COSC579: Scene Geometry

Jeremy Bolton, PhD
Assistant Teaching Professor

GEORGETOWN UNIVERSITY

Overview

- Linear Algebra Review
  - Homogeneous vs non-homogeneous representations
  - Projections and Transformations
- Scene Geometry
- The pinhole projection model
  - Qualitative properties
  - Perspective projection matrix
- Cameras with lenses
  - Depth of focus
  - Field of view
  - Lens aberrations

Scene Geometry, Projection, and Perspective

- Readings
  - Szeliski 2.1

GEORGETOWN UNIVERSITY

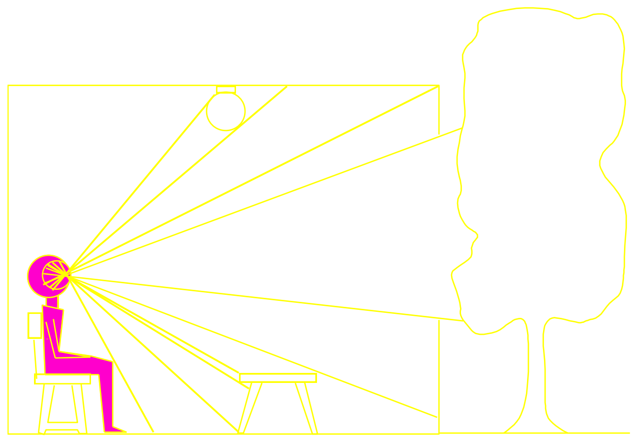
Projection

- Readings
  - Szeliski 2.1

GEORGETOWN UNIVERSITY

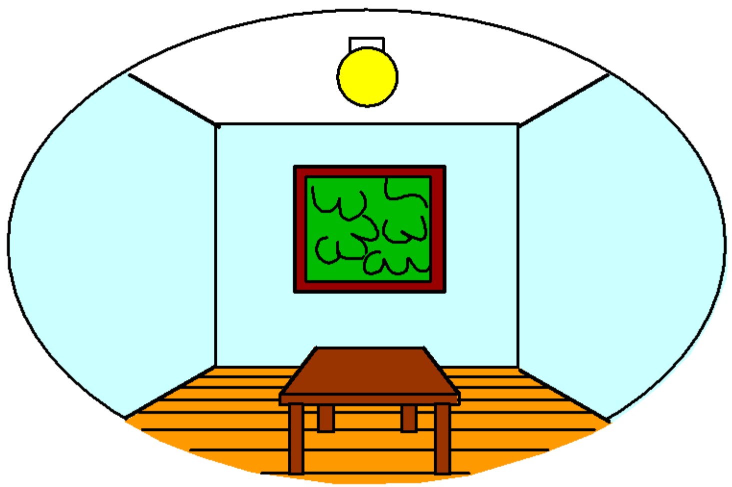
Dimensionality Reduction (3D to 2D)

3D world



Point of observation

2D image



What have we lost?

- Angles
- Distances (lengths)

Linear Algebra and Projections

- Before we dive into camera geometry, lets introduce some notation and review linear algebra.

- 2-D points will be used to represent pixel coordinates: points in a (image) plane. (non-homogeneous)

$$x = (x, y) \in \mathcal{R}^2,$$

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

Homogeneous and Non-homogeneous Vectors

- 2-D points can be represented by a vector and a weight.
  - Homogeneous
  - *P*^{2} is the 2-d projective space

$$\tilde{\boldsymbol{x}} = (\tilde{x}, \tilde{y}, \tilde{w}) \in \mathcal{P}^2$$

$$\tilde{\boldsymbol{x}} = (\tilde{x}, \tilde{y}, \tilde{w}) = \tilde{w}(x, y, 1) = \tilde{w}\bar{\boldsymbol{x}}$$

where $\bar{\boldsymbol{x}} = (x, y, 1)$ is the *augmented vector*.

Why Homogeneous Coordinates?

- Homogeneous representation permits linear matrix operations for transformations used in Computer Vision

How? add one more coordinate:

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

homogeneous image coordinates

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

homogeneous scene coordinates

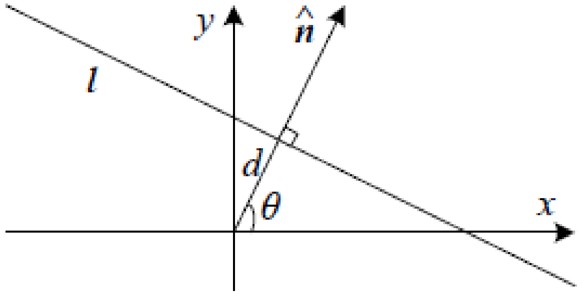
Converting *from* homogeneous coordinates

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

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

Lines

- 2-D Lines can also be represented using homogeneous coordinates $\tilde{l} = (a, b, c)$



- Corresponding line equation

$$\bar{x} \cdot \tilde{l} = ax + by + c = 0.$$

- It is common to normalize the line equation vector

$$l = (\hat{n}_x, \hat{n}_y, d) = (\hat{n}, d) \text{ with } \|\hat{n}\| = 1.$$

- Observe: in this formulation $\hat{n}$ is the unit normal vector perpendicular to the line and d is its distance to the origin

Lines

- Computing the intersection of 2 lines using homogeneous coordinates

$$\tilde{x} = \tilde{l}_1 \times \tilde{l}_2$$

- Computing the line joining two points using homogeneous coordinates

$$\tilde{l} = \tilde{x}_1 \times \tilde{x}_2.$$

GEORGETOWN UNIVERSITY

3-D

- Similar representations exist to model 3-D points: points in a scene.

$$x = (x, y, z) \in \mathcal{R}^3$$

$$\tilde{x} = (\tilde{x}, \tilde{y}, \tilde{z}, \tilde{w}) \in \mathcal{P}^3$$

- 3-D planes specification using homogeneous coordinates

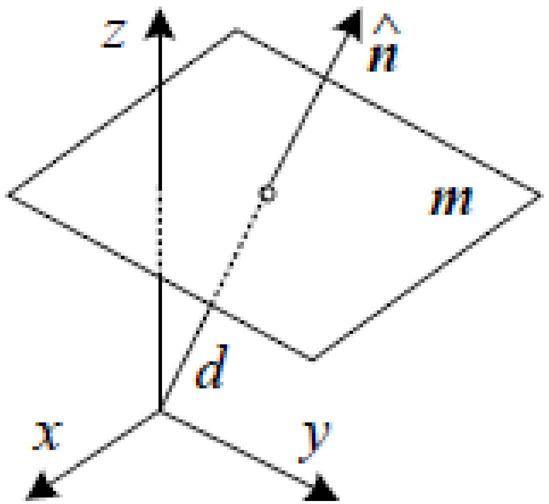
$$\tilde{m} = (a, b, c, d)$$

- Equation of plane

$$\bar{x} \cdot \tilde{m} = ax + by + cz + d = 0.$$

- Normalized

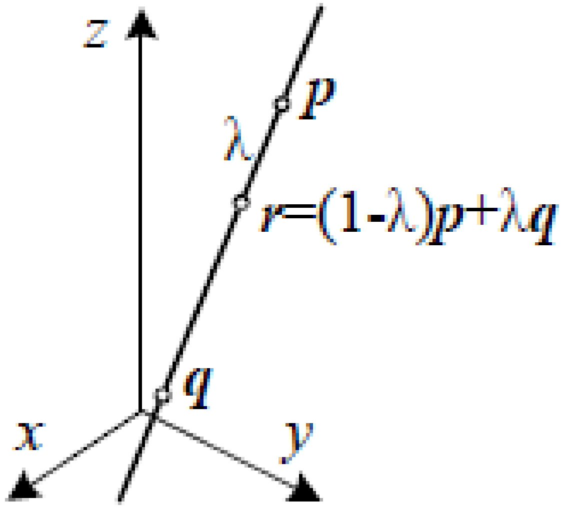
$$m = (\hat{n}_x, \hat{n}_y, \hat{n}_z, d) = (\hat{n}, d) \text{ with } \|\hat{n}\| = 1$$



Lines in 3-D

- Not as easy to represent

- One method: model line segment r as convex combination of two points on the line.
  - Assume p and q are points on the line.
    - $0 \le \lambda \le 1$ $\qquad \tilde{r} = \mu\tilde{p} + \lambda\tilde{q}$
- Homogeneous coordinates



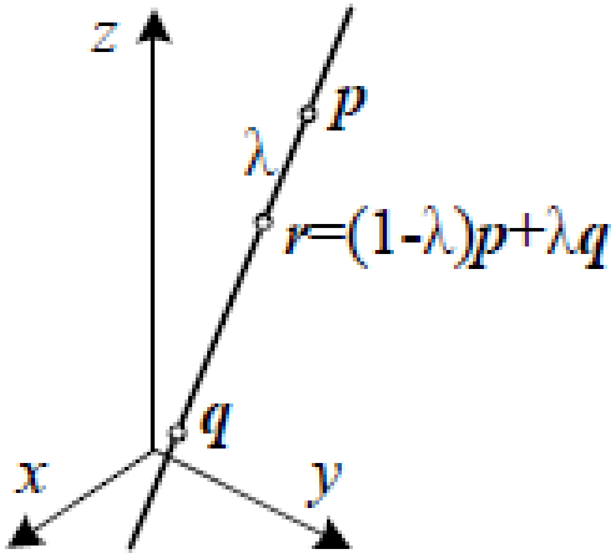
3D line equation, $r = (1 - \lambda)p + \lambda q$.

Lines in 3-D

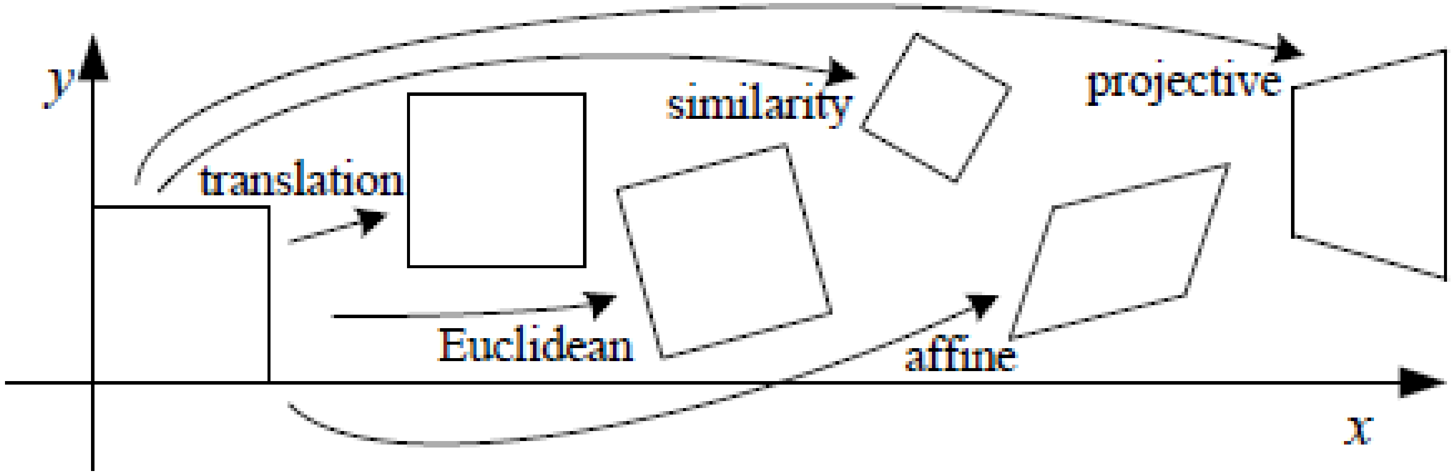
- More generally, we can represent a line in 3-D using a point and a gradient

$$r = p + \lambda \hat{d}$$

$$\hat{d} = [\Delta x, \Delta y, \Delta z]$$



2D Planar Transformations



GEORGETOWN UNIVERSITY

Translations

- Here the augmented vector $\bar{x}$ is used to account for translation t
- Simply a shift

Translation. 2D translations can be written as $x' = x + t$ or

$$x' = \begin{bmatrix} I & t \end{bmatrix} \bar{x}$$

where I is the (2×2) identity matrix or

$$\bar{x}' = \begin{bmatrix} I & t \\ 0^T & 1 \end{bmatrix} \bar{x}$$

Euclidean Transformations

Rotation + translation. This transformation is also known as *2D rigid body motion* or the *2D Euclidean transformation* (since Euclidean distances are preserved). It can be written as $\boldsymbol{x}' = \boldsymbol{R}\boldsymbol{x} + \boldsymbol{t}$ or

$$\boldsymbol{x}' = \begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \end{bmatrix} \bar{\boldsymbol{x}} \tag{2.16}$$

where

$$\boldsymbol{R} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \tag{2.17}$$

is an orthonormal rotation matrix with $\boldsymbol{R}\boldsymbol{R}^T = \boldsymbol{I}$ and $|\boldsymbol{R}| = 1$.

