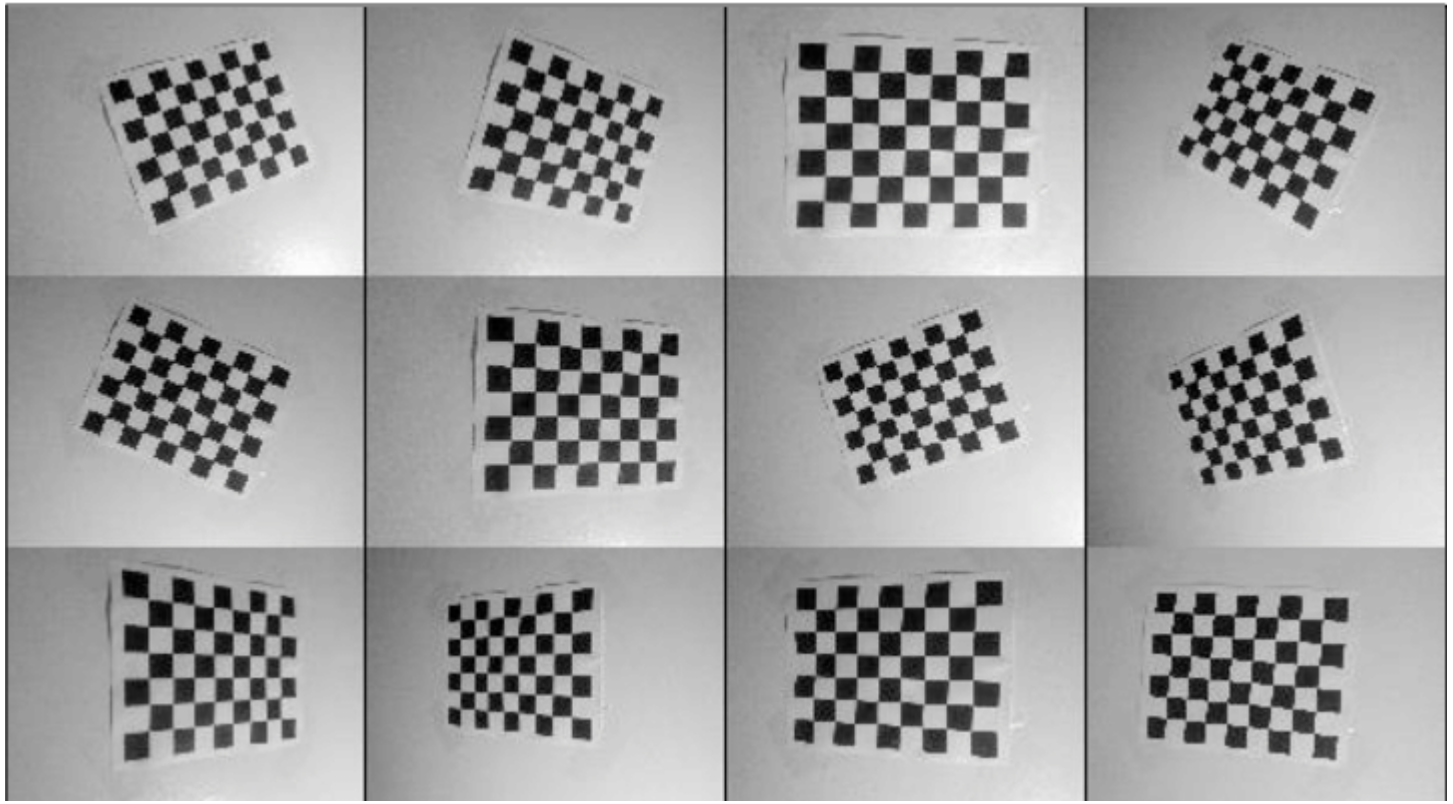
Introduction to Computer Vision

Kaveh Fathian
Assistant Professor
Computer Science Department
Colorado School of Mines

Lecture 11

Geometric Camera Calibration

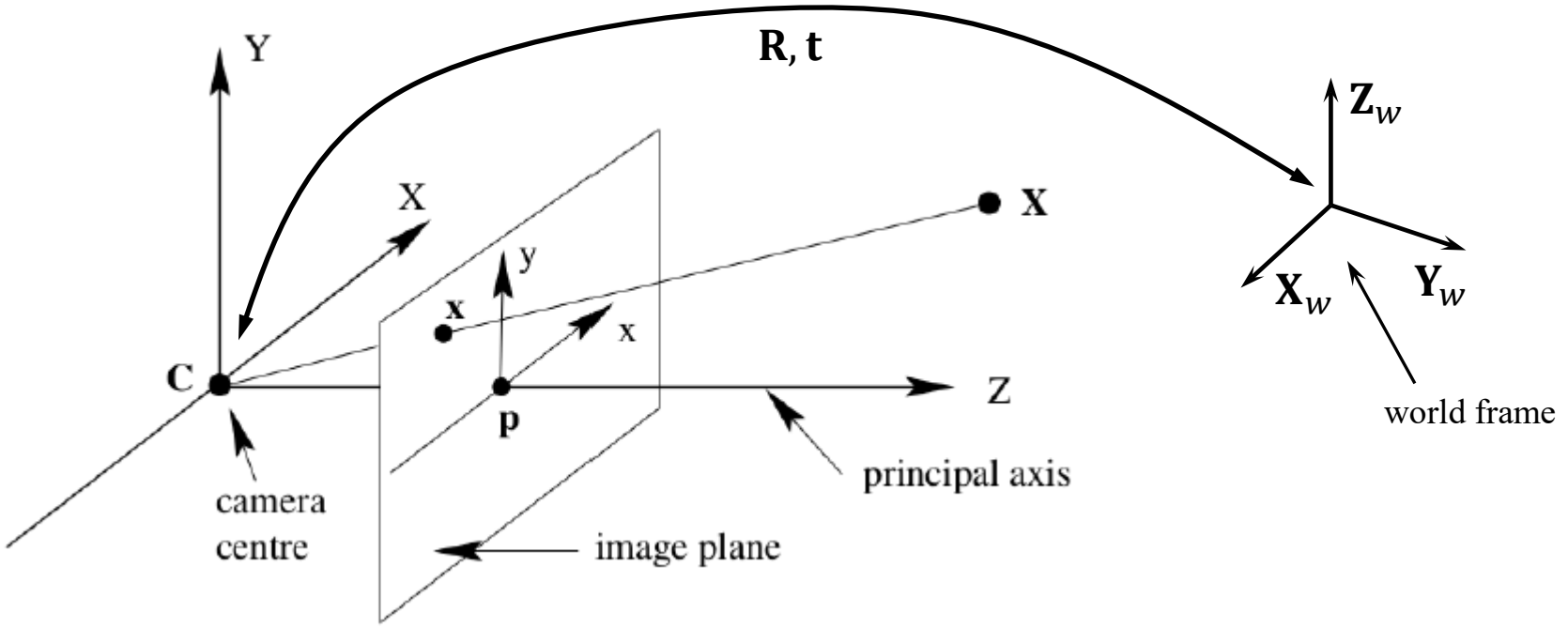


What does it mean to “calibrate a camera”?

Many different ways to calibrate a camera:

- Radiometric calibration
- Color calibration
- Geometric calibration
- Noise calibration
- Lens (or aberration) calibration

Camera Matrix

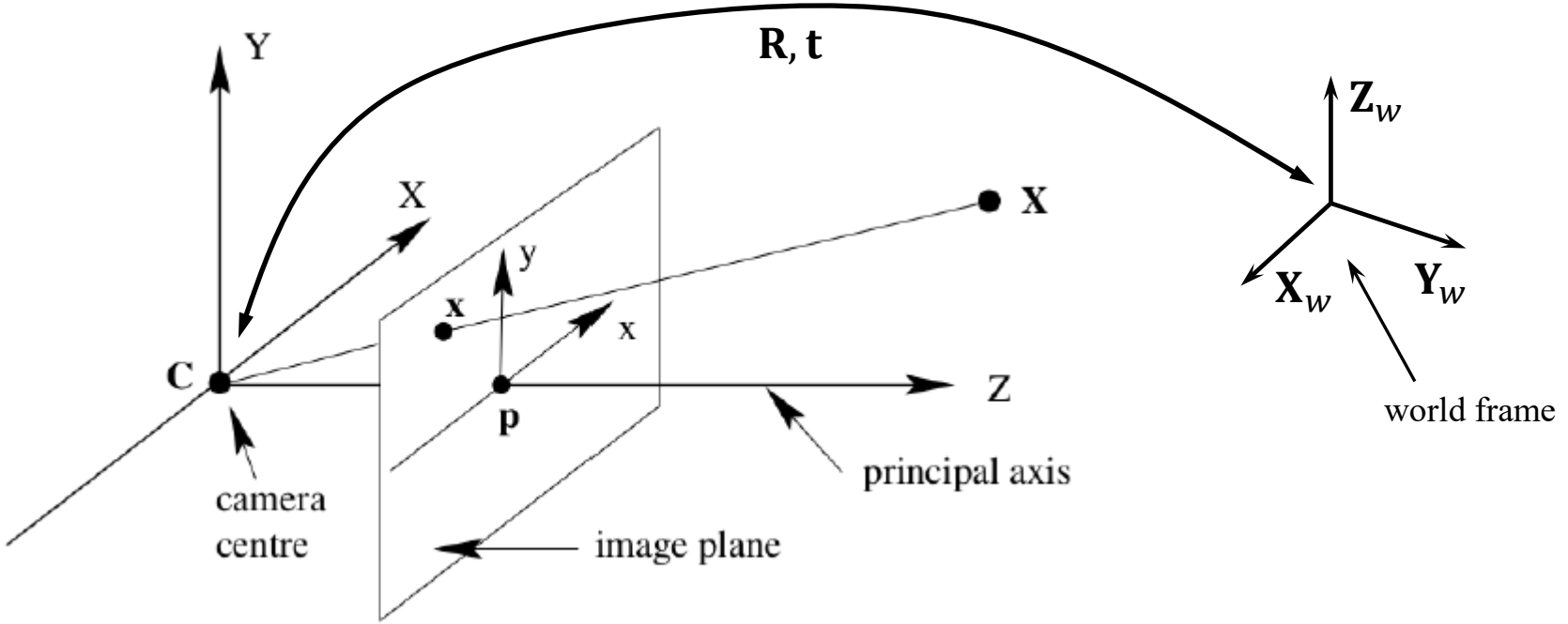


$$\mathbf{x} = \mathbf{K}\underbrace{\begin{bmatrix}\mathbf{R} & \mathbf{t}\end{bmatrix}}\mathbf{X}$$

Extrinsic Matrix

- $\mathbf{x}$: 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)

