Multi-View Geometry - II, III
Homography, Camera Localization, Triangulation

RRC Summer School 2025

Rohit Jayanti

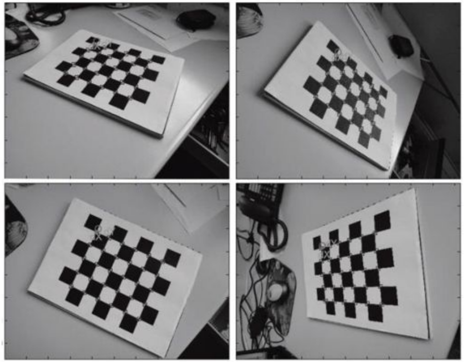
Lecture Objectives

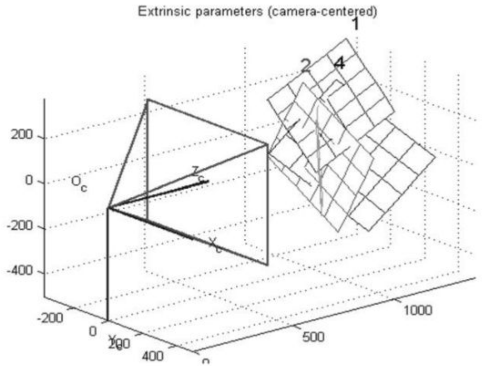
- Homography
  - Zhang's Method (Calibration using Planar grids)
  - Types of 2D Transformations
  - RANSAC
  - Applications
- Camera Localization
  - Perspective-N-Point (PnP)
  - PnP vs DLT
- Triangulation
  - Algebraic Method

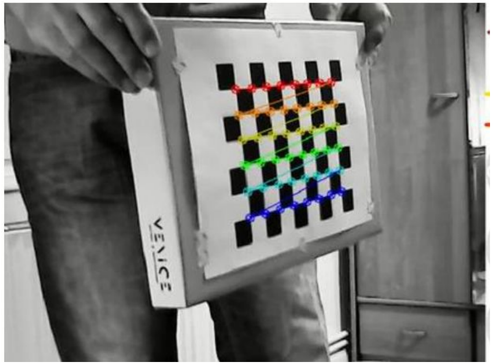
Homography

Zhang’s Method

- From Last Time: Tsai’s Method - requires 3D points in the world that are non-coplanar - not too practical
- CV and Robotics tool-boxes today (OpenCV, ViSP, etc.) support the more convenient Zhang's method (developed in 2000 at Microsoft Research by... well Zhang)
- Uses multiple-views of a planar-grid (commonly a checkerboard)







Zhang's Method

- Again start by writing the perspective projection equation (again, neglecting radial distortion for now). However here are all points are coplanar: $\boldsymbol{Zw} = \mathbf{0}$

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K\left[R \mid T\right] \cdot \begin{bmatrix} X_w \\ Y_w \\ 0 \\ 1 \end{bmatrix} \quad \Rightarrow$$

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

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

Zhang's Method

- Again start by writing the perspective projection equation (again, neglecting radial distortion for now). However here are all points are coplanar: $\boldsymbol{Zw} = \boldsymbol{0}$

$$\lambda\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K\left[R \mid T\right]\cdot\begin{bmatrix} X_w \\ Y_w \\ 0 \\ 1 \end{bmatrix} \quad \Rightarrow$$

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

$$\Rightarrow \lambda\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix}\cdot\begin{bmatrix} X_w \\ Y_w \\ 1 \end{bmatrix}$$

Zhang's Method

- Again start by writing the perspective projection equation (again, neglecting radial distortion for now). However here are all points are coplanar: $\boldsymbol{Zw} = \mathbf{0}$

$$\Rightarrow \lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \cdot \begin{bmatrix} X_w \\ Y_w \\ 1 \end{bmatrix}$$

$$\Rightarrow \lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = H \cdot \begin{bmatrix} X_w \\ Y_w \\ 1 \end{bmatrix}$$

This matrix is called Homography

where h_i^{T} is the i-*th* row of H

$$\Rightarrow \lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} h_1^{\mathrm{T}} \\ h_2^{\mathrm{T}} \\ h_3^{\mathrm{T}} \end{bmatrix} \cdot \begin{bmatrix} X_w \\ Y_w \\ 1 \end{bmatrix}$$

Zhang's Method

$$\Rightarrow \lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} h_1^{\mathrm{T}} \\ h_2^{\mathrm{T}} \\ h_3^{\mathrm{T}} \end{bmatrix} \cdot \boxed{\begin{bmatrix} X_w \\ Y_w \\ 1 \end{bmatrix}} \rightarrow \boldsymbol{P}$$

- Conversion back from homogeneous coordinates to pixel coordinates gives:

$$\begin{aligned} u &= \frac{\lambda u}{\lambda} = \frac{h_1^{\mathrm{T}} \cdot P}{h_3^{\mathrm{T}} \cdot P} \\ v &= \frac{\lambda v}{\lambda} = \frac{h_2^{\mathrm{T}} \cdot P}{h_3^{\mathrm{T}} \cdot P} \end{aligned} \quad \Rightarrow \quad \begin{aligned} (h_1^{\mathrm{T}} - u_i h_3^{\mathrm{T}}) \cdot P_i &= 0 \\ (h_2^{\mathrm{T}} - v_i h_3^{\mathrm{T}}) \cdot P_i &= 0 \end{aligned}$$

Zhang's Method

- Rearranging the terms, we obtain:

$$\begin{matrix} (h_1^T - u_i h_3^T) \cdot P_i = 0 \\ (h_2^T - v_i h_3^T) \cdot P_i = 0 \end{matrix} \quad \Rightarrow \quad \begin{matrix} P_i^T \cdot h_1 + 0 \cdot h_2^T - u_i P_i^T \cdot h_3^T = 0 \\ 0 \cdot h_1^T + P_i^T \cdot h_2^T - v_i P_i^T \cdot h_3^T = 0 \end{matrix} \quad \Rightarrow \begin{pmatrix} P_i^T & 0^T & -u_1 P_i^T \\ 0^T & P_i^T & -v_1 P_i^T \end{pmatrix} \begin{pmatrix} h_1 \\ h_2 \\ h_3 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

