

# Computer Vision

## Projection Matrix and Camera Calibration

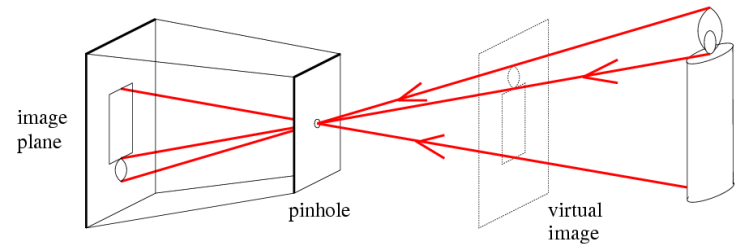
**Dr. Mrinmoy Ghorai**

**Indian Institute of Information Technology**  
**Sri City, Chittoor**

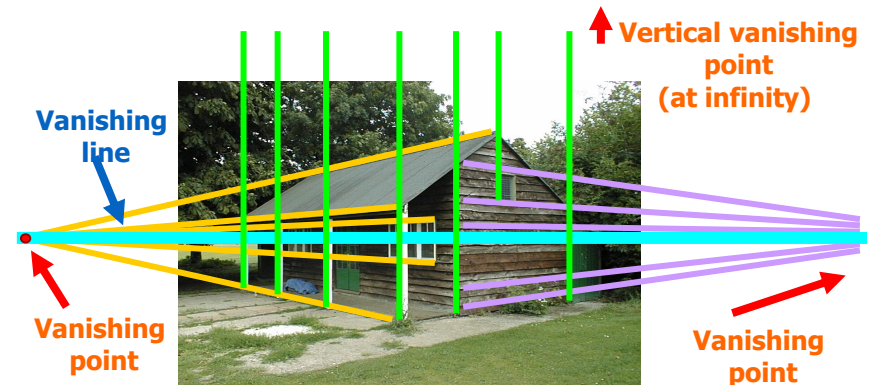


# Last Class

- Pinhole camera model
- Homogeneous coordinates
- Vanishing points and vanishing lines



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



# Perspective and 3D Geometry

- **Camera models and Projective geometry**

- What's the **mapping between image and world** coordinates?

- **Projection Matrix and Camera calibration**

- What's the **projection matrix** between scene and image coordinates?
- How to **calibrate** the projection matrix?

- **Single view metrology and Camera properties**

- How can we measure the **size of 3D objects** in an image?
- What are the important **camera properties**?

- **Photo stitching**

- What's the **mapping from two images** taken **without camera translation**?

- **Epipolar Geometry and Stereo Vision**

- What's the **mapping from two images** taken **with camera translation**?

- **Structure from motion**

- How can we **recover 3D points from multiple images**?

# This class

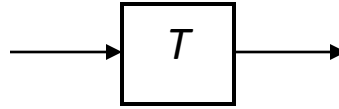
- Common Transformations
- What is the relation between scene point and image point in form of matrix?
- How can we calibrate the camera?

# Common Transformations

# Parametric (global) warping



$$\mathbf{p} = (x, y)$$



$$\mathbf{p}' = (x', y')$$

Transformation  $T$  is a coordinate-changing machine:

$$\mathbf{p}' = T(\mathbf{p})$$

For linear transformations, we can represent  $T$  as a matrix

$$\mathbf{p}' = \mathbf{T}\mathbf{p}$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \mathbf{T} \begin{bmatrix} x \\ y \end{bmatrix}$$

# Common Transformations



original

## Transformed



translation



rotation



aspect



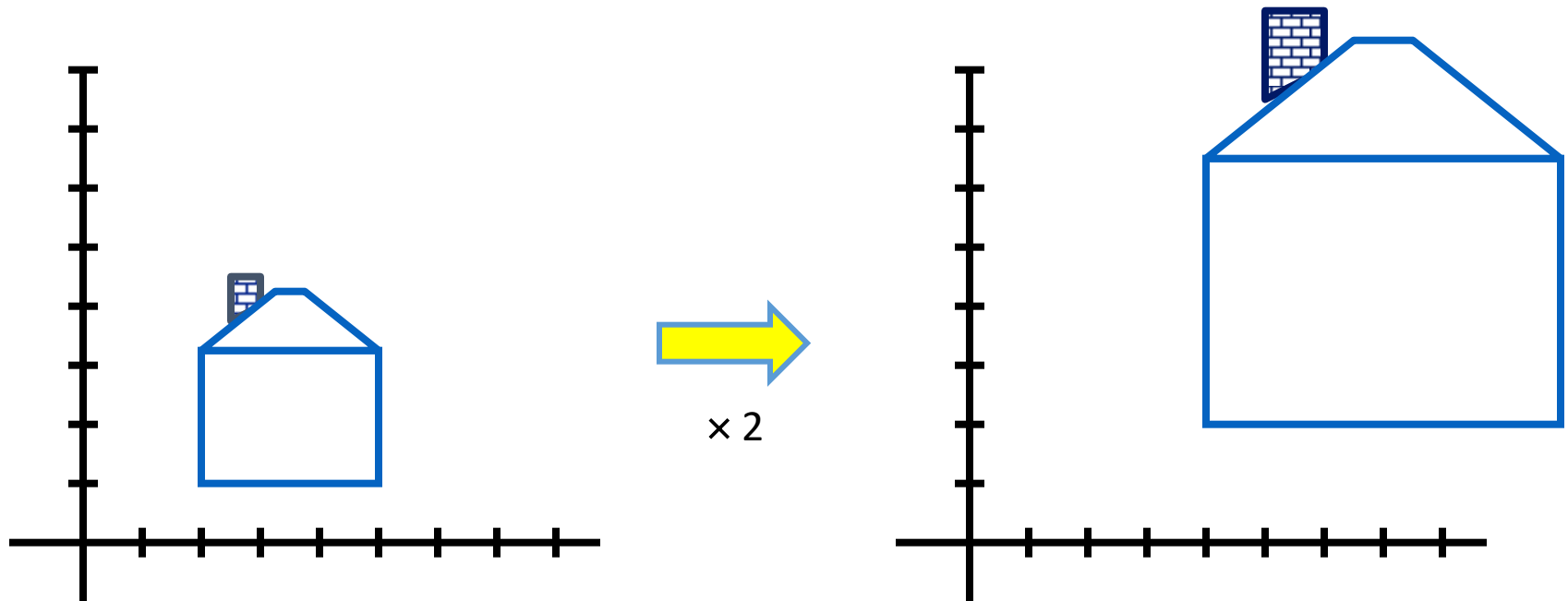
affine



perspective

# Scaling

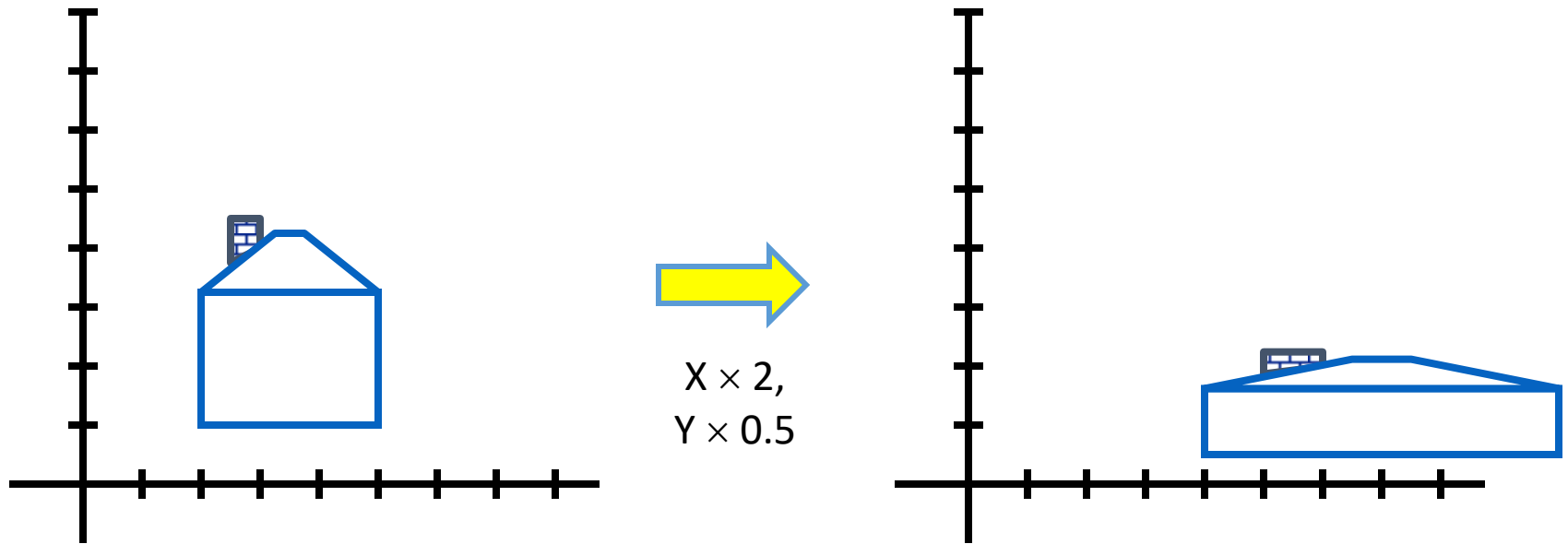
- *Scaling* a coordinate means multiplying each of its components by a scalar
- *Uniform scaling* means this scalar is the same for all components:





# Scaling

- *Non-uniform scaling*: different scalars per component:



# Scaling

- Scaling operation:

