SAIR

Spatial AI & Robotics Lab

CSE 473/573-A
L2: CAMERA MODEL

Chen Wang

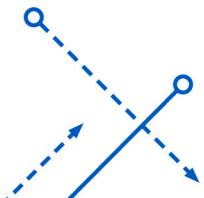
Spatial AI & Robotics Lab

Department of Computer Science and Engineering

University at Buffalo The State University of New York

Content

- Pinhole Camera
  - Perspective Projection
  - Projective Geometry
  - Vanishing Points and Lines
  - Homogeneous Coordinates
  - Matrix form of Perspective Projection
  - Intrinsic and Extrinsic Parameters
- Other type of projection
  - Weak Perspective Projection
  - Photo Finish Photography
  - 360 Imaging
  - Tilt-shift Imaging
- Camera Calibration



SAIR

Spatial AI & Robotics Lab

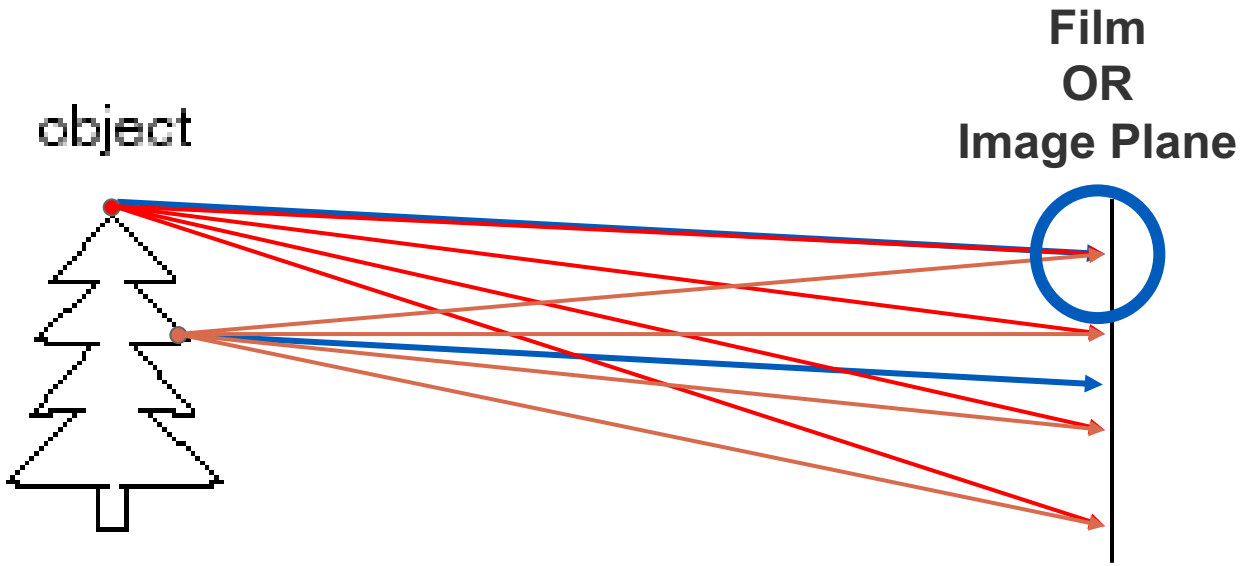
IMAGE FORMATION I

Pinhole Camera

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?

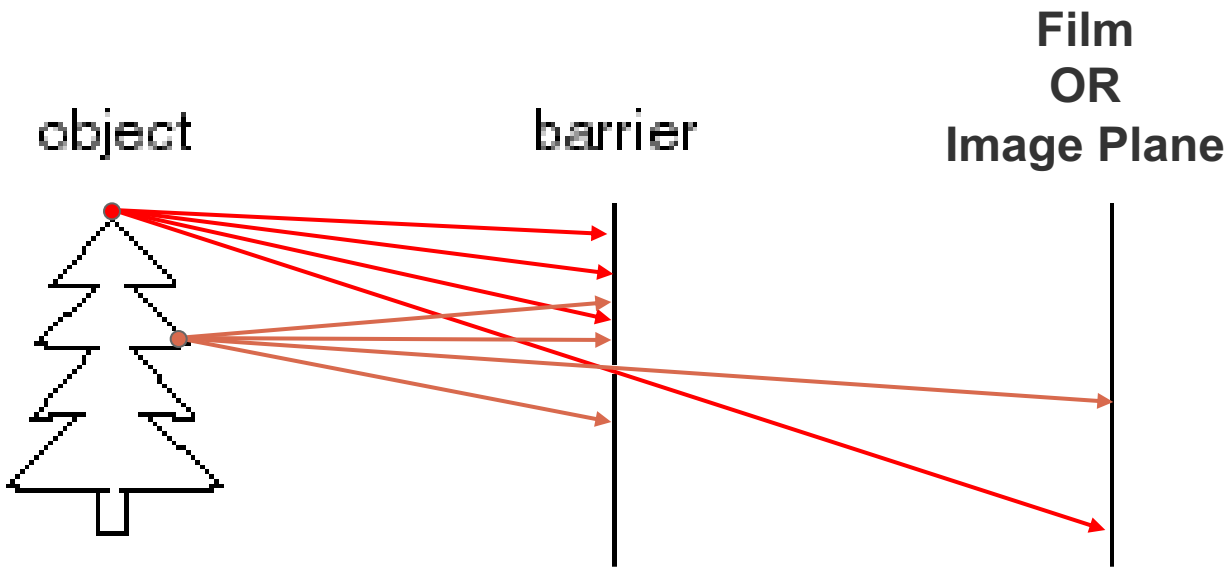


Answer is No….

Pinhole camera

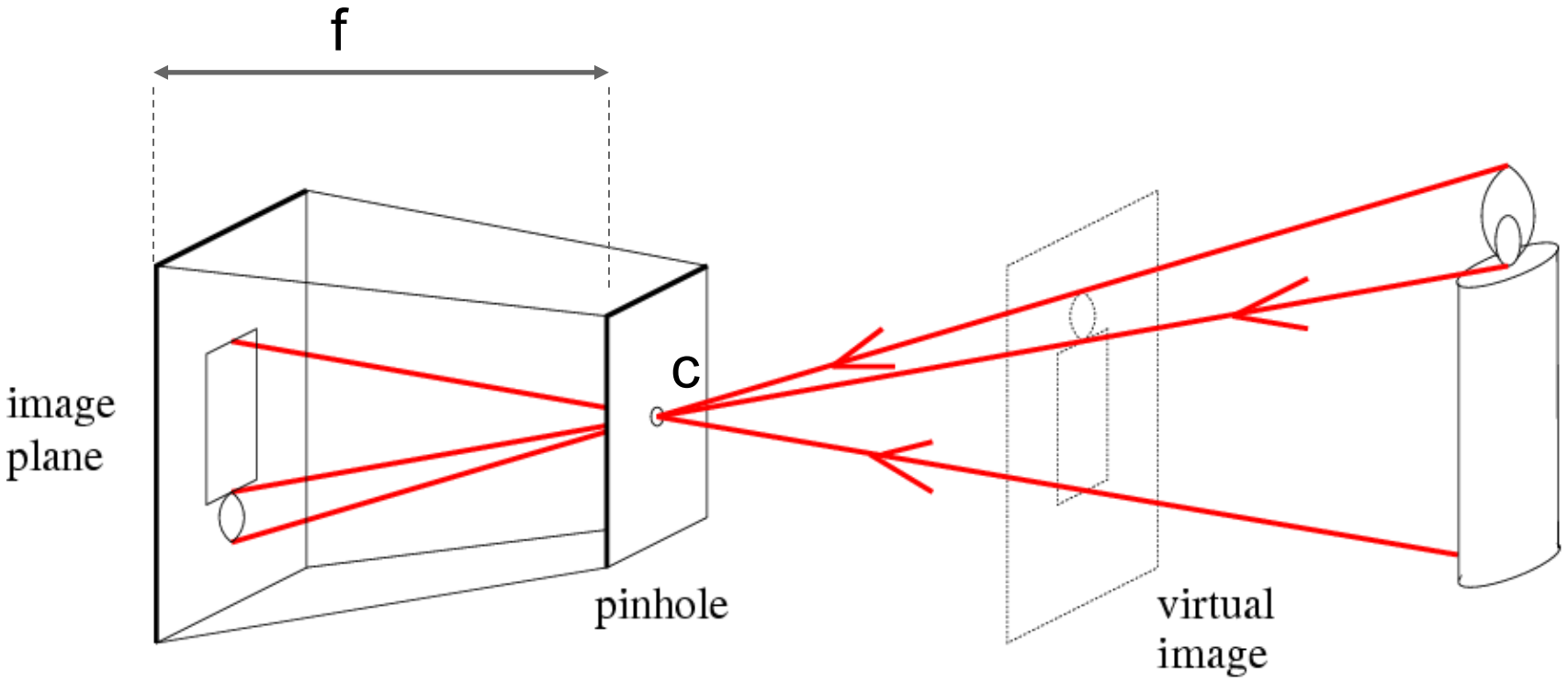
Idea 2: add a barrier to block off most of the rays

- This reduces blurring
- The opening known as the **aperture**



Slide source: Seitz

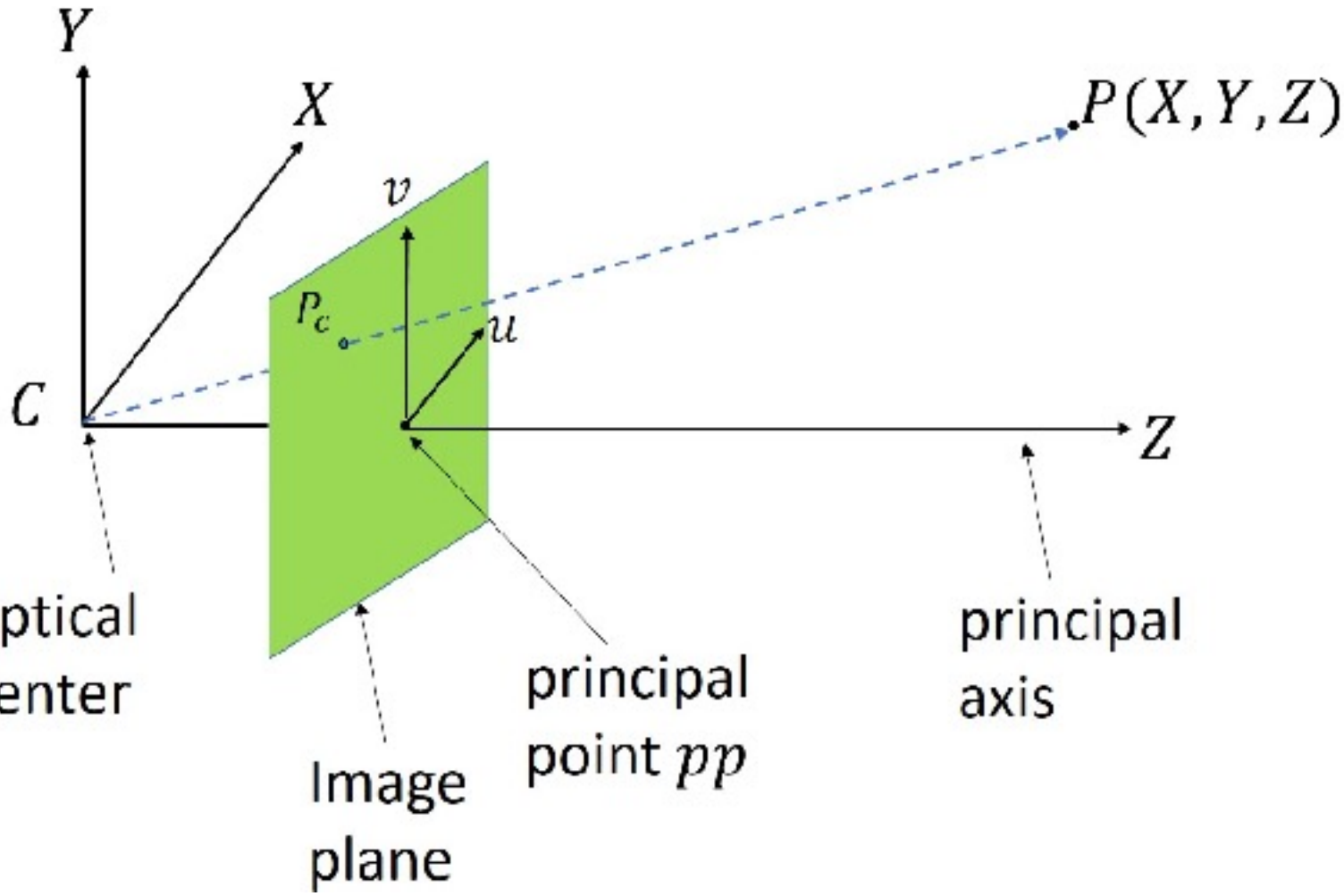
Pinhole camera



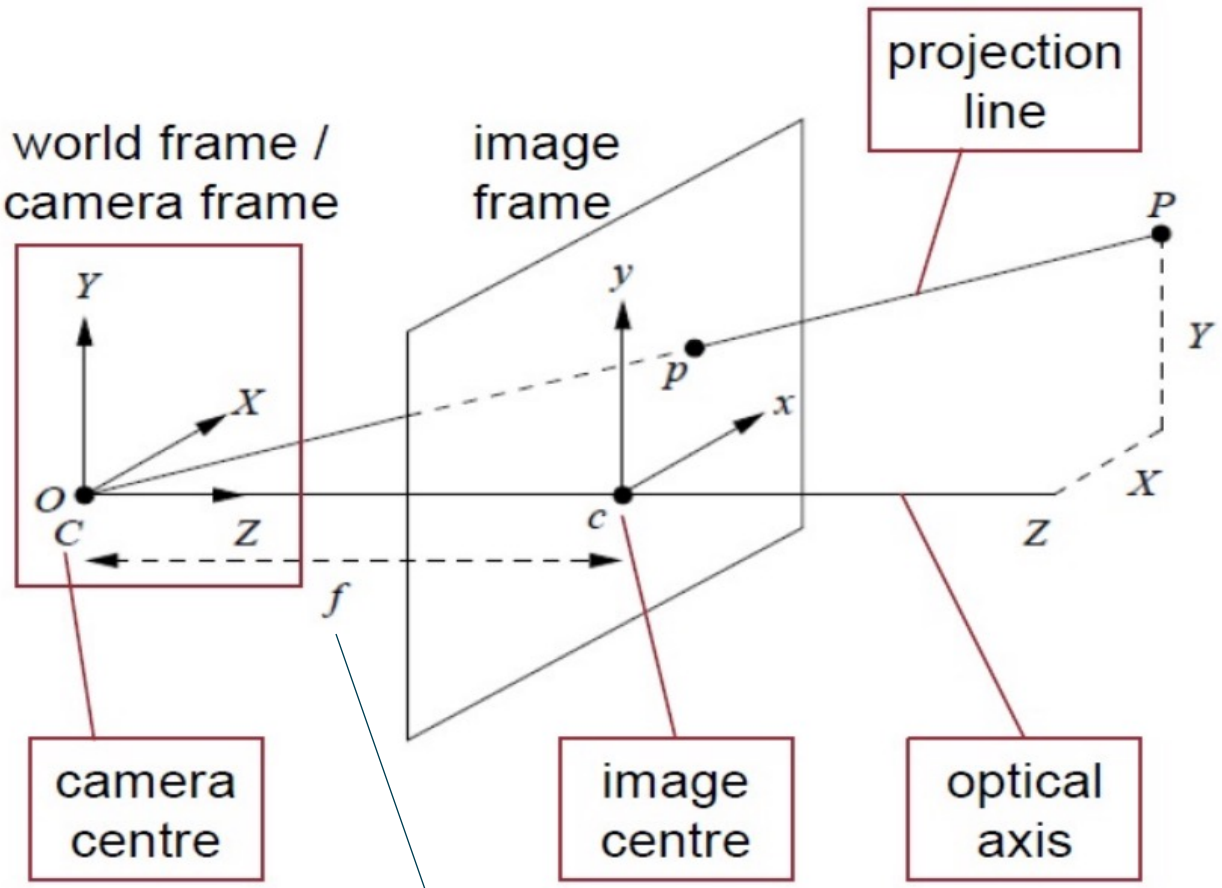
f = focal length
c = center of the camera

