CIS 5800

Machine Perception

Instructor: Lingjie Liu

Lec 9: Feb 24, 2025

Recap: Projective Transformation = Homography = Collineation=Projectivity

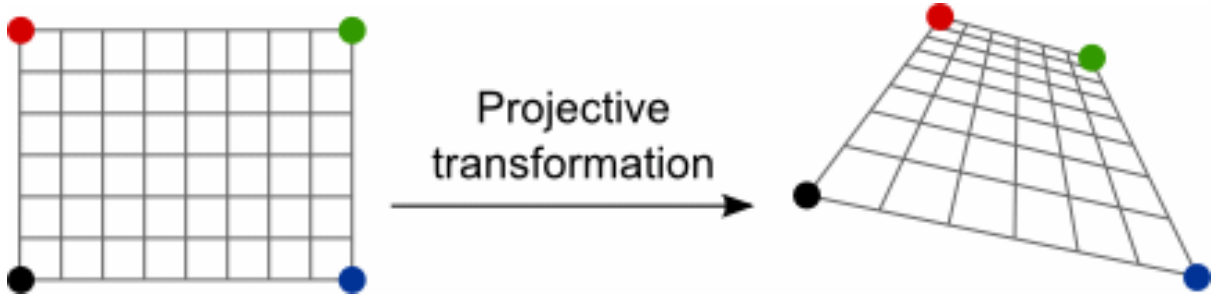
Definition

A **projective transformation** is any invertible matrix transformation $\mathbb{P}^2 \to \mathbb{P}^2$.

A projective transformation H maps p to $p' \sim Hp$

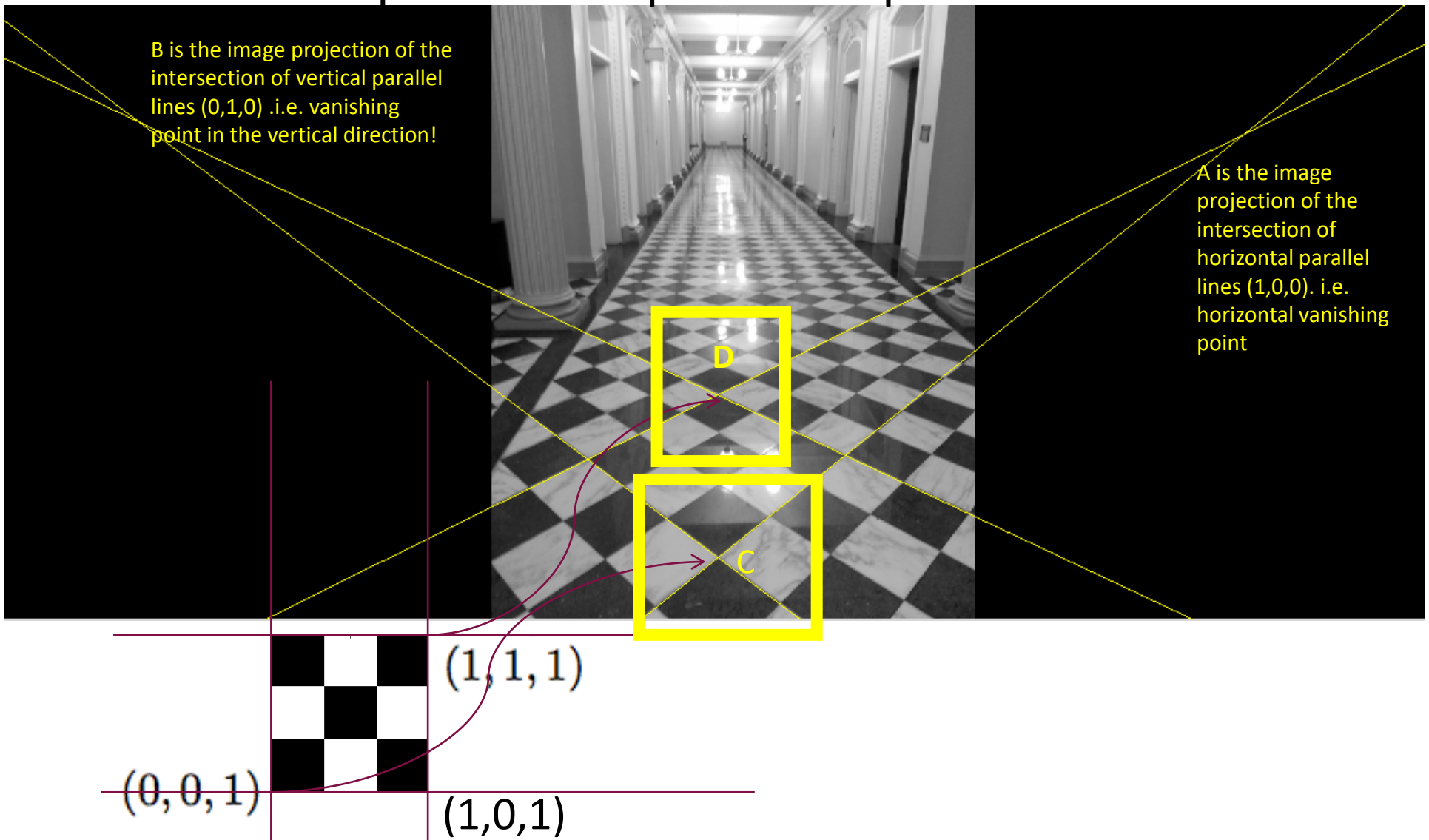
Invertibility means that $\det(H) \neq 0$ and that there exists $\lambda \neq 0$ such that $\lambda p' = Hp$