Scaled Rotations

Scaled rotation. Also known as the *similarity transform*, this transformation can be expressed as $x' = s\boldsymbol{R}x + \boldsymbol{t}$ where s is an arbitrary scale factor. It can also be written as

$$x' = \begin{bmatrix} s\boldsymbol{R} & \boldsymbol{t} \end{bmatrix} \bar{x} = \begin{bmatrix} a & -b & t_x \\ b & a & t_y \end{bmatrix} \bar{x}, \tag{2.18}$$

where we no longer require that $a^2 + b^2 = 1$. The similarity transform preserves angles between lines.

GEORGETOWN UNIVERSITY

Affine

Affine. The affine transformation is written as $x' = A\bar{x}$, where A is an arbitrary 2×3 matrix, i.e.,

$$x' = \begin{bmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \end{bmatrix} \bar{x}. \tag{2.19}$$

Parallel lines remain parallel under affine transformations.

Projective Transformation (Homography)

Projective. This transformation, also known as a *perspective transform* or *homography*, operates on homogeneous coordinates,

$$\tilde{x}' = \tilde{\boldsymbol{H}}\tilde{x}, \tag{2.20}$$

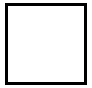
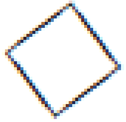
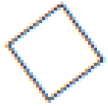
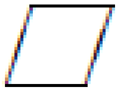
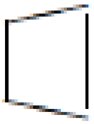
where $\tilde{\boldsymbol{H}}$ is an arbitrary 3×3 matrix. Note that $\tilde{\boldsymbol{H}}$ is homogeneous, i.e., it is only defined up to a scale, and that two $\tilde{\boldsymbol{H}}$ matrices that differ only by scale are equivalent. The resulting homogeneous coordinate $\tilde{x}'$ must be normalized in order to obtain an inhomogeneous result x, i.e.,

$$x' = \frac{h_{00}x + h_{01}y + h_{02}}{h_{20}x + h_{21}y + h_{22}} \quad \text{and} \quad y' = \frac{h_{10}x + h_{11}y + h_{12}}{h_{20}x + h_{21}y + h_{22}}. \tag{2.21}$$

Perspective transformations preserve straight lines (i.e., they remain straight after the transformation).

GEORGETOWN UNIVERSITY

Properties of Transformations

Transformation	Matrix	# DoF	Preserves	Icon
translation	$\begin{bmatrix} \boldsymbol{I} \mid \boldsymbol{t} \end{bmatrix}_{2\times 3}$	2	orientation	
rigid (Euclidean)	$\begin{bmatrix} \boldsymbol{R} \mid \boldsymbol{t} \end{bmatrix}_{2\times 3}$	3	lengths	
similarity	$\begin{bmatrix} s\boldsymbol{R} \mid \boldsymbol{t} \end{bmatrix}_{2\times 3}$	4	angles	
affine	$\begin{bmatrix} \boldsymbol{A} \end{bmatrix}_{2\times 3}$	6	parallelism	
projective	$\begin{bmatrix} \tilde{\boldsymbol{H}} \end{bmatrix}_{3\times 3}$	8	straight lines	

3D transformations

Translation. 3D translations can be written as $\boldsymbol{x}' = \boldsymbol{x} + \boldsymbol{t}$ or

$$\boldsymbol{x}' = \begin{bmatrix} \boldsymbol{I} & \boldsymbol{t} \end{bmatrix} \bar{\boldsymbol{x}}$$

GEORGETOWN UNIVERSITY

3D Euclidean Transformation

Rotation + translation. Also known as 3D *rigid body motion* or the 3D *Euclidean transformation*, it can be written as $\boldsymbol{x}' = \boldsymbol{R}\boldsymbol{x} + \boldsymbol{t}$ or

$$\boldsymbol{x}' = \begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \end{bmatrix} \bar{\boldsymbol{x}} \tag{2.24}$$

where $\boldsymbol{R}$ is a 3×3 orthonormal rotation matrix with $\boldsymbol{R}\boldsymbol{R}^T = \boldsymbol{I}$ and $|\boldsymbol{R}| = 1$. Note that sometimes it is more convenient to describe a rigid motion using

$$\boldsymbol{x}' = \boldsymbol{R}(\boldsymbol{x} - \boldsymbol{c}) = \boldsymbol{R}\boldsymbol{x} - \boldsymbol{R}\boldsymbol{c}, \tag{2.25}$$

where $\boldsymbol{c}$ is the center of rotation (often the camera center).

3D similarity transformation

Scaled rotation. The 3D *similarity transform* can be expressed as $\boldsymbol{x}' = s\boldsymbol{R}\boldsymbol{x} + \boldsymbol{t}$ where s is an arbitrary scale factor. It can also be written as

$$\boldsymbol{x}' = \begin{bmatrix} s\boldsymbol{R} & \boldsymbol{t} \end{bmatrix} \bar{\boldsymbol{x}}. \tag{2.26}$$

This transformation preserves angles between lines and planes.

GEORGETOWN UNIVERSITY

3D Affine Transformation

Affine. The affine transform is written as $x' = A\bar{x}$, where A is an arbitrary 3×4 matrix, i.e.,

$$x' = \begin{bmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \end{bmatrix} \bar{x}. \tag{2.27}$$

Parallel lines and planes remain parallel under affine transformations.

Projective Transformation

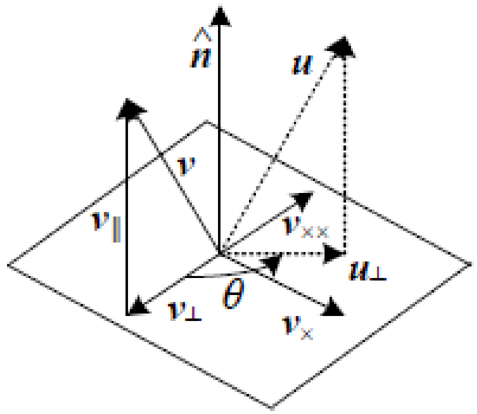
Projective. This transformation, variously known as a *3D perspective transform*, *homography*, or *collineation*, operates on homogeneous coordinates,

$$\tilde{x}' = \tilde{H}\tilde{x}, \tag{2.28}$$

where $\tilde{H}$ is an arbitrary 4×4 homogeneous matrix. As in 2D, the resulting homogeneous coordinate $\tilde{x}'$ must be normalized in order to obtain an inhomogeneous result x. Perspective transformations preserve straight lines (i.e., they remain straight after the transformation).

3D Rotations

- Sequential Rotations Approach (Euler Angle).
  - Perform 3 sequential rotations around 3 cardinal axes.
  - Cons
    - Result depends on order of rotations.
    - Not always practical
- Axis / Angle Approach
  - Rotation can be determined by 1 rotation axis and 1 angle



Rotation around an axis $\hat{n}$ by an angle θ.

GEORGETOWN UNIVERSITY

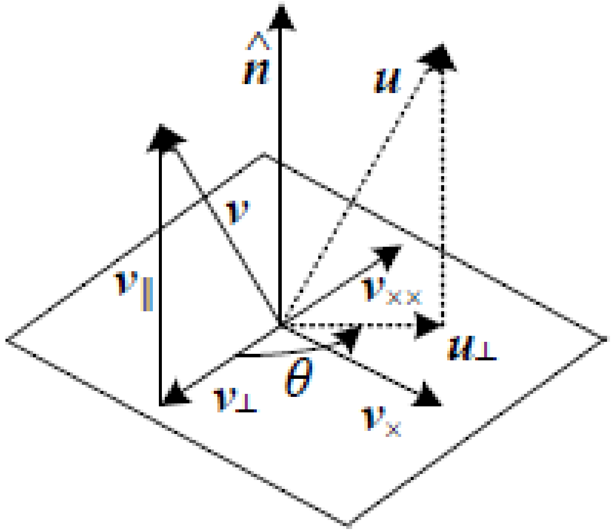
3D Axis/Angle Rotation

- Constructing the rotation matrix R, given $\hat{n}$ and θ

$$\boldsymbol{R}(\hat{\boldsymbol{n}}, \theta) = \boldsymbol{I} + \sin\theta[\hat{\boldsymbol{n}}]_\times + (1 - \cos\theta)[\hat{\boldsymbol{n}}]^2_\times$$

$$\hat{\boldsymbol{n}} = (\hat{n}_x, \hat{n}_y, \hat{n}_z)$$

$$[\hat{\boldsymbol{n}}]_\times = \begin{bmatrix} 0 & -\hat{n}_z & \hat{n}_y \\ \hat{n}_z & 0 & -\hat{n}_x \\ -\hat{n}_y & \hat{n}_x & 0 \end{bmatrix}$$



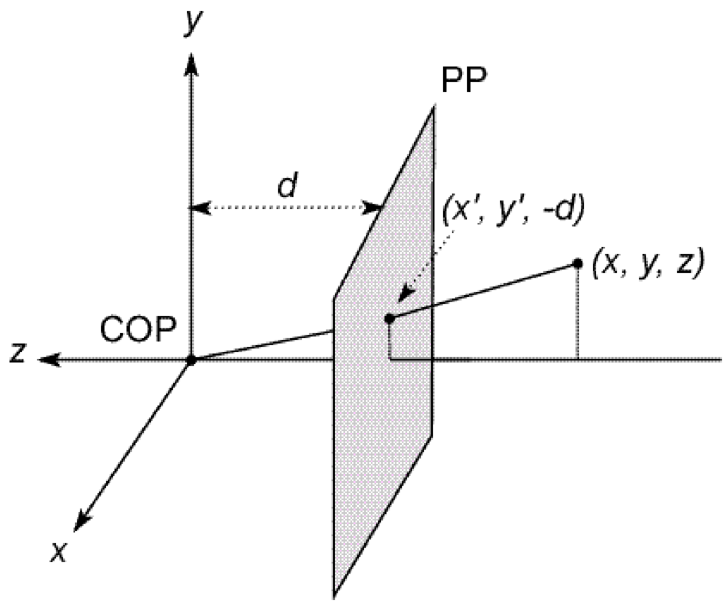
Rotation around an axis $\hat{n}$ by an angle θ.