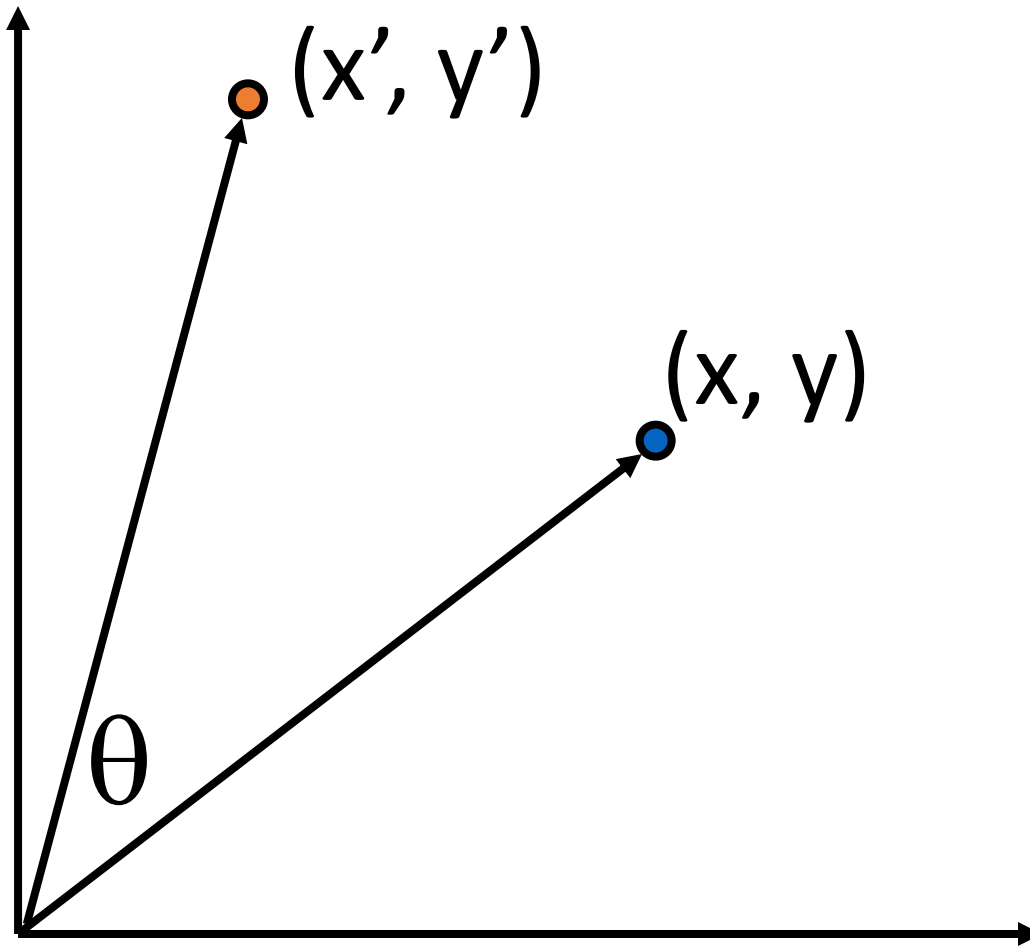
$$x' = ax$$

$$y' = by$$

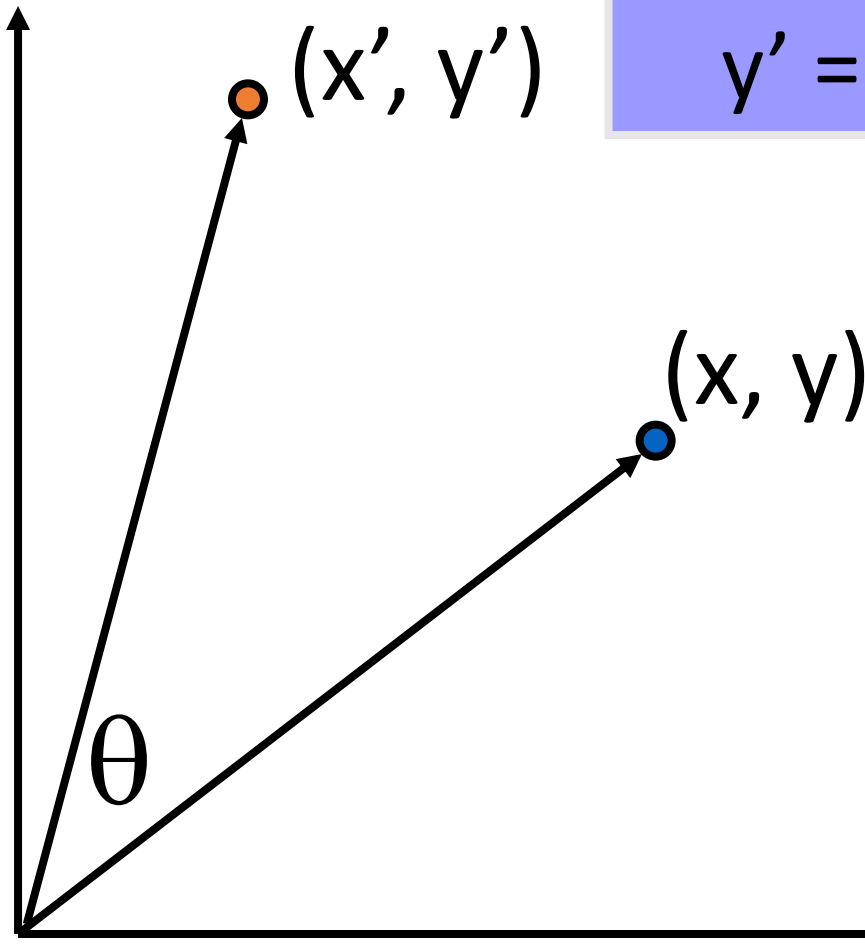
- Or, in matrix form:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \underbrace{\begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix}}_{\text{scaling matrix } S} \begin{bmatrix} x \\ y \end{bmatrix}$$

# 2-D Rotation

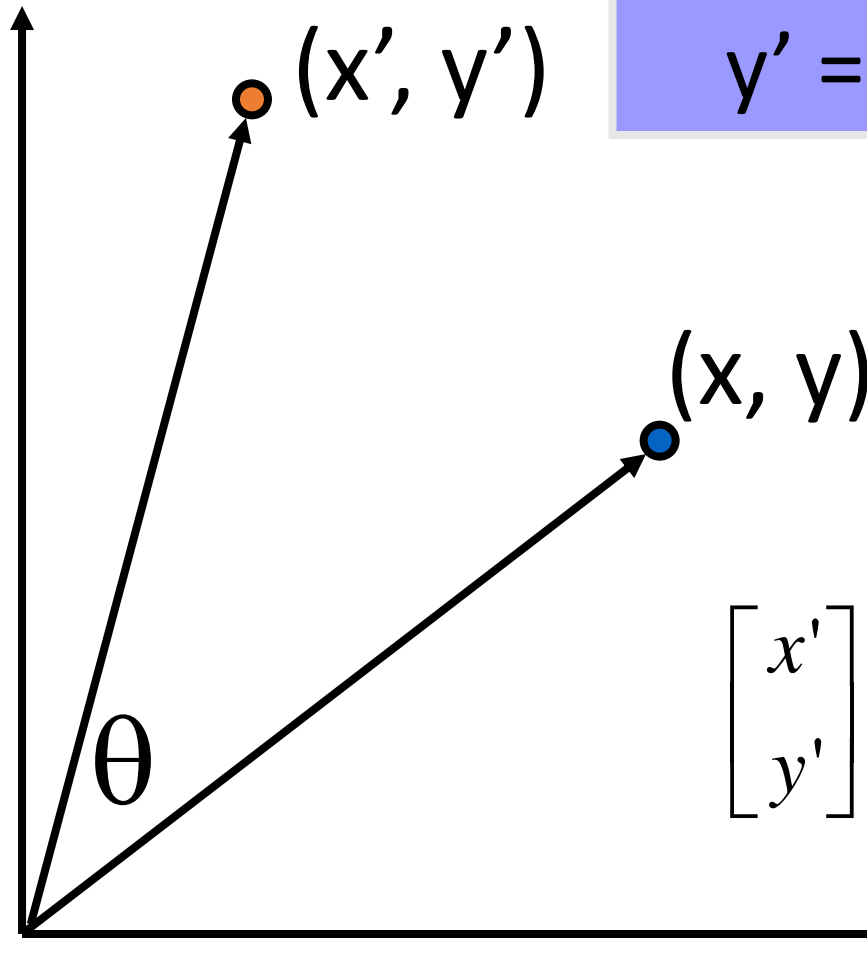


# 2-D Rotation



$$\begin{aligned}x' &= x \cos(\theta) - y \sin(\theta) \\y' &= x \sin(\theta) + y \cos(\theta)\end{aligned}$$

# 2-D Rotation



$$\begin{aligned}x' &= x \cos(\theta) - y \sin(\theta) \\y' &= x \sin(\theta) + y \cos(\theta)\end{aligned}$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \underbrace{\begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}}_{\mathbf{R}} \begin{bmatrix} x \\ y \end{bmatrix}$$

# Basic 2D transformations

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Scale

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & \alpha_x \\ \alpha_y & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Shear

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos \Theta & -\sin \Theta \\ \sin \Theta & \cos \Theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Rotate

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Translate

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Affine

Affine is any combination of translation, scale, rotation, shear

# Affine Transformations

Affine transformations are combinations of

- Linear transformations, and
- Translations

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

or

Properties of affine transformations:

