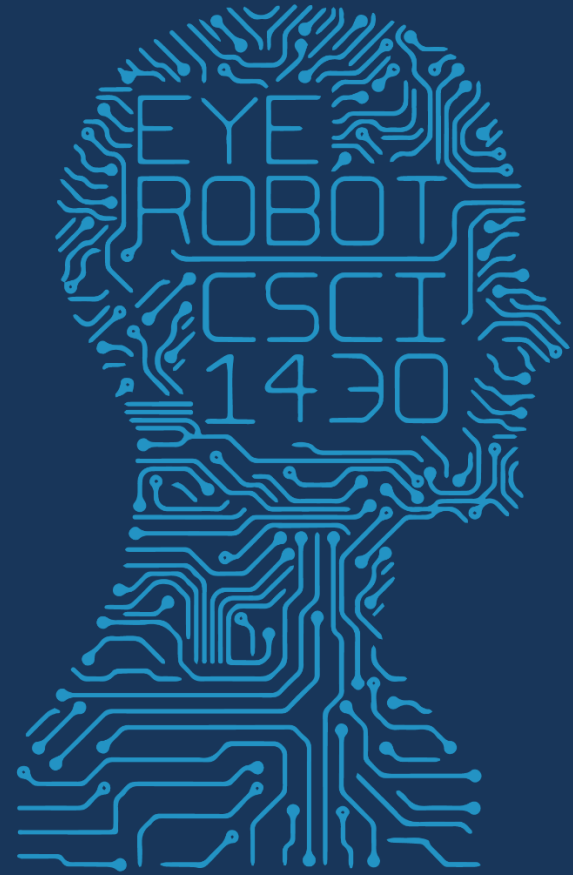




1950

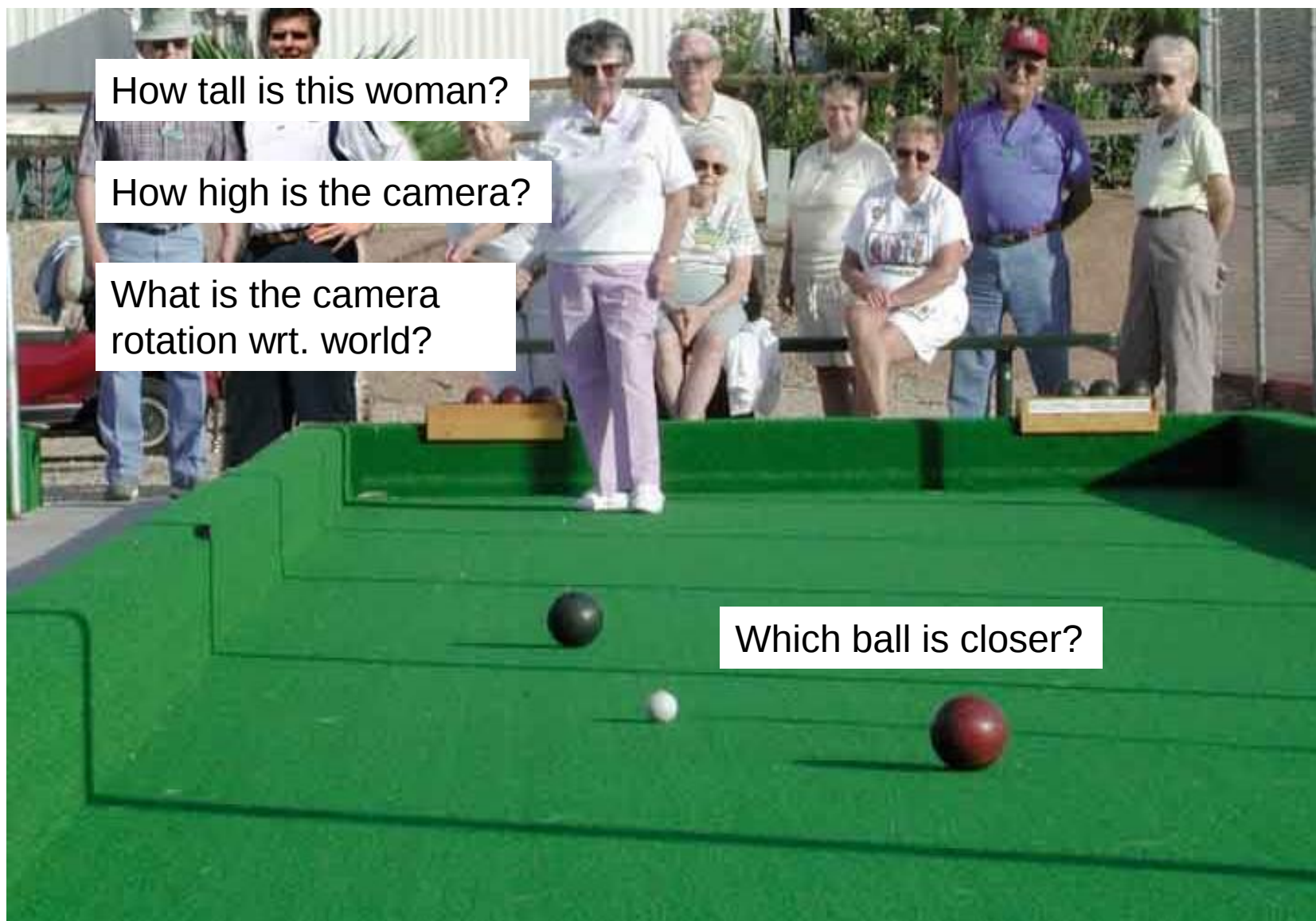
FUTURE VISION



2017 MWF 1PM

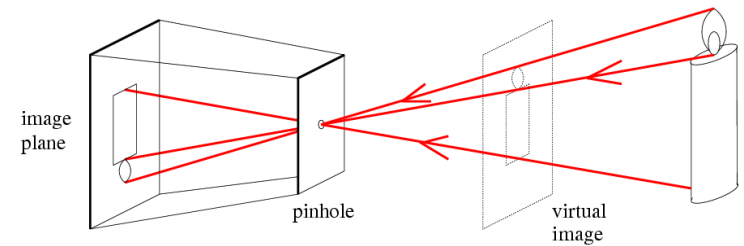
COMPUTER VISION

Cameras and World Geometry



Things to remember

Pinhole camera model and camera (projection) matrix



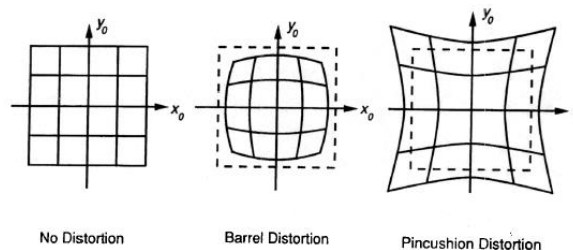
$$\mathbf{x} = \mathbf{K} \begin{bmatrix} \mathbf{R} & \mathbf{t} \end{bmatrix} \mathbf{X}$$

Homogeneous coordinates allow projection as a matrix multiplication

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

Model *approximates* a camera

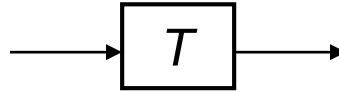
- Lens distortion
- Aperture



Parametric (global) transformations



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



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

Transformation T is a coordinate-changing machine:

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

What does it mean that T is global?

- T is the same for any point $\mathbf{p}$
- T can be described by just a few numbers (parameters)