Observe that we will write either $p' \sim Hp$ or $\lambda p' = Hp$

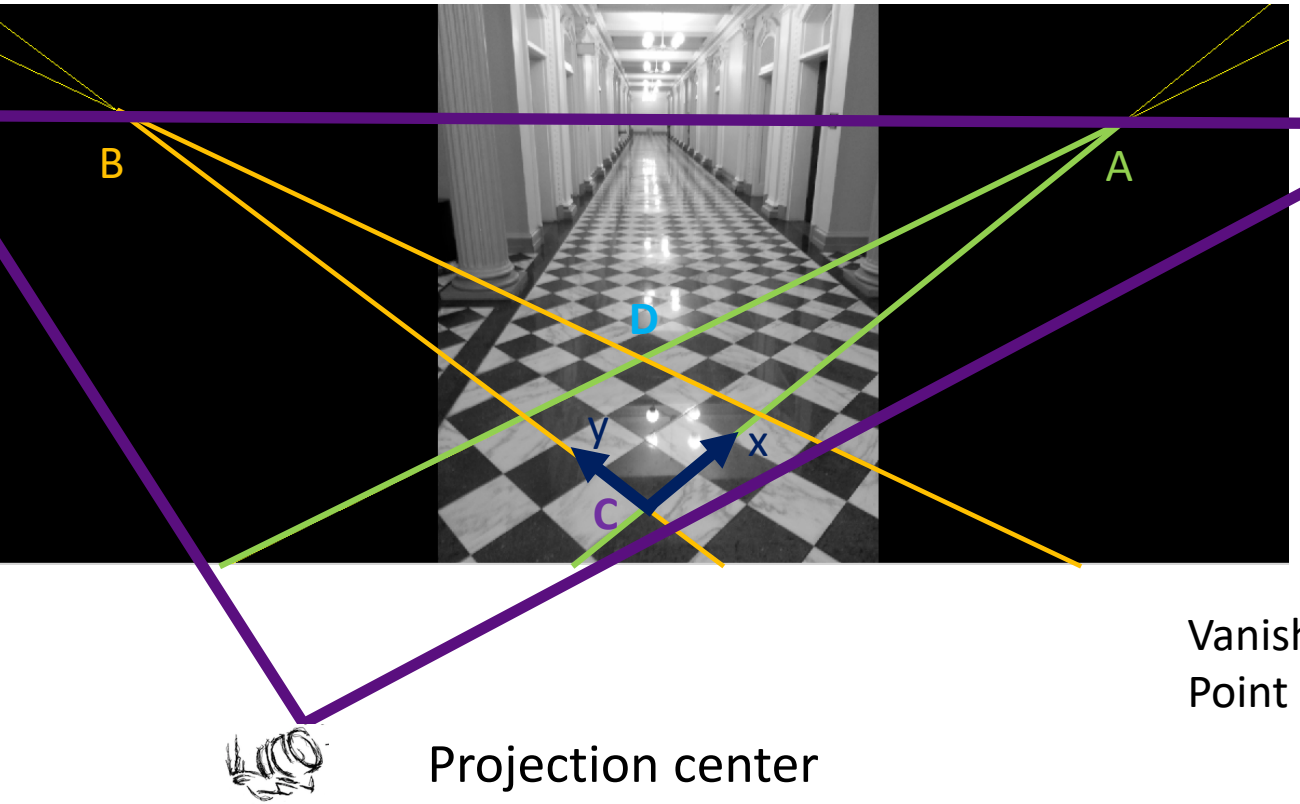


Recap: Computing Homography from 4 Point Correspondences

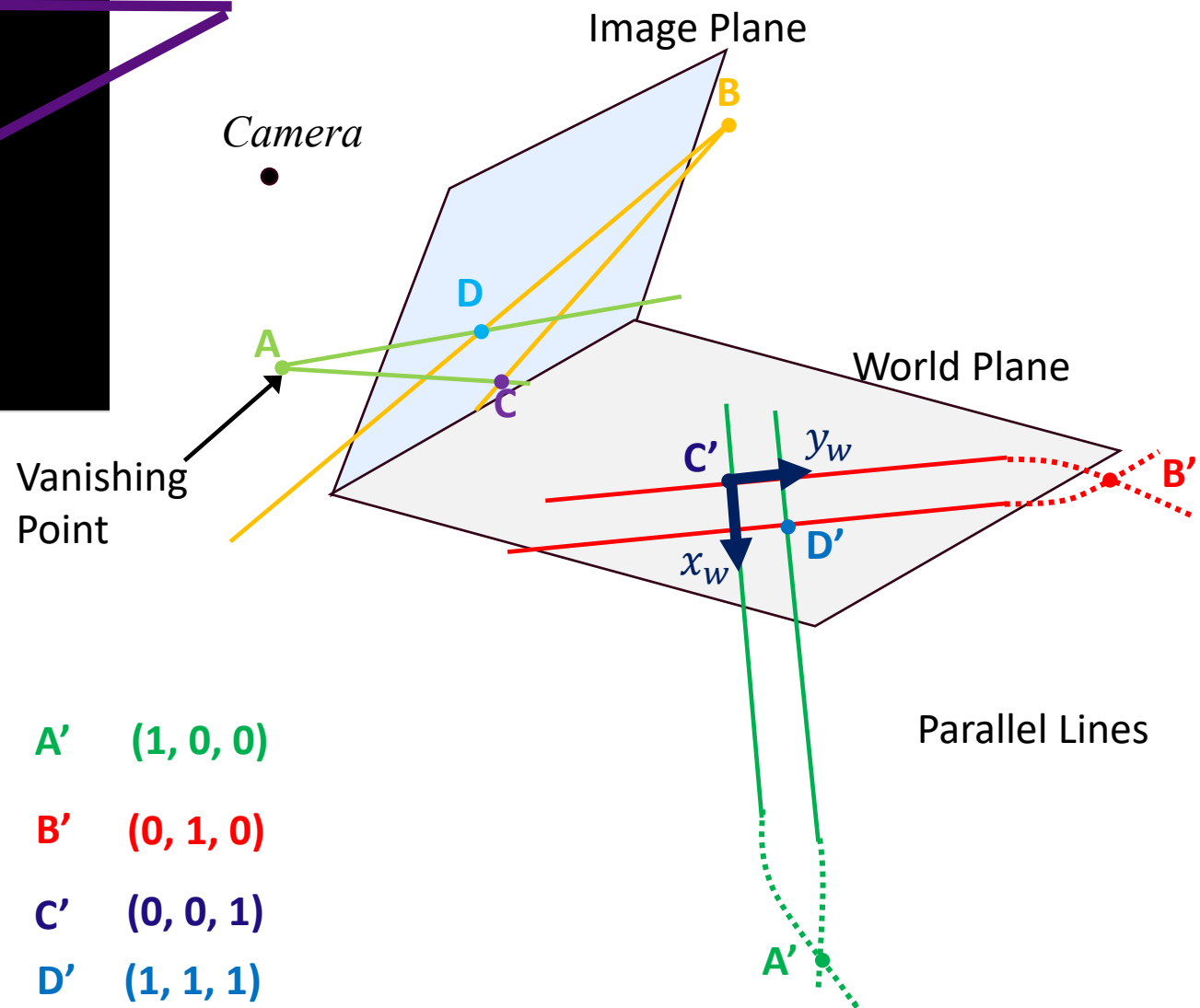
Note: makes sense, because after all, H has 8 degrees of freedom, and each 2D point correspondence pins down 2DOF.



Recap: Computing Homography from 4 Point Correspondences



Projection center



Correspondences:
($P^2 \rightarrow P^2$)

A	a	⟷	A'	(1, 0, 0)
B	b	⟷	B'	(0, 1, 0)
C	c	⟷	C'	(0, 0, 1)
D	d	⟷	D'	(1, 1, 1)

Recall: homography gives pose (given intrinsics K)

Pose from Projective Transformation

Recall the projection from world to camera

$$\begin{pmatrix} u \\ v \\ w \end{pmatrix} \sim K \begin{pmatrix} r_1 & r_2 & r_3 & T \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ W \end{pmatrix}$$

and assume that all points in the world lie in the ground plane $Z = 0$.

