INTRODUCTION TO COMPUTER VISION

Lecture 6 – Depth Estimation

Gyeongsik Moon

Visual Computing and AI Lab

Korea University

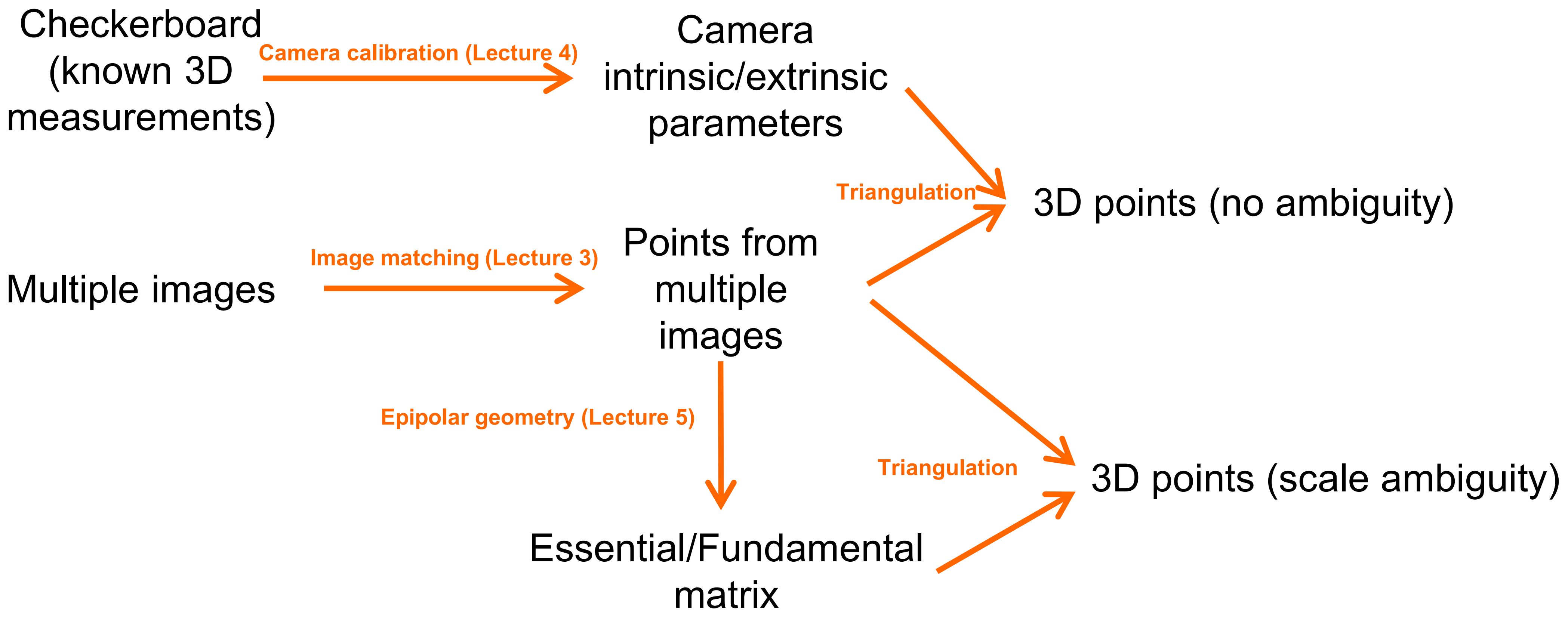
Slides Credit: Prof. Kris Kitani

1. Triangulation
2. Bundle Adjustment

1. Triangulation

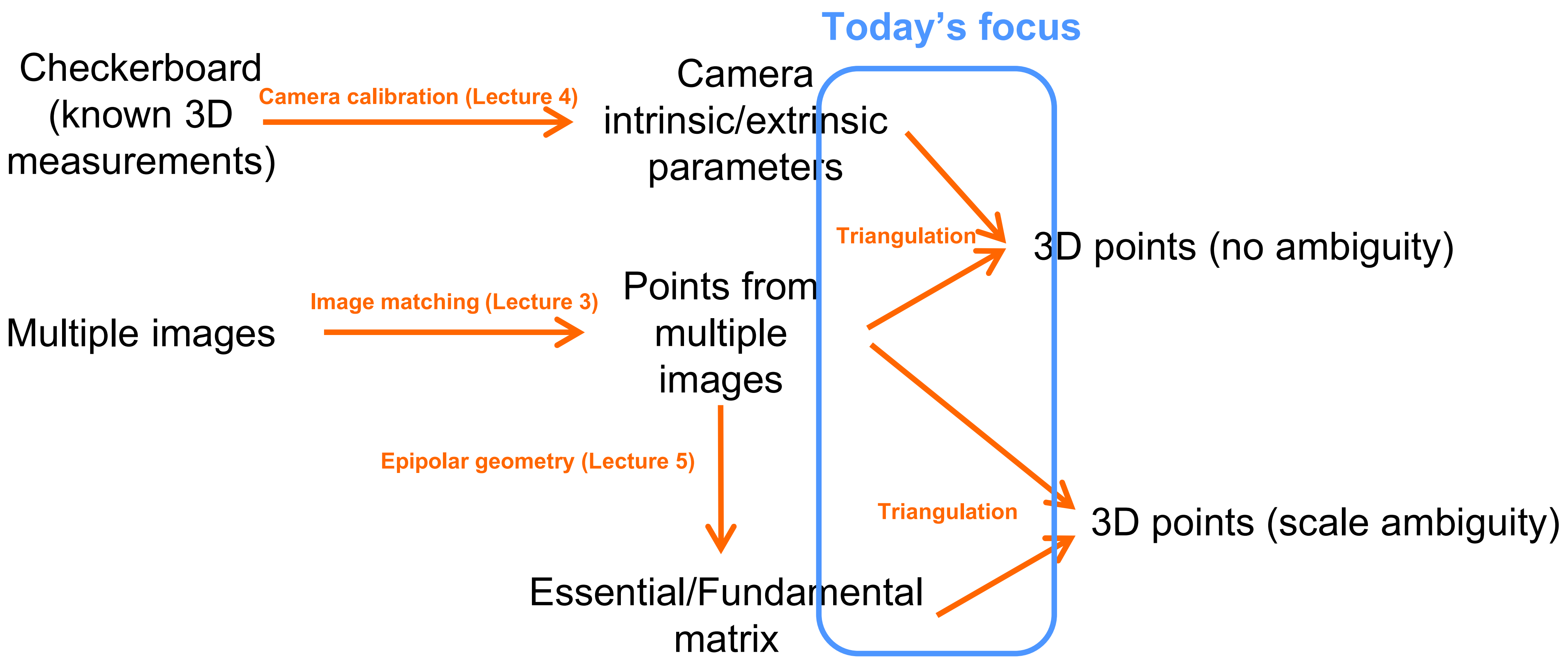
- All the entire pipeline: multi-view geometry theory

