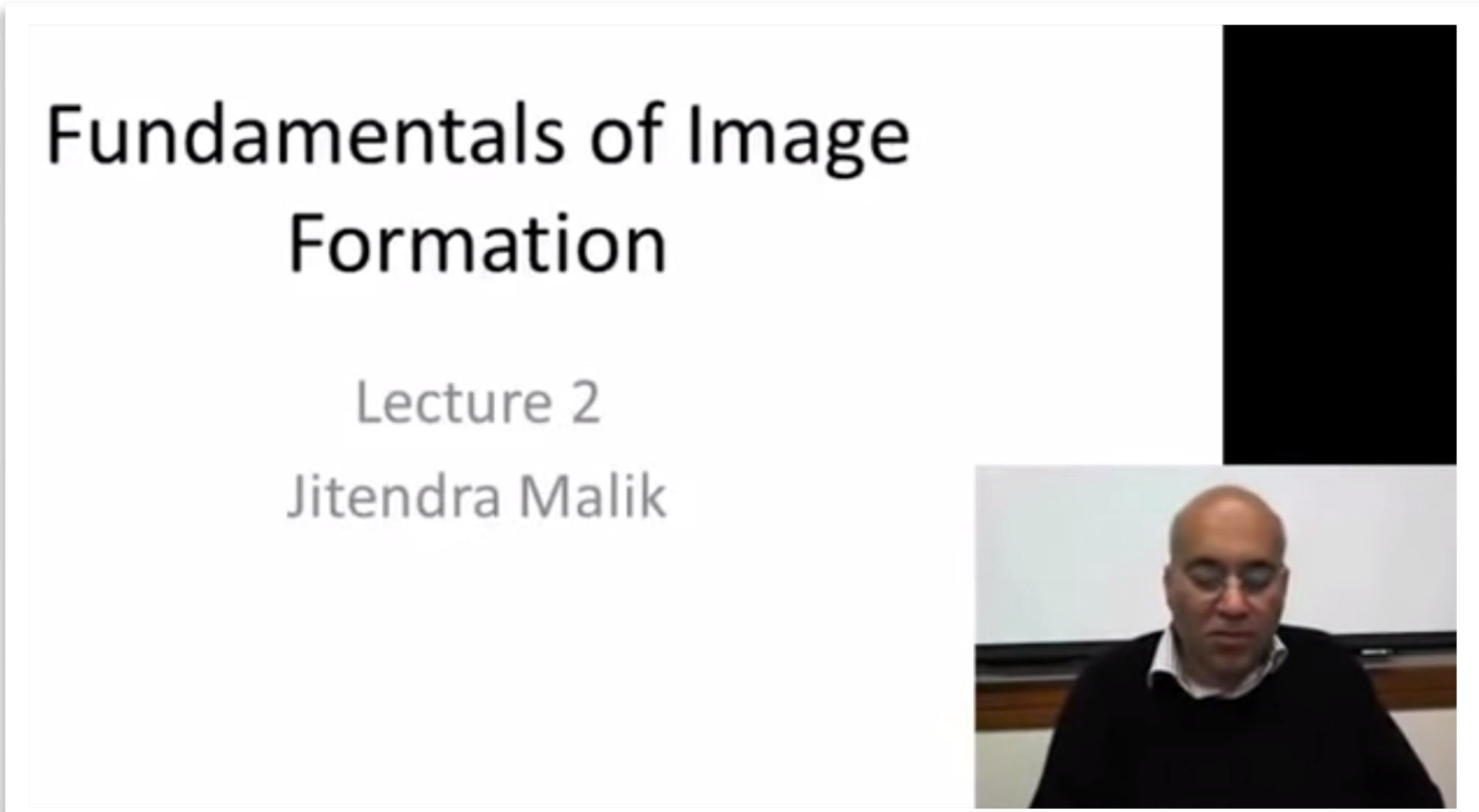
Goal for next few slides: mathematically derive important consequences of perspective projection (e.g., paralel lines meet, closer objects are larger) … *without* machinery of projective geometry

https://en.wikipedia.org/wiki/Projective_geometry



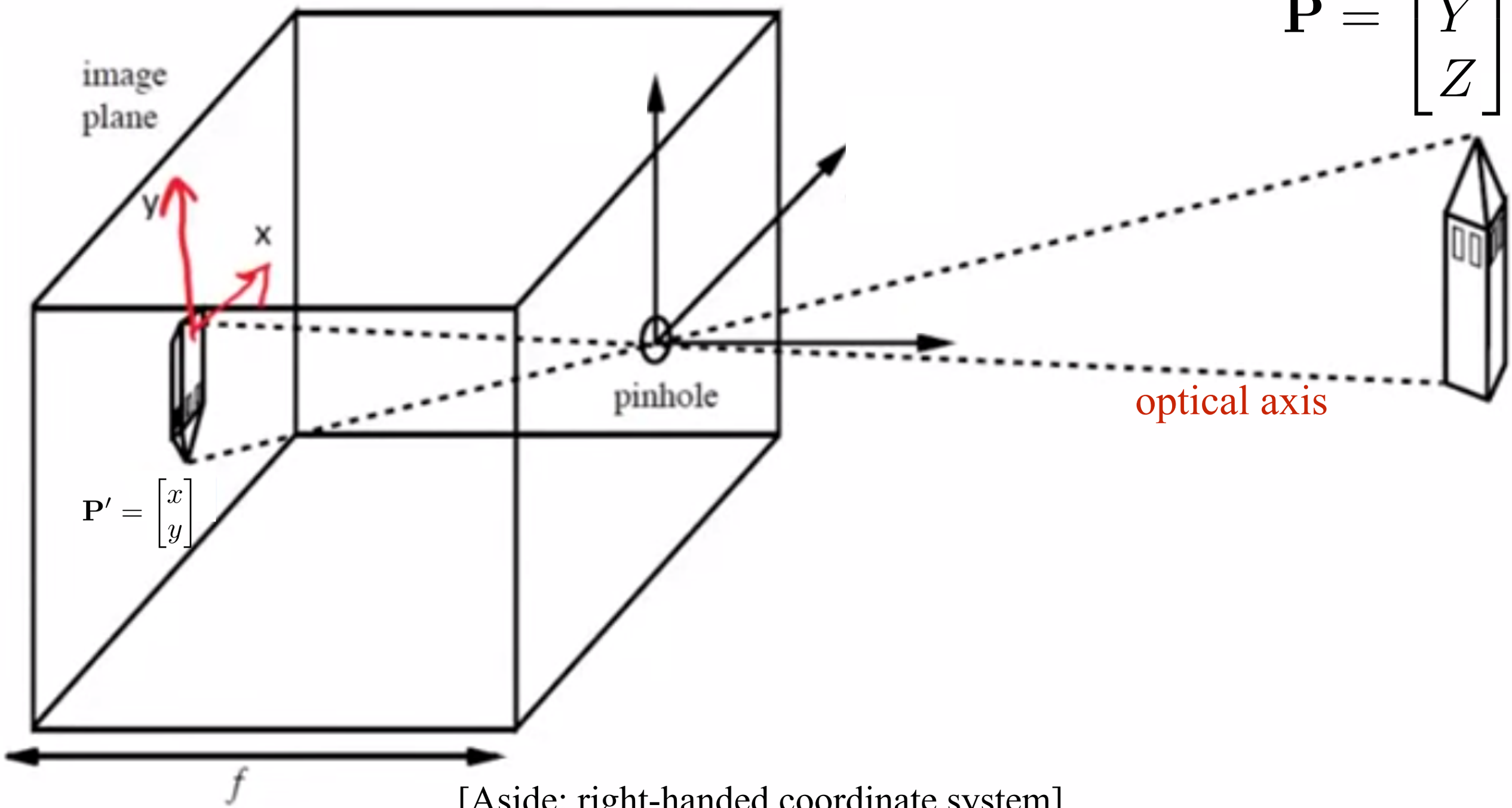
Reference for this material

(sadly no longer publically available)



Pinhole Camera

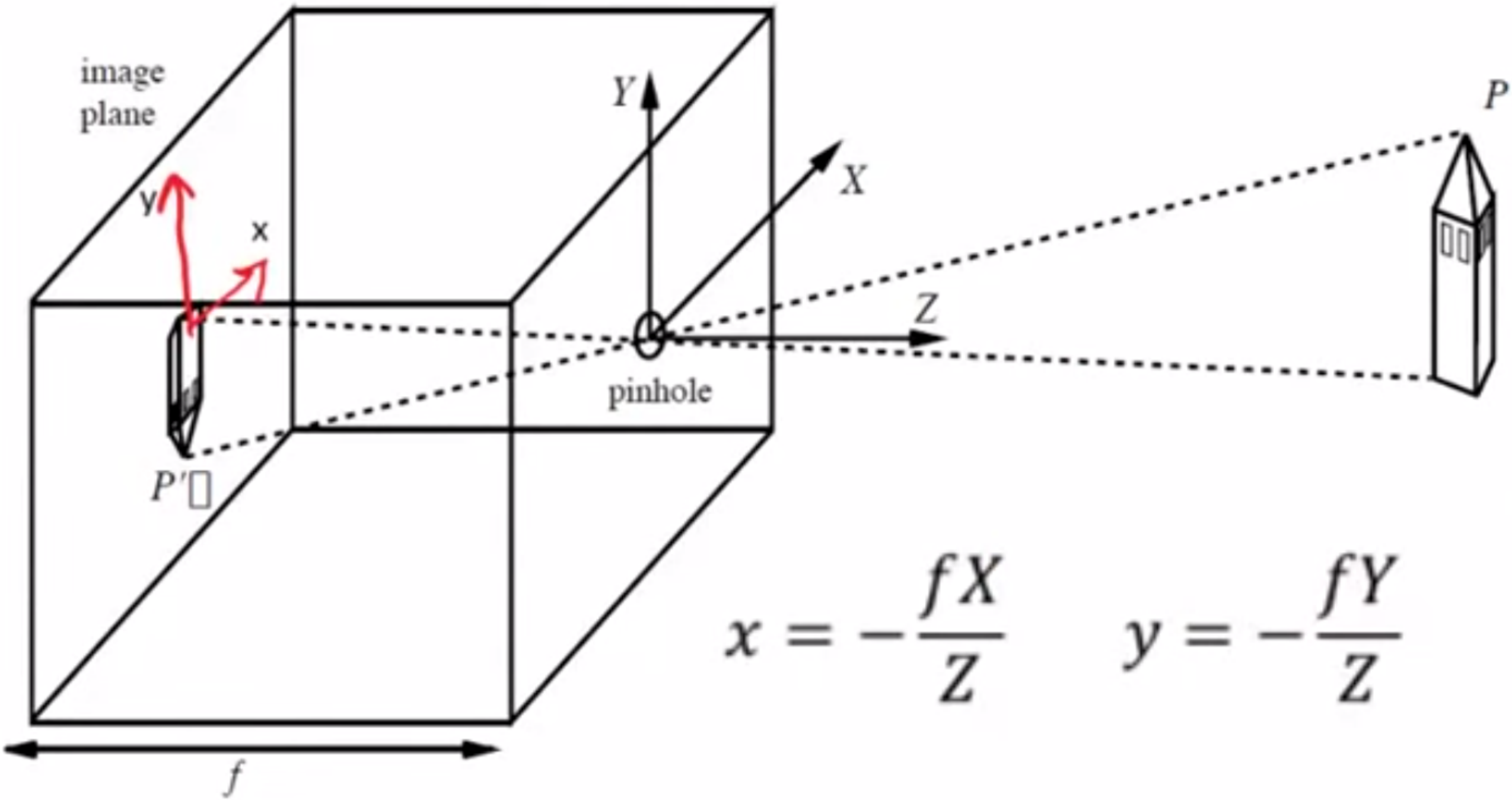
$$\mathbf{P} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$



[Aside: right-handed coordinate system]

How do we compute P'? [on board]

Pinhole Camera

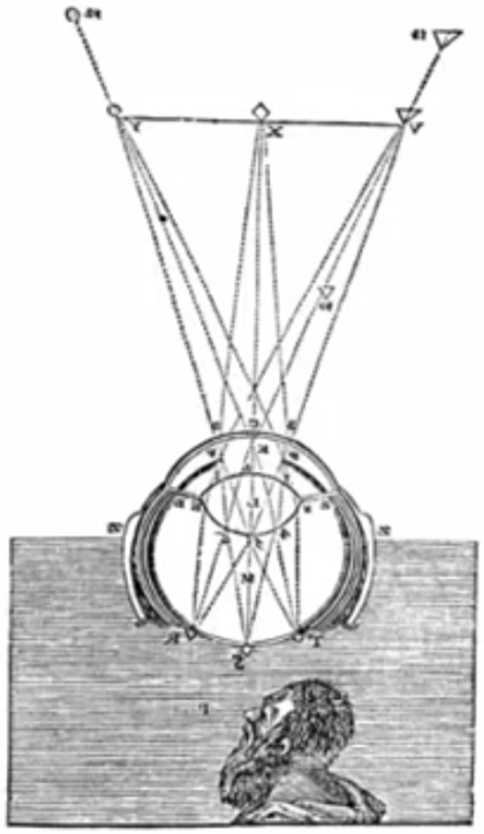


$$x = -\frac{fX}{Z} \qquad y = -\frac{fY}{Z}$$

An incredible amount of insights can be derived from this simple formula

Annoying detail: image inversion

Why don’t we see an upside-down world?