- For n points (from a single view), we can stack all equations into a big matrix:

$$\underbrace{\begin{pmatrix} P_1^T & 0^T & -u_1 P_1^T \\ 0^T & P_1^T & -v_1 P_1^T \\ \cdots & \cdots & \cdots \\ P_n^T & 0^T & -u_n P_n^T \\ 0^T & P_n^T & -v_n P_n^T \end{pmatrix}}_{\text{Q (this matrix is \textbf{known})}} \underbrace{\begin{pmatrix} h_1 \\ h_2 \\ h_3 \end{pmatrix}}_{\text{H (this matrix is \textbf{unknown})}} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{pmatrix} \Rightarrow \quad \mathbf{Q} \cdot \mathbf{H} = 0$$

Applying DLT (again)

$$\mathrm{Q} \cdot \mathrm{H} = 0$$

- Minimal Solution:
    - Q ($2n \times 9$) should have rank 8 to have a unique (up to a scale) non-trivial solution H
    - Each point correspondence provides 2 independent equations
    - Thus, a minimum of 4 non-collinear points is required
- Solution for $\boldsymbol{n \geq 4}$ points
    - It can be solved through Singular Value Decomposition (SVD) (same considerations as before)

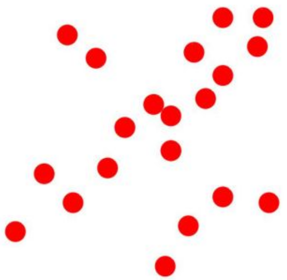
Decomposing H into K, R, and T

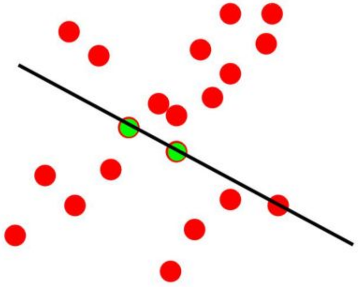
- H can be decomposed by recalling that: $$\begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} = \begin{bmatrix} \alpha_u & 0 & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} r_{11} & r_{12} & t_1 \\ r_{21} & r_{22} & t_2 \\ r_{31} & r_{32} & t_3 \end{bmatrix}$$

- In practice the more views the better, e.g., 20-50 views spanning the entire field of view of the camera for the best calibration results!
- Notice that now each view j has a different homography *Hj* (and so a different *Rj* and Tj). However, $\boldsymbol{K}$ is the same for all views:

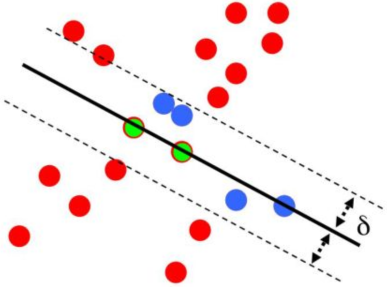
$$\begin{bmatrix} h_{11}^j & h_{12}^j & h_{13}^j \\ h_{21}^j & h_{22}^j & h_{23}^j \\ h_{31}^j & h_{33}^j & h_{33}^j \end{bmatrix} = \begin{bmatrix} \alpha_u & 0 & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} r_{11}^j & r_{12}^j & t_1^j \\ r_{21}^j & r_{22}^j & t_2^j \\ r_{31}^j & r_{32}^j & t_3^j \end{bmatrix}$$

A small gotcha and RANSAC





Randomly sample points and fit model.



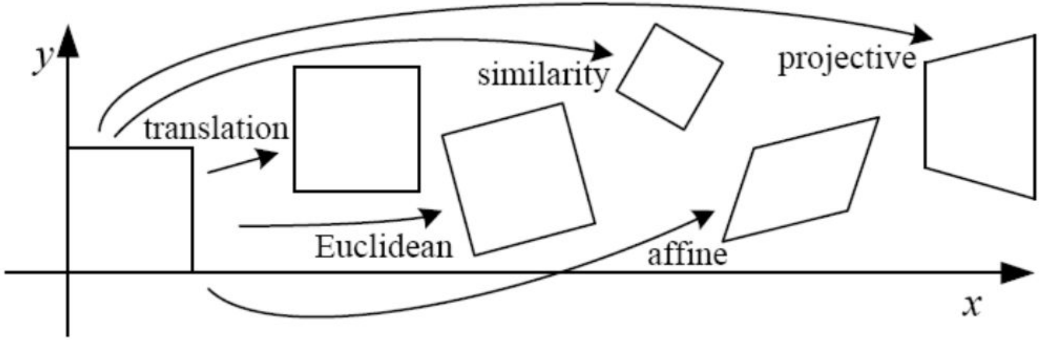
Set thresh and count inliers.

Repeat the above steps for N times. Choose the model which has highest inliers.

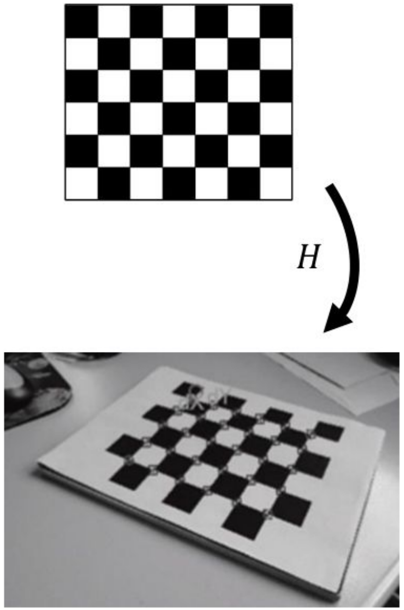
Calculating H with RANSAC

- Find keypoints and match the descriptors to get correspondences. RANSAC outer loop (N samples) as follows :
    1. Choose 4 correspondences.
    2. Compute H.
    3. Find no. of inliers. (based on error e = l2 norm(Hx - x’))
- Choose H_i with highest inliers.
- Recompute H using the model with highest inliers (can be further refined using LM solver).

