Camera Calibration

Computer Vision (CS0029)

Outline

- **Camera system**
- Perspective images
- Intrinsic parameter
- Extrinsic parameter
- Camera Calibration

What is an image?

- A function – a 2D pattern of intensity values
  - $Intensity = I(u, v)$

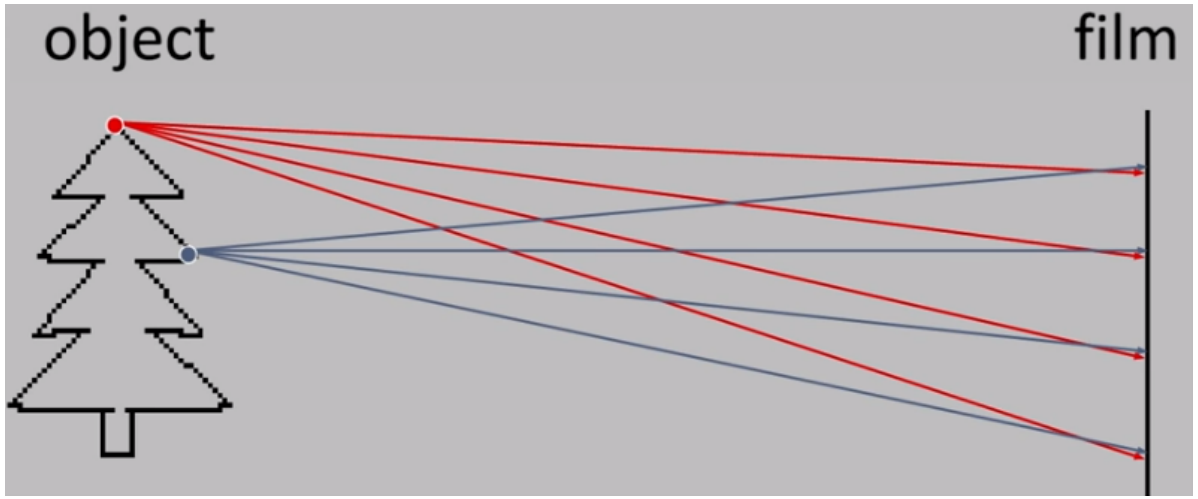
- More general: a 2D projection of 3D points
  - $(u, v) = T(x, y, z)$

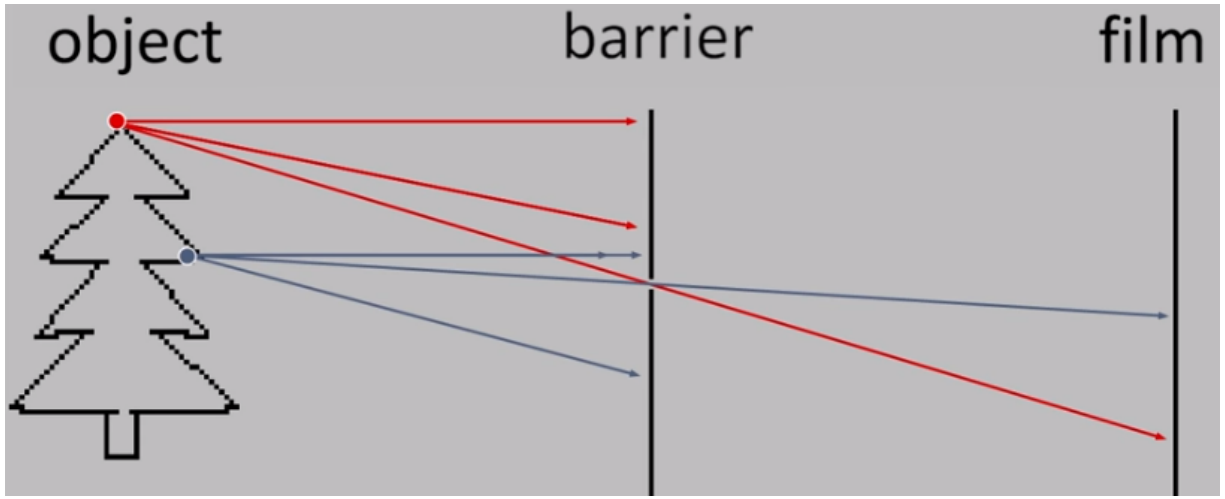
Camera System

- Camera: some devices that allows the **projection** of light from three dimensions to some medium (sensor) that record the light pattern
- When you do the projection, you lose information

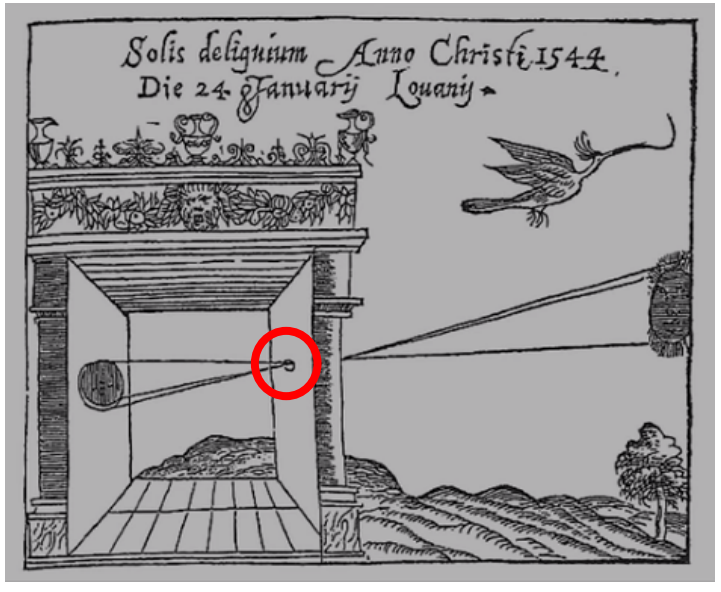
Pinhole Camera Model

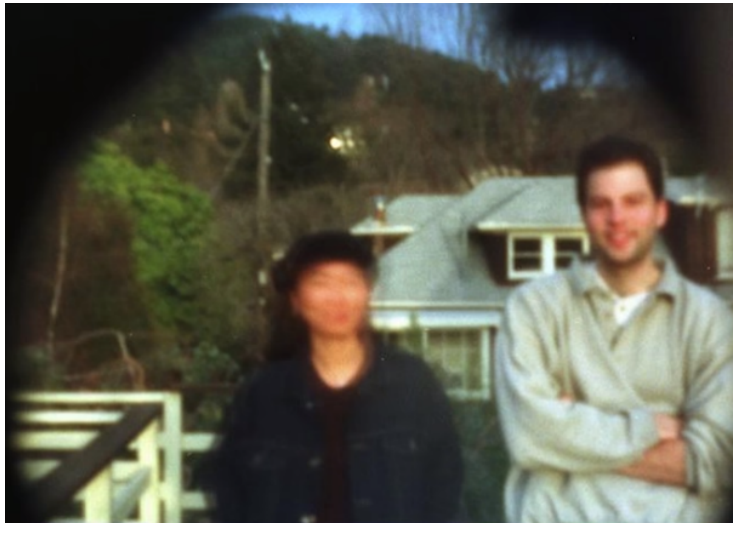
- If we have a sensor, let directly expose to the light….result?
  - All points on the sensor receive almost light from the all points on the object
  - How to fix this? Pinhole camera model
    - The small hole on the barrier is called aperture
    - Object will be upside down in the film

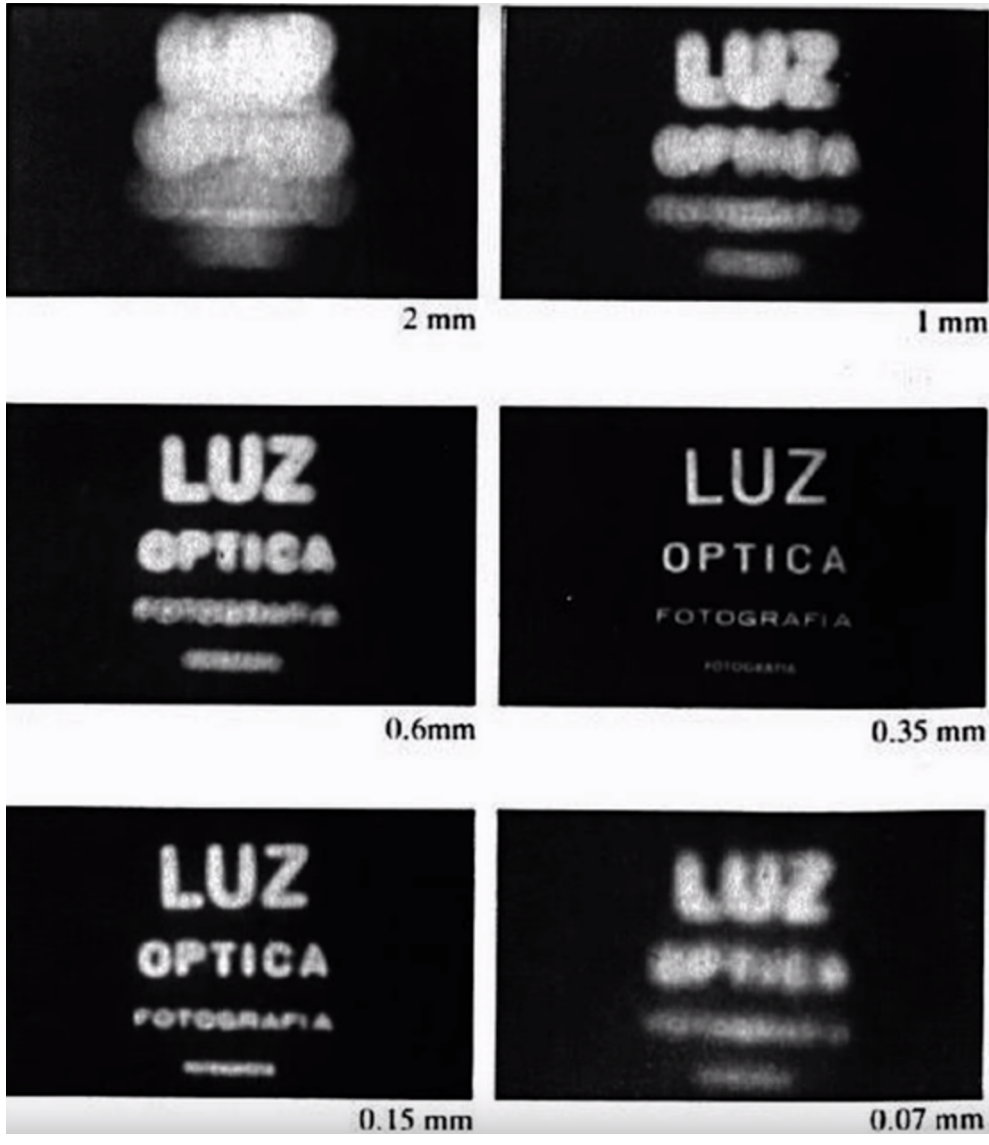




Aperture Size

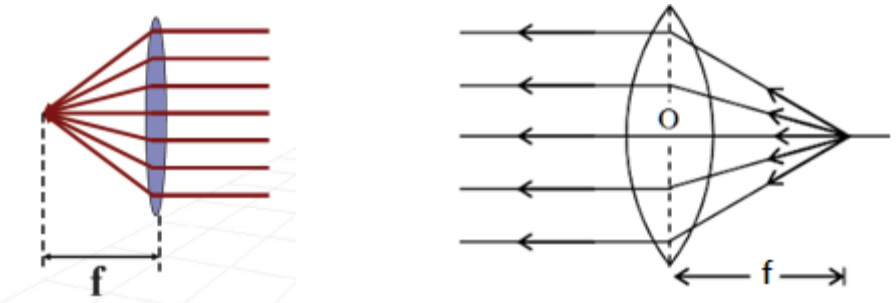


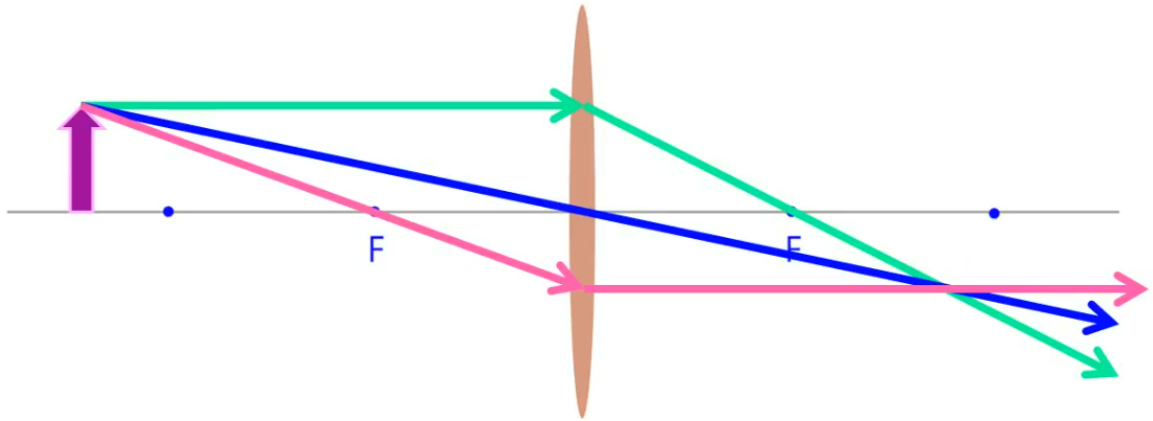




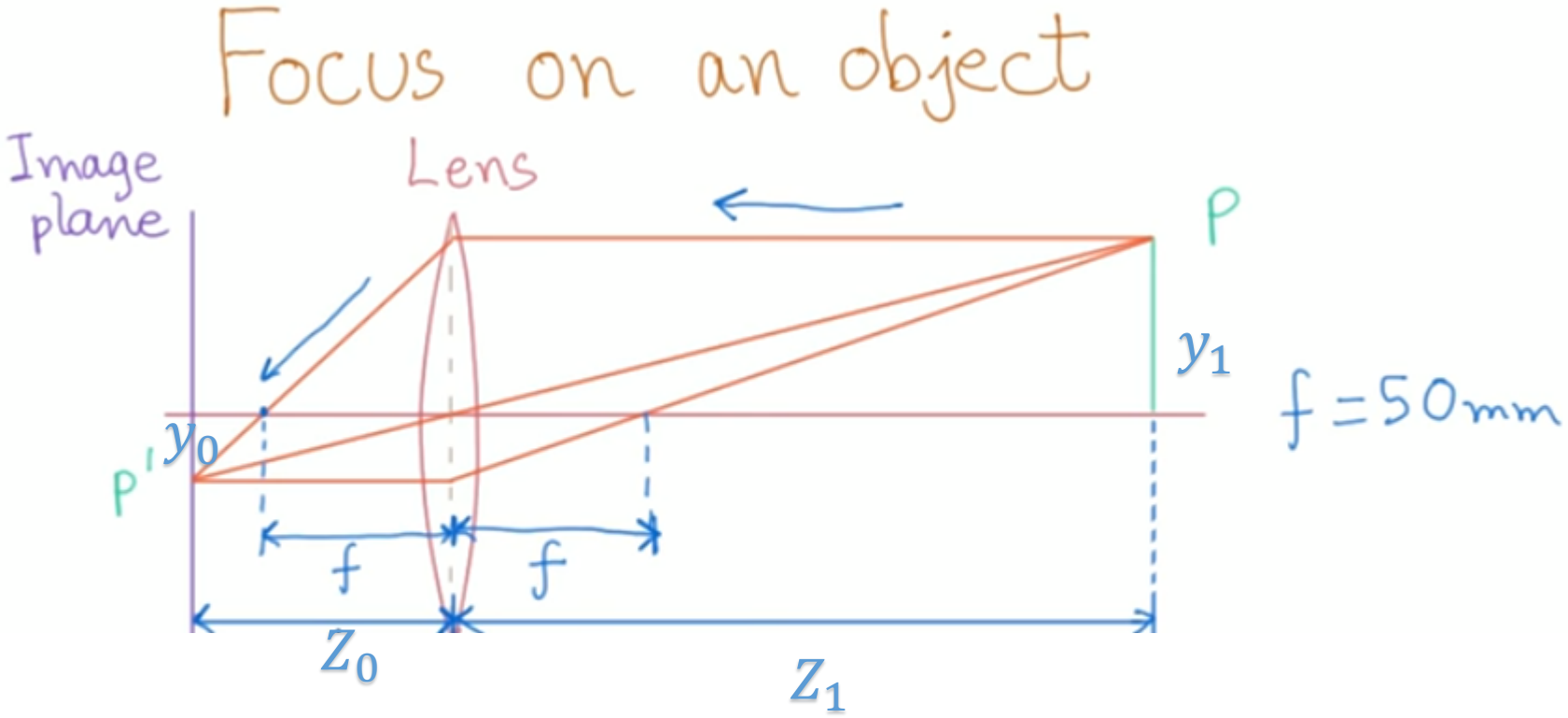
Thin Lens Model

- Focal length f is the distance behind lens where parallel rays (or parallel to the principle axis) converge (i.e. point where rays from infinitely distant source converge)





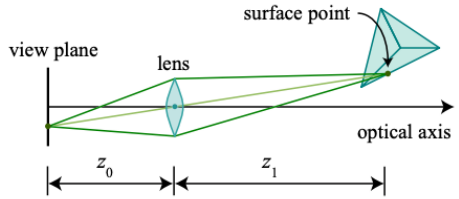
Practice

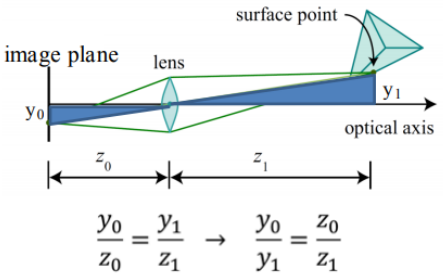


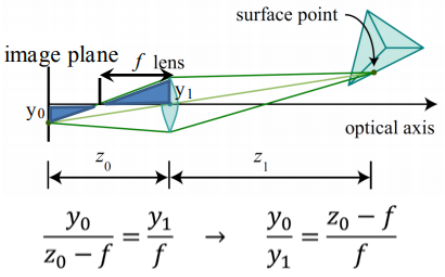
if $Z_1 = 2$,

what is the distance(Z_0)*between image plane and lens to make the object in focus?*

Thin Lens Model







- Solve for focal length f:

$$\frac{y_0}{y_1} = \frac{z_0}{z_1} \qquad \frac{y_0}{y_1} = \frac{z_0 - f}{f} \longrightarrow \frac{z_0}{z_1} = \frac{z_0 - f}{f}$$

Cross multiply

$$z_0 f = z_0 z_1 - z_1 f \longrightarrow z_0 f + z_1 f = z_0 z_1$$

Solve for f

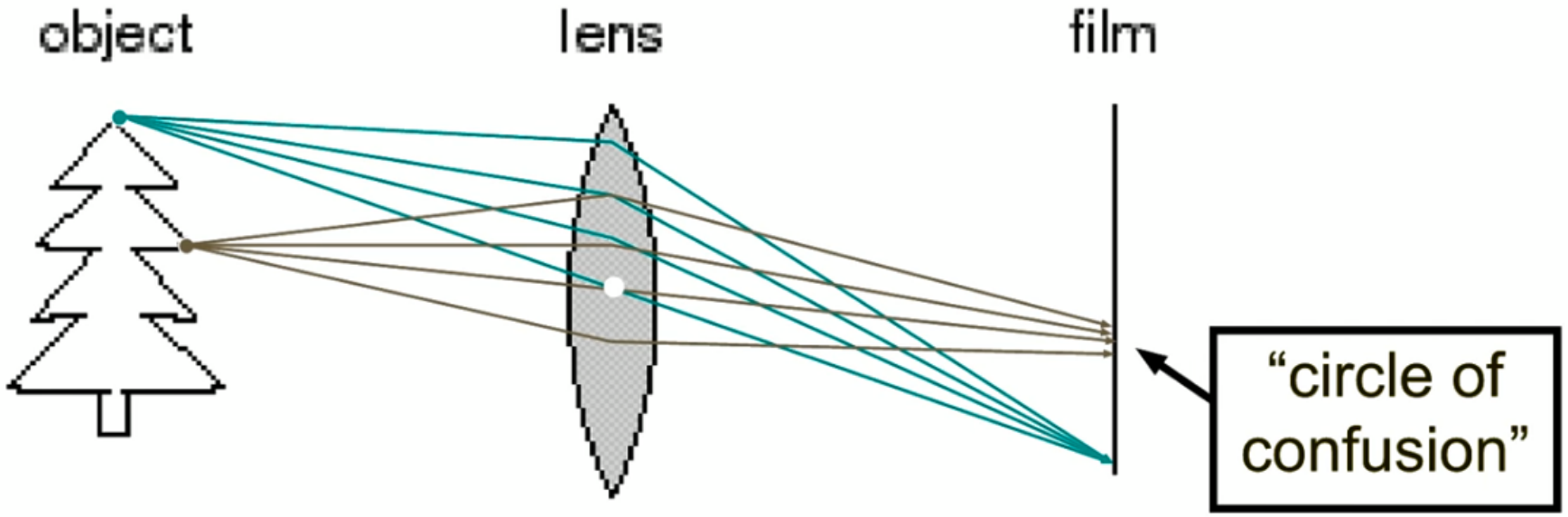
$$f = \frac{z_0 z_1}{z_0 + z_1} \longrightarrow \frac{1}{f} = \frac{z_0 + z_1}{z_0 z_1} \longrightarrow \boxed{\frac{1}{f} = \frac{1}{z_0} + \frac{1}{z_1}}$$

Thin lens equation

Points that satisfy the thin lens equation are "in focus"
(Given f and z_0, the object should be z_1 away)