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



Scaling



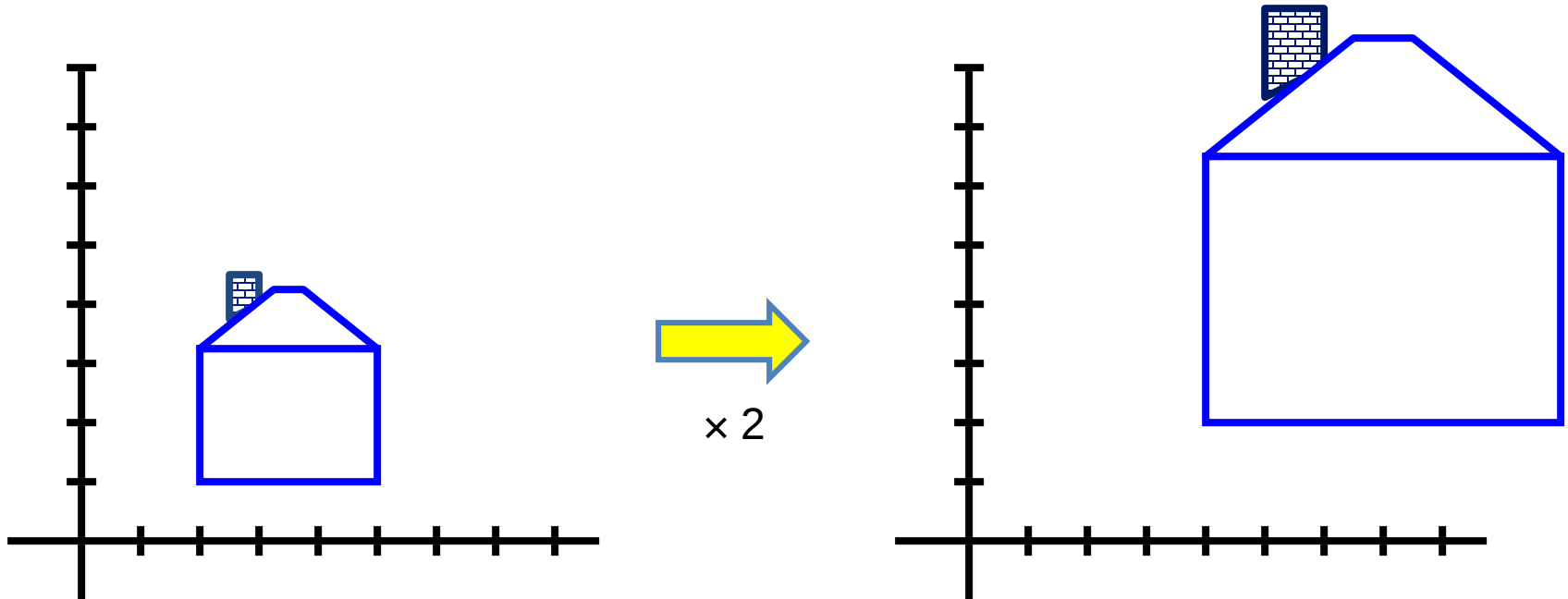
Affine



Perspective

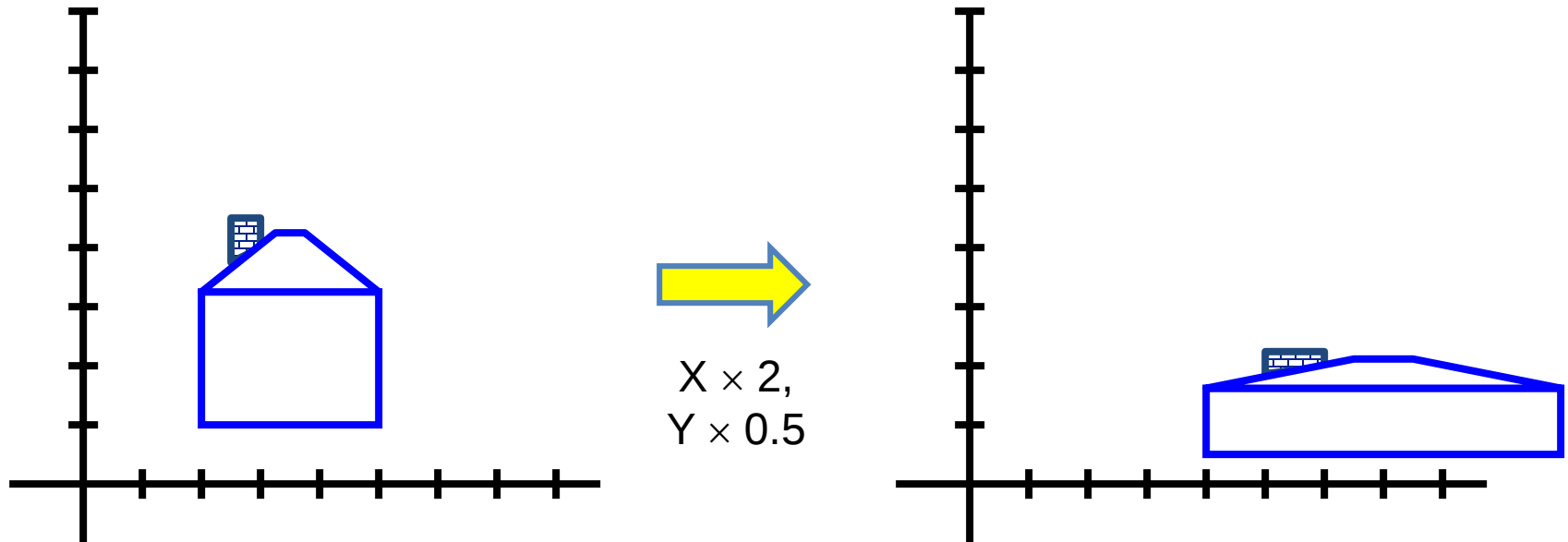
Scaling

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



Scaling

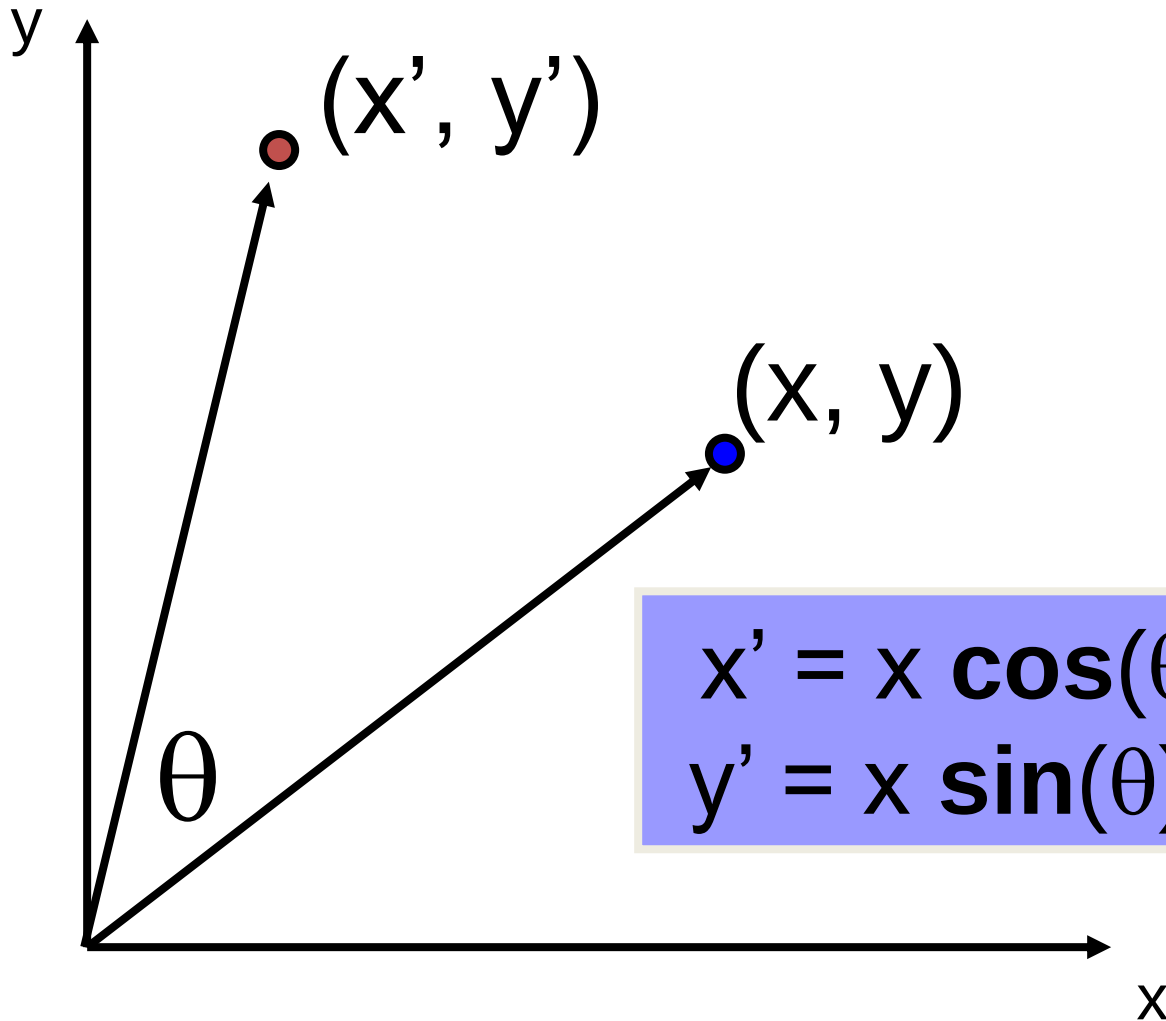
- *Non-uniform scaling*: different scalars per component:



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



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

2-D Rotation

This is easy to capture in matrix form:

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

Even though $\sin(\theta)$ and $\cos(\theta)$ are nonlinear functions of θ ,

- *x' is a linear combination of x and y*
- *y' is a linear combination of x and y*

What is the inverse transformation?

- Rotation by $-\theta$
- For rotation matrices $\mathbf{R}^{-1} = \mathbf{R}^T$

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, and shear

Affine Transformations

Affine transformations are combinations of

- Linear transformations, and
- Translations

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

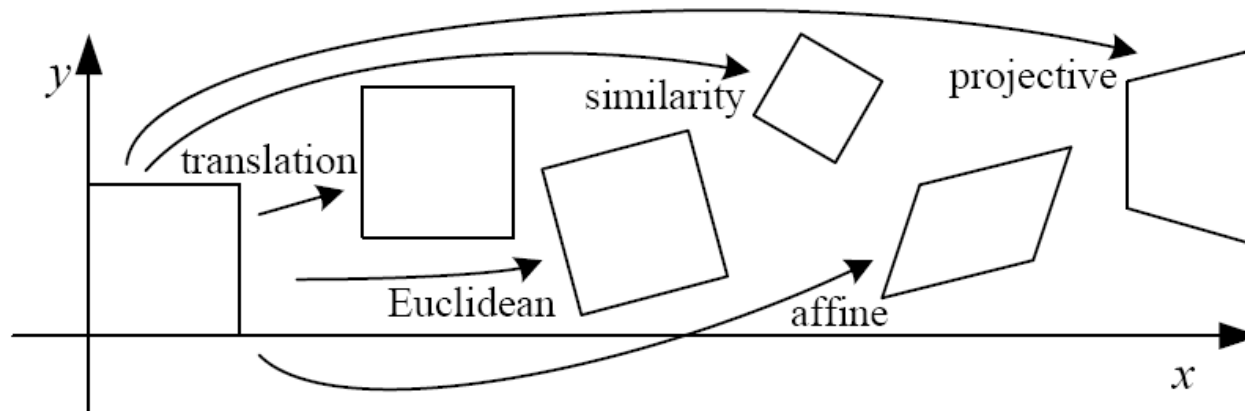
or

Properties of affine transformations:

- Lines map to lines
- Parallel lines remain parallel
- Ratios are preserved
- Closed under composition

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

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	

‘Homography’

Projective Transformations

Projective transformations are combos of

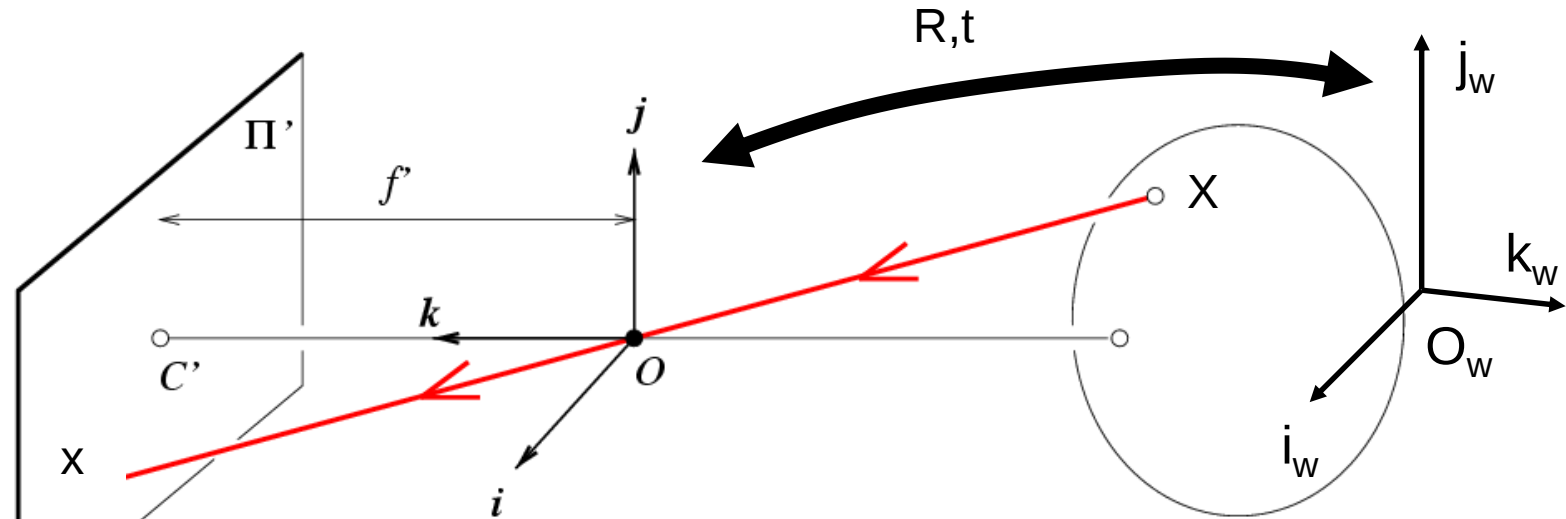
- Affine transformations, and
- Projective warps

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

Properties of projective transformations:

- Lines map to lines
- Parallel lines do not necessarily remain parallel
- Ratios are not preserved
- Closed under composition
- Models change of basis
- Projective matrix is defined up to a scale (8 DOF)

Camera (projection) matrix



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

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

$\mathbf{K}$: Intrinsic Matrix (3x3)

$\mathbf{R}$: Rotation (3x3)

$\mathbf{t}$: Translation (3x1)

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

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

Homogeneous coordinates

- Allows matrix form for 3D transformations
- Take rotation

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

- Now add translation

$$\begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \begin{bmatrix} t_1 \\ t_2 \\ t_3 \end{bmatrix}$$