Slide from Lecture 5

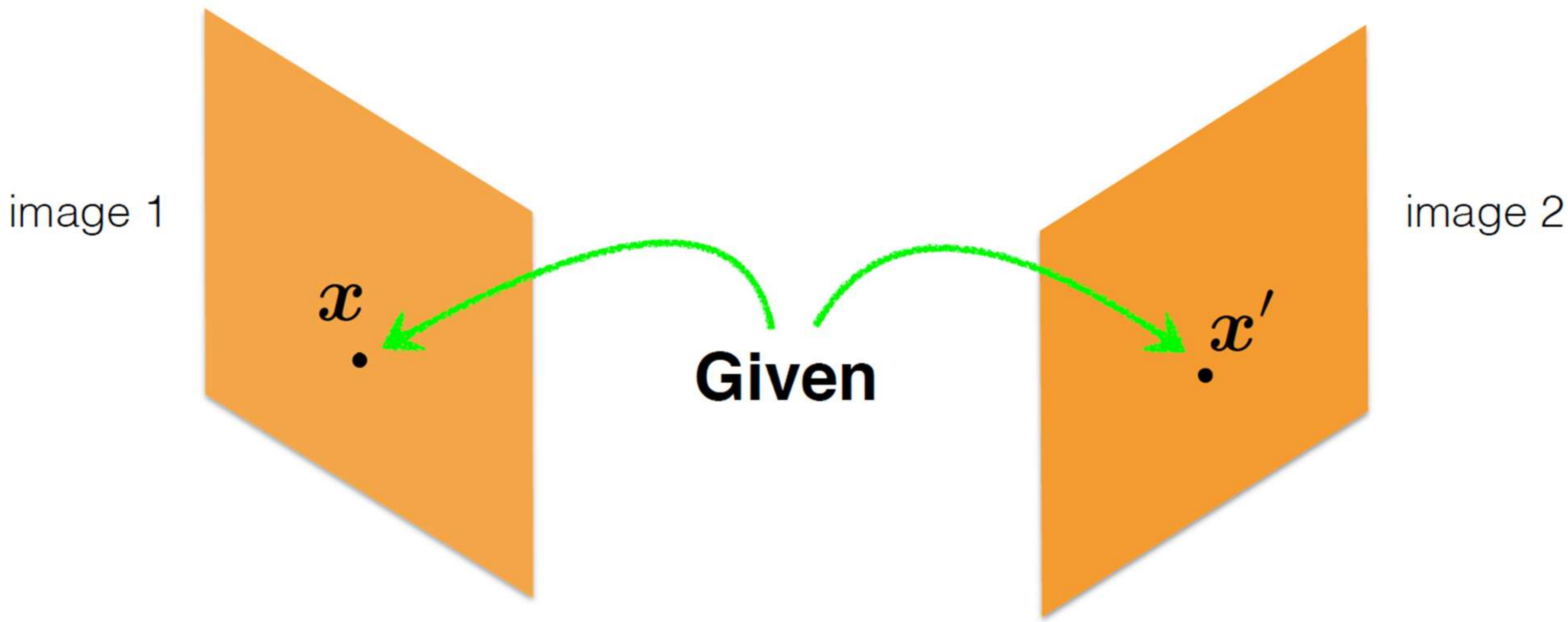


- All the entire pipeline: multi-view geometry theory

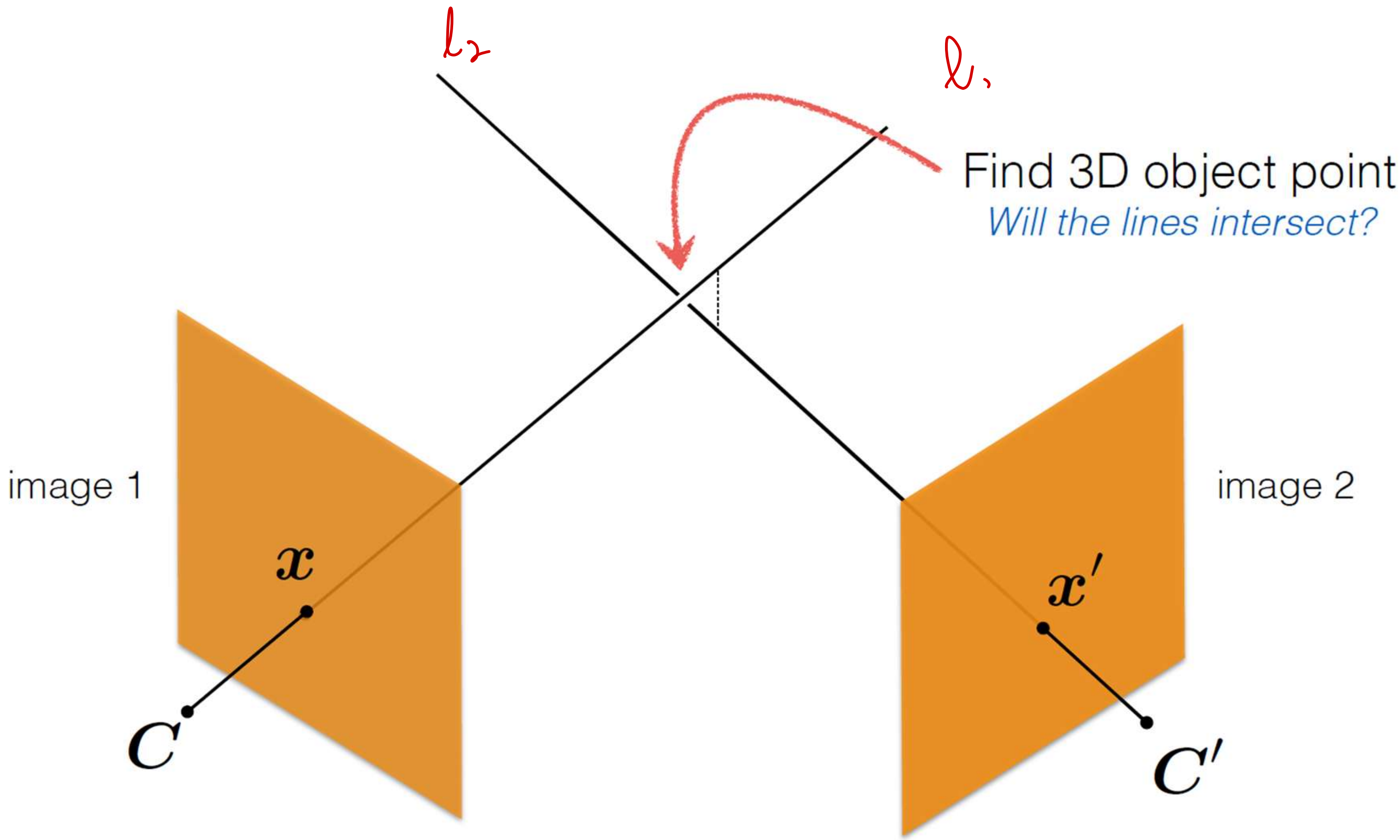
Slide from Lecture 5



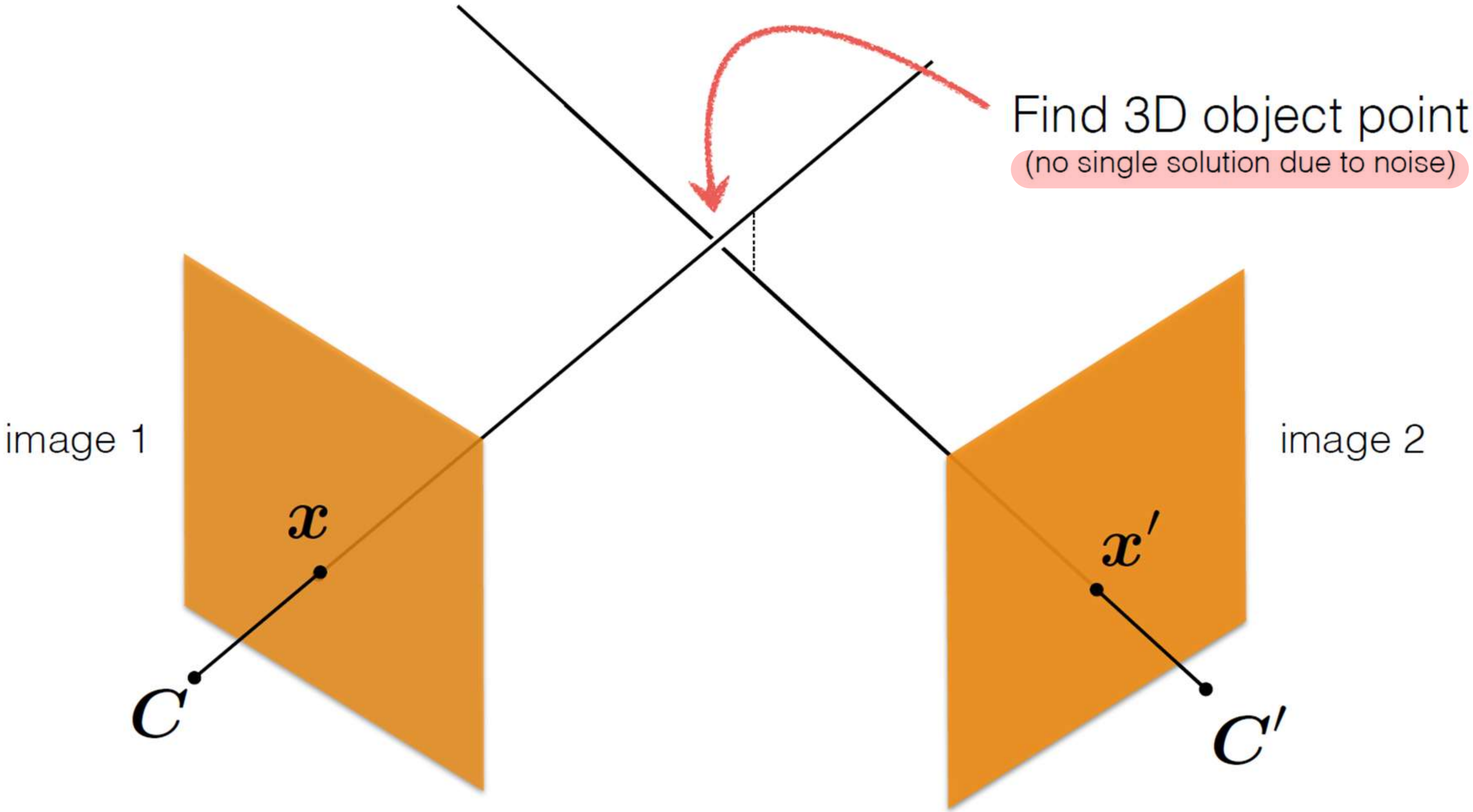
Triangulation



Triangulation



Triangulation



Triangulation

Given a set of (noisy) matched points

$$\{\boldsymbol{x}_i, \boldsymbol{x}'_i\}$$

2D matched point

and camera matrices

$$\mathbf{P}, \mathbf{P}'$$

intrinsic / extrinsic

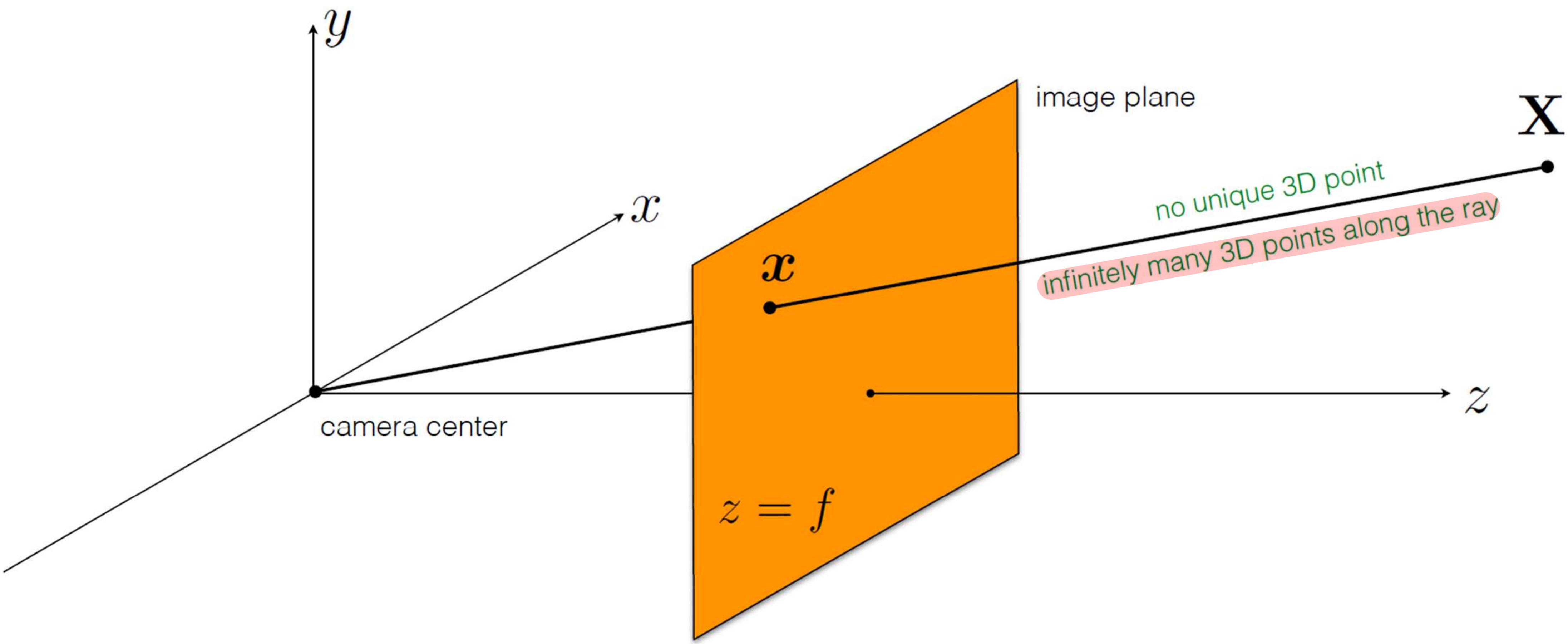
Estimate the 3D point

$$\mathbf{X}$$

$$\overset{2D}{\mathbf{x}} = \mathbf{P}\overset{3D}{\mathbf{X}}$$

known known

*Can we compute **X** from a single correspondence **x**?* in theory, no unique point



y
x
image plane
X
no unique 3D point
infinitely many 3D points along the ray
x
z
camera center
z = f

$$\mathbf{x} = \mathbf{P}\boldsymbol{X}$$

known known

*Can we compute **X** from <u>two</u> correspondences **x** and **x'**?*

$$\mathbf{x} = \mathbf{P}\boldsymbol{X}$$

known known

*Can we compute **X** from <u>two</u> correspondences **x** and **x'**?*

yes if perfect measurements

$$\mathbf{x} = \mathbf{P}\boldsymbol{X}$$

known known

Can we compute ***X*** *from* <u>*two*</u> *correspondences* ***x*** *and* ***x'****?*

yes if perfect measurements

There will not be a point that satisfies both constraints because the measurements are usually noisy

$$\mathbf{x}' = \mathbf{P}'\boldsymbol{X} \qquad \mathbf{x} = \mathbf{P}\boldsymbol{X}$$

assumption: same 3D point.