- Lines map to lines
- Parallel lines remain parallel
- Ratios are preserved

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

# Projective Transformations

Projective transformations are combos of

- Affine transformations, and
- Projective warps

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

Properties of projective transformations:

- Lines map to lines
- Parallel lines do not necessarily remain parallel
- Ratios are not preserved
- Projective matrix is defined up to a scale (8 DOF)





# Projective Transformations (homography)

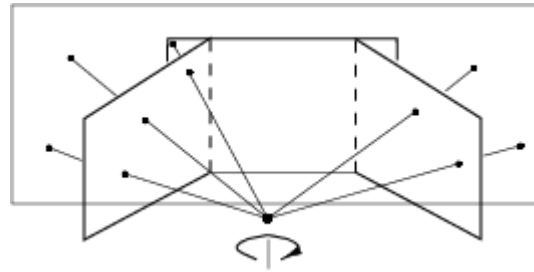
- The transformation between two views of a planar surface



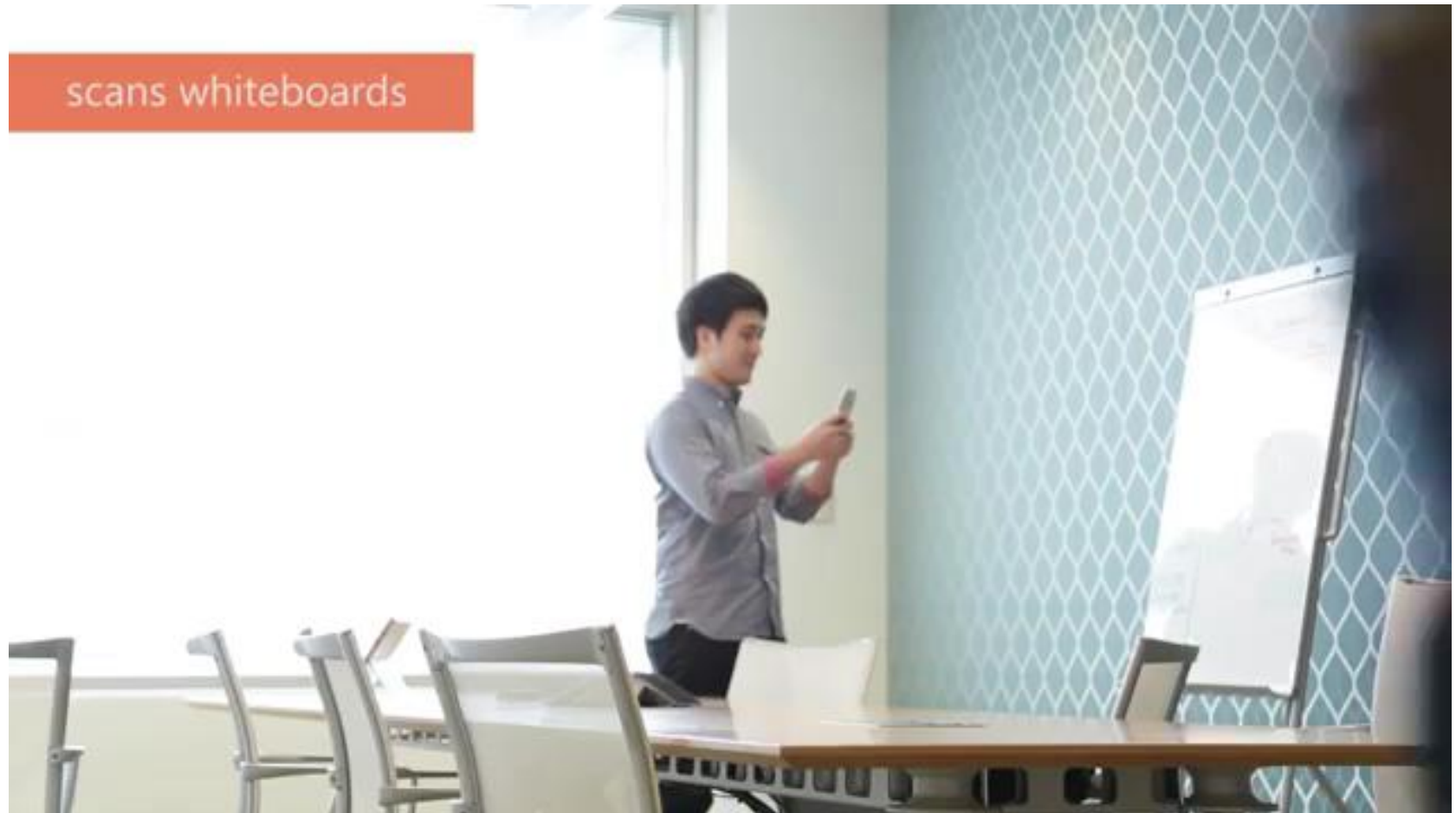
- The transformation between images from two cameras that share the same center



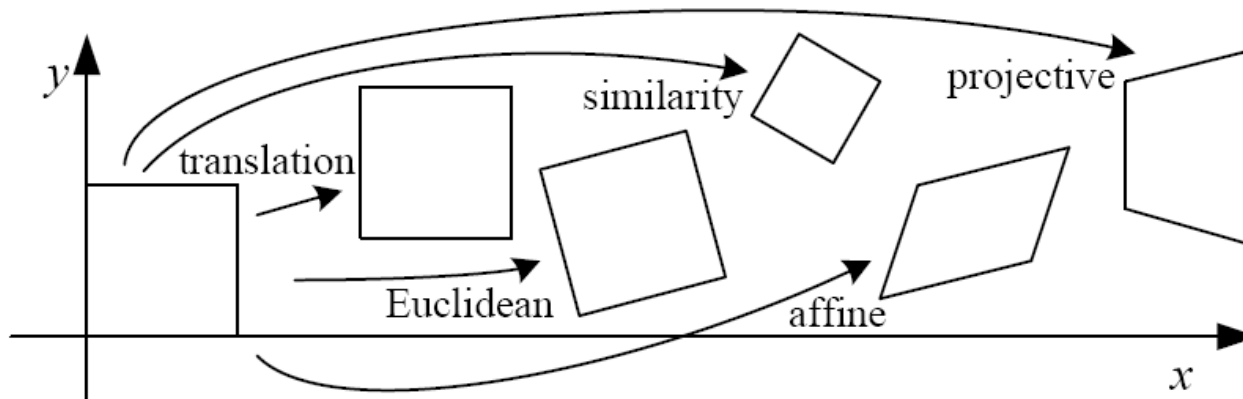
# Application: Panorama stitching








# Application: document scanning



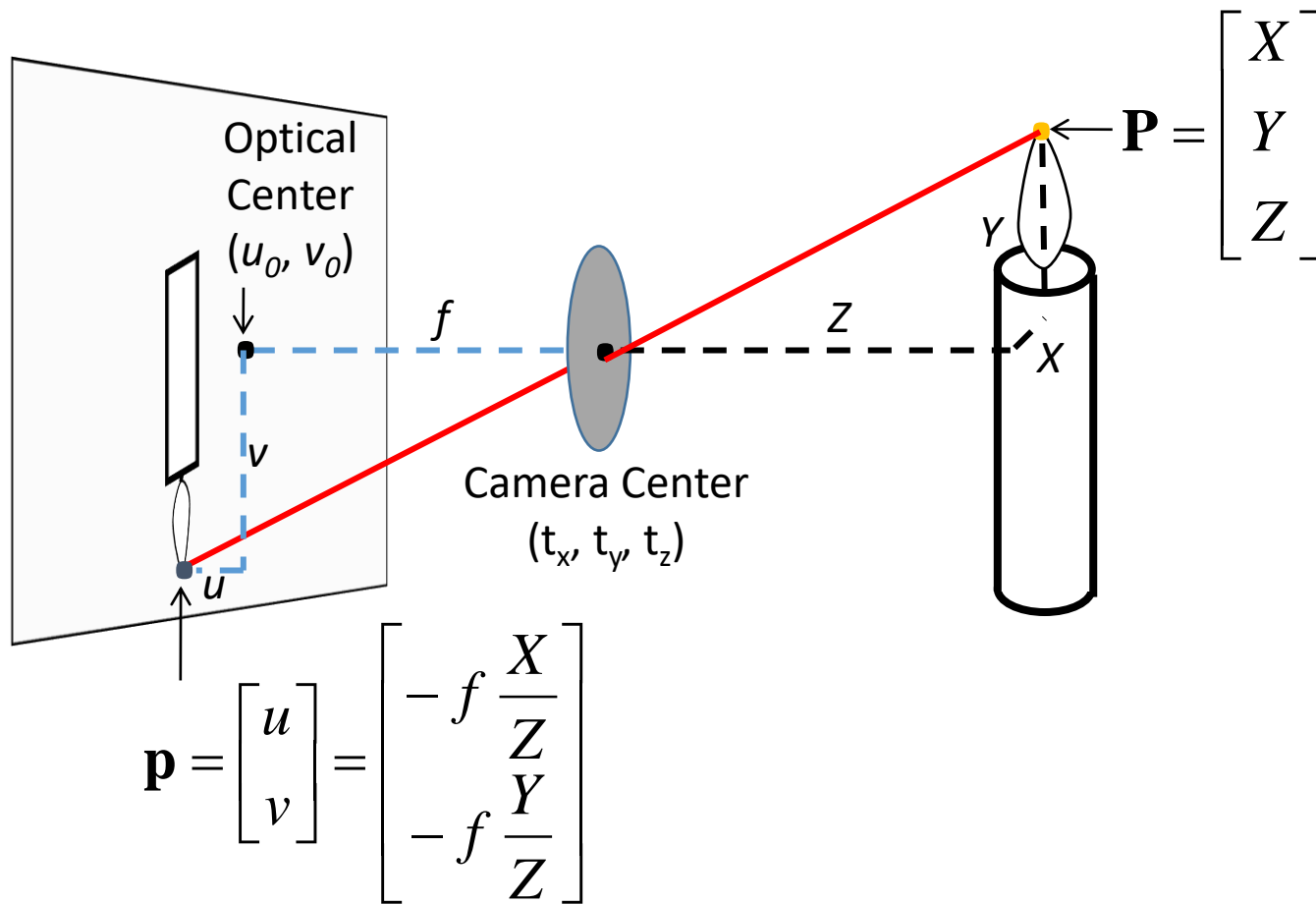
# 2D image transformations (reference table)