Types of 2D Transformations



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



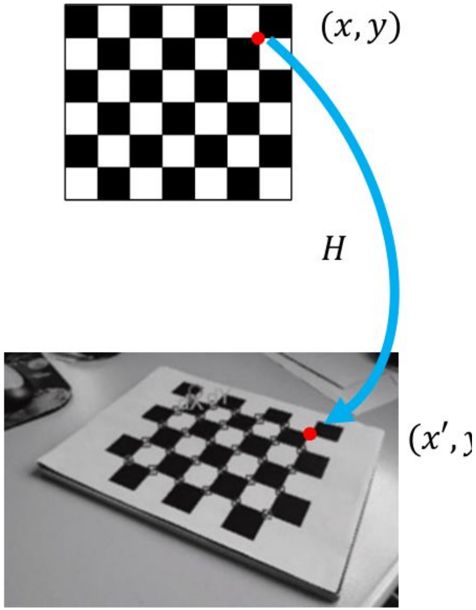
This matrix is called **Homography**

Types of 2D Transformations

- A point (x, y) is transformed into x', y' via:

$$x' = \frac{a_1 x + a_2 y + a_3}{a_7 x + a_8 y + 1}$$

$$y' = \frac{a_4 x + a_5 y + a_6}{a_7 x + a_8 y + 1}$$

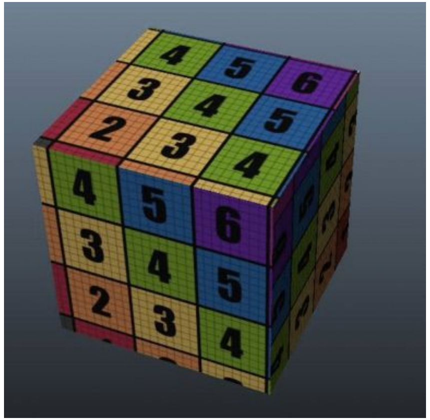


- Homogeneous Coordinates:

$$\lambda \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 \\ a_7 & a_8 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Homography: Applications

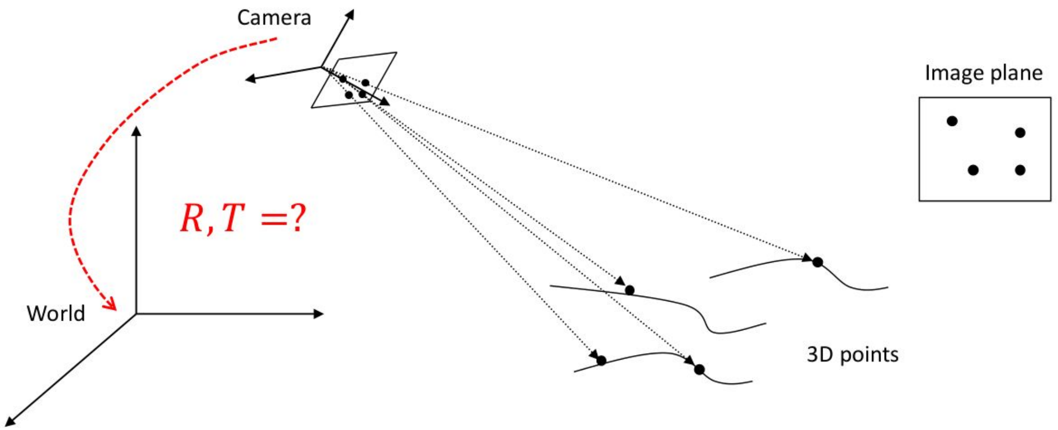
- Removing perspective distortion
- Rendering Planar Textures and Shadows
- Planar Object Tracking (Augmented Reality)
- See [Aruco Markers] and [AprilTag]



Camera Localization

Perspective from n Points (or PnP)

- This is the problem of determining the 6DoF pose of a camera (position and orientation) with respect to the world frame from a set of 3D-2D point correspondences assuming intrinsics are known.
- The DLT can be used to solve this problem but is suboptimal. We want to study algebraic solutions to the problem.

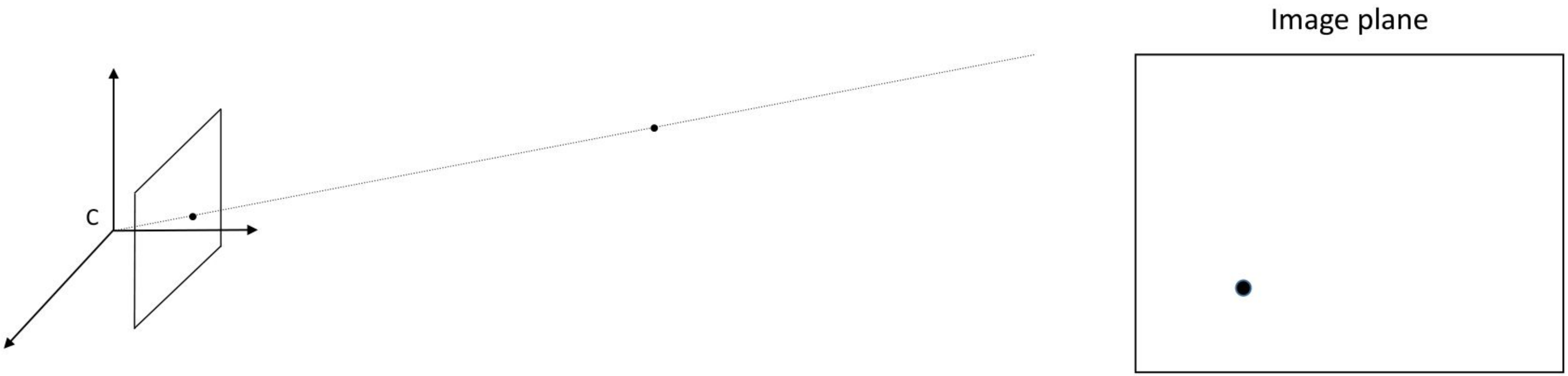


How many points are enough ?