Projecting from 3D to 2D

- Projections
  - Orthographic and scaled orthographic (weak perspective) projection
  - Para-perspective projection
  - Perspective projection

- These projects are used often during image formation.

Modeling projection

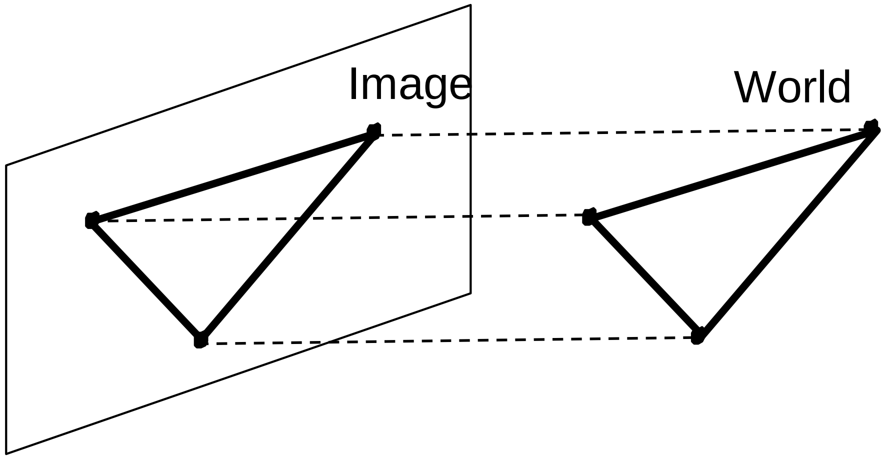


- The coordinate system
  - We will use the pin-hole model as an approximation
  - Put the optical center (**C**enter **O**f **P**rojection) at the origin
  - Put the image plane (**P**rojection **P**lane) *in front* of the COP
    - Why?
  - The camera looks down the *negative* z axis
    - we need this if we want right-handed-coordinates

GEORGETOWN UNIVERSITY

Orthographic projection

- Special case of perspective projection
  - Distance from the COP to the PP is infinite ([ignore z-axis](#))



  - Good approximation for telephoto optics
  - Also called “parallel projection”: (x, y, z) → (x, y)
  - What’s the projection matrix?

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

Variants of orthographic projection

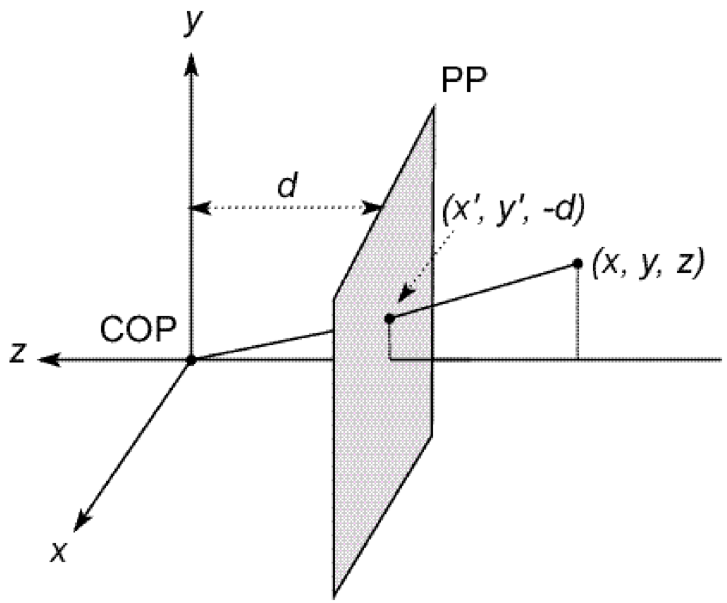
- Scaled orthographic
  - Also called “weak perspective”

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

- Affine projection
  - Also called “para-perspective”

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

Modeling projection



- Projection equations
  - Compute intersection with PP of ray from (x,y,z) to COP
  - Derived using similar triangles (Try this at Home!)

$$(x, y, z) \rightarrow (-d\frac{x}{z}, \ -d\frac{y}{z}, \ -d)$$

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

$$(x, y, z) \rightarrow (-d\frac{x}{z}, \ -d\frac{y}{z})$$

GEORGETOWN UNIVERSITY

Perspective Projection

- Projection is a matrix multiply using homogeneous coordinates:

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

divide by third coordinate

This is known as **perspective projection**

- The matrix is the **projection matrix**
- Can also formulate as a 4x4

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

divide by fourth coordinate

Projection Review

- Reviewed
  - Linear algebra and notation
  - Projections

- Projections are used to model image formation and analysis

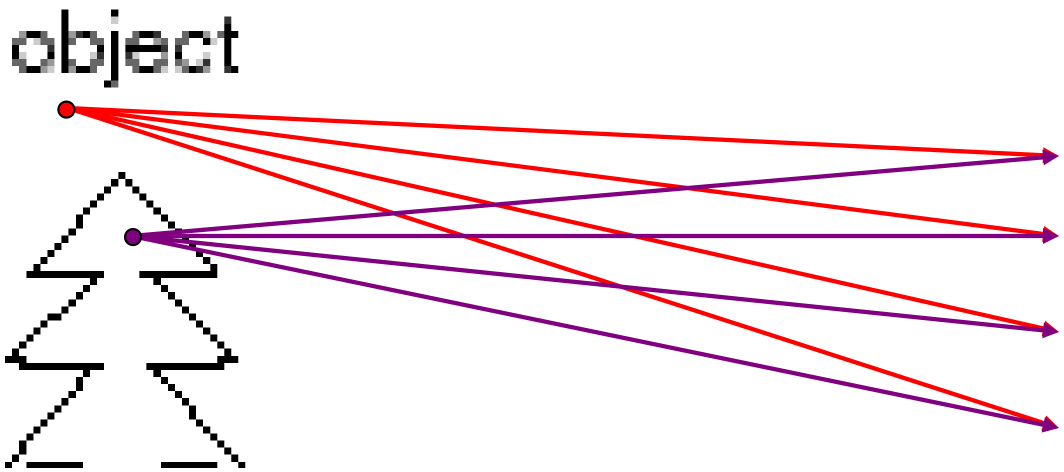
- To see where projections fit in, lets investigate camera models

Camera Models

Jeremy Bolton, PhD
Assistant Teaching Professor

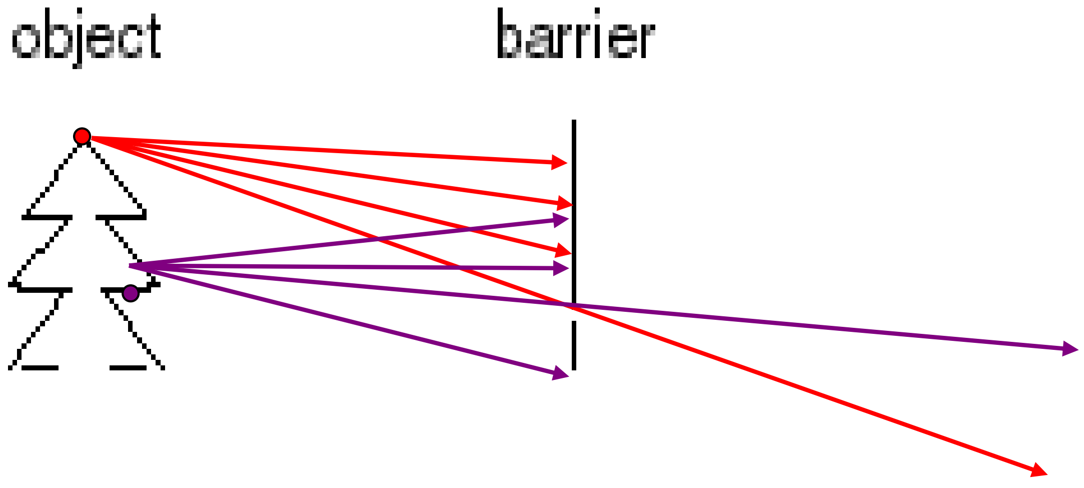
GEORGETOWN UNIVERSITY

Image formation



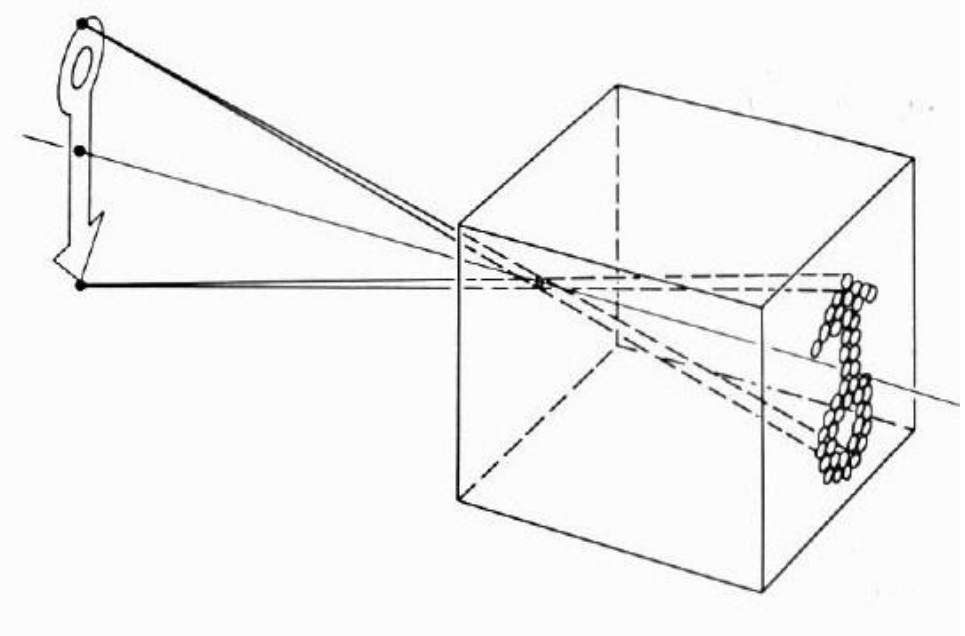
- Let's design a camera
  - Idea 1: put a piece of film in front of an object
  - Do we get a reasonable image?