Name	Matrix	# D.O.F.	Preserves:	Icon
translation	$\begin{bmatrix} I & t \end{bmatrix}_{2 \times 3}$	2	orientation + ...	
rigid (Euclidean)	$\begin{bmatrix} R & t \end{bmatrix}_{2 \times 3}$	3	lengths + ...	
similarity	$\begin{bmatrix} sR & t \end{bmatrix}_{2 \times 3}$	4	angles + ...	
affine	$\begin{bmatrix} A \end{bmatrix}_{2 \times 3}$	6	parallelism + ...	
projective	$\begin{bmatrix} \tilde{H} \end{bmatrix}_{3 \times 3}$	8	straight lines	

What is the relation between scene point and image point in form of matrix?

Projection:  
world coordinates  $\rightarrow$  image coordinates



# Perspective Projection Matrix

- Projection is a matrix multiplication using homogeneous coordinates

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

divide by the third coordinate

# Perspective Projection Matrix

- Projection is a matrix multiplication using homogeneous coordinates

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

divide by the third coordinate



# Perspective Projection Matrix

- Projection is a matrix multiplication using homogeneous coordinates

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

divide by the third coordinate

# Perspective Projection Matrix

- Projection is a matrix multiplication using homogeneous coordinates

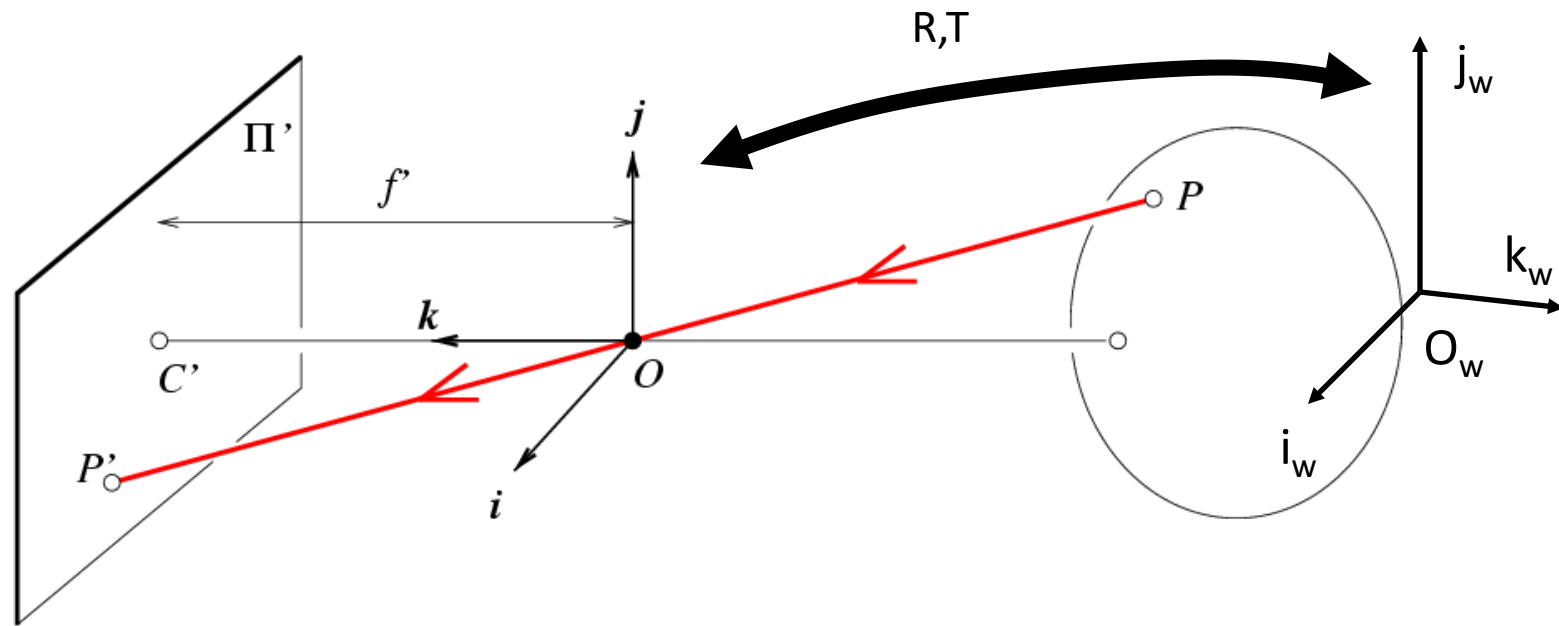
$$\begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} f x \\ f y \\ z \end{bmatrix} \Rightarrow \left( f \frac{x}{z}, f \frac{y}{z} \right)$$

divide by the third coordinate

In practice: lots of coordinate transformations...

$$\begin{pmatrix} \text{2D} \\ \text{point} \\ (3 \times 1) \end{pmatrix} = \begin{pmatrix} \text{Camera to} \\ \text{pixel coord.} \\ \text{trans. matrix} \\ (3 \times 3) \end{pmatrix} \begin{pmatrix} \text{Perspective} \\ \text{projection matrix} \\ (3 \times 3) \end{pmatrix} \begin{pmatrix} \text{World to} \\ \text{camera coord.} \\ \text{trans. matrix} \\ (3 \times 4) \end{pmatrix} \begin{pmatrix} \text{3D} \\ \text{point} \\ (4 \times 1) \end{pmatrix}$$

# Projection matrix



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

$\mathbf{x}$ : Image Coordinates:  $(u, v, 1)$

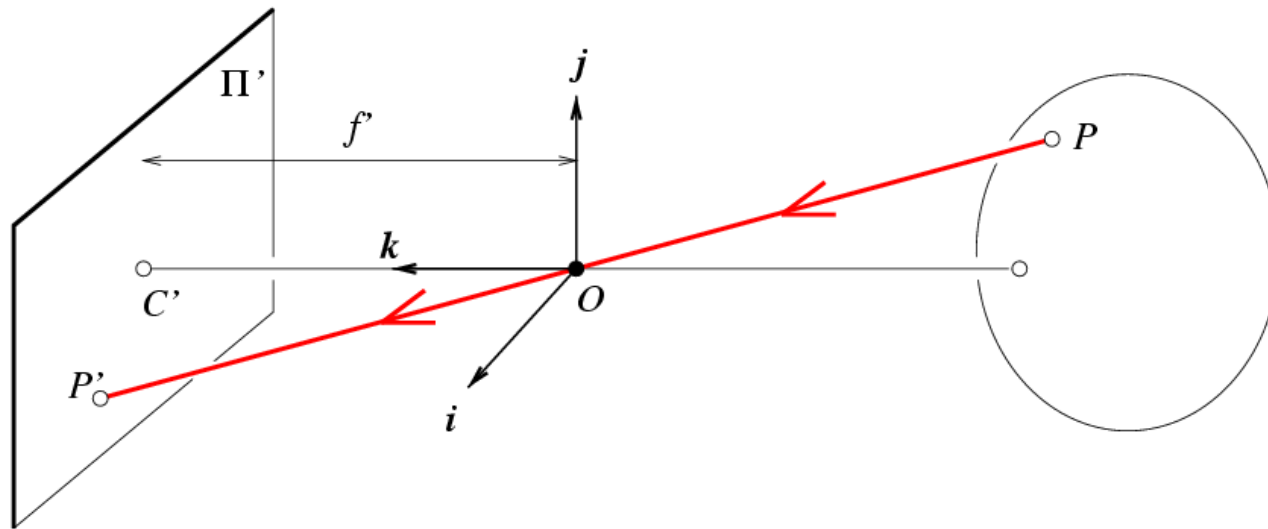
$\mathbf{K}$ : Intrinsic Matrix  $(3 \times 3)$

$\mathbf{R}$ : Rotation  $(3 \times 3)$

$\mathbf{t}$ : Translation  $(3 \times 1)$

$\mathbf{X}$ : World Coordinates:  $(X, Y, Z, 1)$

# Projection matrix under assumptions



## Intrinsic Assumptions

- Unit aspect ratio
- No skew
- Optical center at (0,0)  
(i.e., at image origin)