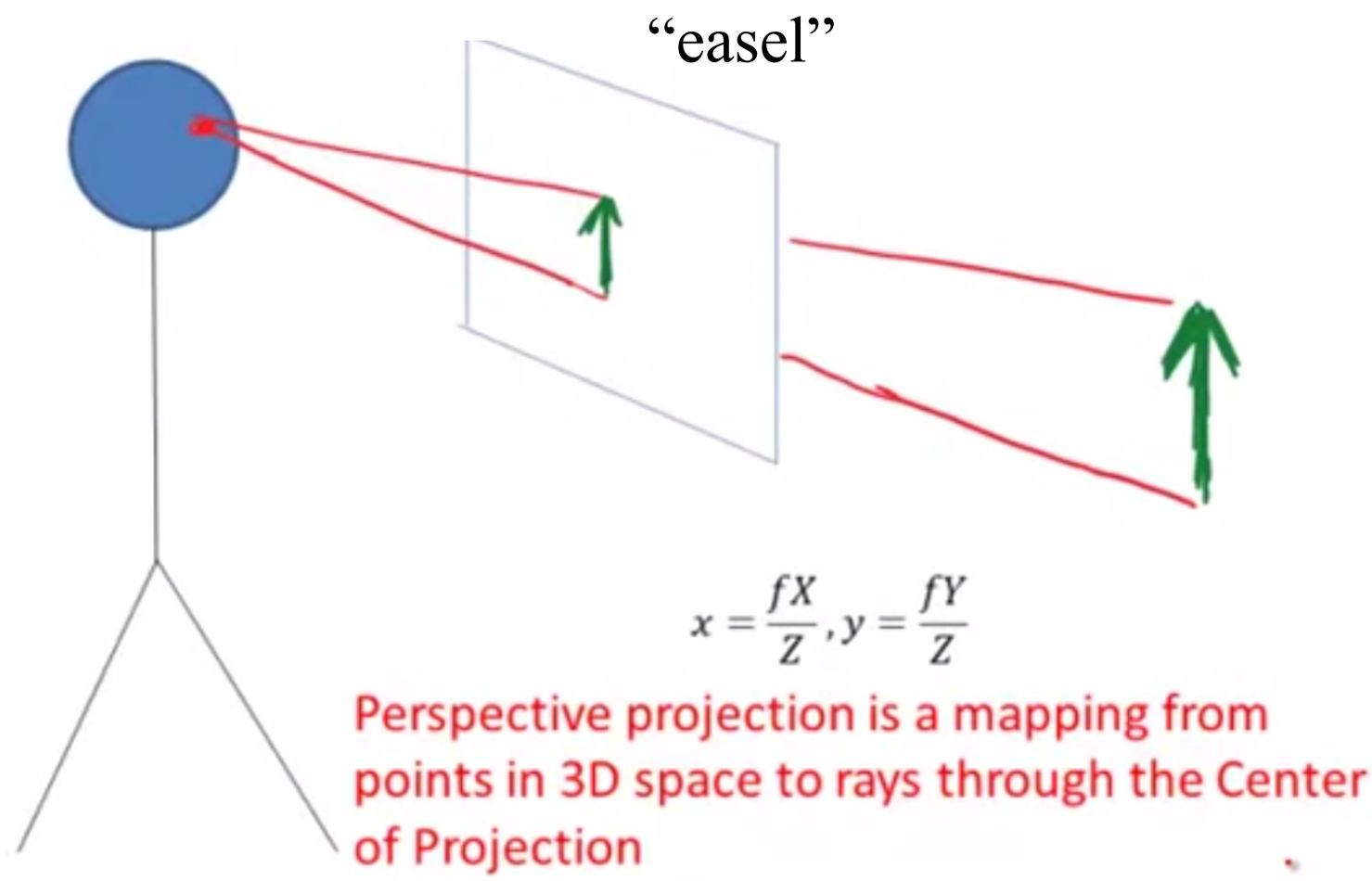


From Descartes(1937), La Dioptrique

Perplexed folks for a while. But software/brain can simply invert this.

$$x = \frac{fX}{Z} \qquad y = \frac{fY}{Z}$$

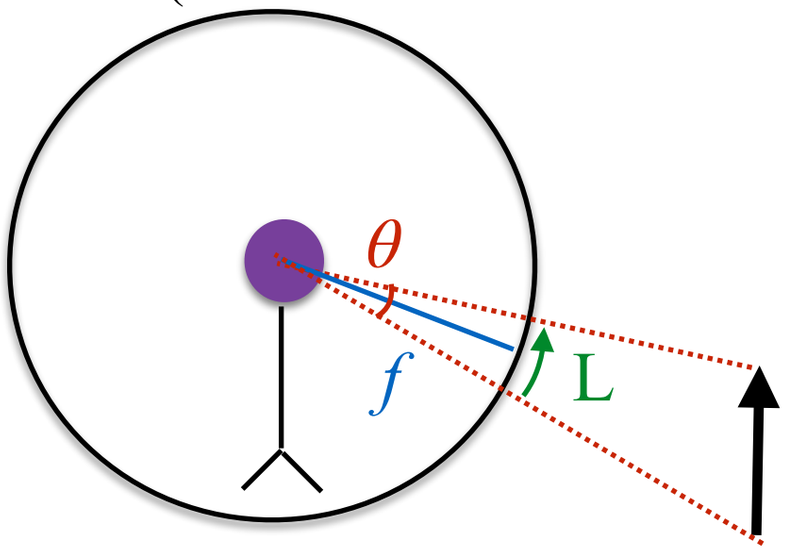
Physical model that avoids inversion

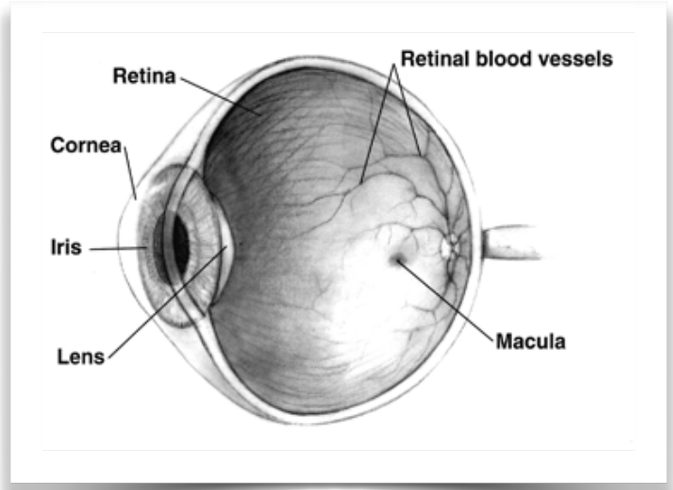


COP = center of projection, pinhole, camera center (such terms will be used interchangeably)
Distance of COP to easel = focal length

Visual angle

(common unit in human vision)





Math is easier for a *spherical* easel (e.g., retina)

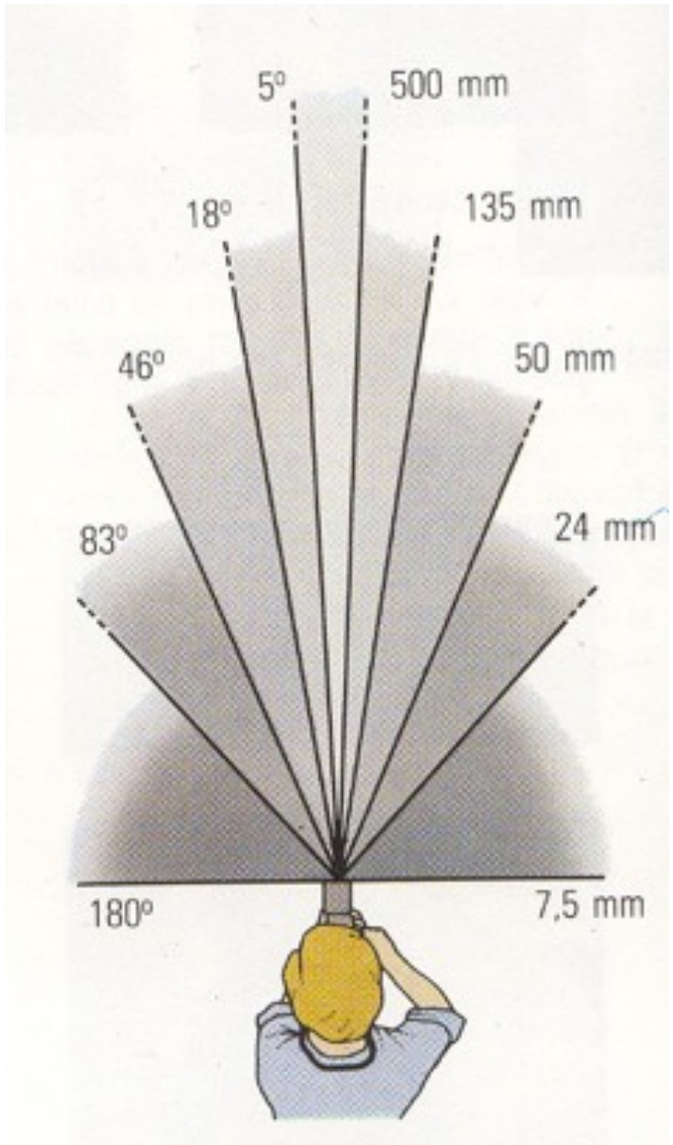
L = length of projection on sphere
θ = units of radians
f = radius (focal length)

$$\frac{\theta}{2\pi} = \frac{L}{circumference} = \frac{L}{2\pi f} \quad \Rightarrow \quad \theta = \frac{L}{f}$$

Human head is 9 inches high. At a distance of 9 feet, it "subtends"
1/12 radians = 4.8 degrees, *regardless of image focal length*

Field of view (FOV)

Apply previous formula to determine the angle subtended by the entire sensored image



24mm

50mm

135mm

$$\theta = \frac{L}{f} \quad \Rightarrow \quad \text{FOV} \approx \frac{\text{total sensor size (diagonal)}}{\text{focal length}} \quad \text{(in radians)}$$

"As we increase the focal length, we decrease the feild-of-view"

Aside: precise formula for flat image plane can be easily derived by "x = fX/Z"

Field of view (FOV)

Reference for flat image plane

Consider a **horizontal cross-section** of the pinhole camera. The right triangle is formed by:

- **One half of the sensor width** $\frac{w}{2}$
- **The focal length** f
- **The half-angle of the horizontal field of view** θ_H

From trigonometry:

$$\tan(\theta_H) = \frac{\text{half the sensor width}}{\text{focal length}} = \frac{w/2}{f}$$

Solving for θ_H:

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

Since FoV is the full angle:

$$\text{Horizontal FoV} = 2\theta_H = 2\tan^{-1}\left(\frac{w}{2f}\right)$$

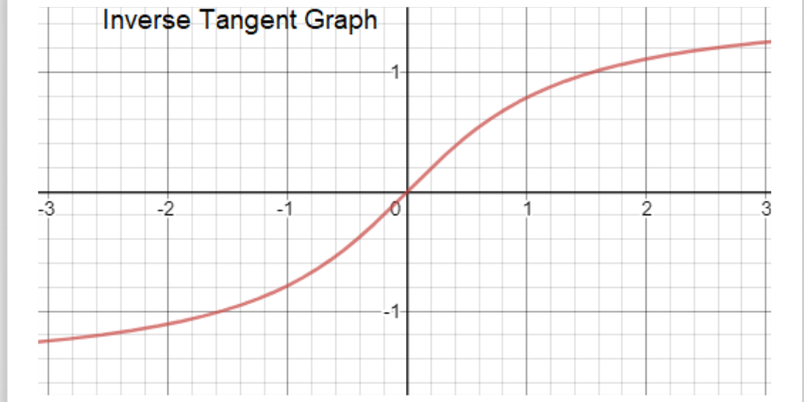
A similar derivation applies to the **vertical FoV** using the sensor height h:

$$\text{Vertical FoV} = 2\tan^{-1}\left(\frac{h}{2f}\right)$$

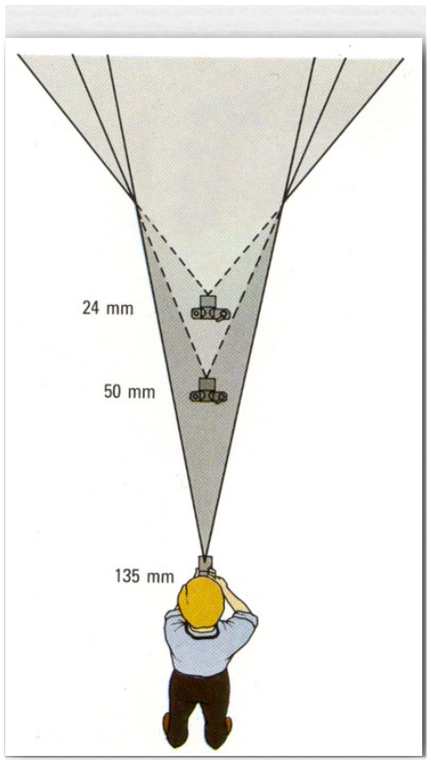
For the **diagonal FoV**, we use the diagonal of the sensor:

$$d = \sqrt{w^2 + h^2}$$

$$\text{Diagonal FoV} = 2\tan^{-1}\left(\frac{d}{2f}\right) = 2\tan^{-1}\left(\frac{\sqrt{w^2+h^2}}{2f}\right)$$



Increasing the focal length and stepping back



What happens to apparant object size and FOV when we double distance to object and double the focal length?

$$x_{new} = \frac{2fX}{2Z} = \frac{fX}{Z} = x_{old}$$

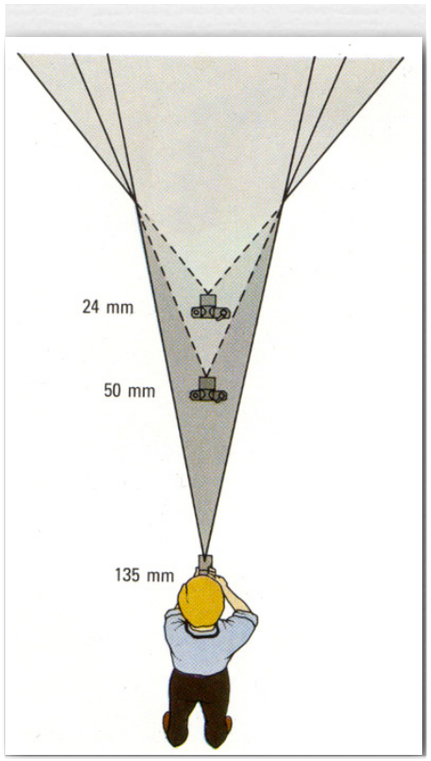
$$FOV_{new} = \frac{\text{sensor size}}{2f} = \frac{1}{2} FOV_{old}$$

Size of object in image remains the same, but FOV decreases.