Pinhole camera



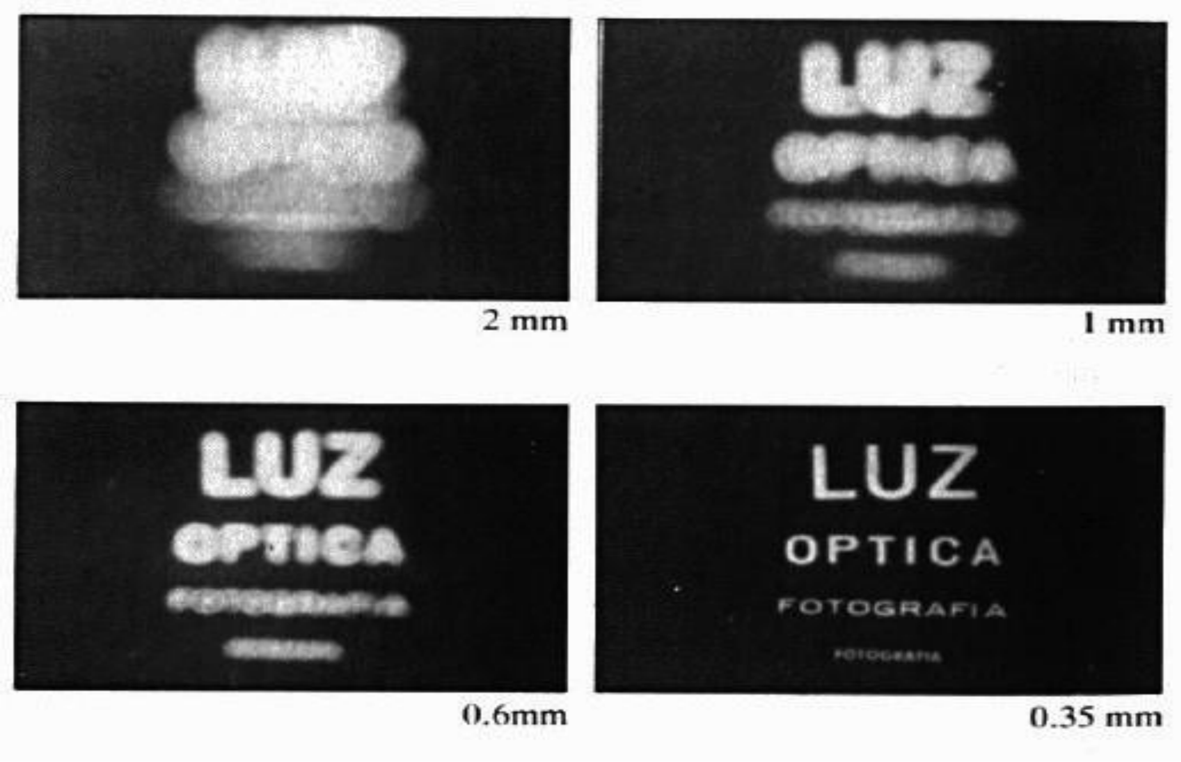
- Add a barrier to block off most of the rays
  - This reduces blurring
  - The opening known as the **aperture**
  - How does this transform the image?

Camera Obscura: Pinhole model



- The first camera
  - Known to Aristotle
  - How does the aperture size affect the image?
- Pinhole model:
  - Captures **pencil of rays** – all rays through a single point
  - The point is called **Center of Projection (focal point)**
  - The image is formed on the **Image Plane**

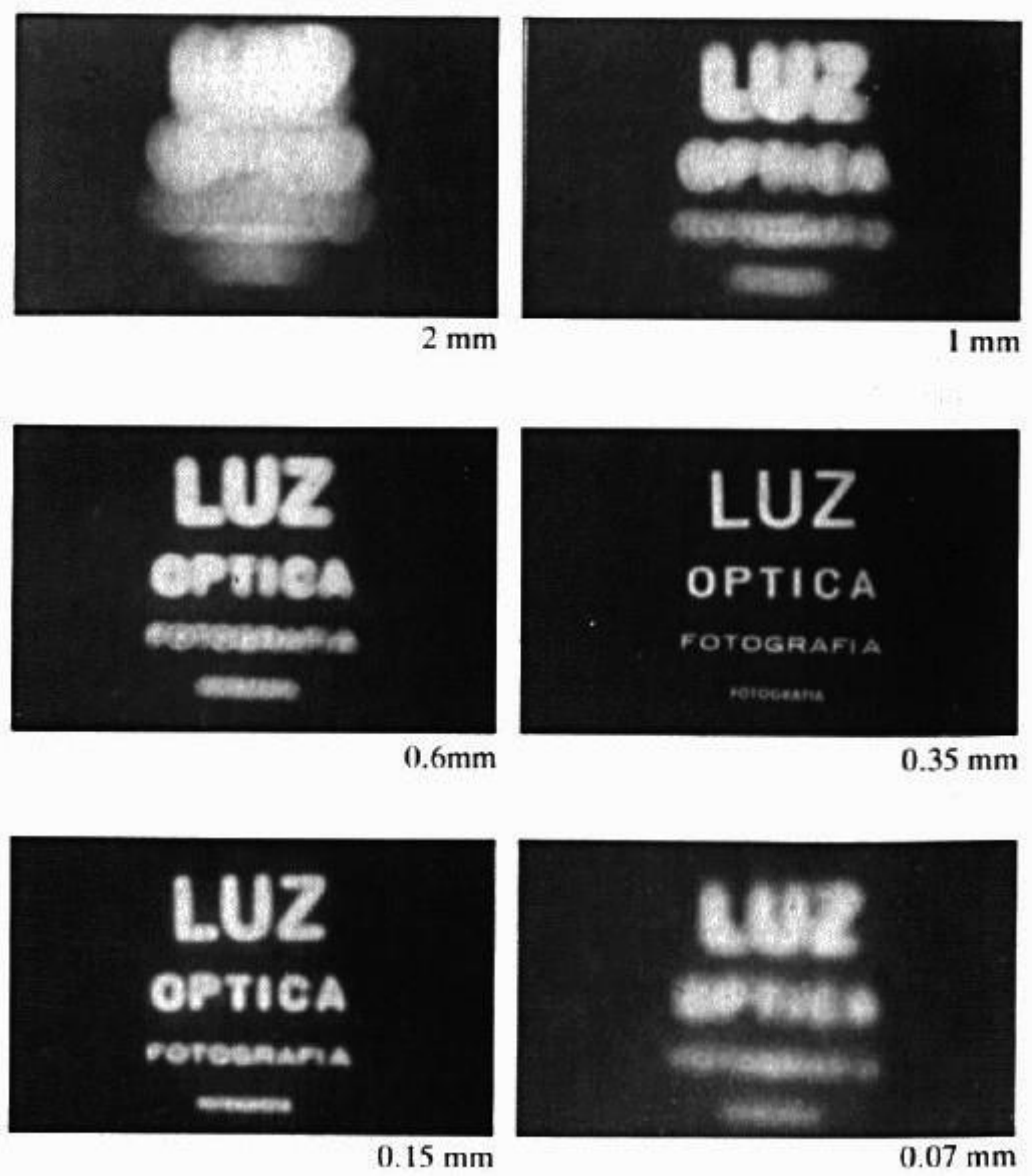
Shrinking the aperture



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

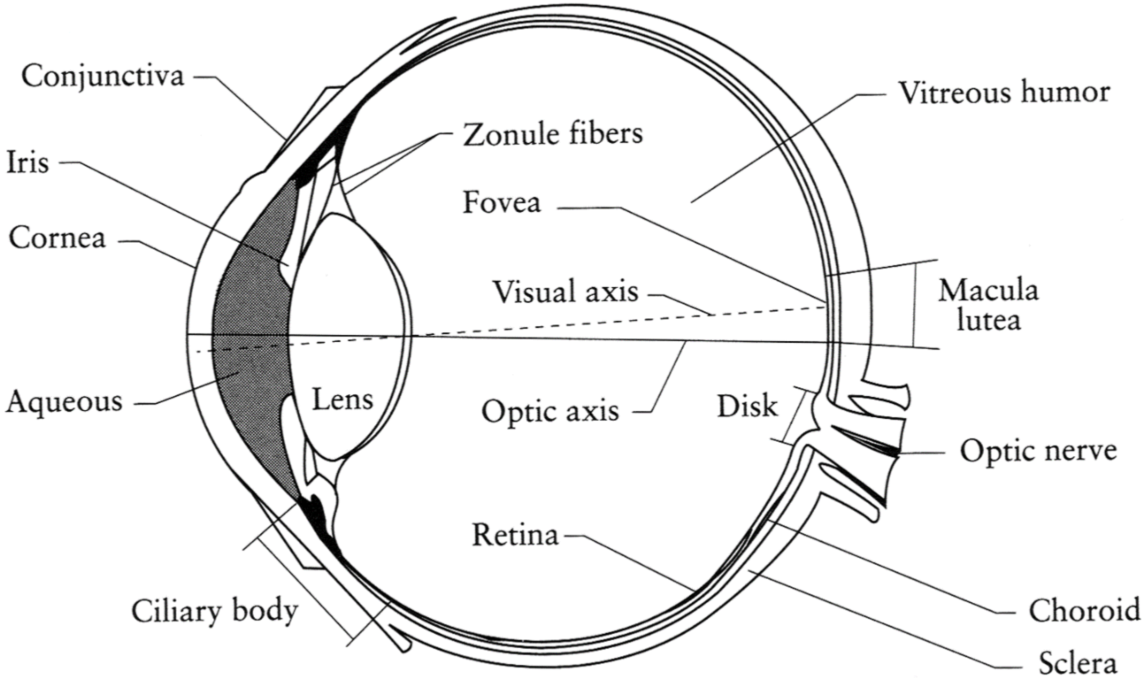
GEORGETOWN UNIVERSITY

Shrinking the aperture



GEORGETOWN UNIVERSITY

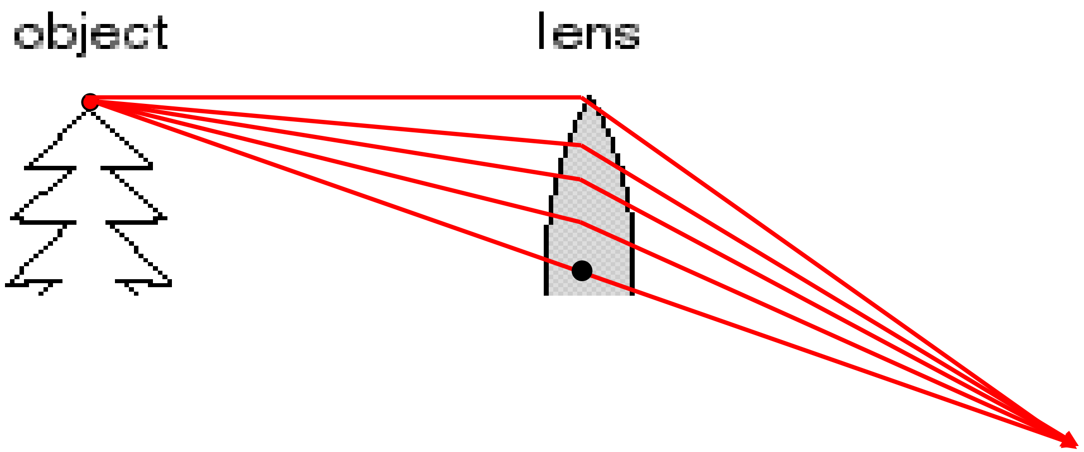
The eye



- 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**

GEORGETOWN UNIVERSITY

Adding a lens

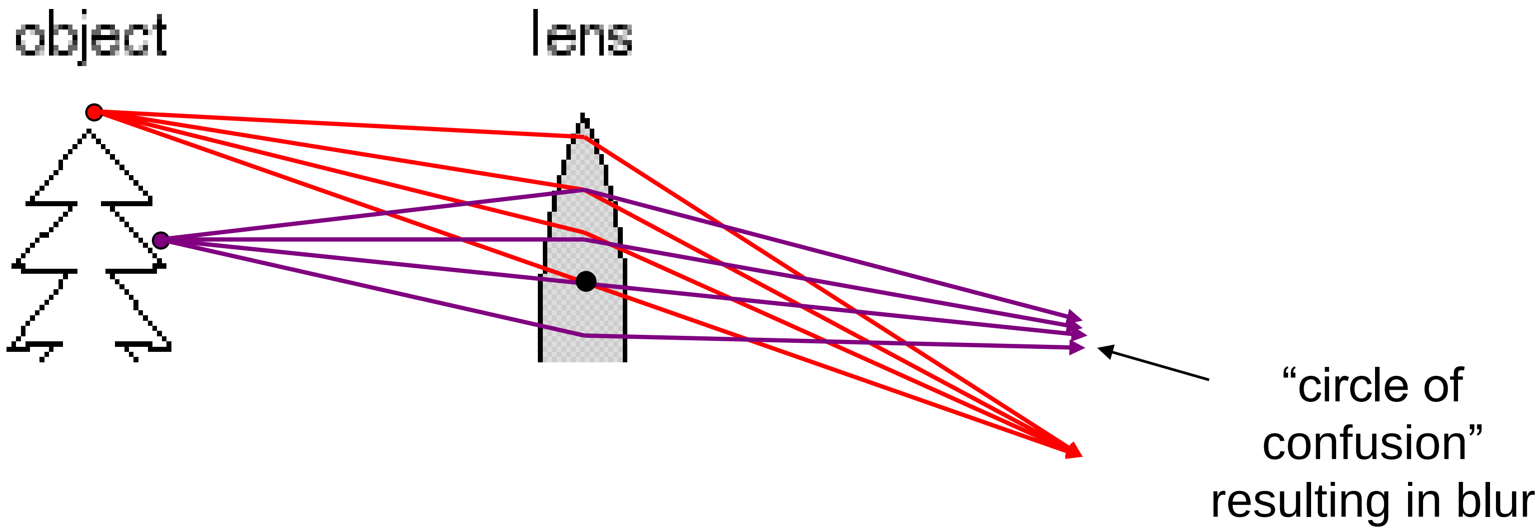


- A lens focuses light onto the film
  - Rays passing through the center are not deviated