Figure from Forsyth

Perspective Projection

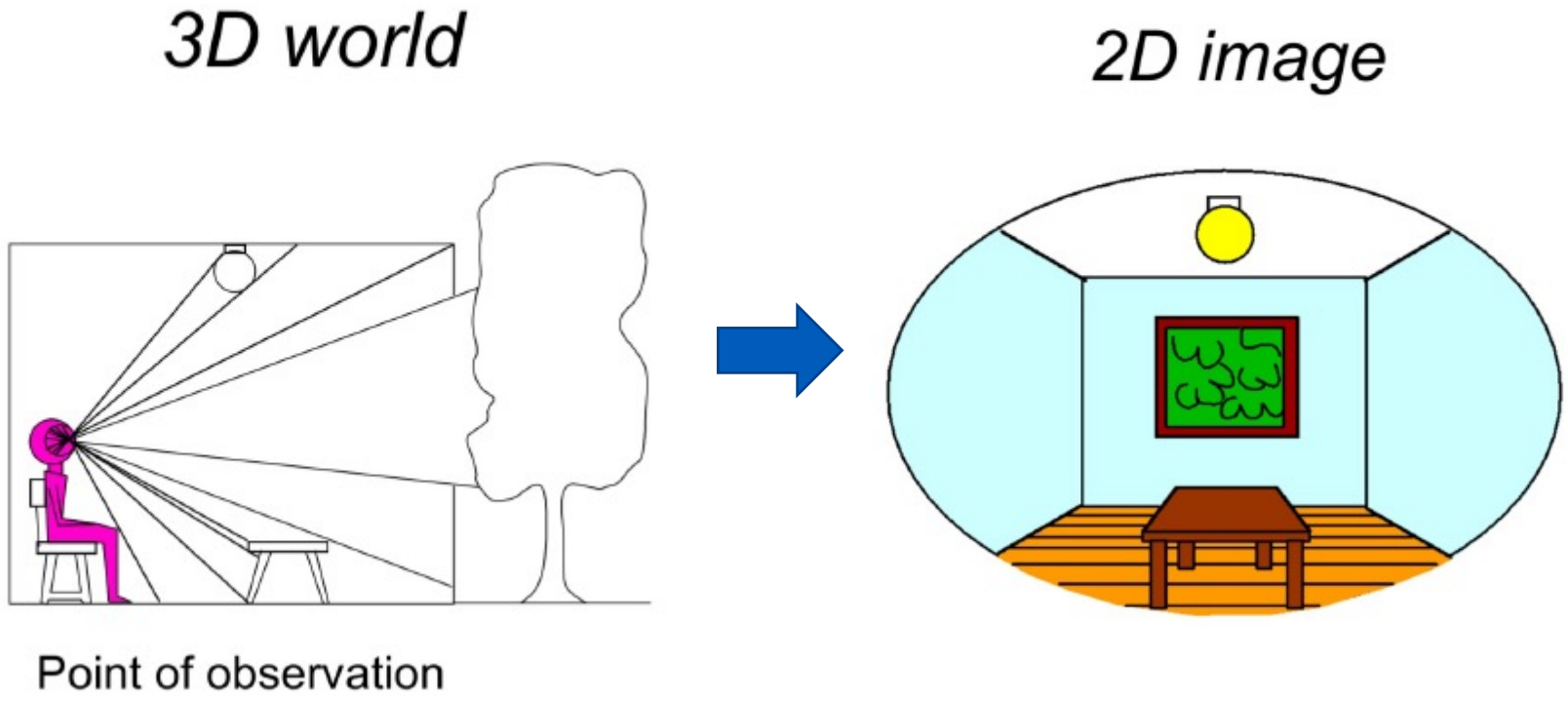


Other names


projection line
world frame / camera frame
image frame
P
Y
y
p
x
X
O
C
Z
c
Z
X
f
camera centre
image centre
optical axis
Focal length

What do we lose?

- Projection from 3D world to 2D Image



Long range objects appear to be small.

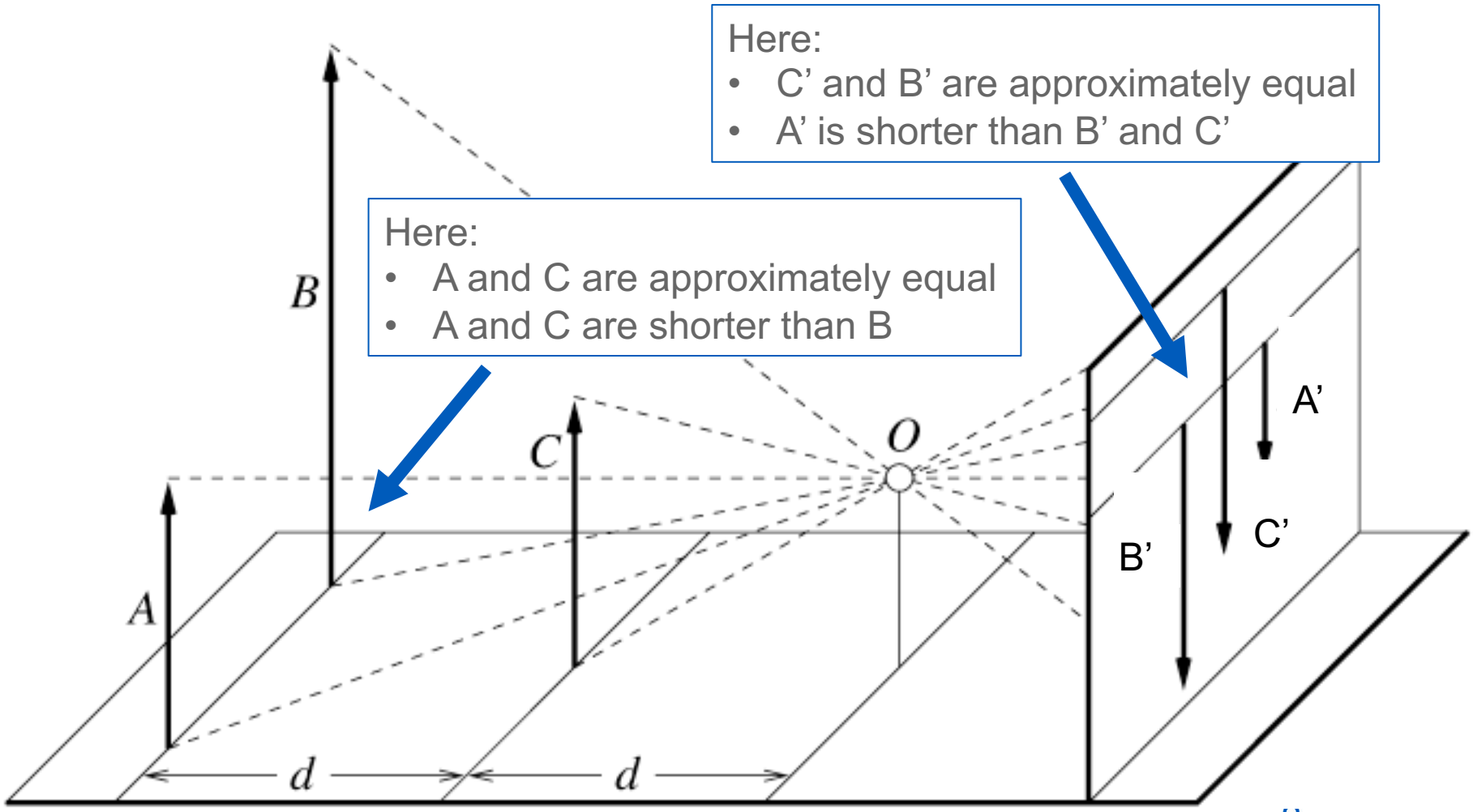
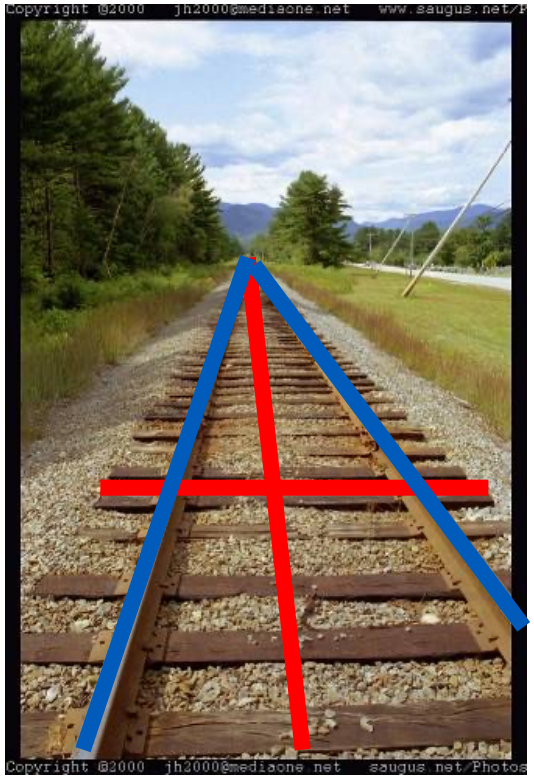


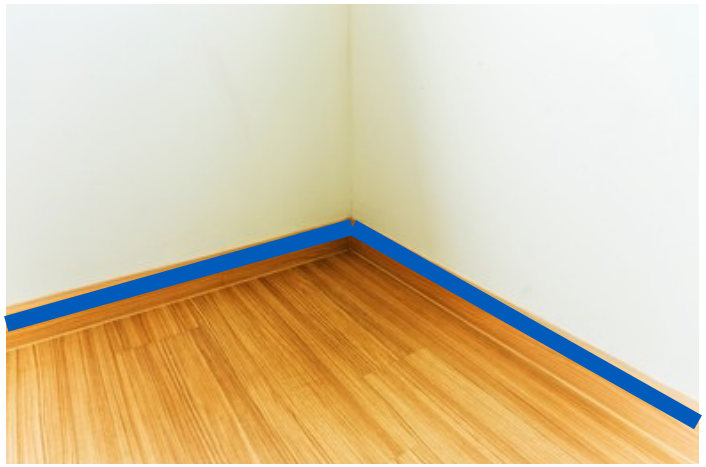
Figure by David Forsyth

Perspective Projection - Example

Perspective Projection - Example

What do we lose geometrically?





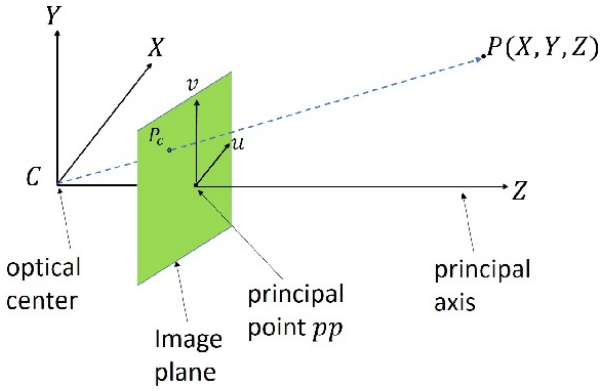
- **Angles**
- **Distances**
- **and therefore Area**

Geometrical Properties

Many-to-one:

Points along same ray map to same point in image.

In the Image



Points map to → ?

- **points**

Lines map to → ?

- **lines**

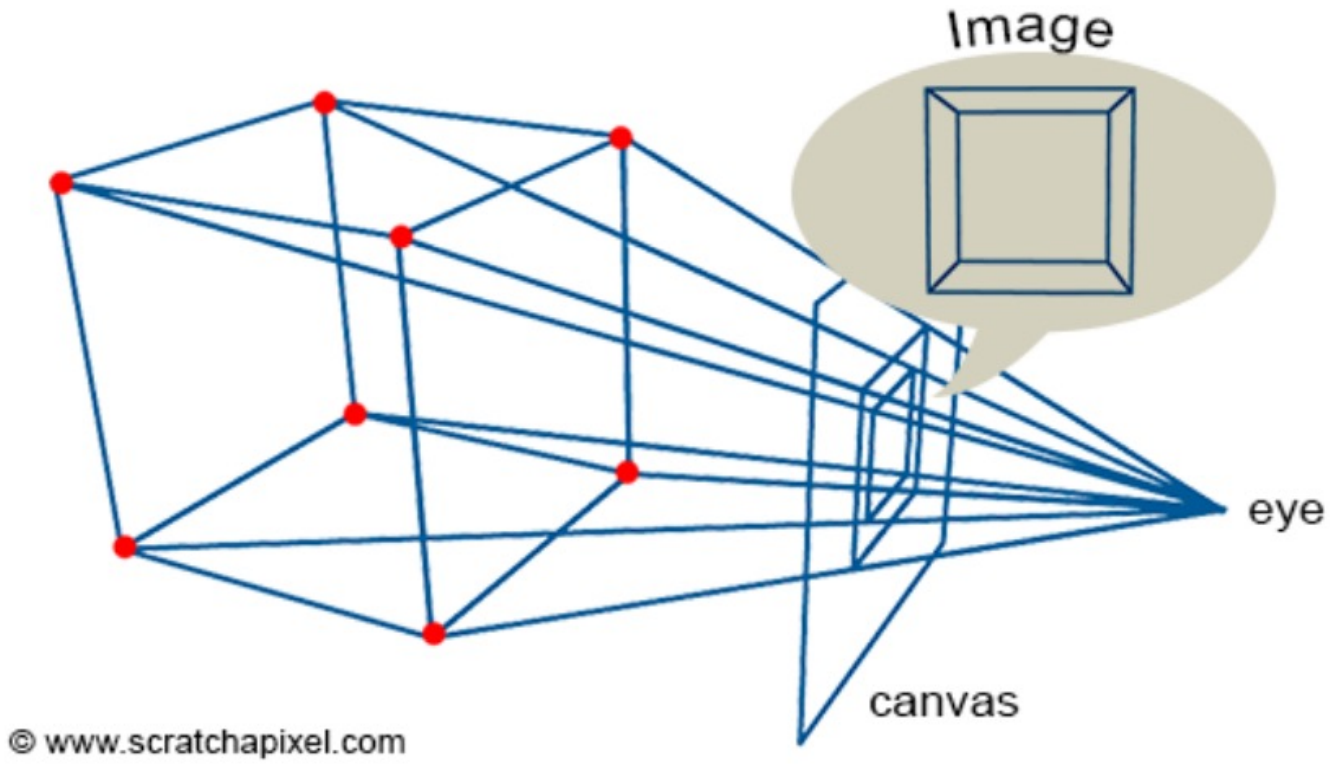
Distances and angles (are / are not **?**) preserved

- **are not**

Degenerate cases:

- Line through focal point projects to a point.
- Plane through focal point projects to line
- Plane perpendicular to image plane projects to part of the image.