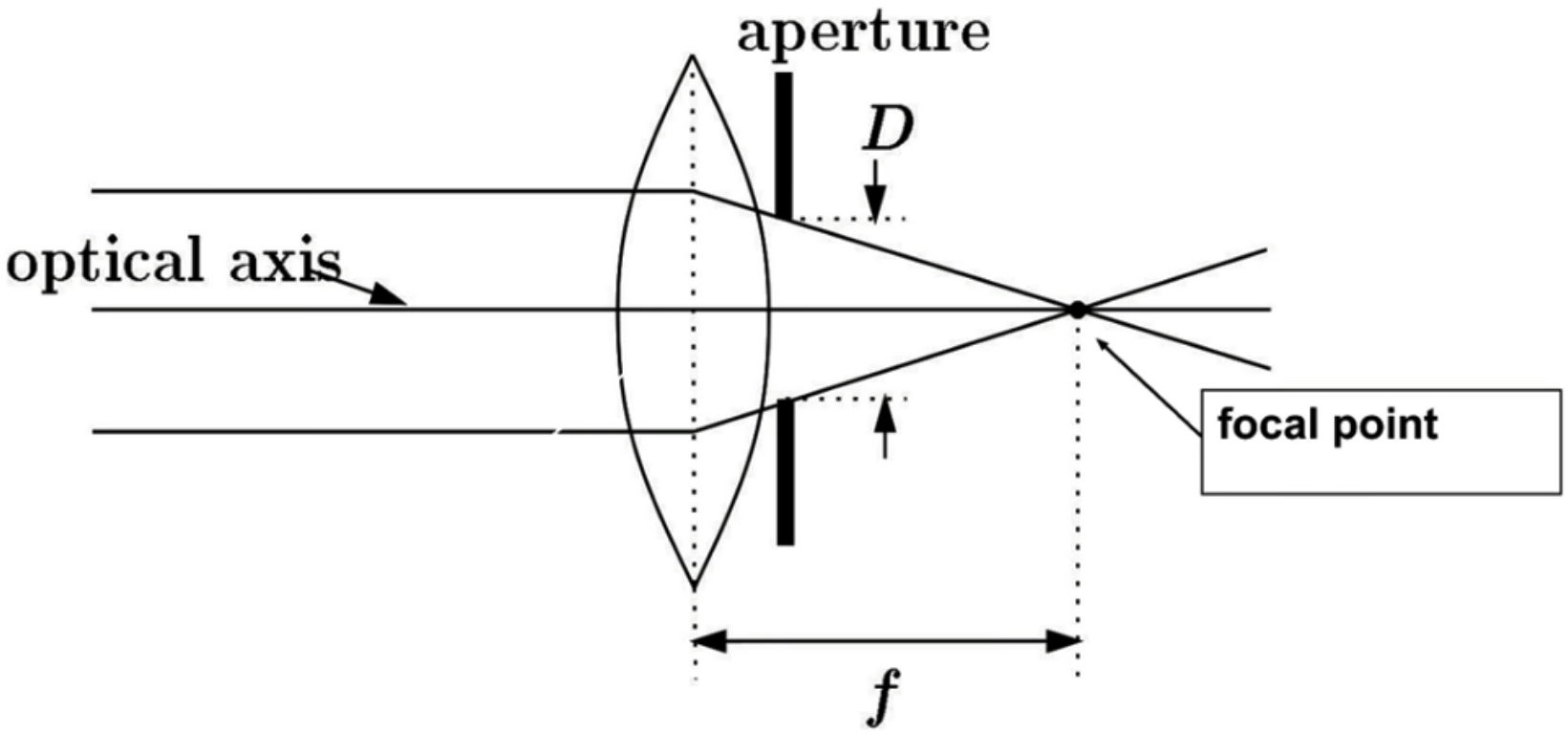
Lens

- A particular distance away will all be focused to the same point on the film
- But other points at different distances will be focused
- Change the shape of the lens can change the focus

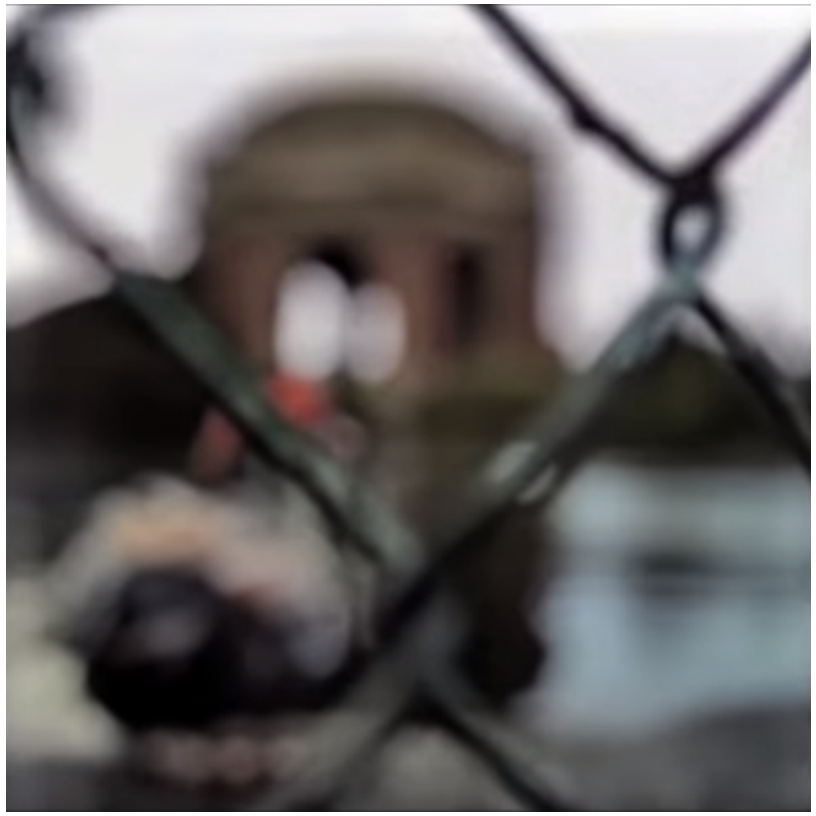


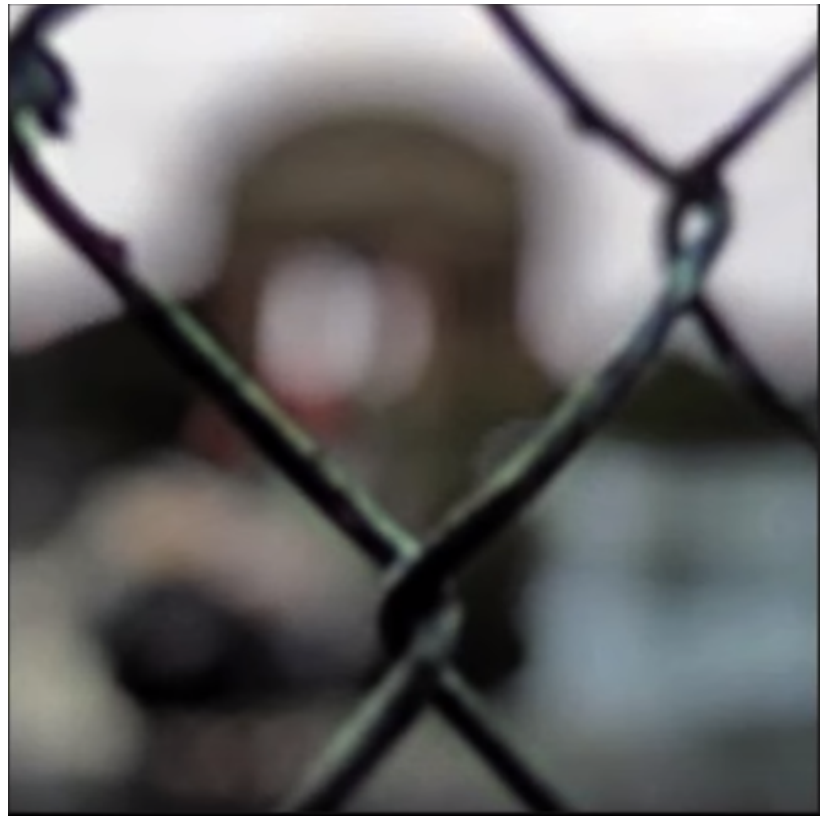
Lens

- Also has an aperture (not just a point now, restrict the set of ray that are coming in)



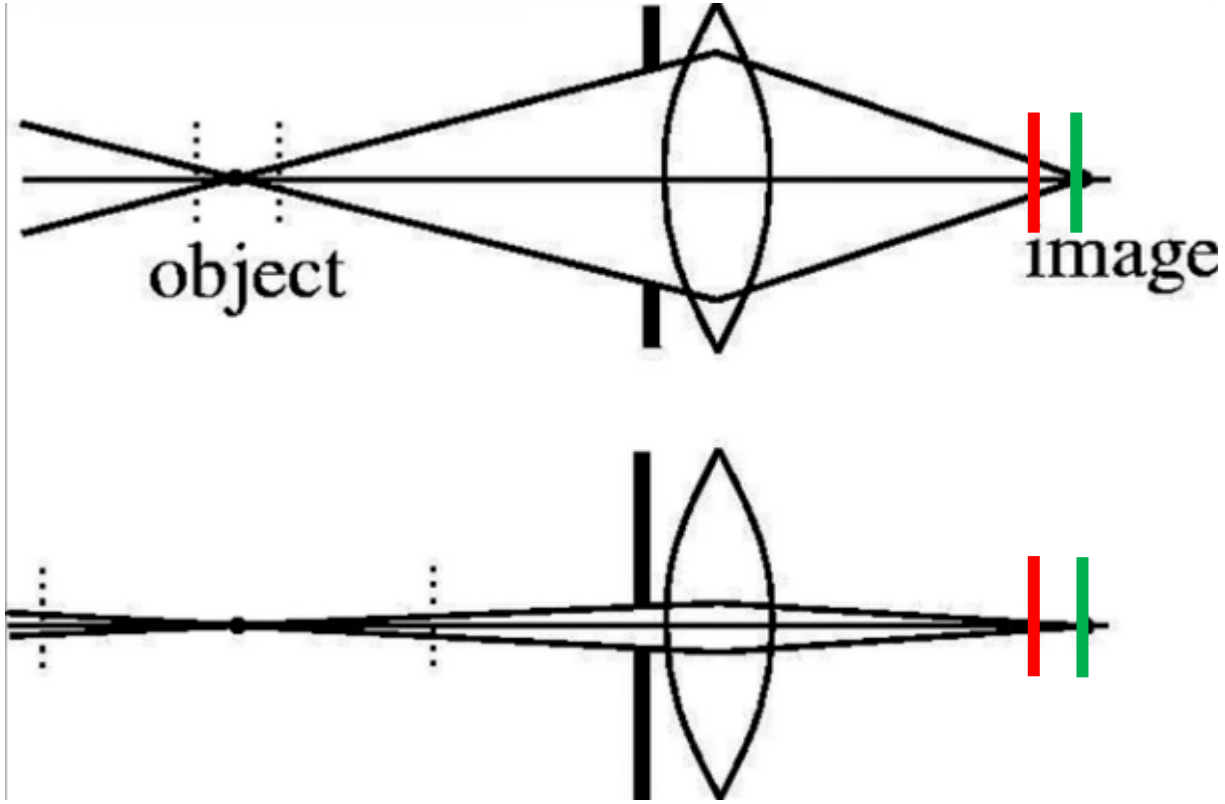
Varying Focus (Depth of Field)





Depth of Field

- Depth of field (DOF) is the distance between the nearest and the farthest objects that are in acceptably sharp focus in an image
- f: aperture, larger f -> smaller aperture size



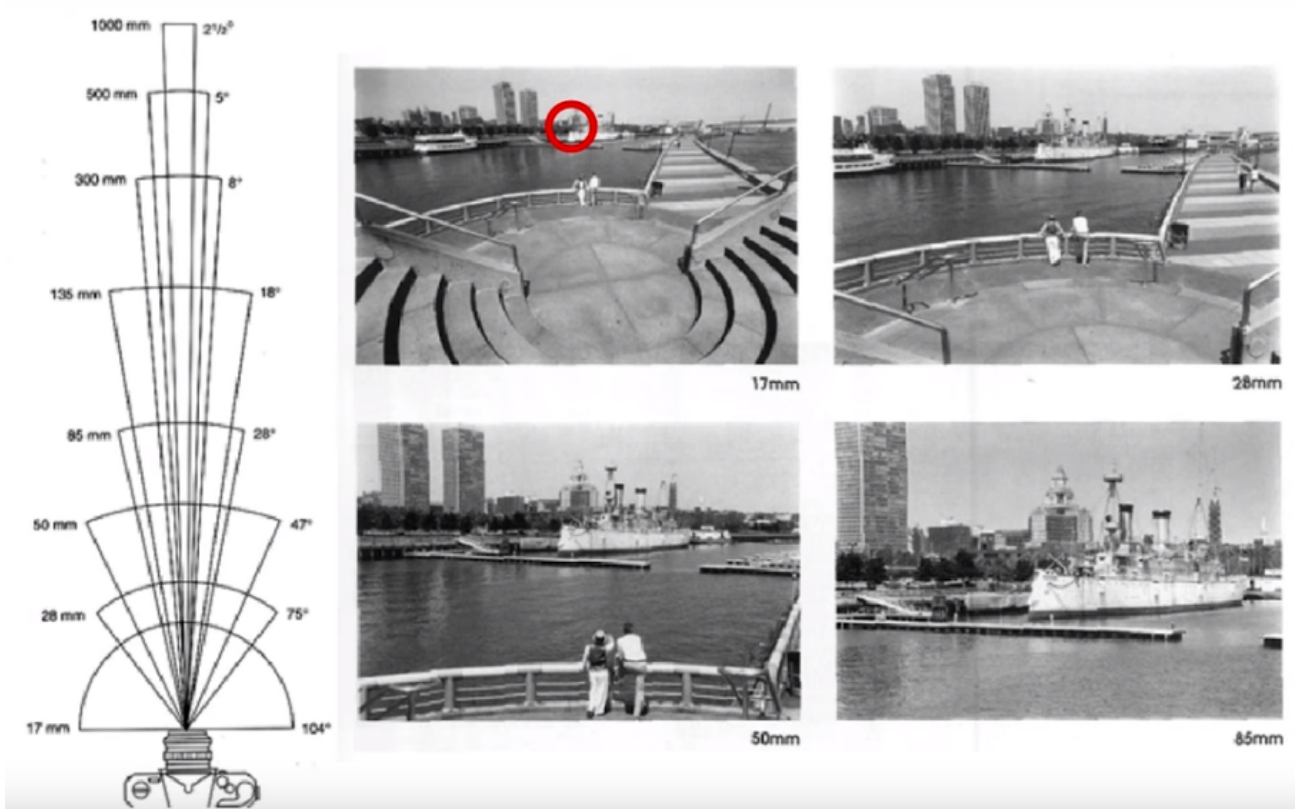
f/5.6=larger aperture

f/32=smaller aperture

Field of View

- How wide the view do we have

Increase the focal length of the lens

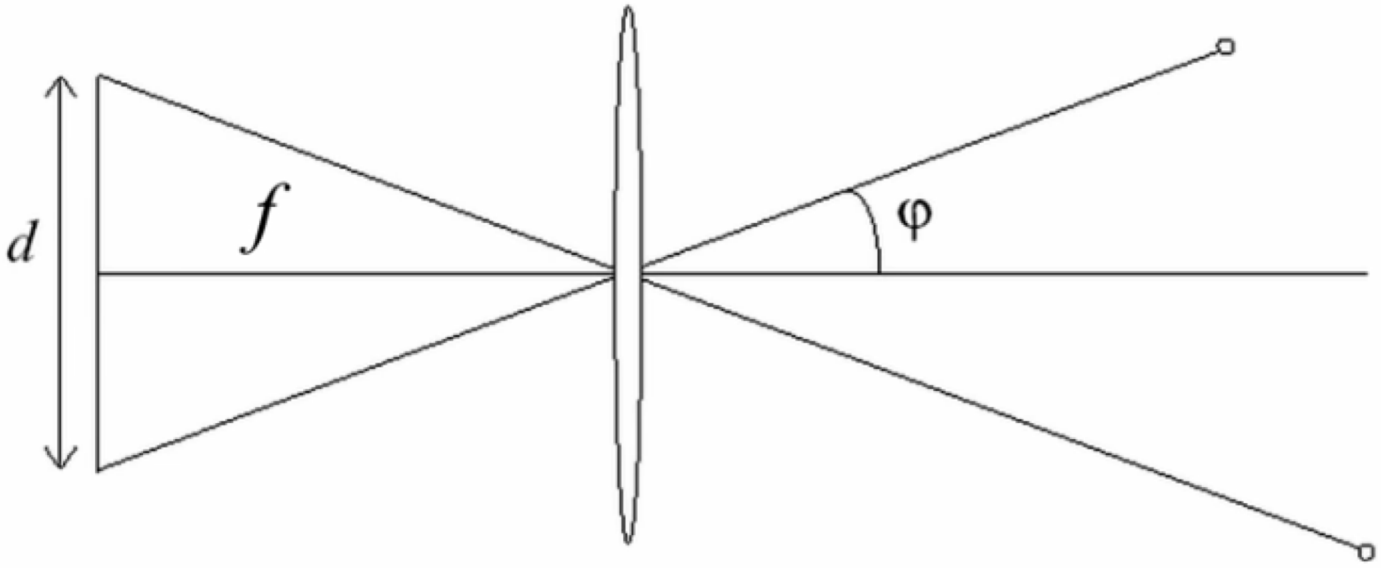


FOV depend on Focal Length

- Longer of the focal length (f) is smaller FOV (when d is fixed)
- Bigger image surface (d) is bigger FOV (when f is fixed)