Homogeneous coordinates

- Allows matrix form for 3D transformations

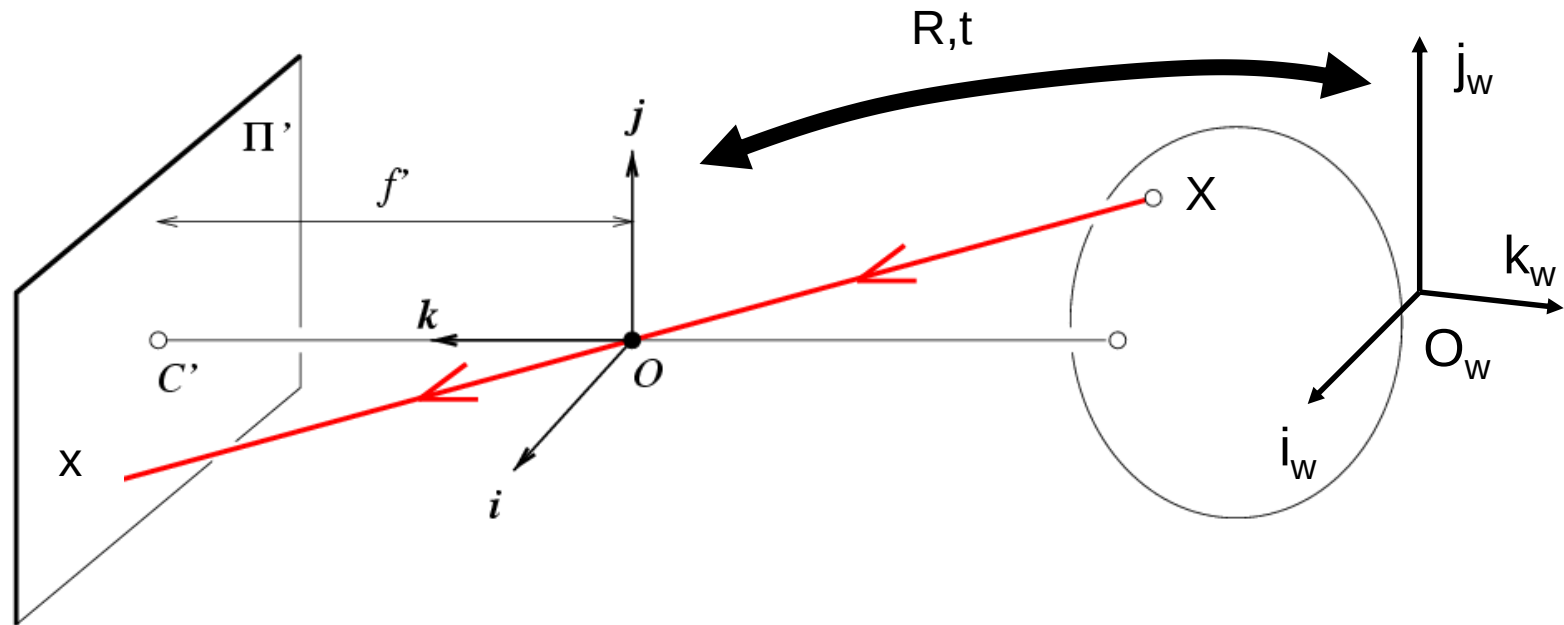
- Take rotation

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

- Now add translation with homogeneous coordinate

$$\begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Camera (projection) matrix



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

$\underbrace{\begin{bmatrix} \mathbf{R} & \mathbf{t} \end{bmatrix}}_{\text{Extrinsic Matrix}}$

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

$\mathbf{K}$: Intrinsic Matrix (3×3)

$\mathbf{R}$: Rotation (3×3)

$\mathbf{t}$: Translation (3×1)

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

Intrinsic and extrinsic matrices

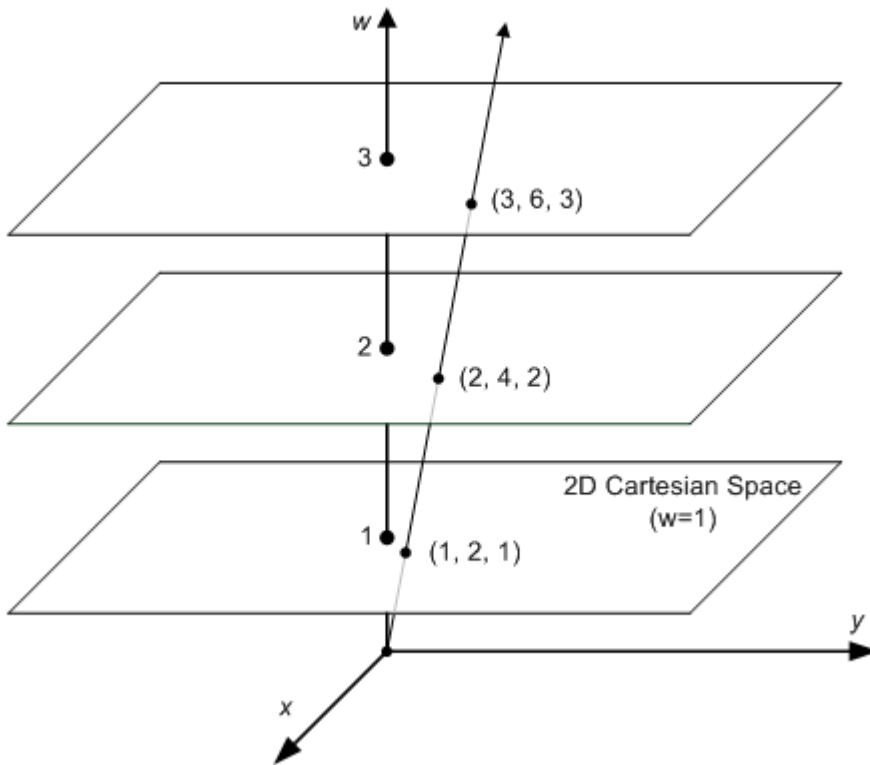
$$\mathbf{x} = \mathbf{K}[\mathbf{R} \quad \mathbf{t}] \mathbf{X}$$



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

Homogeneous coordinates

- Projective
- Point becomes a line



To homogeneous

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

From homogeneous

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

How to calibrate the camera?
(also called “camera resectioning”)

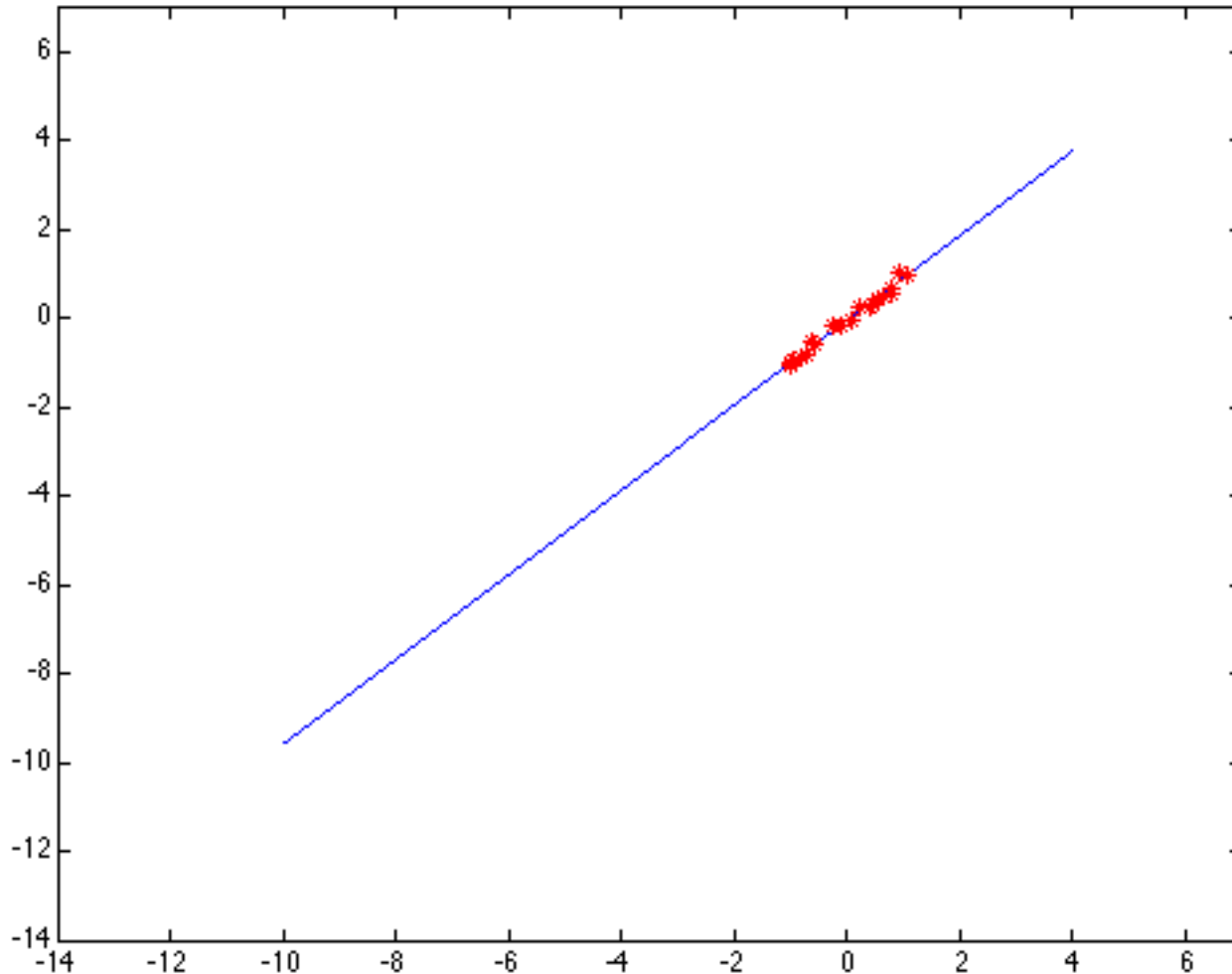
$$\mathbf{x} = \mathbf{K}[\mathbf{R} \quad \mathbf{t}] \mathbf{X}$$

$$\mathbf{x} = \mathbf{M} \mathbf{X}$$

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

Linear least-squares regression!