Need to find the **best fit**

optimize

$$\mathbf{x} = \mathbf{P}\boldsymbol{X}$$

(homogeneous coordinate)

Also, this is a similarity relation because it involves homogeneous coordinates

$$\mathbf{x} = \alpha\mathbf{P}\boldsymbol{X}$$

(inhomogeneous coordinate)

Same ray direction but differs by a scale factor

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \alpha \begin{bmatrix} p_1 & p_2 & p_3 & p_4 \\ p_5 & p_6 & p_7 & p_8 \\ p_9 & p_{10} & p_{11} & p_{12} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

How do we solve for unknowns in a similarity relation?

$$\mathbf{x} = \mathbf{P}\boldsymbol{X}$$

(homogeneous coordinate)

Also, this is a similarity relation because it involves homogeneous coordinates

$$\mathbf{x} = \alpha\mathbf{P}\boldsymbol{X}$$

(inhomogeneous coordinate)

Same ray direction but differs by a scale factor

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \alpha \begin{bmatrix} p_1 & p_2 & p_3 & p_4 \\ p_5 & p_6 & p_7 & p_8 \\ p_9 & p_{10} & p_{11} & p_{12} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

How do we solve for unknowns in a similarity relation?

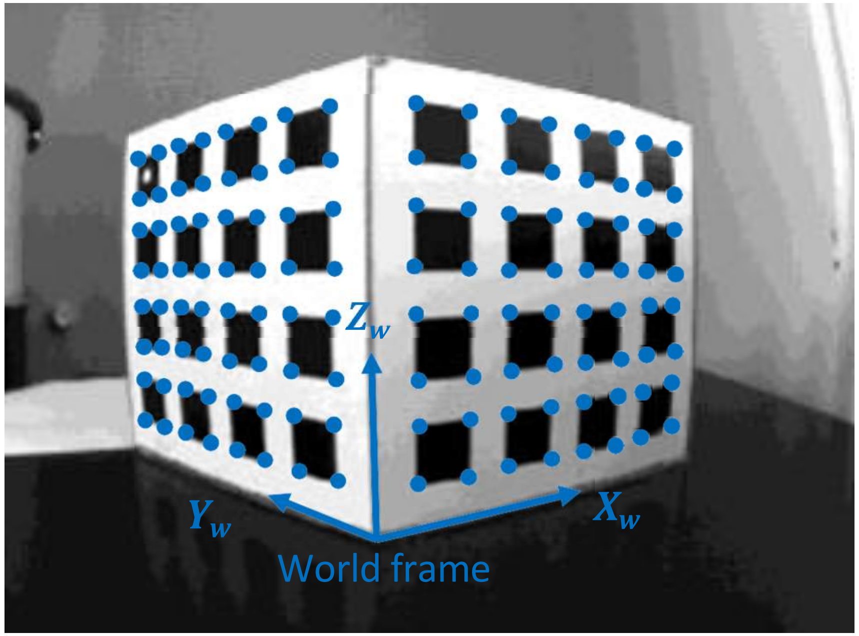
Direct Linear Transform (DLT)

Remove scale factor, convert to linear system and solve with SVD.

Comparison to DLT from Lecture 4

Tsai's Method: Calibration from 3D Objects **Slide from Lecture 4**

- This method was proposed in 1987 by Tsai and consists of measuring the 3D position of $n \geq 6$ **control points** on a 3D calibration target and the **2D coordinates of their projection** in the image.
- Assumption: we know 2D and 3D coordinates of control points
  - 2D: image pre-processing (e.g., corner detectors)
  - 3D: we know actual size of the 3D object and actual positions of control points as well



Tsai, Roger Y. (1987) "A Versatile Camera Calibration Technique for High-Accuracy 3D Machine Vision Metrology Using Off-the-Shelf TV Cameras and Lenses,'' *IEEE Journal of Robotics and Automation*, 1987. PDF.

Slide from Lecture 4

Applying the Direct Linear Transform (DLT) algorithm

The idea of the DLT is to rewrite the perspective projection equation as a **homogeneous linear equation** and solve it by standard methods. Let's write the perspective equation for a generic 3D-2D point correspondence:

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K[R \mid T] \cdot \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \Rightarrow$$

$$\Rightarrow \lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha_u & 0 & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \cdot \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}$$

$$\Rightarrow \lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha_u r_{11} + u_0 r_{31} & \alpha_u r_{12} + u_0 r_{32} & \alpha_u r_{13} + u_0 r_{33} & \alpha_u t_1 + u_0 t_3 \\ \alpha_v r_{21} + v_0 r_{31} & \alpha_v r_{22} + v_0 r_{32} & \alpha_v r_{23} + v_0 r_{33} & \alpha_v t_2 + v_0 t_3 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \cdot \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}$$

① camera parameter
② 2D matching
③ 3D point.

Comparison to DLT from Lecture 4

- Both are for solving linear equations
- Lecture 4: Getting camera parameters from 1) estimated 2D matching and 2) 3D measurements
- Today: Getting 3D points from 1) estimated 2D matching and 2) camera parameters

$$\mathbf{x} = \alpha \mathbf{P} \boldsymbol{X}$$

Same direction but differs by a scale factor

$$\mathbf{x} \times \mathbf{P} \boldsymbol{X} = \mathbf{0}$$

Cross product of two vectors of same direction is zero
(this equality removes the scale factor)

Using the fact that the cross product should be zero

$$\mathbf{x} \times \mathbf{P}\boldsymbol{X} = \mathbf{0}$$

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \times \begin{bmatrix} \boldsymbol{p}_1^\top \boldsymbol{X} \\ \boldsymbol{p}_2^\top \boldsymbol{X} \\ \boldsymbol{p}_3^\top \boldsymbol{X} \end{bmatrix} = \begin{bmatrix} y\boldsymbol{p}_3^\top \boldsymbol{X} - \boldsymbol{p}_2^\top \boldsymbol{X} \\ \boldsymbol{p}_1^\top \boldsymbol{X} - x\boldsymbol{p}_3^\top \boldsymbol{X} \\ x\boldsymbol{p}_2^\top \boldsymbol{X} - y\boldsymbol{p}_1^\top \boldsymbol{X} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

Third line is a linear combination of the first and second lines.
(x times the first line plus y times the second line)

One 2D to 3D point correspondence give you 2 equations

$$\begin{bmatrix} y\boldsymbol{p}_3^\top \boldsymbol{X} - \boldsymbol{p}_2^\top \boldsymbol{X} \\ \boldsymbol{p}_1^\top \boldsymbol{X} - x\boldsymbol{p}_3^\top \boldsymbol{X} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} y\boldsymbol{p}_3^\top - \boldsymbol{p}_2^\top \\ \boldsymbol{p}_1^\top - x\boldsymbol{p}_3^\top \end{bmatrix} \boldsymbol{X} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\mathbf{A}_i \boldsymbol{X} = \mathbf{0}$$

Now we can make a system of linear equations
(two lines for each 2D point correspondence)

Concatenate the 2D points from both images

$$\begin{bmatrix} y\boldsymbol{p}_3^\top - \boldsymbol{p}_2^\top \\ \boldsymbol{p}_1^\top - x\boldsymbol{p}_3^\top \\ y'\boldsymbol{p}_3'^\top - \boldsymbol{p}_2'^\top \\ \boldsymbol{p}_1'^\top - x'\boldsymbol{p}_3'^\top \end{bmatrix} \boldsymbol{X} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$\mathbf{A}\boldsymbol{X} = \mathbf{0}$$

How do we solve homogeneous linear system?