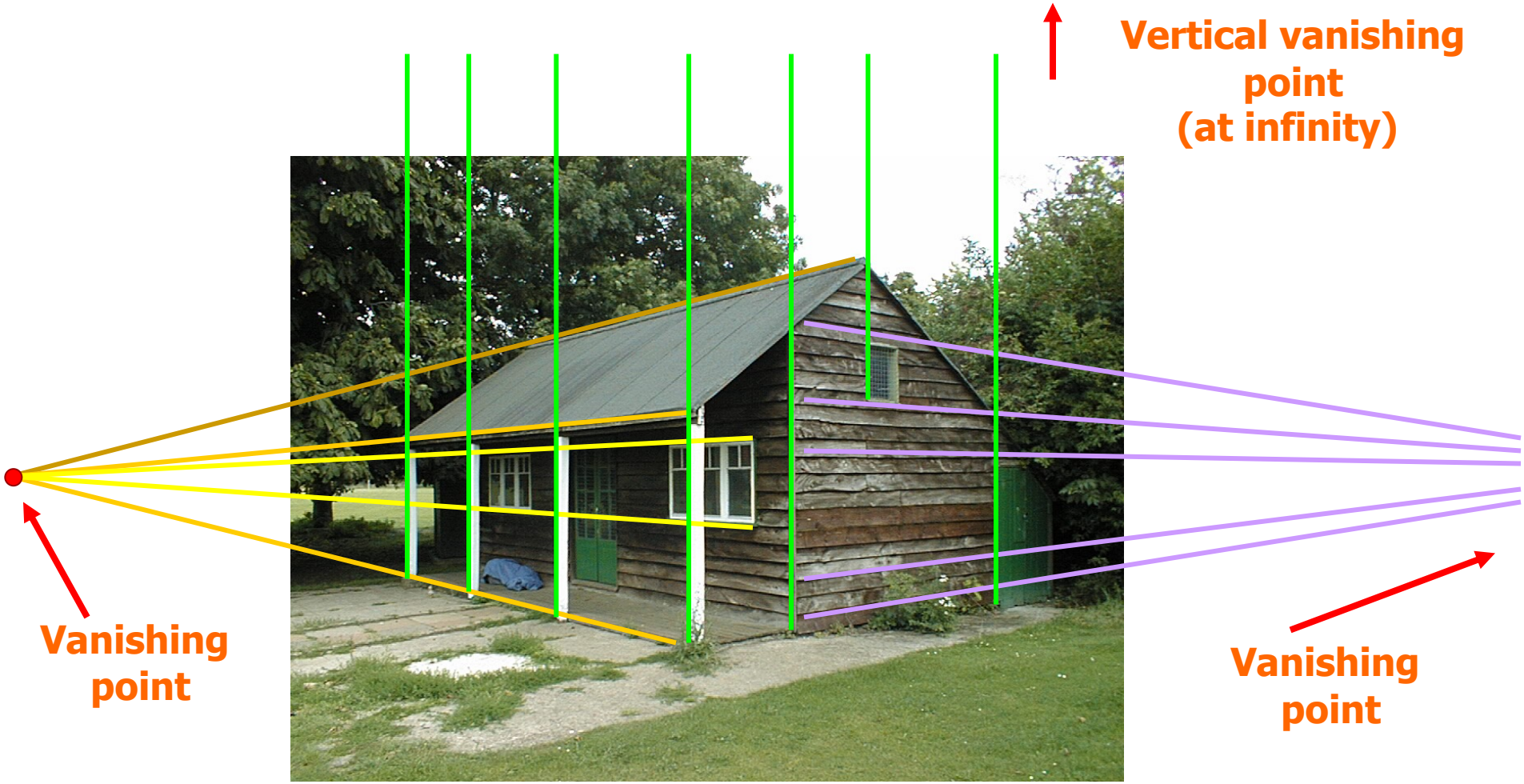
Perspective Projection - Example



Vanishing points and lines

Parallel lines in the world intersect in the image at a “vanishing point”

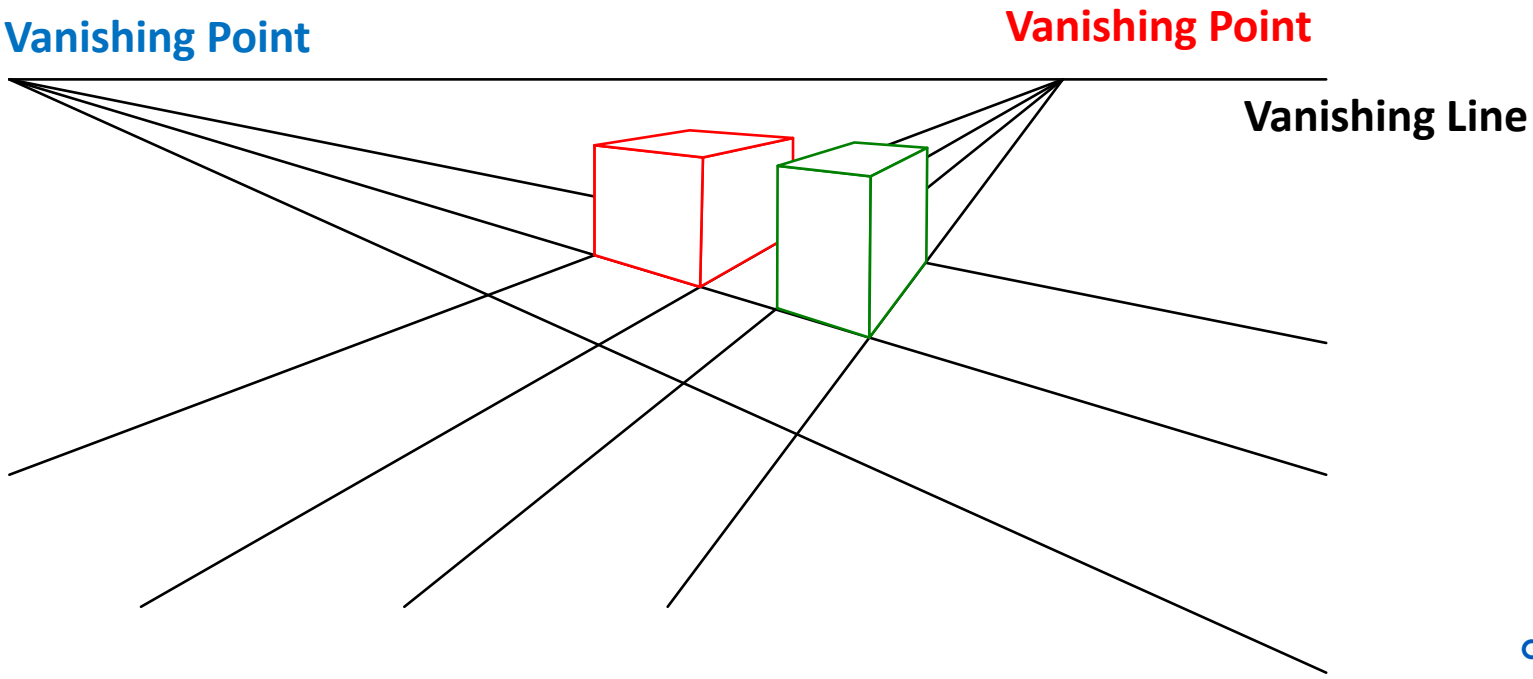
Vanishing points and lines



Slide from Efros, Photo from Criminisi

Vanishing points and lines

- Any two lines, parallel in 3D will meet at a unique vanishing point in image plane.
- All pairs of parallel lines on the same plane in 3D will have vanishing points on a unique vanishing line.

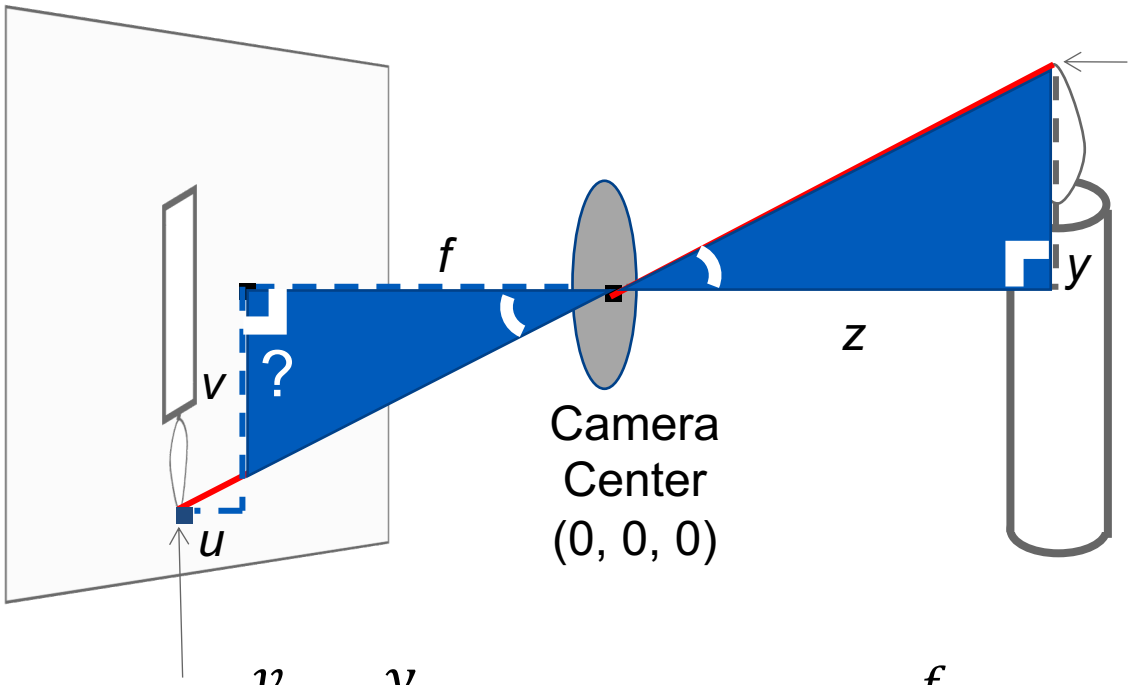


Perspective Projection

Given (x, y, z)
What are u and v?

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

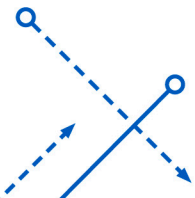
$$x = \begin{bmatrix} u \\ v \end{bmatrix}$$



Sides are Proportional

$$\frac{v}{-f} = \frac{y}{z} \qquad v = -y * \frac{f}{z}$$

$$\frac{u}{-f} = \frac{x}{z} \qquad u = -x * \frac{f}{z}$$



Perspective Projection Equations

- 3D point $\mathbf{P} = (X, Y, Z)^{\mathsf{T}}$ projects to 2D image point $\mathbf{p} = (x, y)^{\mathsf{T}}$.
- By symmetry,

$$\frac{X}{Z} = \frac{x}{f}, \quad \frac{Y}{Z} = \frac{y}{f}$$

i.e.,

$$x = f\frac{X}{Z}, \quad y = f\frac{Y}{Z}$$

- Simplest form of perspective projection.

Preliminary: Homogeneous coordinates

Converting to *homogeneous* coordinates

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

homogeneous image coordinates

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

homogeneous scene coordinates

Converting *from* homogeneous coordinates

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

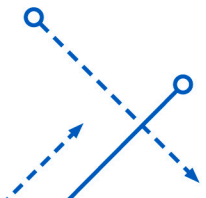
$$\begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} \Rightarrow (x/w, y/w, z/w)$$

- A way of representing N-dimensional coordinates with N+1 numbers.
- It allows perspective projection representing as matrix-vector multiplication.