- 1 point:
  - Infinite Solutions
- 2 points:
  - Infinitely many solutions, but bounded
- 3 Points (non collinear):
  - Up to 4 solutions
- 4 Points:
  - Unique solution

Perspective-1-Point

- 1 point:
  - Infinite Solutions



Perspective-2-Point

- 2 Points:
    - Infinite Solutions

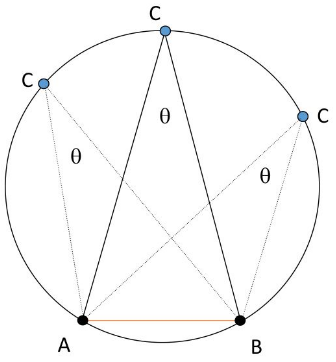
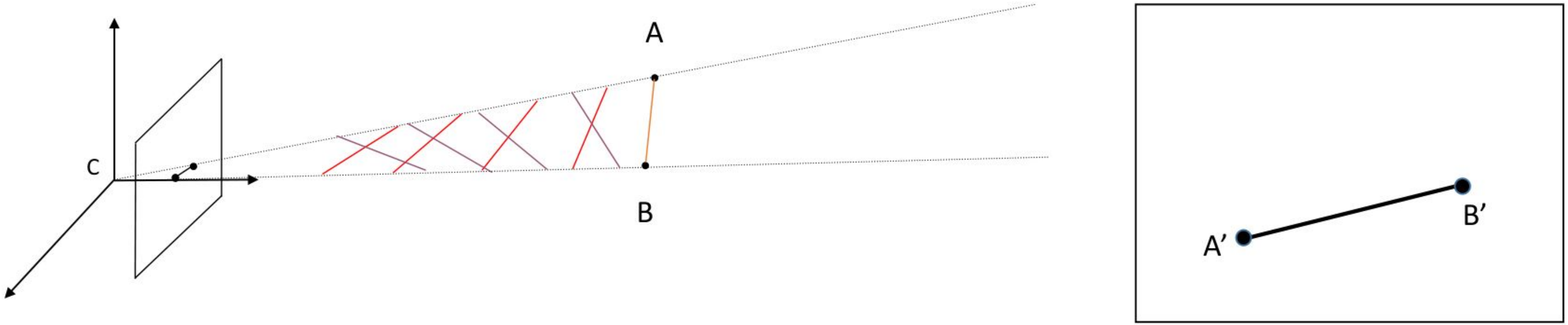


Image plane



Perspective-3-Point (**P3P**)

- 3 Points (non collinear):
  - up to 4 solutions

$$s_1^2 = L_B^2 + L_A^2 - 2L_B L_A \cos\theta_{AB}$$

$$s_2^2 = L_A^2 + L_C^2 - 2L_A L_C \cos\theta_{AC}$$

$$s_3^2 = L_B^2 + L_C^2 - 2L_B L_C \cos\theta_{BC}$$

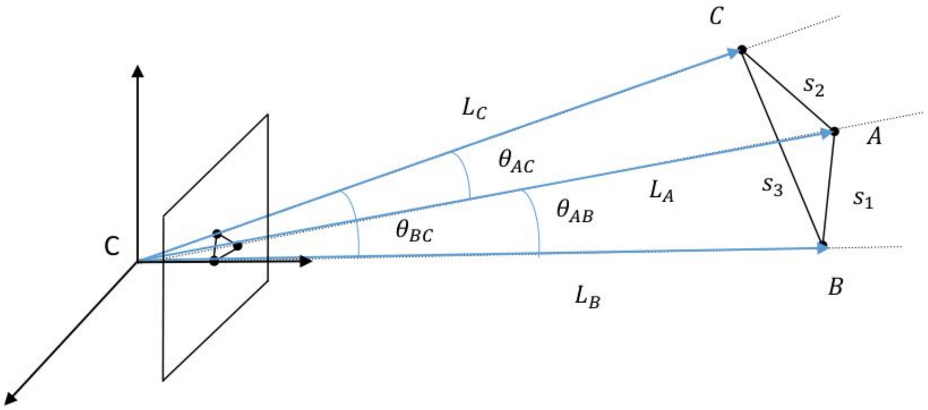
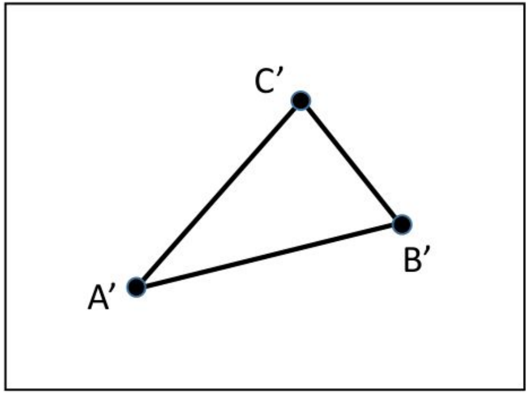
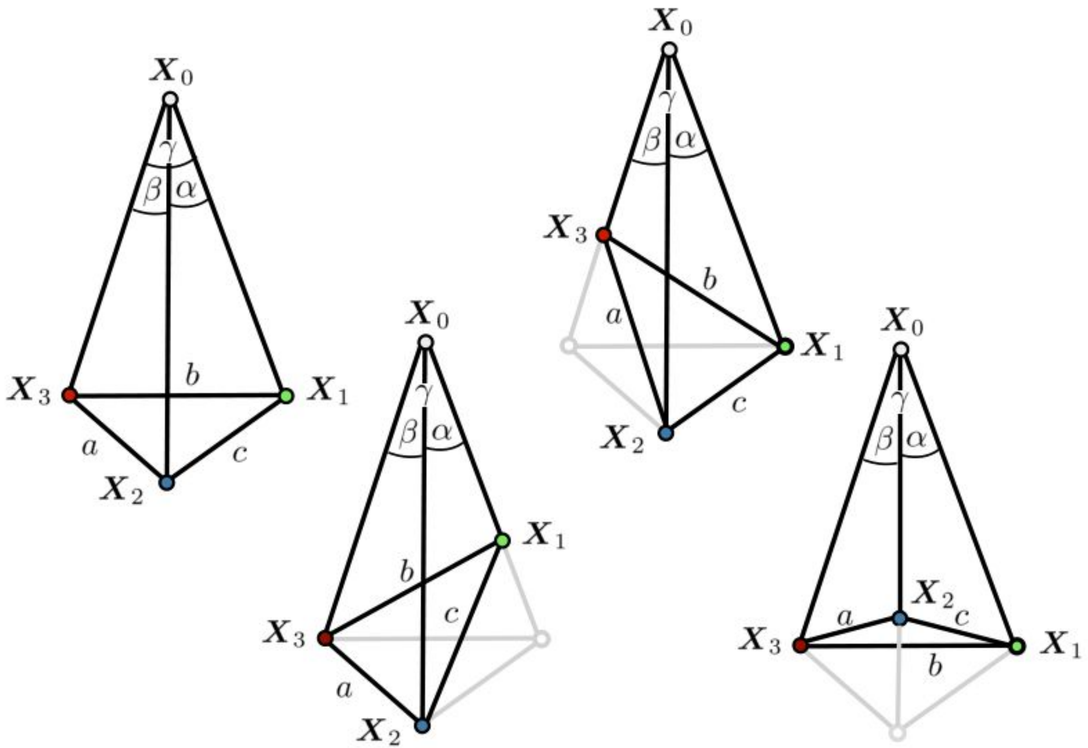