S V D !

Recall: Total least squares

(**Warning:** change of notation. x is a vector of parameters!)

$$E_{\mathrm{TLS}} = \sum_i (\boldsymbol{a}_i \boldsymbol{x})^2$$

$$= \|\mathbf{A}\boldsymbol{x}\|^2 \qquad \text{(matrix form)}$$

$$\|\boldsymbol{x}\|^2 = 1 \qquad \text{constraint}$$

minimize $\|\mathbf{A}\boldsymbol{x}\|^2$

subject to $\|\boldsymbol{x}\|^2 = 1$

$\Rightarrow$

minimize $\dfrac{\|\mathbf{A}\boldsymbol{x}\|^2}{\|\boldsymbol{x}\|^2}$

(Rayleigh quotient)

Solution is the eigenvector
corresponding to smallest eigenvalue of

$$\mathbf{A}^\top \mathbf{A}$$

Img

(3, H, W)

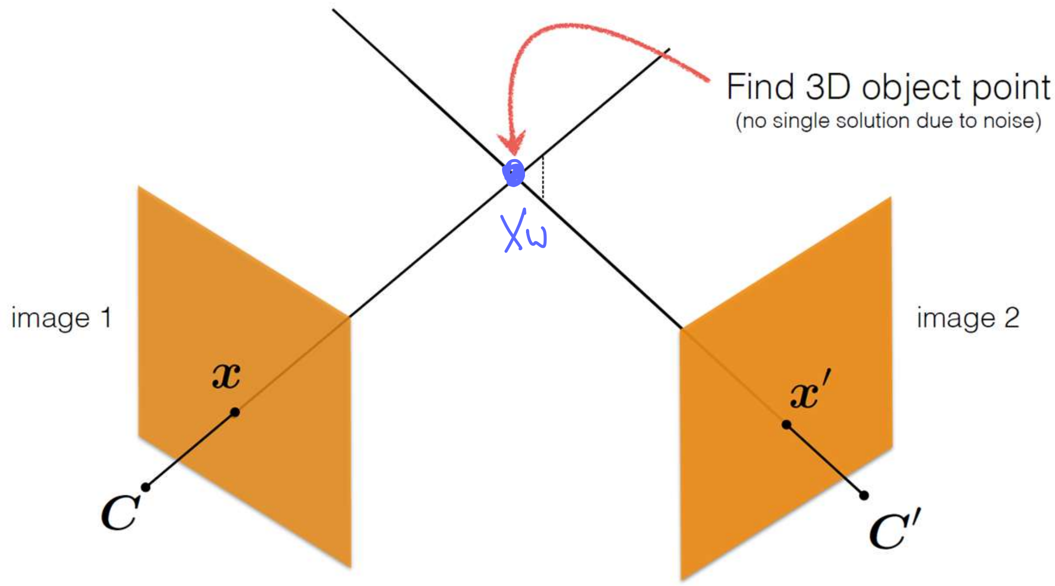
Depth from triangulated points

- Get camera coordinates of a point
- Take z-axis value -> depth!
- We need at least two views to get depth values (unless we rely on learning-based modules)

(u, v, d)

img

depth



$X_w \rightarrow X_{c1}$

$X_w \rightarrow X_{c2}$

Depth map

RGBD

H

W

2. Bundle Adjustment

Bundle Adjustment

- From initial triangulated 3D points, jointly optimizing 3D coordinates and camera parameters by minimizing the reprojection error
- Similar to the non-linear calibration refinement of Lecture 4

① camera calib
② 2D
③ 3D

②③ good
① might not be good
goal to optim

② good
①,③ might not be good.
goal is to optimize these

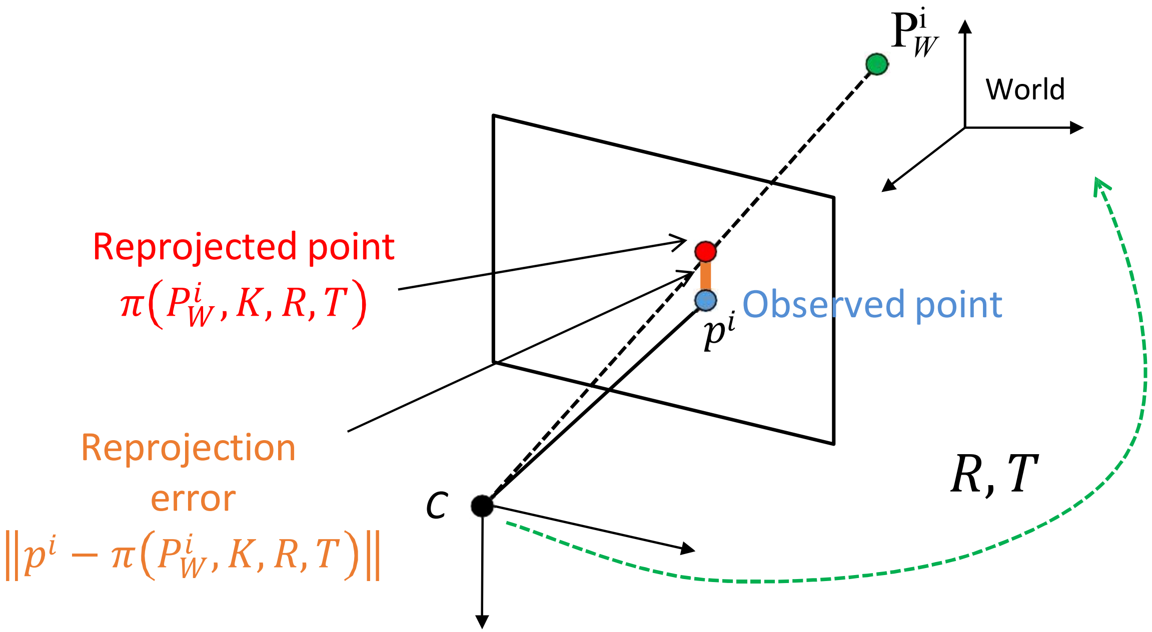
thus. subset.

Reprojection Error

camera calibration

Slide from Lecture 4

- The reprojection error is the **Euclidean distance** (in pixels) between an **observed image point** and the **corresponding** 3D point **reprojected** onto the camera frame.
- The reprojection error gives us a **quantitative measure of the accuracy** of the calibration (**ideally it should be zero**).