## Extrinsic Assumptions

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

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

$\mathbf{K}$

# Remove assumption: known optical center

## Intrinsic Assumptions

- Unit aspect ratio
- No skew

## Extrinsic Assumptions

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

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

# Remove assumption: square pixels

## Intrinsic Assumptions

- No skew

## Extrinsic Assumptions

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

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

# Remove assumption: non-skewed pixels

Intrinsic Assumptions

Extrinsic Assumptions

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

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

Note: different books use different notation for parameters

# Remove assumption: Camera translation

Intrinsic Assumptions

Extrinsic Assumptions

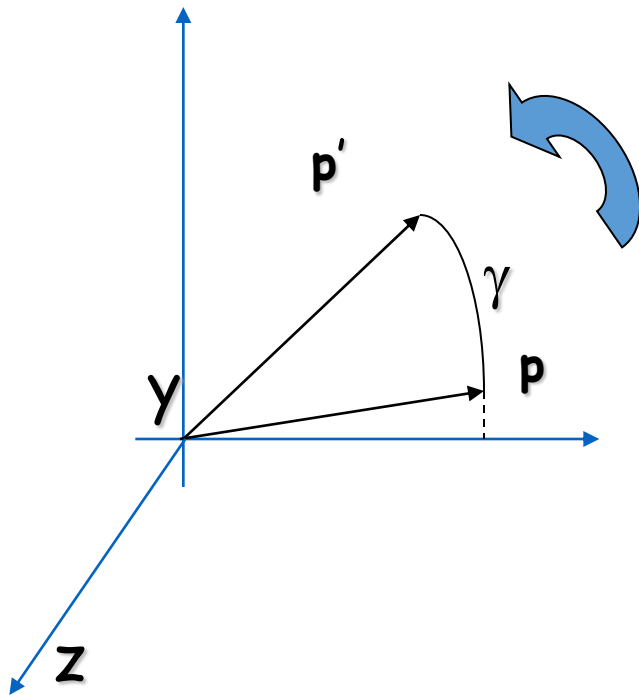
- No rotation

$$\mathbf{x} = \mathbf{K} \begin{bmatrix} \mathbf{I} & \mathbf{t} \end{bmatrix} \mathbf{X} \Rightarrow w \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha & s & u_0 \\ 0 & \beta & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$



# 3D Rotation of Points

Rotation around the coordinate axes, **counter-clockwise**:



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

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

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

Remove all assumption:  
Camera rotation also

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



$$w \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha & s & u_0 \\ 0 & \beta & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Remove all assumption:  
Camera rotation also

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



$$w \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha & s & u_0 \\ 0 & \beta & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Aspect  
Ratio

$$w \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f & s & u_0 \\ 0 & \alpha f & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

# Degrees of freedom

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

5



6

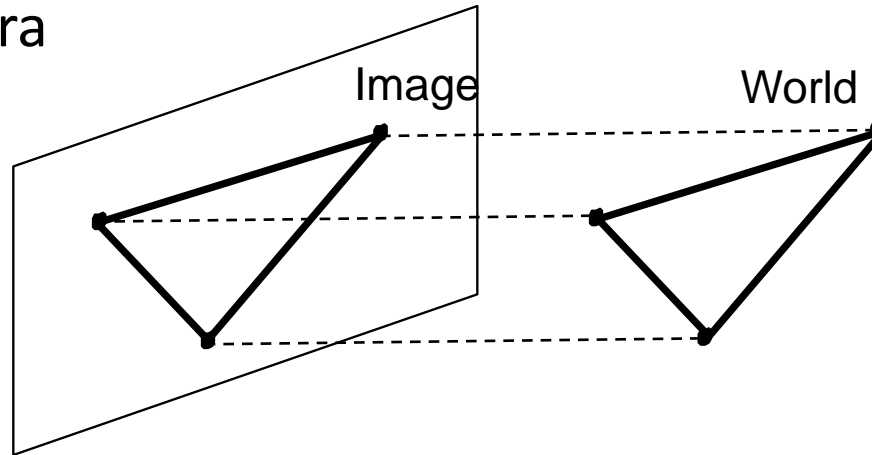
$$w \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha & s & u_0 \\ 0 & \beta & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Aspect  
Ratio

$$w \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f & s & u_0 \\ 0 & \alpha f & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

# Orthographic Projection

- Object dimensions are small compared to distance to camera

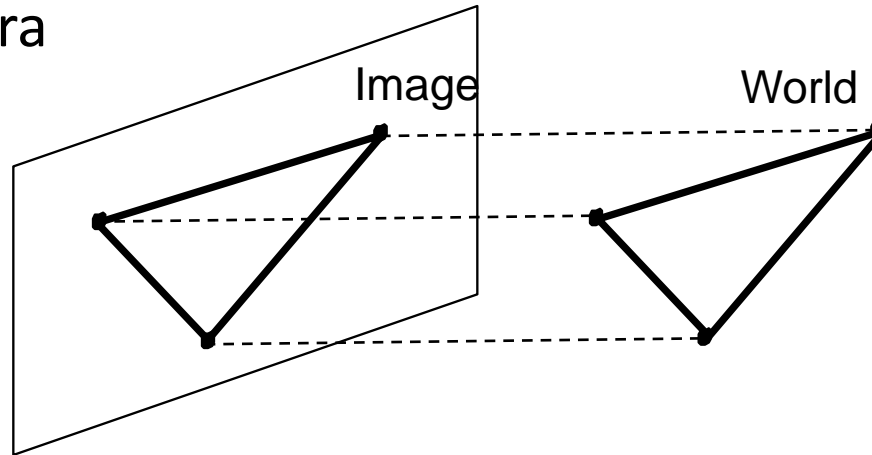


- Good approximation for telephoto optics
- Also called “parallel projection”:  $(x, y, z) \rightarrow (x, y)$
- **What's the projection matrix?**

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

# Orthographic Projection

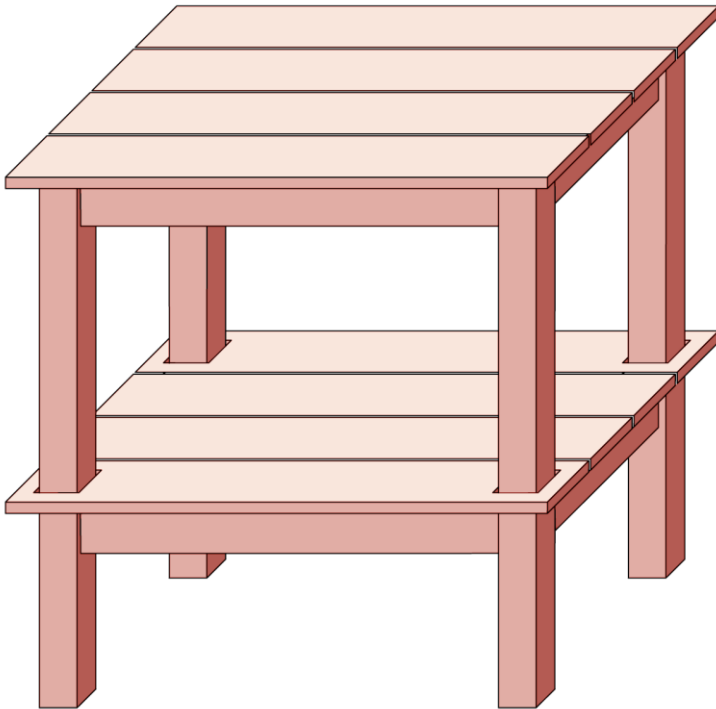
- Object dimensions are small compared to distance to camera



- Good approximation for telephoto optics
- Also called “parallel projection”:  $(x, y, z) \rightarrow (x, y)$
- **What's the projection matrix?**

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

# Orthographic Projection



- What happens to parallel lines?
- What happens to angles?
- What happens to distances?

How can we calibrate the camera?



How to calibrate the camera?

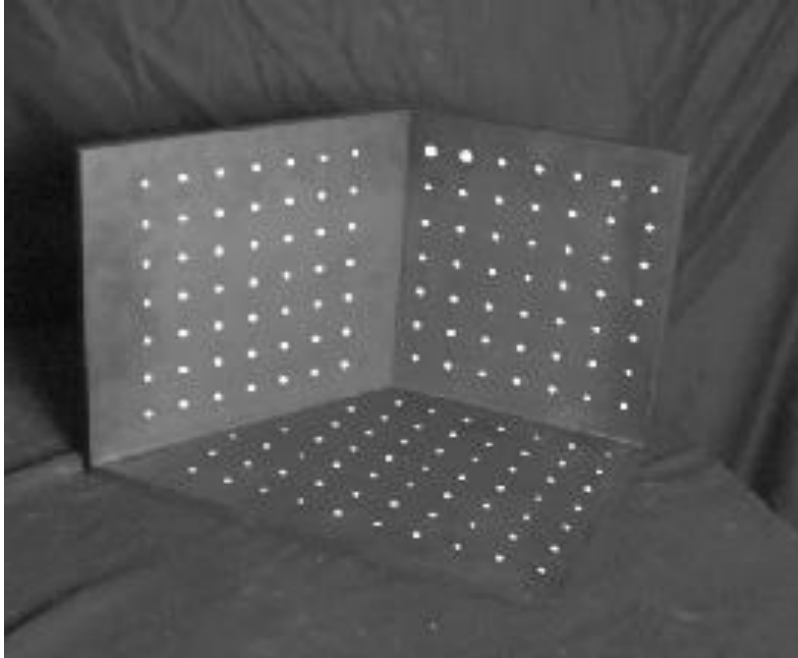
$$\mathbf{x} = \mathbf{K} \begin{bmatrix} \mathbf{R} & \mathbf{t} \end{bmatrix} \mathbf{X}$$

$$\begin{bmatrix} wu \\ wv \\ w \end{bmatrix} = \begin{bmatrix} * & * & * & * \\ * & * & * & * \\ * & * & * & * \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

# Calibrating the Camera

Method 1: Use an object (calibration grid) with known geometry

- Correspond image points to 3d points
- Get least squares solution (or non-linear solution)