GEORGETOWN UNIVERSITY

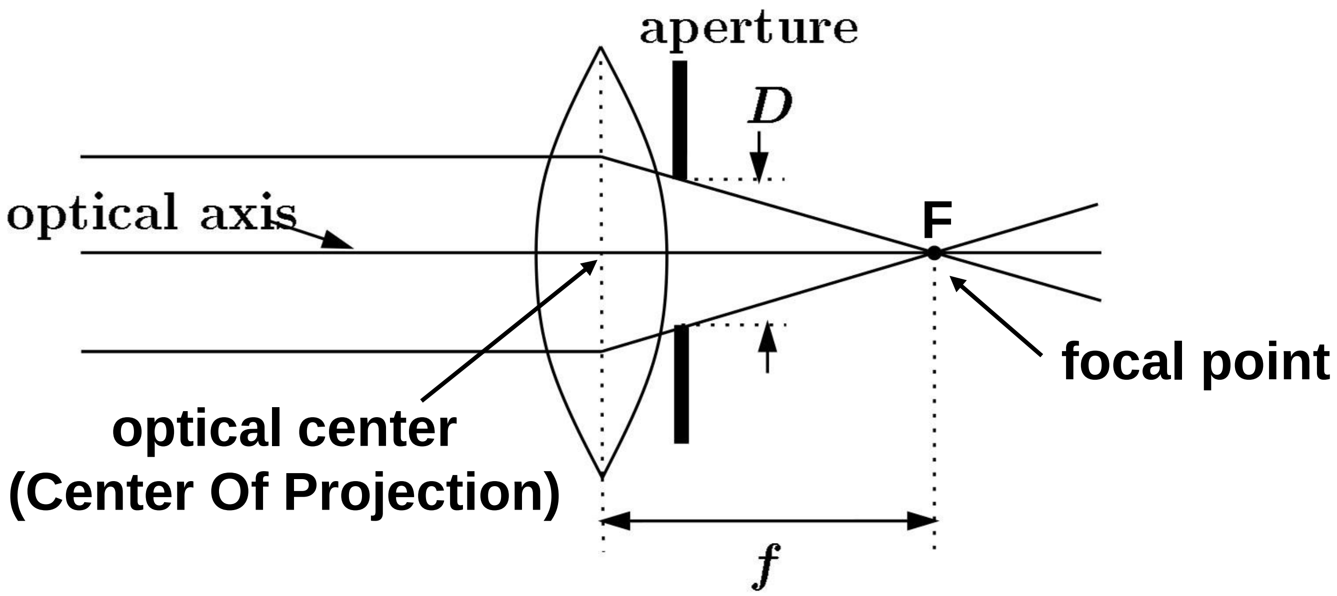
Slide by Steve Seitz

Adding a lens



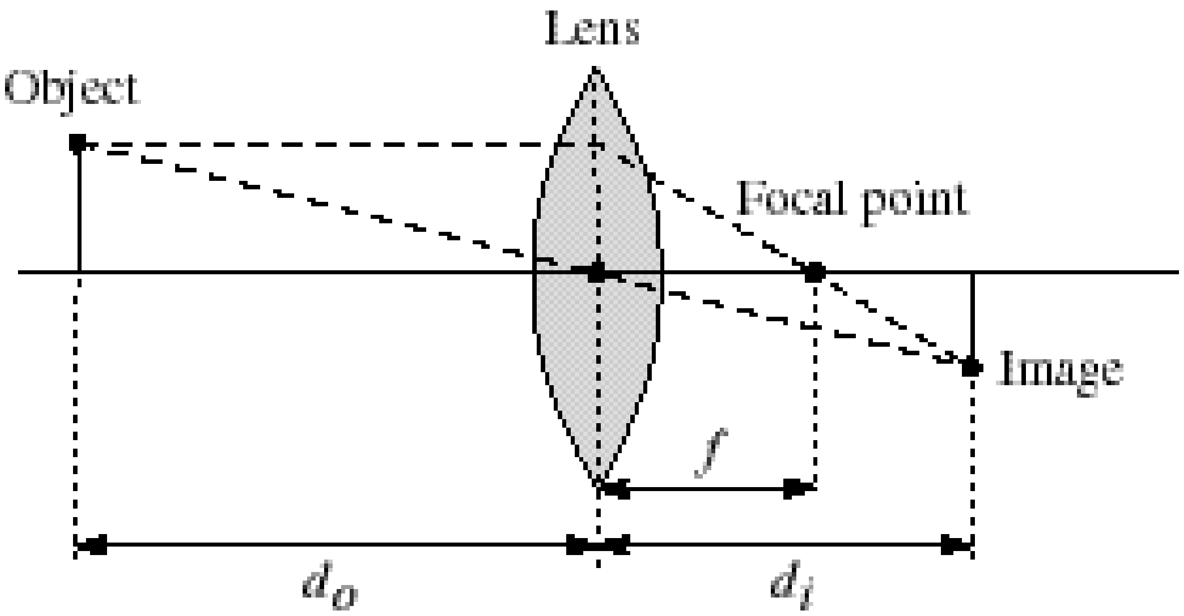
- A lens focuses light onto the film
  - There is a <u>specific distance</u> at which objects are “<u>in focus</u>”
    - other points project to a “circle of confusion” in the image
  - Changing the shape of the lens changes this distance

Lenses



- A lens focuses parallel rays onto a single focal point
  - focal point at a distance *f* beyond the plane of the lens
    - *f* is a function of the shape and index of refraction of the lens
  - Aperture of diameter D restricts the range of rays
    - aperture may be on either side of the lens
  - Lenses are typically spherical (easier to produce)

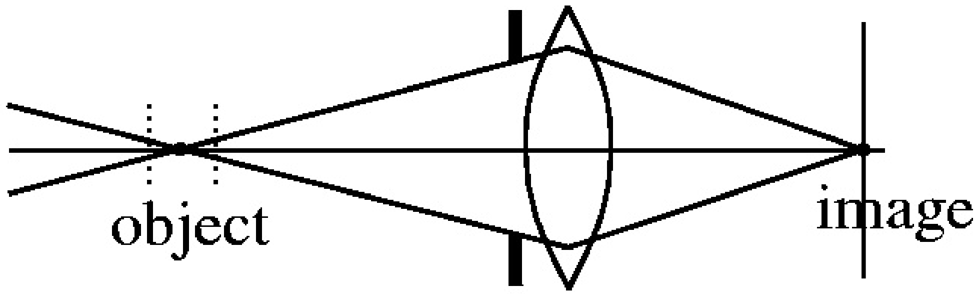
Thin lenses



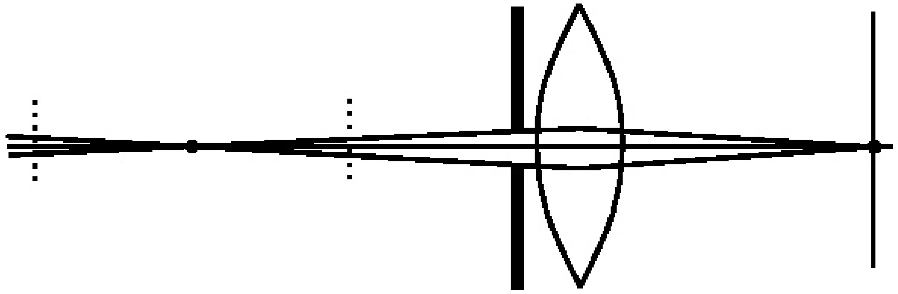
- Thin lens equation: $$\frac{1}{d_o} + \frac{1}{d_i} = \frac{1}{f}$$
  - Any object point satisfying this equation is in focus
  - What is the shape of the focus region?
  - How can we change the focus region?
  - Thin lens applet: http://www.phy.ntnu.edu.tw/java/Lens/lens_e.html (by Fu-Kwun Hwang)

GEORGETOWN UNIVERSITY

Depth of field



f / 5.6



f / 32

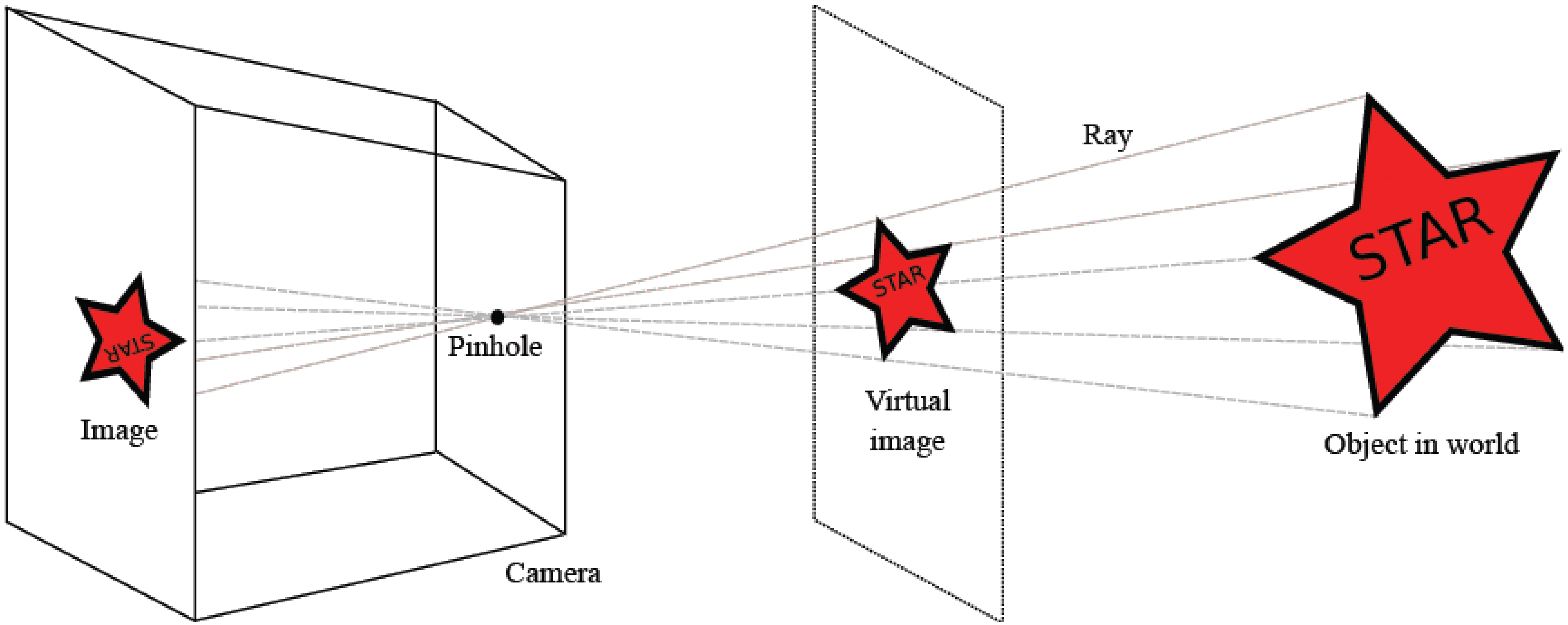
- Changing the aperture size affects depth of field
  - A smaller aperture increases the range in which the object is approximately in focus