Turns out this is a common visual effect…

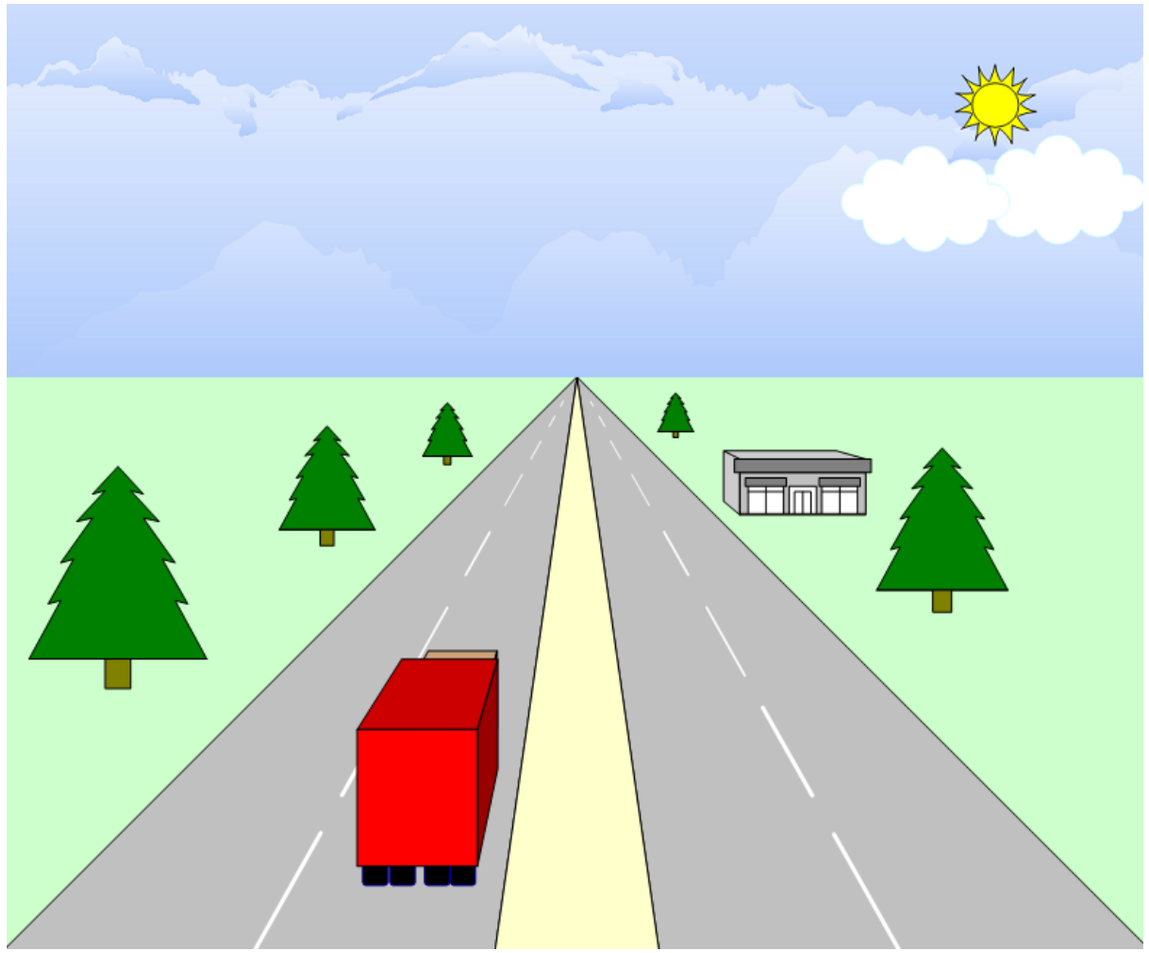
Decreasing the focal length and moving forward

"dolly" effect in cinematography



(see demo_renderCube.ipynb)

Perspective projection



Goal for next few slides: mathematically derive the following consequences of perspective projection

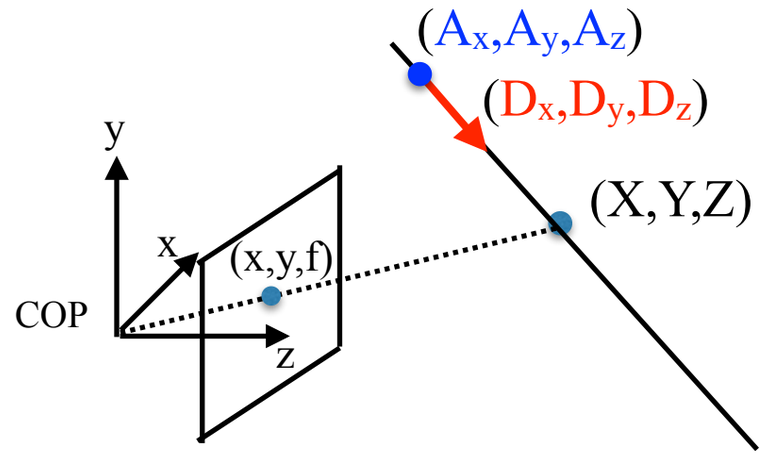
- Closer objects appear larger
- Closer objects are lower in the image
- Parallel lines meet

All these can be simply derived with $x = f\frac{X}{Z}$!

Parallel lines meet: proof

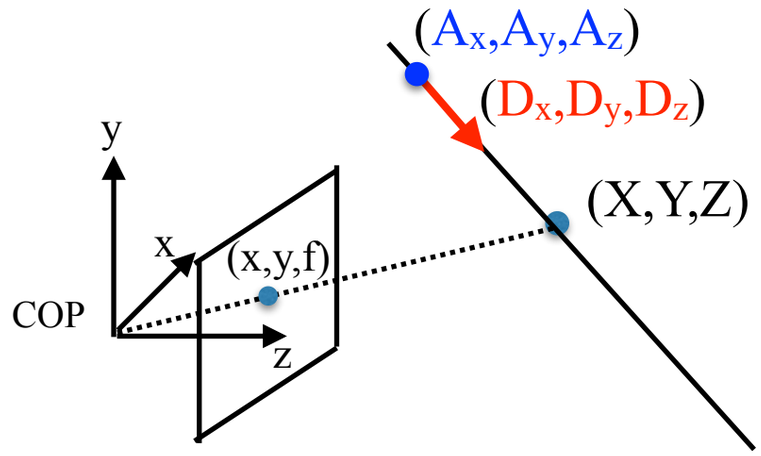
How do we write the equation of a 3D line?

Define an 3D anchor point and a 3D direction



Parallel lines meet: proof

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} A_x \\ A_y \\ A_z \end{bmatrix} + \lambda \begin{bmatrix} D_x \\ D_y \\ D_z \end{bmatrix}$$



Compute projected point (x,y) as lambda approaches infinity [on board]:

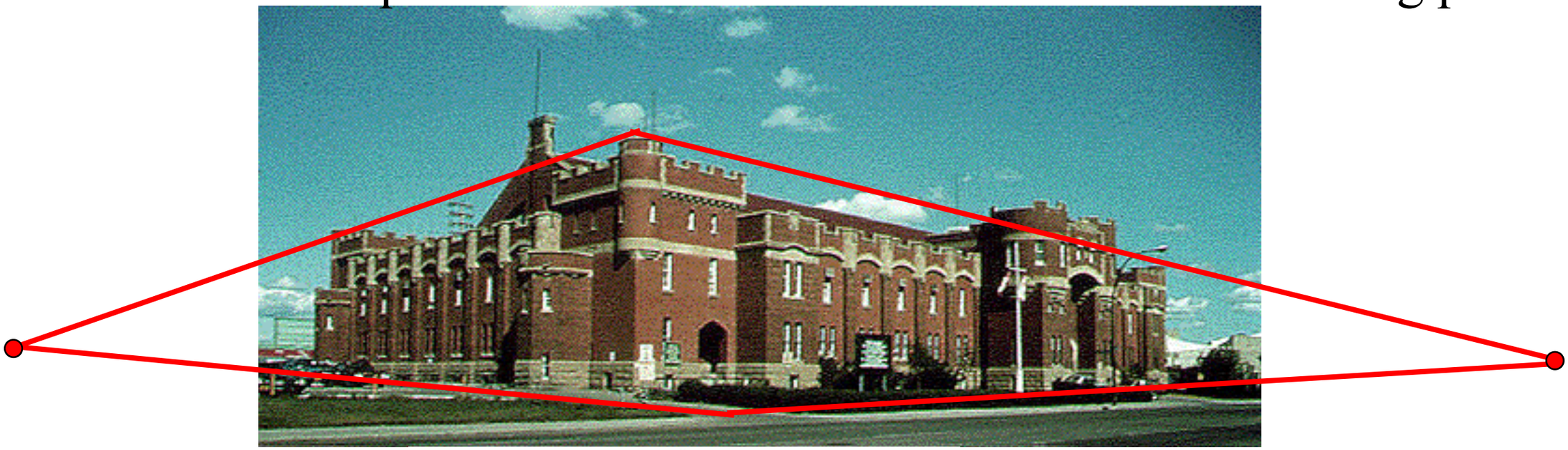
$$x = \frac{fX}{Z} = \frac{f(A_x + \lambda D_x)}{A_z + \lambda D_z} \rightarrow \frac{fD_x}{D_z} \quad \text{as} \quad \lambda \rightarrow \infty \qquad \text{(ditto for y)}$$

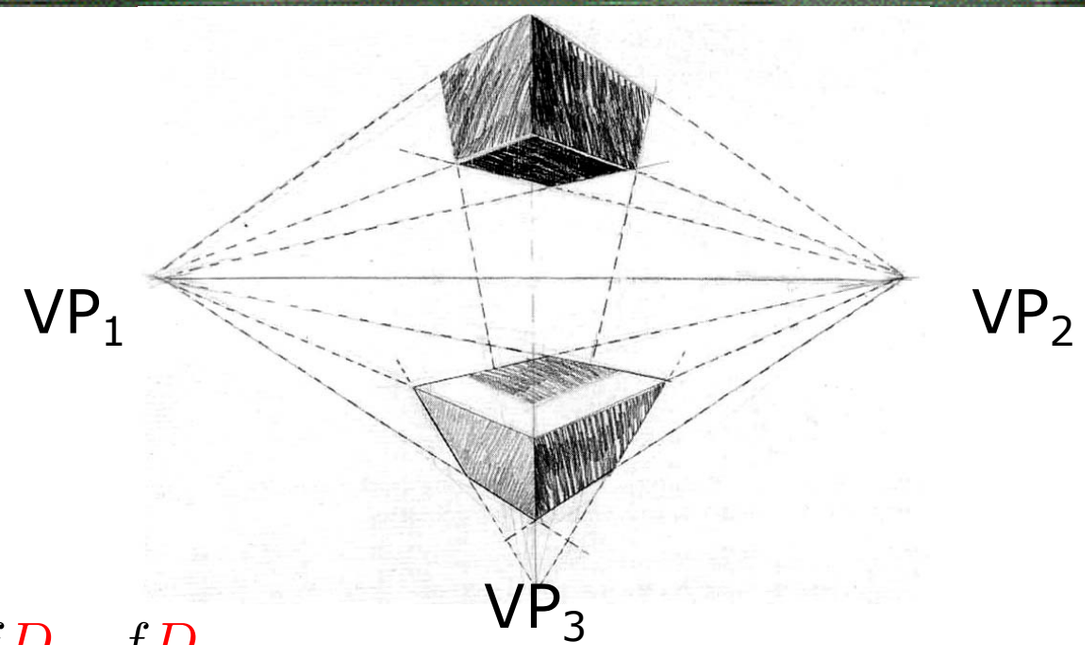
$$(x, y) \rightarrow (\frac{fD_x}{D_z}, \frac{fD_y}{D_z}) \quad \text{as} \quad \lambda \rightarrow \infty$$

- Parallel 3D lines (with identical direction vectors) meet at same 2D image location or “vanishing point”

Special case: manhatten world

Consider a “city-block” world where all lines follow one of 3 directions.
Previous derivation implies that all lines will meet at on of 3 vanishing points.