Camera Matrix



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

Extrinsic Matrix

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

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

Simulate a Camera

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

Unknown Known Known

Camera Calibration

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

Known Unknown Known

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

Intrinsics Calibration

Extrinsics Calibration / Pose Estimation

3D Reconstruction I

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

Known Known Unknown

3D Reconstruction II

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

Known Unknown Unknown

Camera Calibration

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

Known Unknown Known

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

Intrinsics Calibration

Extrinsics Calibration / Pose Estimation

Camera Calibration / Resectioning

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

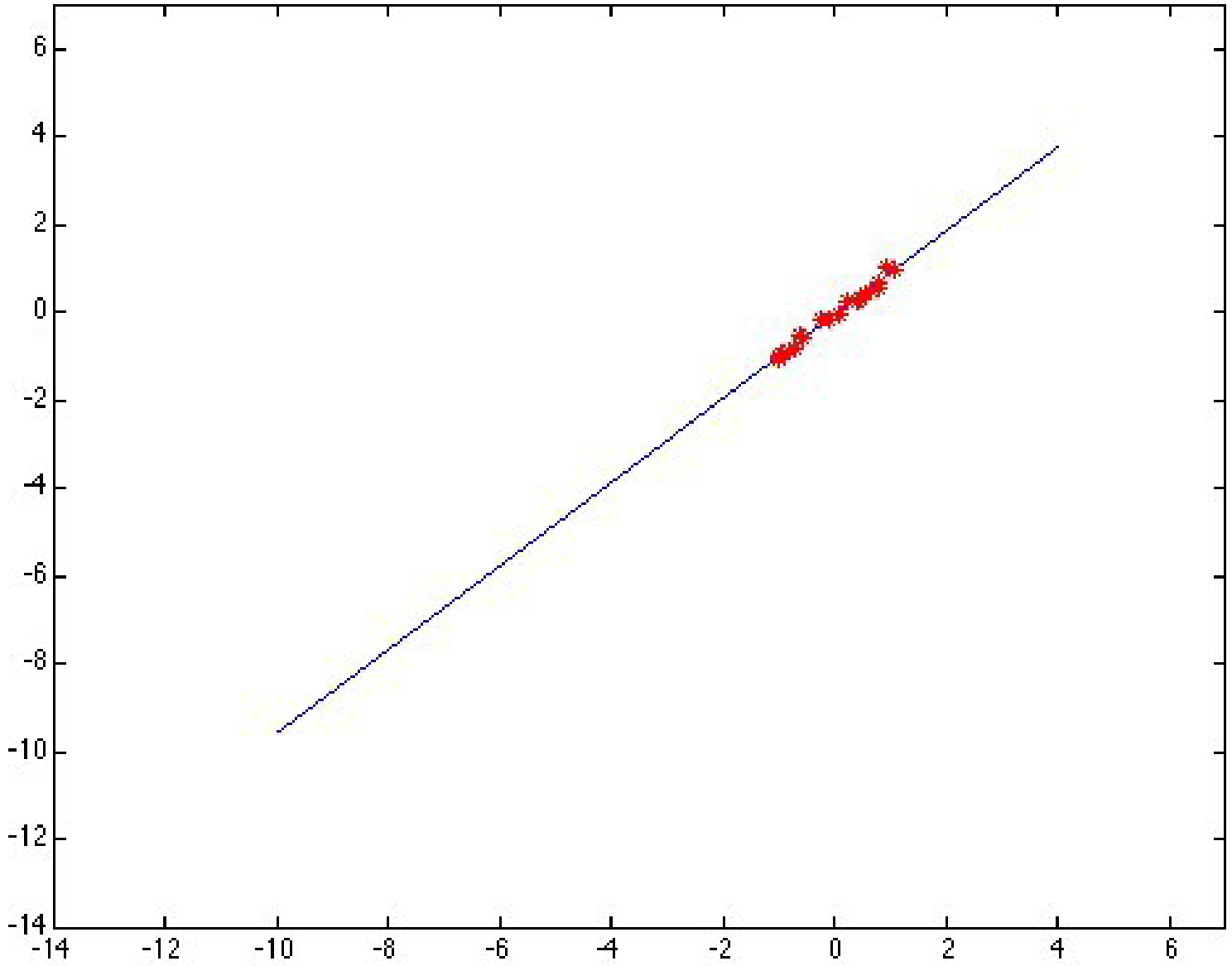
$$\mathbf{x} = \mathbf{M}\mathbf{X}$$

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

Linear least-squares regression!

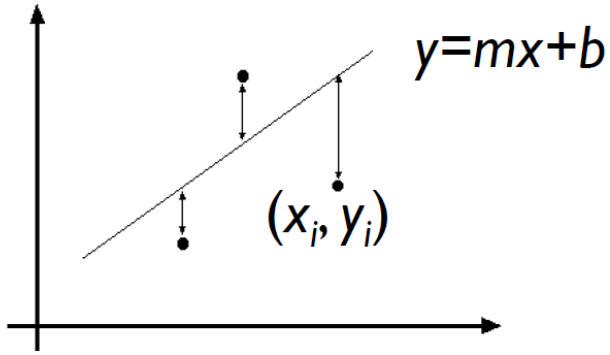
Linear Least-Squares Review

Linear Regression: Fitting a line



Linear Regression: Fitting a line

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



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

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

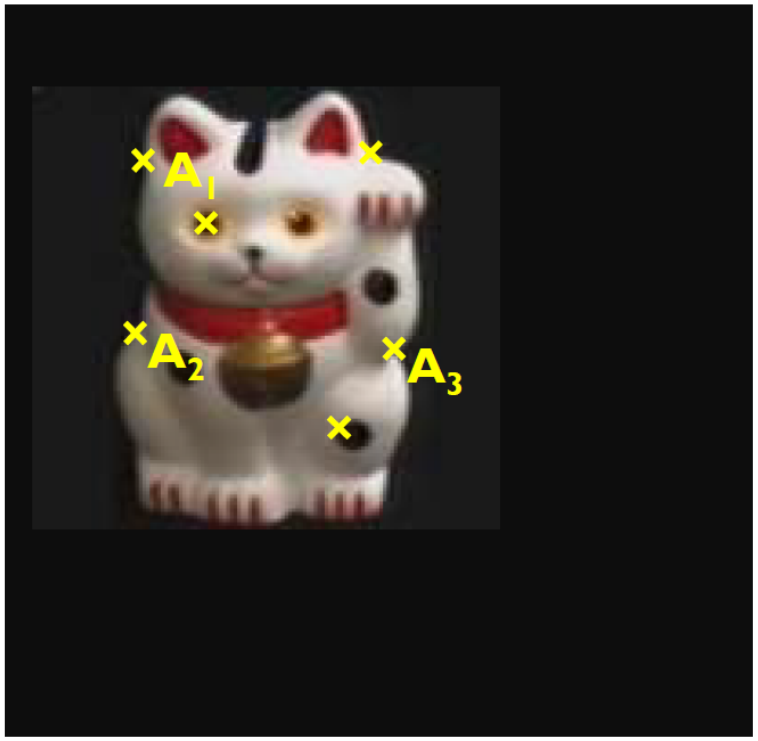
$$= \mathbf{y}^T\mathbf{y} - 2(\mathbf{Ap})^T\mathbf{y} + (\mathbf{Ap})^T(\mathbf{Ap})$$

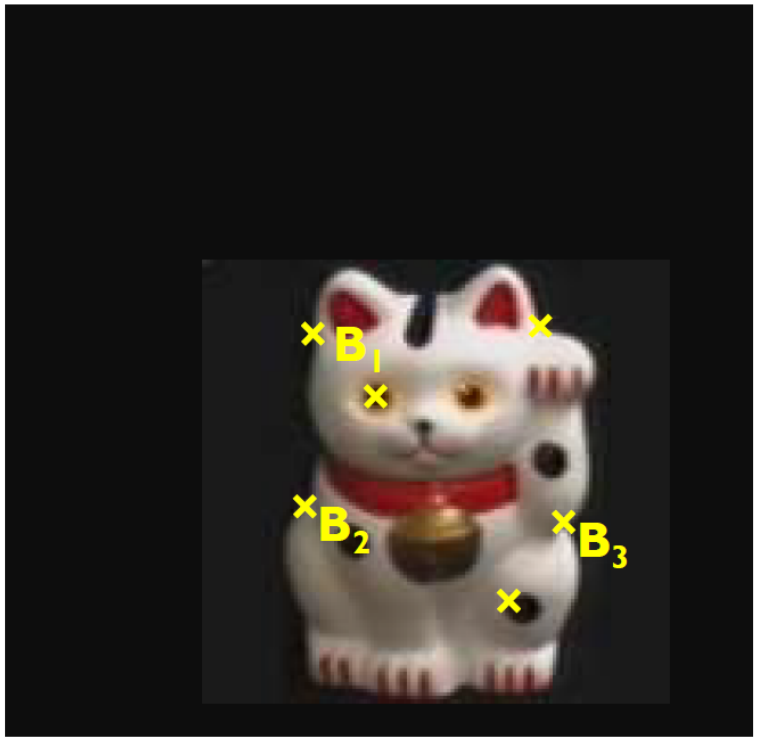
$$\frac{dE}{dp} = 2\mathbf{A}^T\mathbf{Ap} - 2\mathbf{A}^T\mathbf{y} = 0$$

```
Python:
p = np.linalg.lstsq(A,y)[0]
```

$$\mathbf{A}^T\mathbf{Ap} = \mathbf{A}^T\mathbf{y} \Rightarrow \mathbf{p} = \left(\mathbf{A}^T\mathbf{A}\right)^{-1}\mathbf{A}^T\mathbf{y}$$ (Closed form solution)