GEORGETOWN UNIVERSITY

Projection within the context of camera model

- Given our camera model, lets revisit transformations
- Image plane vs. Virtual image plane
- Image Formation
  - Translation projection: center the image coordinates
  - Affine Projection: accounts for camera position and orientation
  - Perspective Projection: image formation, 3d to 2d

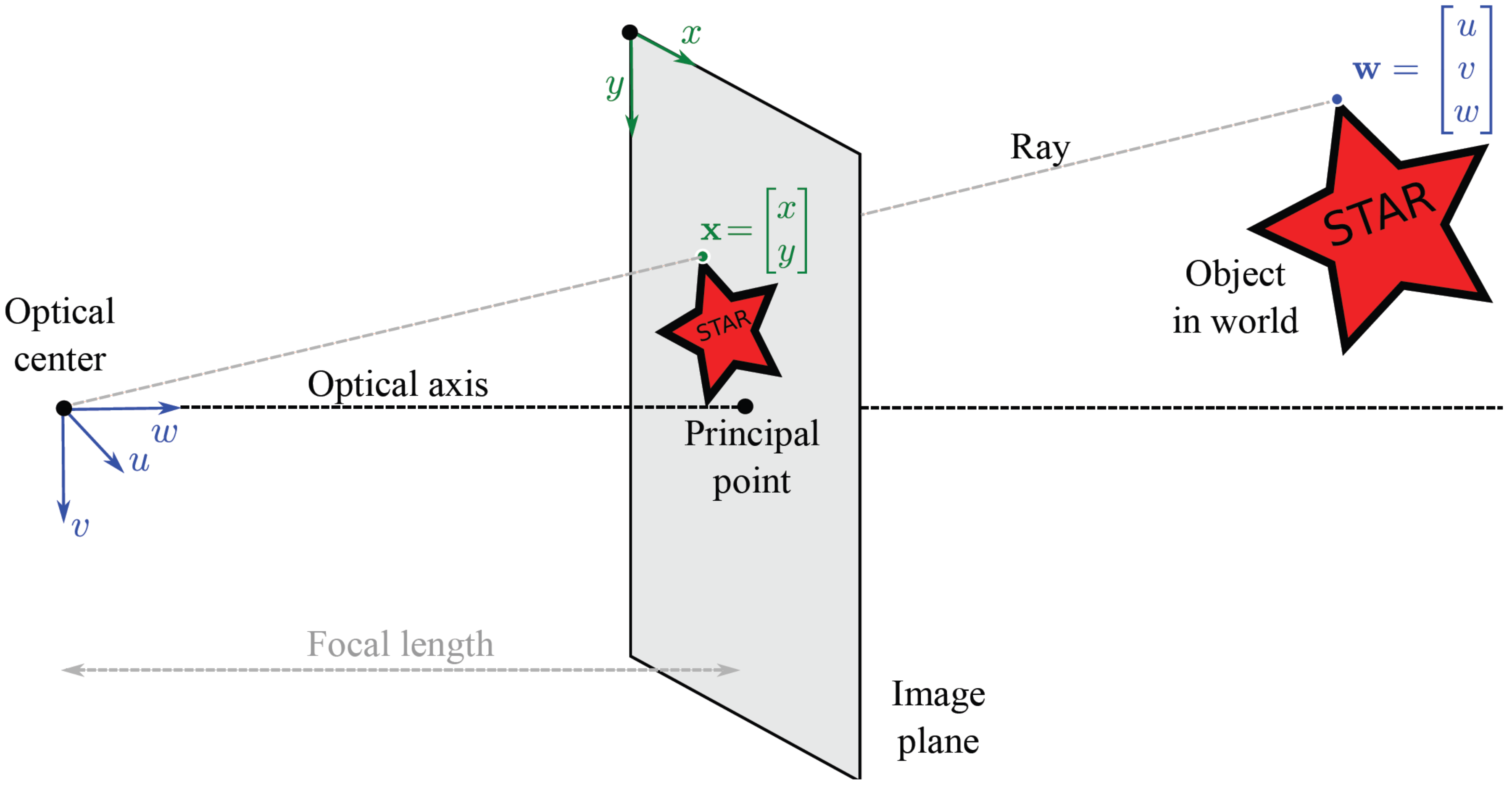
Pinhole camera



Real camera image is inverted

Instead model impossible but more convenient virtual image

Pinhole camera terminology

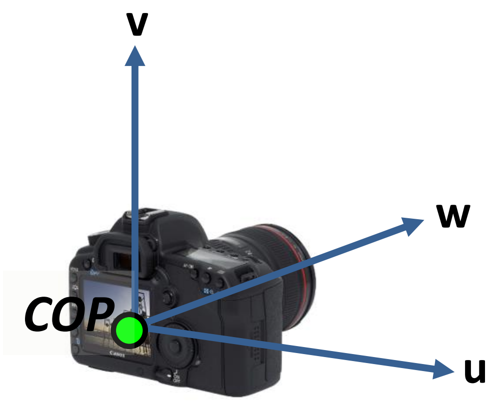


Deriving a Model for a Pinhole Camera

- We will first derive the pinhole camera model algebraically.
- Then we will revisit from a geometric perspective.

Camera parameters

- How can we model the geometry of a camera?



Camera

"The World"

Two important coordinate systems:

1. *World* coordinate system
2. *Camera* coordinate system

GEORGETOWN UNIVERSITY

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)
- The project into the image plane
  - Need to know camera *intrinsics*

Perspective Projection Matrix

Observe: We can solve for image coordinates in terms of real-world coordinates and transformation parameters

- Projection is a matrix multiplication 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 the third coordinate

In practice: lots of coordinate transformations…

$$\begin{pmatrix} \text{2D} \\ \text{point} \\ \text{(3x1)} \end{pmatrix} = \begin{pmatrix} \text{Camera to} \\ \text{pixel coord.} \\ \text{trans. matrix} \\ \text{(3x3)} \end{pmatrix} \begin{pmatrix} \text{Perspective} \\ \text{projection matrix} \\ \text{(3x4)} \end{pmatrix} \begin{pmatrix} \text{World to} \\ \text{camera coord.} \\ \text{trans. matrix} \\ \text{(4x4)} \end{pmatrix} \begin{pmatrix} \text{3D} \\ \text{point} \\ \text{(4x1)} \end{pmatrix}$$

GEORGETOWN UNIVERSITY

Perspective projection (Intrinsics)

$$\underbrace{\begin{bmatrix} -f & 0 & 0 \\ 0 & -f & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\substack{\mathbf{K} \\ \text{(intrinsics)}}} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

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

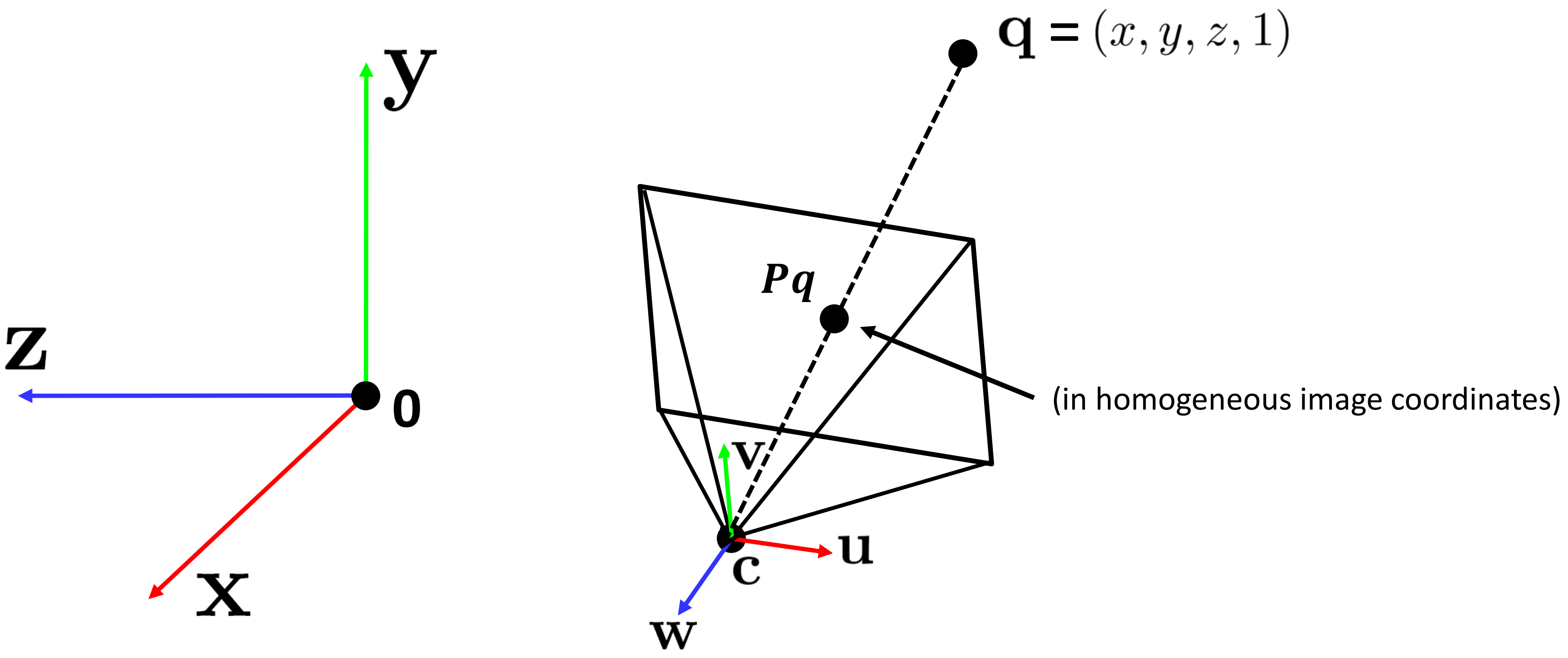
in general, $\mathbf{K} = \begin{bmatrix} -f & s & c_x \\ 0 & -\alpha f & c_y \\ 0 & 0 & 1 \end{bmatrix}$ (upper triangular matrix)

α : **aspect ratio** (1 unless pixels are not square)