d is the "retina" or sensor size

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



Larger Focal Length => Smaller FOV

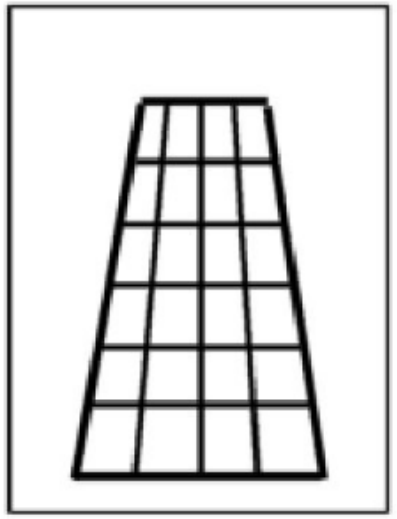
Zooming and Moving

- Are not the same

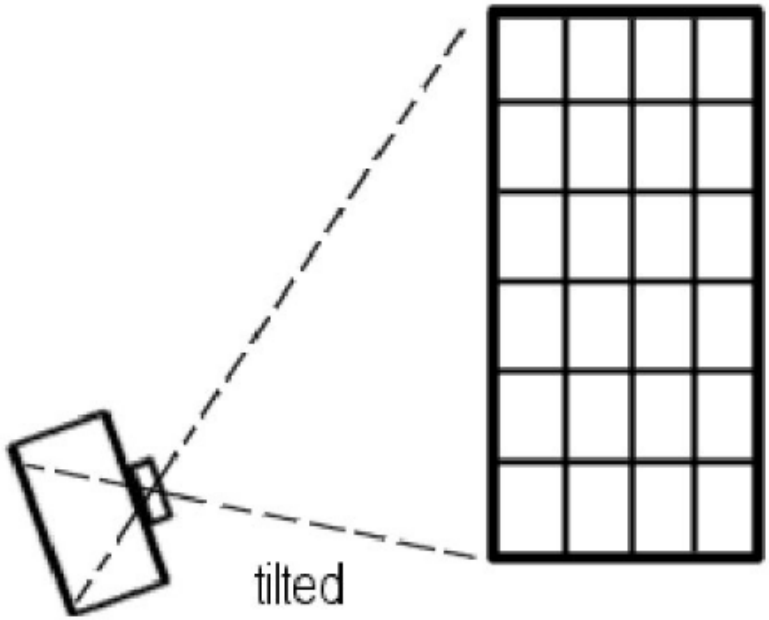
Large FOV, small f
Camera close to car

Small FOV, large f
Camera far from the car

Perspective Distortion



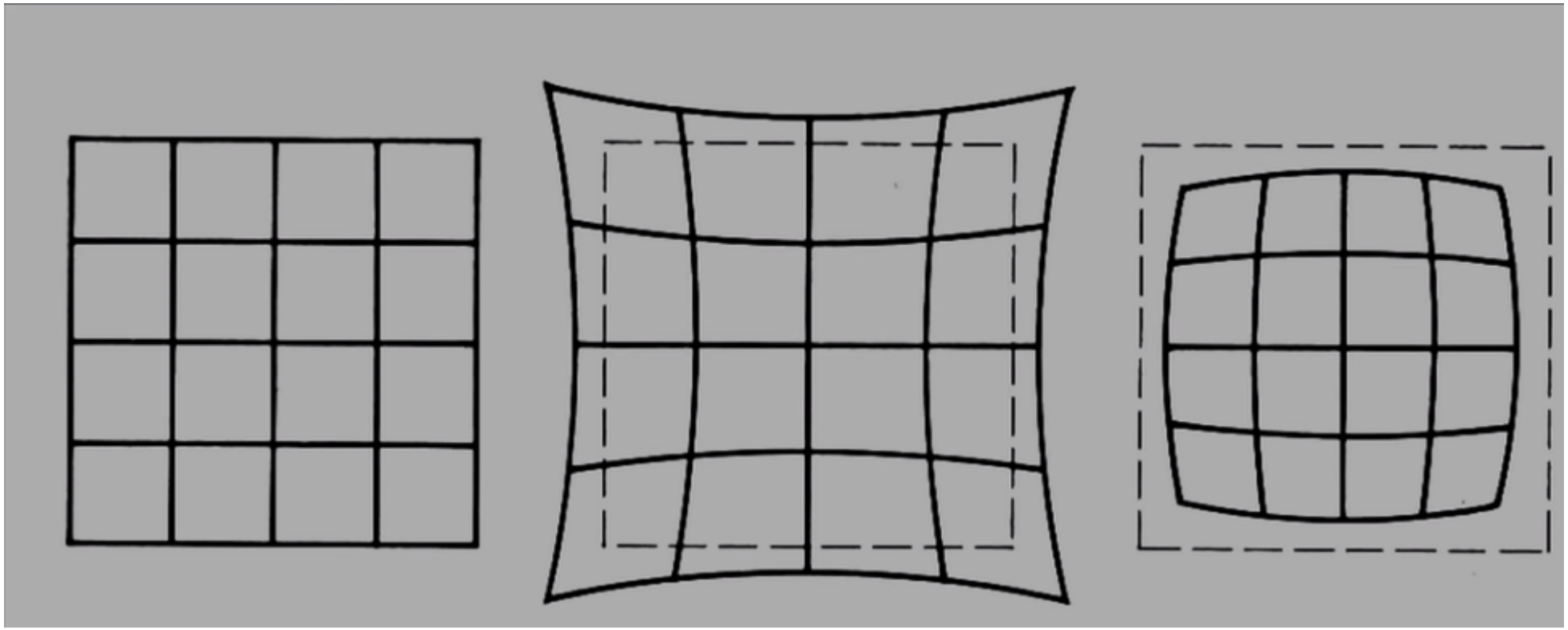
what the photo
will look like



tilted
camera

Geometric Distortion

- Lens are not perfect

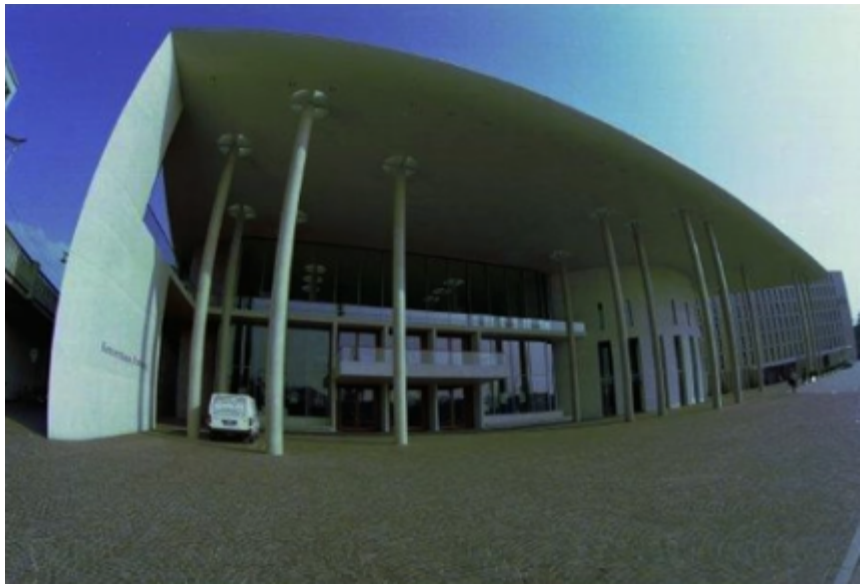


No distortion

Pin cushion

Barrel

Correct Distortion



Chromatic Aberration

- Rays of different wavelength focus in different planes

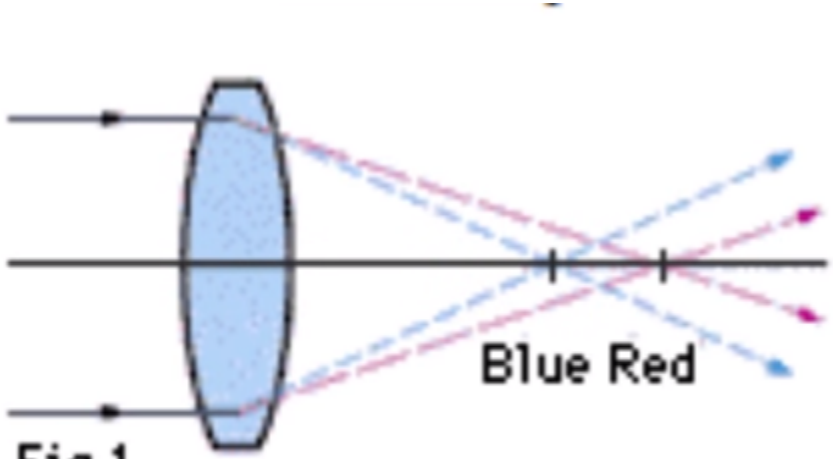


Fig.1
Axial chromatic aderration

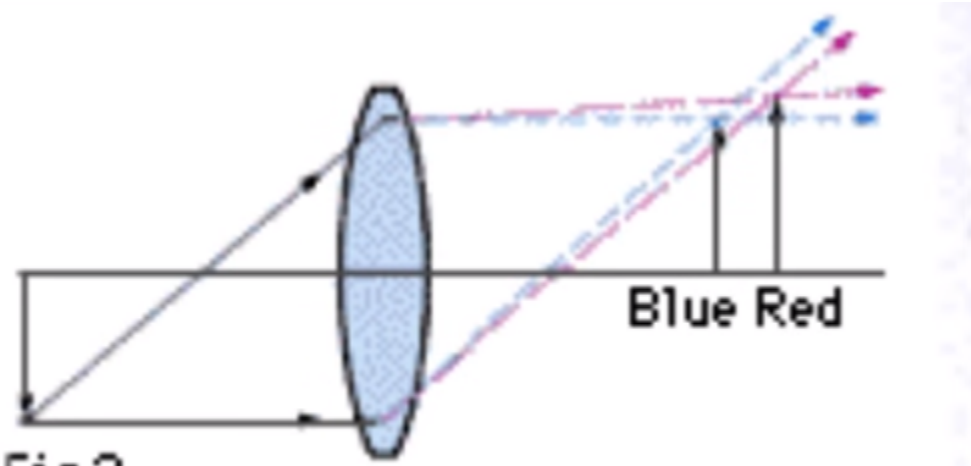
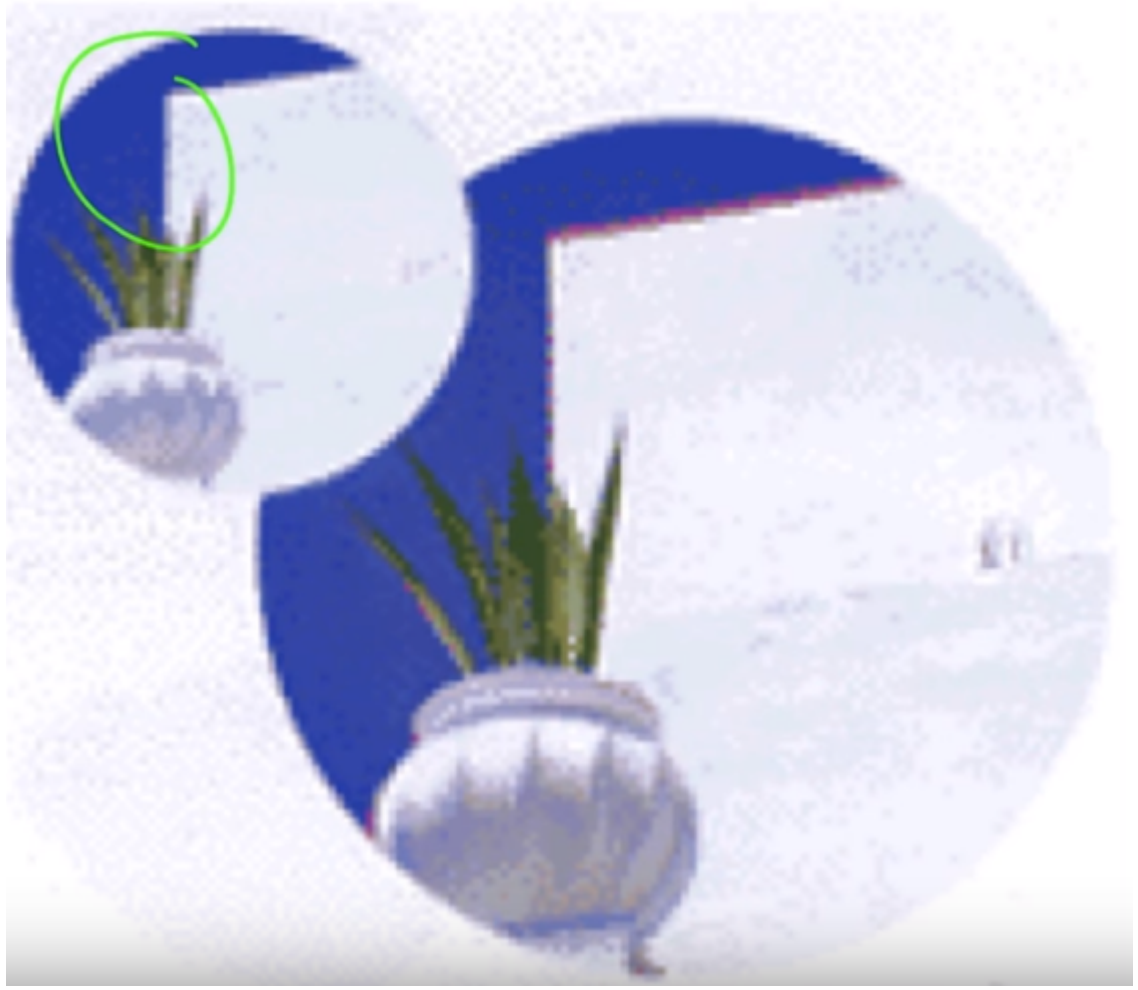


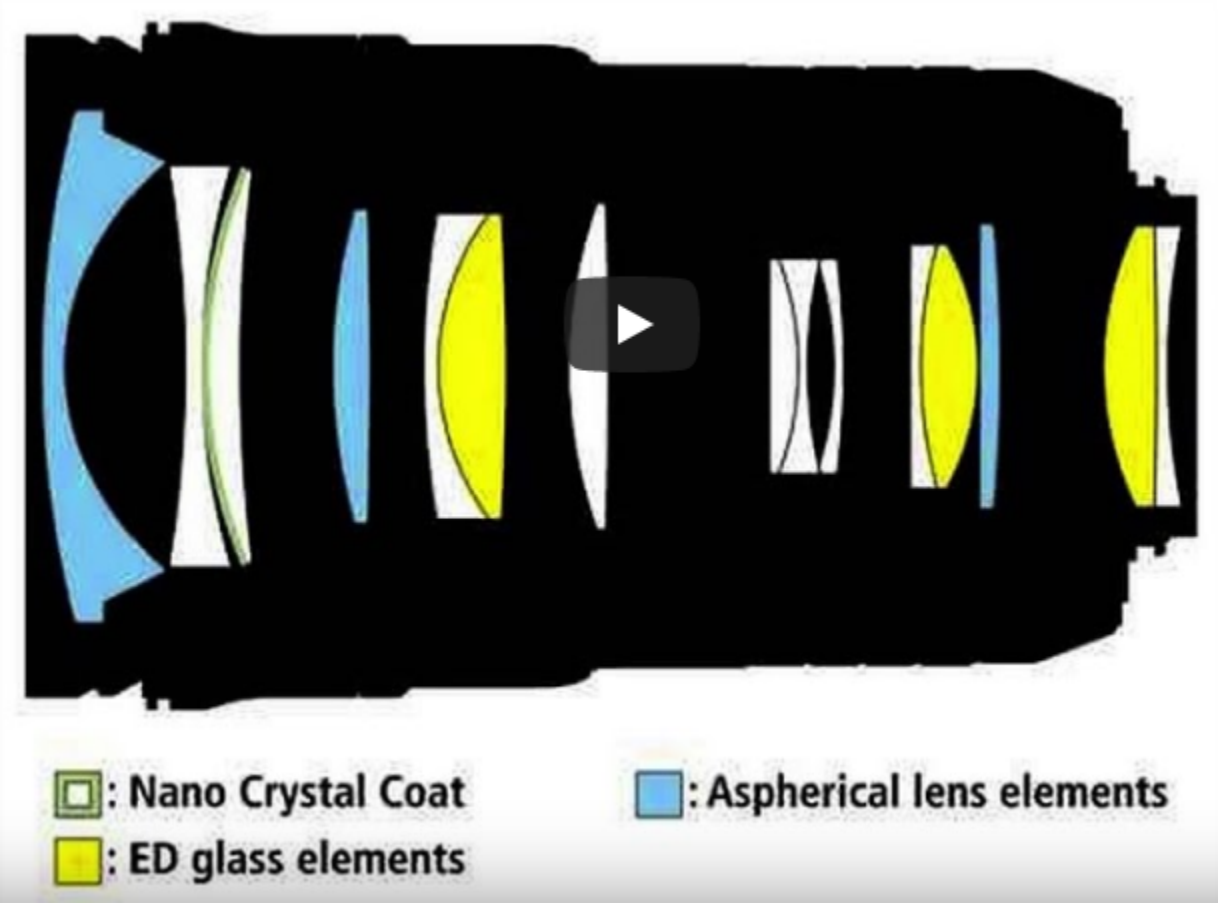
Fig.2
Magnification chromatic aderration

Chromatic Aberration



Lens System

- Money can solve the problem

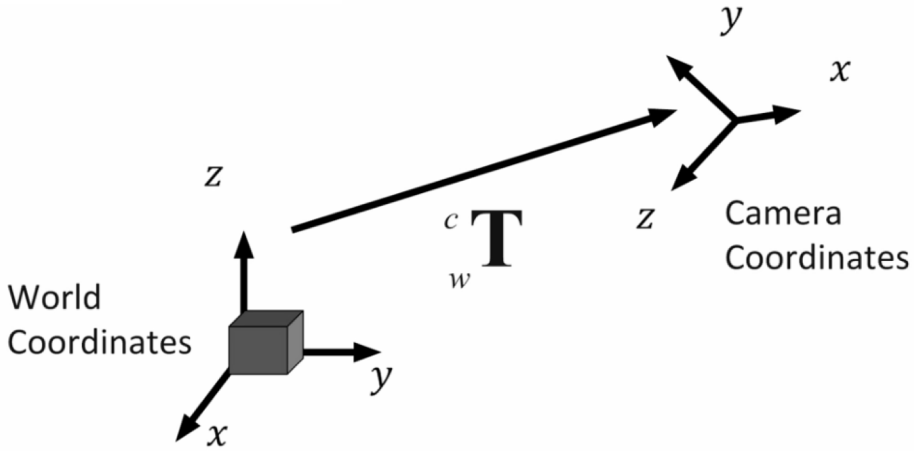


Key Takeaways

- Pin hole
  - Otherwise, every point on the film receive light from all points in real world (blur)
- Aperture size
  - Depth of field
- Focal length
  - Field of view
- Camera is not perfect
  - Chromatic Aberration, distortion…….

Overview of Camera Projection

Matrix Projection: $\mathbf{p} = \begin{bmatrix} su \\ sv \\ s \end{bmatrix} = \begin{bmatrix} * & * & * & * \\ * & * & * & * \\ * & * & * & * \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \mathbf{MP}$

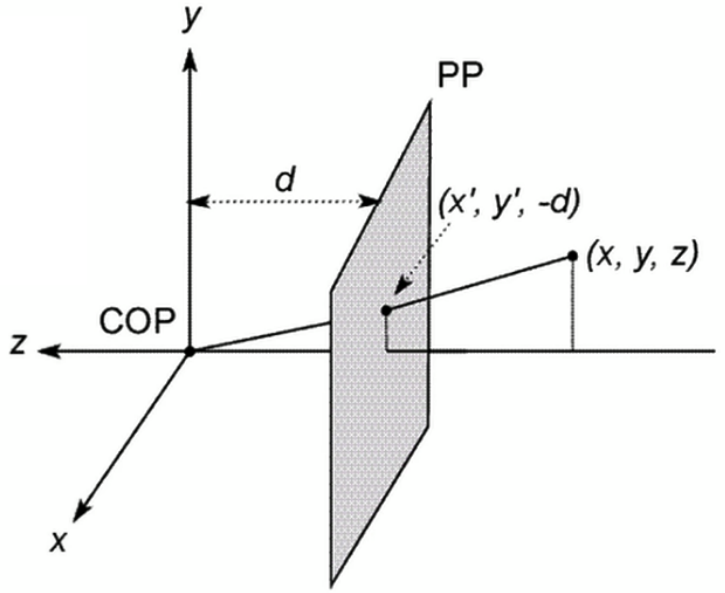


$\mathbf{M}$ can be decomposed into $\mathbf{t} \rightarrow \mathbf{R} \rightarrow$ project $\rightarrow \mathbf{A}$

$$\mathbf{M} = \begin{bmatrix} fa & c & u_c \\ 0 & f & v_c \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{R}_{3x3} & \mathbf{0}_{3x1} \\ \mathbf{0}_{1x3} & 1 \end{bmatrix} \begin{bmatrix} \mathbf{I}_{3x3} & \mathbf{t}_{3x1} \\ \mathbf{0}_{1x3} & 1 \end{bmatrix}$$

intrinsic projection Rotation Translation

$\mathbf{A}$: to image coordinate (pixels)

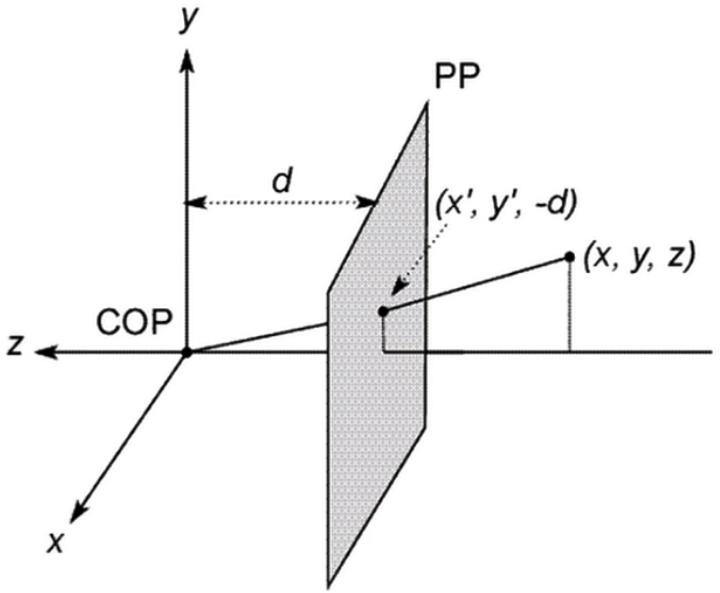


Outline

- Camera system
- **Perspective images**
- Intrinsic parameter
- Extrinsic parameter
- Camera Calibration

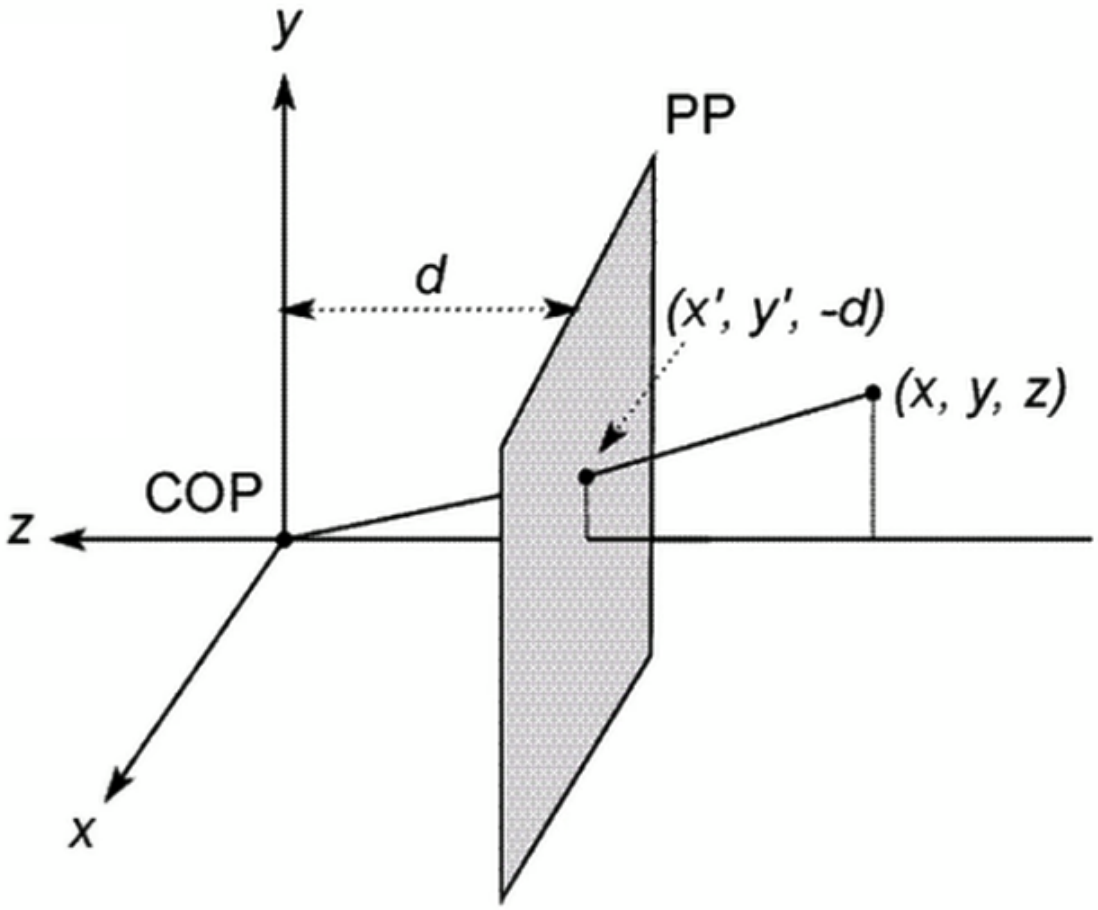
$$\mathbf{M} = \begin{bmatrix} fa & c & u_c \\ 0 & f & v_c \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{R}_{3x3} & \mathbf{0}_{3x1} \\ \mathbf{0}_{1x3} & 1 \end{bmatrix} \begin{bmatrix} \mathbf{I}_{3x3} & \mathbf{t}_{3x1} \\ \mathbf{0}_{1x3} & 1 \end{bmatrix}$$

intrinsic **projection** Rotation Translation



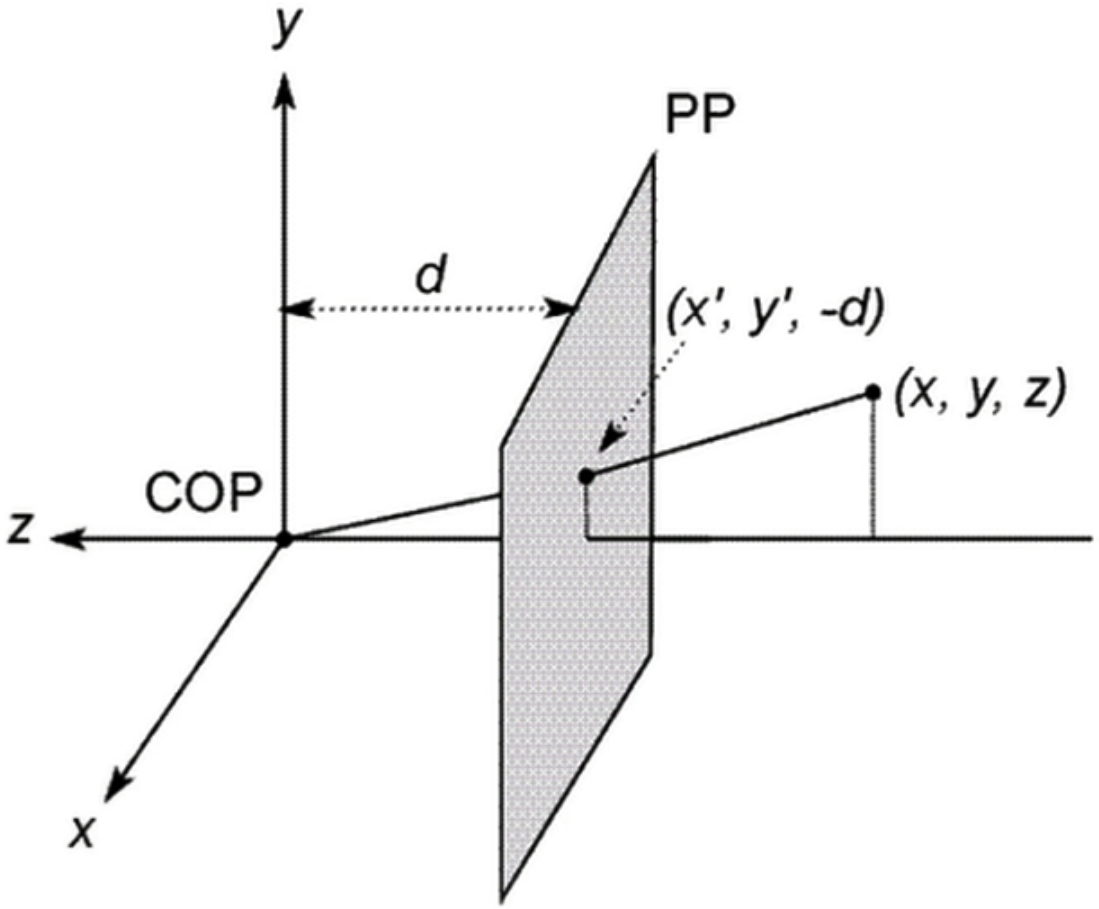
Coordinate System

- Use pin-hole model
- Put the optical center (center of projection) at the origin
- Standard coordinate system
- Put the image plane (projection plane) in front of the COP
  - Just mathematics convenient
  - Z – right hand rule
  - Z – points to the inside of the camera (looks down the negative z axis)



Modeling Projection

- Projection equation
  - Compute intersection with perspective projection of ray from (x, y, z) to COP
  - Derived using similar triangles
  - $(x, y, z) \rightarrow (-d\frac{x}{z}, -d\frac{y}{z}, -d)$
    - Similar triangle
- We got the projection by throwing out the last coordinate
  - $(x', y') \rightarrow (-d\frac{x}{z}, -d\frac{y}{z})$
  - Distant objects are smaller
- How this explain the moon always follow you?