Example: Solving for Translation

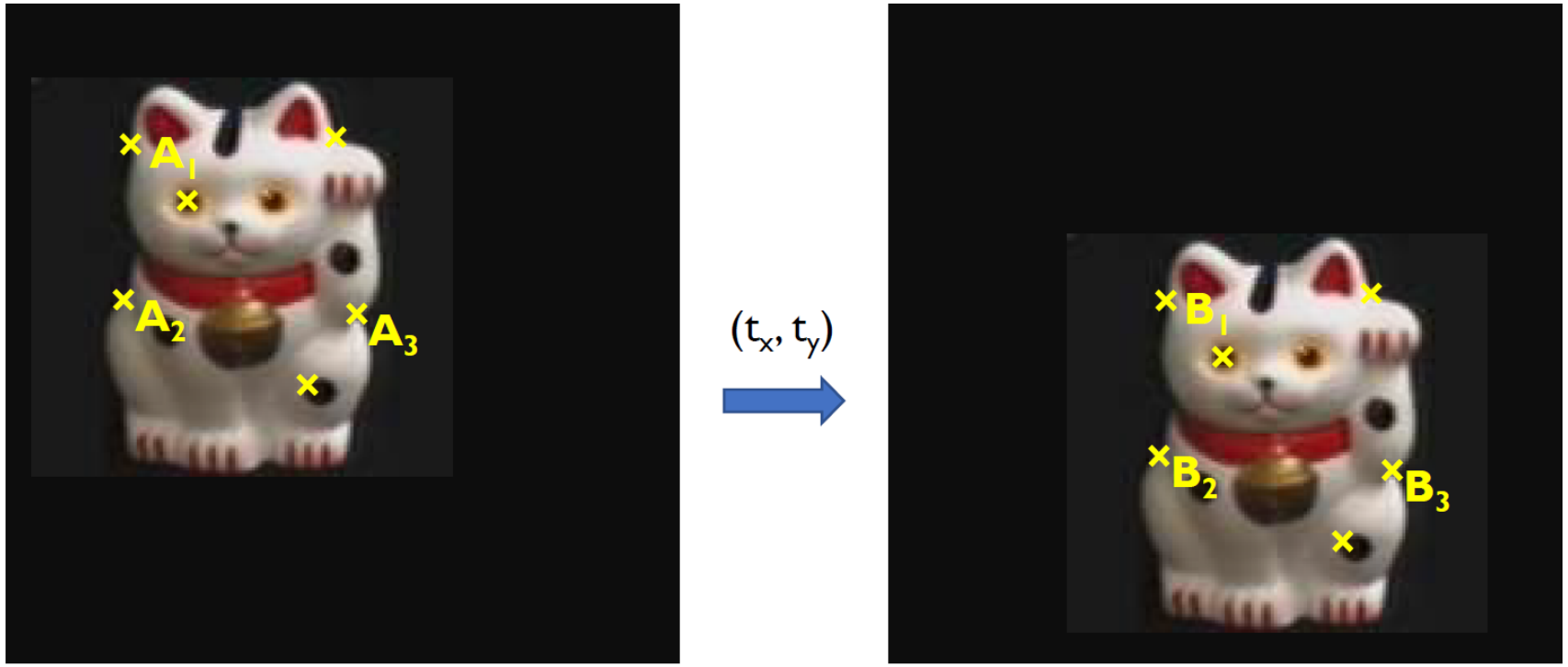




Given matched points in {A} and {B}, estimate the translation of the object

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

Example: Solving for Translation



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

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

Least squares setup

Example: Solving for Rotation | Translation | Scale

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

$$\begin{bmatrix} x_i^B \\ y_i^B \end{bmatrix} = \mathrm{T}\begin{bmatrix} x_i^A \\ y_i^A \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix} \qquad \mathrm{T} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

Camera Calibration

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

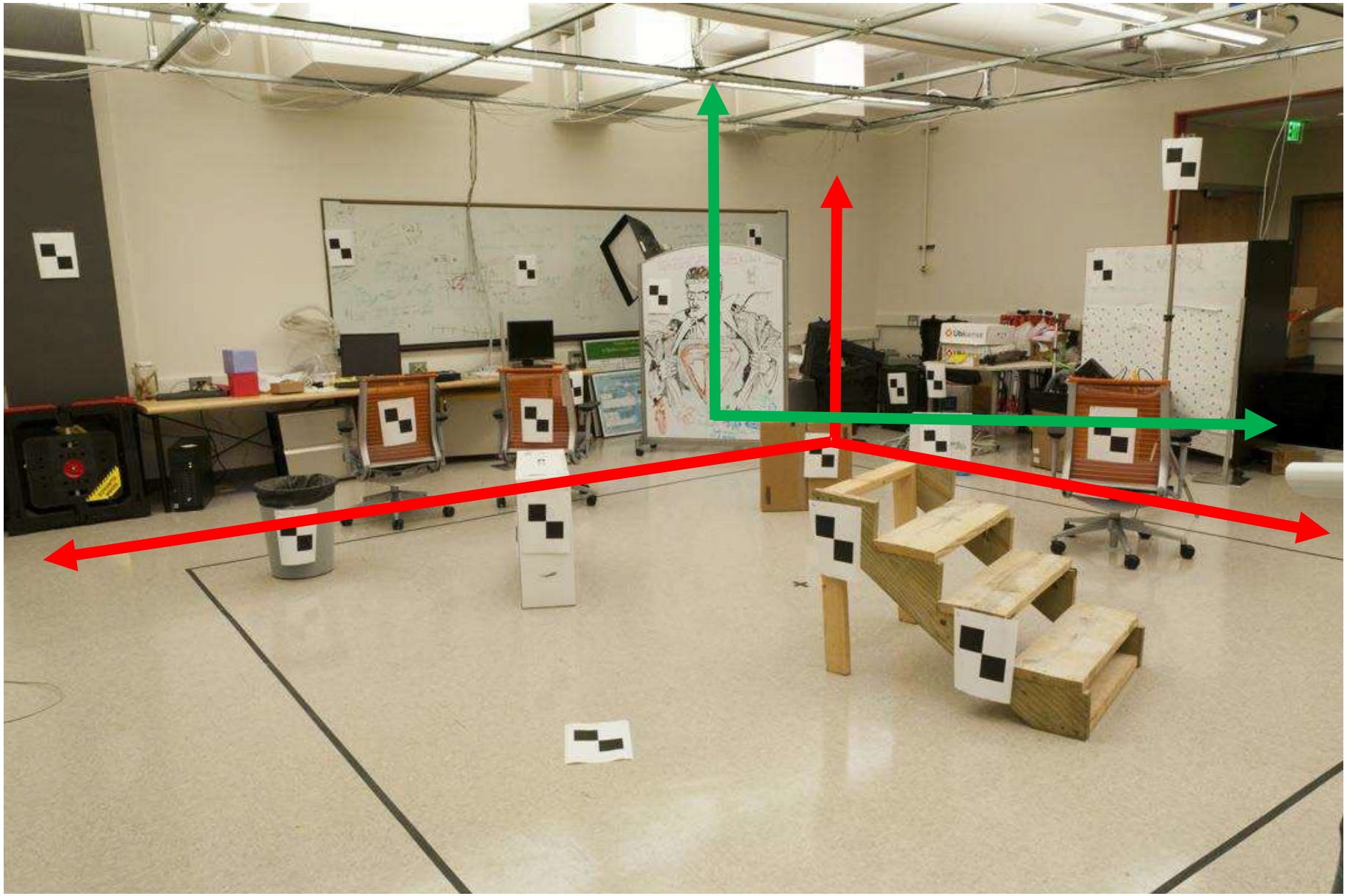
Known Unknown Known

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

Intrinsics Calibration

Extrinsics Calibration / Pose Estimation

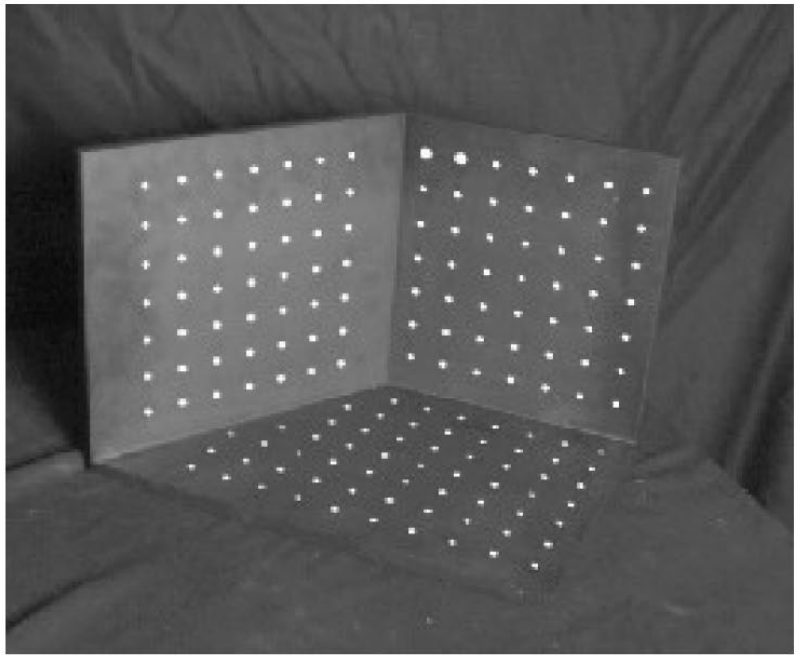
World vs Camera Coordinates



Calibrating the Camera

Use a scene with known geometry

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



Known 2D image cords (px)

Known 3D world locations (m)