Then the transformation reads

$$\begin{pmatrix} u \\ v \\ w \end{pmatrix} \sim K \begin{pmatrix} r_1 & r_2 & T \end{pmatrix} \begin{pmatrix} X \\ Y \\ W \end{pmatrix}$$

And $r_3 = r_1 \times r_2$

Recap: But actually, not quite!

- According to the previous slide $K(r_1\ r_2\ T) = H$, or in other words,

$$K^{-1}H = (r_1, r_2, T) \text{ and } r_3 = r_1 \times r_2$$

If only life were so simple!

- Problem: when we ***estimate*** homographies (e.g. through solving linear systems with 2n equations from $n >= 4$ point correspondences), and then compute $K^{-1}H$, we aren't guaranteed to find a *valid* r_1 and r_2 pair. i.e. an orthonormal pair.
  - So, we need to find a way to first "correct" $(K^{-1}H)_{3\times3}$ to get orthonormal r_1 and r_2. Often called the "Procrustes", or "special orthogonal (SO) Procrustes" problem.
  - And we must solve this in real-time for robotics applications, so preferably an inexpensive approach.

Recap: Full Kabsch algorithm for finding pose via homography

1. Find H up to a scale factor from the point coorrespondences

2. Compute $H' = K^{-1}H$. Let H''s columns be $\begin{pmatrix} a & b & c \end{pmatrix}$

3. Minimize

$$\| \begin{pmatrix} a & b & c \end{pmatrix} - \lambda \begin{pmatrix} r_1 & r_2 & T) \end{pmatrix} \|_F$$

w.r.t. $\lambda \in \mathbb{R}, r_1, r_2, T \in \mathbb{R}^3$
s.t. $r_1^T r_2 = 0$ and $\|r_1\| = \|r_2\| = 1$

Let

$$(a\ b) = U_{3x2} \begin{pmatrix} s_1 & 0 \\ 0 & s_2 \end{pmatrix} V_{2x2}^T.$$

Then

$$\begin{pmatrix} r_1 & r_2 \end{pmatrix} = U_{3x2} V_{2x2}^T \qquad \text{and} \qquad \lambda = \frac{s_1 + s_2}{2}$$

4. $T = c/\lambda$ and $R = \begin{pmatrix} r_1 & r_2 & r_1 \times r_2 \end{pmatrix}$. Scale R to have determinant 1 if needed.

Pose from Point Correspondences, the Perspective N Point Problem (PnP)

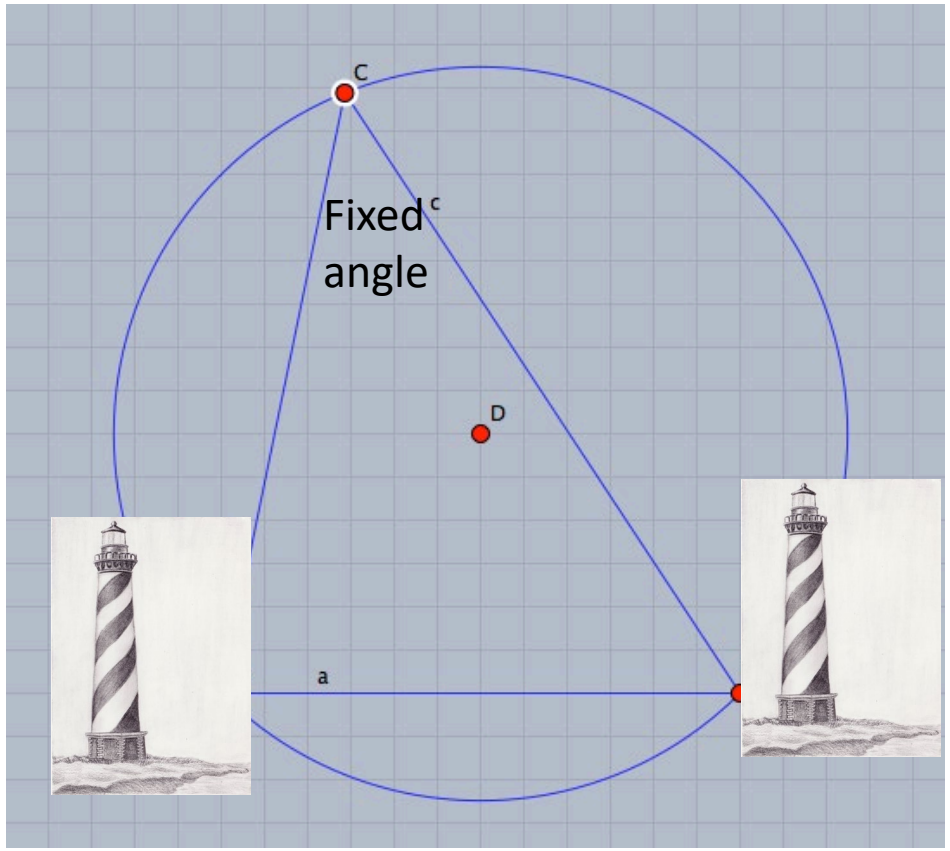
Localization by observing known 3D points from the world?