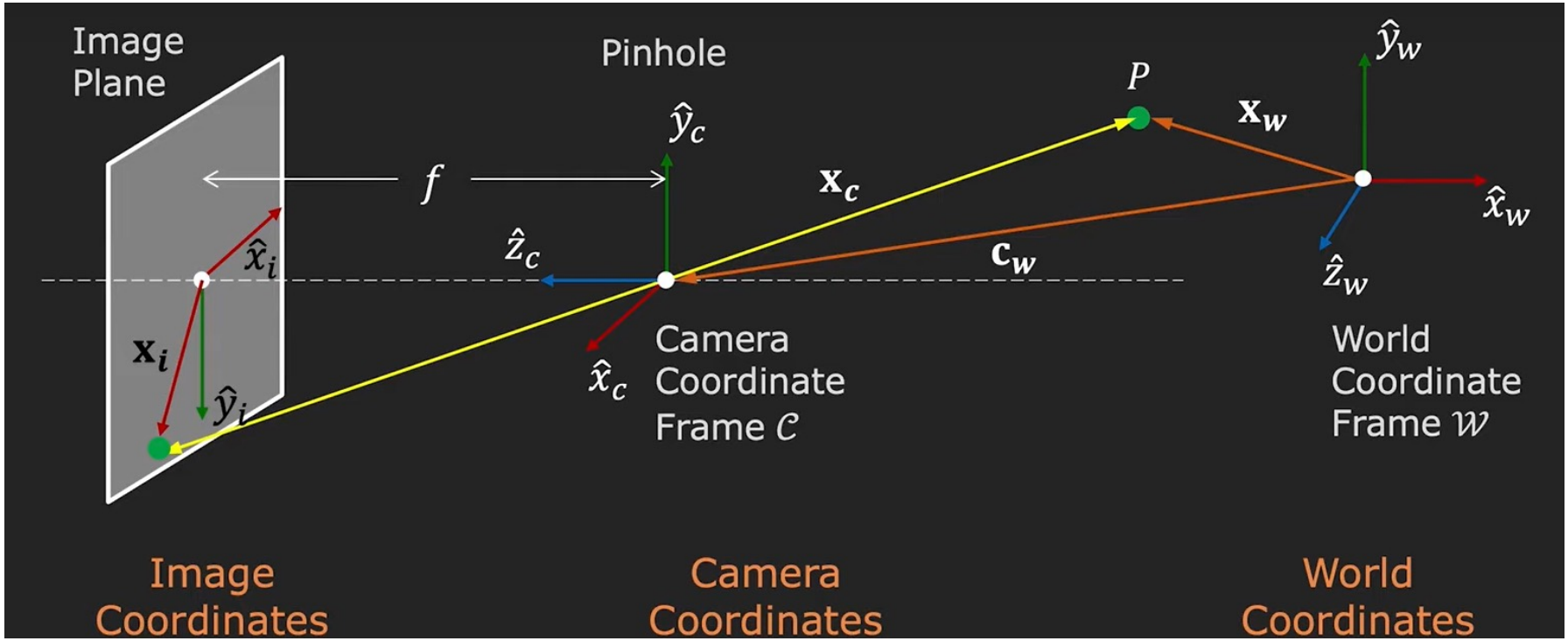
Perspective Projection

- Matrix Form

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



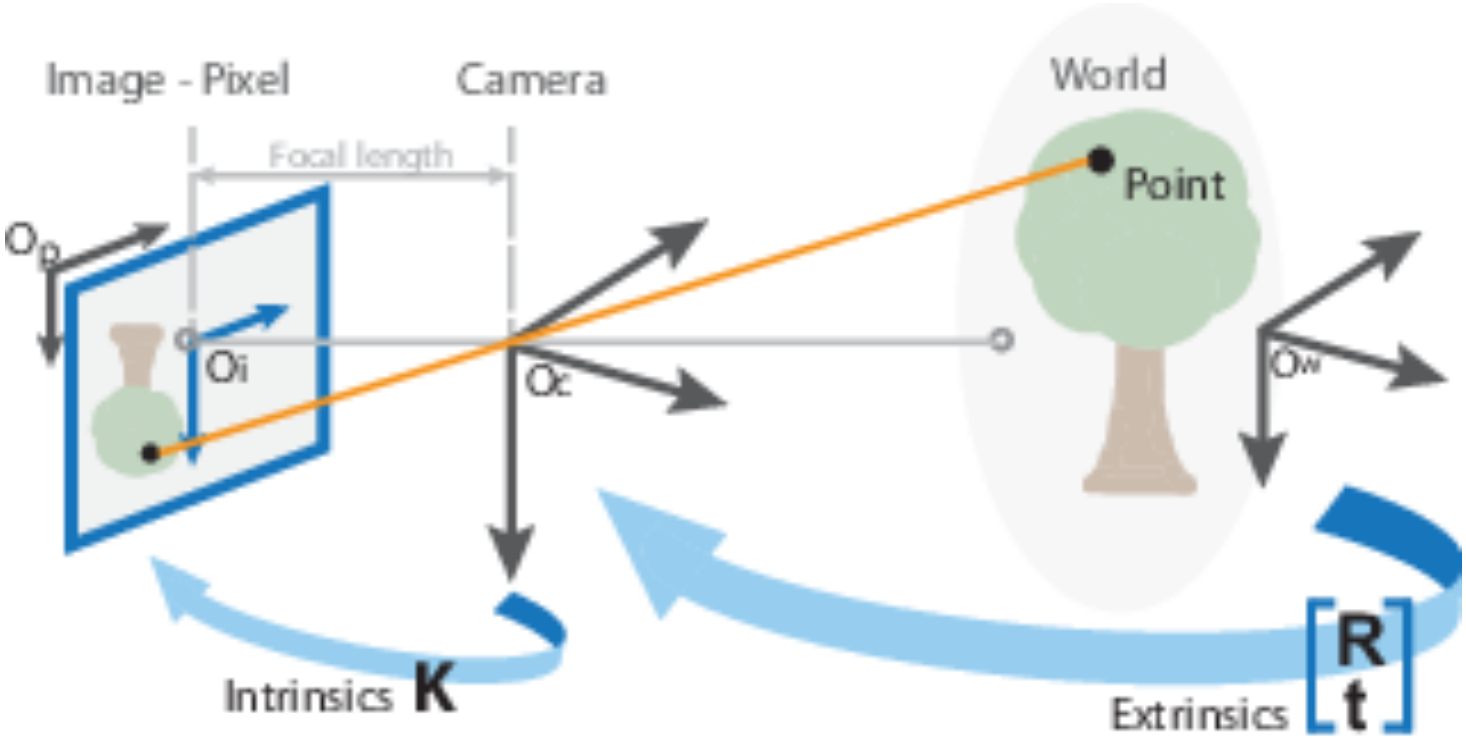
Three different coordinate systems



$\mathbf{X}_i$ is the projection of point P in the world

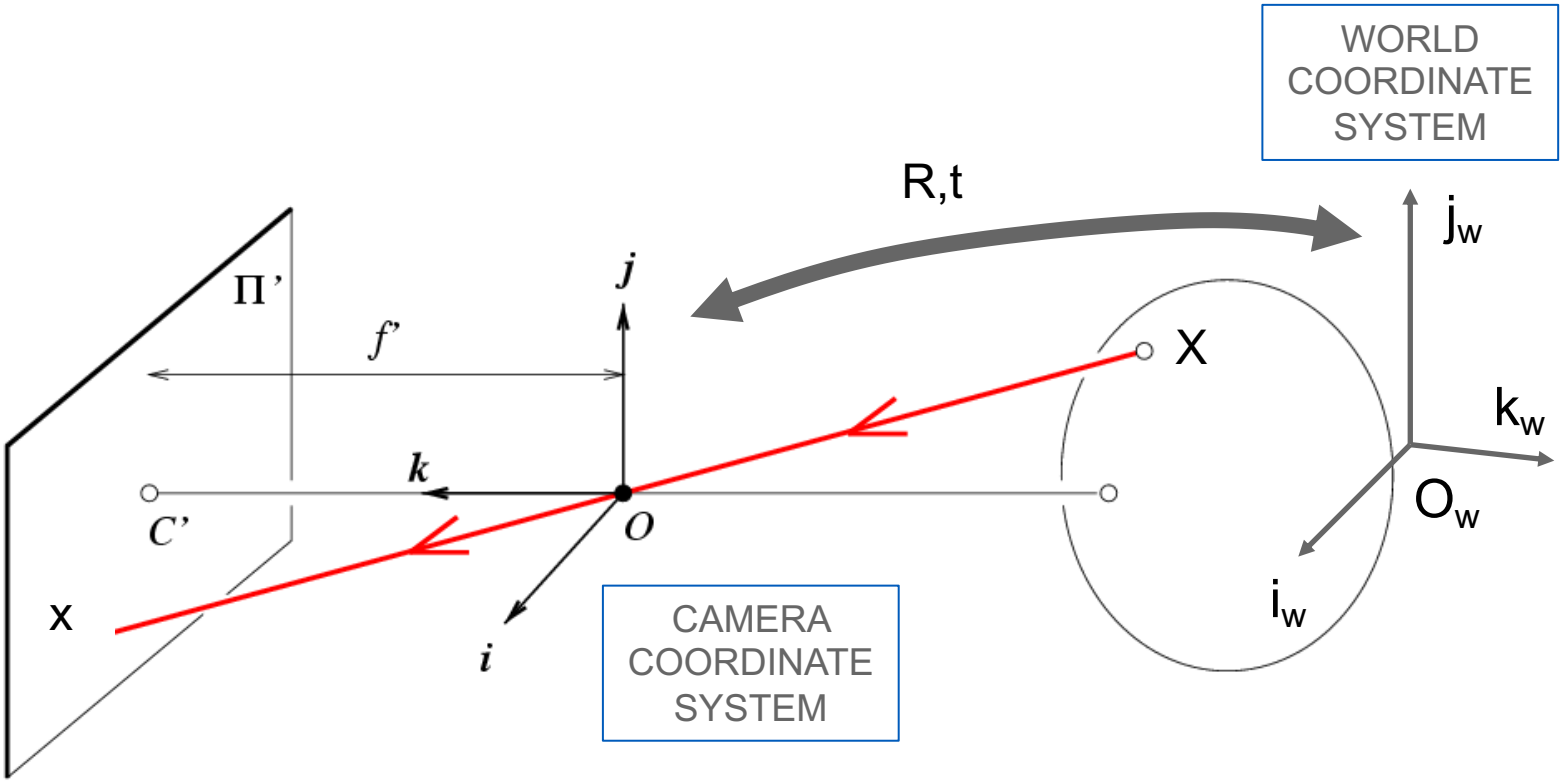
Intrinsic and Extrinsic Parameters

- Perspective equations so far in terms of *camera's* frame.
- Camera's *intrinsic* (Camera) and *extrinsic* (World) parameters needed to calibrate geometry.



K. Grauman

Projection Matrix



$$\mathbf{x} = \mathbf{K}[\mathbf{R} \quad \mathbf{t}]\,\mathbf{X}$$

Extrinsic Matrix

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

Intrinsic and Extrinsic Parameters

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

2D Image Coordinates

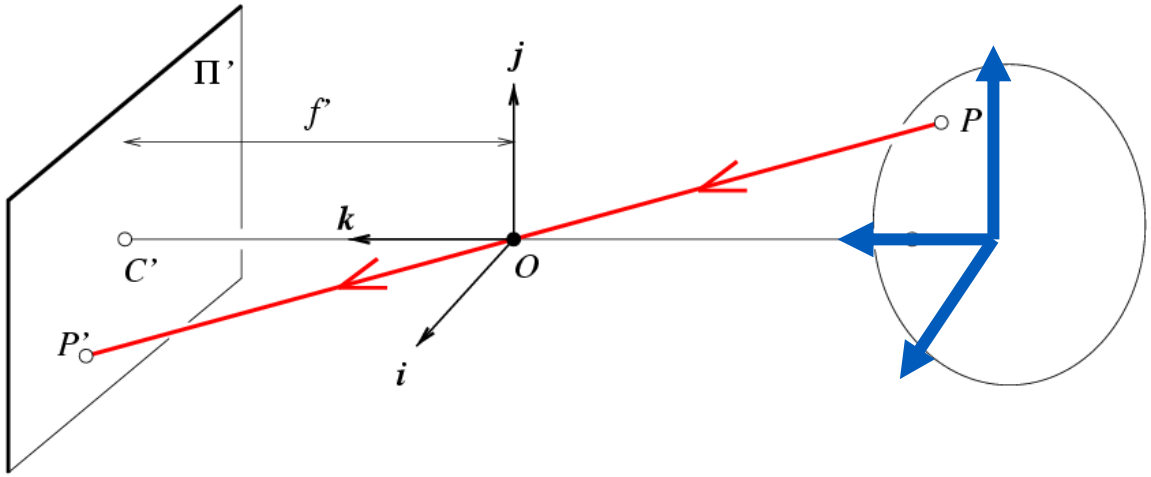
Intrinsic properties (Optical Centre, scaling)

Extrinsic properties (Camera Rotation and translation)

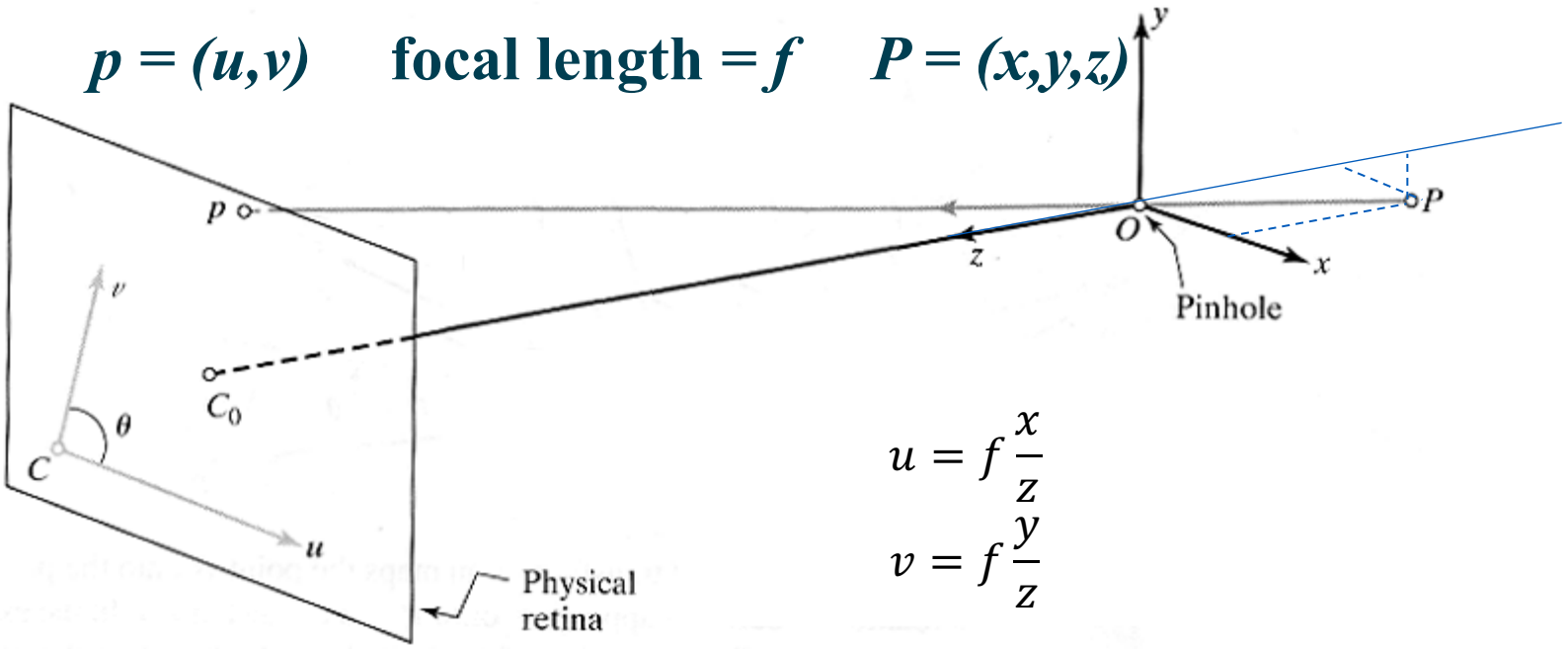
3D World Coordinates

Idealized (Pinhole) Assumptions

- Everything is a unit length
  - Focal Length is 1 unit, pixels are 1 unit
- The camera and the world coordinate systems are aligned with the image plane.
  - No Rotation or Translation