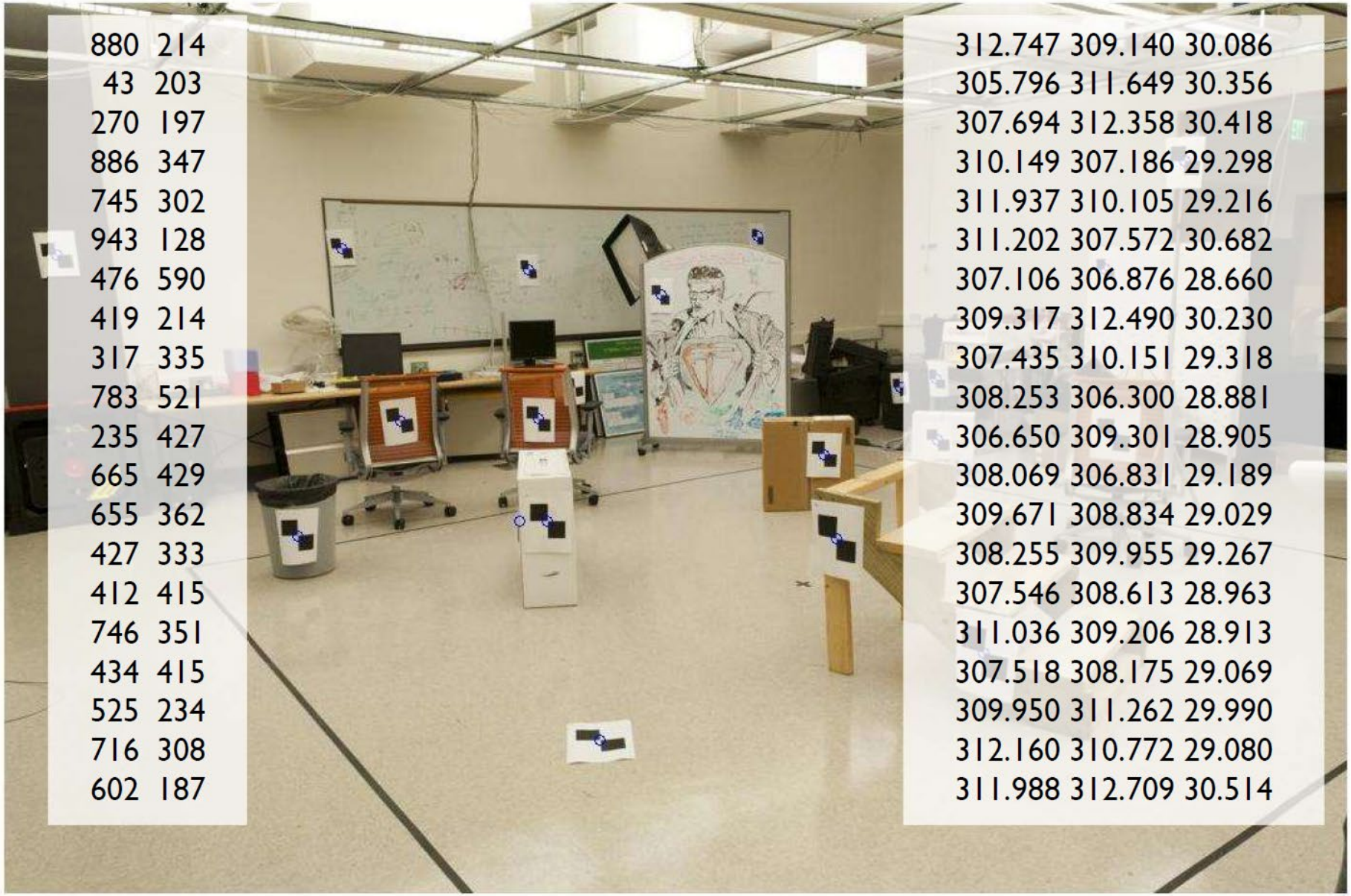
M

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

Unknown Camera Parameters

Known 2D image coords

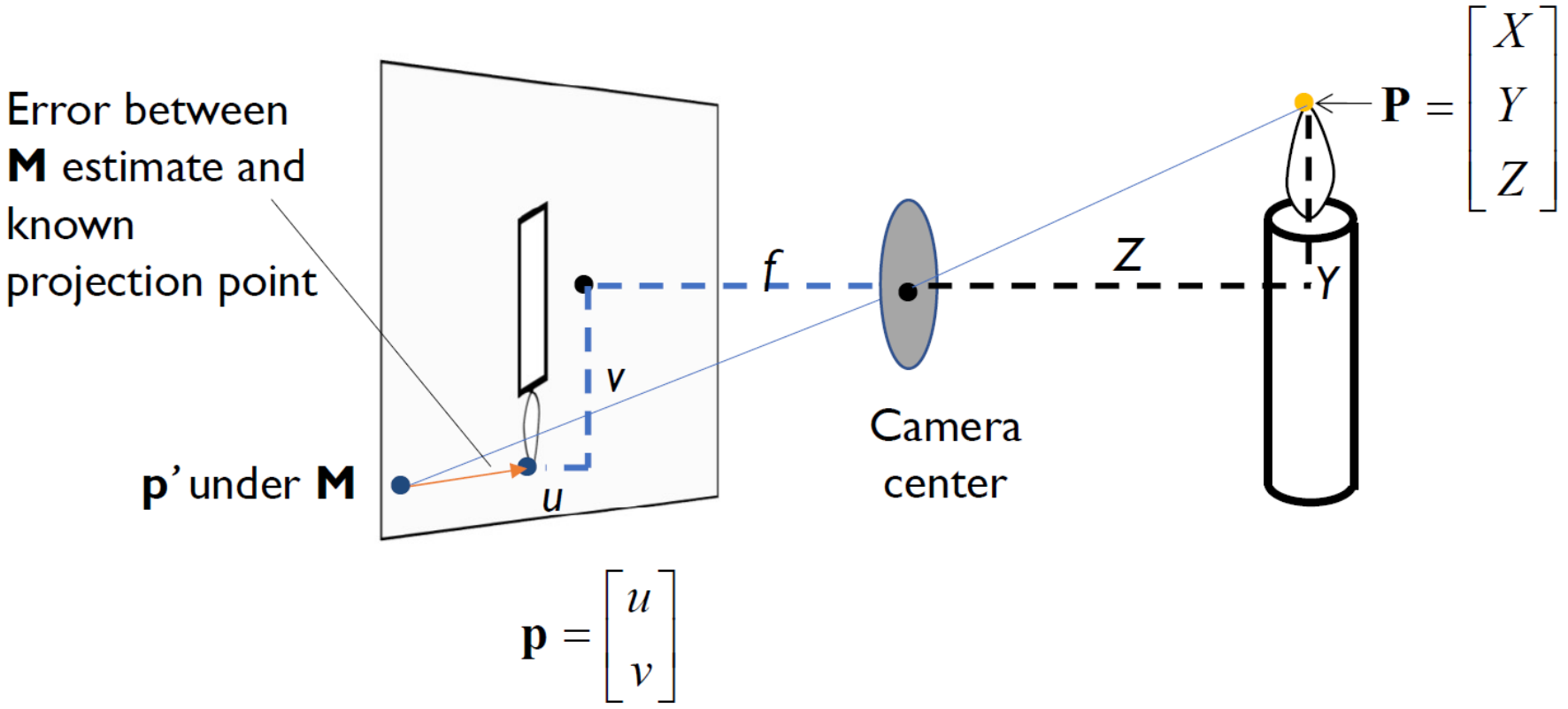
Known 3D world locations



Known 2D image coords	Known 3D world locations
880 214	312.747 309.140 30.086
43 203	305.796 311.649 30.356
270 197	307.694 312.358 30.418
886 347	310.149 307.186 29.298
745 302	311.937 310.105 29.216
943 128	311.202 307.572 30.682
476 590	307.106 306.876 28.660
419 214	309.317 312.490 30.230
317 335	307.435 310.151 29.318
783 521	308.253 306.300 28.881
235 427	306.650 309.301 28.905
665 429	308.069 306.831 29.189
655 362	309.671 308.834 29.029
427 333	308.255 309.955 29.267
412 415	307.546 308.613 28.963
746 351	311.036 309.206 28.913
434 415	307.518 308.175 29.069
525 234	309.950 311.262 29.990
716 308	312.160 310.772 29.080
602 187	311.988 312.709 30.514

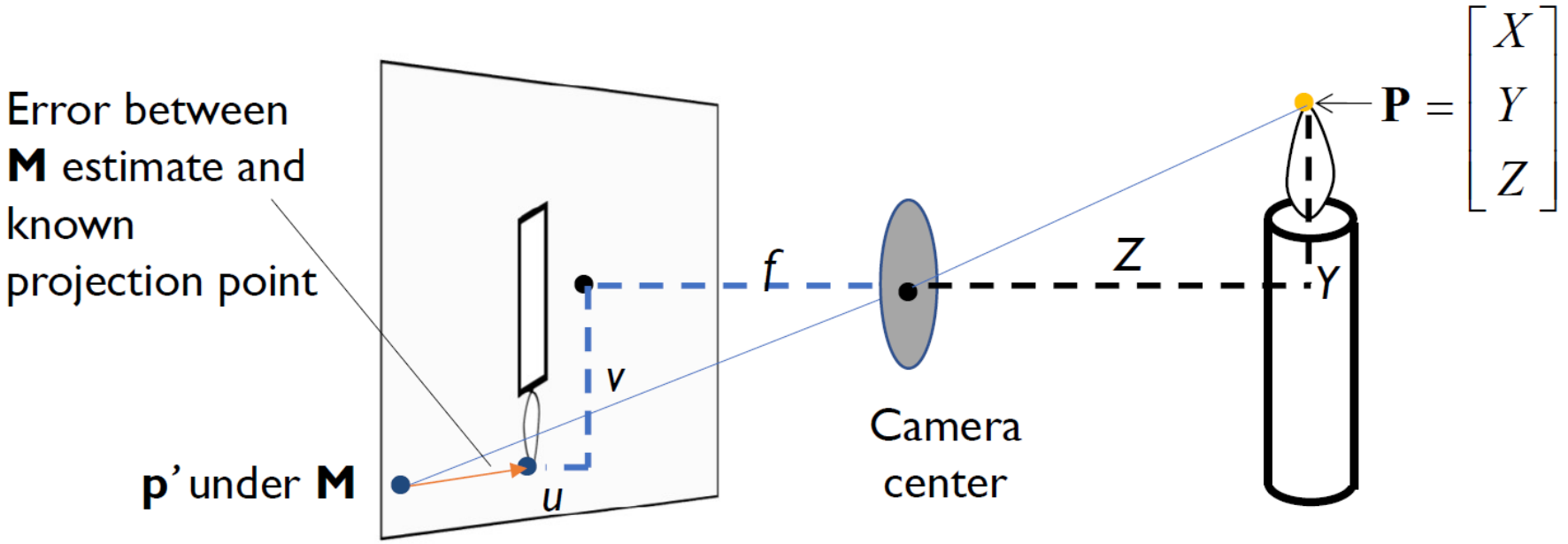
Least Squares Objective

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



Least Squares Objective

- Best **M** occurs when **p'** = **p**, or when **p'** – **p** = 0
- Form these equations from all point evidence
- Solve for model via closed-form regression



$$\mathbf{p} = \begin{bmatrix} u \\ v \end{bmatrix}$$

p = distance from image center

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

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

A **x** **b**

Python Numpy:

```
M = np.linalg.lstsq(A,b)[0]
M = np.append(M,1)
M = np.reshape(M, (3,4))
```

Note: Must reshape M afterwards!

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

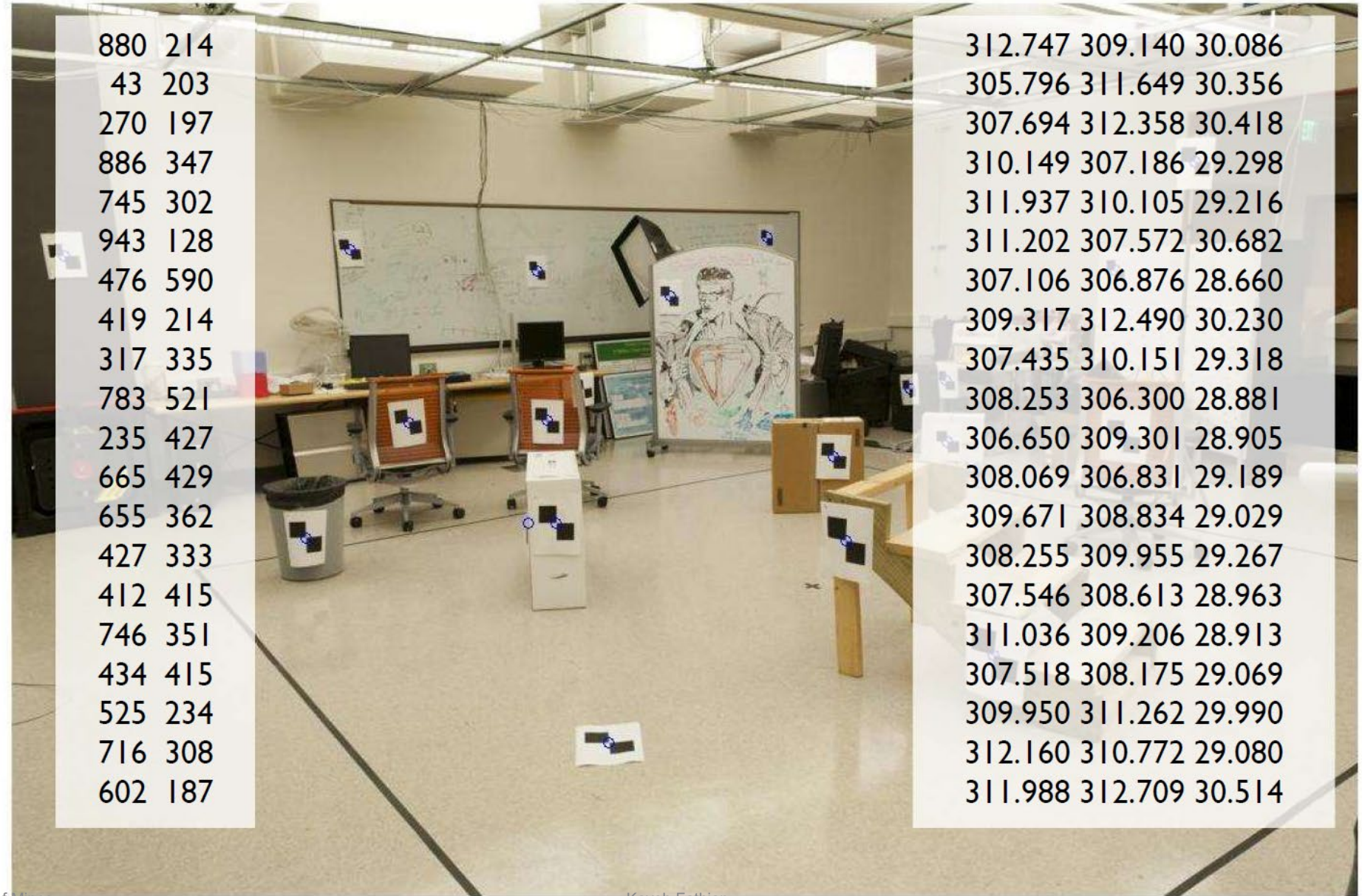
- Or, solve for m's entries using total linear least-squares
- Method 2 – **Ax=0** form
- Find non-trivial solution (not A=0)