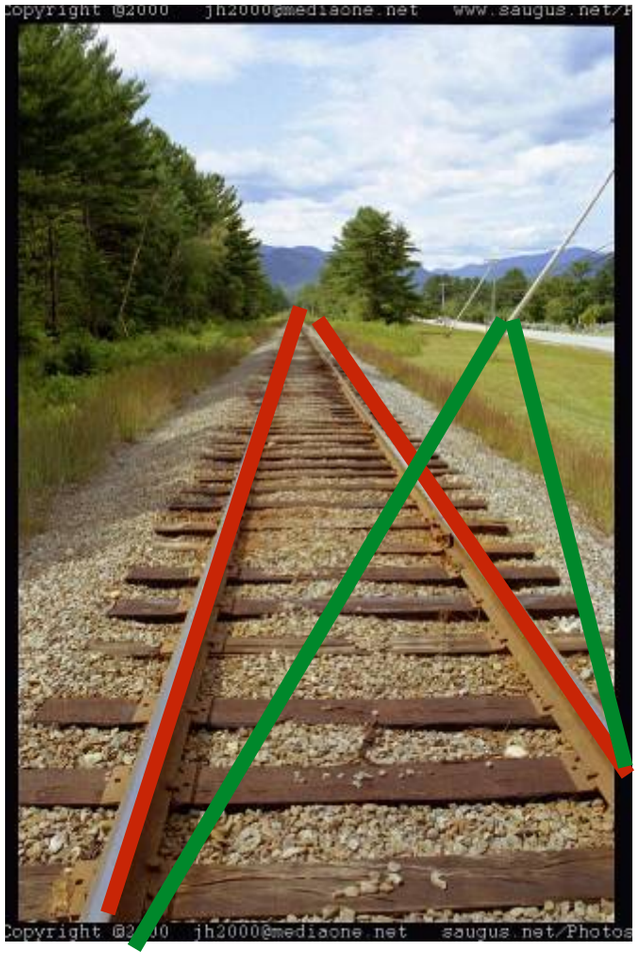


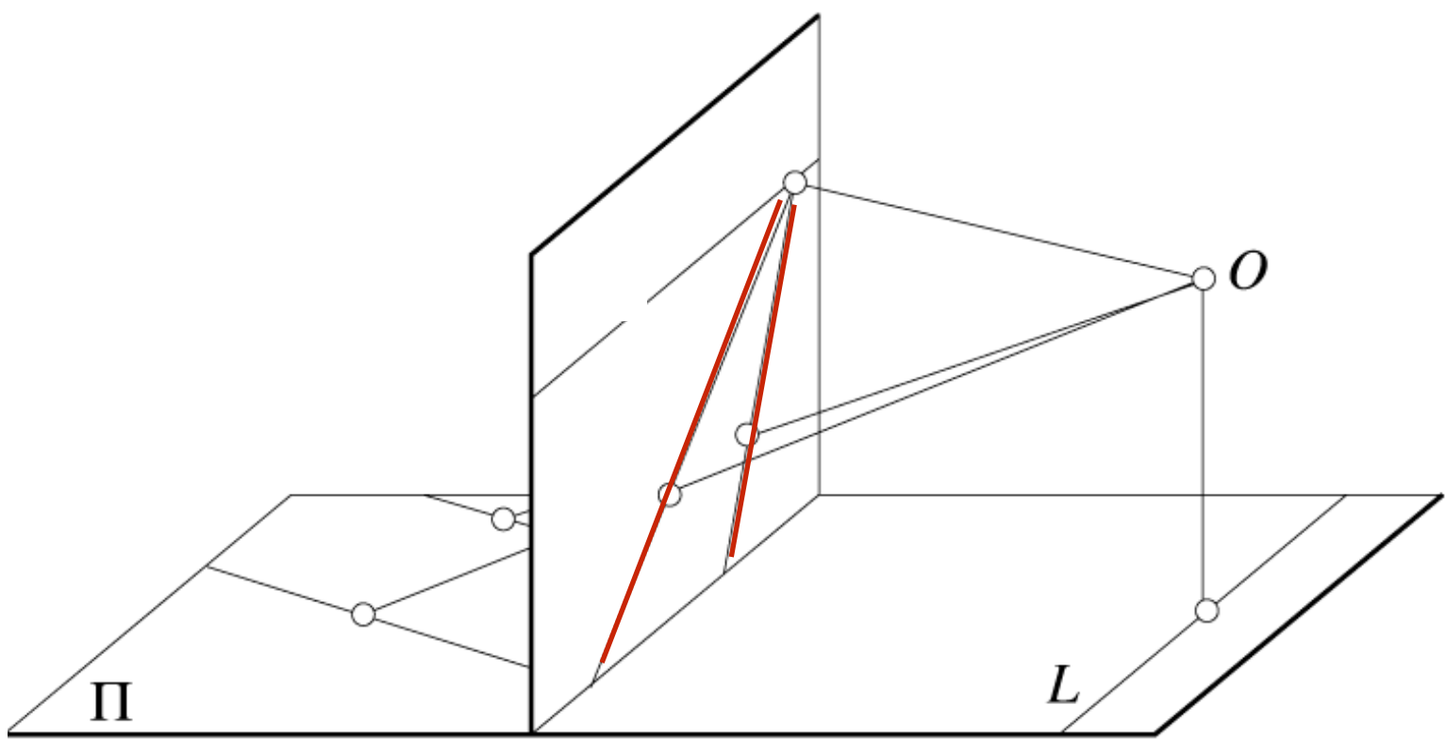
Vanishing point $(\frac{f D_x}{D_z}, \frac{f D_y}{D_z})$ can extend infinitely far beyond image plane as D_z -> 0

It will be useful to talk about *different* points at infinity (motivation for projective geometry in 16-822)

Special case: horizon

What if tracks were oriented 45 degrees to the right. Where would they meet?



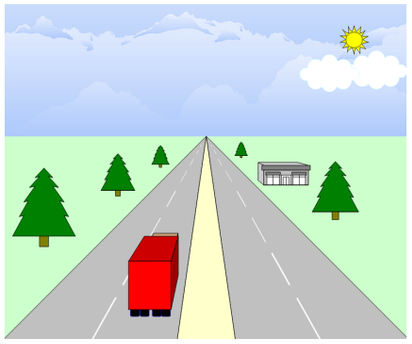


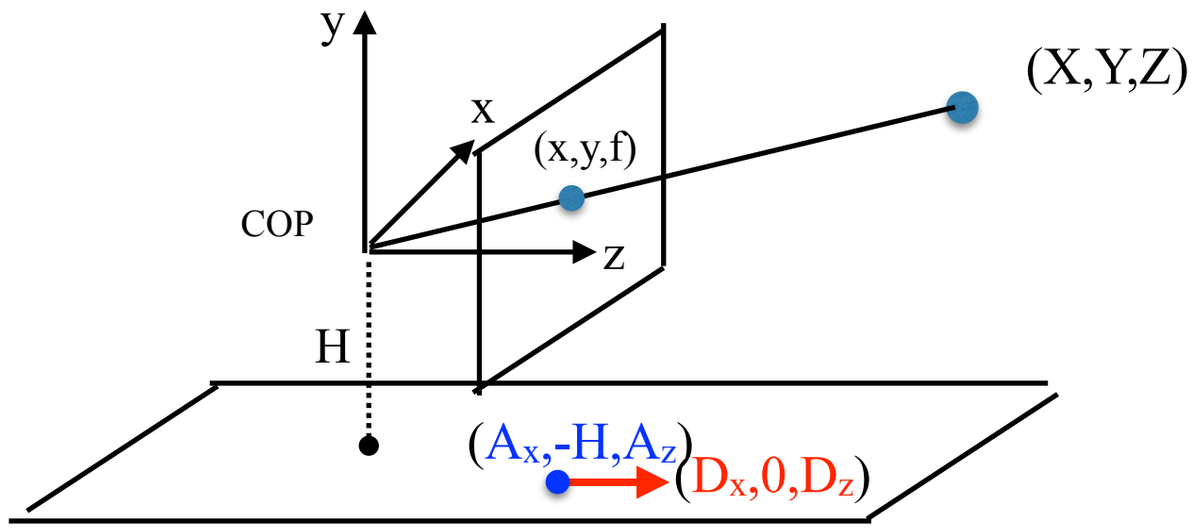
Claim: all 3D lines on ground plane meet at a *set* of vanishing points called the *horizon line*

Horizon line: proof

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} A_x \\ A_y \\ A_z \end{bmatrix} + \lambda \begin{bmatrix} D_x \\ D_y \\ D_z \end{bmatrix} \qquad (x, y) \rightarrow (\frac{f D_x}{D_z}, \frac{f D_y}{D_z}) \quad \text{as} \quad \lambda \rightarrow \infty$$

Eqn of ground plane is Y = -H





Lines on ground plane have an anchor (A_x,-H,A_z) with direction (D_x,0,D_z).
Where are their vanishing points?

$$(\frac{f D_x}{D_z}, 0)$$

Question: why is horizon line not always at center of image?

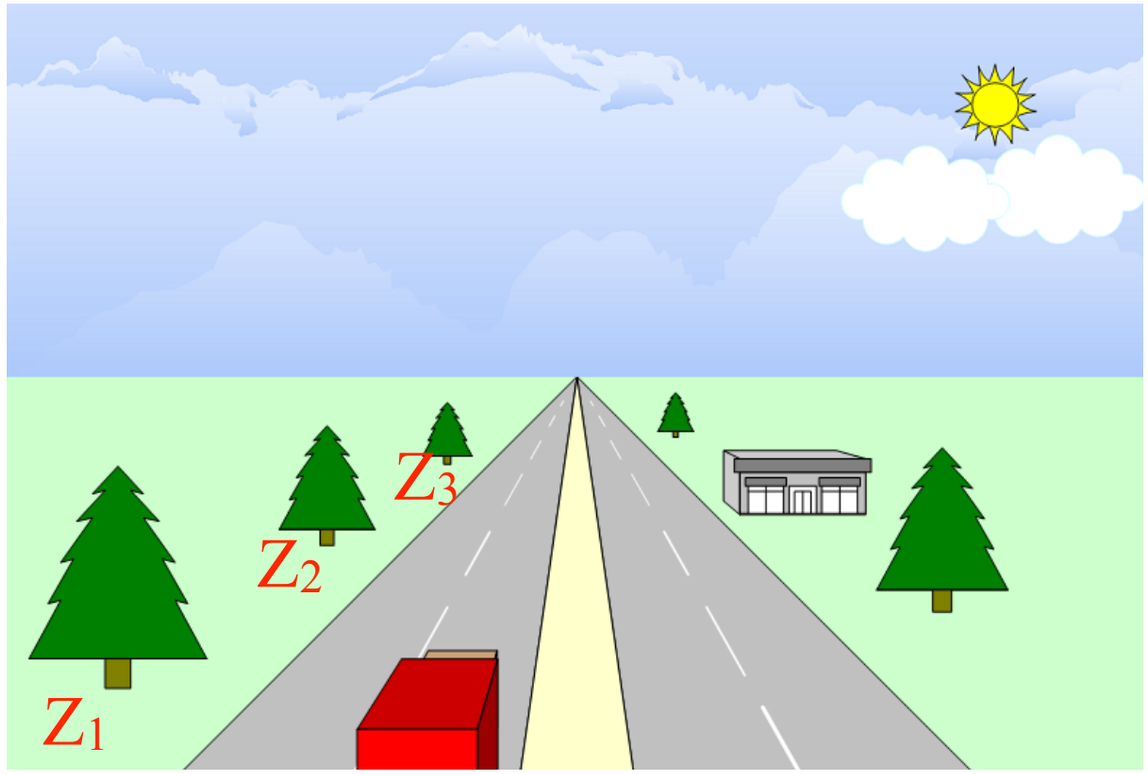
If our head is tilted up or down, than equation of ground plane in camera coordinate system isn't Y = -H

Image y position: proof

Closer objects are lower in the image

Equation of ground plane is Y = -H
A point on ground plane will have y-coordinate=?

$$y = f\frac{Y}{Z} = -f\frac{H}{Z}$$

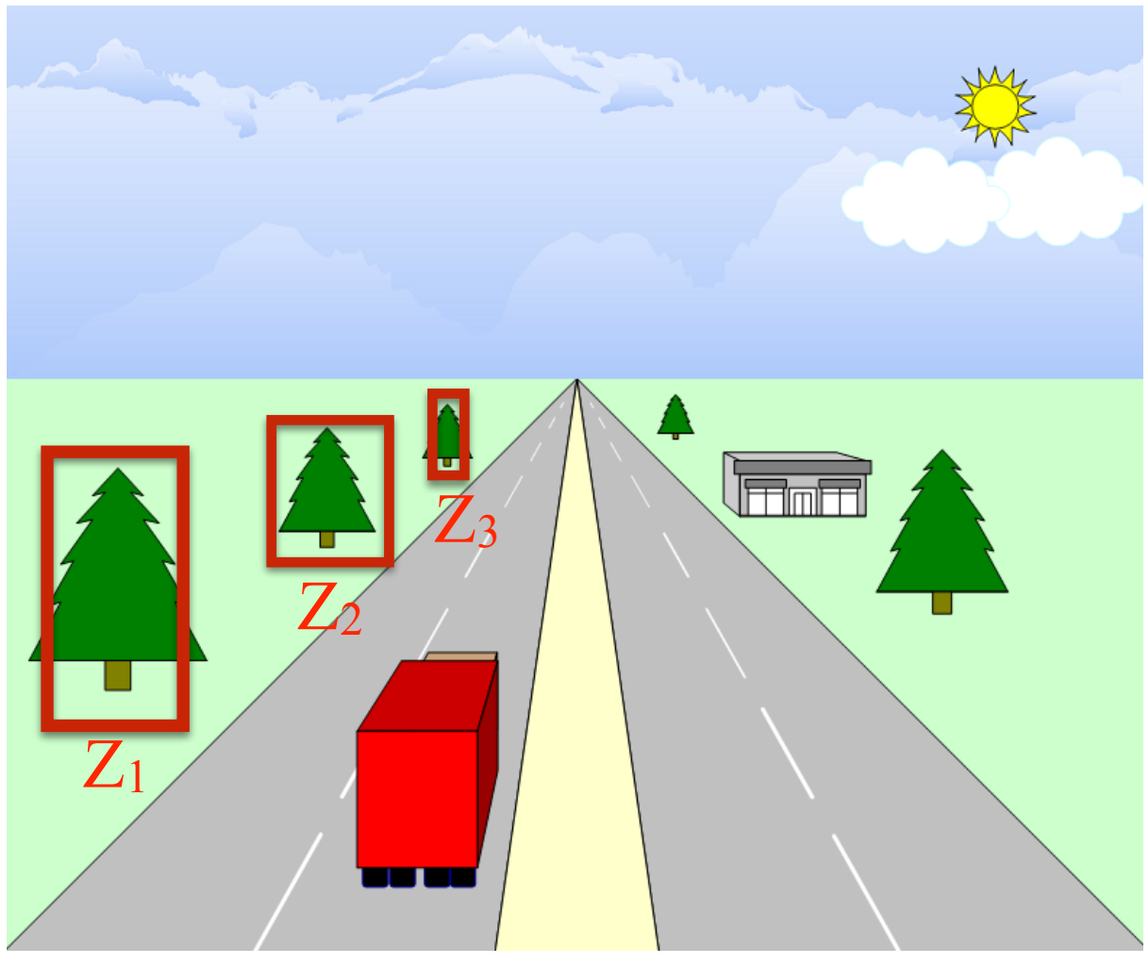


(objects with large Z will appear on horizon)