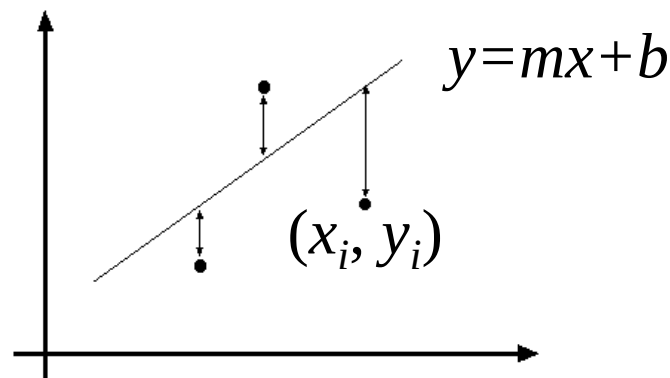
Simple example: Fitting a line



Least squares line fitting

- Data: $(x_1, y_1), \dots, (x_n, y_n)$
- Line equation: $y_i = mx_i + b$
- Find (m, b) to minimize

$$E = \sum_{i=1}^n (y_i - mx_i - b)^2$$



$$E = \sum_{i=1}^n \left(\begin{bmatrix} x_i & 1 \end{bmatrix} \begin{bmatrix} m \\ b \end{bmatrix} - y_i \right)^2 = \left\| \begin{bmatrix} x_1 \\ \vdots \\ x_n \\ 1 \end{bmatrix} \begin{bmatrix} m \\ b \end{bmatrix} - \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} \right\|^2 = \|\mathbf{A}\mathbf{p} - \mathbf{y}\|^2$$

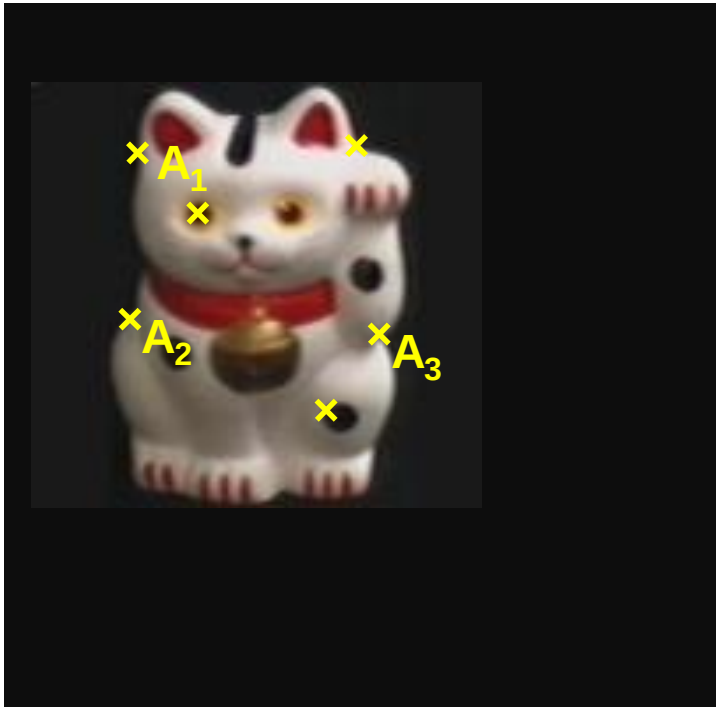
$$= \mathbf{y}^T \mathbf{y} - 2(\mathbf{A}\mathbf{p})^T \mathbf{y} + (\mathbf{A}\mathbf{p})^T (\mathbf{A}\mathbf{p})$$

$$\frac{dE}{dp} = 2\mathbf{A}^T \mathbf{A}\mathbf{p} - 2\mathbf{A}^T \mathbf{y} = 0$$

Matlab: `p = A \ y;`

$$\mathbf{A}^T \mathbf{A}\mathbf{p} = \mathbf{A}^T \mathbf{y} \Rightarrow \mathbf{p} = (\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T \mathbf{y} \quad (\text{Closed form solution})$$

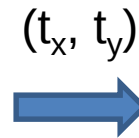
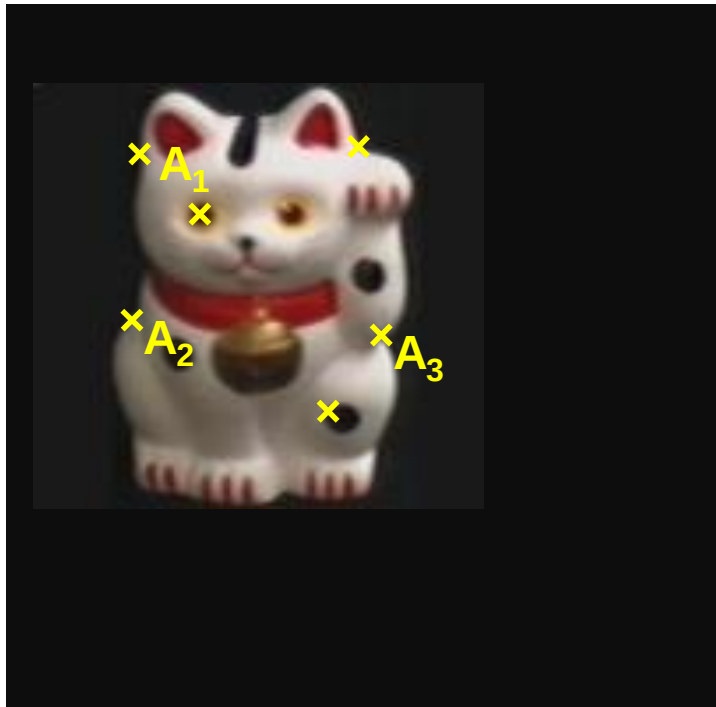
Example: solving for translation



Given matched points in $\{A\}$ and $\{B\}$, estimate the translation of the object

$$\begin{bmatrix} x_i^B \\ y_i^B \end{bmatrix} = \begin{bmatrix} x_i^A \\ y_i^A \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

Example: solving for translation

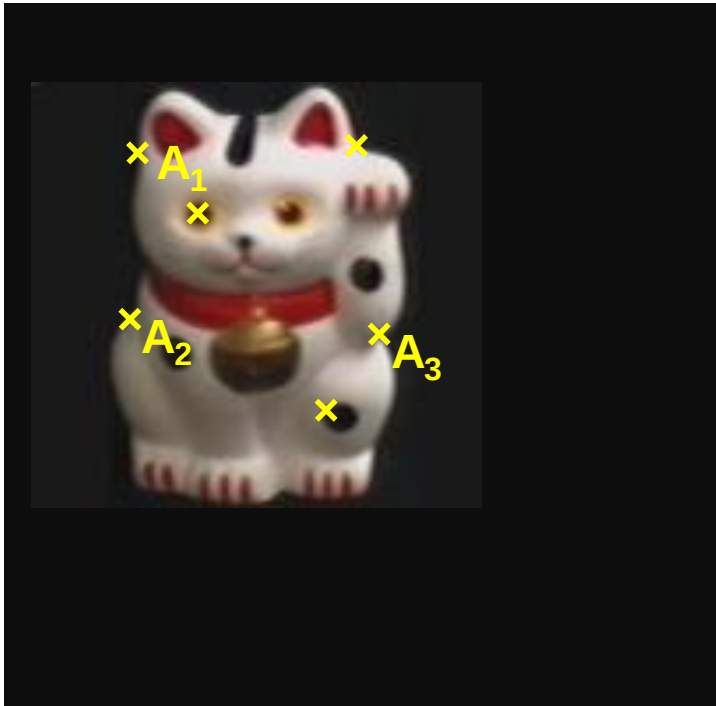


Least squares setup

$$\begin{bmatrix} x_i^B \\ y_i^B \end{bmatrix} = \begin{bmatrix} x_i^A \\ y_i^A \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ \vdots & \vdots \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} t_x \\ t_y \end{bmatrix} = \begin{bmatrix} x_1^B - x_1^A \\ y_1^B - y_1^A \\ \vdots \\ x_n^B - x_n^A \\ y_n^B - y_n^A \end{bmatrix}$$

Example: discovering rot/trans/scale

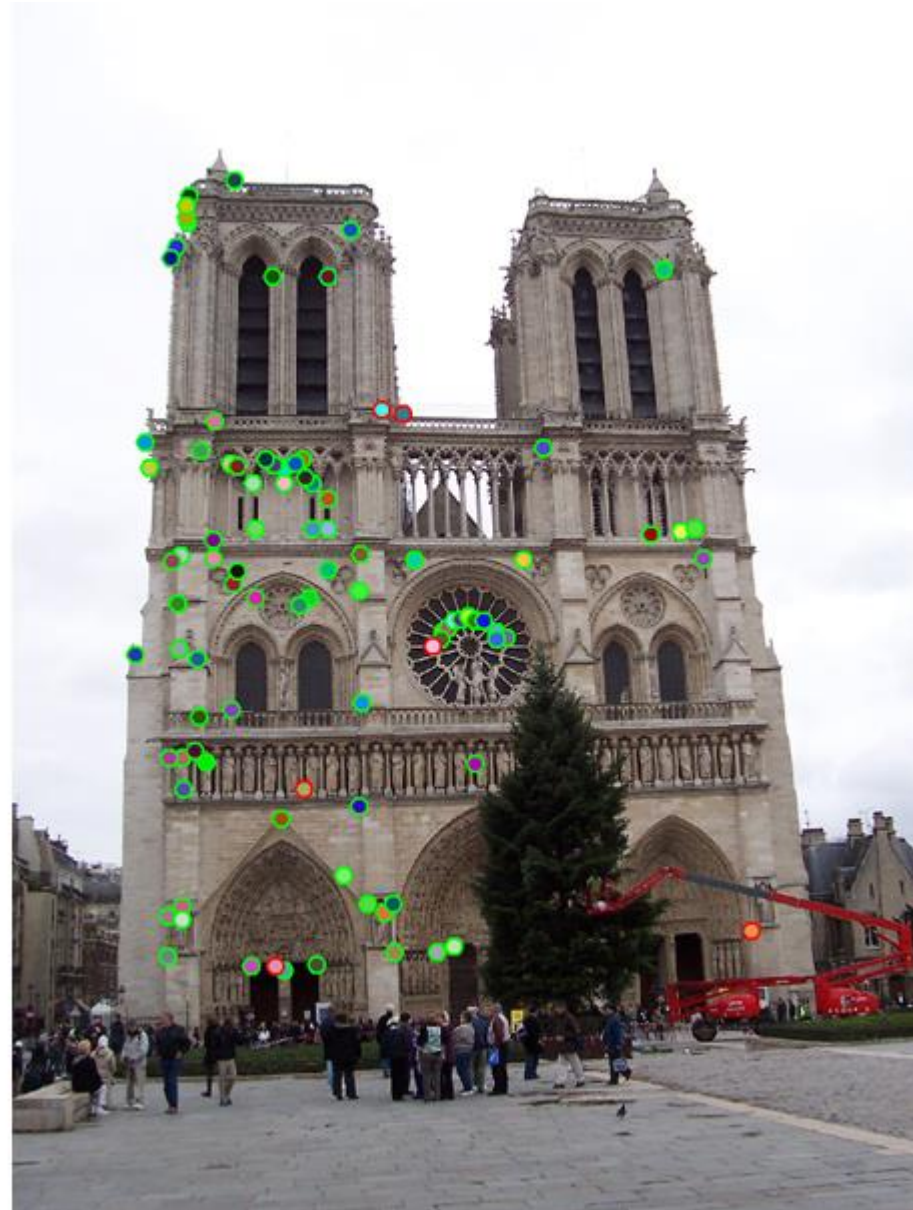
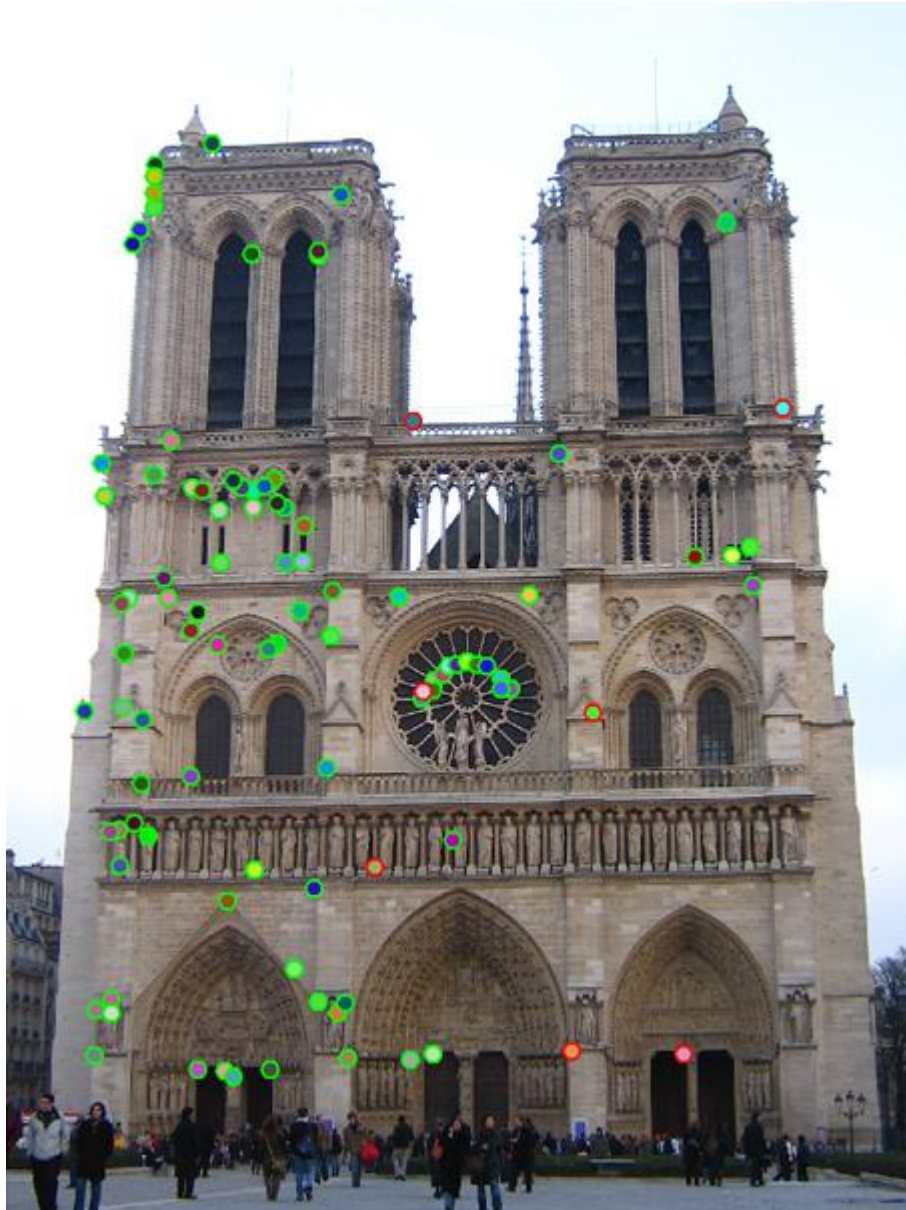