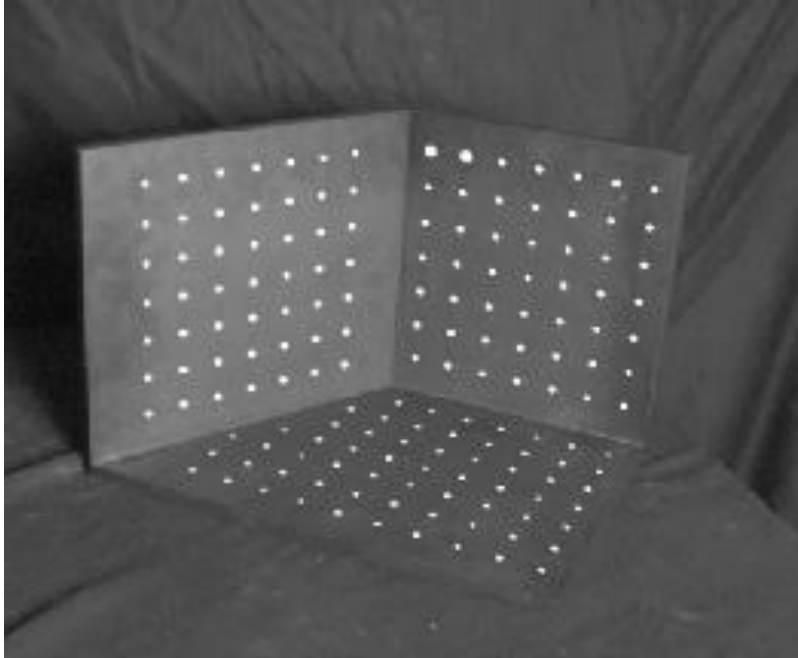


$$\begin{bmatrix} wu \\ wv \\ w \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

# Calibrating the Camera

Method 1: Use an object (calibration grid) with known geometry

- Correspond image points to 3d points
- Get least squares solution (or non-linear solution)



Known 3d  
locations



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

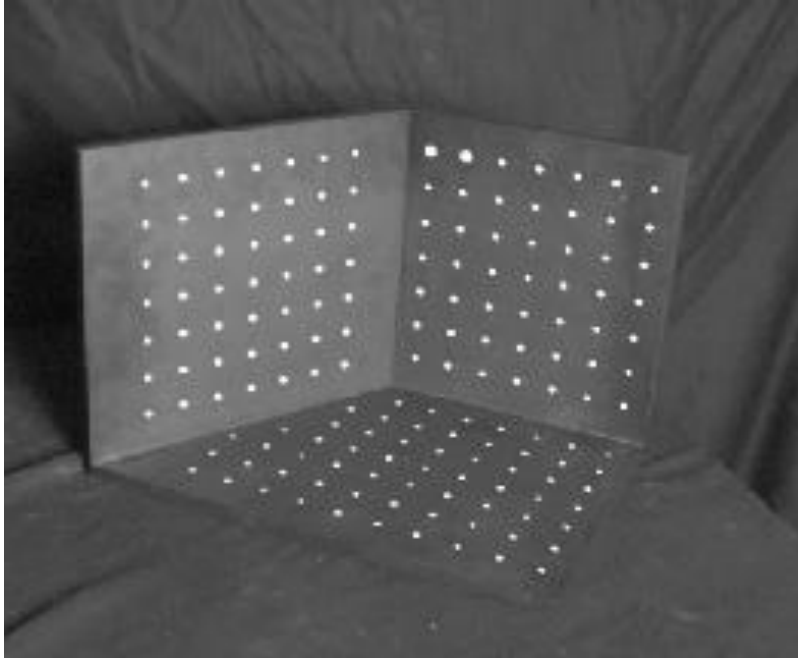
# Calibrating the Camera

Method 1: Use an object (calibration grid) with known geometry

- Correspond image points to 3d points
- Get least squares solution (or non-linear solution)

Known 2d image  
coordinates

Known 3d  
locations



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

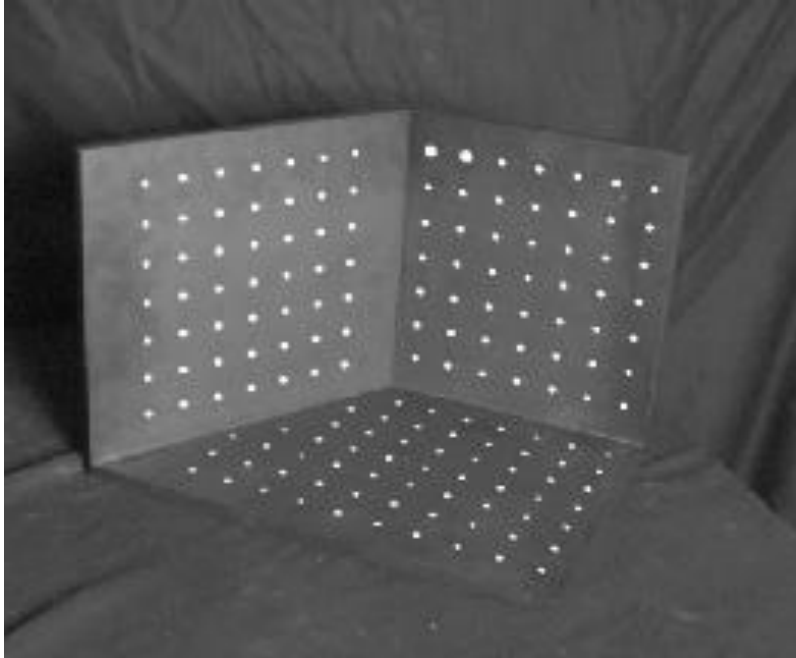
# Calibrating the Camera

Method 1: Use an object (calibration grid) with known geometry

- Correspond image points to 3d points
- Get least squares solution (or non-linear solution)

Known 2d image  
coordinates

Known 3d  
locations



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

Unknown Camera Parameters

## Unknown Camera Parameters



Known 2d  
image coords

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

Known 3d  
locations

## Unknown Camera Parameters



Known 2d  
image coords

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

Known 3d  
locations

$$s = m_{31}X + m_{32}Y + m_{33}Z + m_{34}$$

## Unknown Camera Parameters



Known 2d  
image coords

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

Known 3d  
locations

$$s = m_{31}X + m_{32}Y + m_{33}Z + m_{34}$$

$$su = m_{31}uX + m_{32}uY + m_{33}uZ + m_{34}u = m_{11}X + m_{12}Y + m_{13}Z + m_{14}$$



## Unknown Camera Parameters



Known 2d  
image coords

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

Known 3d  
locations

$$s = m_{31}X + m_{32}Y + m_{33}Z + m_{34}$$

$$su = m_{31}uX + m_{32}uY + m_{33}uZ + m_{34}u = m_{11}X + m_{12}Y + m_{13}Z + m_{14}$$

$$sv = m_{31}vX + m_{32}vY + m_{33}vZ + m_{34}v = m_{21}X + m_{22}Y + m_{23}Z + m_{24}$$

## Unknown Camera Parameters



Known 2d  
image coords

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

Known 3d  
locations

$$s = m_{31}X + m_{32}Y + m_{33}Z + m_{34}$$

$$su = m_{31}uX + m_{32}uY + m_{33}uZ + m_{34}u = m_{11}X + m_{12}Y + m_{13}Z + m_{14}$$

$$sv = m_{31}vX + m_{32}vY + m_{33}vZ + m_{34}v = m_{21}X + m_{22}Y + m_{23}Z + m_{24}$$

$$m_{11}X + m_{12}Y + m_{13}Z + m_{14} - m_{31}uX - m_{32}uY - m_{33}uZ - m_{34}u = 0$$

$$m_{21}X + m_{22}Y + m_{23}Z + m_{24} - m_{31}vX - m_{32}vY - m_{33}vZ - m_{34}v = 0$$

## Unknown Camera Parameters



Known 2d  
image coords

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

Known 3d  
locations

$$m_{11}X + m_{12}Y + m_{13}Z + m_{14} - m_{31}uX - m_{32}uY - m_{33}uZ - m_{34}u = 0$$

$$m_{21}X + m_{22}Y + m_{23}Z + m_{24} - m_{31}vX - m_{32}vY - m_{33}vZ - m_{34}v = 0$$

# Unknown Camera Parameters



Known 2d  
image coords

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

Known 3d  
locations

$$m_{11}X + m_{12}Y + m_{13}Z + m_{14} - m_{31}uX - m_{32}uY - m_{33}uZ - m_{34}u = 0$$

$$m_{21}X + m_{22}Y + m_{23}Z + m_{24} - m_{31}vX - m_{32}vY - m_{33}vZ - m_{34}v = 0$$

- Homogeneous linear system.  
Solve for m's entries using  
linear least squares

$$\begin{bmatrix} X_1 & Y_1 & Z_1 & 1 & 0 & 0 & 0 & 0 & -u_1X_1 & -u_1Y_1 & -u_1Z_1 & -u_1 \\ 0 & 0 & 0 & 0 & X_1 & Y_1 & Z_1 & 1 & -v_1X_1 & -v_1Y_1 & -v_1Z_1 & -v_1 \\ & & & & & & \vdots & & & & & \\ X_n & Y_n & Z_n & 1 & 0 & 0 & 0 & 0 & -u_nX_n & -u_nY_n & -u_nZ_n & -u_n \\ 0 & 0 & 0 & 0 & X_n & Y_n & Z_n & 1 & -v_nX_n & -v_nY_n & -v_nZ_n & -v_n \end{bmatrix} \begin{bmatrix} m_{11} \\ m_{12} \\ m_{13} \\ m_{14} \\ m_{21} \\ m_{22} \\ m_{23} \\ m_{24} \\ m_{31} \\ m_{32} \\ m_{33} \\ m_{34} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}$$