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

```
Python Numpy:
U, S, Vh =
np.linalg.svd(a)
# V = Vh.T
M = Vh[-1,:]
M = np.reshape(M, (3,4))
```

Known 2D image coords

Known 3D world locations



Known 2D image coords	Known 3D world locations
880 214	312.747 309.140 30.086
43 203	305.796 311.649 30.356
270 197	307.694 312.358 30.418
886 347	310.149 307.186 29.298
745 302	311.937 310.105 29.216
943 128	311.202 307.572 30.682
476 590	307.106 306.876 28.660
419 214	309.317 312.490 30.230
317 335	307.435 310.151 29.318
783 521	308.253 306.300 28.881
235 427	306.650 309.301 28.905
665 429	308.069 306.831 29.189
655 362	309.671 308.834 29.029
427 333	308.255 309.955 29.267
412 415	307.546 308.613 28.963
746 351	311.036 309.206 28.913
434 415	307.518 308.175 29.069
525 234	309.950 311.262 29.990
716 308	312.160 310.772 29.080
602 187	311.988 312.709 30.514

Known 2D image coords

Known 3D world locations

1st point

(u_1, v_1)

(X_1, Y_1, Z_1)

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

$$\begin{bmatrix} 312.747 & 309.140 & 30.086 & 1 & 0 & 0 & 0 & 0 & -880\times312.747 & -880\times309.140 & -880\times30.086 & -880 \\ 0 & 0 & 0 & 0 & 312.747 & 309.140 & 30.086 & 1 & -214\times312.747 & -214\times309.140 & -214\times30.086 & -214 \\ & & & & & & \vdots & & & & & \\ X_n & Y_n & Z_n & 1 & 0 & 0 & 0 & 0 & -u_nX_n & -u_nY_n & -u_nZ_n & -u_n \\ 0 & 0 & 0 & 0 & X_n & Y_n & Z_n & 1 & -v_nX_n & -v_nY_n & -v_nZ_n & -v_n \end{bmatrix} \begin{bmatrix} m_{11} \\ m_{12} \\ m_{13} \\ m_{14} \\ m_{21} \\ m_{22} \\ m_{23} \\ m_{24} \\ m_{31} \\ m_{32} \\ m_{33} \\ m_{34} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}$$

Known 2D image coords

Known 3D world locations

2nd point

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

$$\begin{bmatrix} 312.747 & 309.140 & 30.086 & 1 & 0 & 0 & 0 & 0 & -880\times312.747 & -880\times309.140 & -880\times30.086 & -880 \\ 0 & 0 & 0 & 0 & 312.747 & 309.140 & 30.086 & 1 & -214\times312.747 & -214\times309.140 & -214\times30.086 & -214 \\ 305.796 & 311.649 & 30.356 & 1 & 0 & 0 & 0 & 0 & -43\times305.796 & -43\times311.649 & -43\times30.356 & -43 \\ 0 & 0 & 0 & 0 & 305.796 & 311.649 & 30.356 & 1 & -203\times305.796 & -203\times311.649 & -43\times30.356 & -203 \\ & & & & & & \vdots & & & & & \\ X_n & Y_n & Z_n & 1_n & 0 & 0 & 0 & 0 & -u_nX_n & -u_nY_n & -u_nZ_n & -u_n \\ 0 & 0 & 0 & 0 & X_n & Y_n & Z_n & 1 & -v_nX_n & -v_nY_n & -v_nZ_n & -v_n \end{bmatrix} \begin{bmatrix} m_{11} \\ m_{12} \\ m_{13} \\ m_{14} \\ m_{21} \\ m_{22} \\ m_{23} \\ m_{24} \\ m_{31} \\ m_{32} \\ m_{33} \\ m_{34} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}$$

How many points needed to fit the model?

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

Degrees of freedom? 5 6

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

3 Deg of freedom:
- Rotation around x
- Rotation around y
- Rotation around z

3 Deg of freedom:
- Translation x
- Translation y
- Translation z

How many points needed to fit the model?

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

Degrees of freedom? 5 6

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

- **M** is 3x4, so 12 unknowns, but 11 deg of freedom as projective space has scale ambiguity
- One equation per deg of freedom ➔ 5 1/2 point correspondences determines a solution (e.g., either u or v for last point).
- More than 5 1/2 point correspondences ➔ overdetermined system of equations
- Least squares is finding the solution that best satisfies the overdetermined system
- Why use more than 6 points? Robustness to error in feature points.

Factorize M back to K [R | T]

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

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

- We can also use RQ factorization (not QR)
  - R in RQ is not rotation matrix R; crossed names!
- R (upper triangular or 'Right' triangular) is K
- Q (orthogonal basis) is R
- T, the last column of [R | T], is inv(K) * last column of M.
  - 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/

Geometric camera calibration

Advantages:

- Very simple to formulate.
- Analytical solution.

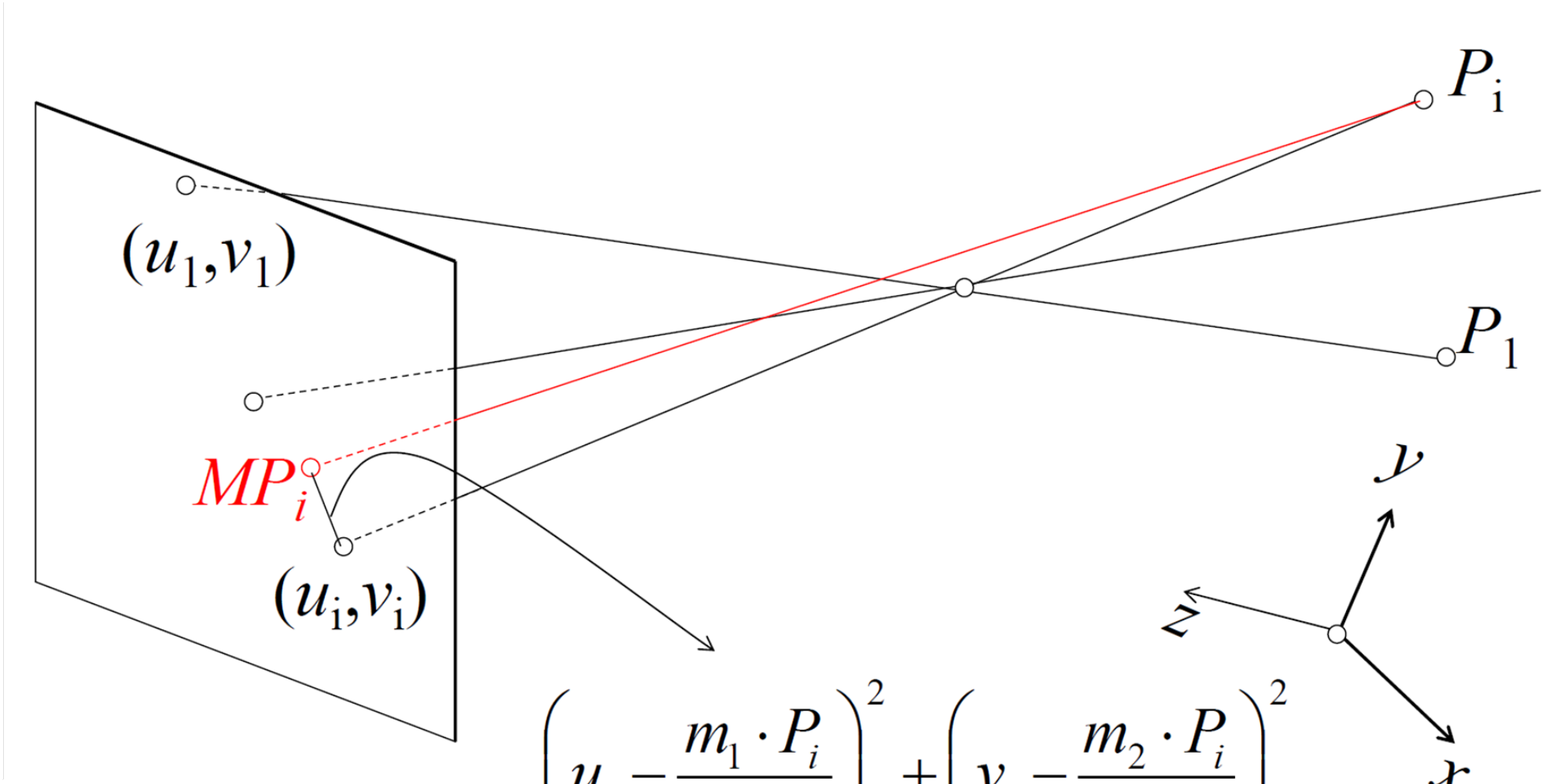
Disadvantages:

- Doesn't model radial distortion.
- Hard to impose constraints (e.g., known *f*).
- Doesn't minimize the correct error function.