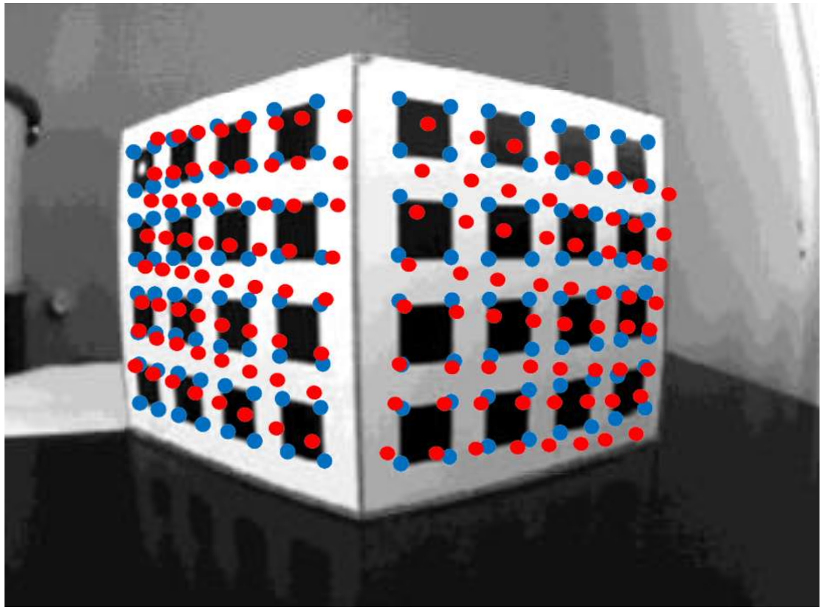
Non-Linear Calibration Refinement

Slide from Lecture 4

- The calibration parameters K, R, T determined by the DLT can be refined by minimizing the following cost:

$$K, R, T, lens\ distortion = argmin_{K,R,T,lens} \sum_{i=1}^{n} \left\| p^i - \pi(P_W^i, K, R, T) \right\|^2$$

- This time we also include the **lens distortion** (can be set to 0 for initialization)
- Can be minimized using **Levenberg–Marquardt** (more robust than Gauss-Newton to local minima)



Control points (observed points)

Reprojected points $\pi(P_W^i, K, R, T)$

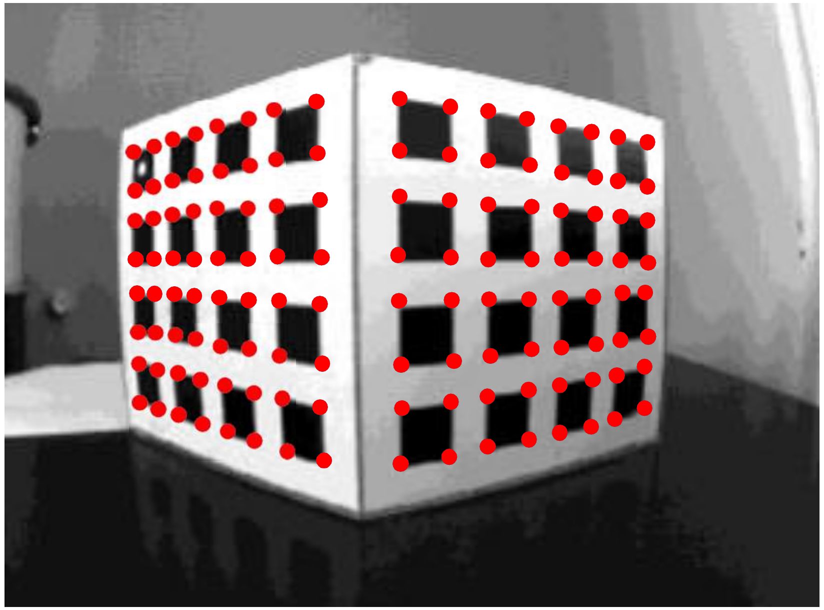
Non-Linear Calibration Refinement

Slide from Lecture 4

- The calibration parameters K, R, T determined by the DLT can be refined by minimizing the following cost:

$$K, R, T, lens\ distortion = argmin_{K,R,T,lens} \sum_{i=1}^{n} \left\| p^i - \pi\left(P_W^i, K, R, T\right) \right\|^2$$

- This time we also include the **lens distortion** (can be set to 0 for initialization)
- Can be minimized using **Levenberg–Marquardt** (more robust than Gauss-Newton to local minima)



Control points (observed points)

Reprojected points $\pi\left(P_W^i, K, R, T\right)$

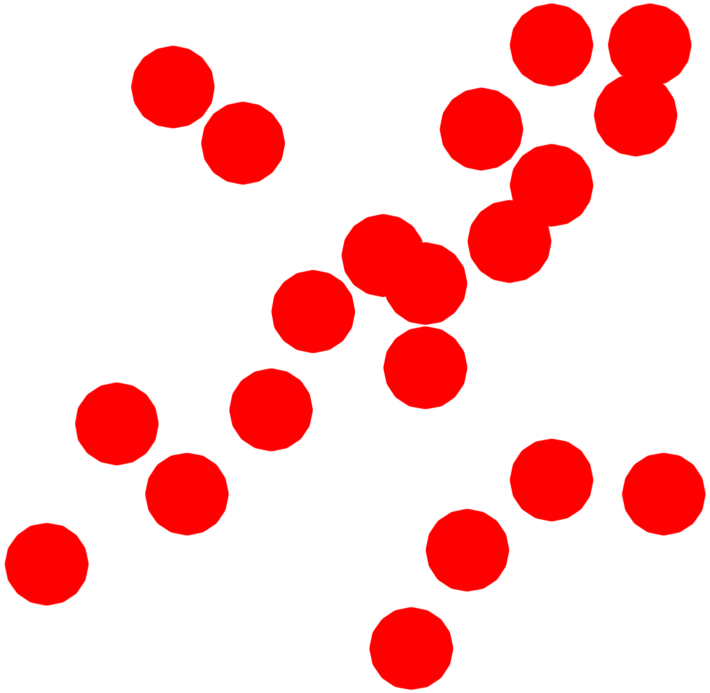
Outlier Rejection

- There could be wrong correspondences across multi-view images -> wrong ones are called ‘outliers’
- We should not use them for the bundle adjustment
- How can we reject outliers? -> Use RANSAC!

RANSAC

(RANdom SAmple Consensus) :

Fischler & Bolles in '81.