Intrinsic parameters: from World to Pixel



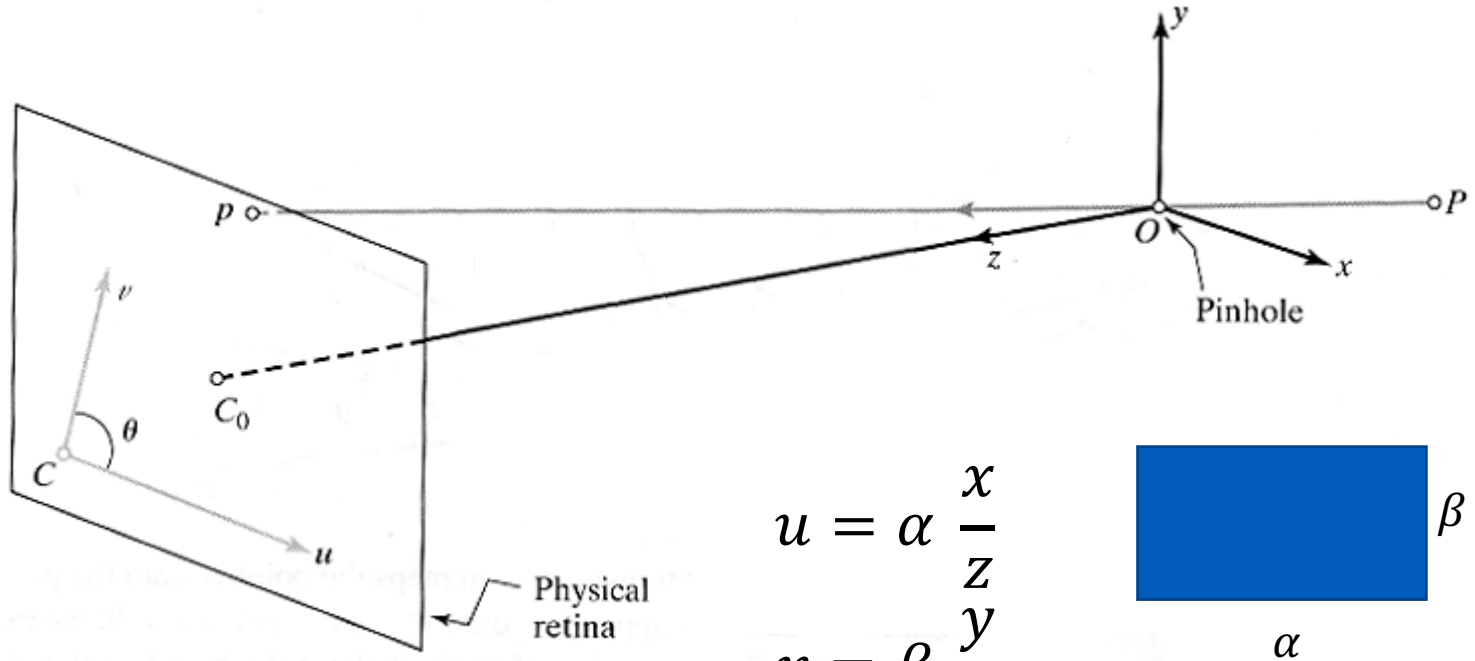
$$u = f\frac{x}{z}$$

$$v = f\frac{y}{z}$$

Perspective projection

Forsyth&Ponce

Intrinsic parameters: Relax pixels aspect

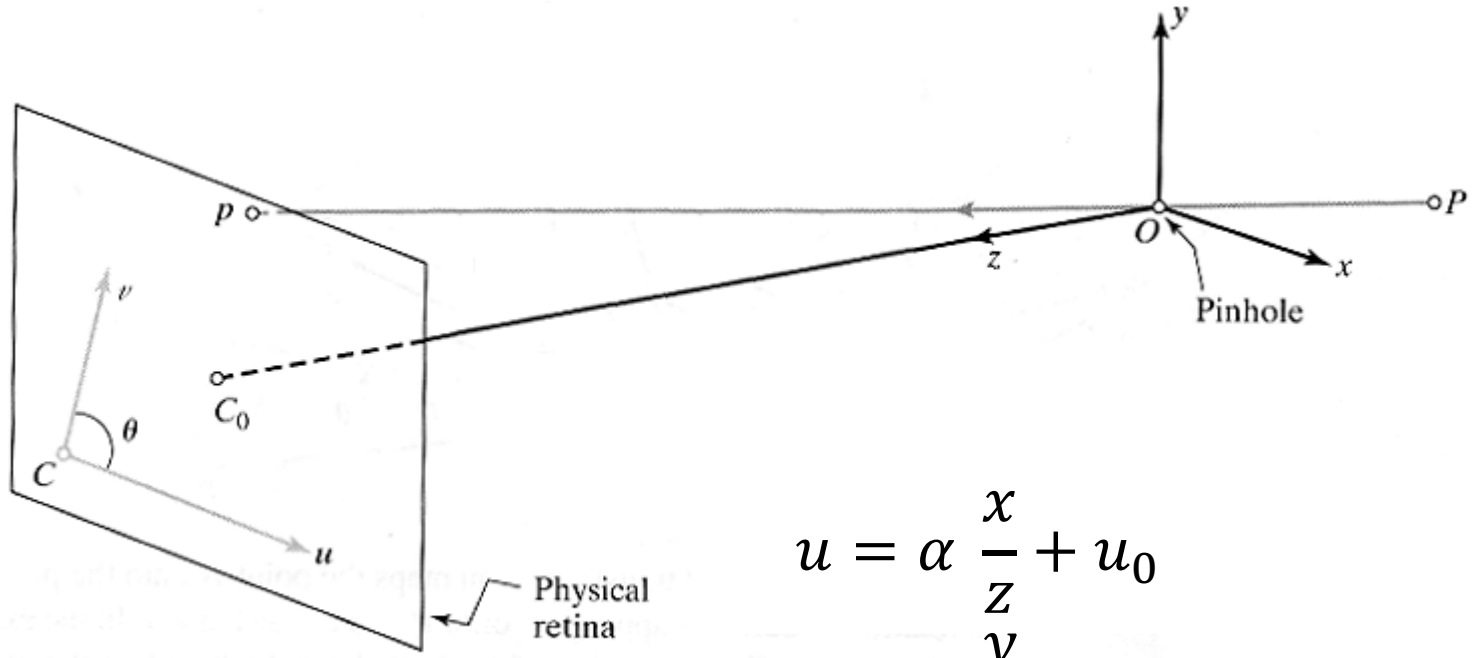


$$u = \alpha \frac{x}{z}$$

$$v = \beta \frac{y}{z}$$

Pixels may not be square

Intrinsic parameters: Relax image center



$$u = \alpha \frac{x}{z} + u_0$$

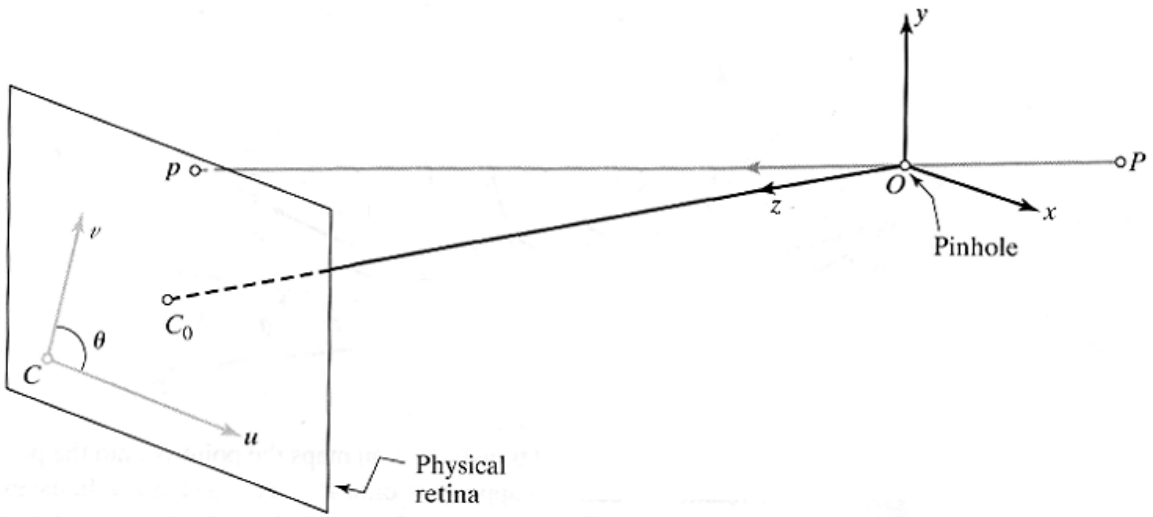
$$v = \beta \frac{y}{z} + v_0$$

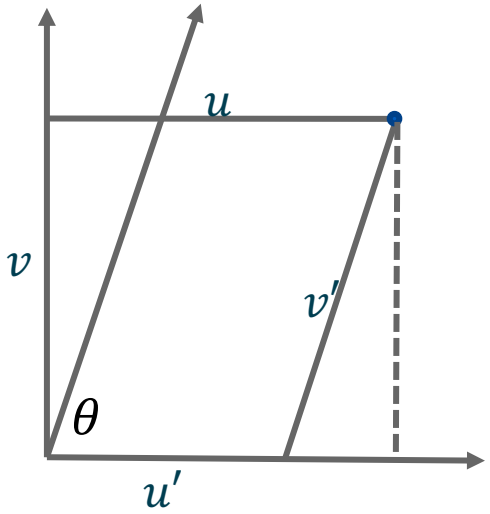
Origin of our camera pixel coordinates is offset

Intrinsic parameters

Relax the image plane is orthogonal to the camera axis:

May be skew between camera pixel axes





$$v' \sin(\theta) = v$$

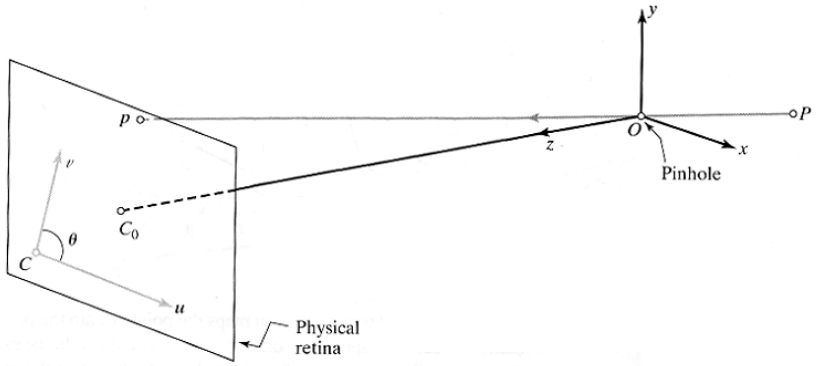
$$u' = u - \cos(\theta) v' = u - \cot(\theta) v$$

$$u = \alpha \frac{x}{z} - \alpha \cot(\theta) \frac{y}{z} + u_0$$

$$v = \frac{\beta}{\sin(\theta)} \frac{y}{z} + v_0$$

Intrinsic parameters: Matrix form

- Using homogenous coordinates



$$u = \alpha \frac{x}{z} - \alpha \cot(\theta) \frac{y}{z} + u_0$$

$$v = \frac{\beta}{\sin(\theta)} \frac{y}{z} + v_0$$

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} \alpha & -\alpha\cot(\theta) & u_0 & 0 \\ 0 & \frac{\beta}{\sin(\theta)} & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

Projection matrix

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

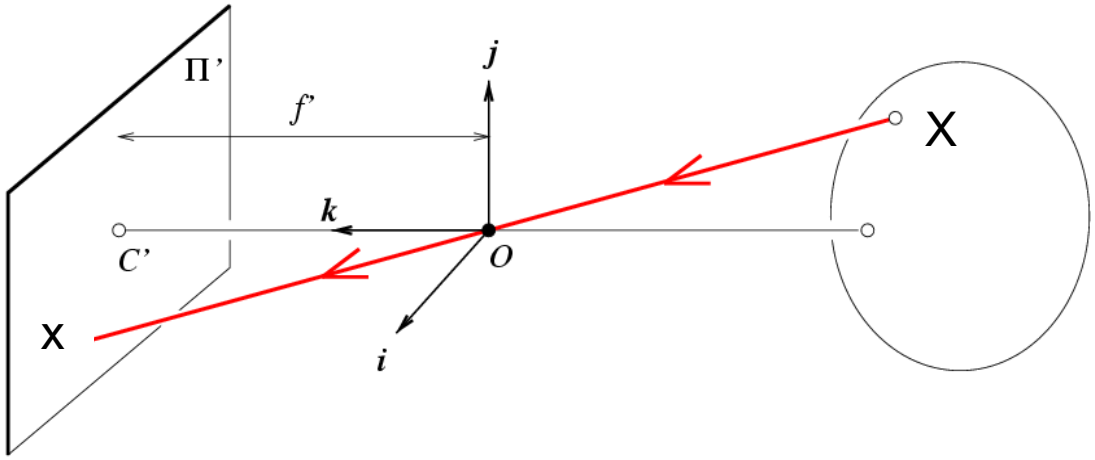
x: Image Coordinates: (u,v,1)
K: Intrinsic Matrix (3x3)
R: Rotation (3x3)
t: Translation (3x1)
X: World Coordinates: (X,Y,Z,1)

Extrinsic Assumptions

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

Intrinsic Assumptions

- Optical center at (0,0)
- Unit aspect ratio
- No skew



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

K

Slide Credit: Savarese

Remove assumption: known optical center

$$\mathbf{x} = \mathbf{K}[\mathbf{R} \quad \mathbf{t}]\,\mathbf{X}$$

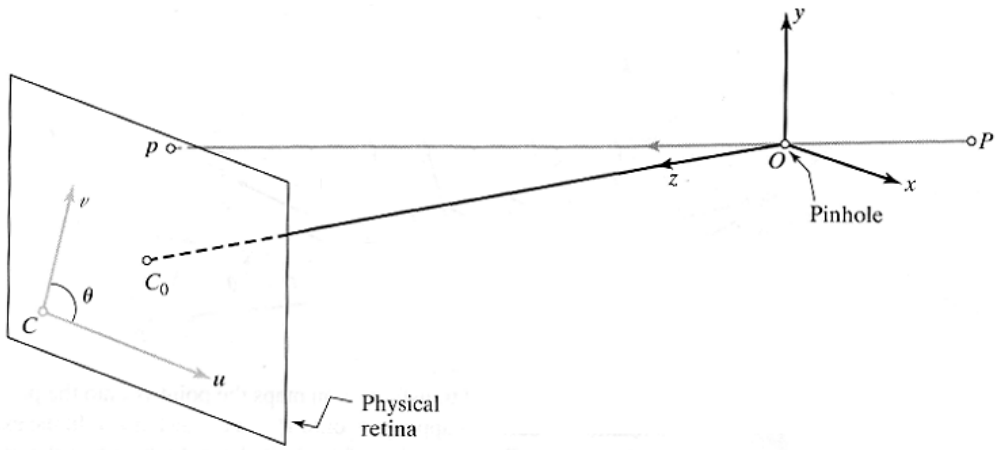
x: Image Coordinates: (u,v,1)
K: Intrinsic Matrix (3x3)
R: Rotation (3x3)
t: Translation (3x1)
X: World Coordinates: (X,Y,Z,1)

Extrinsic Assumptions

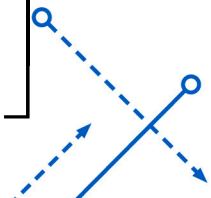
- No rotation
- Camera at (0,0,0)

Intrinsic Assumptions

- ~~Optical center at (0,0)~~
- Unit aspect ratio
- No skew



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



Remove assumption: square pixels

$$\mathbf{x} = \mathbf{K}[\mathbf{R} \quad \mathbf{t}]\mathbf{X}$$

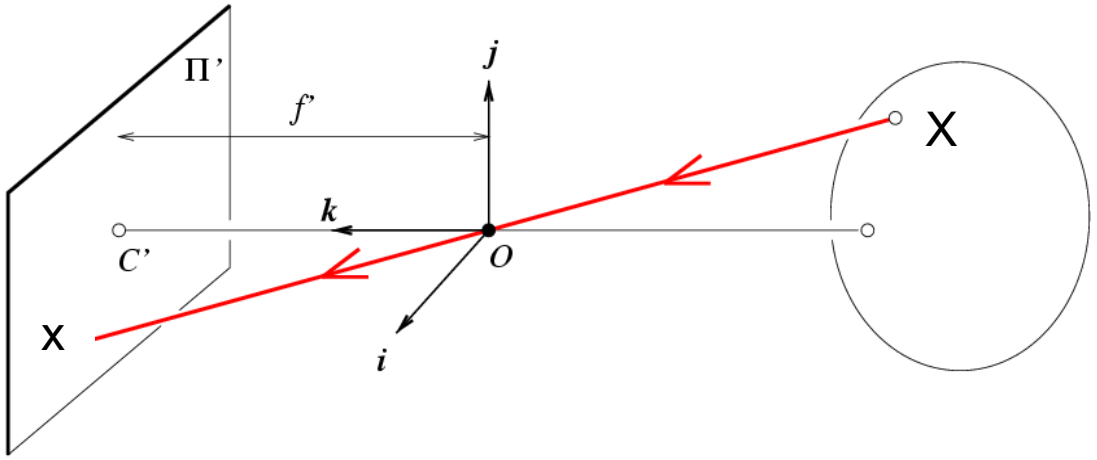
x: Image Coordinates: (u,v,1)
K: Intrinsic Matrix (3x3)
R: Rotation (3x3)
t: Translation (3x1)
X: World Coordinates: (X,Y,Z,1)

Extrinsic Assumptions

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

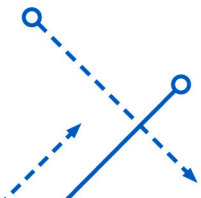
Intrinsic Assumptions

- ~~Optical center at (0,0)~~
- ~~Unit aspect ratio~~
- No skew



$$\mathbf{x} = \mathbf{K}[\mathbf{I} \quad \mathbf{0}]\,\mathbf{X}$$

$$w\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha & 0 & u_0 & 0 \\ 0 & \beta & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$