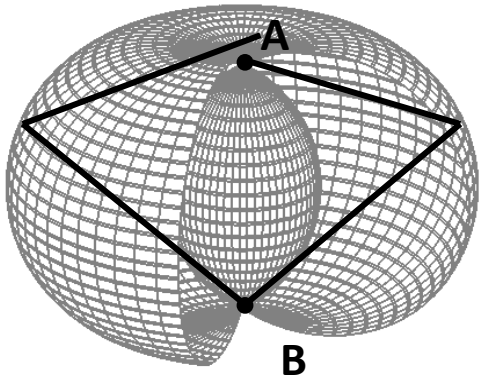
A real problem for autonomous cars, for example!
GPS: ~ a few feet accuracy. Just not good enough.

Instead, autonomous cars rely on 3D maps of the world to localize!

Navigation with “bearings” from 2 points

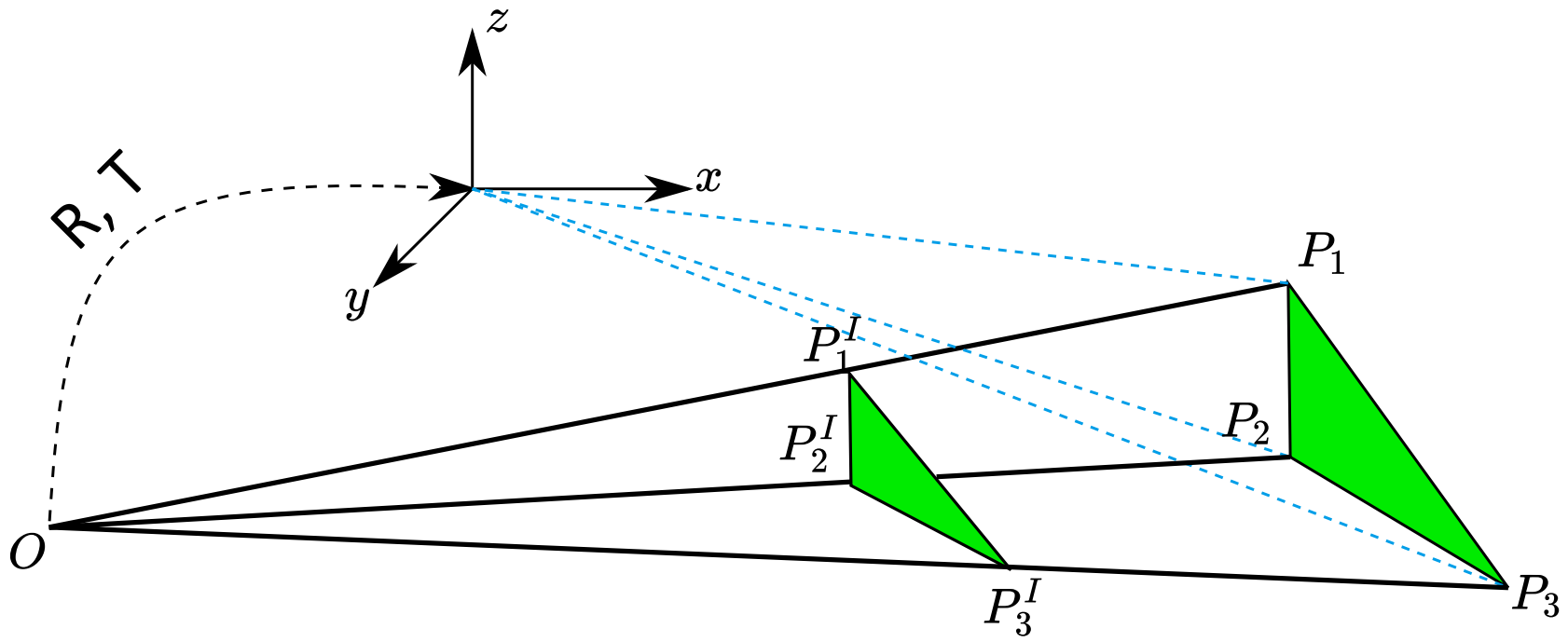
If I observe two lighthouses being some fixed angle θ apart, where am I?