Given matched points in $\{A\}$ and $\{B\}$, estimate the transformation matrix

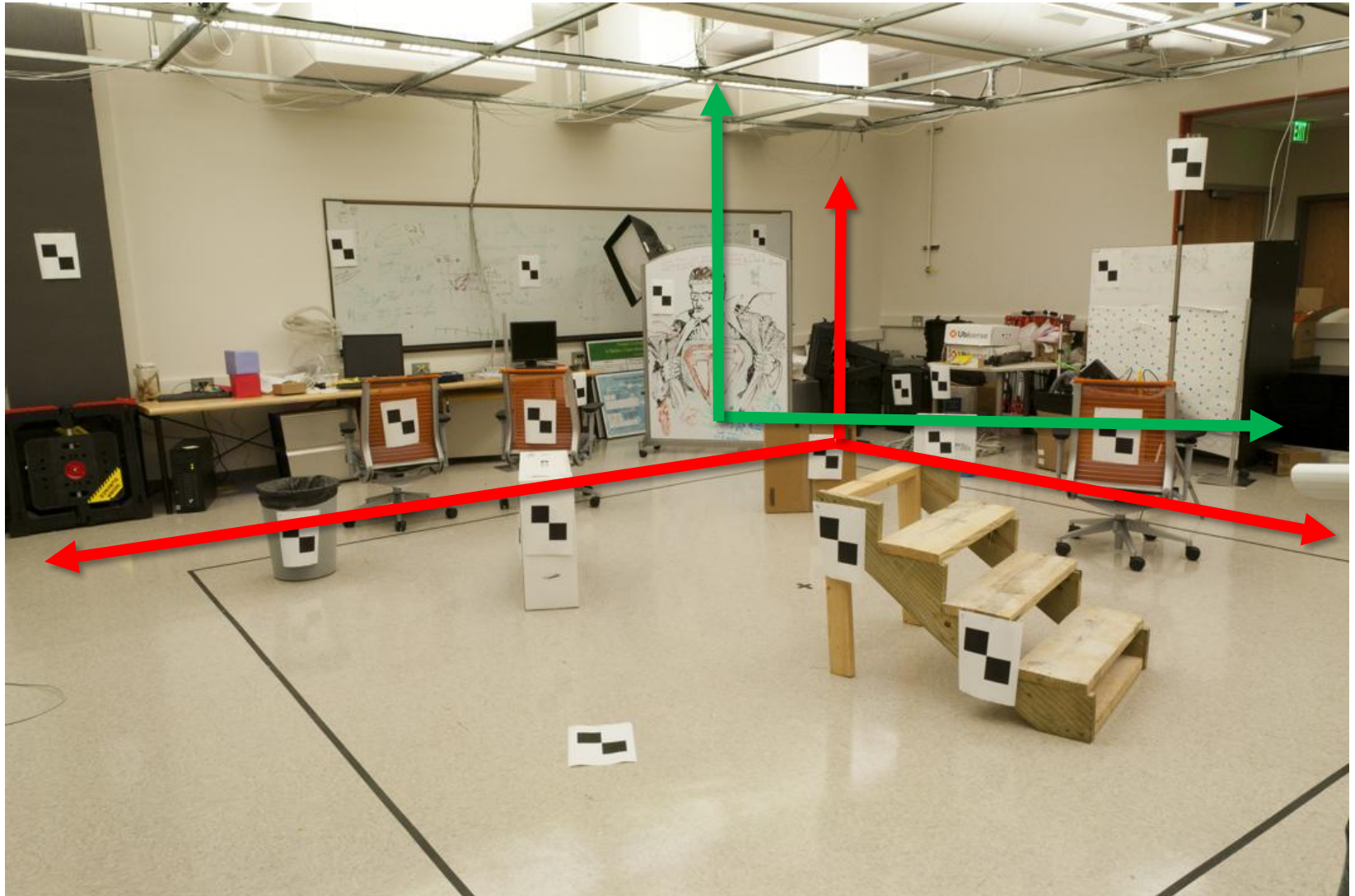
$$\begin{bmatrix} x_i^B \\ y_i^B \end{bmatrix} = T \begin{bmatrix} x_i^A \\ y_i^A \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

$$T = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

Are these transformations enough?



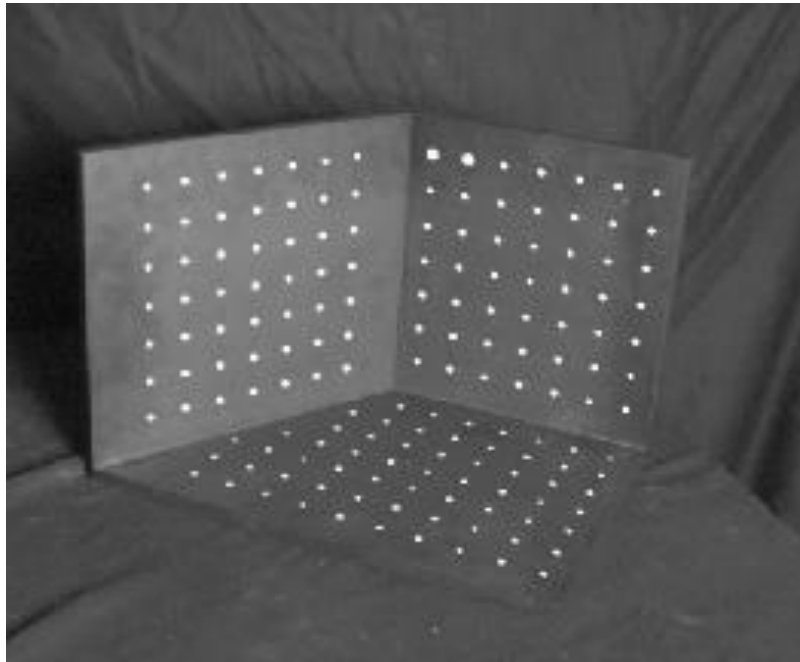
World vs Camera coordinates



Calibrating the Camera

Use an scene with **known** geometry

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



Known 2d
image coords

Known 3d
world locations

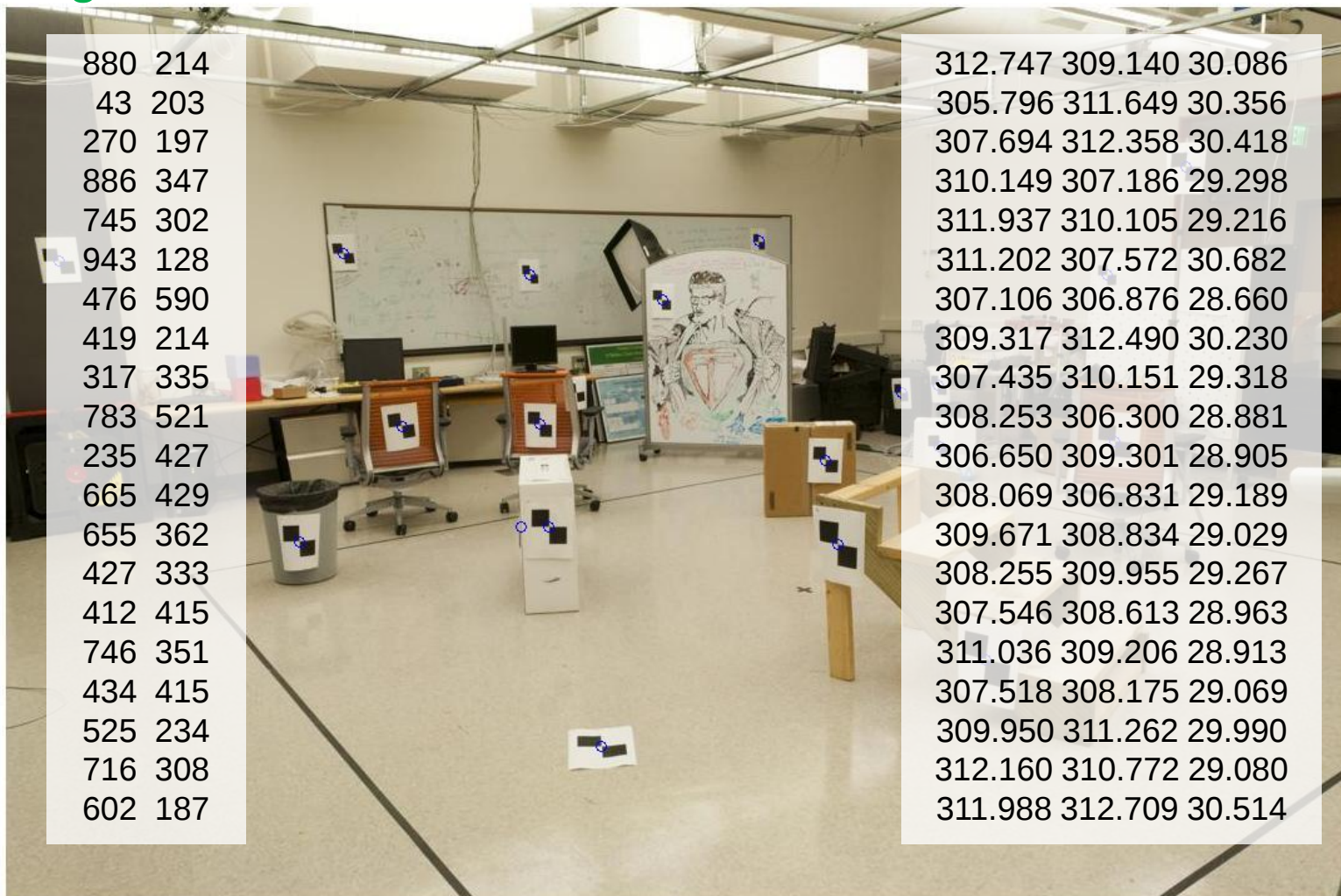
$$\begin{matrix} \downarrow & & \downarrow \\ \begin{bmatrix} su \\ sv \\ s \end{bmatrix} & = & \begin{matrix} \mathbf{M} \\ \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \end{bmatrix} \end{matrix} & \begin{matrix} \downarrow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

Unknown Camera Parameters

How do we calibrate a camera?