Image plane



Perspective-3-Point (**P3P**) - 4 solutions



Perspective-3-Point (**P3P**)

$$s_1^2 = L_B^2 + L_A^2 - 2L_BL_A\cos\theta_{AB}$$

$$s_2^2 = L_A^2 + L_C^2 - 2L_AL_C\cos\theta_{AC}$$

$$s_3^2 = L_B^2 + L_C^2 - 2L_BL_C\cos\theta_{BC}$$

- It is known that $\boldsymbol{n}$ independent polynomial equations, in $\boldsymbol{n}$ unknowns, can have no more solutions than the product of their respective degrees. Thus, the system can have a maximum of 8 solutions. However, because every term in the system is either a constant or of second degree, for every real positive solution there is a negative solution.
- Thus, with 3 points, there are at most 4 valid (positive) solutions.

Perspective-3-Point (**P3P**)

$$s_1^2 = L_B^2 + L_A^2 - 2L_B L_A \cos\theta_{AB}$$

$$s_2^2 = L_A^2 + L_C^2 - 2L_A L_C \cos\theta_{AC}$$

$$s_3^2 = L_B^2 + L_C^2 - 2L_B L_C \cos\theta_{BC}$$

- By defining $\boldsymbol{x} = \boldsymbol{LB/LA}$, it can be shown that the system can be reduced to a 4th order equation:

$$G_0 + G_1 x + G_2 x^2 + G_3 x^3 + G_4 x^4 = 0$$

- How can we disambiguate the 4 solutions? How do we determine R and T?

Perspective-3-Point (**P3P**)

$$s_1^2 = L_B^2 + L_A^2 - 2L_B L_A \cos\theta_{AB}$$

$$s_2^2 = L_A^2 + L_C^2 - 2L_A L_C \cos\theta_{AC}$$

$$s_3^2 = L_B^2 + L_C^2 - 2L_B L_C \cos\theta_{BC}$$

- By defining $\boldsymbol{x} = \boldsymbol{LB/LA}$, it can be shown that the system can be reduced to a 4th order equation:

$$G_0 + G_1 x + G_2 x^2 + G_3 x^3 + G_4 x^4 = 0$$

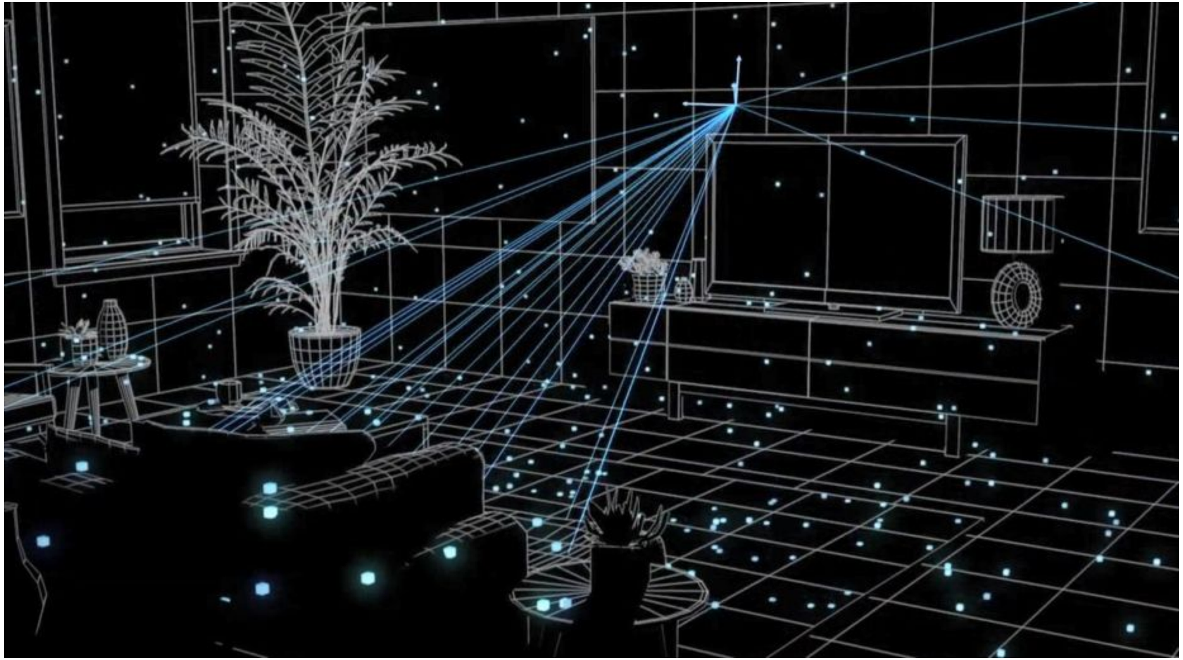
- How can we disambiguate the 4 solutions? How do we determine R and T?
- A 4th point can be used to disambiguate the solutions. A classification of the four solutions and the determination of R and T from the point distances was given by Gao's algorithm, implemented in OpenCV (solvePnP_P3P)