Two points are not sufficient.
Camera could be anywhere on an upper semicircle in 2D.
Worse in 3D: anywhere on a toroid in 3D. Need more points!

The *Perspective* 3-Point Problem



- Given the point correspondences, find camera pose R, T

What are the differences from 4-Point Algorithm?

Jingnan Shi, https://jingnanshi.com/blog/pnp_minimal.html

P3P v.s. Homography

- Why P3P needs only three point correspondences, while computing Homography needs four point correspondences?

Simplified 3-Point Problem w. 3D Camera Coordinates

A triangle's world 3D coordinates $P_i \in \mathbb{R}^3$ are known, and its camera-centric 3D coordinates $P_i^c \in \mathbb{R}^3$ are known

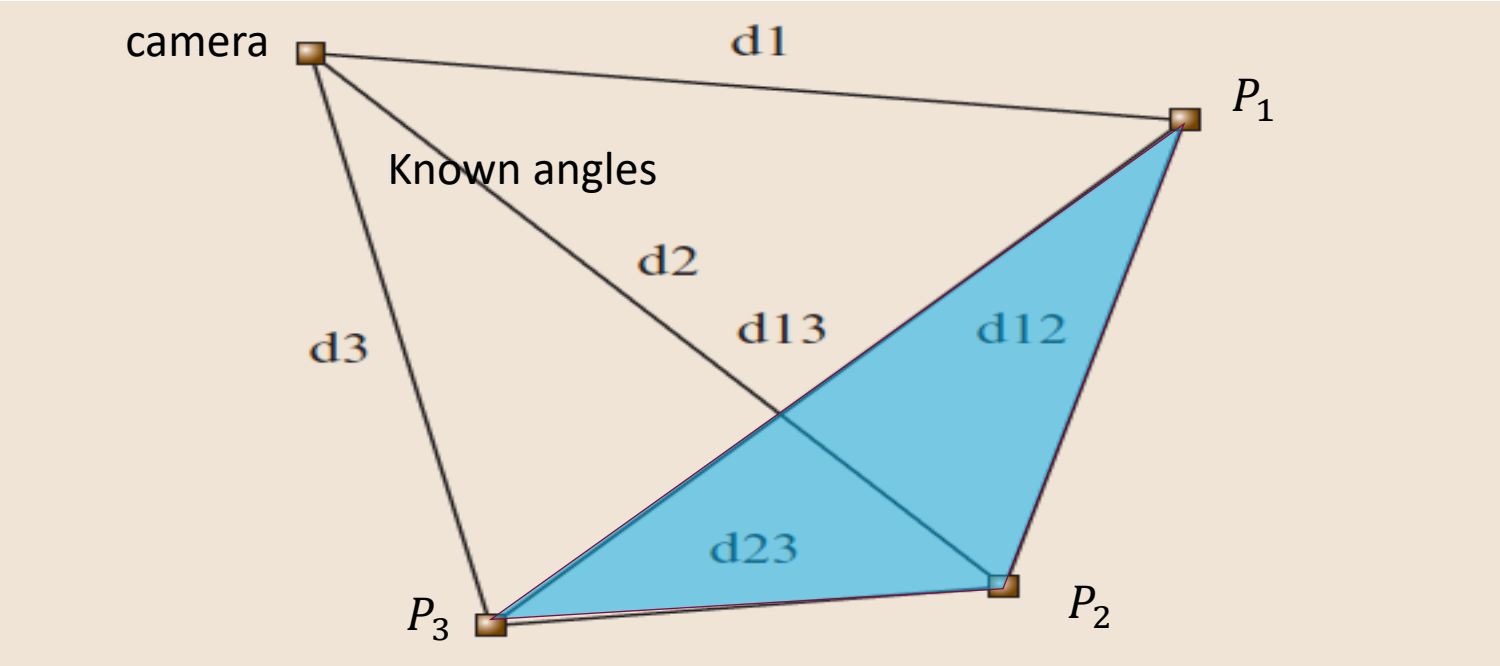
The 3D->3D 3-Point Problem: Find camera pose R, T such that

$$P_1^c = RP_1 + T$$
$$P_2^c = RP_2 + T$$
$$P_3^c = RP_3 + T$$

Full camera coordinates may come from depth cameras, but otherwise, we only have pixel coordinates.

Plan: starting from only pixel coordinates, first reduce the problem to 3D->3D.