Image height: proof

Closer objects appear larger

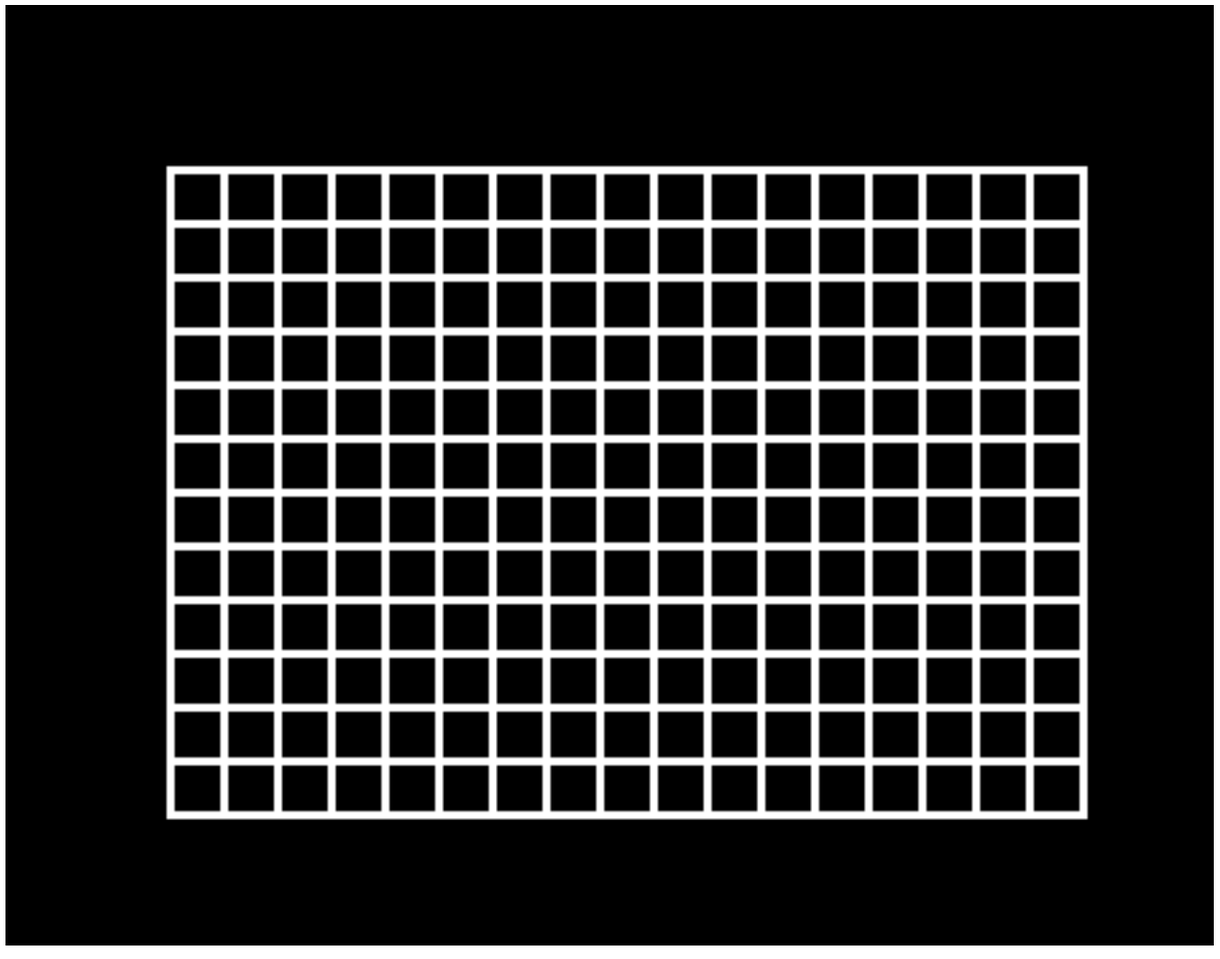
Bottom of tree: (X,-H,Z)
Top of tree: (X,L-H,Z)

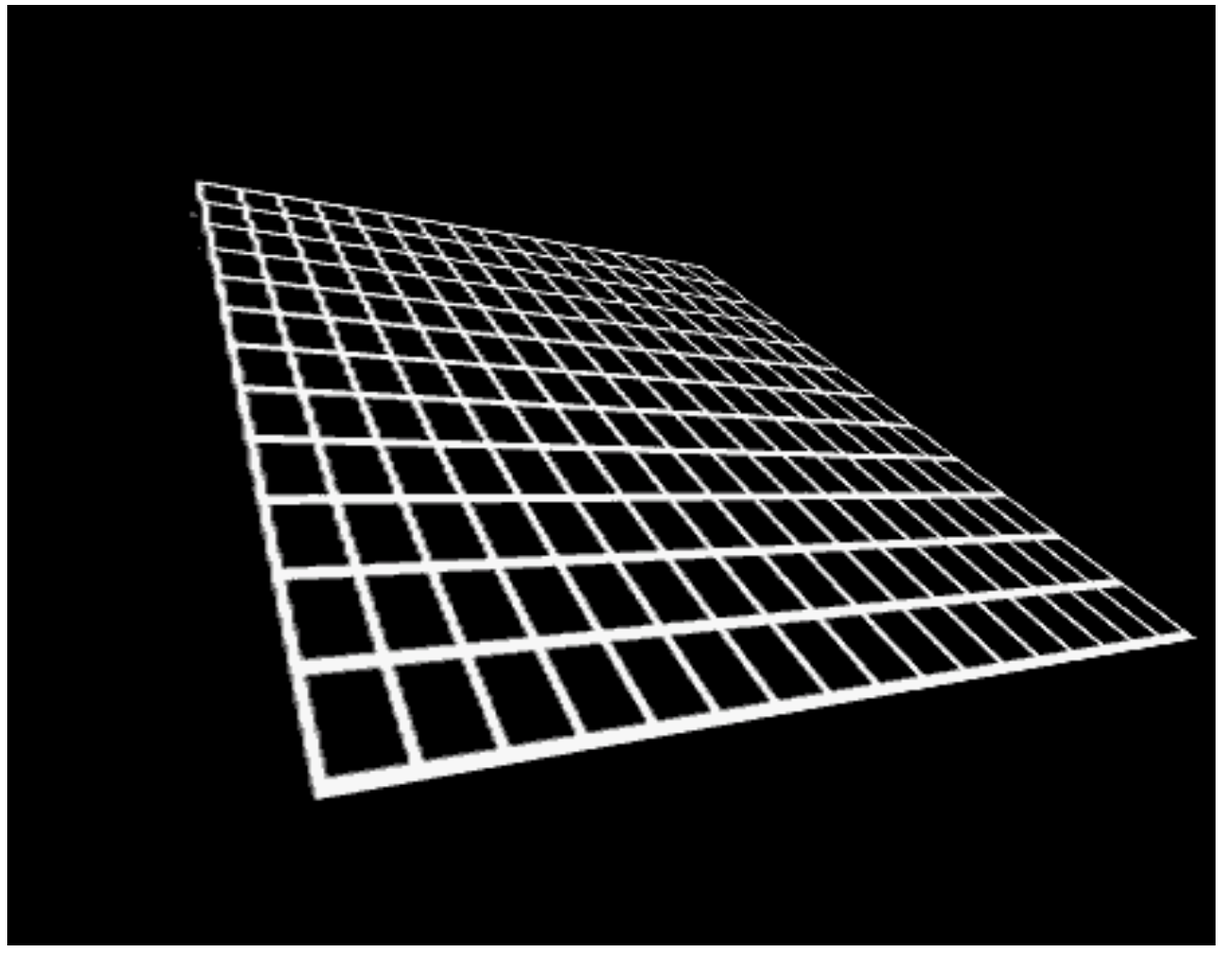


$$y_{top} - y_{bot} = \frac{f(L-H)}{Z} - \frac{-fH}{Z} = \frac{fL}{Z}$$

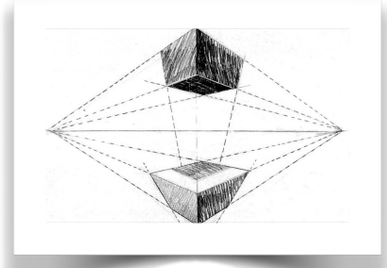
Consequence of derivations for image height and parallel lines

distances and angles aren’t preserved in camera projection





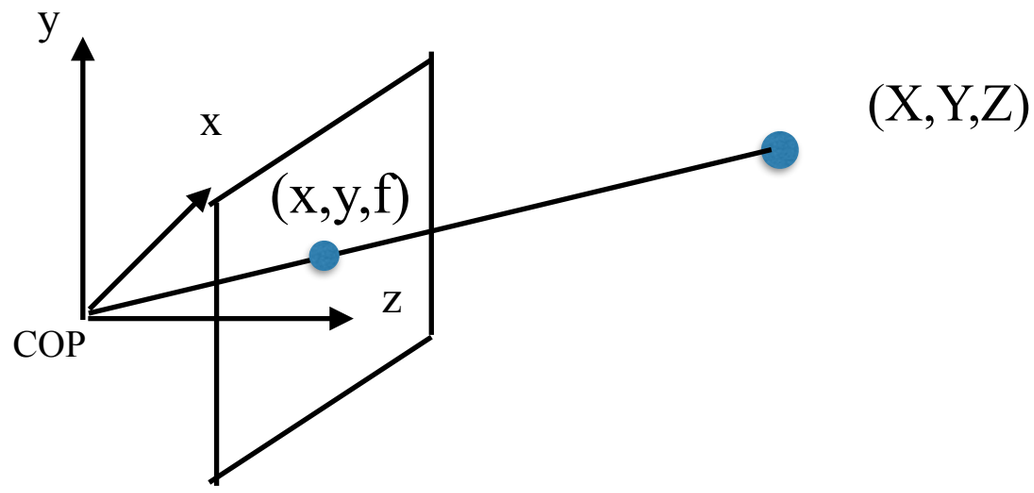
Agenda



- Pinhole optics
  - Perspective projection (vanishing points, horizon, object height)
  - **Camera matrices (intrinsics + extrinsics)**
  - Homographies (2 views of plane, rotation)

- Camera models
  - Properties of camera matrices (DOF, geometric intuition, pixel2rays)
  - Simplified cameras: orthographic, scaled orthographic, paraperspective, affine
  - Camera calibration (DLT v reprojection error)

Perspective projection revisited



$$x = \frac{f}{Z}X$$

$$y = \frac{f}{Z}Y$$

We'd like to write projection operation as a matrix, but dividing by Z makes projection fundamentally *nonlinear*…

Perspective projection revisited

$$\lambda \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

Notational collision: this is *not* the same λ from: $\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} A_x \\ A_y \\ A_z \end{bmatrix} + \lambda \begin{bmatrix} D_x \\ D_y \\ D_z \end{bmatrix}$

Given (X,Y,Z) and f, compute (x,y) and lambda:

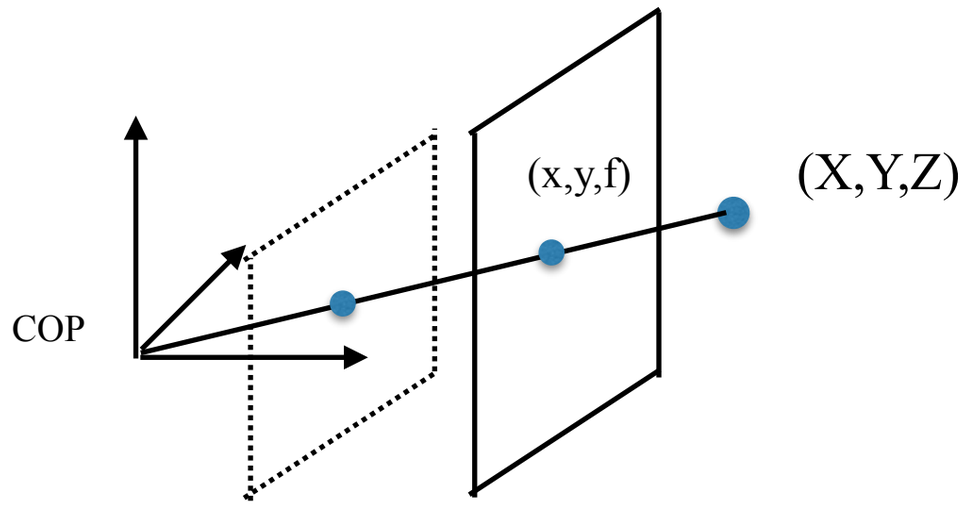
$$\lambda x = f X$$

$$\lambda = Z$$

$$x = \frac{\lambda x}{\lambda} = \frac{f X}{Z}$$

Use “lambda” as a notational device to force “division-by-Z”

Special case: f = 1 (*normalized* image plane)



$$\lambda \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

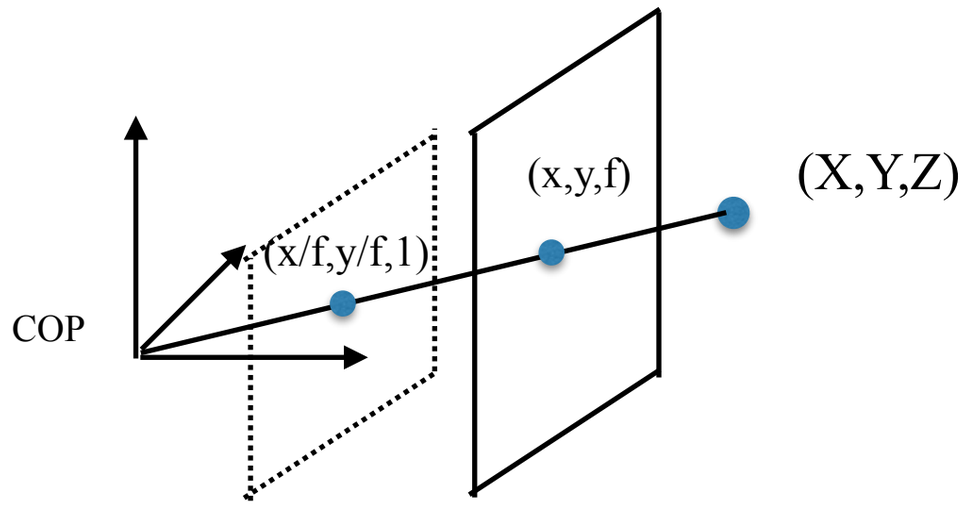
Equations are *really* simple for f = 1

In this special case, λ = depth of pixel (x,y)

$$Z \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

(we'll sometimes assume this for notational simplicity)

Special case: f = 1 (*normalized* image plane)



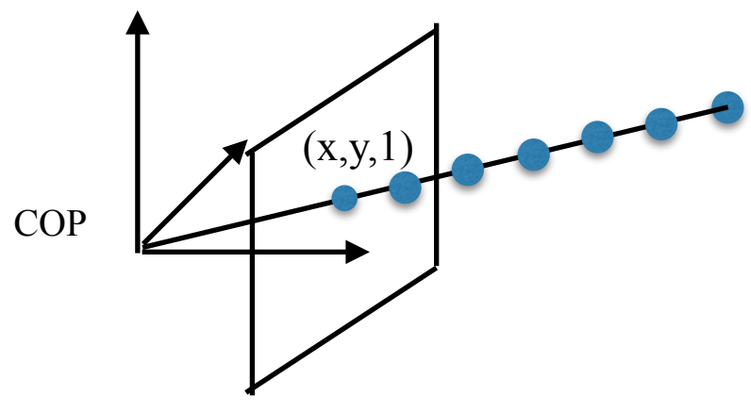
$$\lambda \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

Ques: given an image with focal length ‘f’, can we transform it to obtain a *normalized* image with a focal length of ‘1’?