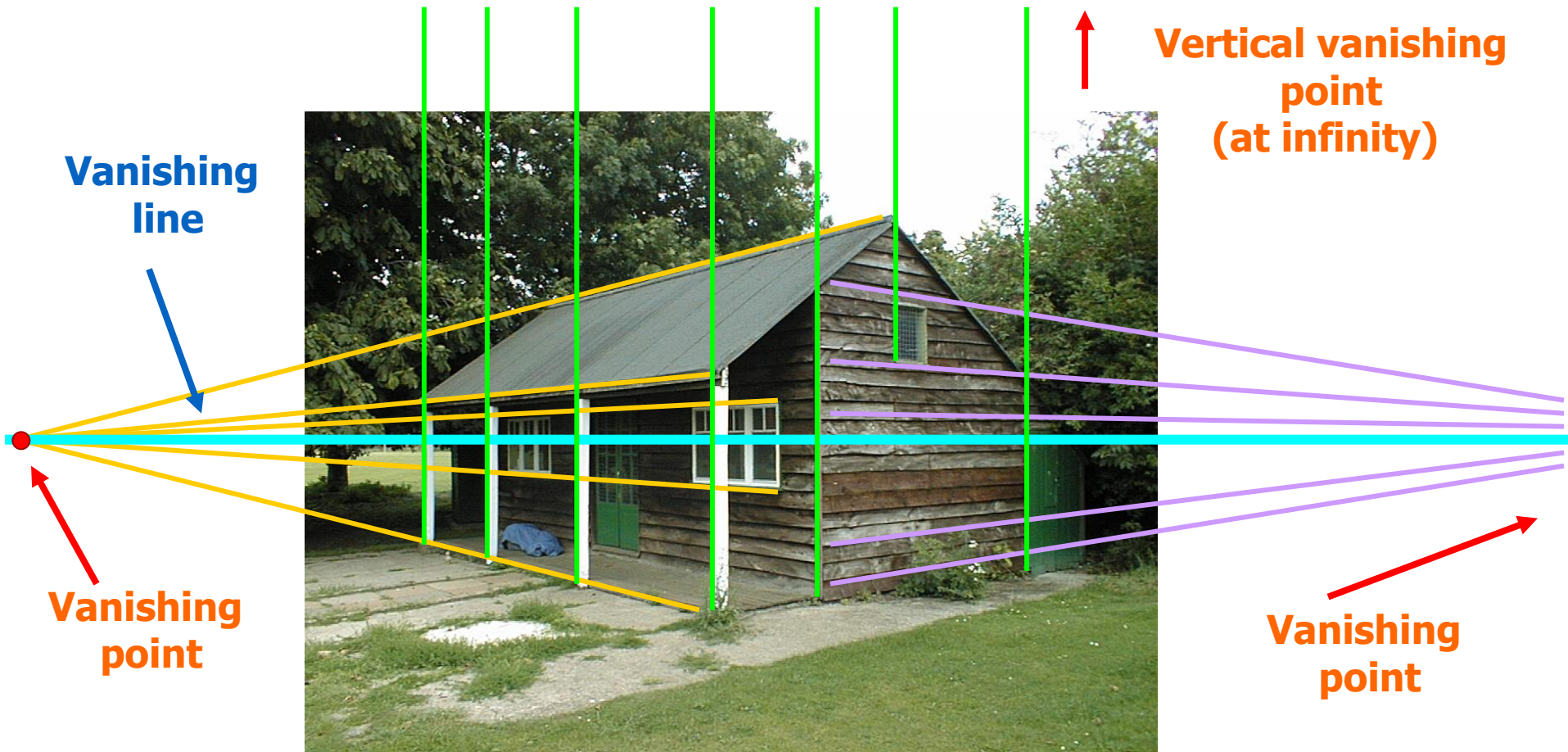
# Can we factorize $M$ back to $K [R \mid T]$ ?

- Yes!
  - We can use  $QR$  factorization
- 
- <http://ksimek.github.io/2012/08/14/decompose/>
  - [https://en.wikipedia.org/wiki/QR\\_decomposition](https://en.wikipedia.org/wiki/QR_decomposition)

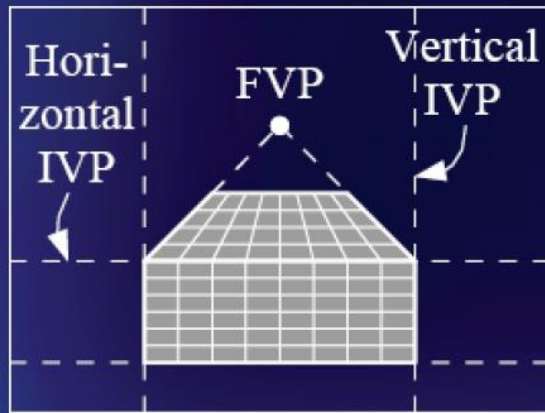
# Calibrating the Camera

## Method 2: Use vanishing points

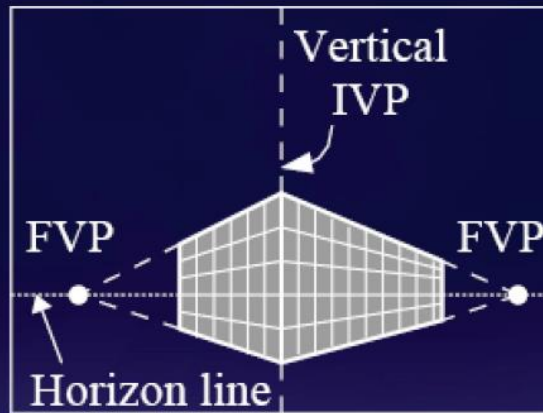
- Find vanishing points corresponding to orthogonal directions



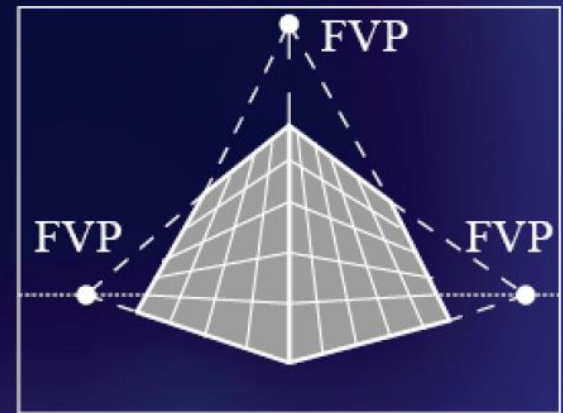
# Calibration from vanishing points



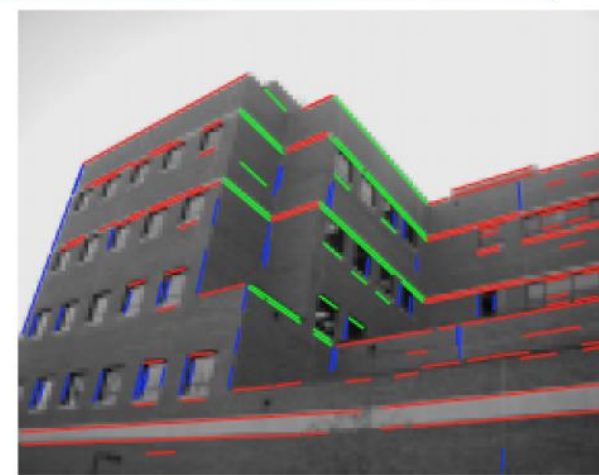
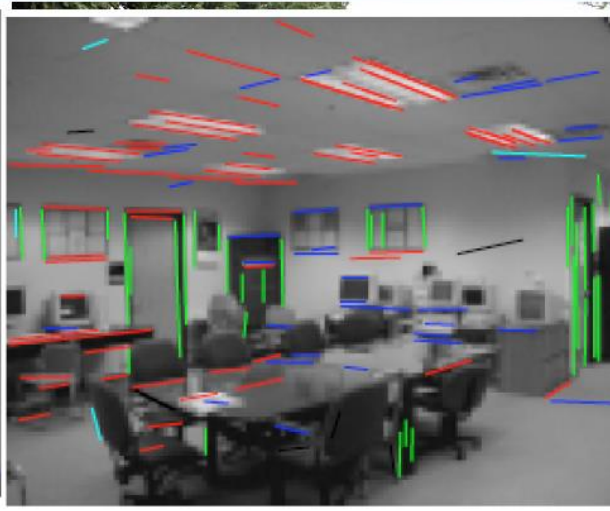
1 finite vanishing point,  
2 infinite vanishing points



2 finite vanishing points,  
1 infinite vanishing point



3 finite vanishing points



# Calibration by orthogonal vanishing points

- Intrinsic camera matrix
  - Use orthogonality as a constraint
  - Model  $\mathbf{K}$  with only  $f, u_0, v_0$

$$\mathbf{p}_i = \mathbf{K} \mathbf{R} \mathbf{X}_i$$



# Calibration by orthogonal vanishing points

- Intrinsic camera matrix
  - Use orthogonality as a constraint
  - Model  $\mathbf{K}$  with only  $f, u_0, v_0$

$$\mathbf{p}_i = \mathbf{K} \mathbf{R} \mathbf{X}_i$$

$$\mathbf{X}_i = \mathbf{R}^{-1} \mathbf{K}^{-1} \mathbf{p}_i = \mathbf{R}^T \mathbf{K}^{-1} \mathbf{p}_i$$