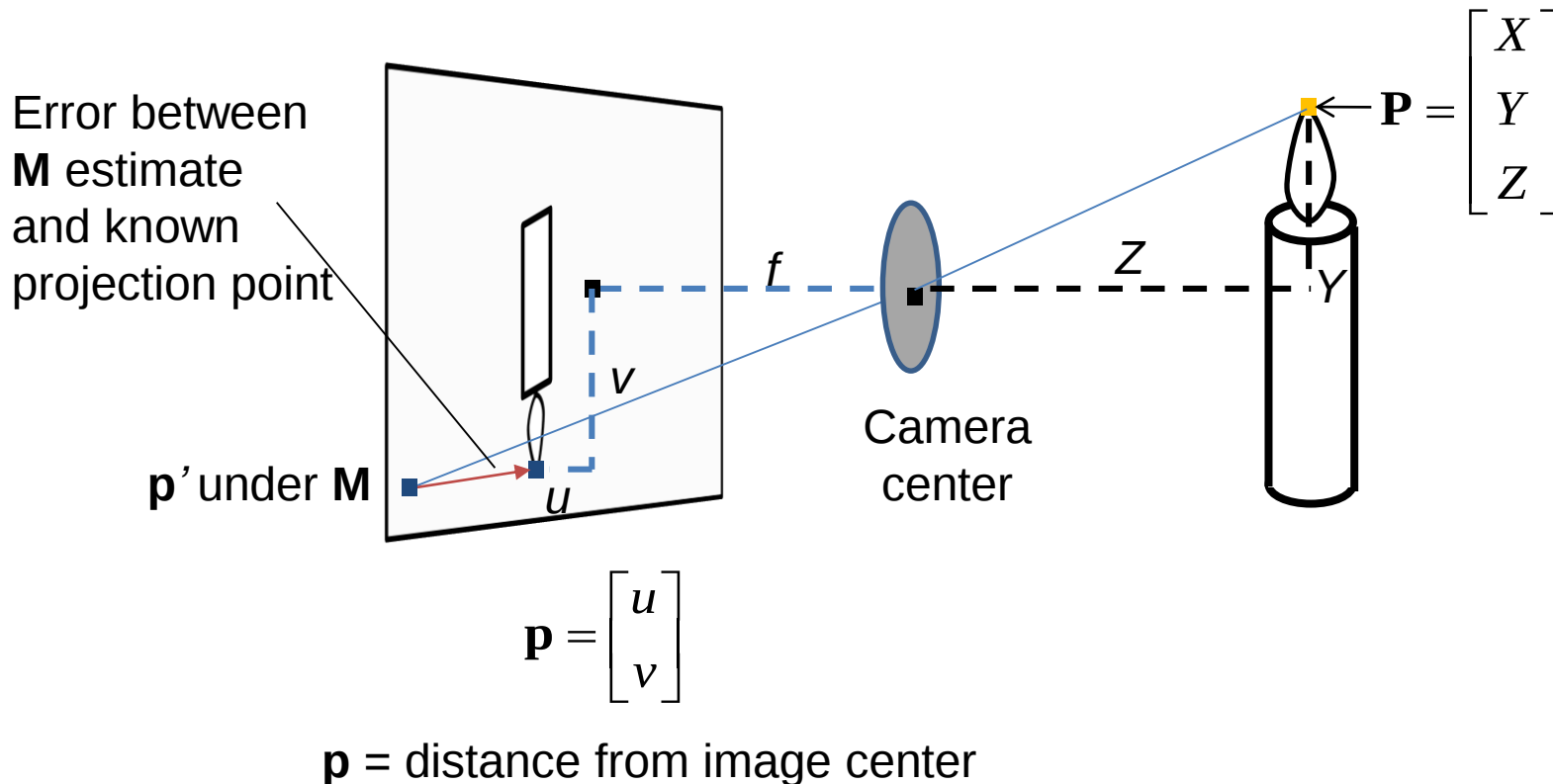
Known 2d
image coords

Known 3d
world locations



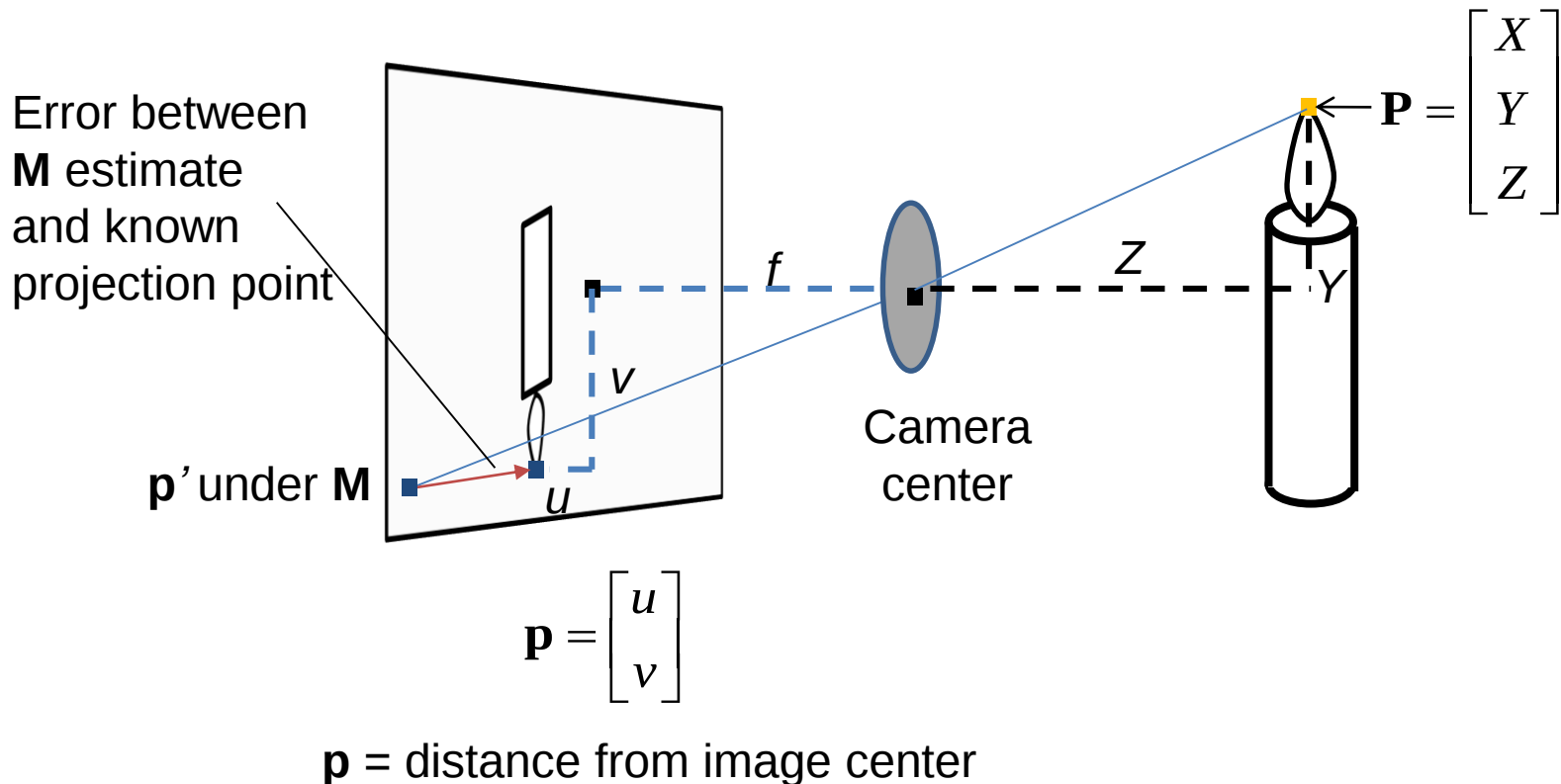
What is least squares doing?

- Given 3D point evidence, find best **M** which minimizes error between estimate (**p'**) and known corresponding 2D points (**p**).



What is least squares doing?

- Best $\mathbf{M}$ occurs when $\mathbf{p}' = \mathbf{p}$, or when $\mathbf{p}' - \mathbf{p} = 0$
- Form these equations from all point evidence
- Solve for model via closed-form regression



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

First, work out
where X,Y,Z
projects to under
candidate **M**.

$$su = m_{11}X + m_{12}Y + m_{13}Z + m_{14}$$

$$sv = m_{21}X + m_{22}Y + m_{23}Z + m_{24}$$

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

Two equations
per 3D point
correspondence

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

$$v = \frac{m_{21}X + m_{22}Y + m_{23}Z + m_{24}}{m_{31}X + m_{32}Y + m_{33}Z + m_{34}}$$

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

Next, rearrange into form
where all **M** coefficients are
individually stated in terms
of X,Y,Z,u,v.

-> Allows us to form lsq
matrix.

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

$$v = \frac{m_{21}X + m_{22}Y + m_{23}Z + m_{24}}{m_{31}X + m_{32}Y + m_{33}Z + m_{34}}$$

$$(m_{31}X + m_{32}Y + m_{33}Z + m_{34})u = m_{11}X + m_{12}Y + m_{13}Z + m_{14}$$

$$(m_{31}X + m_{32}Y + m_{33}Z + m_{34})v = m_{21}X + m_{22}Y + m_{23}Z + m_{24}$$

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

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

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

Next, rearrange into form
where all **M** coefficients are
individually stated in terms
of X,Y,Z,u,v.

-> Allows us to form lsq
matrix.