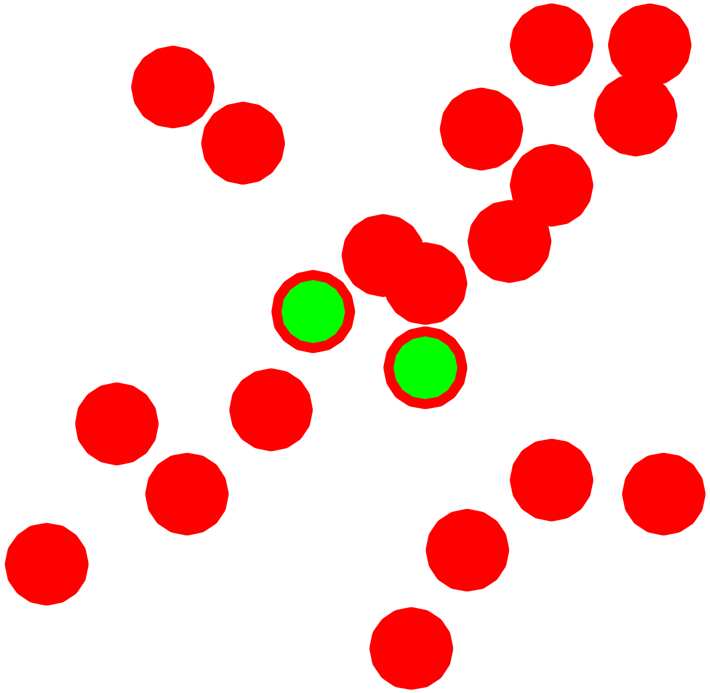
Algorithm:

1. **Sample** (randomly) the number of points required to fit the model
2. **Solve** for model parameters using samples
3. **Score** by the fraction of inliers within a preset threshold of the model

Repeat 1-3 until the best model is found with high confidence

RANSAC

Line fitting example



Algorithm:

linear line.

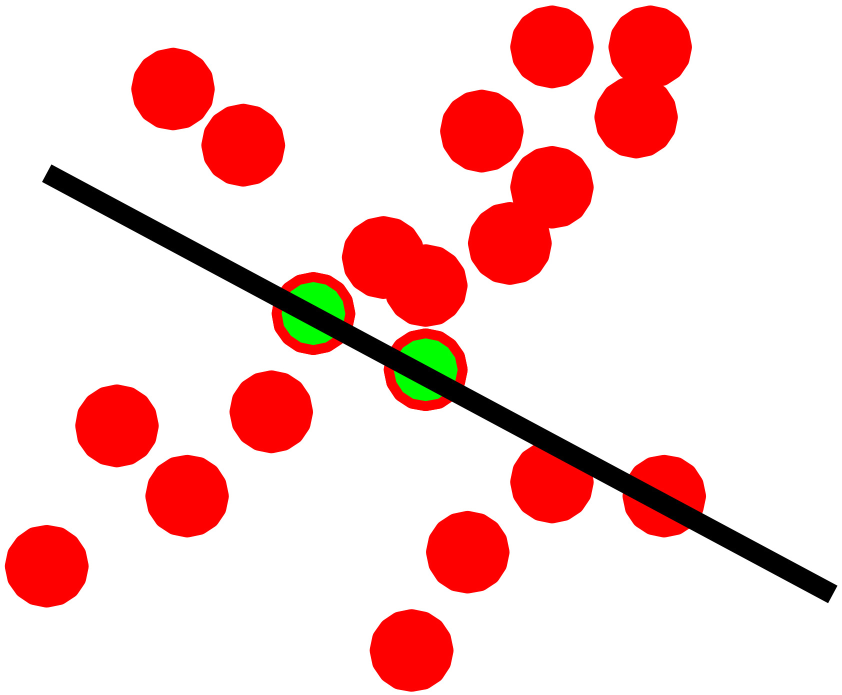
1. **Sample** (randomly) the number of points required to fit the model (#=2)
2. **Solve** for model parameters using samples
3. **Score** by the fraction of inliers within a preset threshold of the model

Repeat 1-3 until the best model is found with high confidence

Illustration by Savarese

RANSAC

Line fitting example



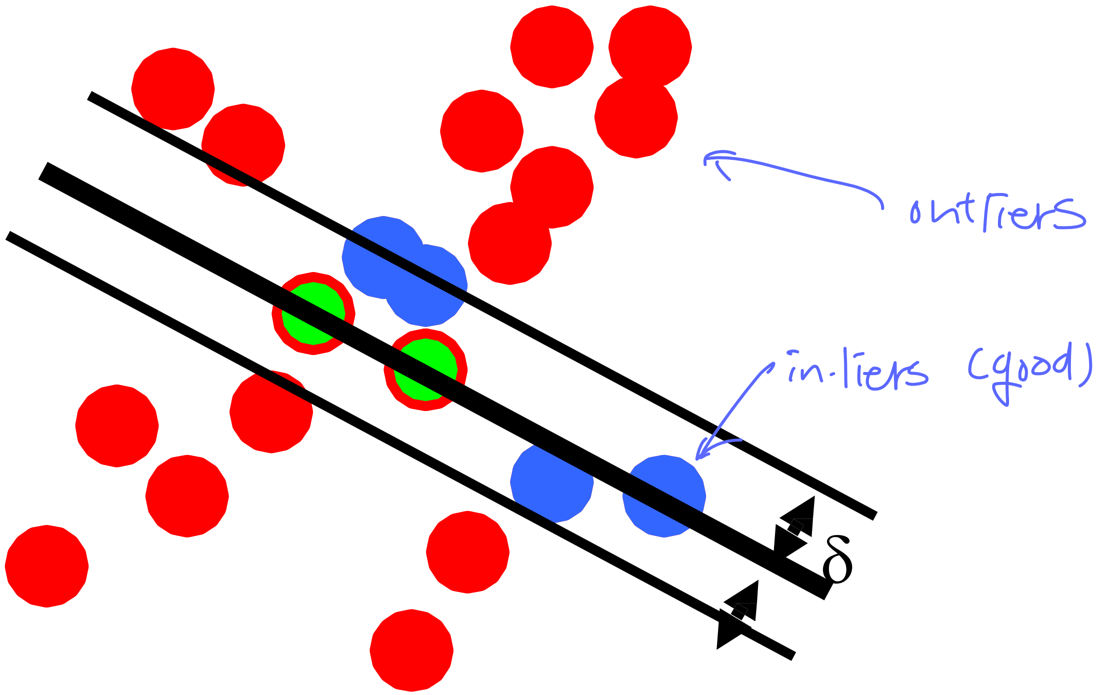
Algorithm:

1. **Sample** (randomly) the number of points required to fit the model (#=2)
2. **Solve** for model parameters using samples
3. **Score** by the fraction of inliers within a preset threshold of the model