Self Study - Modern Solutions to P3P and n >= 4

- A more modern version of P3P was developed by Kneip in 2011 and directly solves for the camera's pose (not distances from the points). This solution inspired the algorithm currently used in OpenCV (solvePnP_AP3P)
- An efficient algebraic solution to the PnP problem for $n \geq 4$ was developed by Lepetit in 2009 and was named EPnP (Efficient PnP) and can be found in OpenCV (solvePnP_EPnP)
    - EPnP expresses the n world's points as a weighted sum of four virtual control points

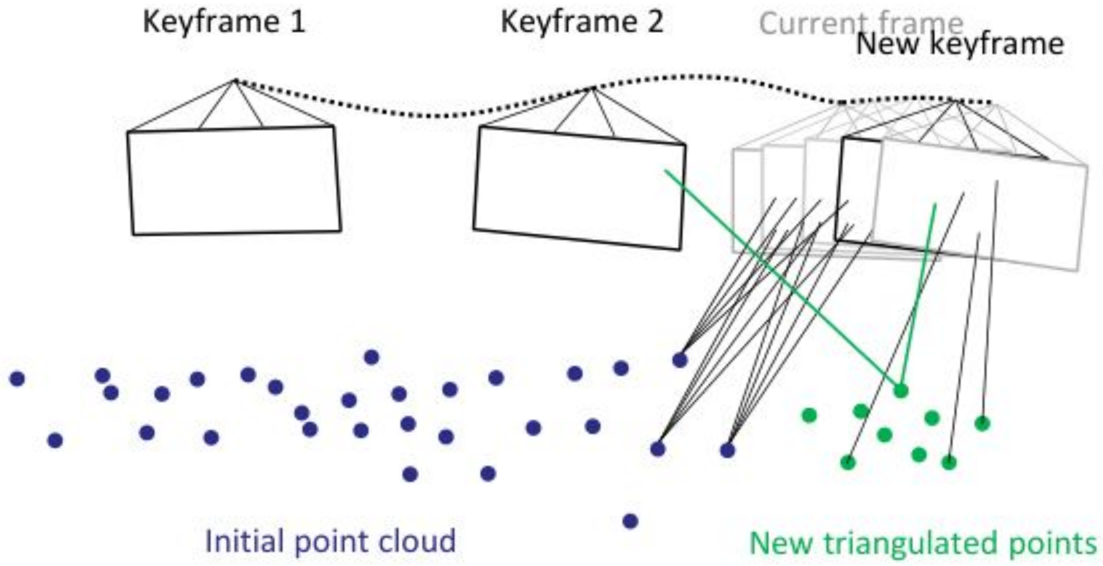
PnP: Applications and Discussion

- Localization: Given a 3D point cloud (map), determine the pose of the camera



PnP: Applications and Discussion

- Localization: Given a 3D point cloud (map), determine the pose of the camera

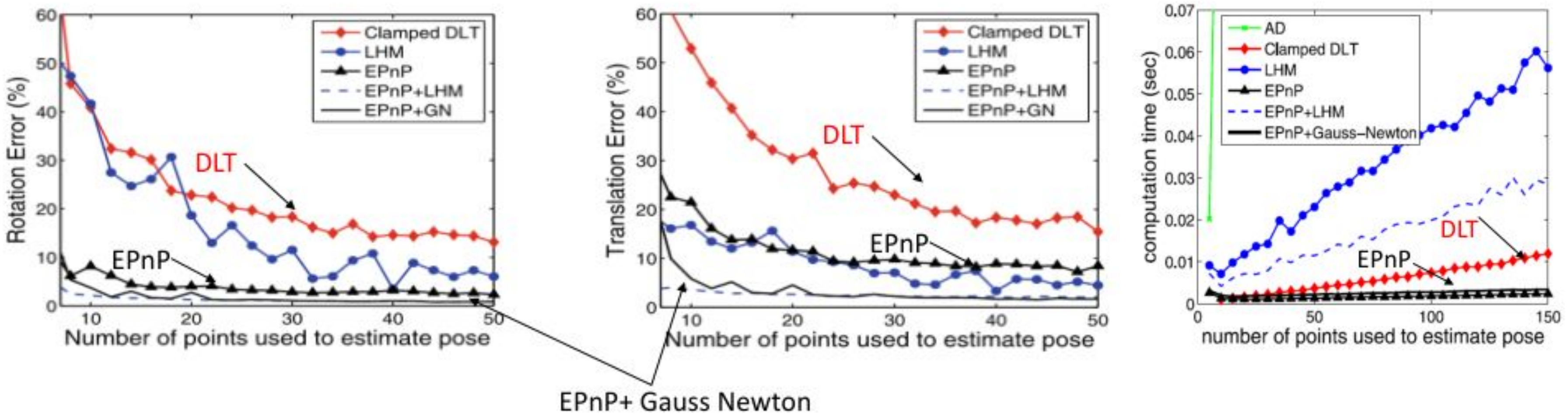


PnP: Applications and Discussion

- All PnP problems (solved by DLT, EPnP, or P3P algorithms) are prone to errors if there are outliers in the set of 3D-2D point correspondences.
- PnP with RANSAC can be found in OpenCV's (solvePnPRansac)