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

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

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

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

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

- Finally, solve for m's entries using linear least squares
- Method 1 – **Ax=b** form

$$\begin{bmatrix} X_1 & Y_1 & Z_1 & 1 & 0 & 0 & 0 & 0 & -u_1 X_1 & -u_1 Y_1 & -u_1 Z_1 \\ 0 & 0 & 0 & 0 & X_1 & Y_1 & Z_1 & 1 & -v_1 X_1 & -v_1 Y_1 & -v_1 Z_1 \\ & & & & & & \vdots & & & & \\ X_n & Y_n & Z_n & 1 & 0 & 0 & 0 & 0 & -u_n X_n & -u_n Y_n & -u_n Z_n \\ 0 & 0 & 0 & 0 & X_n & Y_n & Z_n & 1 & -v_n X_n & -v_n Y_n & -v_n Z_n \end{bmatrix} \begin{bmatrix} m_{11} \\ m_{12} \\ m_{13} \\ m_{14} \\ m_{21} \\ m_{22} \\ m_{23} \\ m_{24} \\ m_{31} \\ m_{32} \\ m_{33} \end{bmatrix} = \begin{bmatrix} u_1 \\ v_1 \\ \vdots \\ u_n \\ v_n \end{bmatrix}$$

$M = A \setminus Y;$
 $M = [M; 1];$
 $M = \text{reshape}(M, [], 3)';$

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

- Or, solve for m's entries using total linear least-squares.
- Method 2 – **Ax=0** form

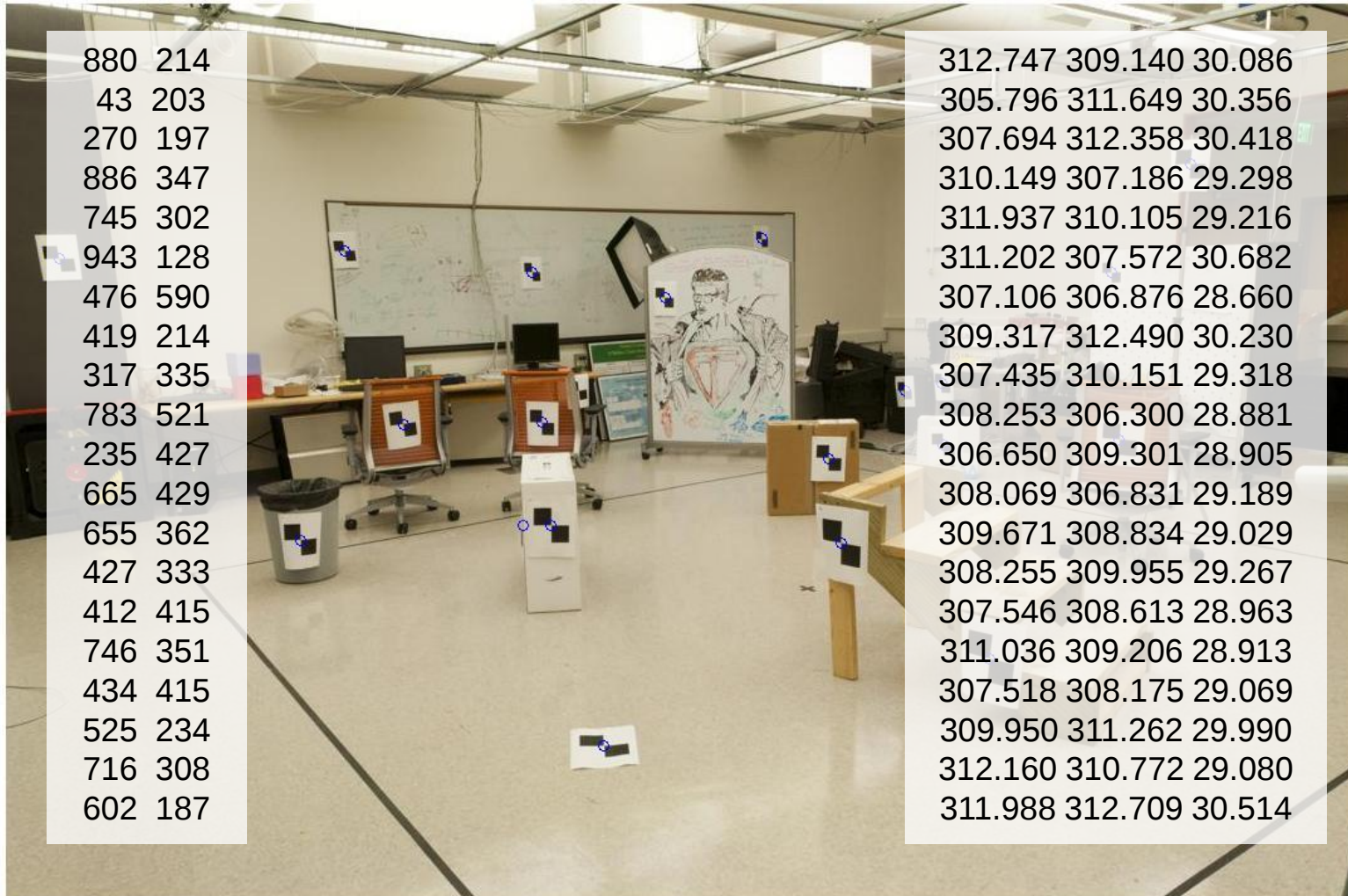
$$\begin{bmatrix} X_1 & Y_1 & Z_1 & 1 & 0 & 0 & 0 & 0 & -u_1 X_1 & -u_1 Y_1 & -u_1 Z_1 & -u_1 \\ 0 & 0 & 0 & 0 & X_1 & Y_1 & Z_1 & 1 & -v_1 X_1 & -v_1 Y_1 & -v_1 Z_1 & -v_1 \\ & & & & \vdots & & & & & & & \\ X_n & Y_n & Z_n & 1 & 0 & 0 & 0 & 0 & -u_n X_n & -u_n Y_n & -u_n Z_n & -u_n \\ 0 & 0 & 0 & 0 & X_n & Y_n & Z_n & 1 & -v_n X_n & -v_n Y_n & -v_n Z_n & -v_n \end{bmatrix} \begin{bmatrix} m_{11} \\ m_{12} \\ m_{13} \\ m_{14} \\ m_{21} \\ m_{22} \\ m_{23} \\ m_{24} \\ m_{31} \\ m_{32} \\ m_{33} \\ m_{34} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}$$

$[U, S, V] = \text{svd}(A);$
 $M = V(:, \text{end});$
 $M = \text{reshape}(M, [], 3)';$

How do we calibrate a camera?

Known 2d
image coords

Known 3d
world locations



Known 2d image coords

Known 3d world locations



Projection error defined by two equations – one for u and one for v

$$\begin{bmatrix} 312.747 & 309.140 & 30.086 & 1 & 0 & 0 & 0 & 0 & -880 \times 312.747 & -880 \times 309.140 & -880 \times 30.086 & -880 \\ 0 & 0 & 0 & 0 & 312.747 & 309.140 & 30.086 & 1 & -214 \times 312.747 & -214 \times 309.140 & -214 \times 30.086 & -214 \\ & & & & \vdots & & & & & & & \\ X_n & Y_n & Z_n & 1 & 0 & 0 & 0 & 0 & -u_n X_n & -u_n Y_n & -u_n Z_n & -u_n \\ 0 & 0 & 0 & 0 & X_n & Y_n & Z_n & 1 & -v_n X_n & -v_n Y_n & -v_n Z_n & -v_n \end{bmatrix} \begin{bmatrix} m_{11} \\ m_{12} \\ m_{13} \\ m_{14} \\ m_{21} \\ m_{22} \\ m_{23} \\ m_{24} \\ m_{31} \\ m_{32} \\ m_{33} \\ m_{34} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}$$

Known 2d image coords

Known 3d world locations

2nd point



Projection error defined by two equations – one for u and one for v

$$\begin{bmatrix} 312.747 & 309.140 & 30.086 & 1 & 0 & 0 & 0 & 0 & -880 \times 312.747 & -880 \times 309.140 & -880 \times 30.086 & -880 \\ 0 & 0 & 0 & 0 & 312.747 & 309.140 & 30.086 & 1 & -214 \times 312.747 & -214 \times 309.140 & -214 \times 30.086 & -214 \\ 305.796 & 311.649 & 30.356 & 1 & 0 & 0 & 0 & 0 & -43 \times 305.796 & -43 \times 311.649 & -43 \times 30.356 & -43 \\ 0 & 0 & 0 & 0 & 305.796 & 311.649 & 30.356 & 1 & -203 \times 305.796 & -203 \times 311.649 & -203 \times 30.356 & -203 \\ & & & & \vdots & & & & & & & \\ X_n & Y_n & Z_n & 1_n & 0 & 0 & 0 & 0 & -u_n X_n & -u_n Y_n & -u_n Z_n & -u_n \\ 0 & 0 & 0 & 0 & X_n & Y_n & Z_n & 1 & -v_n X_n & -v_n Y_n & -v_n Z_n & -v_n \end{bmatrix} \begin{bmatrix} m_{11} \\ m_{12} \\ m_{13} \\ m_{14} \\ m_{21} \\ m_{22} \\ m_{23} \\ m_{24} \\ m_{31} \\ m_{32} \\ m_{33} \\ m_{34} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

How many points do I need to fit the model?

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



Degrees of freedom?

$$w \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{matrix} & \overset{5}{\begin{bmatrix} \alpha & s & u_0 \\ 0 & \beta & v_0 \\ 0 & 0 & 1 \end{bmatrix}} & \overset{6}{\underbrace{\begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}}_{\text{Rotation}}} \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} \end{matrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Think 3:

- Rotation around x
- Rotation around y
- Rotation around z

How many points do I need to fit the model?

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



Degrees of freedom?

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

$\mathbf{M}$ is 3x4, so 12 unknowns, but projective scale ambiguity – 11 deg. freedom.
One equation per unknown $\rightarrow$ 5 1/2 point correspondences determines a solution (e.g., either u or v).

More than 5 1/2 point correspondences $\rightarrow$ overdetermined, many solutions to $\mathbf{M}$.
Least squares is finding the solution that best satisfies the overdetermined system.

Why use more than 6? Robustness to error in feature points.

Calibration with linear method

- Advantages
 - Easy to formulate and solve
 - Provides initialization for non-linear methods
- Disadvantages
 - Doesn't directly give you camera parameters
 - Doesn't model radial distortion
 - Can't impose constraints, such as known focal length
- Non-linear methods are preferred
 - Define error as difference between projected points and measured points
 - Minimize error using Newton's method or other non-linear optimization

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

- Yes!
- We can directly solve for the individual entries of $K [R \mid T]$.

$\mathbf{a}_n$ = nth
column of A

Extracting camera parameters

$$\frac{M}{\rho} = \begin{pmatrix} \boxed{\alpha \mathbf{r}_1^T - \alpha \cot \theta \mathbf{r}_2^T + u_0 \mathbf{r}_3^T} & \boxed{\alpha t_x - \alpha \cot \theta t_y + u_0 t_z} \\ \boxed{\frac{\beta}{\sin \theta} \mathbf{r}_2^T + v_0 \mathbf{r}_3^T} & \boxed{\frac{\beta}{\sin \theta} t_y + v_0 t_z} \\ \boxed{\mathbf{r}_3^T} & \boxed{t_z} \end{pmatrix} = K \begin{bmatrix} R & T \end{bmatrix}$$

A $\mathbf{b}$

$$K = \begin{bmatrix} \alpha & -\alpha \cot \theta & u_0 \\ 0 & \frac{\beta}{\sin \theta} & v_0 \\ 0 & 0 & 1 \end{bmatrix}$$

Box 1

$$A = \begin{bmatrix} \mathbf{a}_1^T \\ \mathbf{a}_2^T \\ \mathbf{a}_3^T \end{bmatrix} \quad \mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

Estimated values

Intrinsic

$$\rho = \frac{\pm 1}{|\mathbf{a}_3|} \quad \begin{aligned} u_o &= \rho^2 (\mathbf{a}_1 \cdot \mathbf{a}_3) \\ v_o &= \rho^2 (\mathbf{a}_2 \cdot \mathbf{a}_3) \end{aligned}$$

$$\cos \theta = \frac{(\mathbf{a}_1 \times \mathbf{a}_3) \cdot (\mathbf{a}_2 \times \mathbf{a}_3)}{|\mathbf{a}_1 \times \mathbf{a}_3| \cdot |\mathbf{a}_2 \times \mathbf{a}_3|}$$

Extracting camera parameters

$$\frac{\mathcal{M}}{\rho} = \left(\begin{array}{c|c} \alpha \mathbf{r}_1^T - \alpha \cot \theta \mathbf{r}_2^T + u_0 \mathbf{r}_3^T & \alpha t_x - \alpha \cot \theta t_y + u_0 t_z \\ \hline \frac{\beta}{\sin \theta} \mathbf{r}_2^T + v_0 \mathbf{r}_3^T & \frac{\beta}{\sin \theta} t_y + v_0 t_z \\ \hline \mathbf{r}_3^T & t_z \end{array} \right) = \mathbf{K} \begin{bmatrix} \mathbf{R} & \mathbf{T} \end{bmatrix}$$