For these reasons, ***nonlinear methods*** are preferred

- Define error function E between projected 3D points and image positions
  - E is nonlinear function of intrinsics, extrinsics, radial distortion
- Minimize E using nonlinear optimization techniques

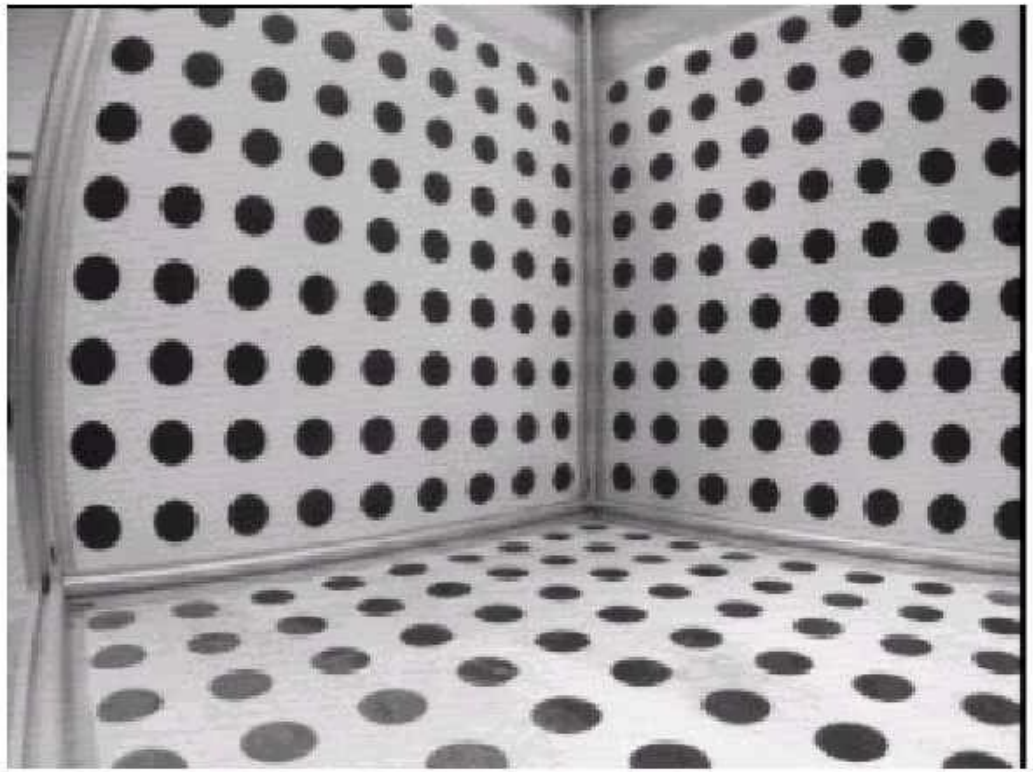
Minimizing reprojection error



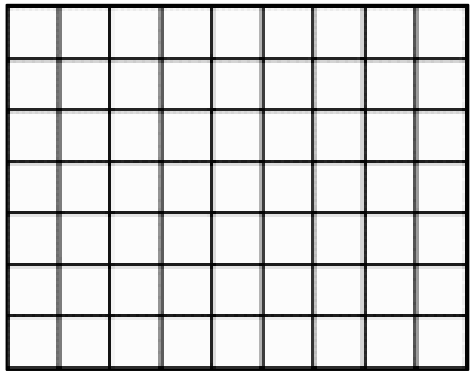
$$\left(u_i - \frac{m_1 \cdot P_i}{m_3 \cdot P_i}\right)^2 + \left(v_i - \frac{m_2 \cdot P_i}{m_3 \cdot P_i}\right)^2$$

Is this equivalent to what we were doing previously?

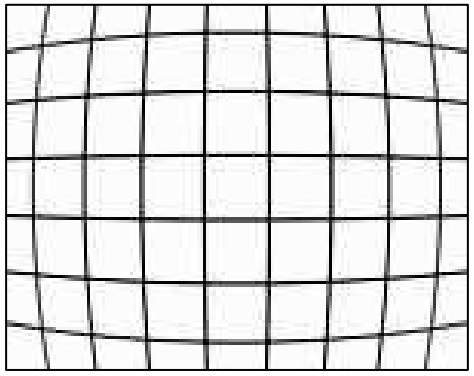
Radial distortion



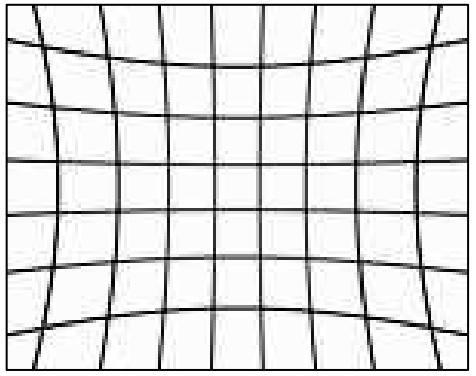
What causes this distortion?



no distortion

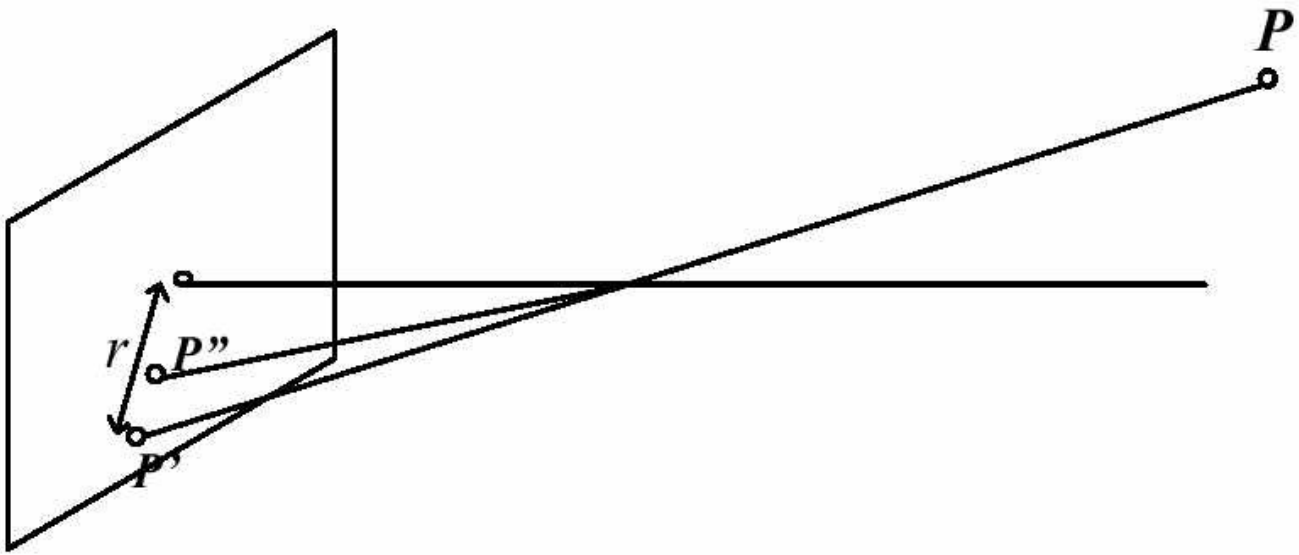


barrel distortion



pincushion distortion

Radial distortion model



Ideal:

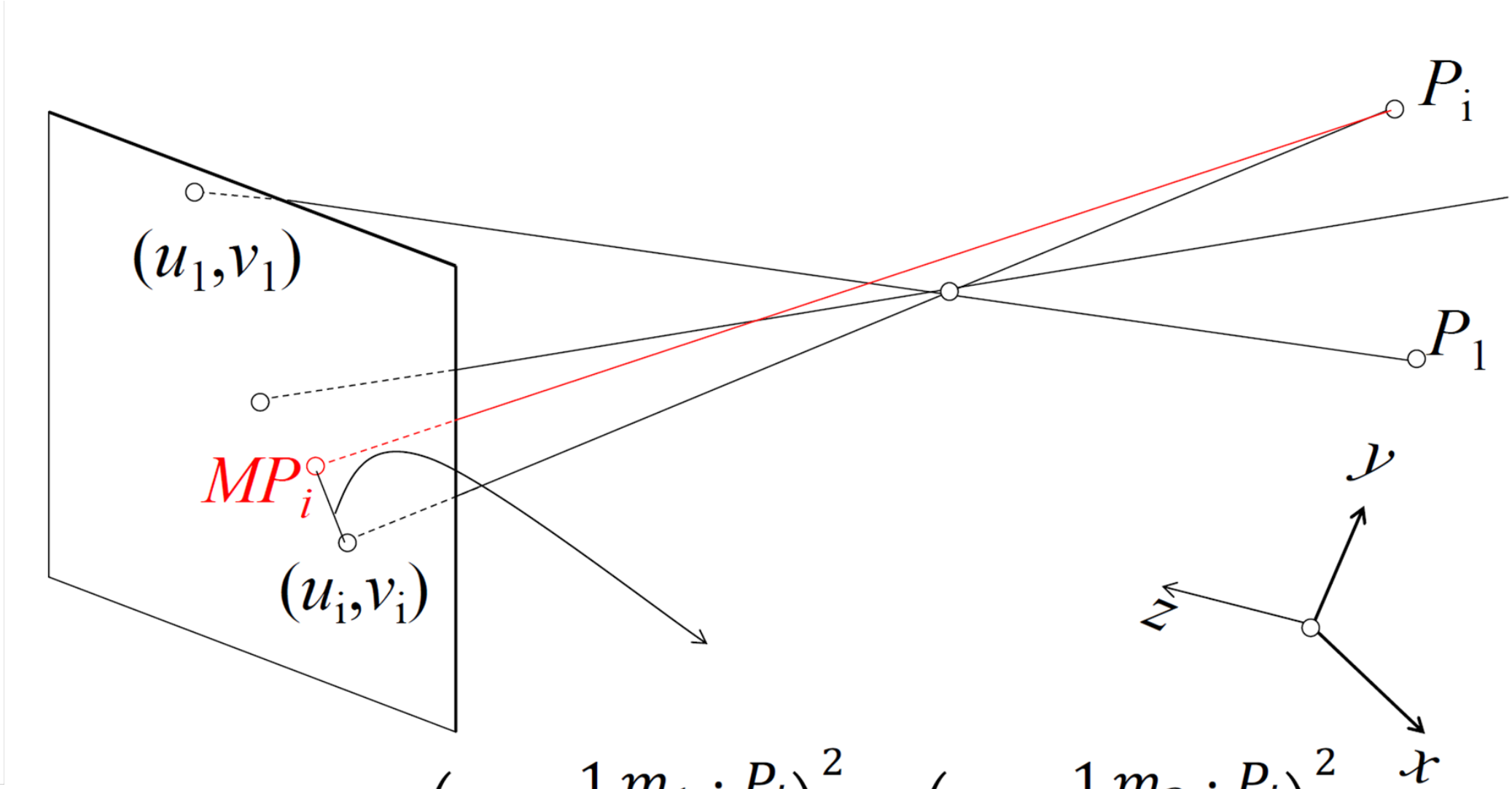
$$x' = f\frac{x}{z} \qquad y' = f\frac{y}{z}$$

Distorted:

$$x'' = \frac{1}{\lambda}x' \qquad y'' = \frac{1}{\lambda}y'$$

$$\lambda = 1 + k_1 r^2 + k_2 r^4 + \cdots$$

Minimizing reprojection error with radial distortion



Add distortions to reprojection error:

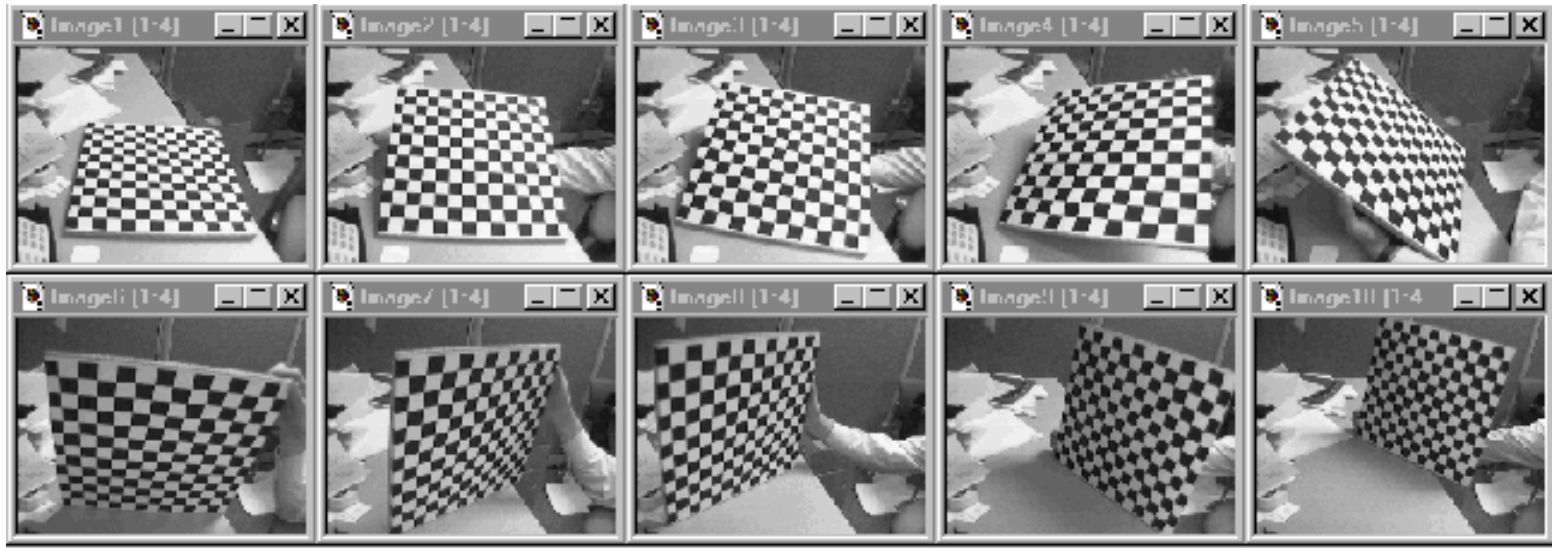
$$\left(u_i - \frac{1}{\lambda}\frac{m_1 \cdot P_i}{m_3 \cdot P_i}\right)^2 + \left(v_i - \frac{1}{\lambda}\frac{m_2 \cdot P_i}{m_3 \cdot P_i}\right)^2$$

Correcting radial distortion

before

after

Alternative: Multi-plane calibration

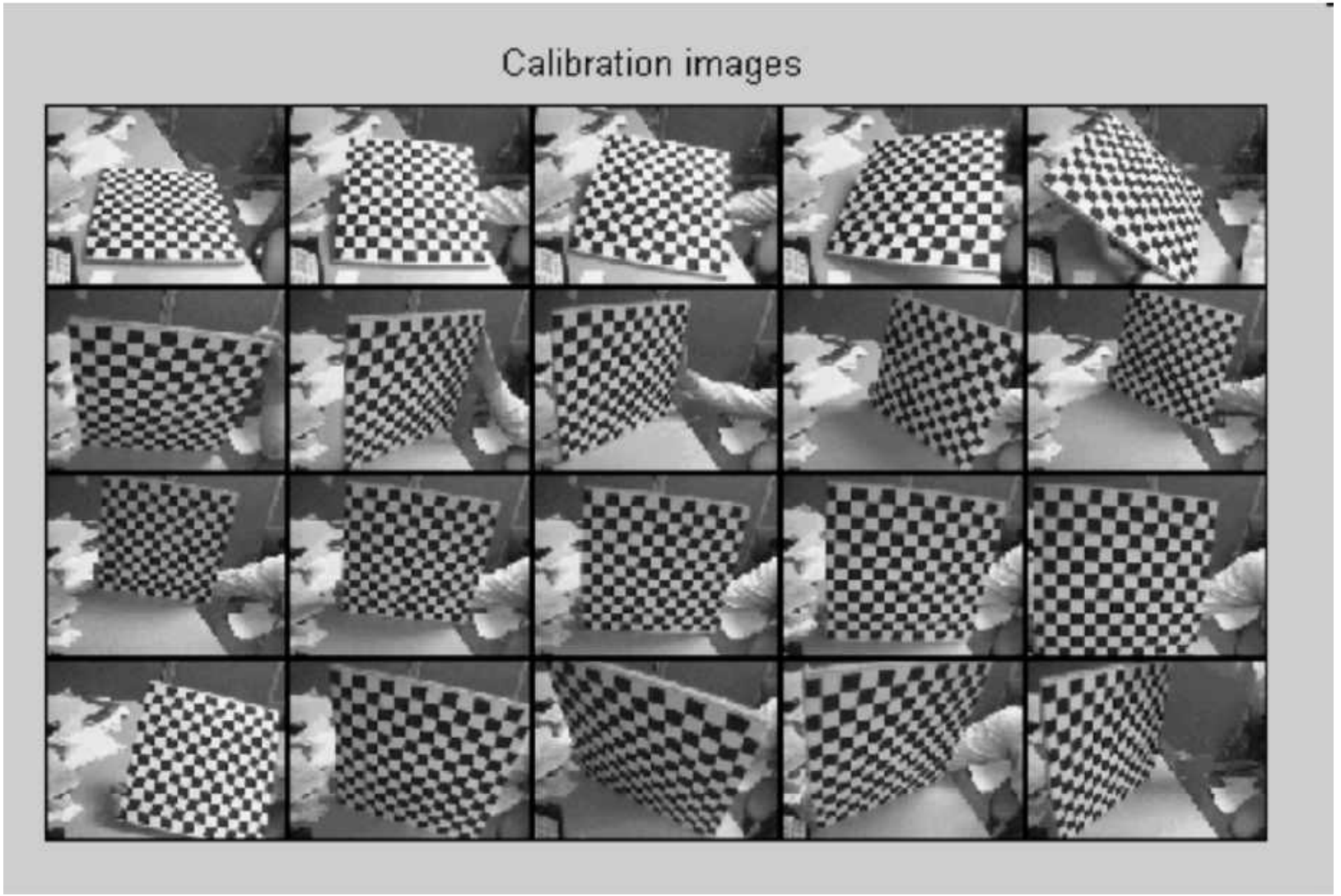


Advantages:

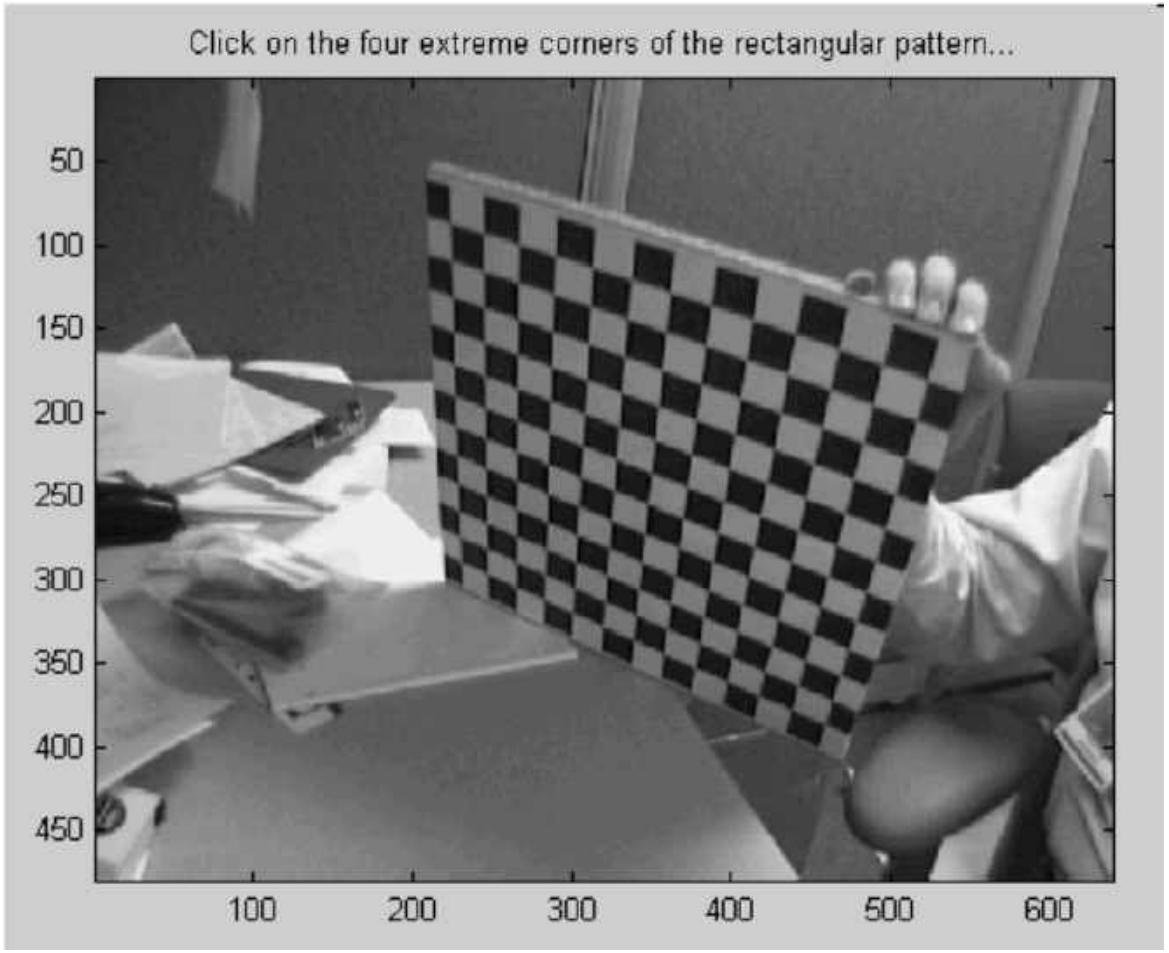
- Only requires a plane
- Don't have to know positions/orientations
- Great code available online!
  - Matlab version: http://www.vision.caltech.edu/bouguetj/calib_doc/index.html
  - Also available on OpenCV.

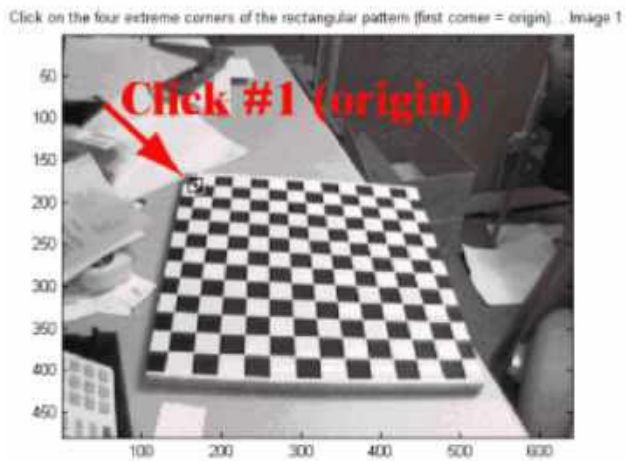
Disadvantage: Need to solve non-linear optimization problem.