Homogenous Coordinate

- Is this a linear transformation?
  - No- division by the (not constant) Z is non-linear
- Trick: add one more coordinate:

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

homogeneous image (2D) coordinates

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

homogeneous scene (3D) coordinates

Homogenous Coordinate

- Converting from homogeneous coordinates:

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

- This makes homogenous coordinates invariant under scale
  - Multiple whole coordinate system by a. After converting back to standard coordinate system, everything is the same

Perspective Projection

- Why homogenous system?
  - Projection is a matrix multiply using homogeneous coordinates
    - The main reason is for translation

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

Perspective Projection

- How does scaling the projection matrix change the transformation?

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} ax \\ ay \\ az \\ 1 \end{bmatrix} = \begin{bmatrix} ax \\ ay \\ az \end{bmatrix} = \begin{bmatrix} \frac{ax}{az} \\ \frac{ay}{az} \end{bmatrix} \Rightarrow \left(\frac{x}{z}, \frac{y}{z}\right) \Rightarrow (u, v)$$

Perspective Projection

- Why homogenous system?
  - Projection is a matrix multiply using homogeneous coordinates
    - The main reason is for translation

$$\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} \frac{x'}{z'} \\ \frac{y'}{z'} \end{bmatrix} \Rightarrow (u, v)$$

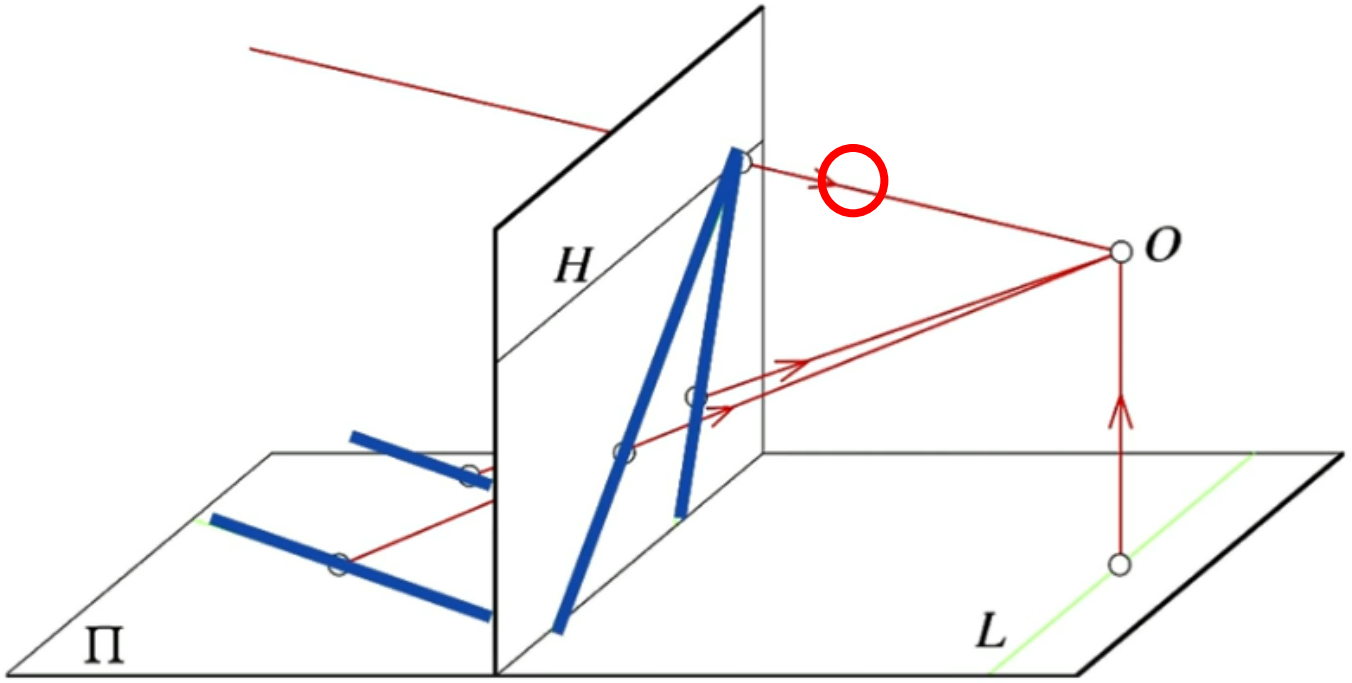
$$\begin{bmatrix} a & b & c & 0 \\ d & e & f & 0 \\ g & h & i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \\ t_z \\ 1 \end{bmatrix} =$$

$$= \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} a & b & c & 0 \\ d & e & f & 0 \\ g & h & i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c & t_x \\ d & e & f & t_y \\ g & h & i & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} \frac{x'}{z'} \\ \frac{y'}{z'} \end{bmatrix} \Rightarrow (u, v)$$

Can you explain why vanishing point happen in the image by what we learn?

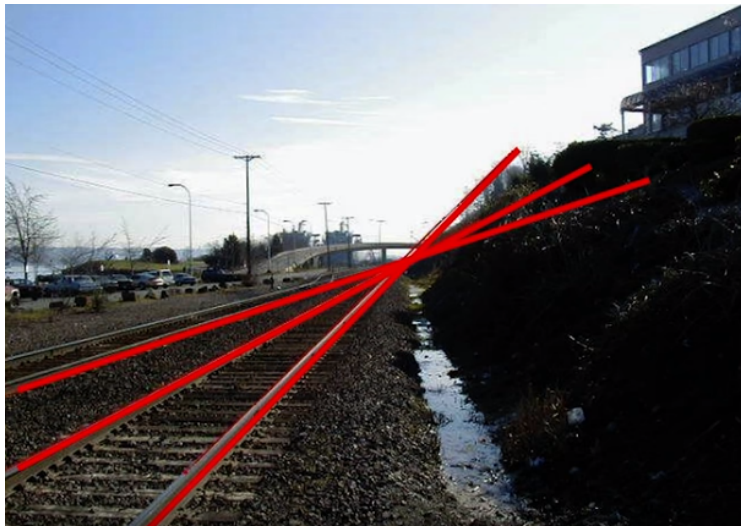
Parallel Lines

- Vanishing point



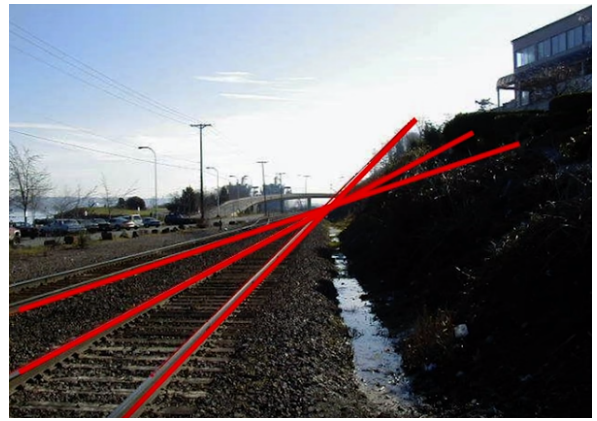
$$(x', y') \rightarrow (-d\frac{x}{z}, -d\frac{y}{z})$$

- Hint : line in 3D space
  - $x(t) = x_0 + at$
  - $y(t) = y_0 + bt$
  - $z(t) = z_0 + ct$



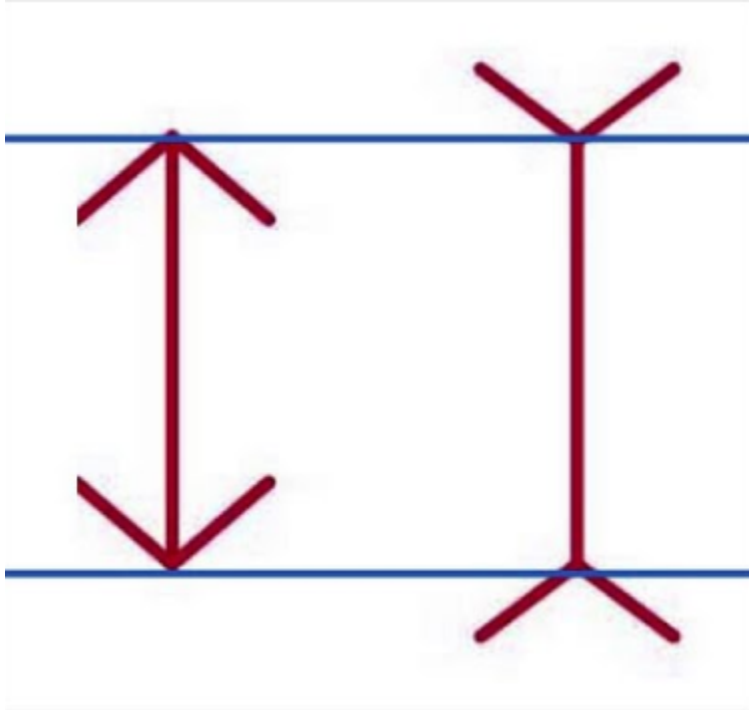
See the Vanishing Point from Math

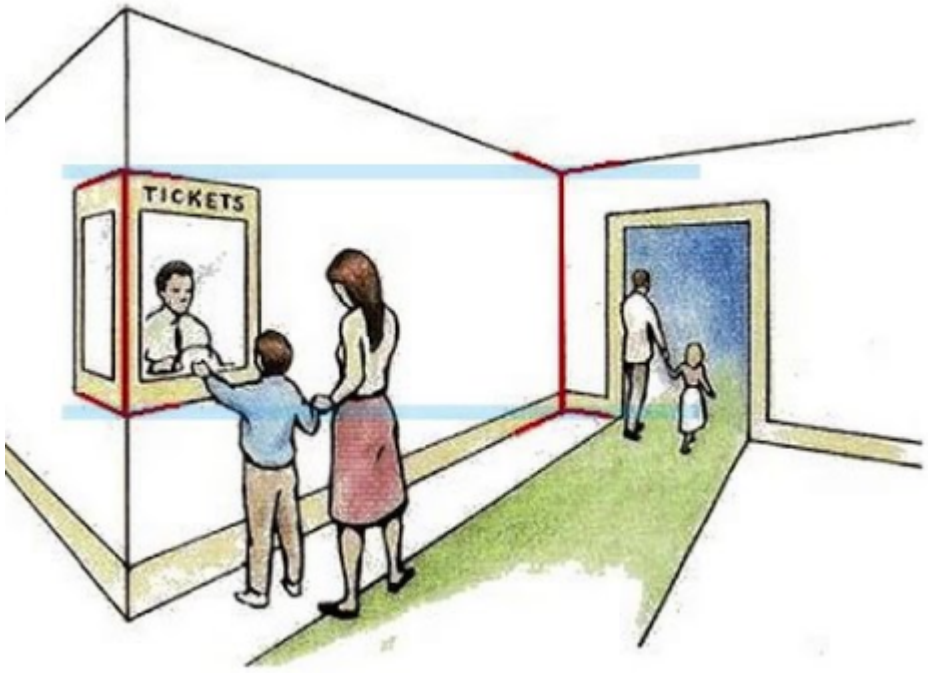
- Line in 3D space
  - $x(t) = x_0 + at$
  - $y(t) = y_0 + bt$
  - $z(t) = z_0 + ct$
- Perspective projection of the line
  - $x'(t) = \frac{x}{z} = \frac{x_0+at}{z_0+ct}$
  - $y'(t) = \frac{y}{z} = \frac{y_0+bt}{z_0+ct}$
- In the limit as t $\rightarrow \pm\infty$, we have (for $c \neq 0$)
  - $x'(t) \rightarrow \frac{a}{c}$
  - $y'(t) \rightarrow \frac{b}{c}$
  - x_0, y_0, z_0 are missing
  - No matter where the line starts, as long as their direction is the same, they will converge to same point in the image
  - So, what does that mean if c = 0



Human Vision Illusion

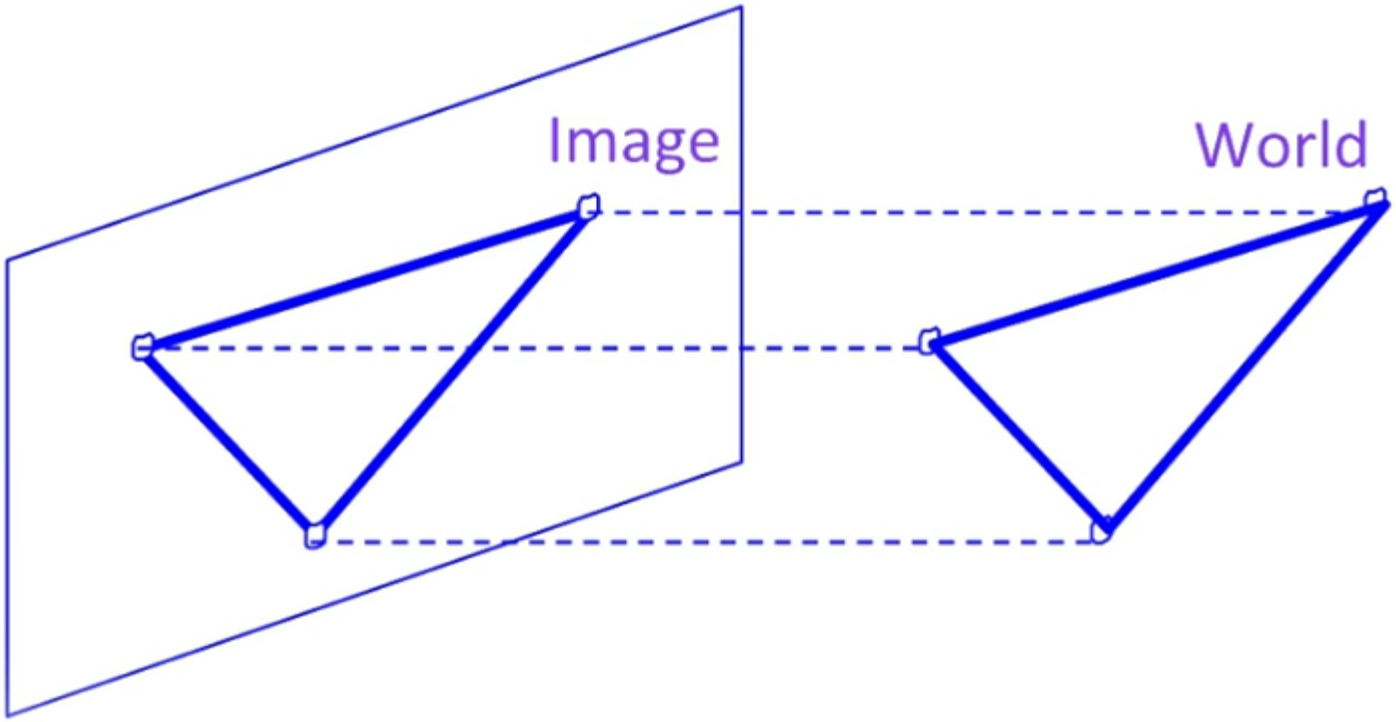
- Human brain will try to undo the projection transformation





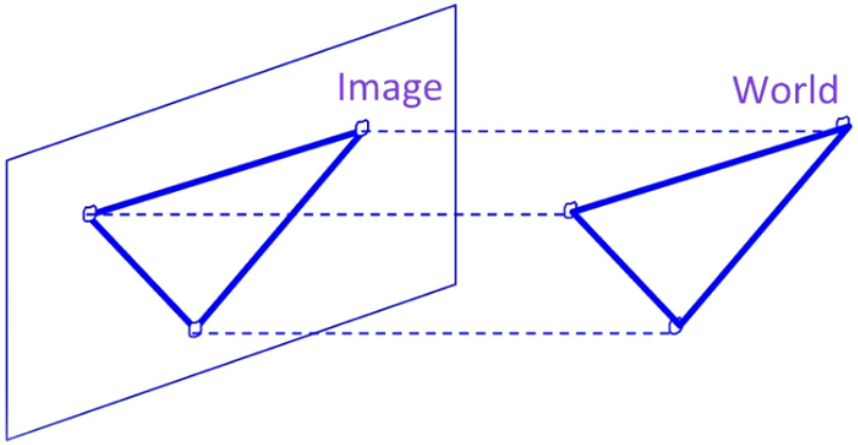
Other Model: Orthographic Projection

- Special case of perspective projection
  - Distance from the COP to the image plane is infinite
  - Also called parallel projection $(x, y, z) \ -> \ (x, y)$: telephoto lens model



Other Model: Orthographic Projection

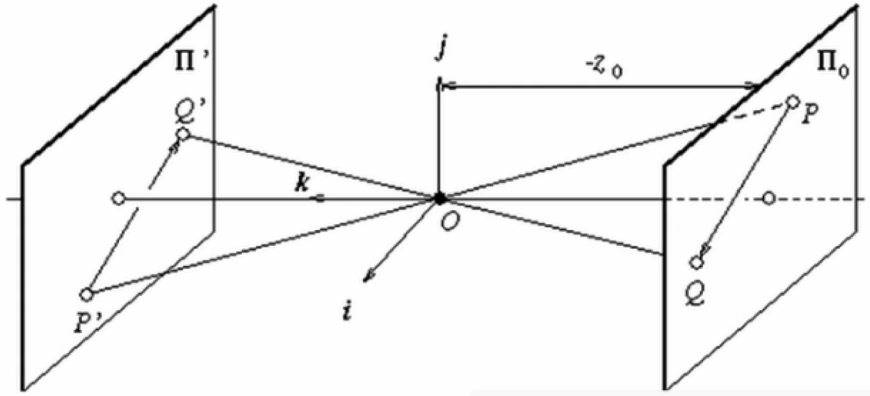
- Special case of perspective projection
  - Distance from the COP to the image plane is infinite
  - Also called parallel projection $(x, y, z) \; -> \; (x, y)$: telephoto lens model



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

Other Model: Weak Perspective

- Perspective effects, but not over the scale of individual objects



$$(x, y, z) \rightarrow \left(\frac{fx}{z_0}, \frac{fy}{z_0} \right)$$

$$(x, y, z) \rightarrow \left(\frac{fx}{z_0}, \frac{fy}{z_0} \right) \quad \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1/s \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ 1/s \end{bmatrix} \Rightarrow (sx, sy)$$

Three Camera Projections

3-d point position → 2-d image

(1) Perspective: $(x, y, z) \rightarrow \left(\frac{fx}{z}, \frac{fy}{z} \right)$

(2) Weak perspective: $(x, y, z) \rightarrow \left(\frac{fx}{z_0}, \frac{fy}{z_0} \right)$

(3) Orthographic: $(x, y, z) \rightarrow (x, y)$

Key Takeaways

- Homogeneous coordinate
  - One more element in the point
  - Keep scale invariant
  - Make the perspective projection (translation, rotation and scale transformation) becomes a linear transformation (a matrix multiplication)
    - $p' = M * p$ (M is transformation matrix, p is point in 3D, p' is point on image)
- Convert a point in homogeneous coordinate back to standard coordinate

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

- The differences between perspective projection, orthographical projection and weak perspective projection.