P3P from Pixels



A triangle's world coordinates P_i are known, and its pixel coordinates are known

Pixels → "Calibrated coordinates"

RGB images only provide pixel coordinates u, v for each vertex, not camera-centric 3D coordinates. Convert these to:

"Calibrated coordinates": $\boldsymbol{p_i} \sim K^{-1}(u_i \quad v_i \quad 1)^T$

These are essentially image plane coordinates with principal point as origin, and focal length set to 1.

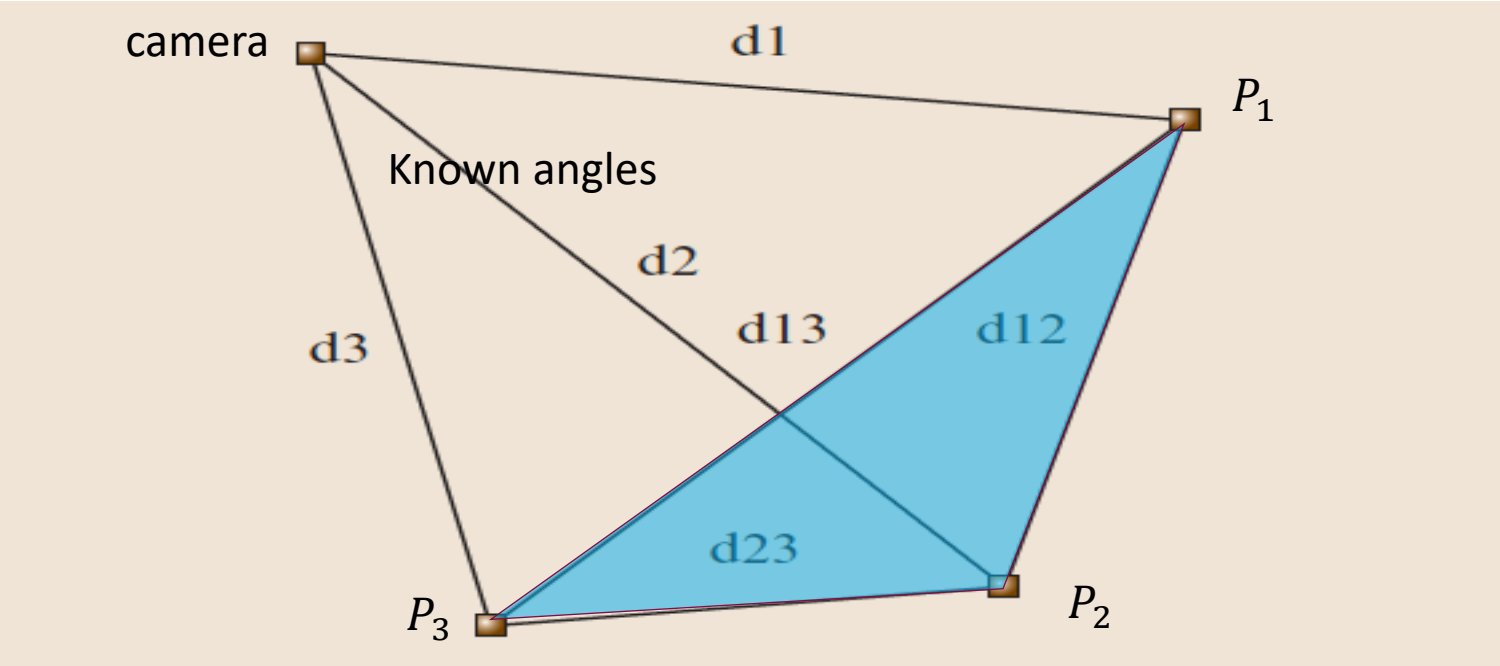
Euclidean interpretation: $\boldsymbol{p_i}$ is a vector in camera coordinates, originating from the camera center and pointing toward the 3D point corresponding to pixel coordinates (u_i, v_i).

Calibrated coordinates + distance = Camera-centric 3D

- After writing as calibrated coordinates, camera-centric 3D coordinates is just one step away: finding the distance of the 3D point along that direction, i.e. “depth”

- To reduce P3P to 3D->3D as we had planned, we will need to find those depths. Indeed, that is the first step of P3P.