EPnP vs DLT

- If a camera is calibrated, only R and T need to be determined. In this case, should we use DLT or EPnP?
- EPnP is up to **10x more** [robust to noise, accurate, and efficient]



EPnP vs DLT

Calibrated camera (i.e., instrinc parameters are known)	**Uncalibrated camera** (i.e., intrinsic parameters unknown)
Either DLT or EPnP can be used	**Only DLT** can be used

EPnP: minimum number of points: **3 (P3P) +1** for disambiguation
DLT: Minimum number of points: **4 if coplanar, 6 if non-coplanar**

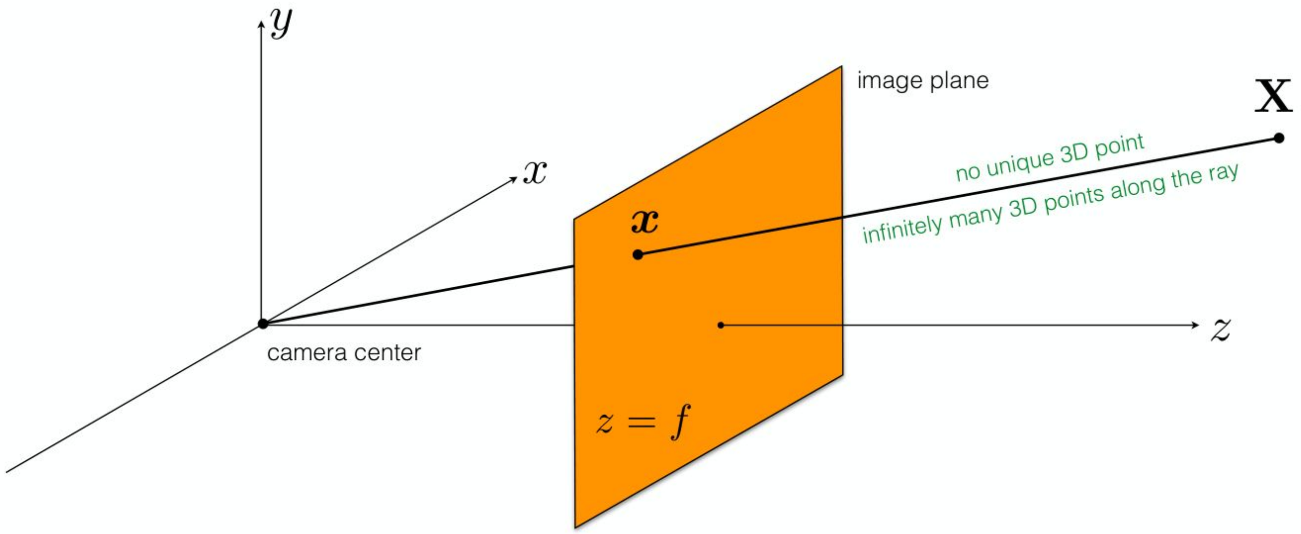
The output of both DLT and EPnP can be refined via **non-linear optimization** by minimizing the sum of squared reprojection errors

Triangulation

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

known known

*Can we compute **X** from a single correspondence **x**?*


y
image plane
$\mathbf{X}$
no unique 3D point
infinitely many 3D points along the ray
x
$\boldsymbol{x}$
z
camera center
$z = f$