# Calibration by orthogonal vanishing points

- Intrinsic camera matrix
  - Use orthogonality as a constraint
  - Model  $\mathbf{K}$  with only  $f, u_0, v_0$

For vanishing points

$$\mathbf{p}_i = \mathbf{K} \mathbf{R} \mathbf{X}_i \qquad \mathbf{X}_i^T \mathbf{X}_j = 0$$
$$\mathbf{X}_i = \mathbf{R}^{-1} \mathbf{K}^{-1} \mathbf{p}_i = \mathbf{R}^T \mathbf{K}^{-1} \mathbf{p}_i$$

# Calibration by orthogonal vanishing points

- Intrinsic camera matrix
  - Use orthogonality as a constraint
  - Model  $\mathbf{K}$  with only  $f, u_0, v_0$

For vanishing points

$$\mathbf{p}_i = \mathbf{K} \mathbf{R} \mathbf{X}_i \qquad \mathbf{X}_i^T \mathbf{X}_j = 0$$

$$\mathbf{X}_i = \mathbf{R}^{-1} \mathbf{K}^{-1} \mathbf{p}_i = \mathbf{R}^T \mathbf{K}^{-1} \mathbf{p}_i$$

$$\mathbf{p}_i^T (\mathbf{K}^{-1})^T (\mathbf{R}^T)^T (\mathbf{R}^T) (\mathbf{K}^{-1}) \mathbf{p}_j = 0$$

# Calibration by orthogonal vanishing points

- Intrinsic camera matrix
  - Use orthogonality as a constraint
  - Model  $\mathbf{K}$  with only  $f, u_0, v_0$

For vanishing points

$$\mathbf{p}_i = \mathbf{K} \mathbf{R} \mathbf{X}_i \qquad \mathbf{X}_i^T \mathbf{X}_j = 0$$

$$\mathbf{X}_i = \mathbf{R}^{-1} \mathbf{K}^{-1} \mathbf{p}_i = \mathbf{R}^T \mathbf{K}^{-1} \mathbf{p}_i$$

$$\mathbf{p}_i^T (\mathbf{K}^{-1})^T (\mathbf{R}^T)^T (\mathbf{R}^T) (\mathbf{K}^{-1}) \mathbf{p}_j = 0$$

$$\mathbf{p}_i^T \mathbf{K}^{-T} \mathbf{R} \mathbf{R}^T \mathbf{K}^{-1} \mathbf{p}_j = 0$$

# Calibration by orthogonal vanishing points

- Intrinsic camera matrix
  - Use orthogonality as a constraint
  - Model  $\mathbf{K}$  with only  $f, u_0, v_0$

For vanishing points

$$\mathbf{p}_i = \mathbf{K} \mathbf{R} \mathbf{X}_i \qquad \mathbf{X}_i^T \mathbf{X}_j = 0$$

$$\mathbf{X}_i = \mathbf{R}^{-1} \mathbf{K}^{-1} \mathbf{p}_i = \mathbf{R}^T \mathbf{K}^{-1} \mathbf{p}_i$$

$$\mathbf{p}_i^T (\mathbf{K}^{-1})^T (\mathbf{R}^T)^T (\mathbf{R}^T) (\mathbf{K}^{-1}) \mathbf{p}_j = 0$$

$$\mathbf{p}_i^T \mathbf{K}^{-T} \mathbf{R} \mathbf{R}^T \mathbf{K}^{-1} \mathbf{p}_j = 0$$

$$\mathbf{p}_i^T \mathbf{K}^{-T} \mathbf{K}^{-1} \mathbf{p}_j = 0$$

# Calibration by orthogonal vanishing points

- Intrinsic camera matrix
  - Use orthogonality as a constraint
  - Model  $\mathbf{K}$  with only  $f, u_0, v_0$

For vanishing points

$$\mathbf{p}_i = \mathbf{K} \mathbf{R} \mathbf{X}_i \quad \mathbf{X}_i^T \mathbf{X}_j = 0$$

$$\mathbf{X}_i = \mathbf{R}^{-1} \mathbf{K}^{-1} \mathbf{p}_i = \mathbf{R}^T \mathbf{K}^{-1} \mathbf{p}_i$$

$$\mathbf{p}_i^T (\mathbf{K}^{-1})^T (\mathbf{R}^T)^T (\mathbf{R}^T) (\mathbf{K}^{-1}) \mathbf{p}_j = 0$$

$$\mathbf{p}_i^T \mathbf{K}^{-T} \mathbf{R} \mathbf{R}^T \mathbf{K}^{-1} \mathbf{p}_j = 0$$

$$\mathbf{p}_i^T \mathbf{K}^{-T} \mathbf{K}^{-1} \mathbf{p}_j = 0$$

- Similarly

$$\mathbf{p}_i^T \mathbf{K}^{-T} \mathbf{K}^{-1} \mathbf{p}_k = 0$$

$$\mathbf{p}_j^T \mathbf{K}^{-T} \mathbf{K}^{-1} \mathbf{p}_k = 0$$

# Calibration by orthogonal vanishing points

- Intrinsic camera matrix
  - Use orthogonality as a constraint
  - Model  $\mathbf{K}$  with only  $f, u_0, v_0$

For vanishing points

$$\mathbf{p}_i = \mathbf{K} \mathbf{R} \mathbf{X}_i \quad \mathbf{X}_i^T \mathbf{X}_j = 0$$

$$\mathbf{X}_i = \mathbf{R}^{-1} \mathbf{K}^{-1} \mathbf{p}_i = \mathbf{R}^T \mathbf{K}^{-1} \mathbf{p}_i$$

$$\mathbf{p}_i^T (\mathbf{K}^{-1})^T (\mathbf{R}^T)^T (\mathbf{R}^T) (\mathbf{K}^{-1}) \mathbf{p}_j = 0$$

$$\mathbf{p}_i^T \mathbf{K}^{-T} \mathbf{R} \mathbf{R}^T \mathbf{K}^{-1} \mathbf{p}_j = 0$$

$$\mathbf{p}_i^T \mathbf{K}^{-T} \mathbf{K}^{-1} \mathbf{p}_j = 0$$

- What if you don't have three finite vanishing points?
  - Two finite VP: get valid  $u_0, v_0$  closest to image center; solve  $f$
  - One finite VP:  $u_0, v_0$  is at vanishing point; can't solve for  $f$

# Things to remember

## • Projection Matrix?

### • Extrinsic Matrix

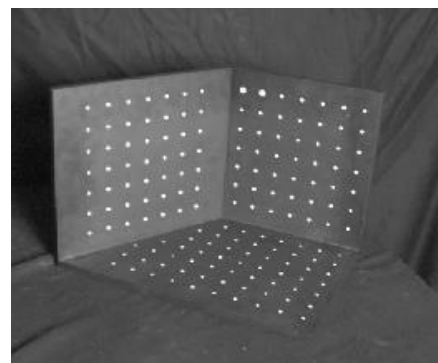
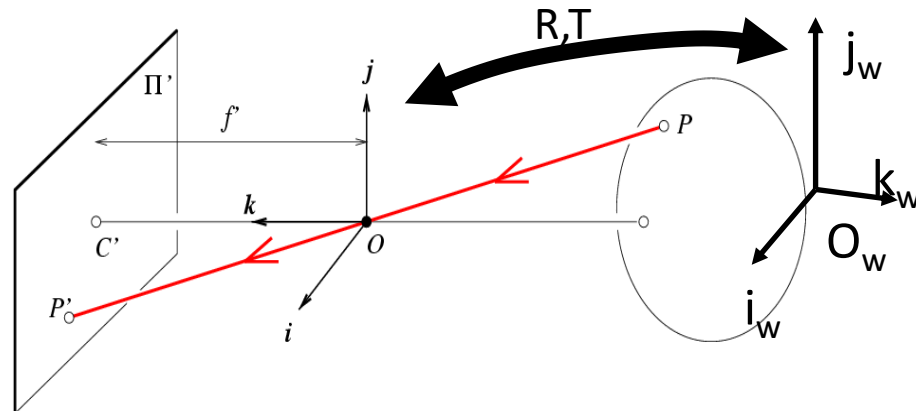
- Camera Rotation
- Camera Translation

### • Intrinsic Matrix

- Focal Length
- Optical Center
- Aspect Ratio
- Skewness

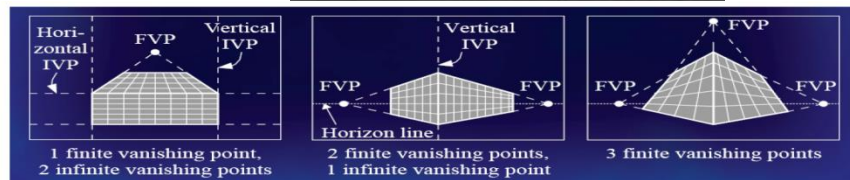
### • Degree of Freedom

- 5- Intrinsic
- 6- Extrinsic



## • Calibrate the camera?

- Use an object with known geometry
- Use vanishing points





# Acknowledgements

- Thanks to the following researchers for making their teaching/research material online
  - Forsyth
  - Steve Seitz
  - Noah Snavely
  - J.B. Huang
  - Derek Hoiem
  - J. Hays
  - J. Johnson
  - R. Girshick
  - S. Lazebnik
  - K. Grauman
  - Antonio Torralba
  - Rob Fergus
  - Leibe
  - And many more .....

Next Class:  
How can we measure the size of 3D objects  
from an image?