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

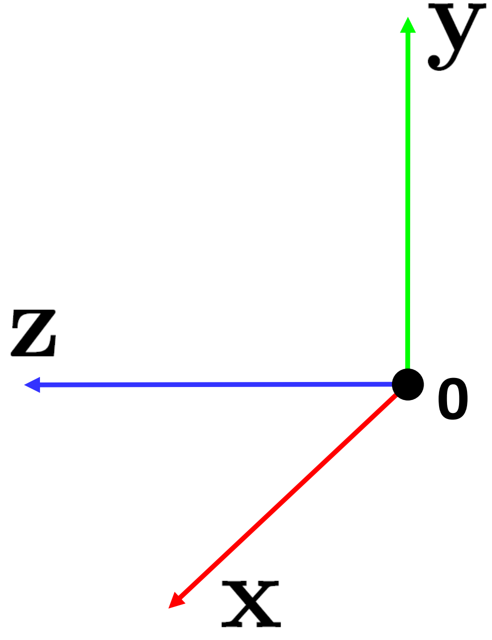
(c_x, c_y) : **principal point** ((0,0) unless optical axis doesn't intersect projection plane at origin)

Projection matrix (Extrinsics)

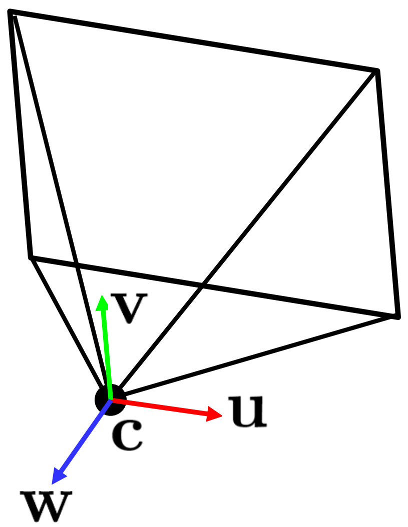


Extrinsics

- How do we get the camera to "canonical form"?
  - (Center of projection at the origin, x-axis points right, y-axis points up, z-axis points backwards)



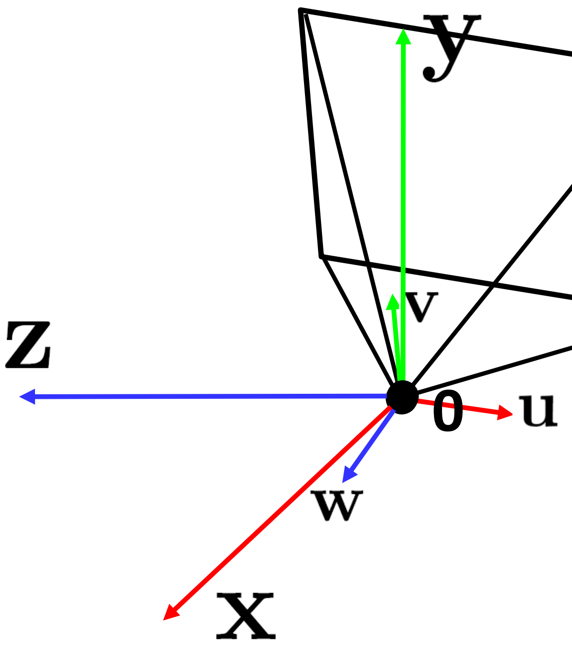
Step 1: Translate by -**c**



GEORGETOWN UNIVERSITY

Extrinsics

- How do we get the camera to “canonical form”?
  - (Center of projection at the origin, x-axis points right, y-axis points up, z-axis points backwards)



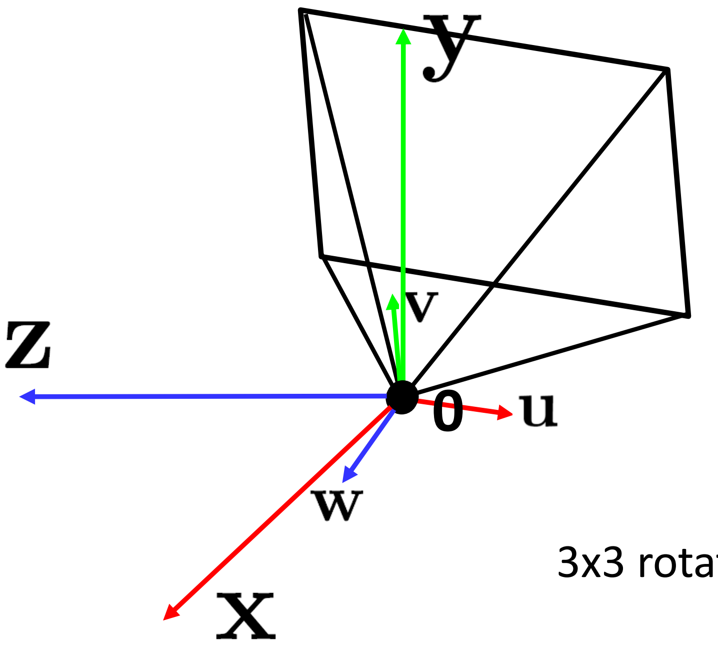
Step 1: Translate by -**c**

How do we represent translation as a matrix multiplication?

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

Extrinsics

- How do we get the camera to “canonical form”?
  - (Center of projection at the origin, x-axis points right, y-axis points up, z-axis points backwards)



Step 1: Translate by -**c**

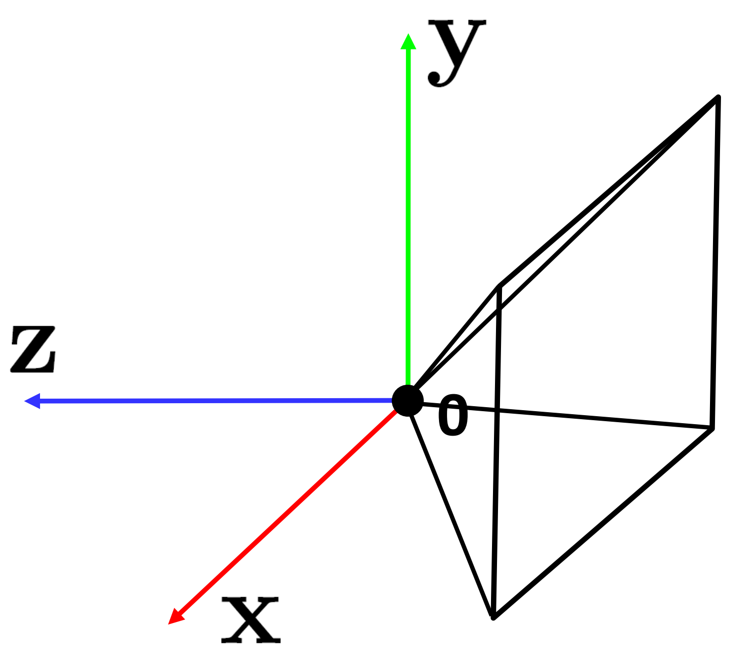
Step 2: Rotate by **R**

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

3x3 rotation matrix

Extrinsics

- How do we get the camera to "canonical form"?
  - (Center of projection at the origin, x-axis points right, y-axis points up, z-axis points backwards)



Step 1: Translate by -**c**

Step 2: Rotate by **R**

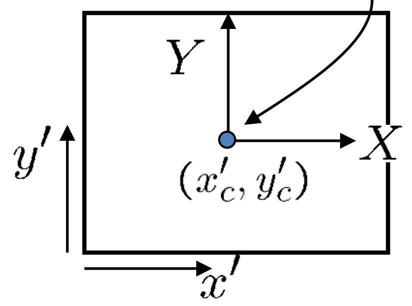
$$\mathbf{R} = \begin{bmatrix} \mathbf{u}^T \\ \mathbf{v}^T \\ \mathbf{w}^T \end{bmatrix}$$

Camera parameters

A camera is described by several parameters

- Translation t of the optical center from the origin of world coords
- Rotation R of the image plane
- focal length f, principle point (x'_c, y'_c),
- blue parameters are called "extrinsics," red are "intrinsics"

Projection equation



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

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

$$\mathbf{P}_{3x4} = \mathbf{K}[\mathbf{R}\,|\,\mathbf{t}] = \begin{bmatrix} -f & s & x'_c \\ 0 & -f & y'_c \\ 0 & 0 & 1 \end{bmatrix} [\mathbf{R}_{3x3} \,|\, \mathbf{t}_{3x1}]$$

$$\tilde{\mathbf{P}}_{4x4} = \tilde{\mathbf{K}}\mathbf{E} = \underbrace{\begin{bmatrix} \mathbf{K} & \mathbf{0}^T \\ \mathbf{0}^\mathbf{T} & 1 \end{bmatrix}}_{\text{intrinsics}} \begin{bmatrix} \mathbf{I}_{3x3} & \mathbf{0} \\ \mathbf{0}_{1x3} & 0 \end{bmatrix} \underbrace{\begin{bmatrix} \mathbf{R}_{3x3} & \\ \mathbf{0}_{1x3} & \end{bmatrix}}_{\text{rotation}} \underbrace{\begin{bmatrix} & \mathbf{t}_{3x1} \\ & 1 \end{bmatrix}}_{\text{translation}}$$

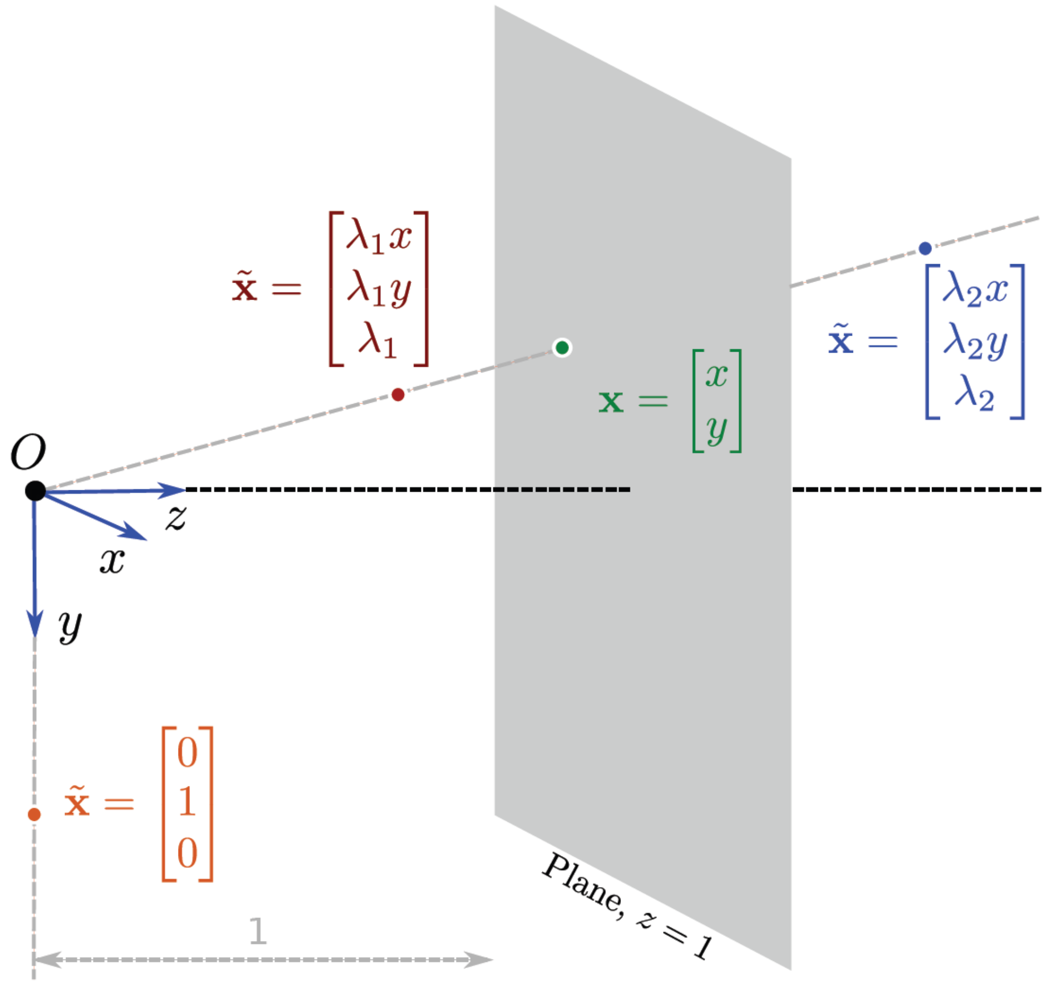
- Note: The definitions of these parameters are **not** completely standardized