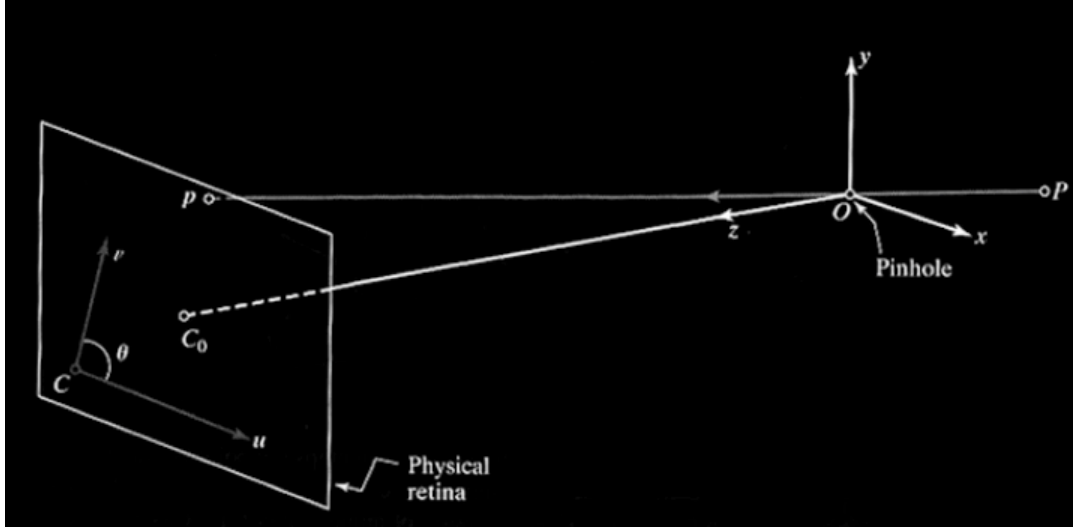
Outline

- Camera system
- Perspective images
- **Intrinsic parameter**
- Extrinsic parameter
- Camera Calibration

$$\mathbf{M} = \begin{bmatrix} fa & c & u_c \\ 0 & f & v_c \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{R}_{3x3} & \mathbf{0}_{3x1} \\ \mathbf{0}_{1x3} & 1 \end{bmatrix} \begin{bmatrix} \mathbf{I}_{3x3} & \mathbf{t}_{3x1} \\ \mathbf{0}_{1x3} & 1 \end{bmatrix}$$

intrinsic projection Rotation Translation

Intrinsic Parameter

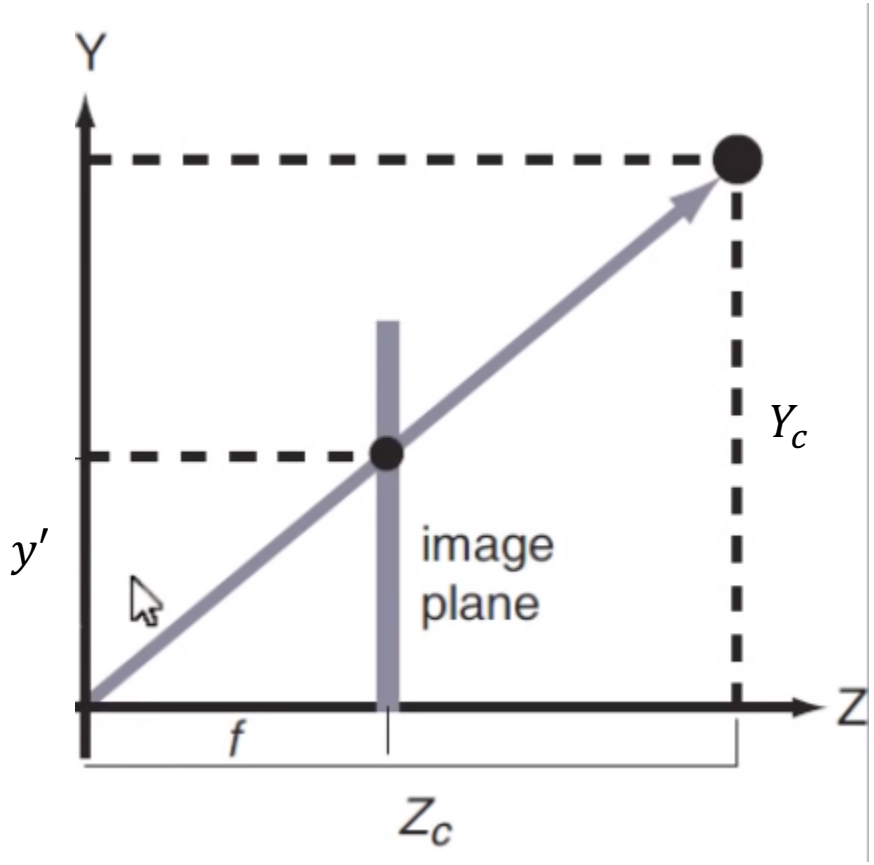


- We already know how to project a point to image plane, any other else inside the camera?
- The real world is not perfect….

- We usually use “mm” to represent the focal length and image space, but what we want in the end is the coordinate in pixels
  - **Rasterization (image space to pixel)**
- Is the image center really at “center”?. Is the center the origin?
  - **Assign pixel space origin and image space origin**
- Radial distortion
- Is the u and v on the image plane really perpendicular?

Intrinsic Parameter

- $\frac{x'}{f} = \frac{X_c}{Z_c}, \frac{y'}{f} = \frac{Y_c}{Z_c}$
- $x' = f\frac{X_c}{Z_c},\ \ y' = f\frac{Y_c}{Z_c}$
  - $x'' = \frac{x'}{d_x} + c_x = \frac{fX_c}{d_x Z_c} + c_x$
  - $y'' = \frac{y'}{d_y} + c_y = \frac{fY_c}{d_y Z_c} + c_y$
  - c_x, c_y : shift (0,0) of the image corner
  - d_x, d_y: scale x' and y' by physical dimension of pixel (d_x, d_y could be different)



Intrinsic Parameter

- $K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$
  - $f_x = \frac{f}{d_x}$
  - $f_y = \frac{f}{d_y}$
  - Internal parameters

Intrinsic Parameter

- $\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \sim K \begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix}$
- $\begin{bmatrix} \frac{F}{d_x} & 0 & c_x \\ 0 & \frac{F}{d_y} & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} \frac{F}{d_x} * X + c_x Z \\ \frac{F}{d_y} * Y + c_y Z \\ Z \end{bmatrix} \sim \begin{bmatrix} \frac{FX}{d_x Z} + c_x \\ \frac{FY}{d_y Z} + c_y \\ 1 \end{bmatrix}$

Key Takeaways

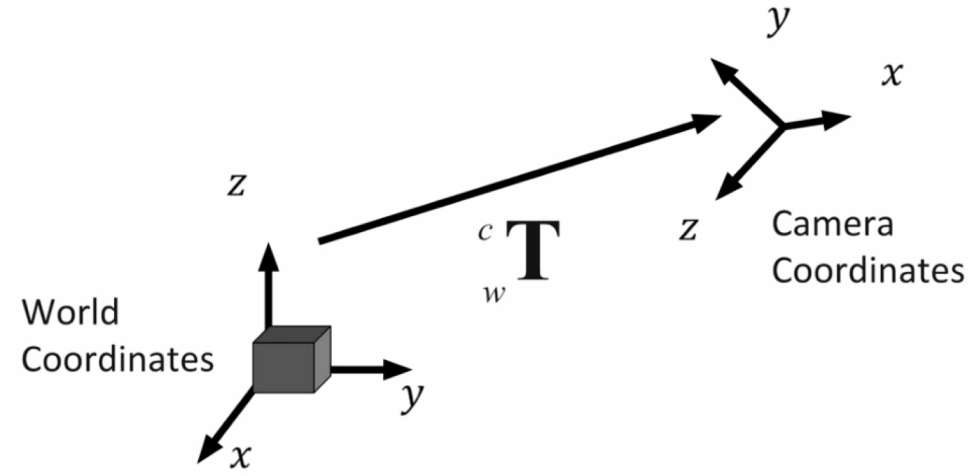
- Intrinsic parameters or camera matrix
  - $K = \begin{bmatrix} f_x & c & c_x \\ 0 & a * f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$
  - a : aspect ratio (we did not talk about this)
  - c: skew (we did not talk about this)
  - $f_x = \frac{f}{d_x}$
  - $f_y = \frac{f}{d_y}$
  - f is needed for the projection
  - d_x, d_y: unit conversion (change the unit in the image space to one pixel width/height)
  - c_x, c_y: move the origin from center to left upper corner

Outline

$$\mathbf{M} = \begin{bmatrix} fa & c & u_c \\ 0 & f & v_c \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{R}_{3x3} & \mathbf{0}_{3x1} \\ \mathbf{0}_{1x3} & 1 \end{bmatrix} \begin{bmatrix} \mathbf{I}_{3x3} & \mathbf{t}_{3x1} \\ \mathbf{0}_{1x3} & 1 \end{bmatrix}$$

intrinsic *projection* **rotation** **translation**

- Camera system
- Perspective images
- Intrinsic parameter
- **Extrinsic parameter:** convert a point from world coordinate system to camera coordinate system
- Camera Calibration



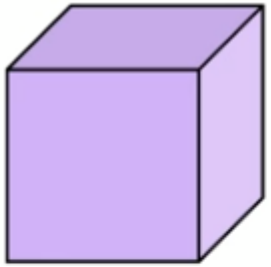
Practice

- How many degrees of freedom are there in specifying the extrinsic parameters?

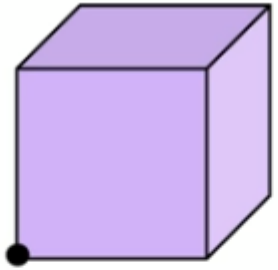
- (a) 5
- (b) 6
- (c) 3
- (d) 9

Rigid Body Transformation

- Need a way to specify the size degrees-of-freedom of a rigid body. Why 6?

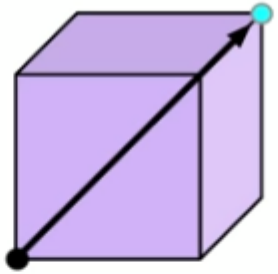


A rigid body is a collection of points whose positions relative to each other can't change



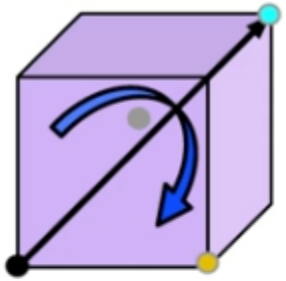
Fix one point, three DOF

3



Fix second point, two more DOF (must maintain distance constraint)

+2

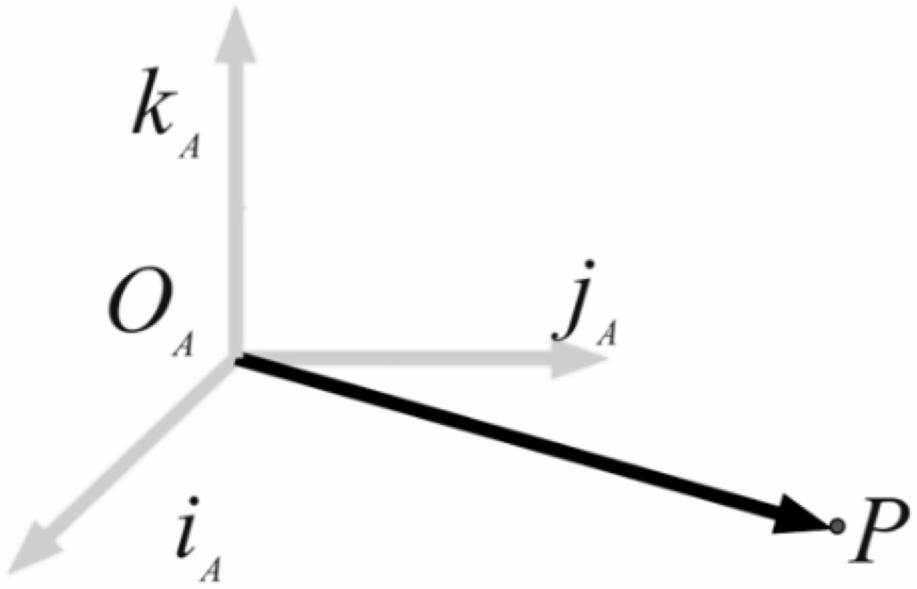


Third point adds one more DOF, for rotation around line

+1

Notation

- Superscript references coordinate frame
- ${}^{A}P$ is coordinates of P in frame A
- ${}^{B}P$ is coordinates of P in frame B



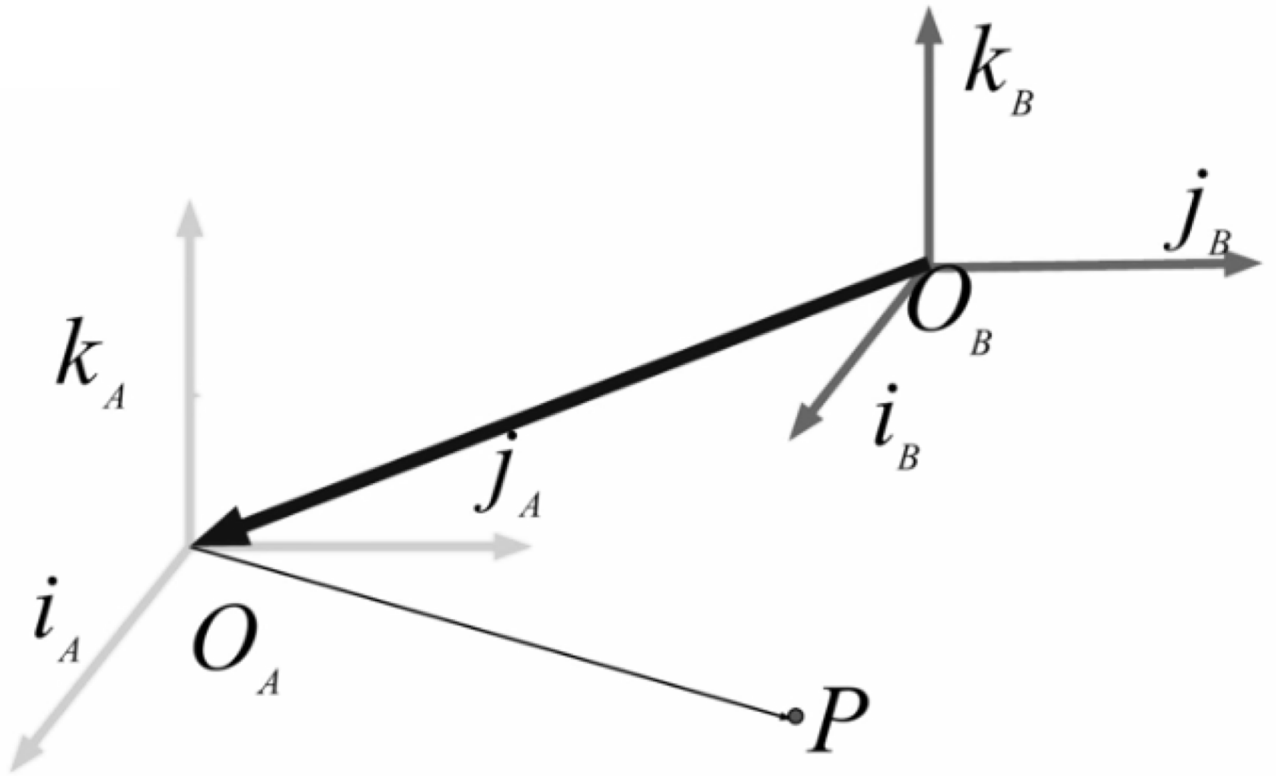
$$ {}^{A}P = \begin{pmatrix} {}^{A}x \\ {}^{A}y \\ {}^{A}z \end{pmatrix} \Leftrightarrow \overline{OP} = \left({}^{A}x \cdot \overline{i_A} \right) + \left({}^{A}y \cdot \overline{j_A} \right) + \left({}^{A}z \cdot \overline{k_A} \right) $$

Translation Only

$$^{B}P = {}^{A}P + {}^{B}(O_A)$$

or

$$^{B}P = {}^{B}(O_A) + {}^{A}P$$



Translation

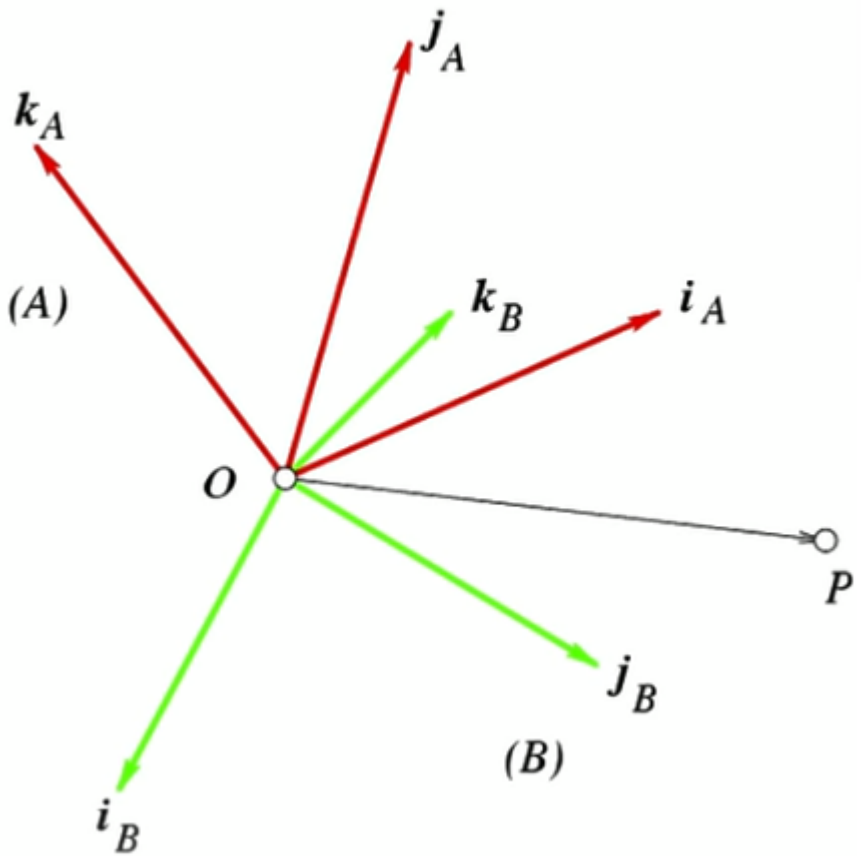
- Using homogeneous coordinates, translation can be expressed as a matrix multiplication
  - ${}^{B}P = {}^{A}P + {}^{B}O_{A}$

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- $\begin{pmatrix} {}^{B}P \\ 1 \end{pmatrix} = \begin{bmatrix} I & {}^{B}O_{A} \\ 0^{T} & 1 \end{bmatrix} \begin{bmatrix} {}^{A}P \\ 1 \end{bmatrix}$

$$[0 \quad 0 \quad 0]$$

Rotation



$$\overrightarrow{OP} = \begin{pmatrix} i_A & j_A & k_A \end{pmatrix} \begin{pmatrix} {}^{A}x \\ {}^{A}y \\ {}^{A}z \end{pmatrix} = \begin{pmatrix} i_B & j_B & k_B \end{pmatrix} \begin{pmatrix} {}^{B}x \\ {}^{B}y \\ {}^{B}z \end{pmatrix}$$

$$^{B}P = {}^{B}_{A}R\,{}^{A}P$$

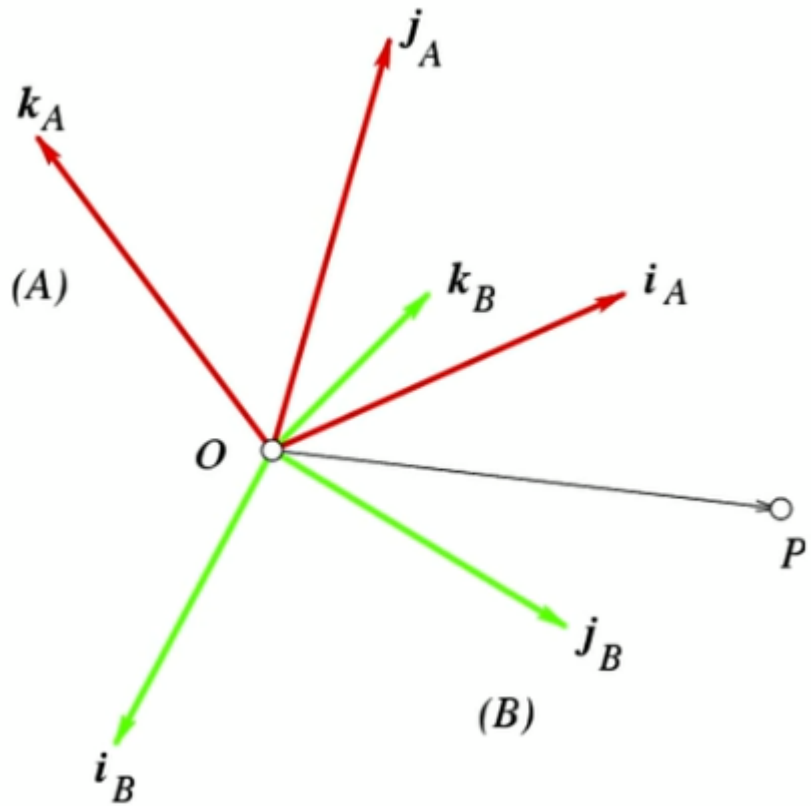
${}^{B}_{A}R$ *means describing frame A in the coordinate system of frame B*

Rotation Matrix

$$
{}^{B}_{A}R = \begin{bmatrix} \mathbf{i}_A.\mathbf{i}_B & \mathbf{j}_A.\mathbf{i}_B & \mathbf{k}_A.\mathbf{i}_B \\ \mathbf{i}_A.\mathbf{j}_B & \mathbf{j}_A.\mathbf{j}_B & \mathbf{k}_A.\mathbf{j}_B \\ \mathbf{i}_A.\mathbf{k}_B & \mathbf{j}_A.\mathbf{k}_B & \mathbf{k}_A.\mathbf{k}_B \end{bmatrix}
$$

$$
= \begin{bmatrix} {}^{B}\mathbf{i}_A & {}^{B}\mathbf{j}_A & {}^{B}\mathbf{k}_A \end{bmatrix}
$$