Resize, or scale image by 1/f: x’ = x/f, y’= y/f

Homogenous notation

Define *equivalence* class of 3D points that project to same image point



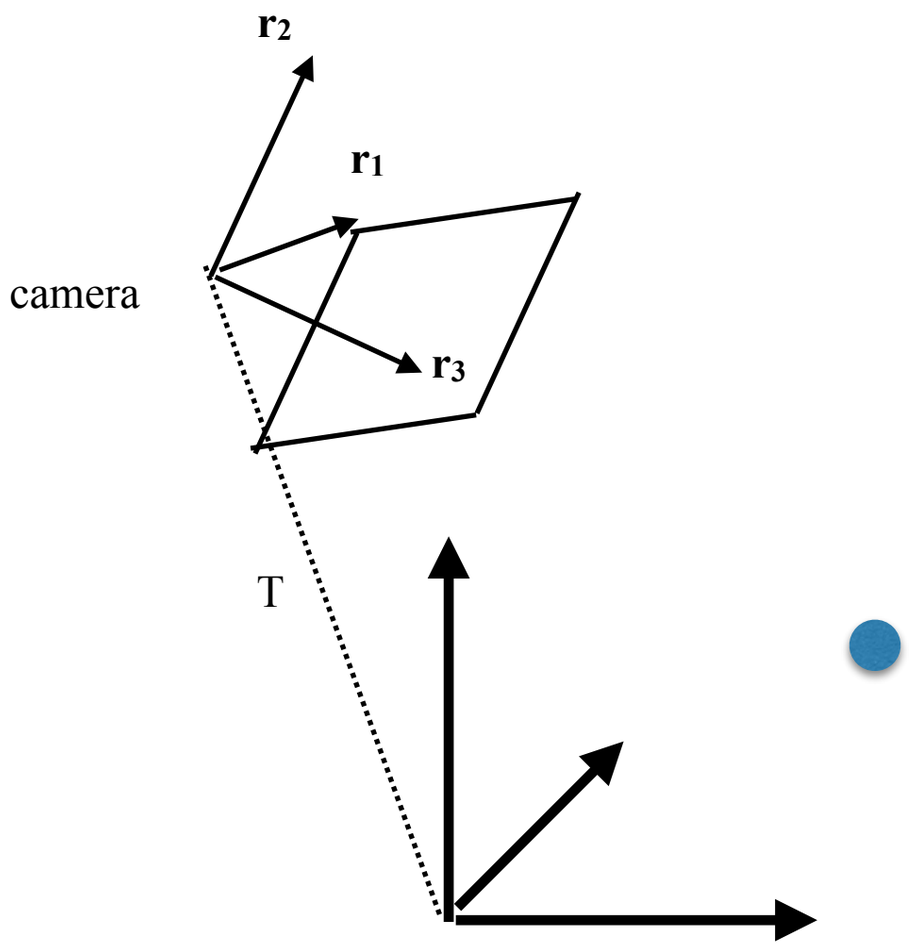
$$Z \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} xZ \\ yZ \\ Z \end{bmatrix}$$

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \sim \begin{bmatrix} xZ \\ yZ \\ Z \end{bmatrix}$$

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \equiv \begin{bmatrix} xZ \\ yZ \\ Z \end{bmatrix}$$

e.g., read $\sim$ and $\equiv$ as “equivalent up to scale”

Where we are headed…



We want to understand the equations for observing a 3D point (X,Y,Z) under a particular camera

How do we encode the *position* and *orientation* of the camera within the world?

Encode camera position with 3D rigid-body transformation

3D translations

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + T = \begin{bmatrix} X + t_x \\ Y + t_y \\ Z + t_z \end{bmatrix}$$

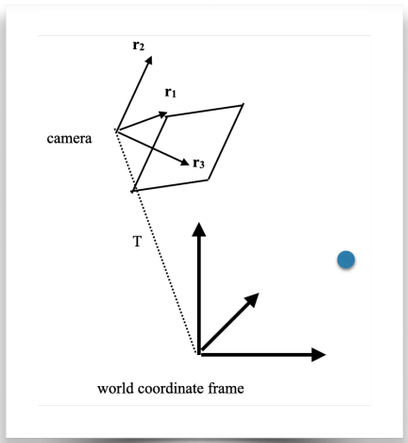
3D rotations

$$R \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

$$R \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + T = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} R & T \end{bmatrix}$$

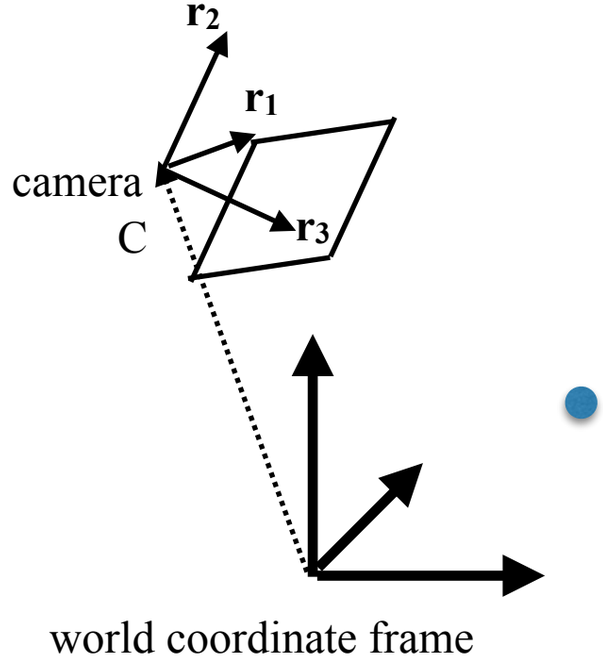
3x3 3x1



Alternative perspective: change of basis / coordinate system

Think of camera as *moving* through fixed world coordinate frame

$$R\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + T = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$



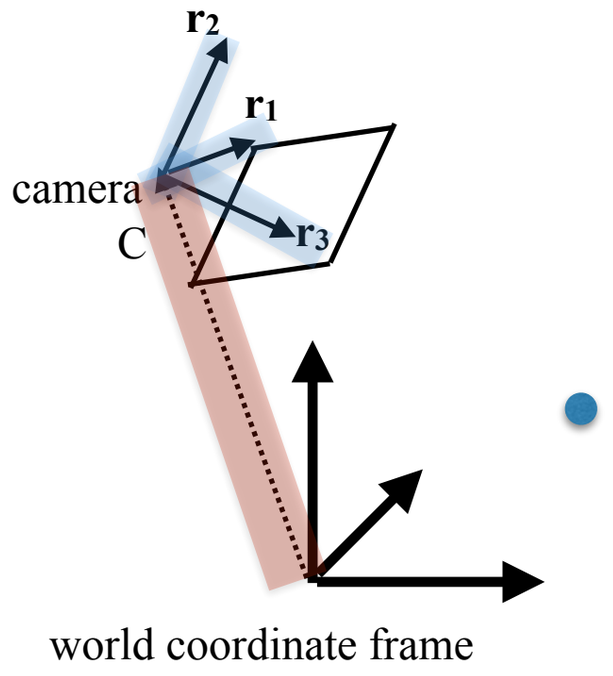
Let us write **C** and $\mathbf{r_1},\mathbf{r_2},\mathbf{r_3}$ as 3-vectors representing the camera center and camera axes directions *in the world coordinate frame*.
What is the 3x4 matrix that converts a point from world coordinates to camera coordinates?

Alternative perspective: change of basis / coordinate system

Think of camera as *moving* through fixed world coordinate frame

$$R \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + T = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

3x3 3x1



$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \left(\begin{bmatrix} X_w \\ Y_w \\ Z_w \end{bmatrix} - \begin{bmatrix} C_x \\ C_y \\ C_z \end{bmatrix} \right)$$

point in camera coordinates

point in world coordinates

$$= [R \quad -RC] \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}$$

3x3 3x1

where -RC is the position of the world origin $(X_w, Y_w, Z_w) = (0,0,0)$ in the camera coordinate frame

Negative sign implies if camera shifts *up* by 2 units, the point appears to shift *down* by 2 units

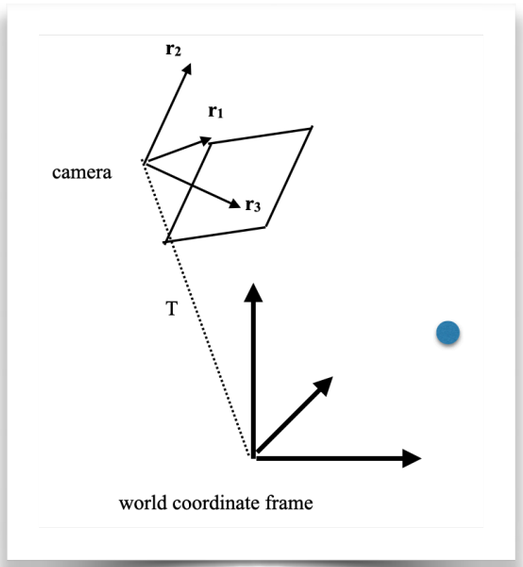
Camera projection

$$\lambda \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Camera instrinsic matrix K (can include skew & non-square pixel size)

Camera extrinsics (rotation and translation)

3D point in world coordinates



Recall: homogenous notation is shorthand for: $x = \frac{fX'}{Z'}$ where $\begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$

Fancier intrinsics

$$\lambda \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \underbrace{\begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\mathrm{K}} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

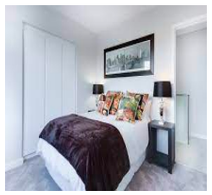
$$x' = x_s + o_x$$
$$y' = y_s + o_y$$

} shifted origin

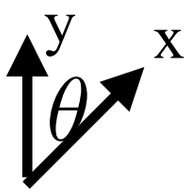
image cropping

$$x_s = s_x x$$
$$y_s = s_y y$$

} non-square pixels (sometimes ignored)



aspect ratio resizing



} skewed image axes (often ignored)

(rare)

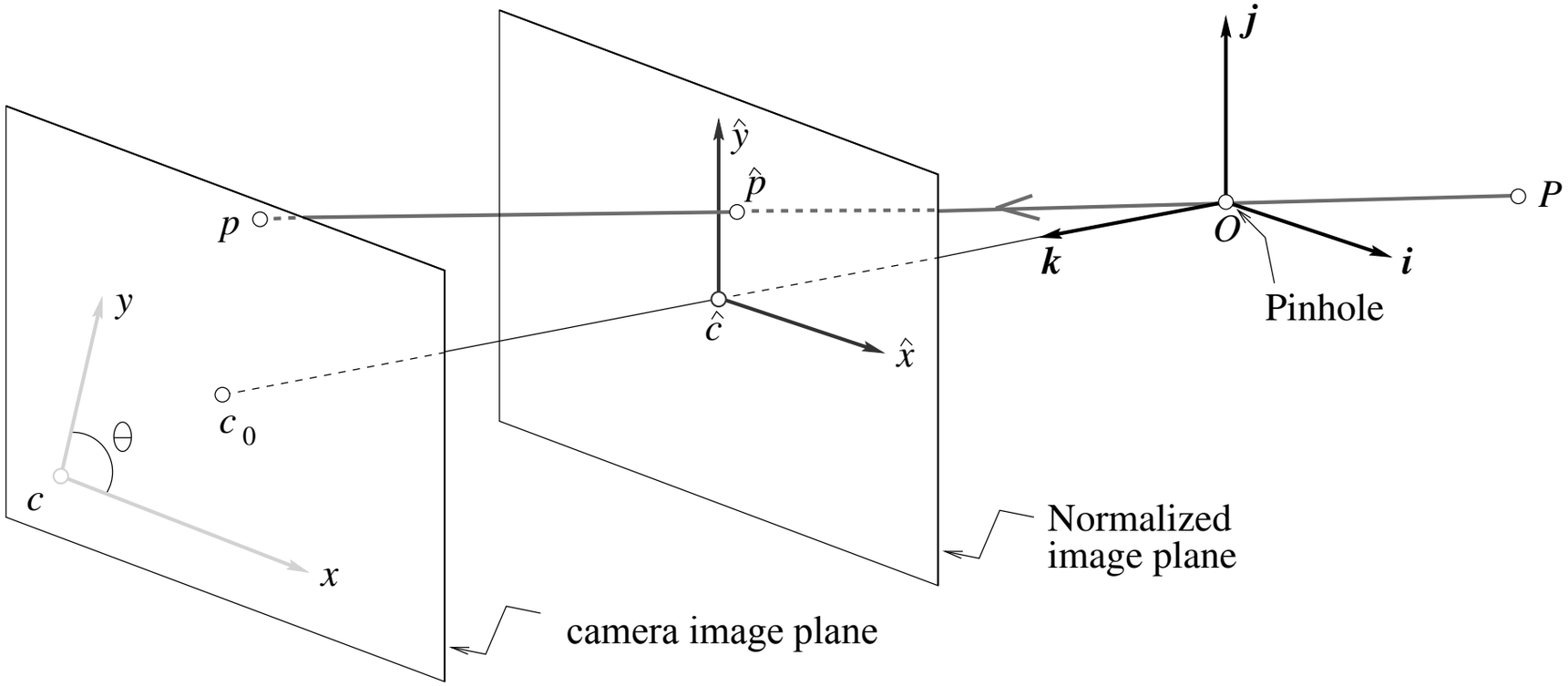
$$K = \begin{bmatrix} s_x & s_\theta & o_x \\ 0 & s_y & o_y \\ 0 & 0 & 1 \end{bmatrix}$$

(obtain classic instrinics by setting $s_x, s_y = f$ and $s_\theta, o_x, o_y = 0$)

Normalized image plane

$$\lambda \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & s_\theta & o_x \\ 0 & s_y & o_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

$$\mathbf{K} \qquad\qquad \mathbf{[R\ t]}$$



Given known intrinsic matrix **K**, it'd be convenient to work with normalized image coordinates $\hat{x}$, $\hat{y}$ (corresponding to a simplified $\hat{\mathbf{K}}$ = Identity) by *warping* the image with…

$$\begin{bmatrix} \hat{x} \\ \hat{y} \\ 1 \end{bmatrix} = K^{-1} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Ques: what kind of warp?