Pinhole Camera Model

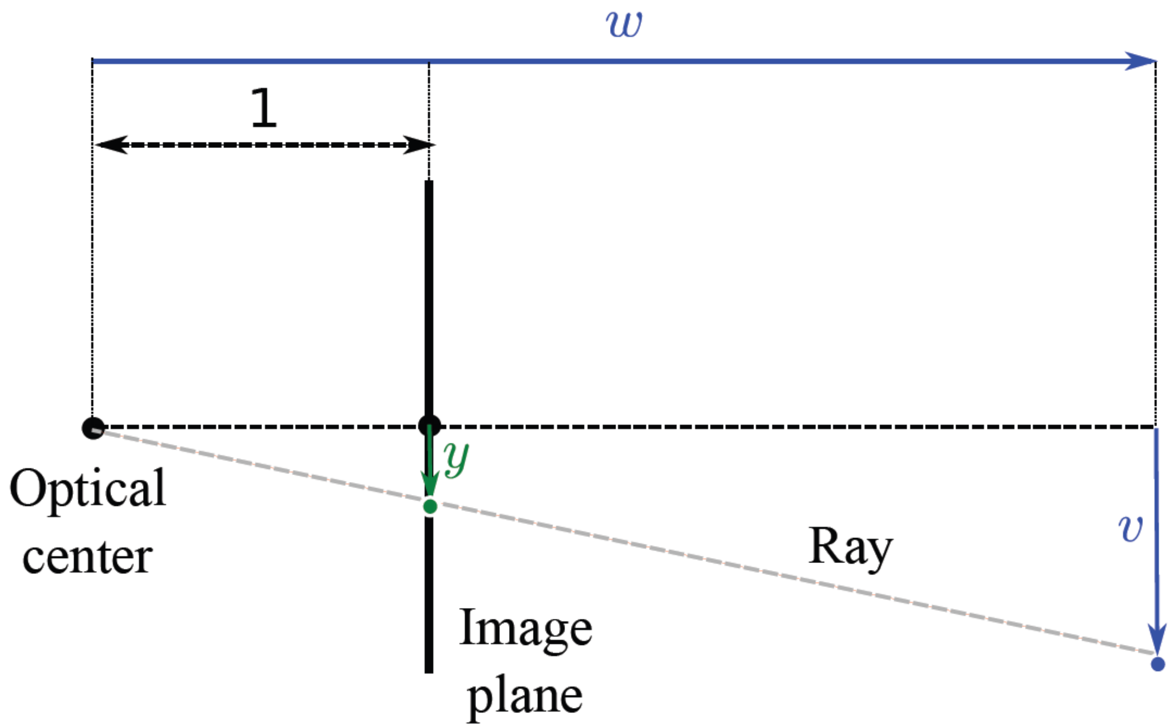
- Derive from a geometric perspective.

- Using Princes Notation.

Geometric interpretation of homogeneous coordinates



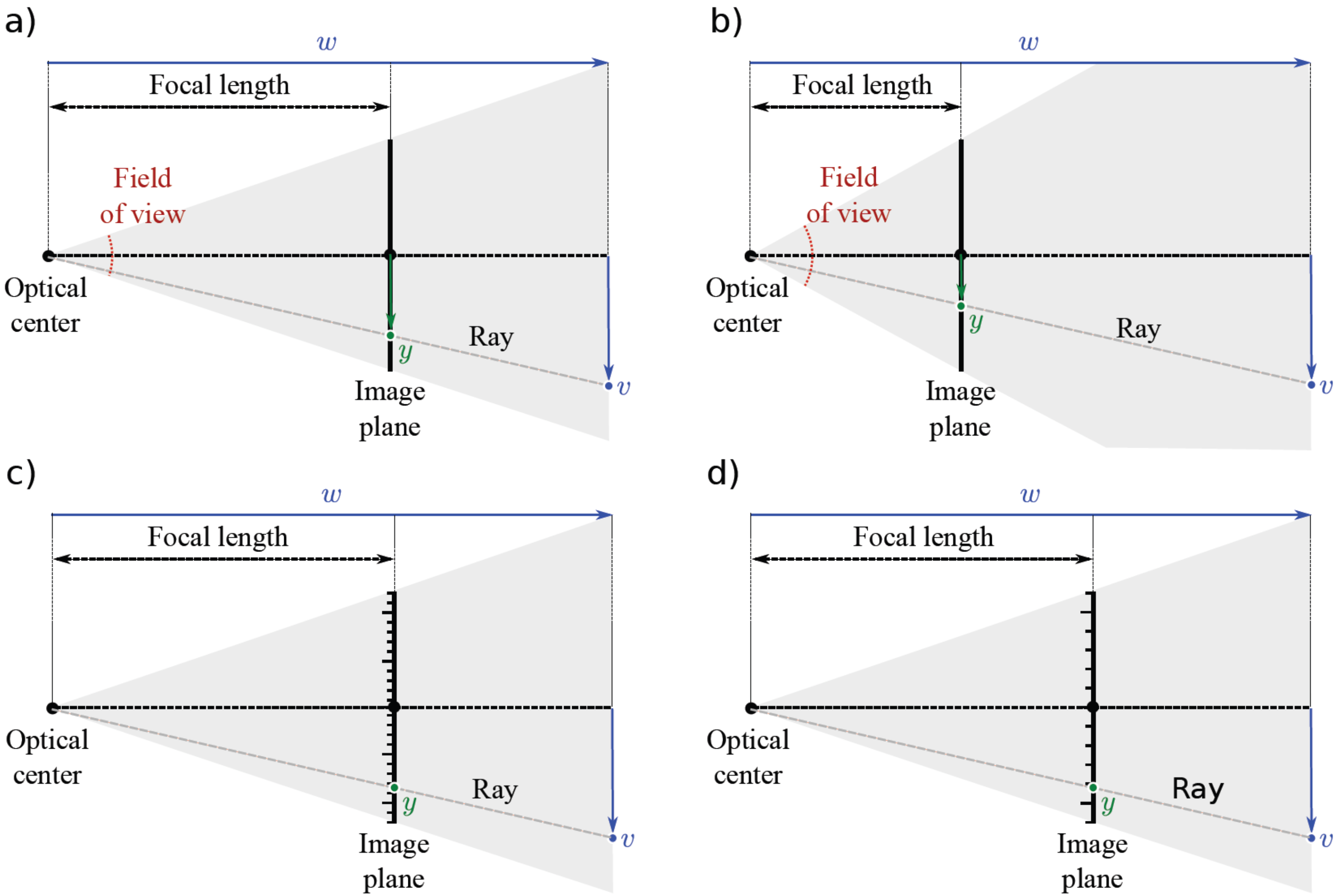
Normalized Camera



By similar triangles:

$$x = \frac{u}{w} \qquad y = \frac{v}{w}$$

Focal length parameters



Focal length parameters

Can model both

- the effect of the distance to the focal plane
- the density of the receptors

with a single **focal length parameter f = d = ϕ**

$$x = \frac{\phi u}{w} \qquad y = \frac{\phi v}{w}$$

<u>To be overly cumbersome note:</u> In practice, the receptors may not be square:

$$x = \frac{\phi_x u}{w} \qquad y = \frac{\phi_y v}{w}$$

So use different focal length parameter for x and y dims

Offset parameters

- Current model assumes that pixel (0,0) is where the principal ray strikes the image plane (i.e. the center)
- Model offset to center

$$
\begin{aligned}
x &= \frac{\phi_x u}{w} + \delta_x \\
y &= \frac{\phi_y v}{w} + \delta_y
\end{aligned}
$$

Skew parameter

- Finally, add skew parameter
- Accounts for image plane being not exactly perpendicular to the principal ray

$$
\begin{aligned}
x &= \frac{\phi_x u + \gamma v}{w} + \delta_x \\
y &= \frac{\phi_y v}{w} + \delta_y
\end{aligned}
$$

Pinhole camera in homogeneous coordinates

Camera model:

$$\begin{aligned} x &= \frac{\phi_x u + \gamma v}{w} + \delta_x \\ y &= \frac{\phi_y v}{w} + \delta_y, \end{aligned}$$

In homogeneous coordinates:

$$\lambda \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} \phi_x & \gamma & \delta_x & 0 \\ 0 & \phi_y & \delta_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} u \\ v \\ w \\ 1 \end{bmatrix}$$

(linear!)

Pinhole camera in homogeneous coordinates

$$\lambda \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} \phi_x & \gamma & \delta_x & 0 \\ 0 & \phi_y & \delta_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} u \\ v \\ w \\ 1 \end{bmatrix}$$

Writing out these three equations

$$\begin{aligned} \lambda x &= \phi_x u + \gamma v + \delta_x w \\ \lambda y &= \phi_y v + \delta_y w \\ \lambda &= w. \end{aligned}$$

Eliminate λ to retrieve original equations

GEORGETOWN UNIVERSITY

Position and orientation of camera

- Position $\mathbf{w}=(u,v,w)^T$ of point in the world is generally not expressed in the frame of reference of the camera.
- Transform using 3D transformation

$$\begin{bmatrix} u' \\ v' \\ w' \end{bmatrix} = \begin{bmatrix} \omega_{11} & \omega_{12} & \omega_{13} \\ \omega_{21} & \omega_{22} & \omega_{23} \\ \omega_{31} & \omega_{32} & \omega_{33} \end{bmatrix} \begin{bmatrix} u \\ v \\ w \end{bmatrix} + \begin{bmatrix} \tau_x \\ \tau_y \\ \tau_z \end{bmatrix}$$

or

$$\mathbf{w}' = \boldsymbol{\Omega}\mathbf{w} + \boldsymbol{\tau}$$

Point in frame of reference of camera

Point in frame of reference of world

Complete pinhole camera model

$$
\begin{aligned}
x &= \frac{\phi_x(\omega_{11}u + \omega_{12}v + \omega_{13}w + \tau_x) + \gamma(\omega_{21}u + \omega_{22}v + \omega_{23}w + \tau_y)}{\omega_{31}u + \omega_{32}v + \omega_{33}w + \tau_z} + \delta_x \\
y &= \frac{\phi_y(\omega_{21}u + \omega_{22}v + \omega_{23}w + \tau_y)}{\omega_{31}u + \omega_{32}v + \omega_{33}w + \tau_z} + \delta_y.
\end{aligned}
$$

- Intrinsic parameters (stored as intrinsic matrix)

$$\{\phi_x, \phi_y, \gamma, \delta_x, \delta_y\}$$

$$\mathbf{\Lambda} = \begin{bmatrix} \phi_x & \gamma & \delta_x \\ 0 & \phi_y & \delta_y \\ 0 & 0 & 1 \end{bmatrix}$$

- Extrinsic parameters

$$\{\mathbf{\Omega}, \boldsymbol{\tau}\}$$

At the end of the day…

- Scene geometry consists of internal and external parameters
- Observed light is focused onto an image plane
- The image is then captured (Image Acquisition)
- But first we will review radiometry: physics of LIGHT and COLOR