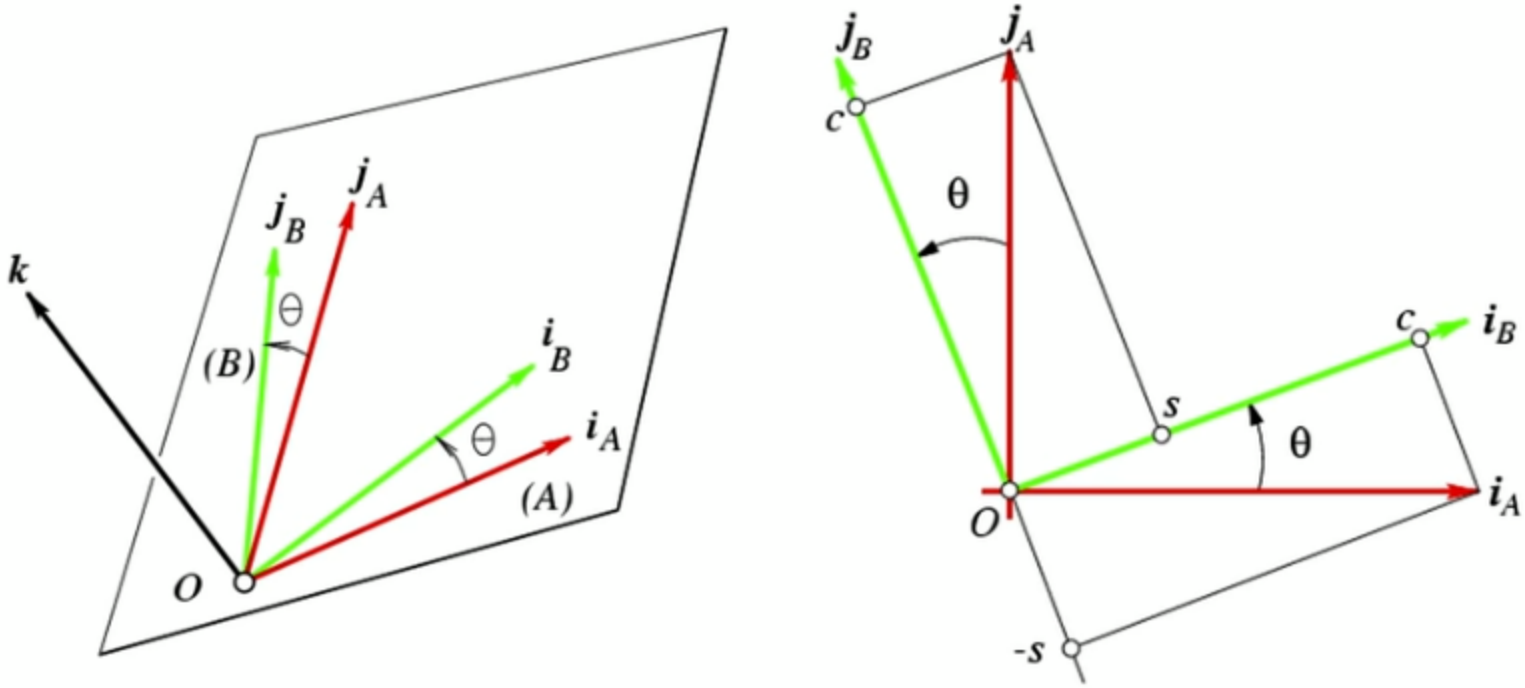
$$
= \begin{bmatrix} {}^{A}\mathbf{i}_B{}^{T} \\ {}^{A}\mathbf{j}_B{}^{T} \\ {}^{A}\mathbf{k}_B{}^{T} \end{bmatrix}
$$



Example: Rotation about Z-axis

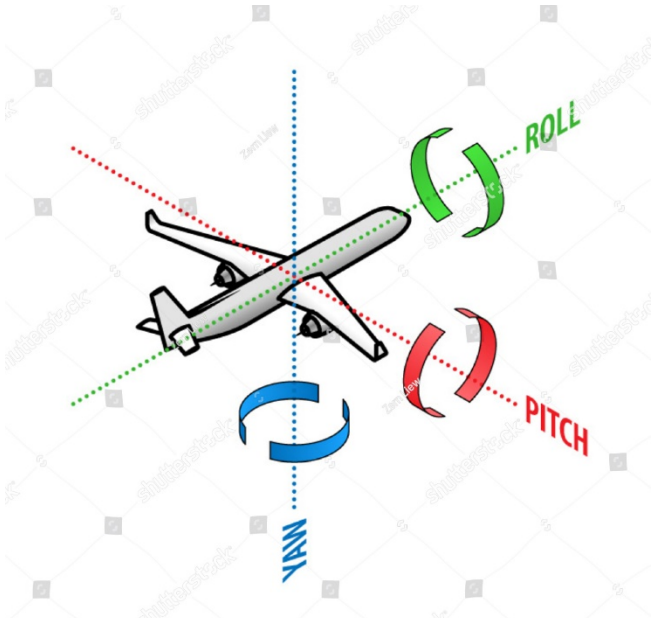
- Rotation matrix

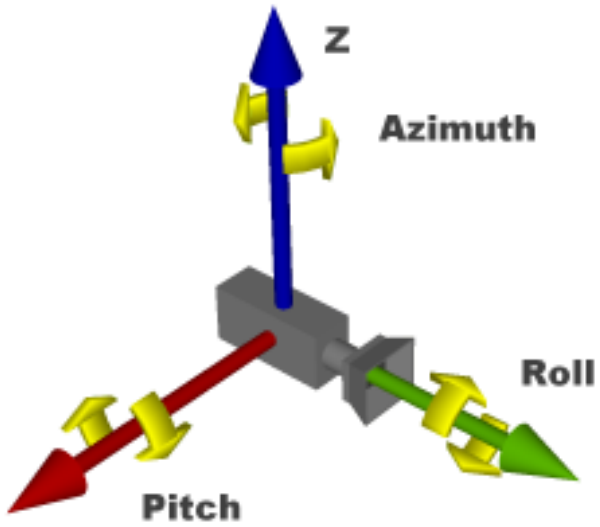
$$R_z(\theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



Arbitrary Rotation

- Combine 3 to get arbitrary rotation
- Pitch roll and yaw (or azimuth, elevation, roll)
- Three basic matrices: order matters





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

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

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

Rotation in Homogeneous Coordinates

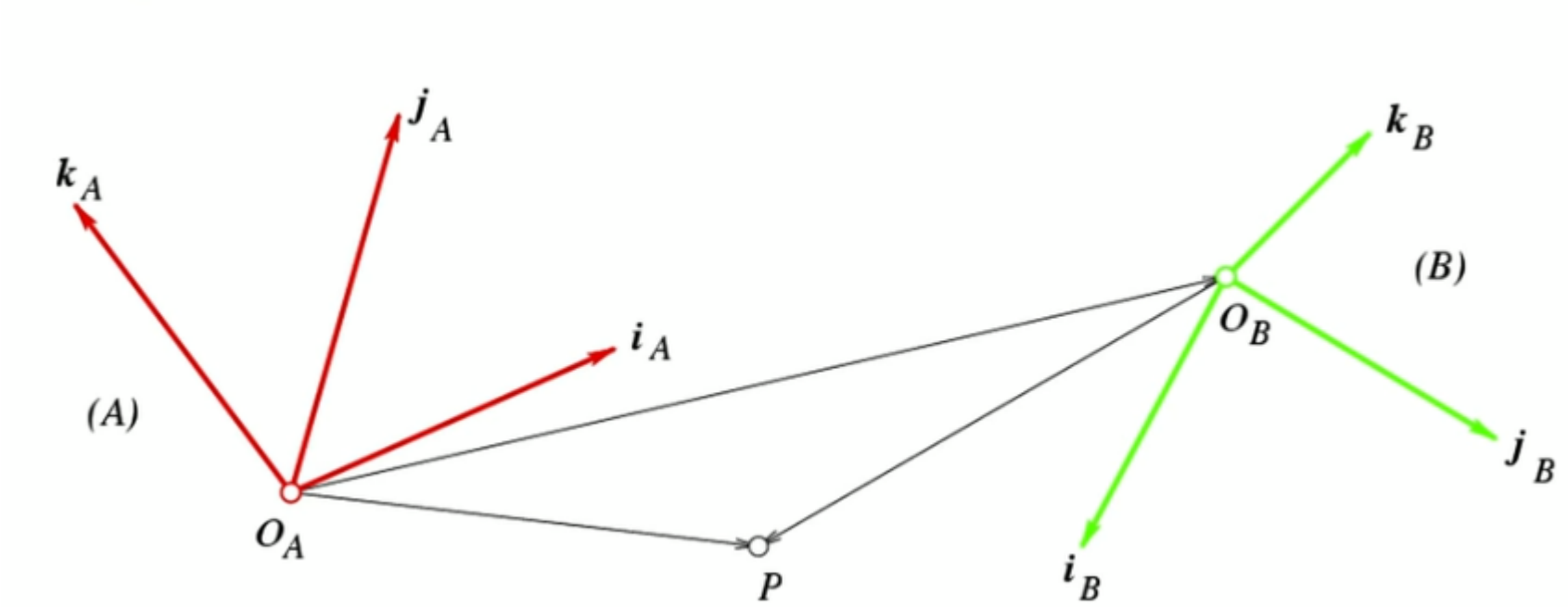
- Using homogeneous coordinates, rotation can be expressed as a matrix multiplication
  - Rotation is not commutative

$$ {}^{B}P = {}^{B}_{A}R \, {}^{A}P $$

$$ \begin{bmatrix} {}^{B}P \\ 1 \end{bmatrix} = \begin{bmatrix} {}^{B}_{A}R & 0 \\ \mathbf{0}^{T} & 1 \end{bmatrix} \begin{bmatrix} {}^{A}P \\ 1 \end{bmatrix} $$

Total Rigid Transformation

- Rotate then offset…



$$^{B}P = {}^{B}_{A}R\,{}^{A}P + {}^{B}O_{A}$$

Total Rigid Transformation

- Unified treatment using homogeneous coordinates

$$\begin{bmatrix} {}^{B}P \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & {}^{B}O_{A} \\ \mathbf{0}^{T} & 1 \end{bmatrix} \begin{bmatrix} {}^{B}_{A}R & 0 \\ \mathbf{0}^{T} & 1 \end{bmatrix} \begin{bmatrix} {}^{A}P \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} {}^{B}_{A}R & {}^{B}O_{A} \\ \mathbf{0}^{T} & 1 \end{bmatrix} \begin{bmatrix} {}^{A}P \\ 1 \end{bmatrix}$$

Total Rigid Transformation

- Transformation from A to B

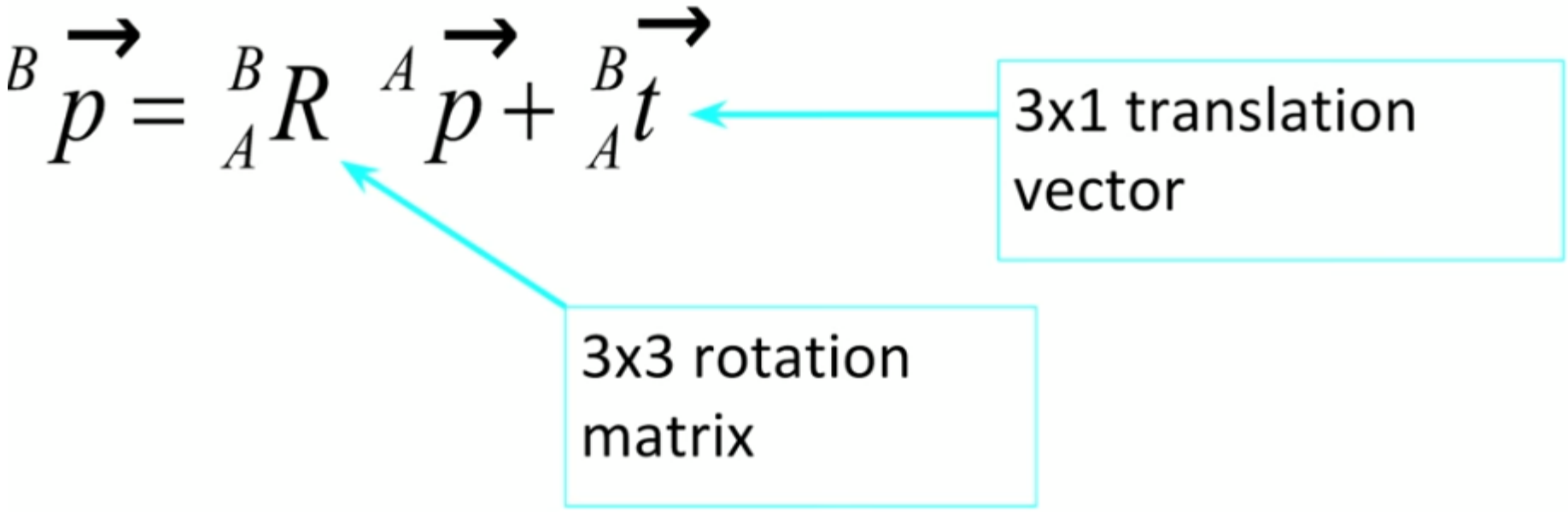
$$\begin{bmatrix} {}^{B}P \\ 1 \end{bmatrix} = \begin{bmatrix} {}^{B}_{A}R & {}^{B}O_{A} \\ \mathbf{0}^{T} & 1 \end{bmatrix} \begin{bmatrix} {}^{A}P \\ 1 \end{bmatrix} = {}^{B}_{A}T \begin{bmatrix} {}^{A}P \\ 1 \end{bmatrix}$$

- Transformation from B to A

$$\begin{bmatrix} {}^{A}P \\ 1 \end{bmatrix} = {}^{A}_{B}T \begin{bmatrix} {}^{B}P \\ 1 \end{bmatrix} = \left({}^{B}_{A}T \right)^{-1} \begin{bmatrix} {}^{B}P \\ 1 \end{bmatrix}$$

Translation and Rotation

- From frame A to B: non-homogeneous (“regular”) coordinate



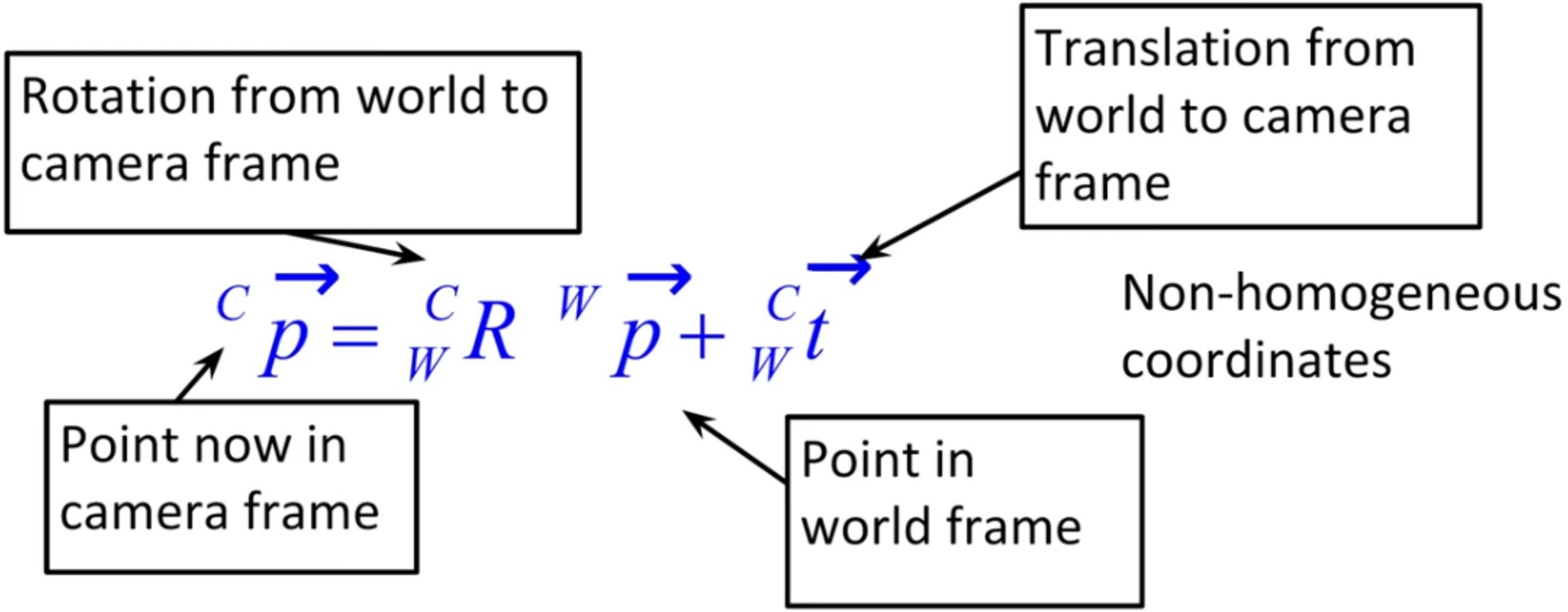
Translation and Rotation

- Homogeneous coordinates:
  - Allows us to write coordinate transforms as a single matrix

$$ {}^{B}\vec{p} = {}^{B}_{A}T \; {}^{A}\vec{p} $$

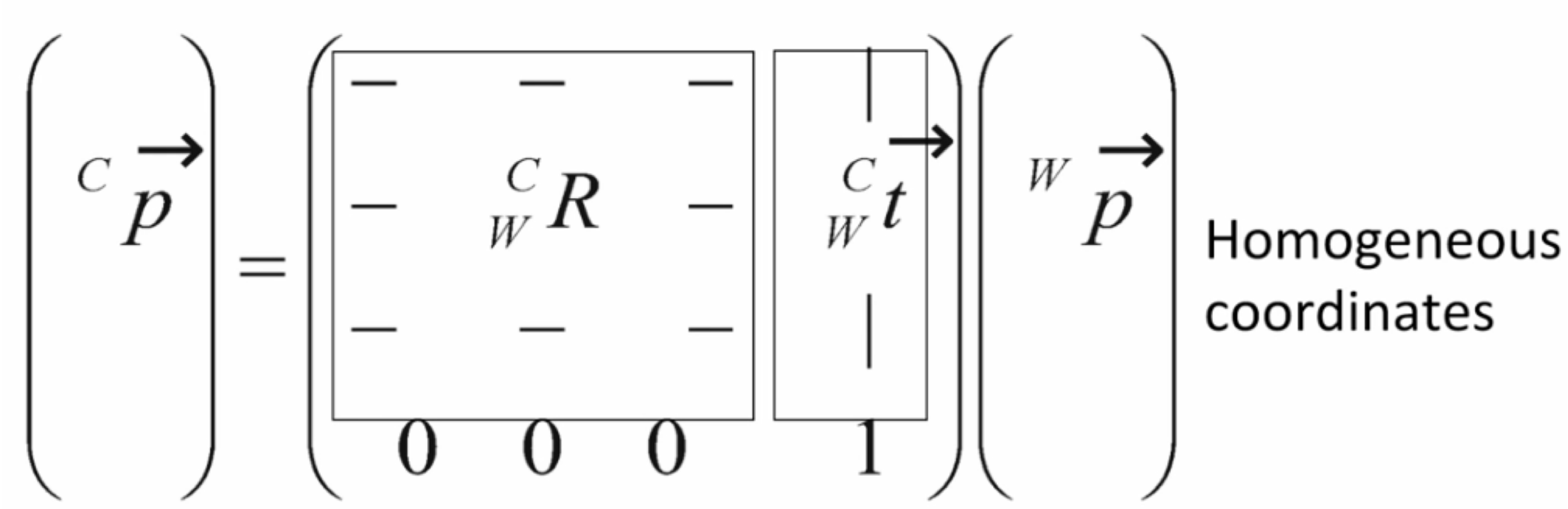
$$ {}^{B}\vec{p} = \begin{pmatrix} \begin{pmatrix} & {}^{B}_{A}R & \end{pmatrix} & \begin{matrix} | \\ {}^{B}_{A}\vec{t} \\ | \end{matrix} \\ 0 \quad 0 \quad 0 & 1 \end{pmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} $$

From World to Camera



From World to Camera

- From world to camera is the “extrinsic” parameter matrix (4x4)



Outline

- Camera system
- Perspective images
- Extrinsic parameter
- Intrinsic parameter
- **Camera Calibration**

Today: Calibration

- What are the camera parameters?

- Where are the light sources?
- What is the mapping from radiance to pixel color?

Why Calibrate?

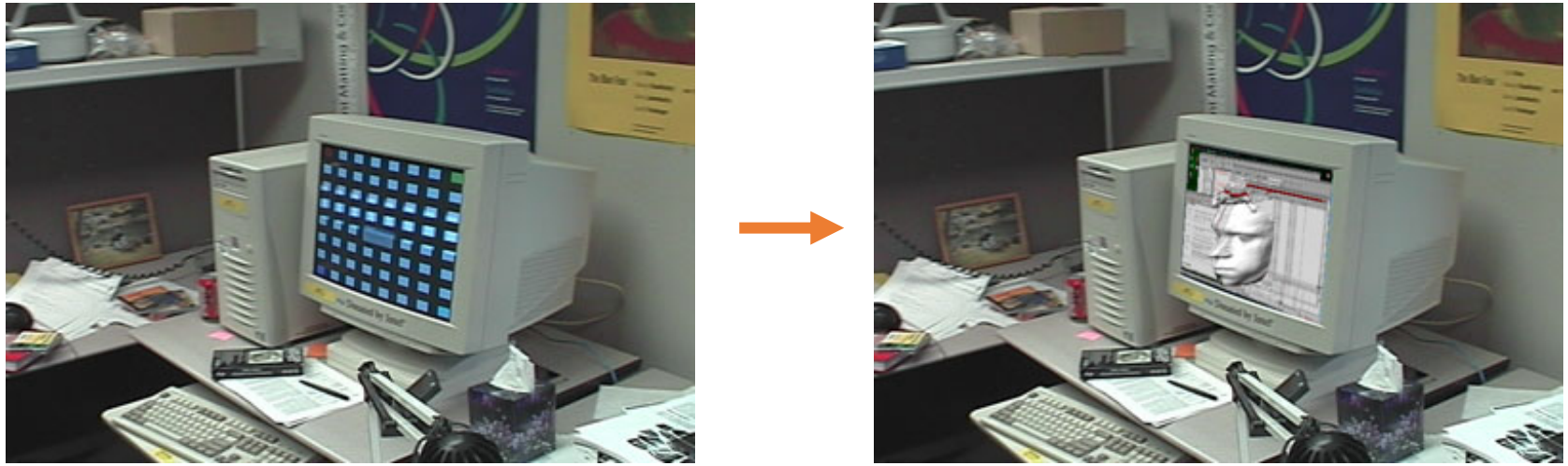
- Want to solve for 3D geometry

- Alternative approach
  - Solve for 3D shape without known cameras
    - Structure from motion (unknown extrinsics)
    - Self calibration (unknown intrinsics & extrinsics)

- Why pre-calibrating the camera?
  - Simplifies the 3D reconstruction problem
    - fewer parameters to solve for later on
  - Improves accuracy
  - Not too hard to do
  - Eliminates certain ambiguities (scale of scene)

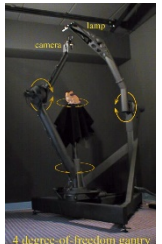
Applications

- 3D Modeling
- Match Move



Images courtesy of Brett Allen (“Vision for Graphics”, winter `01)

- Image-Based Rendering



Light field capture
and rendering
(Levoy & Hanrahan, 96)

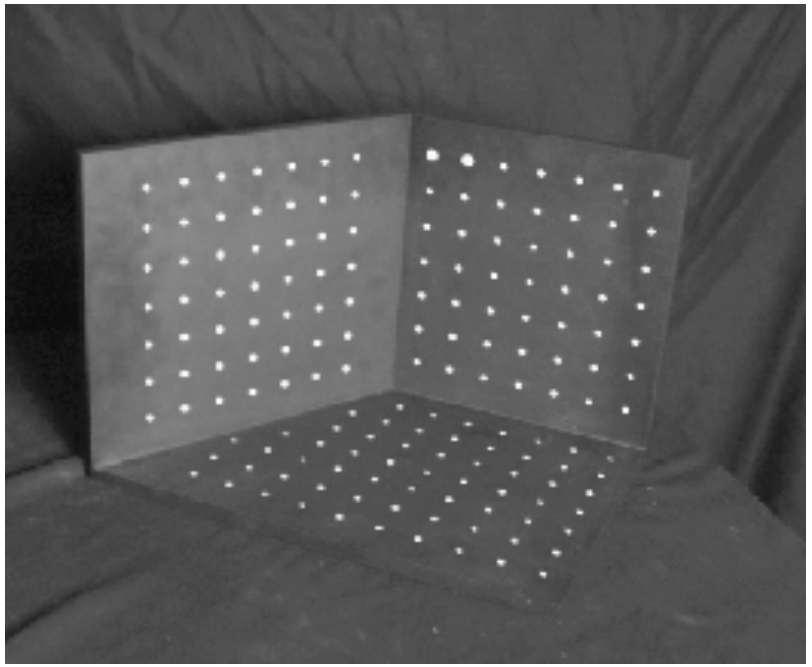
Camera Parameters

- Usually simplify to “computable stuff”
  - Intrinsics:
    - scale factor (“focal length”)
    - aspect ratio
    - principle point
    - radial distortion
  - Extrinsics
    - optical center
    - camera orientation
- How does this relate to projection matrix?