Remove assumption: non-skewed pixels

$$\mathbf{x} = \mathbf{K}[\mathbf{R} \quad \mathbf{t}]\,\mathbf{X}$$

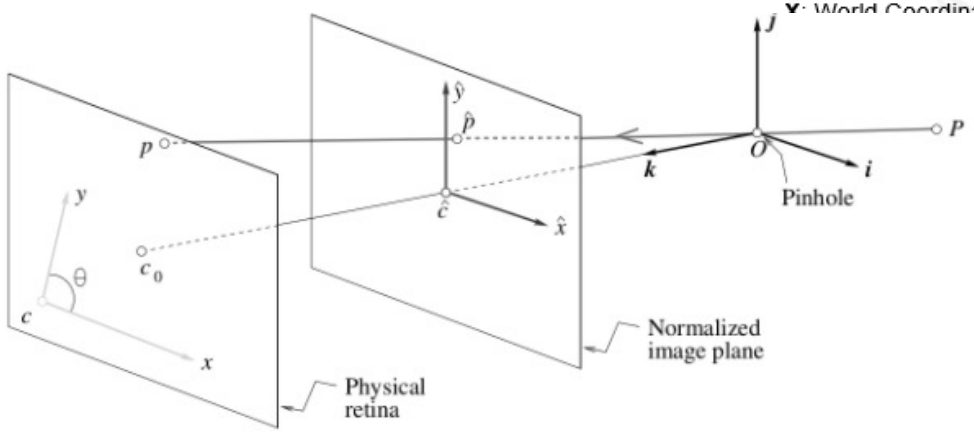
x: Image Coordinates: (u,v,1)
K: Intrinsic Matrix (3x3)
R: Rotation (3x3)
t: Translation (3x1)
X: World Coordinates: (X,Y,Z,1)

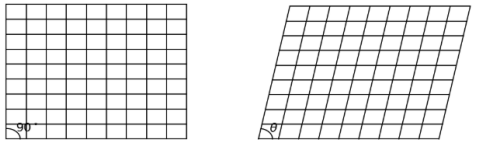
Extrinsic Assumptions

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

Intrinsic Assumptions

- ~~Optical center at (0,0)~~
- ~~Unit aspect ratio~~
- ~~No skew~~





$$\mathbf{x} = \mathbf{K}[\mathbf{I} \quad \mathbf{0}]\,\mathbf{X} \quad \Rightarrow \quad w\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha & s & u_0 \\ 0 & \beta & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Note: different books use different notation for parameters

SAIR

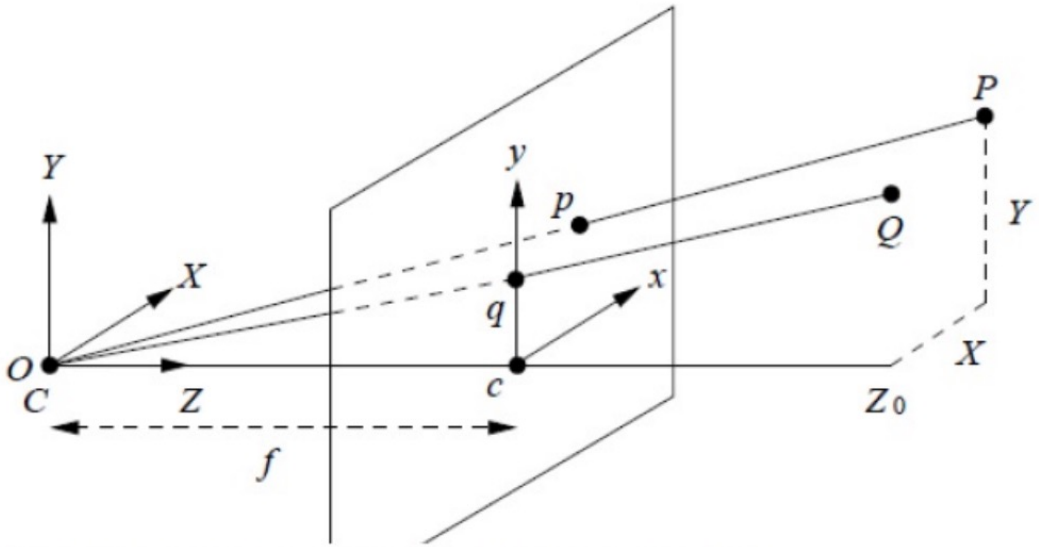
Spatial AI & Robotics Lab

IMAGE FORMATION I

Other types of projection

Weak Perspective Projection

- When you zoom your camera very far, the depth changes in the scene are negligible when compared to the distance from camera



$$u = f\frac{x}{z}$$
$$v = f\frac{y}{z}$$

- Scene depth << distance to camera.
- Z is the same for all scene points, say Z_0

$$x = sX, \quad y = sY, \quad s = \frac{f}{Z_0} \text{ for all scene points.}$$

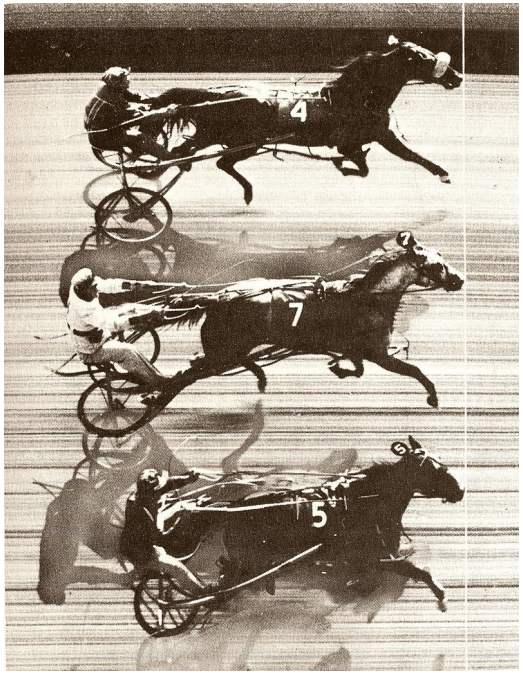
Weak Perspective Projection - Example

Qingming Festival by the Riverside Zhang Zeduan ~900 AD

As a comparison

Photo Finish Photography (strip photography)

- A photo-finish image is made by a camera with an extremely thin, vertical slit.
- A photo-finish camera records an object moving past its exposure slit over time. 1D array