Affine

(invertible for non-zero s_x and s_y)

Alternate Notation for Camera Projection

$$\lambda \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & s_\theta & o_x \\ 0 & s_y & o_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

$$= K_{3\times3} \begin{bmatrix} R_{3\times3} & T_{3\times1} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

[Using rows x columns]

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

Alternate notation cont'd

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

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \sim \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \equiv \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

I suggest thinking of all this clunky notation as "shorthand" for the following:

$$\lambda \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} =$$

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \sim$$

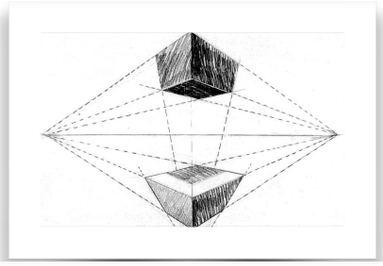
$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \equiv$$

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

$$y = \frac{m_{21}X + m_{22}Y + m_{23}Z + m_{14}}{m_{31}X + m_{32}Y + m_{33}Z + m_{34}}$$

Agenda

- Pinhole optics
  - Perspective projection (vanishing points, horizon, object height)
  - Camera matrices (intrinsics + extrinsics)
  - **Homographies (2 views of plane, rotation)**

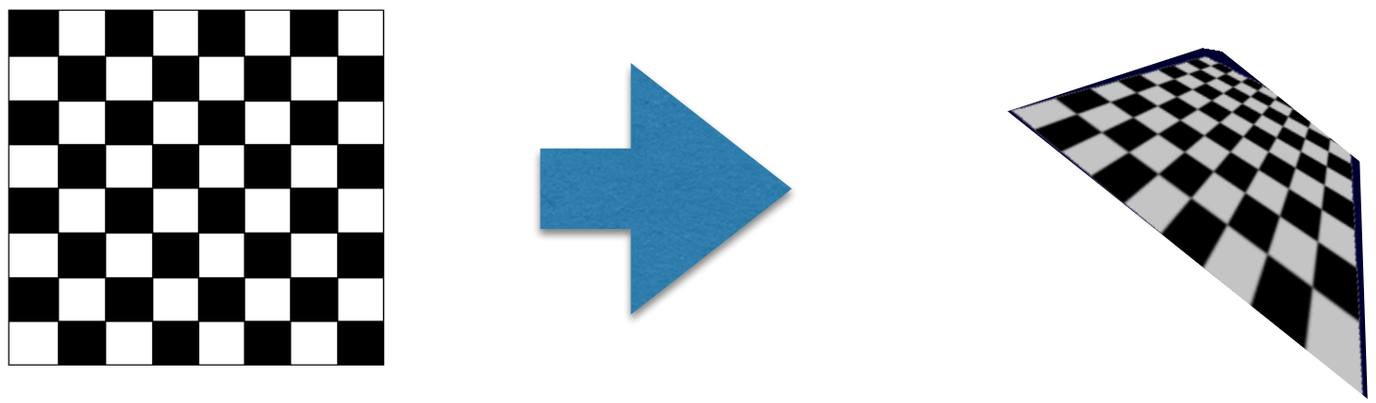


$$\lambda \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & s_\theta & o_x \\ 0 & s_y & o_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

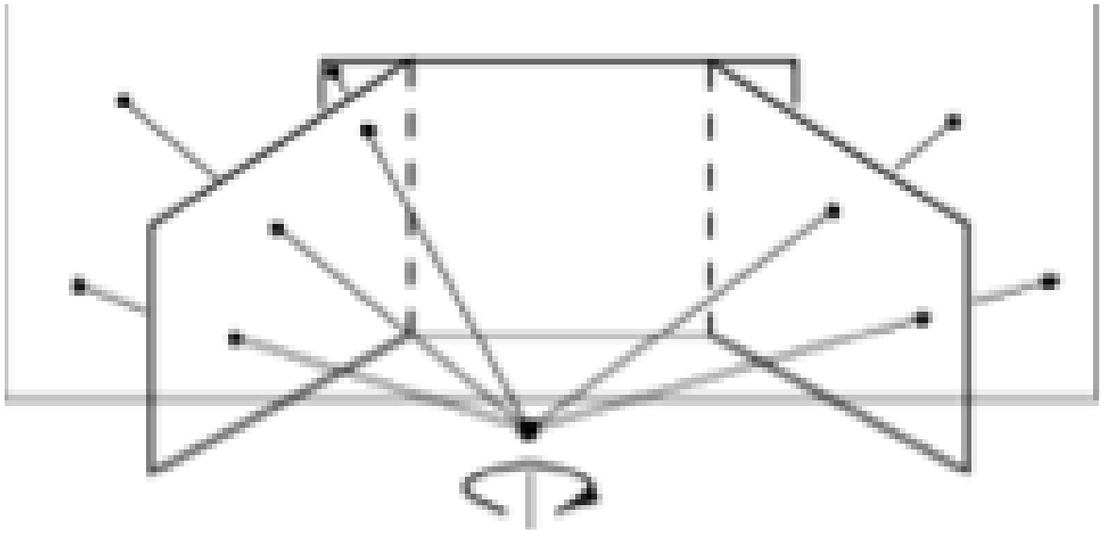
- Camera models
  - Properties of camera matrices (DOF, geometric intuition, pixel2rays)
  - Simplified cameras: orthographic, scaled orthographic, paraperspective, affine
  - Camera calibration (DLT v reprojection error)

Homography transformations

1. Models perspective effects for a planar scene



2. Models perspective effects from camera rotations



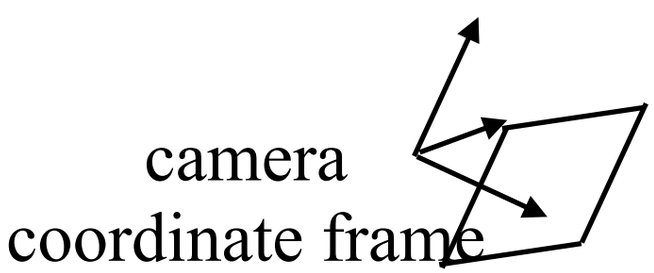
Let's analyze the projection of a 3D plane

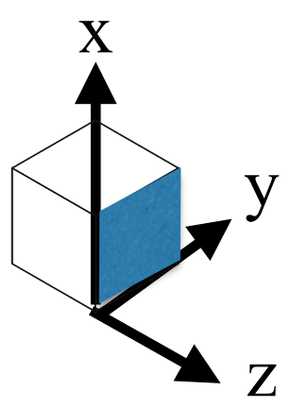
Place world coordinate frame on blue object plane

$$\lambda \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$ Plug in Z=0

$$= \begin{bmatrix} m_{11} & m_{12} & m_{14} \\ m_{21} & m_{22} & m_{24} \\ m_{31} & m_{32} & m_{34} \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}$$

Convert between 2D location on object plane and 2D image coordinate with a 3X3 matrix H



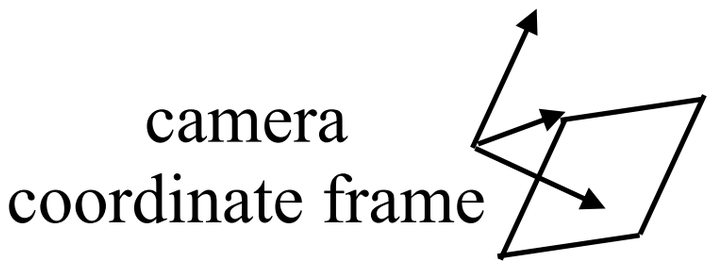


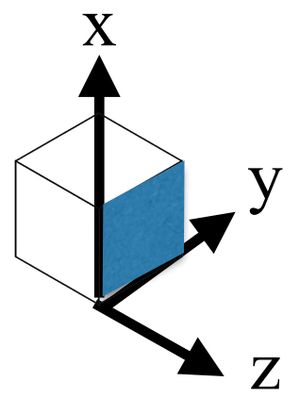
world coordinate frame

$$x = \frac{m_{11}X + m_{12}Y + m_{14}}{m_{31}X + m_{32}Y + m_{34}}$$

$$y = \frac{m_{21}X + m_{22}Y + m_{24}}{m_{31}X + m_{32}Y + m_{34}}$$

Inverse homographies





world coordinate frame

Given a point in world plane (X,Y), compute image position (x,y) as follows:

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \equiv H \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}$$

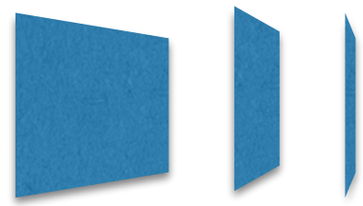
$$x = \frac{m_{11}X + m_{12}Y + m_{14}}{m_{31}X + m_{32}Y + m_{34}}$$
$$y = \frac{m_{21}X + m_{22}Y + m_{24}}{m_{31}X + m_{32}Y + m_{34}}$$

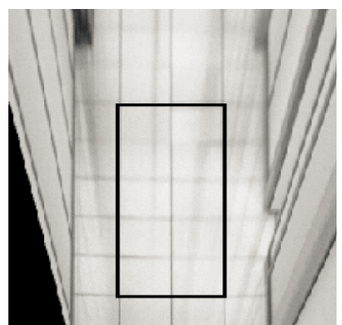
Given a point in image (x,y), compute position on world plane (X,Y) as follows:

$$\begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} \equiv H^{-1} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$X = \frac{m'_{11}x + m'_{12}y + m'_{14}}{m'_{31}x + m'_{32}y + m'_{34}}$$
$$Y = \frac{m'_{21}x + m'_{22}y + m'_{24}}{m'_{31}x + m'_{32}y + m'_{34}}$$

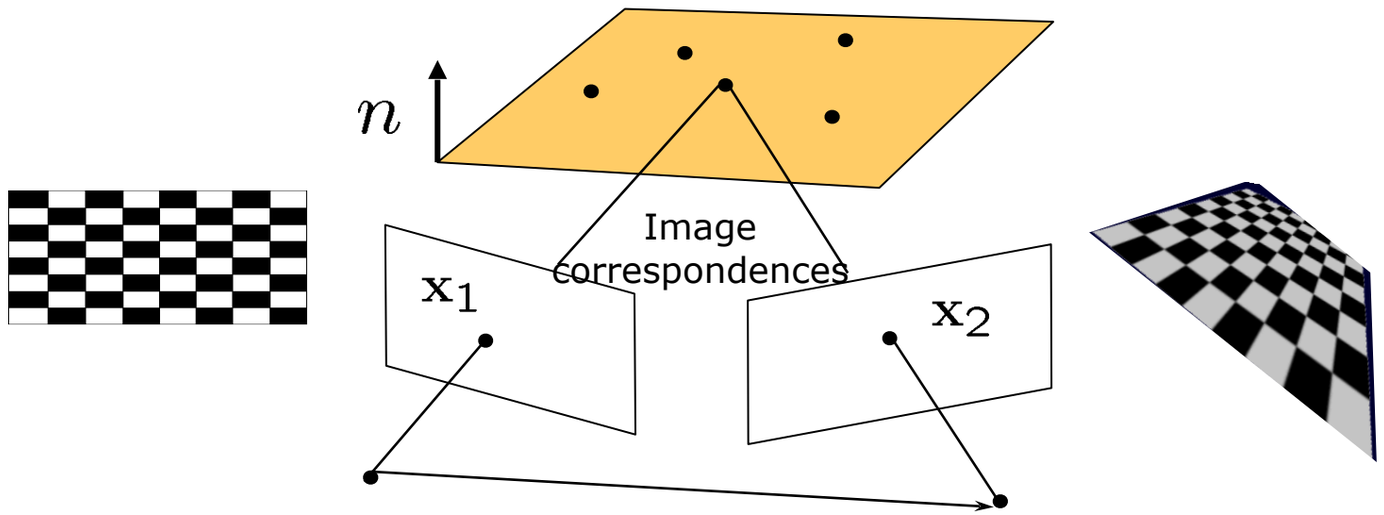
[Aside: H is *usually* invertible]





useful for frontalizing!