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

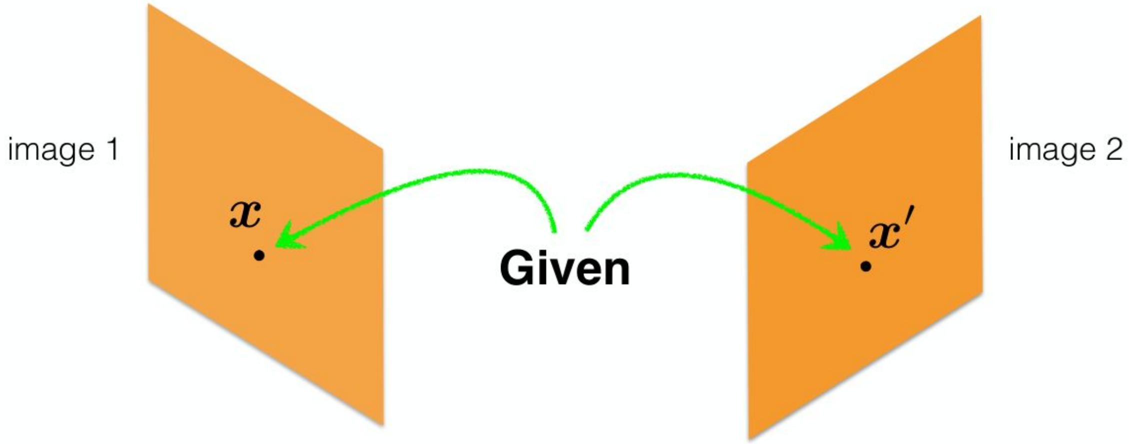
known known

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

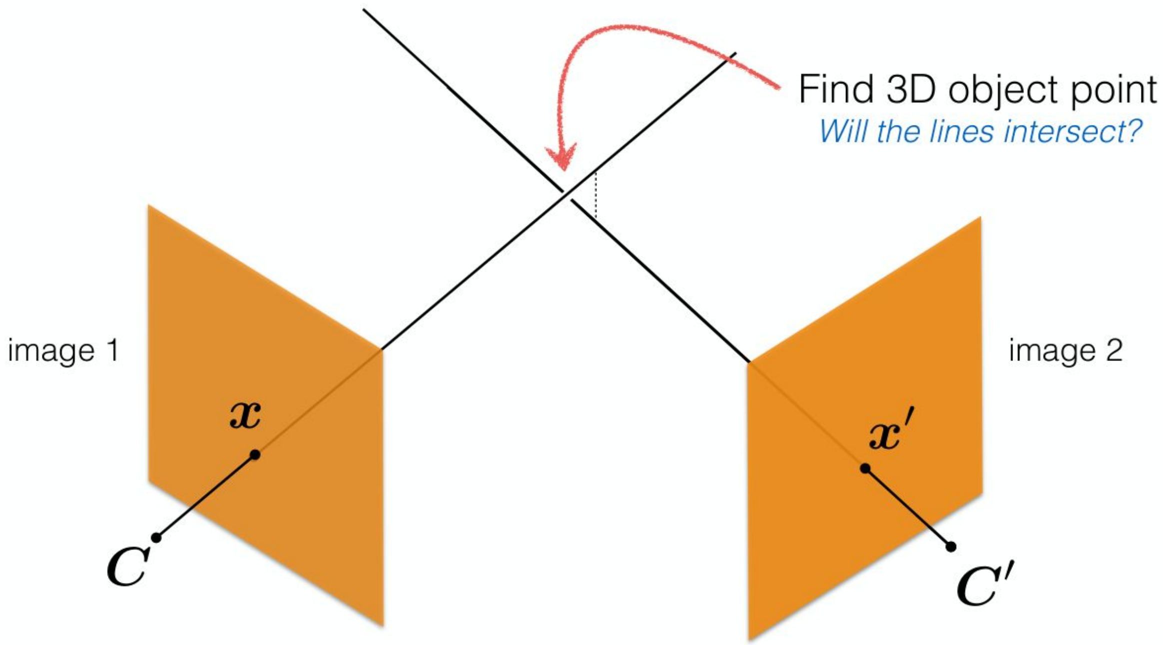
$$P = KR\left[I \mid -X_0\right]$$

$$\begin{bmatrix} 1 & 0 & 0 & -x_0 \\ 0 & 1 & 0 & -y_0 \\ 0 & 0 & 1 & -z_0 \end{bmatrix}$$

Triangulation



Triangulation



Triangulation

Given a set of (noisy) matched points

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

and camera matrices

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

Estimate the 3D point

$$\mathbf{X}$$

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

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

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

Need to find the **best fit**

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

(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

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

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

Remove scale factor, convert to linear system and solve with

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

(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

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

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

Same direction but differs by a scale factor

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

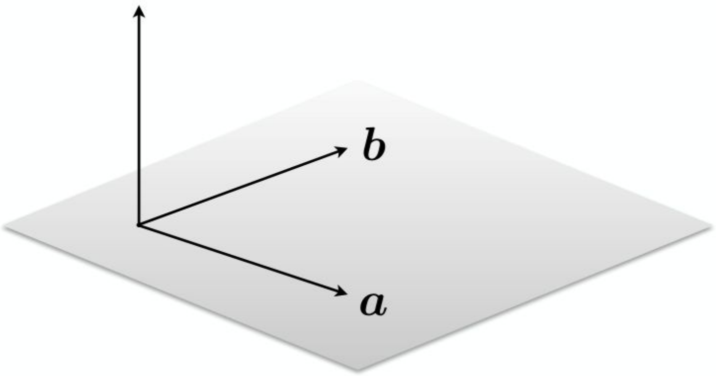
Cross product of two vectors of same direction is zero
(this equality removes the scale factor)

Recall: Cross Product

Vector (cross) product

takes two vectors and returns a vector perpendicular to both

$$\boldsymbol{c} = \boldsymbol{a} \times \boldsymbol{b}$$



$$\boldsymbol{c} \cdot \boldsymbol{a} = 0 \qquad \boldsymbol{c} \cdot \boldsymbol{b} = 0$$

$$\boldsymbol{a} \times \boldsymbol{b} = \begin{bmatrix} a_2 b_3 - a_3 b_2 \\ a_3 b_1 - a_1 b_3 \\ a_1 b_2 - a_2 b_1 \end{bmatrix}$$

cross product of two vectors in the same direction is zero

$$\boldsymbol{a} \times \boldsymbol{a} = 0$$

remember this!!!

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

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \alpha \begin{bmatrix} \text{---} & \boldsymbol{p}_1^\top & \text{---} \\ \text{---} & \boldsymbol{p}_2^\top & \text{---} \\ \text{---} & \boldsymbol{p}_3^\top & \text{---} \end{bmatrix} \begin{bmatrix} | \\ \boldsymbol{X} \\ | \end{bmatrix}$$

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

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

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

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

sanity check! dimensions?

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

How do we solve homogeneous linear system?

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 !

Resources

- Cyrill Stachniss' Mobile Sensing and Robotics II (2021) [Youtube]
  - Lectures 29-31: Zhang's Method and P3P
- Steven Lavalle's brilliant lecture series on Virtual Reality through NPTEL [Youtube]
  - Lecture Video 49 [Youtube]: Perspective n-point problem
  - Fun fact: Founding Chief Scientist at Oculus VR (now Meta)

Acknowledgments:
Slides and Images adapted from Andreas Geiger, Wikiwand, David Scaramuzza, RRC SS-23/24

Missing Topics and Inverted Classroom Approach:

- Self Study on Epipolar Geometry from Cyrill Stachniss' MSR-II Playlist
    - Fundamental and Essential Matrix Intro [Lec 32]
    - Relative Orientation and Properties of F, E [Lec 33]
    - Epipolar Geometry Construction [Lec 35]
    - Estimating F, E using 8 Point Algorithm / Nister's 5 Point Algorithm [Lec 36][Lec 37]

- Doubts and Discussion on the `#module-4-multiview-geometry` channel over at Slack