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

Goal of Calibration

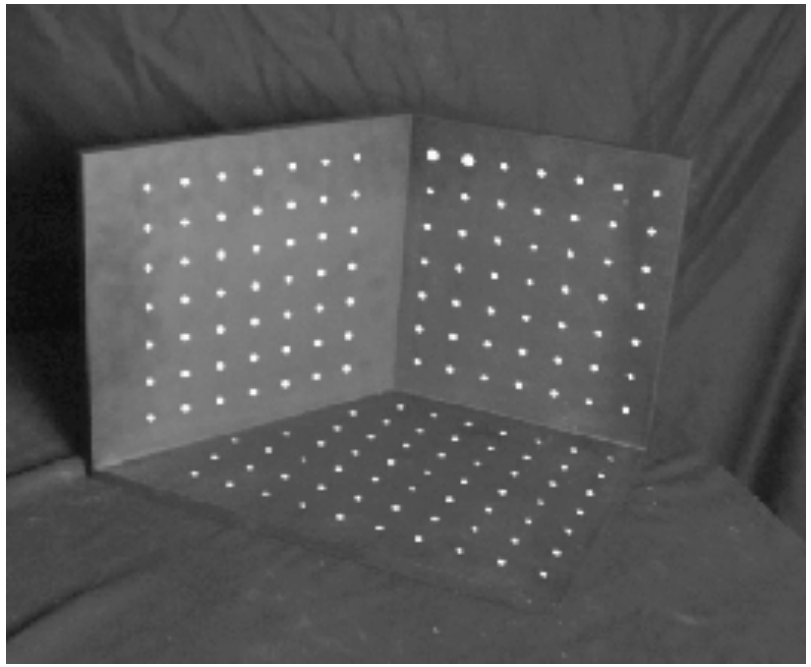
- Place a known object in the scene
  - identify correspondence between image and scene
  - compute mapping from scene to image



- Problem: must know geometry very accurately
  - how to get this info?

Estimate Projection Matrix

- Place a known object in the scene
  - identify correspondence between image and scene
  - compute mapping from scene to image



$$\begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} \sim \begin{bmatrix} m_{00} & m_{01} & m_{02} & m_{03} \\ m_{10} & m_{11} & m_{12} & m_{13} \\ m_{20} & m_{21} & m_{22} & 1 \end{bmatrix} \begin{bmatrix} X_i \\ Y_i \\ Z_i \\ 1 \end{bmatrix}$$

Direct Linear Calibration

$$\begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} \sim \begin{bmatrix} m_{00} & m_{01} & m_{02} & m_{03} \\ m_{10} & m_{11} & m_{12} & m_{13} \\ m_{20} & m_{21} & m_{22} & 1 \end{bmatrix} \begin{bmatrix} X_i \\ Y_i \\ Z_i \\ 1 \end{bmatrix}$$

$$u_i = \frac{m_{00}X_i + m_{01}Y_i + m_{02}Z_i + m_{03}}{m_{20}X_i + m_{21}Y_i + m_{22}Z_i + 1}$$

$$v_i = \frac{m_{10}X_i + m_{11}Y_i + m_{12}Z_i + m_{13}}{m_{20}X_i + m_{21}Y_i + m_{22}Z_i + 1}$$

$$u_i(m_{20}X_i + m_{21}Y_i + m_{22}Z_i + 1) = m_{00}X_i + m_{01}Y_i + m_{02}Z_i + m_{03}$$
$$v_i(m_{20}X_i + m_{21}Y_i + m_{22}Z_i + 1) = m_{10}X_i + m_{11}Y_i + m_{12}Z_i + m_{13}$$

$$\begin{bmatrix} X_i & Y_i & Z_i & 1 & 0 & 0 & 0 & 0 & -u_iX_i & -u_iY_i & -u_iZ_i \\ 0 & 0 & 0 & 0 & X_i & Y_i & Z_i & 1 & -v_iX_i & -v_iY_i & -v_iZ_i \end{bmatrix} \begin{bmatrix} m_{00} \\ m_{01} \\ m_{02} \\ m_{03} \\ m_{10} \\ m_{11} \\ m_{12} \\ m_{13} \\ m_{20} \\ m_{21} \\ m_{22} \end{bmatrix} = \begin{bmatrix} u_i \\ v_i \end{bmatrix}$$

Direct Linear Calibration

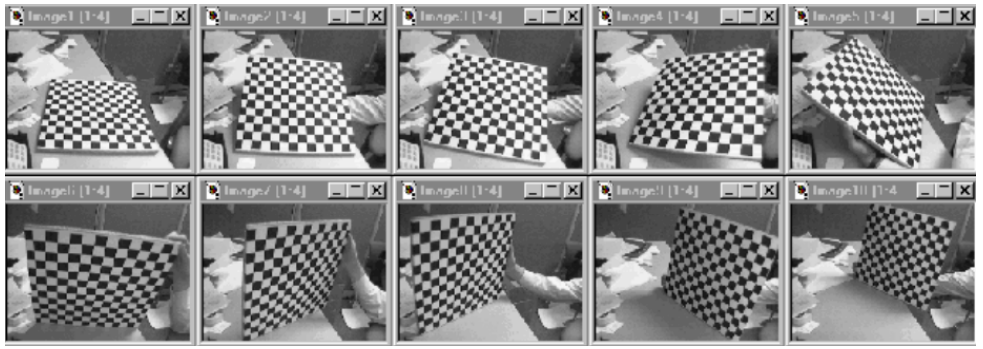
$$\begin{bmatrix} X_1 & Y_1 & Z_1 & 1 & 0 & 0 & 0 & 0 & u_iX_1 & u_iY_1 & u_iZ_i \\ 0 & 0 & 0 & 0 & X_1 & Y_1 & Z_1 & 1 & v_1X_1 & v_1Y_1 & v_1Z_1 \\ & & & & & & \vdots & & & & \\ X_n & Y_n & Z_n & 1 & 0 & 0 & 0 & 0 & u_nX_n & u_nY_n & u_nZ_n \\ 0 & 0 & 0 & 0 & X_n & Y_n & Z_n & 1 & v_nX_n & v_nY_n & v_nZ_n \end{bmatrix} \begin{bmatrix} m_{00} \\ m_{01} \\ m_{02} \\ m_{03} \\ m_{10} \\ m_{11} \\ m_{12} \\ m_{13} \\ m_{20} \\ m_{21} \\ m_{22} \end{bmatrix} = \begin{bmatrix} u_1 \\ v_1 \\ \vdots \\ u_n \\ v_n \end{bmatrix}$$

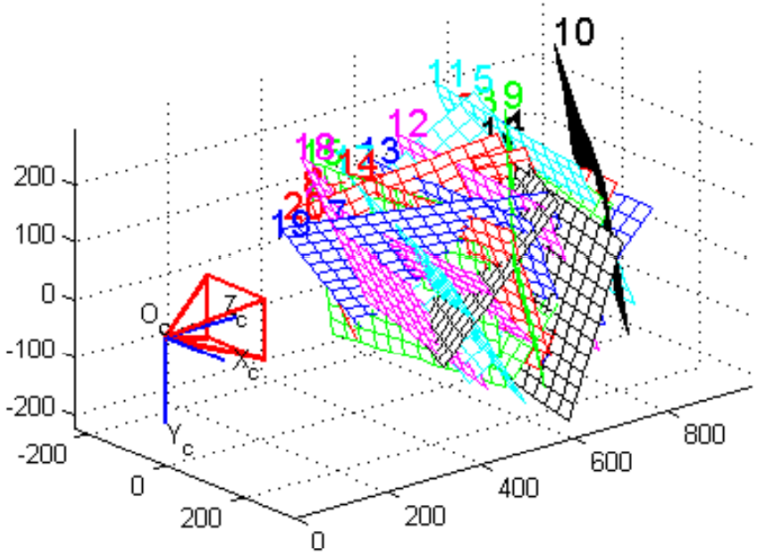
Can solve for m_{ij} by linear least squares

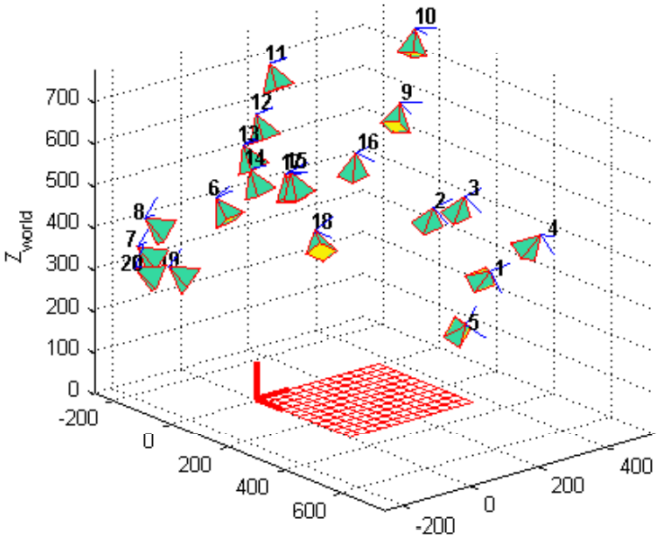
$$minimize \left\| \begin{bmatrix} X_1 & Y_1 & Z_1 & 1 & 0 & 0 & 0 & 0 & u_iX_1 & u_iY_1 & u_iZ_i \\ 0 & 0 & 0 & 0 & X_1 & Y_1 & Z_1 & 1 & v_1X_1 & v_1Y_1 & v_1Z_1 \\ & & & & & & \vdots & & & & \\ X_n & Y_n & Z_n & 1 & 0 & 0 & 0 & 0 & u_nX_n & u_nY_n & u_nZ_n \\ 0 & 0 & 0 & 0 & X_n & Y_n & Z_n & 1 & v_nX_n & v_nY_n & v_nZ_n \end{bmatrix} \begin{bmatrix} m_{00} \\ m_{01} \\ m_{02} \\ m_{03} \\ m_{10} \\ m_{11} \\ m_{12} \\ m_{13} \\ m_{20} \\ m_{21} \\ m_{22} \end{bmatrix} - \begin{bmatrix} u_1 \\ v_1 \\ \vdots \\ u_n \\ v_n \end{bmatrix} \right\|$$

Multi-plane Calibration (Internal parameter)

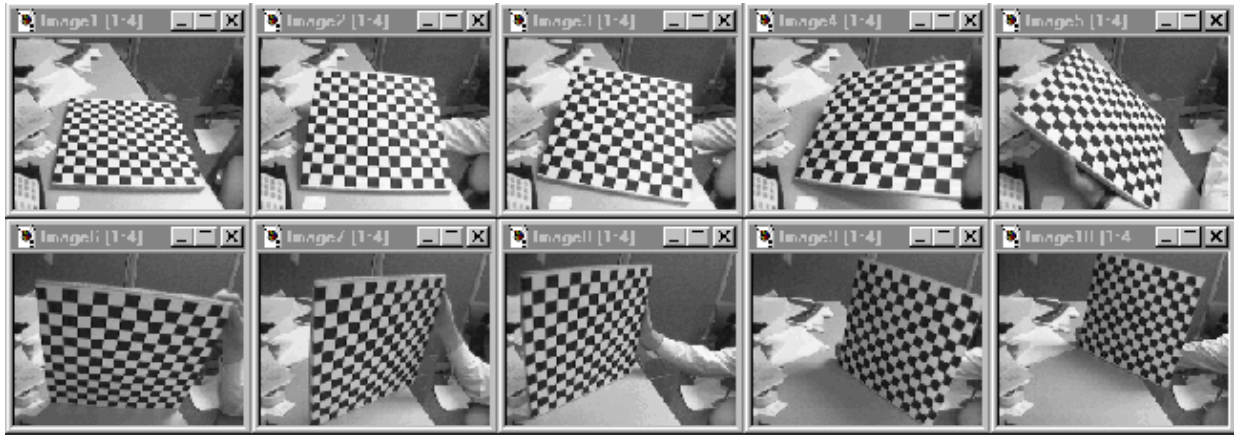
- For complicated task in CV, it is required that a camera be calibrated
  - 3D vision
- Use pin-hole camera model
- Separate intrinsics / extrinsics







Alternative: Multi-plane calibration

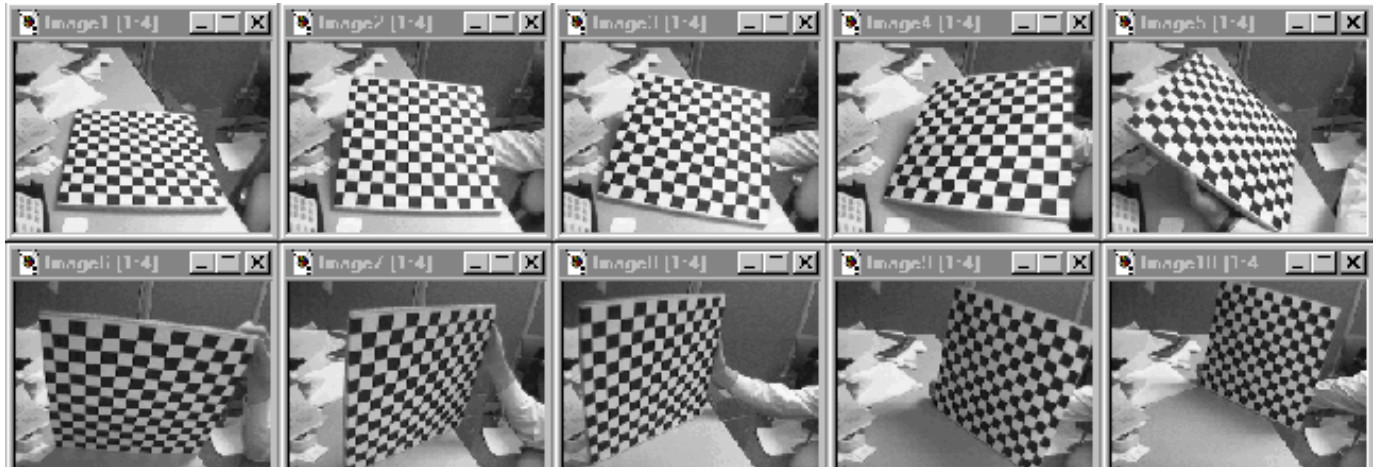


Images courtesy Jean-Yves Bouguet, Intel Corp.

Advantage

- Only requires a plane
- Don’t have to know positions/orientations
- Good code available online!
  - Zhengyou Zhang’s web site: http://research.microsoft.com/~zhang/Calib/
  - Intel’s OpenCV library: http://www.intel.com/research/mrl/research/opencv/
  - Matlab version by Jean-Yves Bouget: http://www.vision.caltech.edu/bouguetj/calib_doc/index.html

Alternative: Multi-plane calibration

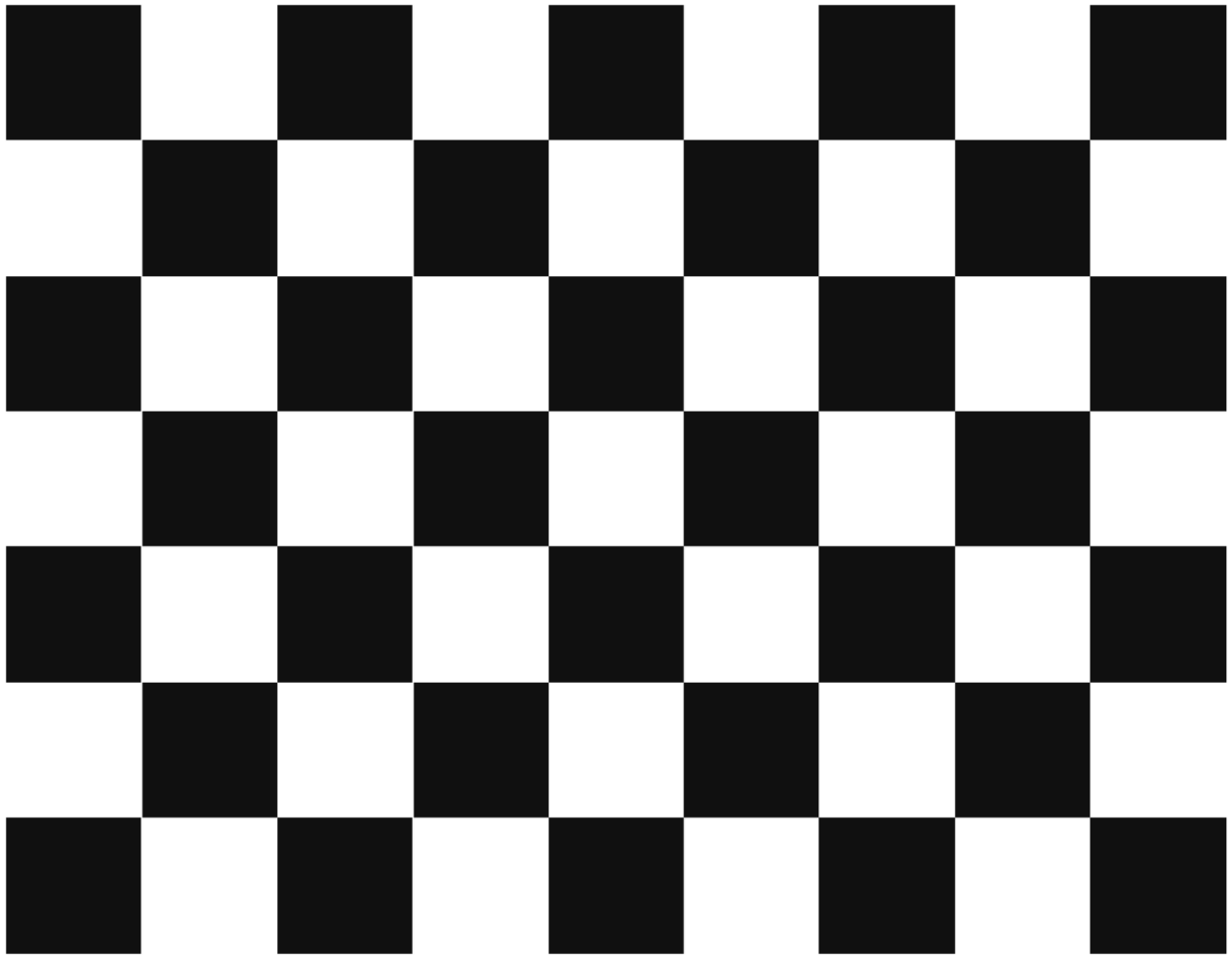


Images courtesy Jean-Yves Bouguet, Intel Corp.

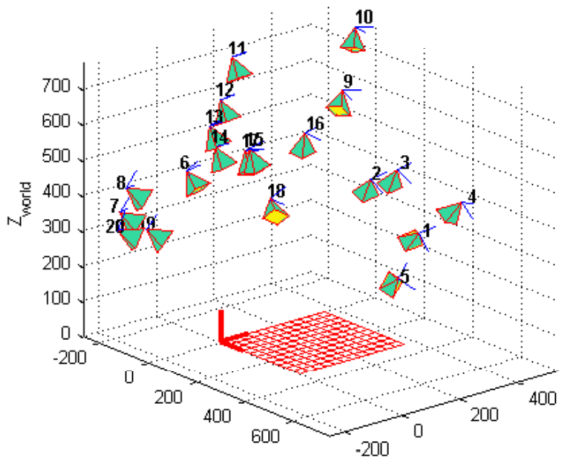
Need 3D -> 2D correspondence

- User provided (lots ‘O clicking)
- User seeded (some clicking)
- Fully automatic?

Checkerboard for Camera Calibration



Calibration from (unknown) Planes



- Assume that the plane is fixed at z=0

- $$s\begin{bmatrix}x\\y\\1\end{bmatrix} \sim K_{3*3}[R \mid t]_{3*3}\begin{bmatrix}X\\Y\\0\\1\end{bmatrix} = K[r_1\ r_2\ r_3\ t]\begin{bmatrix}X\\Y\\0\\1\end{bmatrix} = K[r_1\ r_2\ t]\begin{bmatrix}X\\Y\\1\end{bmatrix} = H_{3*3}\begin{bmatrix}x\\y\\1\end{bmatrix} = [h_1\ h_2\ h_3]\begin{bmatrix}x\\y\\1\end{bmatrix}$$

- We assume H is known, here (it can be solved, just we do not discuss the details here)
  - By given the grid size, the checkerboard plane is only defined by 6 unknowns. We have a lot of mapping between 3D points and 2D points (corner)

- How to separate intrinsic (K) and extrinsic ($[r_1\ r_2\ t]$)?