1960 Olympic tryouts at Palo Alto, CA

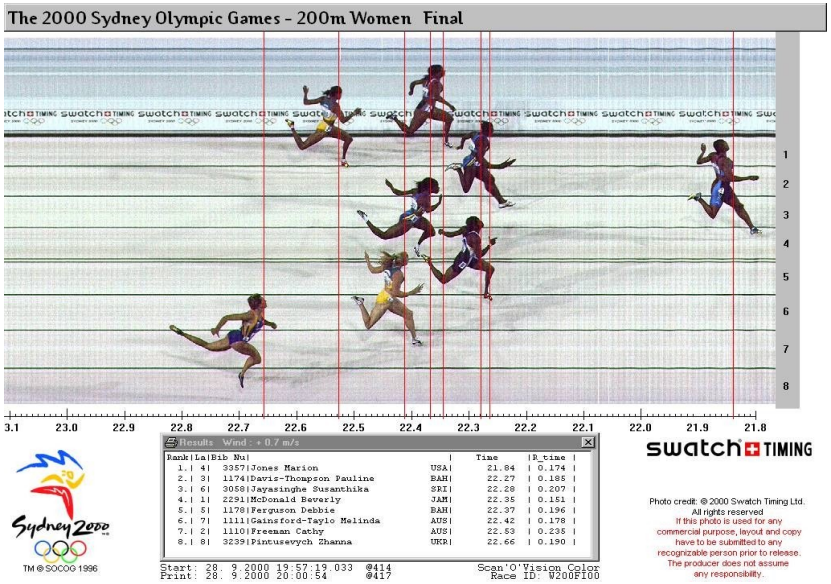
Photo Finish Example

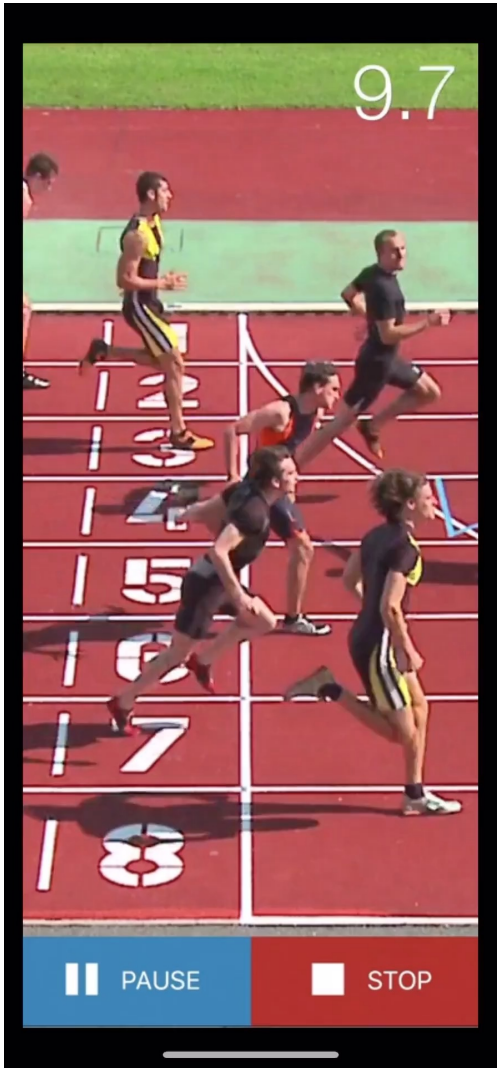
SprintTimer - Photo Finish 4+

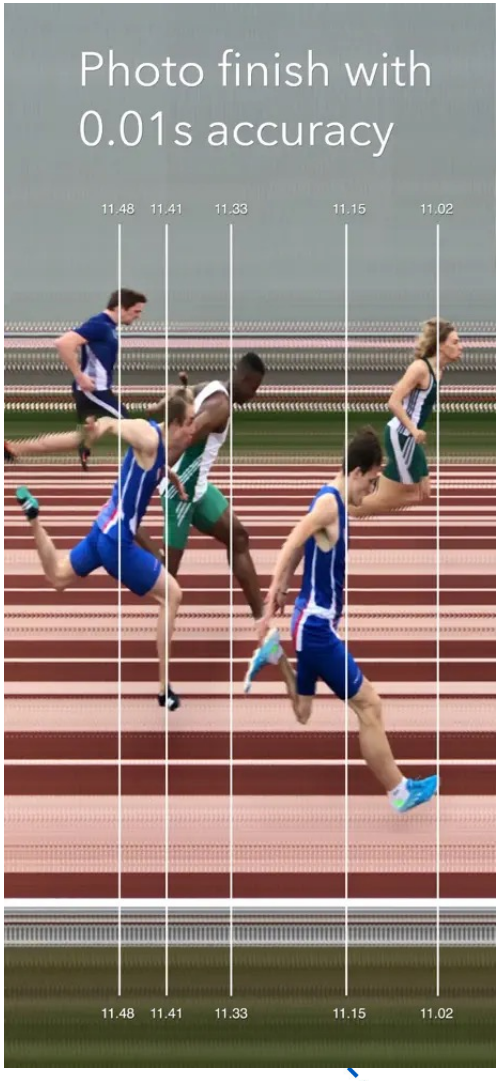
Professional Sports Timer

Sten Kaiser

2,490.00₸ · Offers In-App Purchases

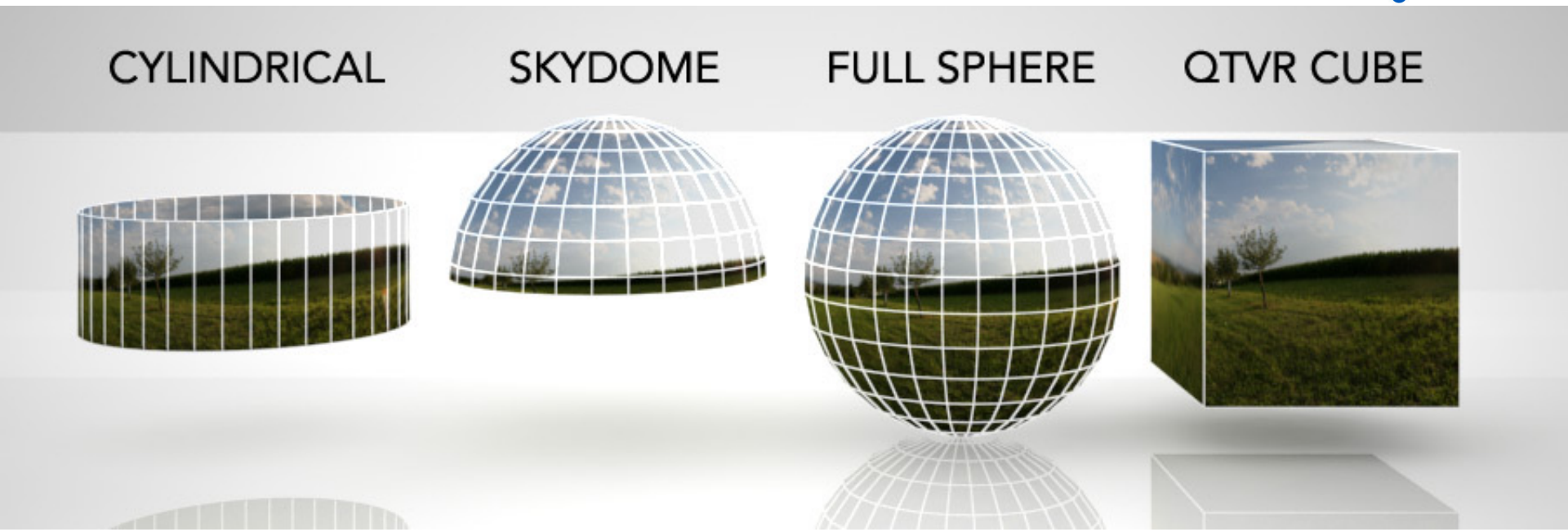






360 Imaging

360 Camera and Projection Mode

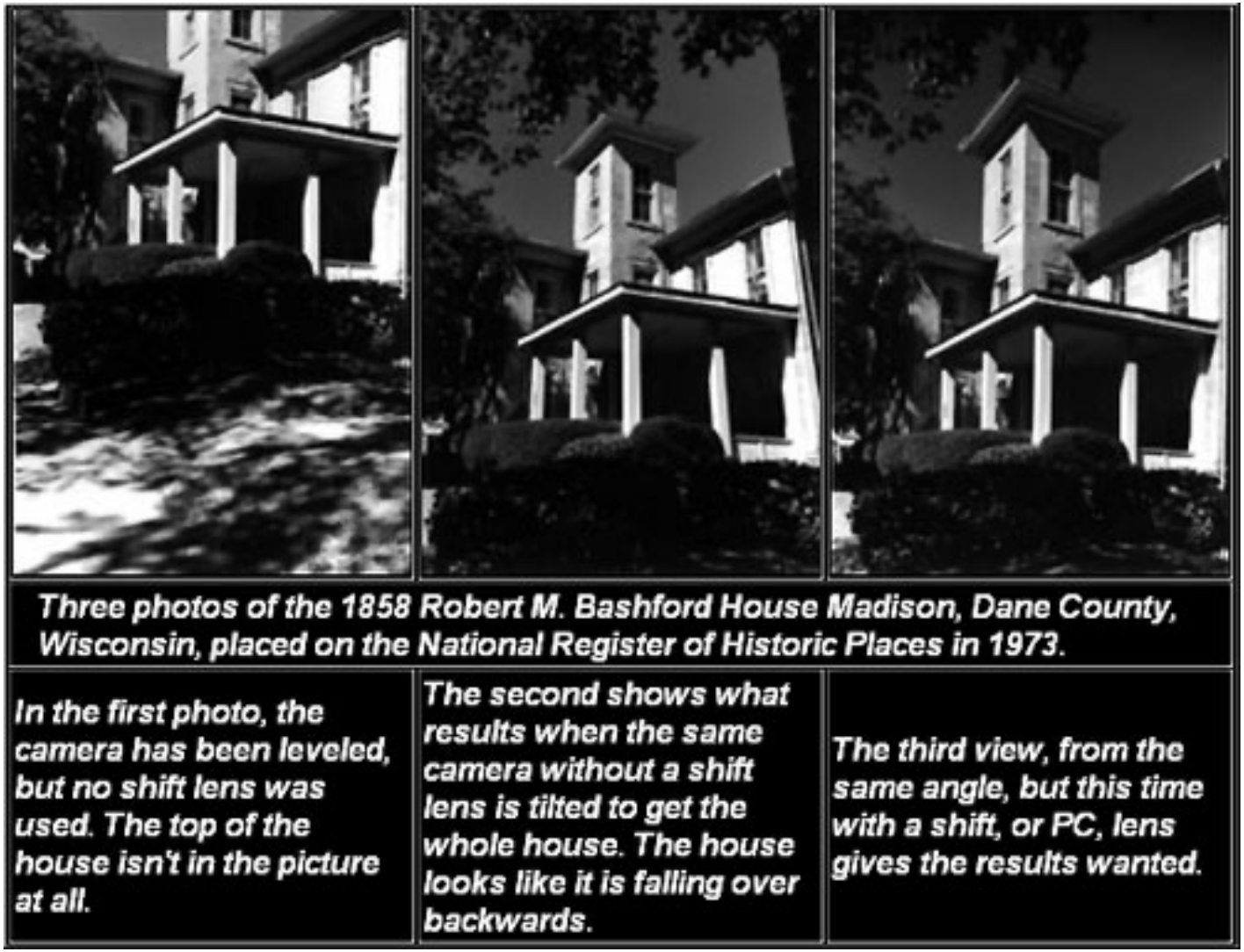


Tilt-shift Imaging

Note how the focus plane is along the train, and how the blurring of the background proceeds from left to right.

Often used to simulate a miniature scene

Tilt-shift perspective correction



Three photos of the 1858 Robert M. Bashford House Madison, Dane County, Wisconsin, placed on the National Register of Historic Places in 1973.

In the first photo, the camera has been leveled, but no shift lens was used. The top of the house isn't in the picture at all.

The second shows what results when the same camera without a shift lens is tilted to get the whole house. The house looks like it is falling over backwards.

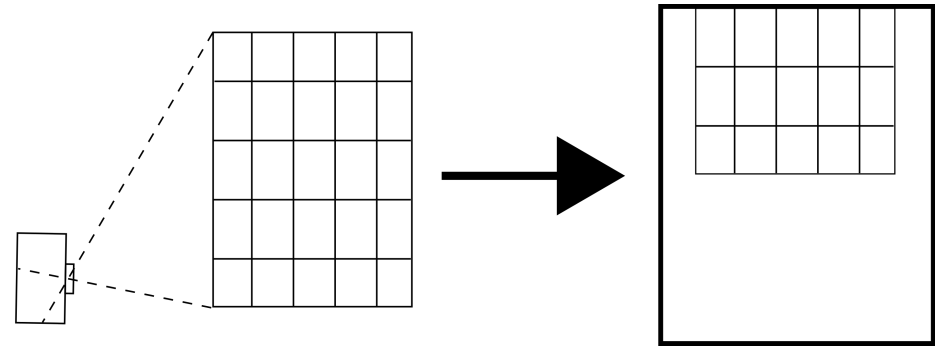
The third view, from the same angle, but this time with a shift, or PC, lens gives the results wanted.

http://en.wikipedia.org/wiki/Tilt-shift_photography

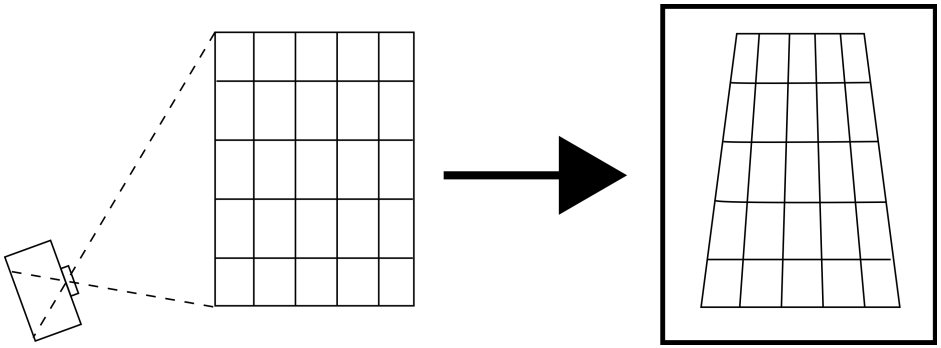
Tilt-shift Camera

http://en.wikipedia.org/wiki/Tilt-shift_photography

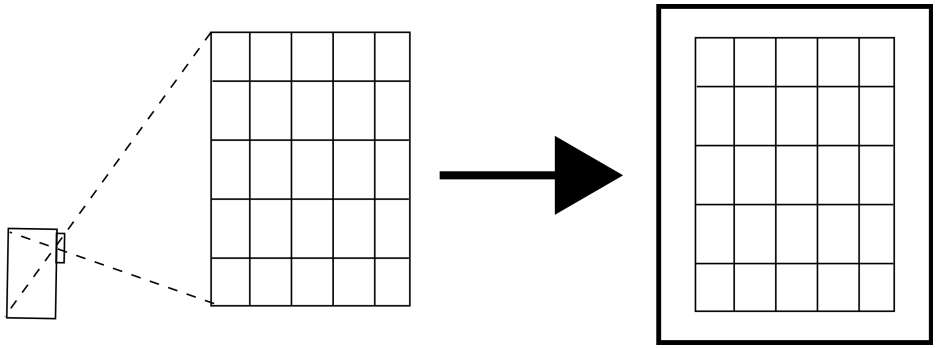
Tilt-shift Correction



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



Tilting the camera upwards results in perspective distortion.

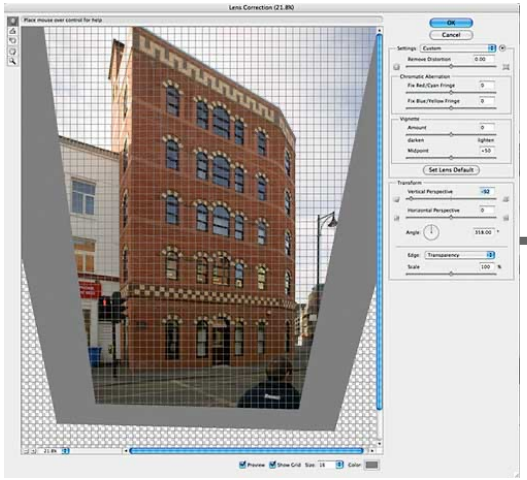


Shifting the lens upwards results in a picture of the entire subject without perspective distortion.

Tilt-shift Software Correction

Normal Camera

Tilt-shift Camera



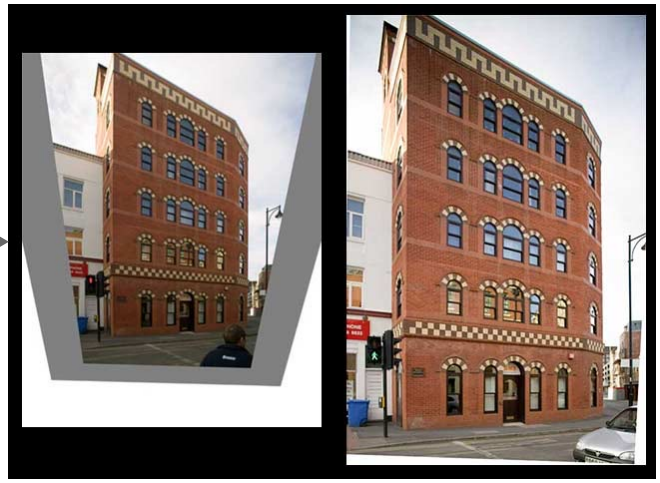


Image transformation is in the next lecture.

SAIR

Spatial AI & Robotics Lab

IMAGE FORMATION

Camera Calibration

Camera Calibration (Simplest form)

- Intrinsic + Extrinsic combined

Camera to Pixel

$$\begin{bmatrix}\tilde{u}\\ \tilde{v}\\ \tilde{w}\end{bmatrix} = \begin{bmatrix}f_x & 0 & o_x & 0\\ 0 & f_y & o_y & 0\\ 0 & 0 & 1 & 0\end{bmatrix}\begin{bmatrix}x_c\\ y_c\\ z_c\\ 1\end{bmatrix}$$

World to Camera

$$\begin{bmatrix}x_c\\ y_c\\ z_c\\ 1\end{bmatrix} = \begin{bmatrix}r_{11} & r_{12} & r_{13} & t_x\\ r_{21} & r_{22} & r_{23} & t_y\\ r_{31} & r_{32} & r_{33} & t_z\\ 0 & 0 & 0 & 1\end{bmatrix}\begin{bmatrix}x_w\\ y_w\\ z_w\\ 1\end{bmatrix}$$

$$\tilde{\mathbf{u}} = M_{int}\,\tilde{\mathbf{x}}_c \qquad \tilde{\mathbf{x}}_c = M_{ext}\tilde{\mathbf{x}}_w$$

Combining the above two equations, we get the full projection matrix P:

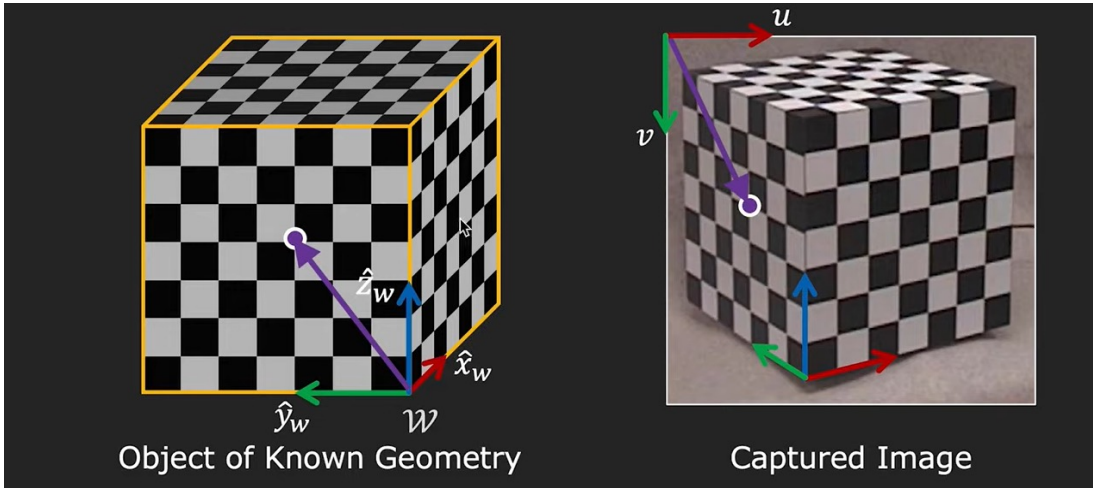
$$\tilde{\mathbf{u}} = M_{int}M_{ext}\,\tilde{\mathbf{x}}_w = P\,\tilde{\mathbf{x}}_w$$

$$\begin{bmatrix}\tilde{u}\\ \tilde{v}\\ \tilde{w}\end{bmatrix} = \begin{bmatrix}p_{11} & p_{12} & p_{13} & p_{14}\\ p_{21} & p_{22} & p_{23} & p_{24}\\ p_{31} & p_{32} & p_{33} & p_{34}\end{bmatrix}\begin{bmatrix}x_w\\ y_w\\ z_w\\ 1\end{bmatrix}$$

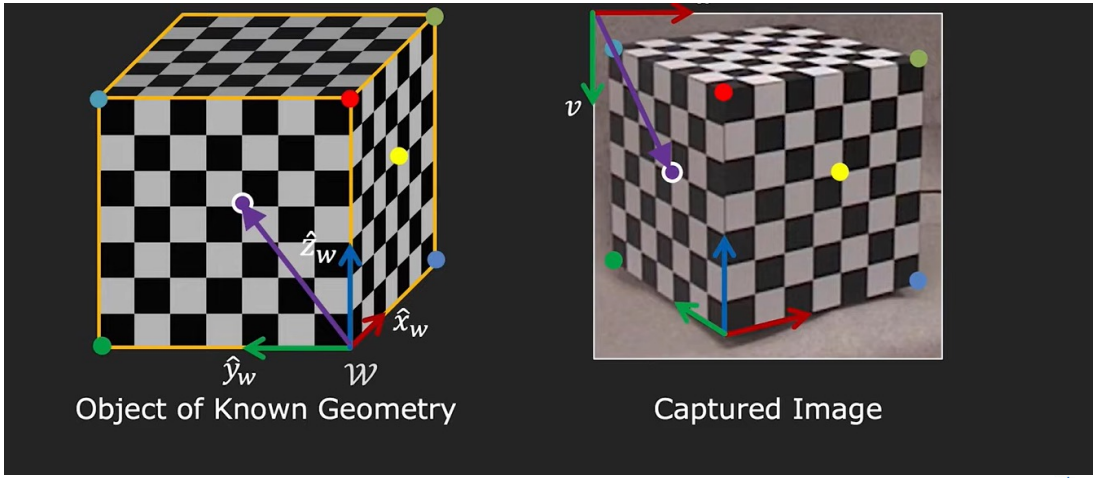
Point Correspondence

- Get **correspondence** of **world points** and **image points**

$$\mathbf{u} = \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} 56 \\ 115 \end{bmatrix}$$
(pixels)



$$\mathbf{x}_w = \begin{bmatrix} x_w \\ y_w \\ z_w \end{bmatrix} = \begin{bmatrix} 0 \\ 3 \\ 4 \end{bmatrix}$$
(inches)



Expanding Projection Matrix

$$w\begin{bmatrix} u^{(i)} \\ v^{(i)} \\ 1 \end{bmatrix} \equiv \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix} \begin{bmatrix} x_w^{(i)} \\ y_w^{(i)} \\ z_w^{(i)} \\ 1 \end{bmatrix}$$

Known — Unknown — Known

Expanding the matrix as linear equations:

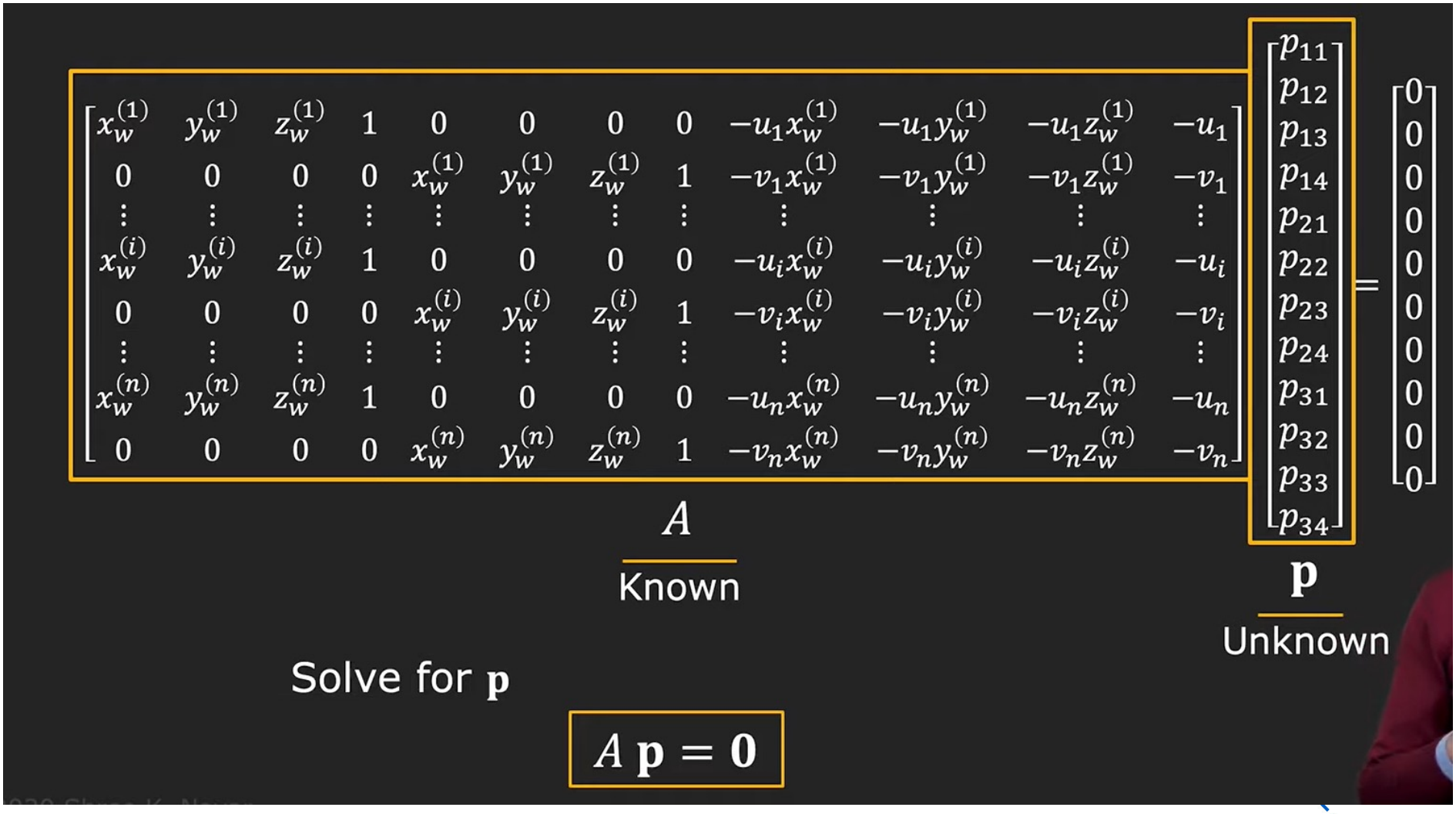
$$u^{(i)} = \frac{p_{11}x_w^{(i)} + p_{12}y_w^{(i)} + p_{13}z_w^{(i)} + p_{14}}{p_{31}x_w^{(i)} + p_{32}y_w^{(i)} + p_{33}z_w^{(i)} + p_{34}}$$

$$v^{(i)} = \frac{p_{21}x_w^{(i)} + p_{22}y_w^{(i)} + p_{23}z_w^{(i)} + p_{24}}{p_{31}x_w^{(i)} + p_{32}y_w^{(i)} + p_{33}z_w^{(i)} + p_{34}}$$

Expanded Projection matrix

$$\begin{bmatrix} x_w^{(1)} & y_w^{(1)} & z_w^{(1)} & 1 & 0 & 0 & 0 & 0 & -u_1 x_w^{(1)} & -u_1 y_w^{(1)} & -u_1 z_w^{(1)} & -u_1 \\ 0 & 0 & 0 & 0 & x_w^{(1)} & y_w^{(1)} & z_w^{(1)} & 1 & -v_1 x_w^{(1)} & -v_1 y_w^{(1)} & -v_1 z_w^{(1)} & -v_1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_w^{(i)} & y_w^{(i)} & z_w^{(i)} & 1 & 0 & 0 & 0 & 0 & -u_i x_w^{(i)} & -u_i y_w^{(i)} & -u_i z_w^{(i)} & -u_i \\ 0 & 0 & 0 & 0 & x_w^{(i)} & y_w^{(i)} & z_w^{(i)} & 1 & -v_i x_w^{(i)} & -v_i y_w^{(i)} & -v_i z_w^{(i)} & -v_i \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_w^{(n)} & y_w^{(n)} & z_w^{(n)} & 1 & 0 & 0 & 0 & 0 & -u_n x_w^{(n)} & -u_n y_w^{(n)} & -u_n z_w^{(n)} & -u_n \\ 0 & 0 & 0 & 0 & x_w^{(n)} & y_w^{(n)} & z_w^{(n)} & 1 & -v_n x_w^{(n)} & -v_n y_w^{(n)} & -v_n z_w^{(n)} & -v_n \end{bmatrix} \begin{bmatrix} p_{11} \\ p_{12} \\ p_{13} \\ p_{14} \\ p_{21} \\ p_{22} \\ p_{23} \\ p_{24} \\ p_{31} \\ p_{32} \\ p_{33} \\ p_{34} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Solution space is scale-invariant



Objective/Loss Function

We want $A\mathbf{p}$ as close to 0 as possible and $\|\mathbf{p}\|^2 = 1$:

$$\min_{\mathbf{p}} \|A\mathbf{p}\|^2 \text{ such that } \|\mathbf{p}\|^2 = 1$$

$$\min_{\mathbf{p}} (\mathbf{p}^T A^T A \mathbf{p}) \text{ such that } \mathbf{p}^T \mathbf{p} = 1$$

Loss function $L(\mathbf{p}, \lambda)$:

$$L(\mathbf{p}, \lambda) = \mathbf{p}^T A^T A \mathbf{p} - \lambda(\mathbf{p}^T \mathbf{p} - 1)$$

Solve Objective Function

Taking derivatives of $L(\mathbf{p}, \lambda)$ w.r.t $\mathbf{p}$: $2A^TA\mathbf{p} - 2\lambda\mathbf{p} = \mathbf{0}$

$$\boxed{A^TA\mathbf{p} = \lambda\mathbf{p}}$$ Eigenvalue Problem

Eigenvector $\mathbf{p}$ with smallest eigenvalue λ of matrix A^TA minimizes the loss function $L(\mathbf{p})$.

- An explanation

Estimation K, R

$$P = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix} = \begin{bmatrix} f_x & 0 & o_x & 0 \\ 0 & f_y & o_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

M_{int} M_{ext}

$$\begin{bmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ p_{31} & p_{32} & p_{33} \end{bmatrix} = \begin{bmatrix} f_x & 0 & o_x \\ 0 & f_y & o_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} = KR$$

Given that K is an Upper Right Triangular matrix and R is an Orthonormal matrix, it is possible to uniquely "decouple" K and R from their product using "QR factorization".

Wiki: QR Factorization

Estimating Translation

$$P = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix} = \begin{bmatrix} f_x & 0 & o_x & 0 \\ 0 & f_y & o_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

M_{int} M_{ext}

That is:

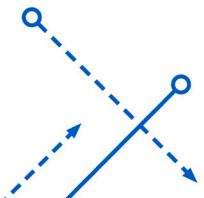
$$\begin{bmatrix} p_{14} \\ p_{24} \\ p_{34} \end{bmatrix} = \begin{bmatrix} f_x & 0 & o_x \\ 0 & f_y & o_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} = K\mathbf{t}$$

Therefore:

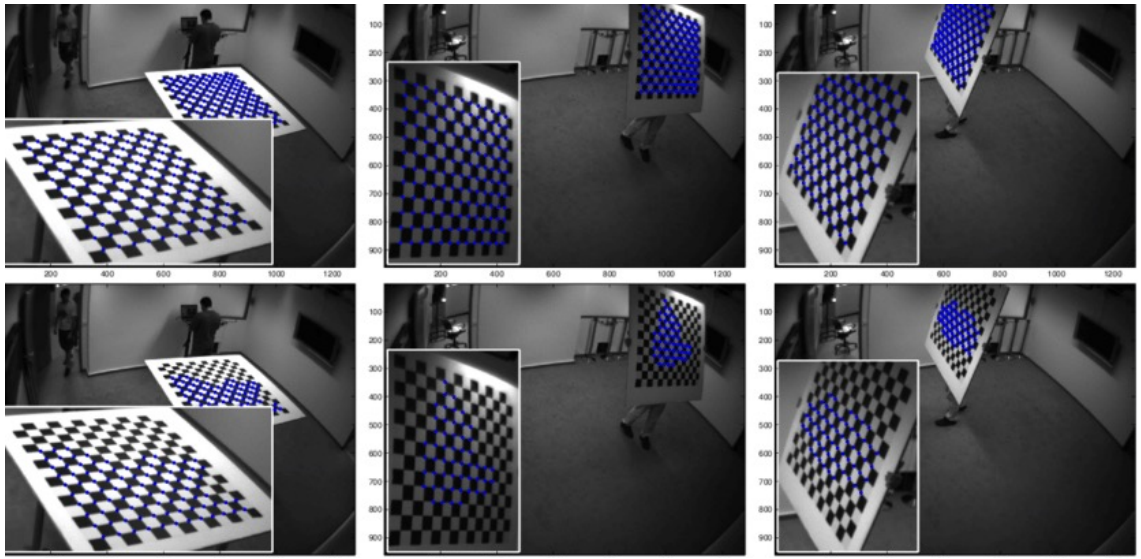
$$\mathbf{t} = K^{-1} \begin{bmatrix} p_{14} \\ p_{24} \\ p_{34} \end{bmatrix}$$

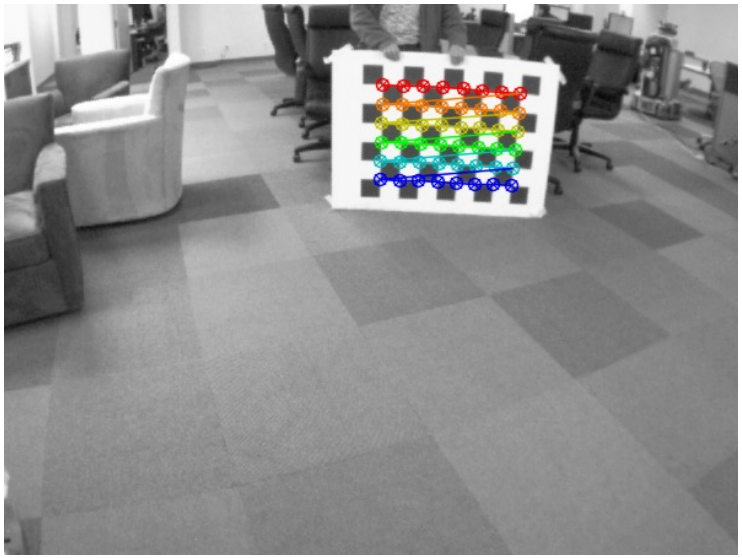
Steps of Camera Calibration

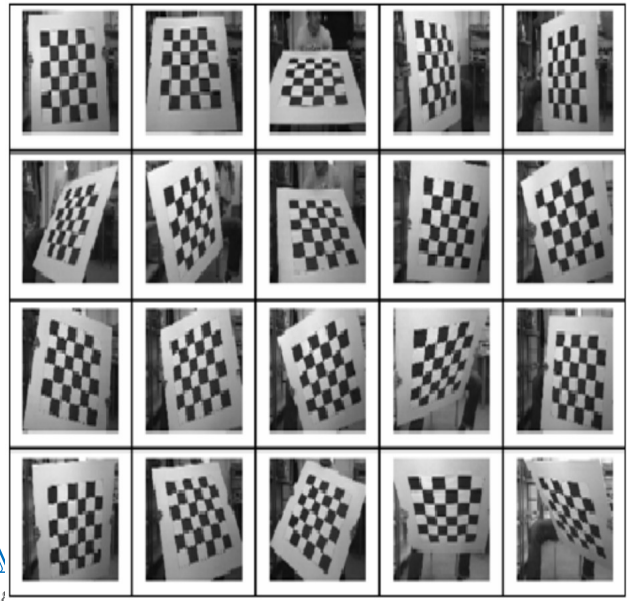
- From a set of corresponding points build A
- Compute the eigenvector **p** of A^TA for the smallest eigenvalue.
- Rearrange **p** as **P** as a 3x4 matrix
- Take the 3 x 3 matrix and compute the QR decomposition
  - Q is K (Calibration matrix)
  - R is rotation matrix
- Compute $T = K^{-1}P_{i4}$

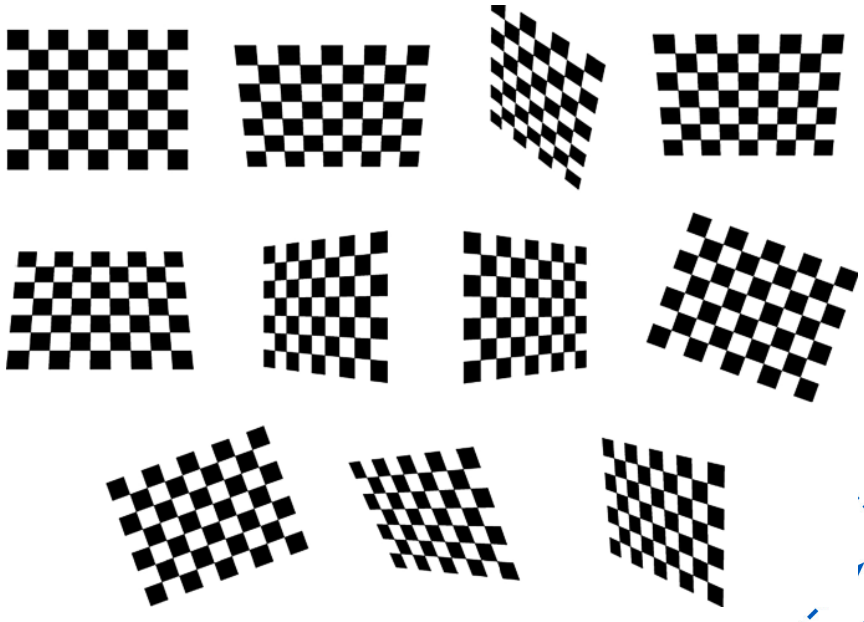


Cover more poses









Project 1: Camera Calibration

- The only **algorithm libraries** that are allowed to use.
  - PyTorch: Tensor operations.
  - PyPose: paper to read
    - Rotation representation, 2nd order optimizer may be useful
- You are encouraged to write a general calibration for
  - Differentiable ops (easy combine with learning)
  - Any distortions (Introduce next week)
  - Any devices (CPU, GPT, Apple Silicon GPU)
  - Batched data (parallel computing)
- Formal release of project will be next week.