A
b

$$\mathbf{A} = \begin{bmatrix} \mathbf{a}_1^T \\ \mathbf{a}_2^T \\ \mathbf{a}_3^T \end{bmatrix} \quad \mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

Estimated values

Intrinsic

$$\alpha = \rho^2 |\mathbf{a}_1 \times \mathbf{a}_3| \sin \theta$$

$$\beta = \rho^2 |\mathbf{a}_2 \times \mathbf{a}_3| \sin \theta$$

Extracting camera parameters

$$\frac{\mathcal{M}}{\rho} = \left(\begin{array}{c|c} \alpha \mathbf{r}_1^T - \alpha \cot \theta \mathbf{r}_2^T + u_0 \mathbf{r}_3^T & \alpha t_x - \alpha \cot \theta t_y + u_0 t_z \\ \hline \frac{\beta}{\sin \theta} \mathbf{r}_2^T + v_0 \mathbf{r}_3^T & \frac{\beta}{\sin \theta} t_y + v_0 t_z \\ \hline \mathbf{r}_3^T & t_z \end{array} \right) = \mathbf{K} \begin{bmatrix} \mathbf{R} & \mathbf{T} \end{bmatrix}$$

A
b

$$\mathbf{A} = \begin{bmatrix} \mathbf{a}_1^T \\ \mathbf{a}_2^T \\ \mathbf{a}_3^T \end{bmatrix} \quad \mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

Estimated values

Extrinsic

$$\mathbf{r}_1 = \frac{(\mathbf{a}_2 \times \mathbf{a}_3)}{|\mathbf{a}_2 \times \mathbf{a}_3|} \quad \mathbf{r}_3 = \frac{\pm \mathbf{a}_3}{|\mathbf{a}_3|}$$

$$\mathbf{r}_2 = \mathbf{r}_3 \times \mathbf{r}_1$$

$$\mathbf{T} = \rho \mathbf{K}^{-1} \mathbf{b}$$

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

- Yes!
- We can also use RQ factorization (not QR)
 - R in RQ is not rotation matrix R ; crossed names!
- R (right diagonal) is K
- Q (orthogonal basis) is R .
- T , the last column of $[R \mid T]$, is $\text{inv}(K) * \text{last column of } M$.
 - But you need to do a bit of post-processing to make sure that the matrices are valid. See <http://ksimek.github.io/2012/08/14/decompose/>

Recovering the camera center

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

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

This is not the camera center C .

It is $-\mathbf{RC}$, as the point is rotated before t_x , t_y , and t_z are added

So we need $-\mathbf{R}^{-1} \mathbf{K}^{-1} \mathbf{m}_4$ to get C .

$$\begin{bmatrix} su \\ sv \\ s \end{bmatrix} = \underbrace{\begin{bmatrix} * & * & * \\ * & * & * \\ * & * & * \end{bmatrix}}_{\mathbf{Q}} \underbrace{\begin{bmatrix} * \\ * \\ * \end{bmatrix}}_{\mathbf{m}_4} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

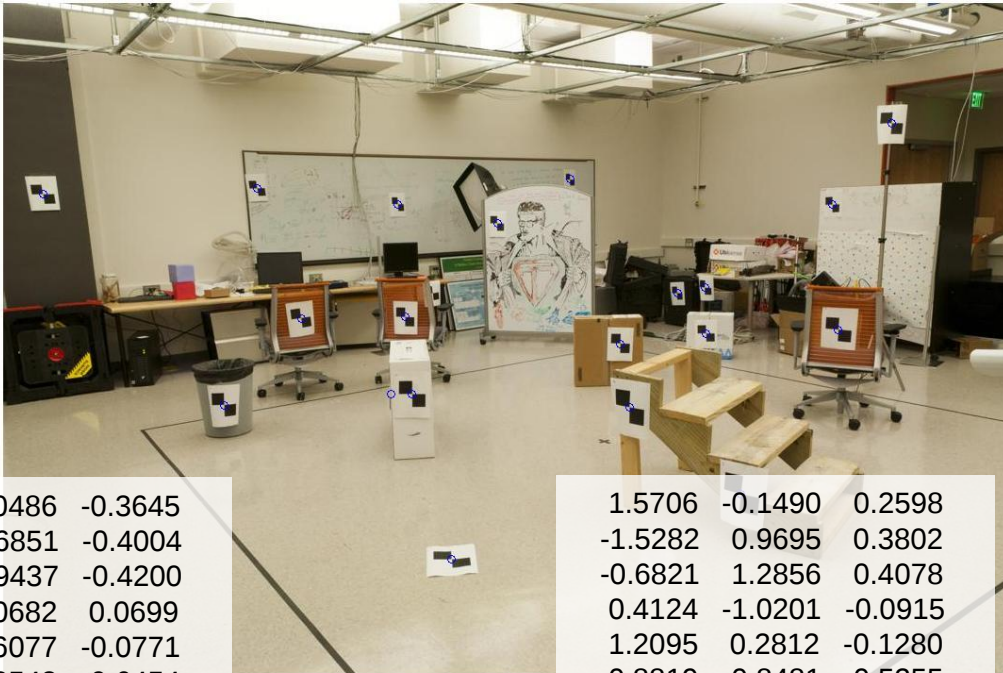
This is $\mathbf{t} \times \mathbf{K}$

So $\mathbf{K}^{-1} \mathbf{m}_4$ is $\mathbf{t}$

$\mathbf{Q}$ is $\mathbf{K} \times \mathbf{R}$.

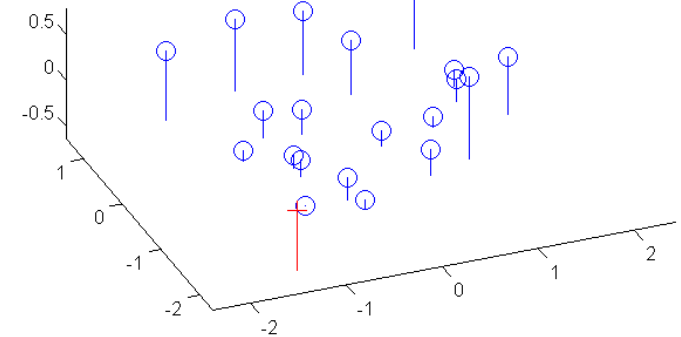
So we just need $-\mathbf{Q}^{-1} \mathbf{m}_4$

Estimate of camera center

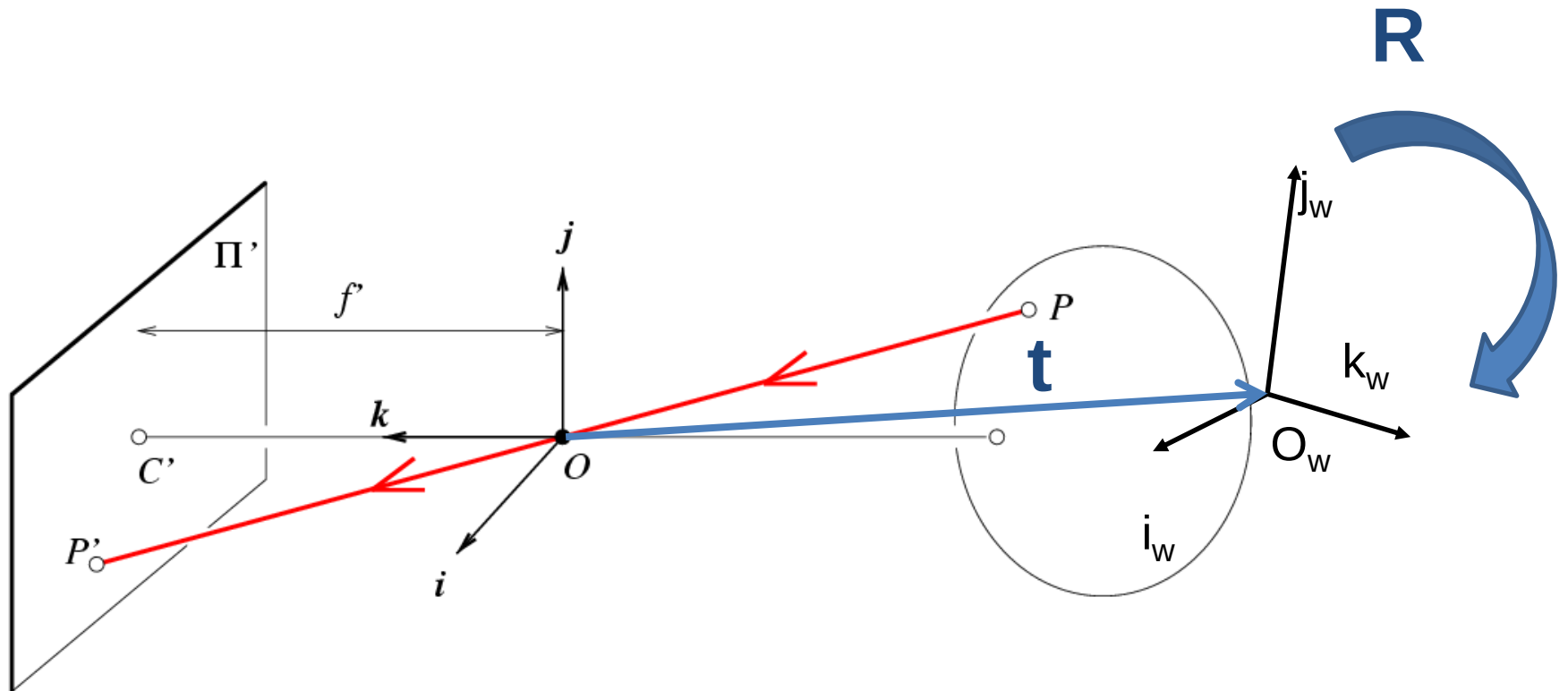


1.0486	-0.3645
-1.6851	-0.4004
-0.9437	-0.4200
1.0682	0.0699
0.6077	-0.0771
1.2543	-0.6454
-0.2709	0.8635
-0.4571	-0.3645
-0.7902	0.0307
0.7318	0.6382
-1.0580	0.3312
0.3464	0.3377
0.3137	0.1189
-0.4310	0.0242
-0.4799	0.2920
0.6109	0.0830
-0.4081	0.2920
-0.1109	-0.2992
0.5129	-0.0575
0.1406	-0.4527

1.5706	-0.1490	0.2598
-1.5282	0.9695	0.3802
-0.6821	1.2856	0.4078
0.4124	-1.0201	-0.0915
1.2095	0.2812	-0.1280
0.8819	-0.8481	0.5255
-0.9442	-1.1583	-0.3759
0.0415	1.3445	0.3240
-0.7975	0.3017	-0.0826
-0.4329	-1.4151	-0.2774
-1.1475	-0.0772	-0.2667
-0.5149	-1.1784	-0.1401
0.1993	-0.2854	-0.2114
-0.4320	0.2143	-0.1053
-0.7481	-0.3840	-0.2408
0.8078	-0.1196	-0.2631
-0.7605	-0.5792	-0.1936
0.3237	0.7970	0.2170
1.3089	0.5786	-0.1887
1.2323	1.4421	0.4506



Oriented and Translated Camera



ONE DIFFICULT EXAMPLE...