Calibration from (unknown) Planes

- For every camera position, we have a projective transformation 3x3 matrix
  - $H_i = K[R_i\ t_i] = K\left[r_1^i\ r_2^i\ t^i\right]$
  - $\left[r_1^i\ r_2^i\ t^i\right] = \mathrm{K}^{-1}[\mathrm{h}_1^{\mathrm{i}}\ \mathrm{h}_2^{\mathrm{i}}\ \mathrm{h}_3^{\mathrm{i}}]$
  - $\left\|r_j^i\right\| = 1$ and $\left(r_1^i\right)^T\left(r_2^i\right) = 0$
  - 2 constraints: $\left(r_1^i\right)^T\left(r_1^i\right) = \left(r_2^i\right)^T\left(r_2^i\right)$ and $\left(r_1^i\right)^T\left(r_2^i\right) = 0$

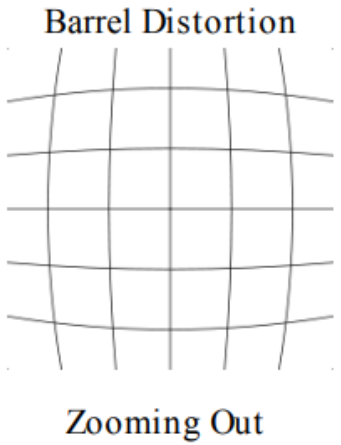
  - *K* is fixed for the same camera
  - $H_i = [R_i\ t_i]$ will be changed if you change the camera position in the world

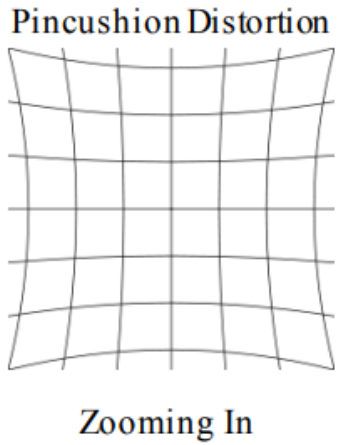
Calibration from (unknown) Planes

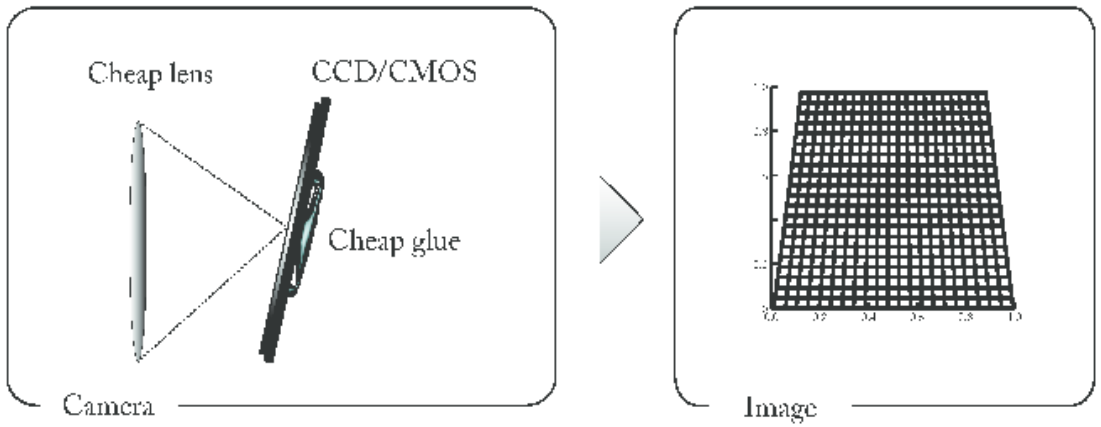
- First equation:
  - $\left(h_1^i\right)^T (KK^T)^{-1} (h_1^i) = \left(h_2^i\right)^T (KK^T)^{-1} (h_2^i)$
- Second equation:
  - $\left(h_1^i\right)^T (KK^T)^{-1} \left(h_2^i\right) = 0$
- $(KK^T)^{-1} = \left(\begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}^T\right)^{-1}$
  - 2 Equations, 4 or (5) unknowns
  - We can increase the number of equations by taking more checkerboard images. K is the intrinsic parameter, we only have 5 unknowns

Radial Distortion

- **Radial distortion** and tangential distortion
- Radial distortion: spherical shape of the lens
- tangential distortion: physical elements in a lens not being perfectly aligned





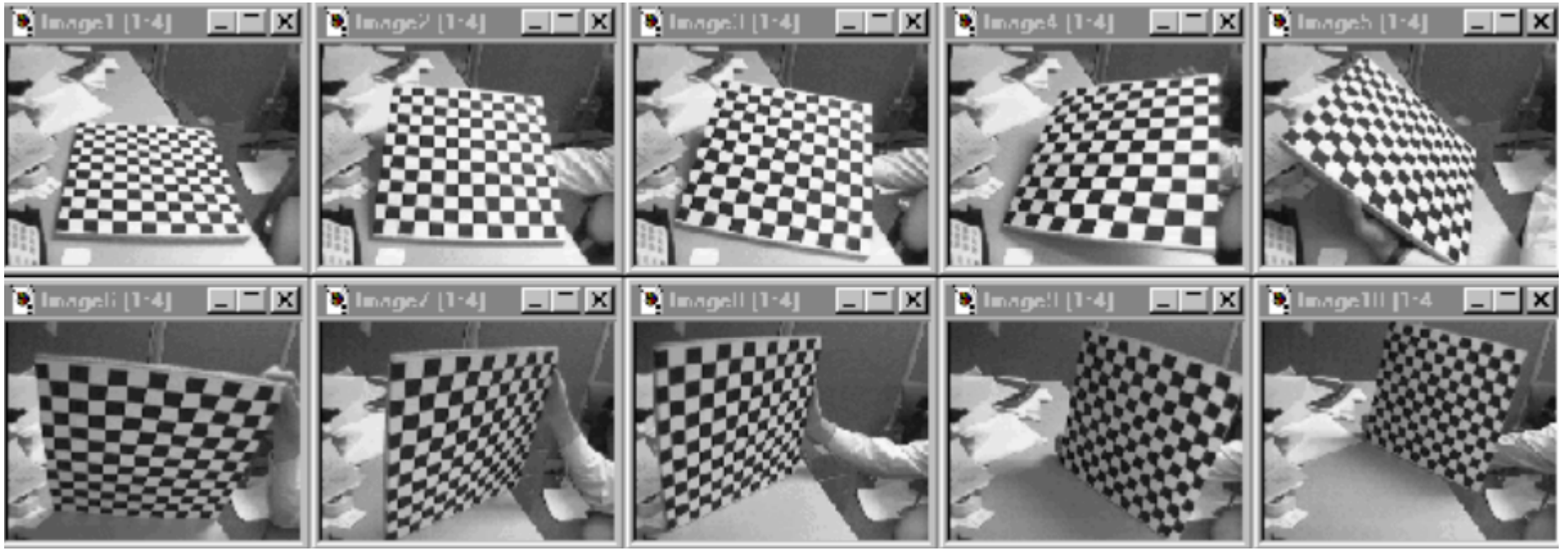


Radial Distortion

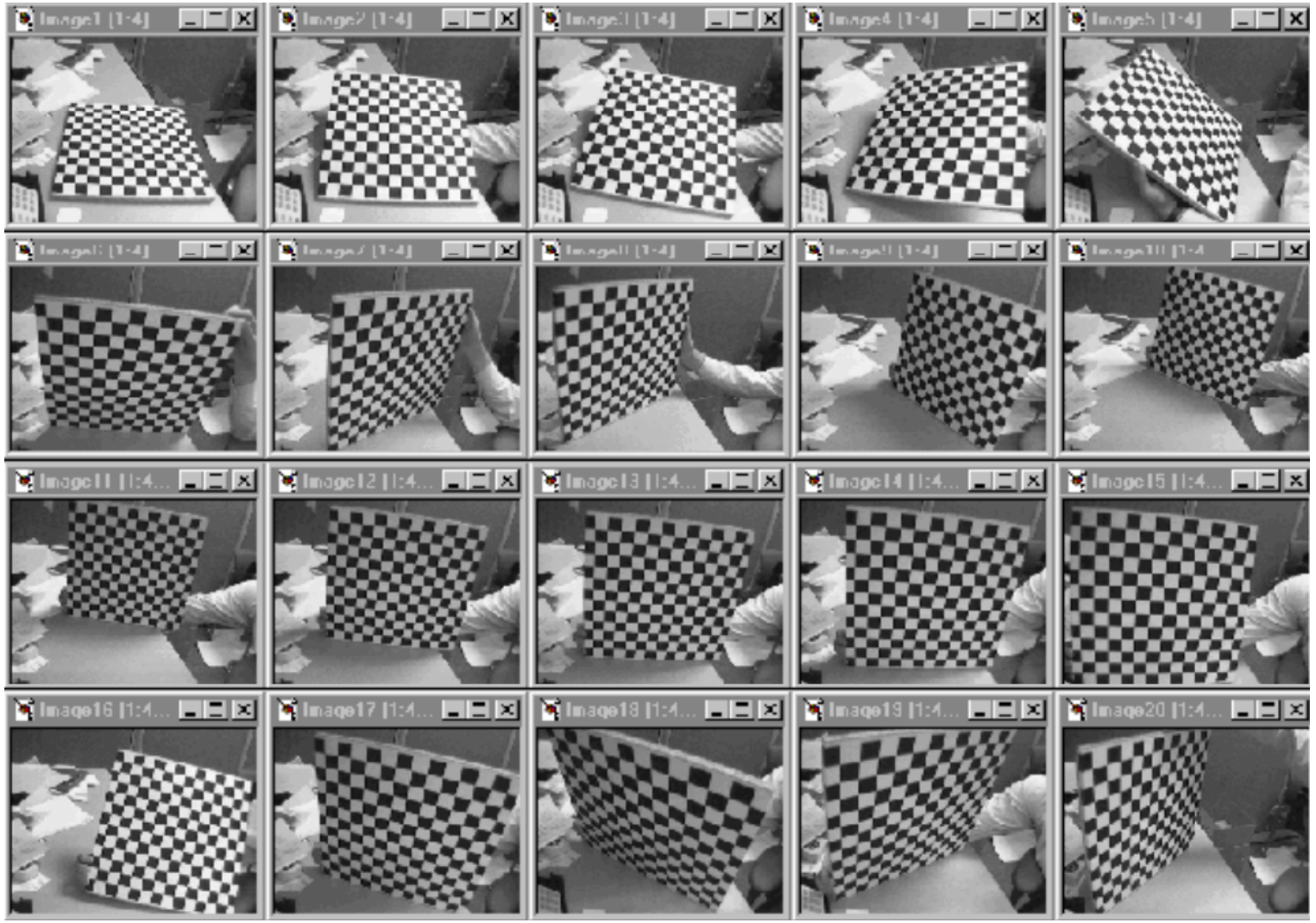
- Remove distortion by correcting image measurements to those that would have been obtained under perfect linear camera action
- Radial distortion
  - $\begin{bmatrix} x'_{dist} \\ y'_{dist} \end{bmatrix} = (1 + k_1 r + k_2 r^2 + k_3 r^3) \begin{bmatrix} x'_{un} \\ y'_{un} \end{bmatrix}$ (from Tylor expansion)
    - $r = \sqrt{{x'_{un}}^2 + {y'_{un}}^2}$
    - $\begin{bmatrix} x'_{un} \\ y'_{un} \end{bmatrix}$: ideal point
    - $\begin{bmatrix} x'_{dist} \\ y'_{dist} \end{bmatrix}$: point with distortion

Overview of the Process

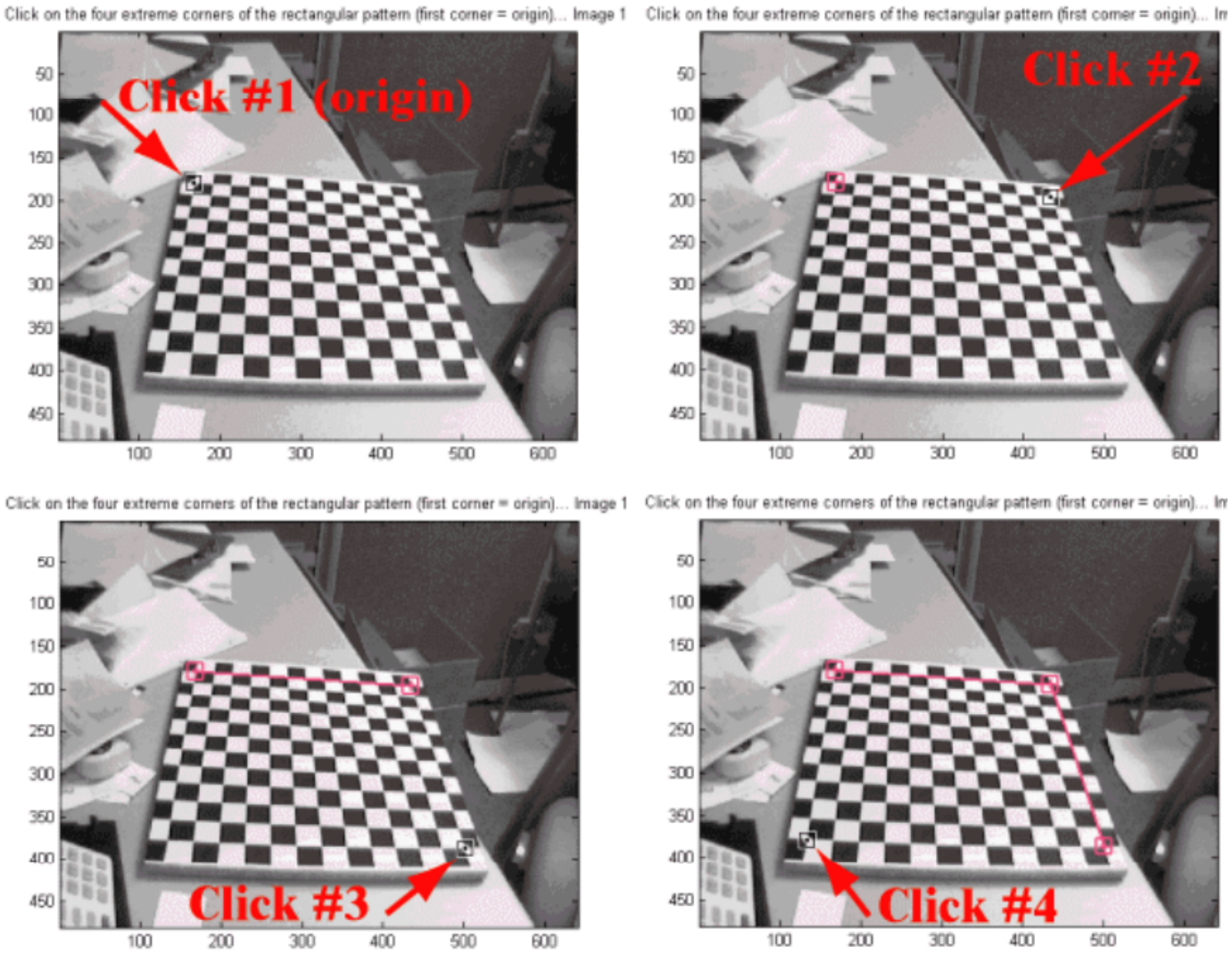
- Only require a plan with known geometric structure
- Don’t have to know position/orientation
- A lot of libraries to use



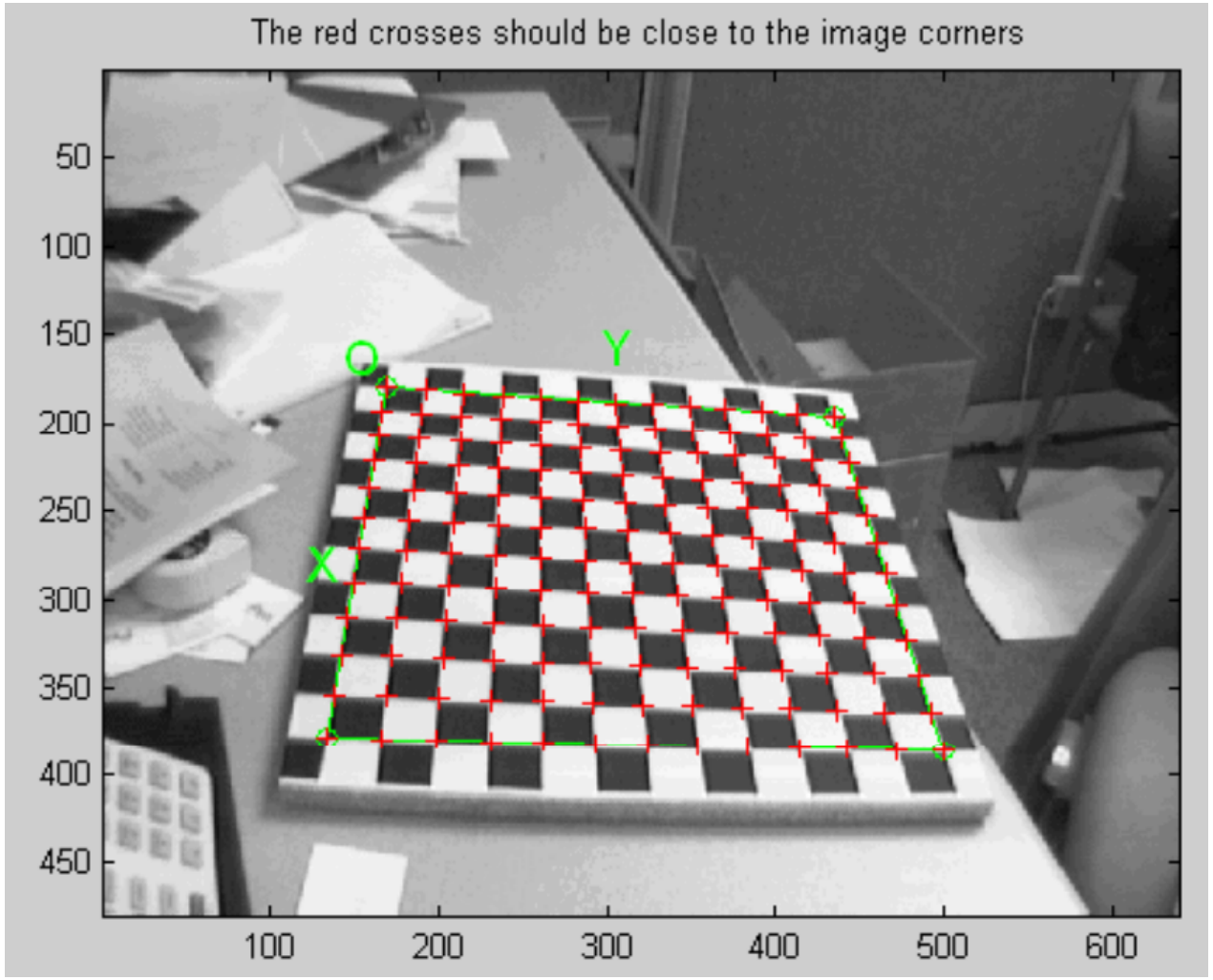
Step1: Checkerboard Images

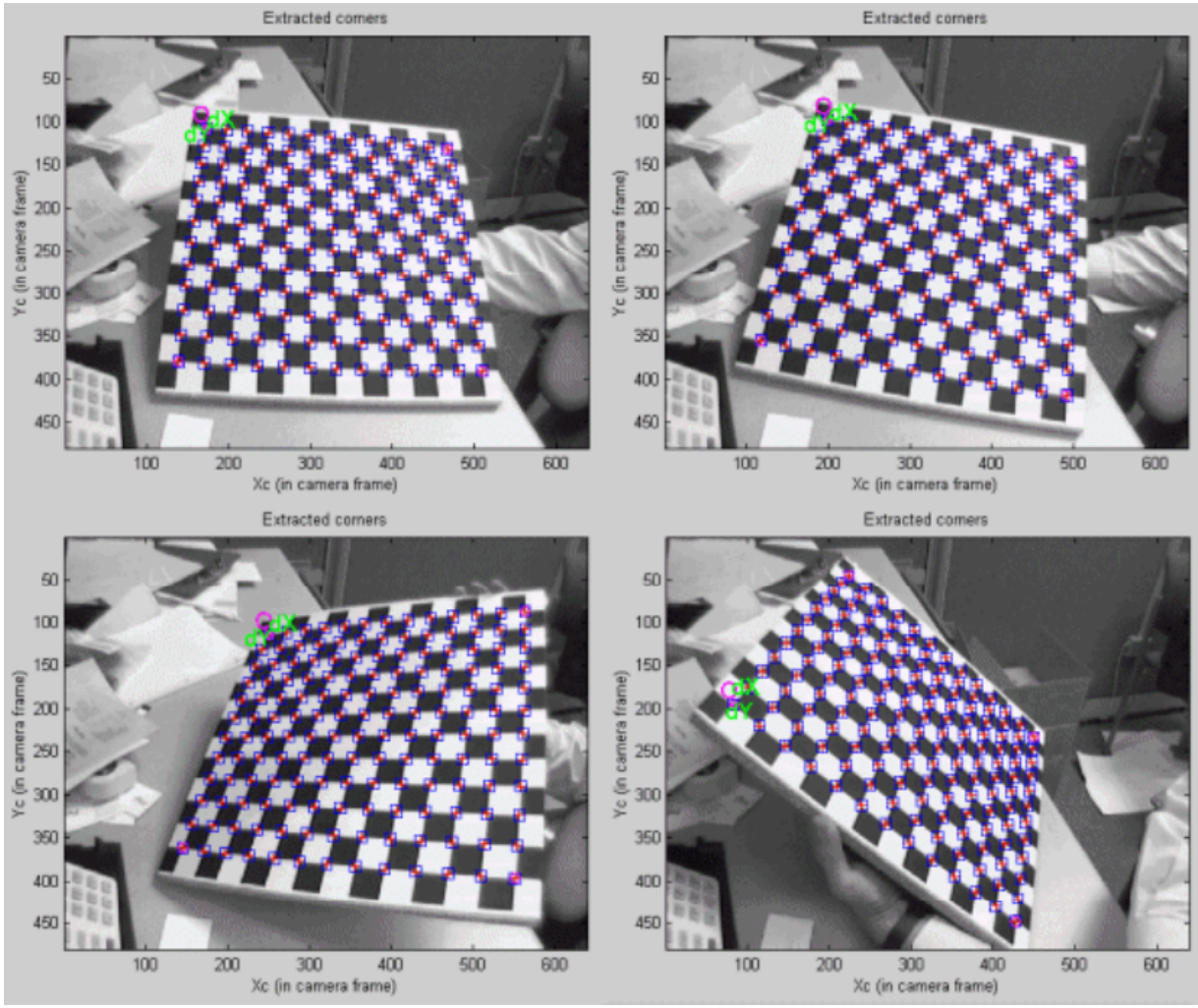


Step 2: Specify Corner Order



Step3: Corner Extraction





Result

```
Camera Matrix:
[[  3.07720341e+03   0.00000000e+00   1.98637884e+03]
 [  0.00000000e+00   3.07770770e+03   1.49526450e+03]
 [  0.00000000e+00   0.00000000e+00   1.00000000e+00]]
Distortion
[[  2.97959558e-01  -1.53709028e+00  -1.62236989e-03   5.30990672e-04
    2.48635337e+00]]
```