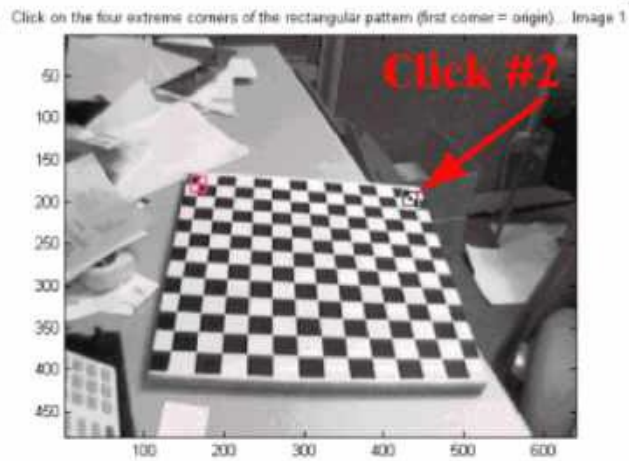
Step-by-step demonstration

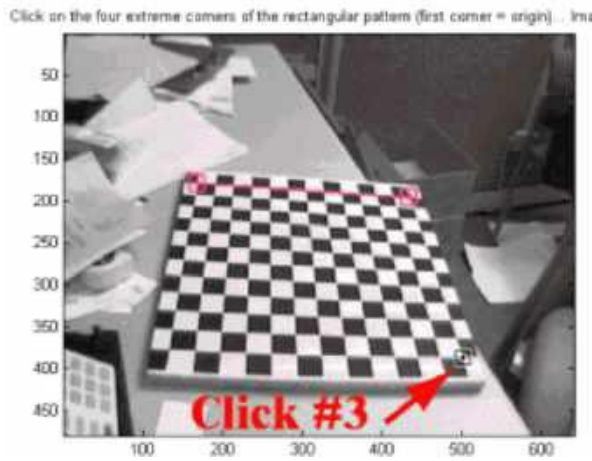


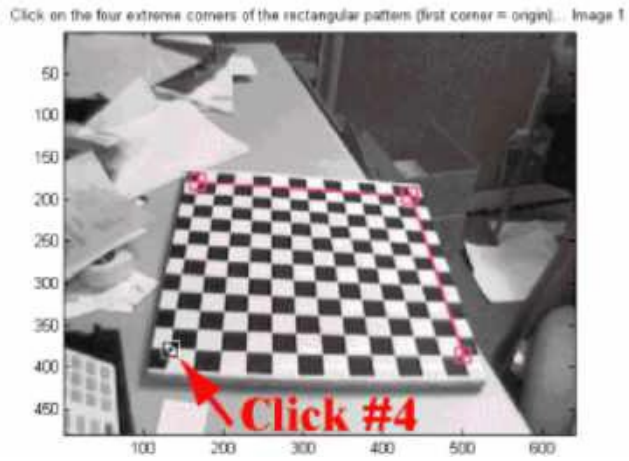
Step-by-step demonstration



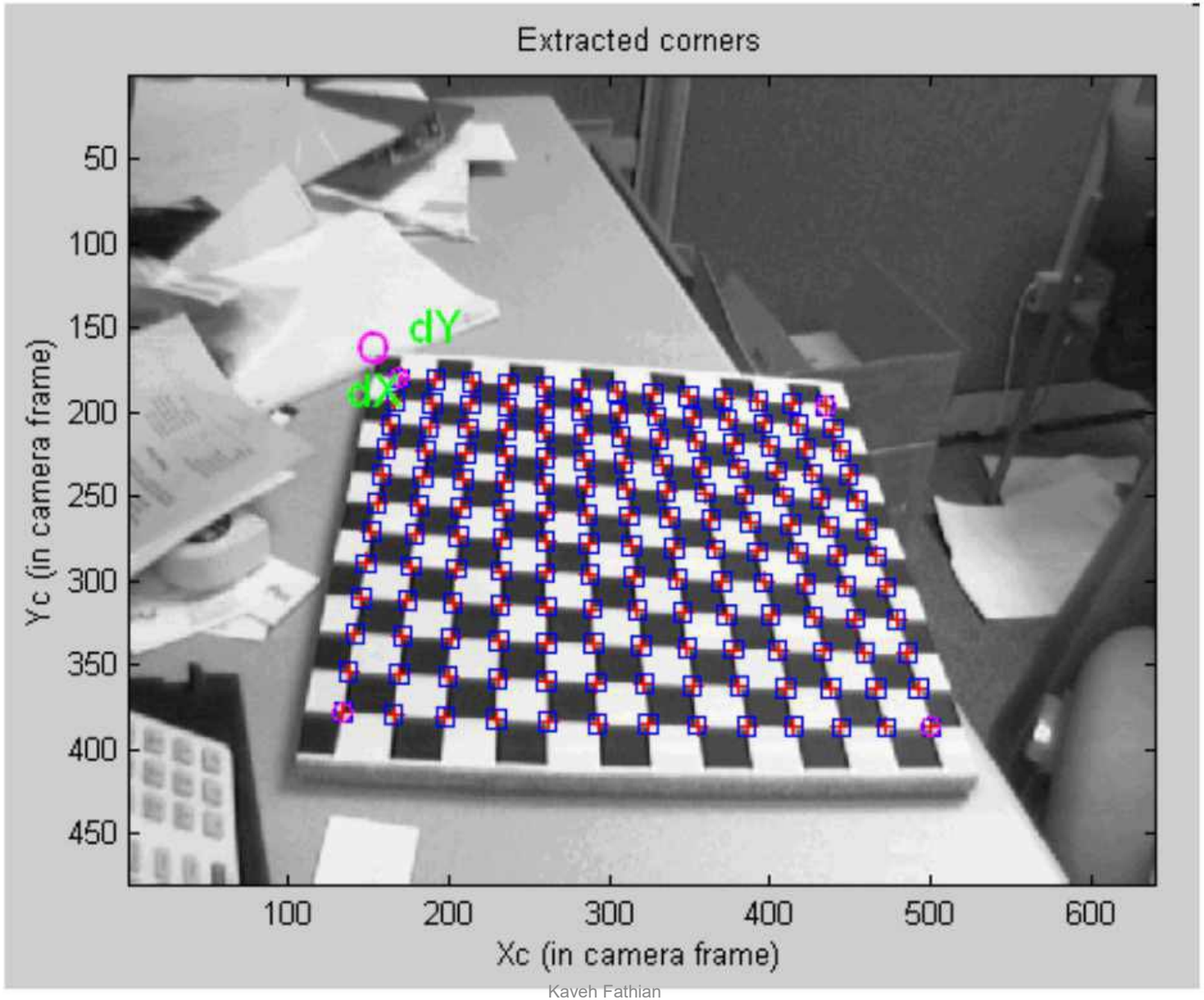




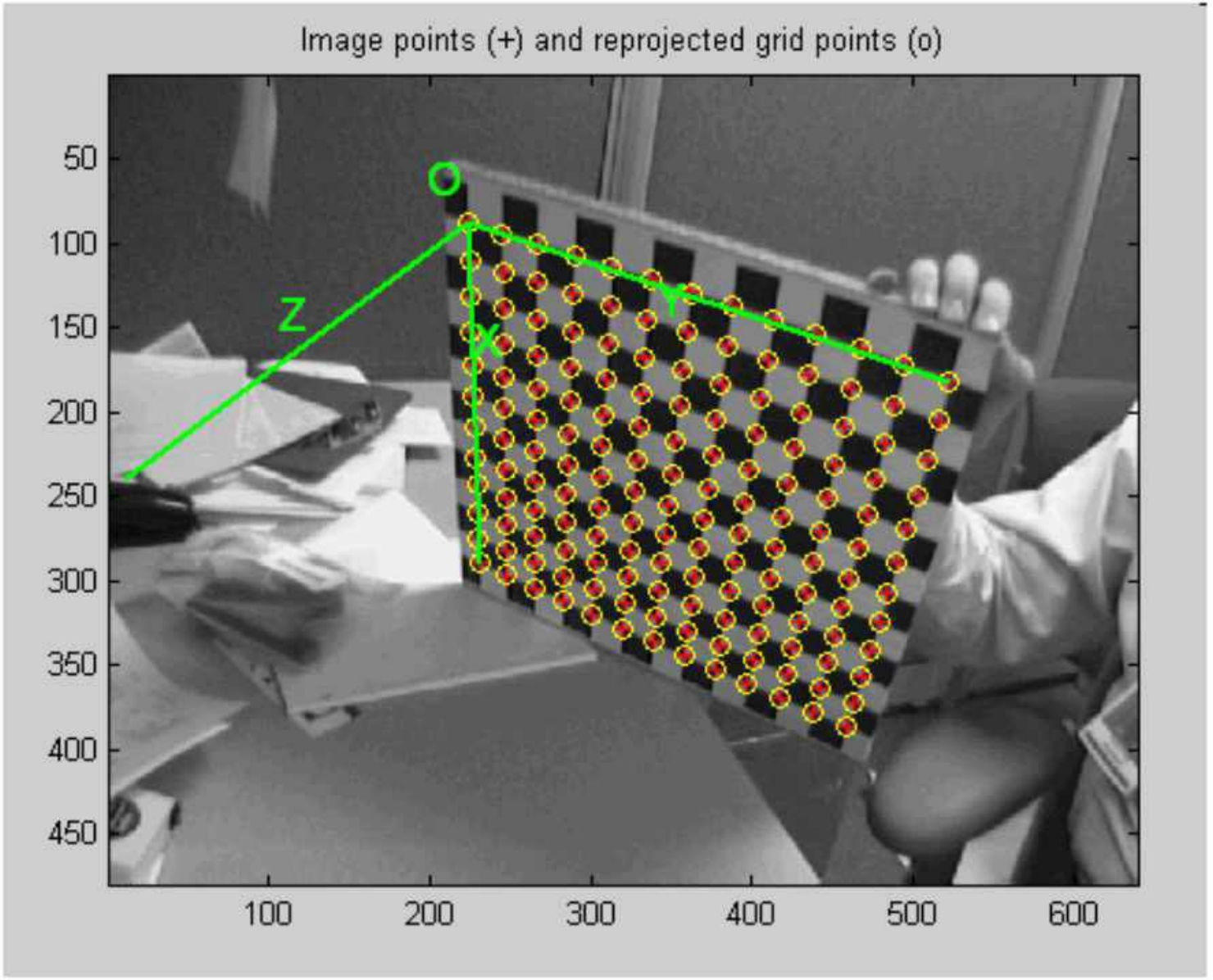




Step-by-step demonstration



Step-by-step demonstration



Step-by-step demonstration