Repeat 1-3 until the best model is found with high confidence

RANSAC

Line fitting example



$$N_I = 6$$

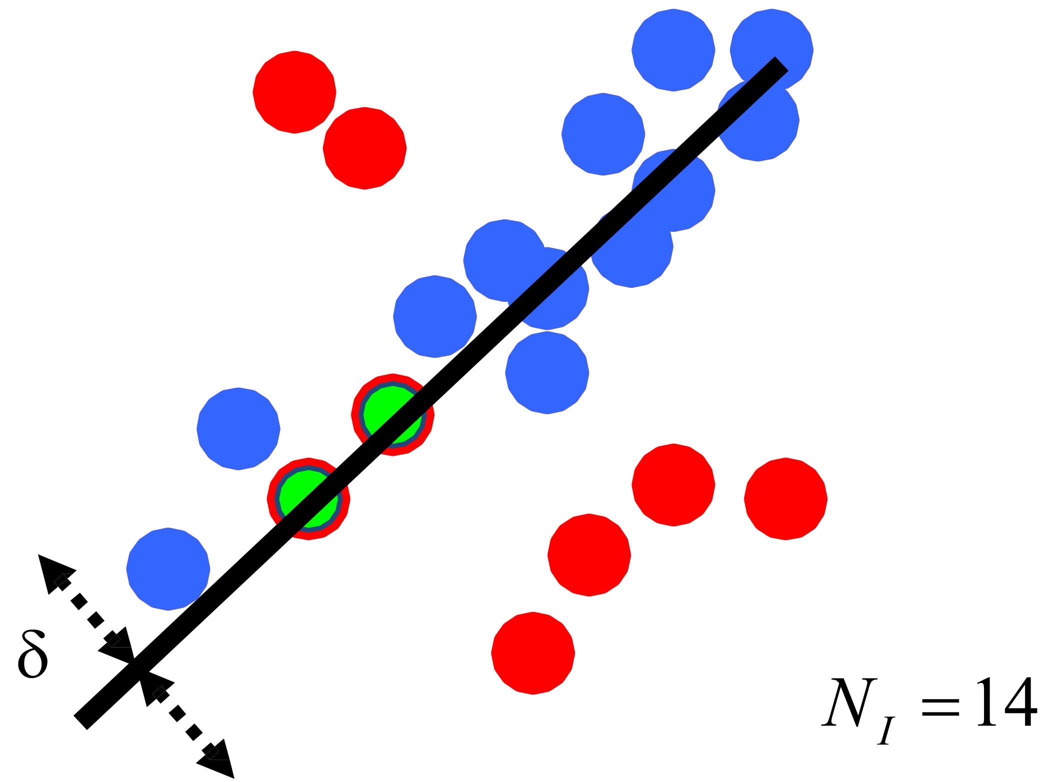
Algorithm:

1. **Sample** (randomly) the number of points required to fit the model (#=2)
2. **Solve** for model parameters using samples
3. **Score** by the fraction of inliers within a preset threshold of the model

Repeat 1-3 until the best model is found with high confidence

keep repeat.

RANSAC



Algorithm:

1. **Sample** (randomly) the number of points required to fit the model (#=2)
2. **Solve** for model parameters using samples
3. **Score** by the fraction of inliers within a preset threshold of the model

Repeat 1-3 until the best model is found with high confidence

How to choose parameters?

- Number of samples *N*
  - Choose *N* so that, with probability *p*, at least one random sample is free from outliers (e.g. p=0.99) (outlier ratio: *e*)
- Number of sampled points *s*
  - Minimum number needed to fit the model
- Distance threshold δ
  - Choose δ so that a good point with noise is likely (e.g., prob=0.95) within threshold
  - Zero-mean Gaussian noise with std. dev. σ: $t^2=3.84\sigma^2$

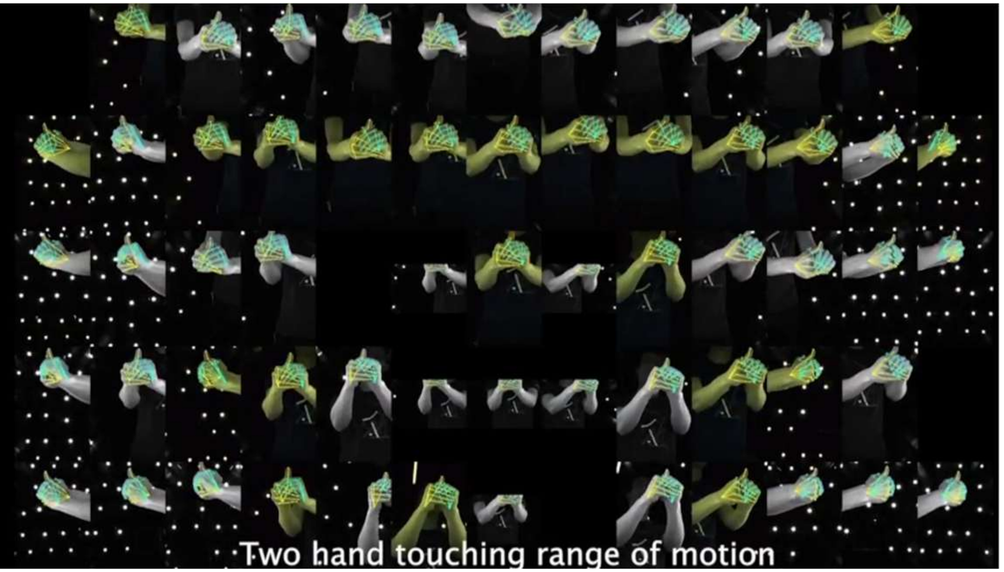
$$\boxed{N = \log(1-p)/\log\left(1-(1-e)^s\right)}$$

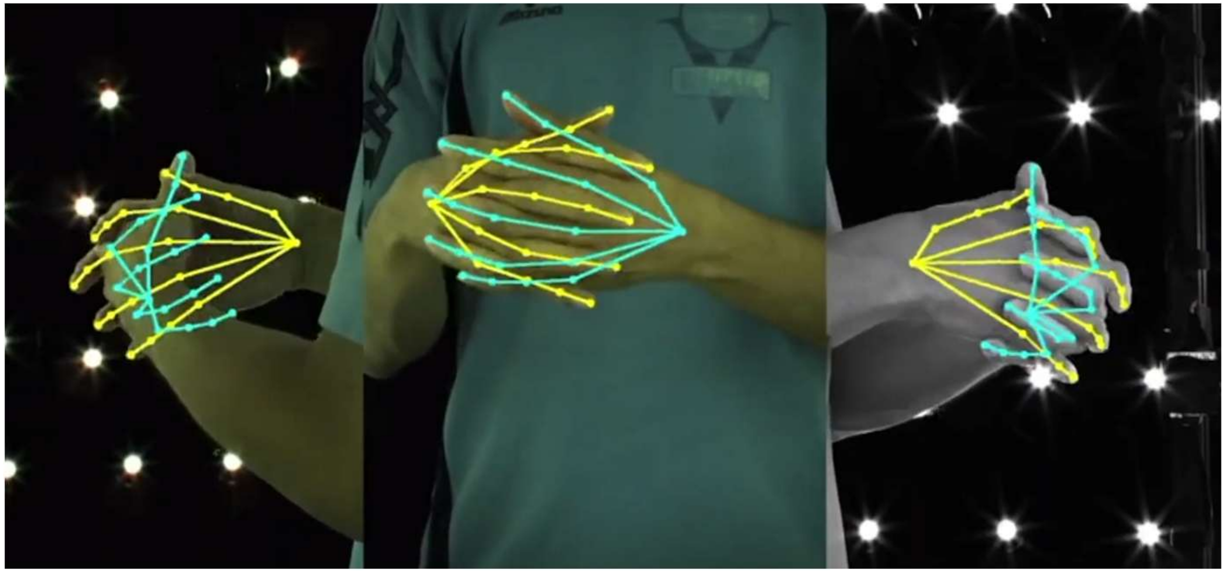
no need to remember

just experimental result

RANSAC for 3D lifting

100 is ideal. but 5-10 is good for human.





Algorithm:

1. Sample (randomly) the number of ~~points~~ view points. required for the triangulation
2. Triangulate the selected 2D points to the 3D space
3. Project the triangulated 3D points to all image space and check reprojection error. Reject viewpoints with huge error.

Repeat 1-3 until the best model is found with high confidence

Depth from triangulated points and bundle adjustment

- Get camera coordinates of a point with triangulation and bundle adjustment
- Take z-axis value -> depth!
- We need at least two views to get depth values (unless we rely on learning-based modules)