Relating 2 camera views of the same 3D plane



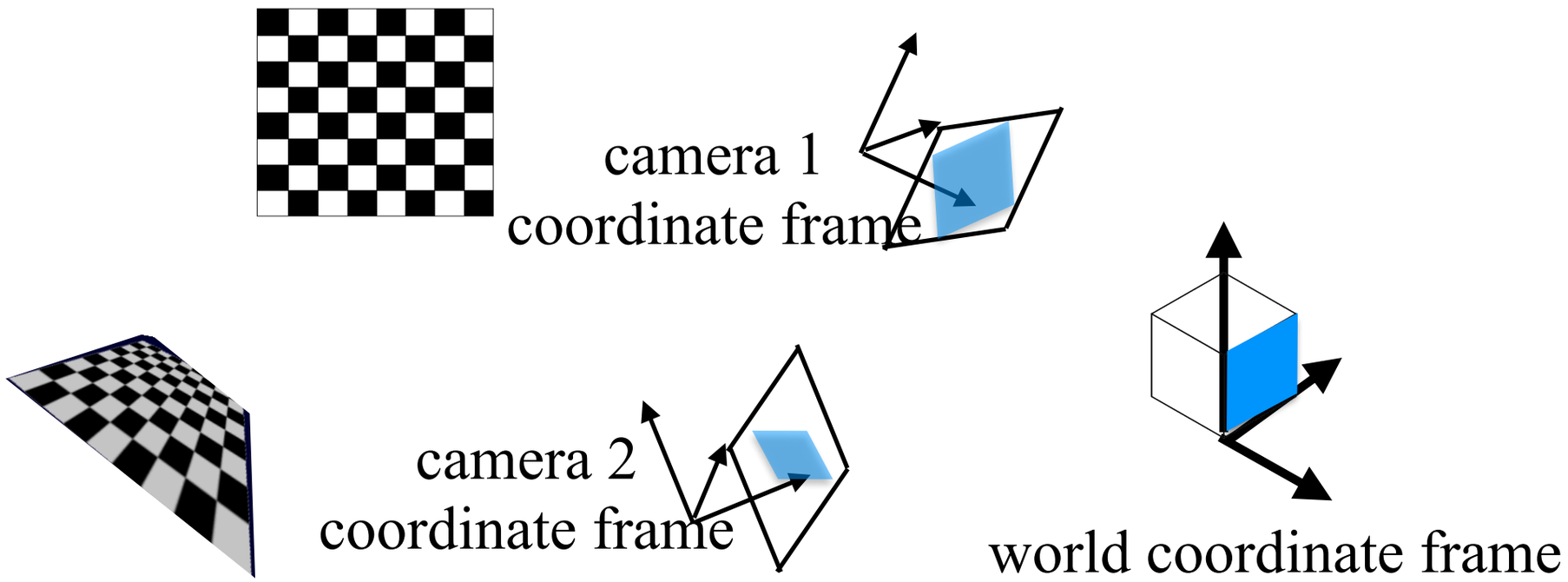
$$\begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} \equiv H_1 \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} \equiv H_2 \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}$$

$$\Rightarrow \quad \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} \equiv H_2 H_1^{-1} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix}$$

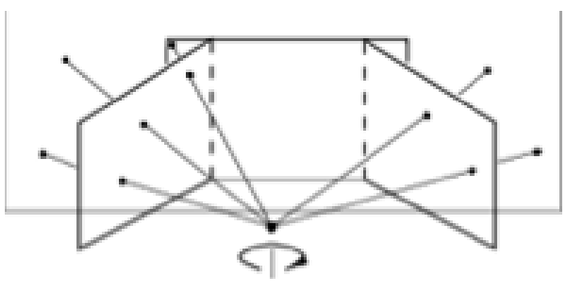
Relating 2 camera views of the same 3D plane: another perspective

Place world coordinate frame on the *normalized image* plane (f=1) of another camera!



$$\lambda \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix}$$

Special case of 2 views: rotations about camera center



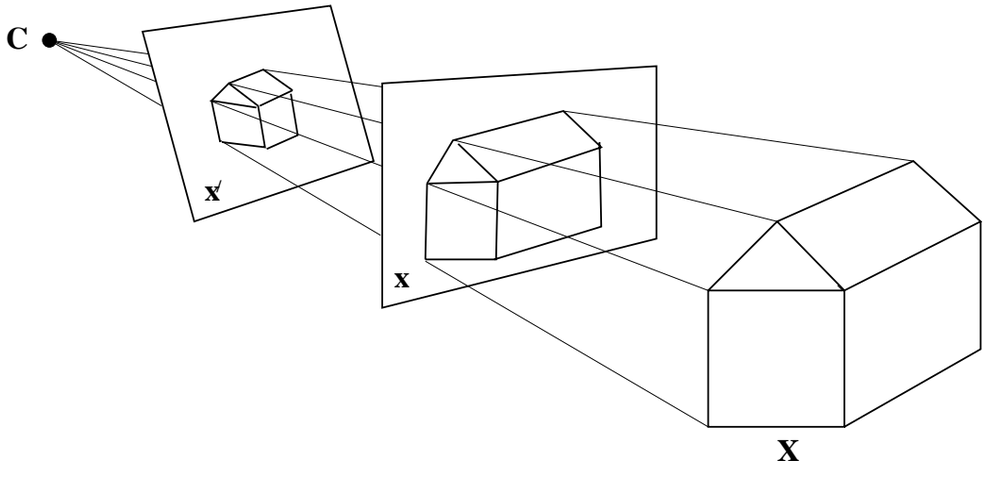
Relation between 3D camera coordinates: $\begin{bmatrix} X_2 \\ Y_2 \\ Z_2 \end{bmatrix} = R \begin{bmatrix} X_1 \\ Y_1 \\ Z_1 \end{bmatrix}$ K_2

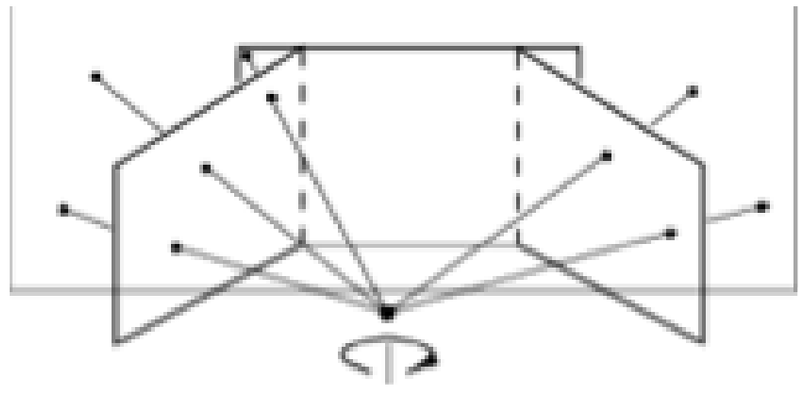
3D->2D projection: $\lambda_2 \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = \begin{bmatrix} f_2 & 0 & 0 \\ 0 & f_2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_2 \\ Y_2 \\ Z_2 \end{bmatrix}$

...

Combining both: $\lambda \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = K_2 R K_1^{-1} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix}$

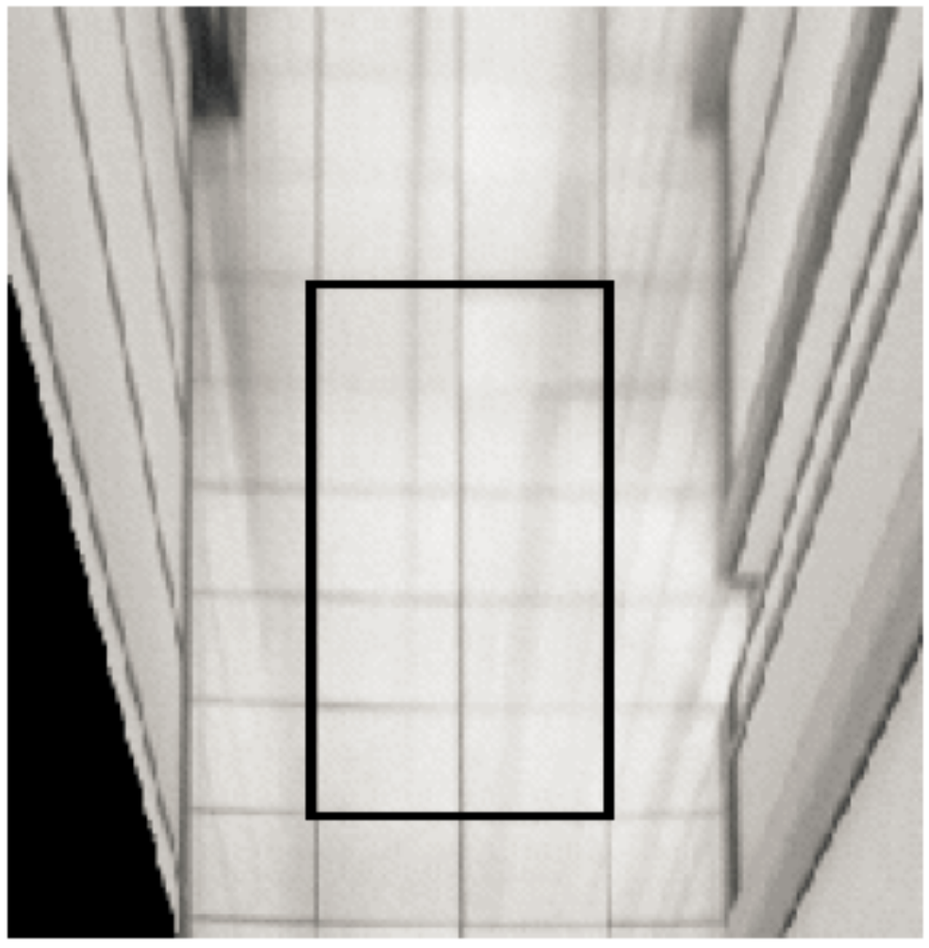
Geometric intuition: look at image of a planar image





3D images are modelled with planar transformations,
regardless of 3D scene geometry!

“Frontalizing” planes using homographies



from Hartley & Zisserman

Take-home points for homographies

$$\lambda \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix}$$

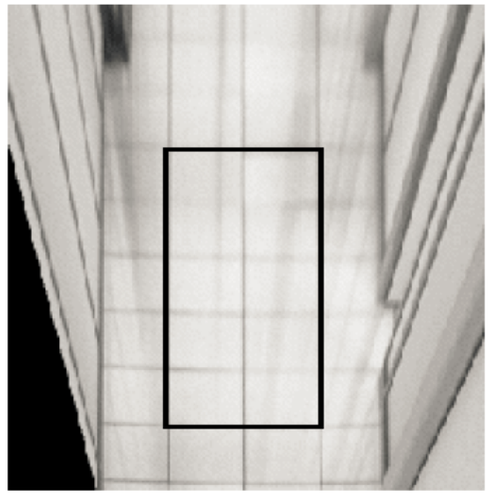
- If camera rotates about its center, then the images are related by a homography irrespective of scene depth.

- If the scene is planar, then images from any two cameras are related by a homography.

- Homography mapping is a 3x3 matrix with 8 degrees of freedom.

Estimating homographies

Given (x_1,y_1) and H, how do we compute (x_2,y_2)?



from Hartley & Zisserman

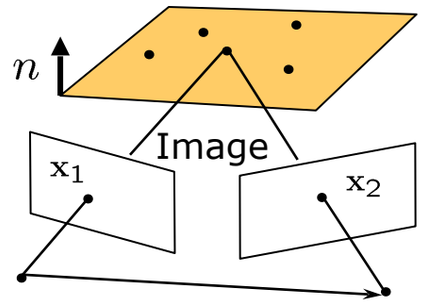
$$\lambda \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix}$$

$$x_2 = \frac{\lambda x_2}{\lambda} = \frac{ax_1 + by_1 + c}{gx_1 + hy_1 + i}$$

Estimating Homographies (via the Direct Linear Transform)

[*kind of* similar to linear least squares for estimating warp]

Given corresponding 2D points in left and right image, estimate H



$$x_2(gx_1 + hy_1 + i) = ax_1 + by_1 + c$$

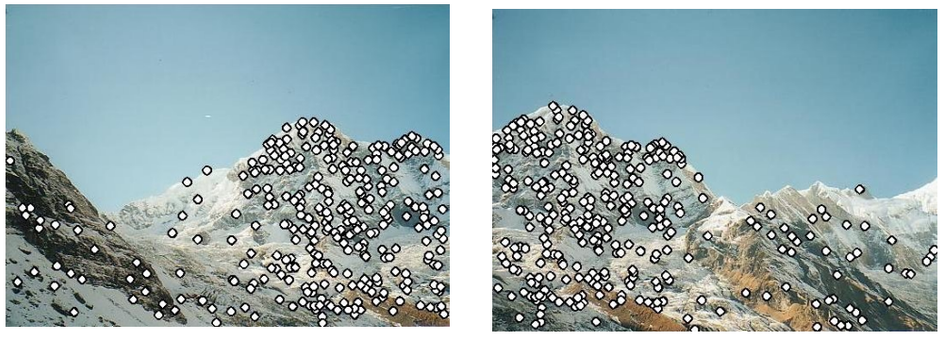
$$\vdots$$

$$AH(:) = \begin{bmatrix} 0 \\ 0 \\ \vdots \end{bmatrix}$$ Homogenous linear system

How many degrees of freedom in H? 8

How many corresponding points needed? 4

RANSAC for homography fitting



Given 2 images with interest points and candidate matches

Repeat ***k*** times:

- Draw ***n*** points uniformly at random (from left image)
- Fit (homography) warp to these ***n*** points and their correspondences
- Find inliers among the remaining left-image points (i.e., whose warped positions land close to right-image correspondence)
- Return warp with largest inlier set

(Optionally return line with optimal least-squares inlier fit)

Source: M. Pollefeys