P3P from ~~Pixels~~ Calibrated Coordinates



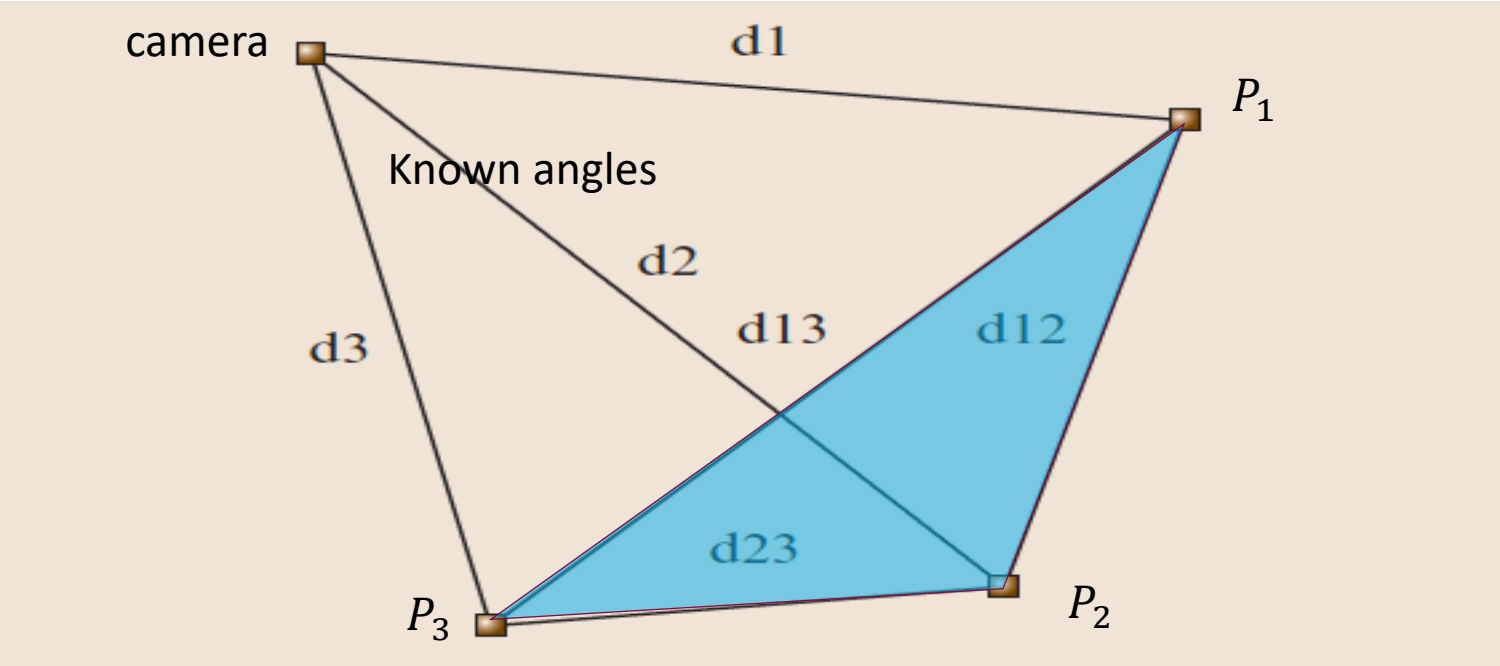
A triangle's world coordinates P_i are known, and its camera-centric calibrated coordinates p_i are known

The P3P problem: Find λ_i, R, T such that

$$\lambda_1 p_1 = RP_1 + T$$
$$\lambda_2 p_2 = RP_2 + T$$
$$\lambda_3 p_3 = RP_3 + T$$

Q: Are λ_i the same as "depths" d_i?
A: No, because p_i are not unit vectors.

P3P from Calibrated Coordinates



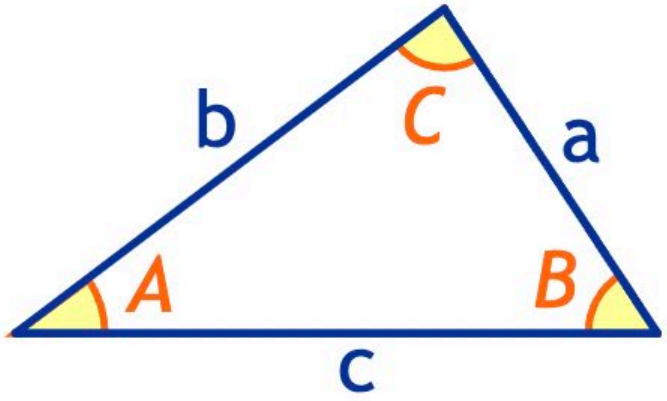
A triangle's world coordinates P_i are known, and its camera-centric calibrated coordinates p_i are known

The P3P problem: Find d_i, R, T such that

$$\frac{d_i}{||p_i||_2} p_i = RP_i + T, \qquad \forall i = 1,2,3$$

High School Flashback: Triangle Cosine Law

For any triangle:



a, **b** and **c** are sides.

C is the angle opposite side c

The Law of Cosines (also called the **Cosine Rule**) says:

$$c^2 = a^2 + b^2 - 2ab\cos(C)$$

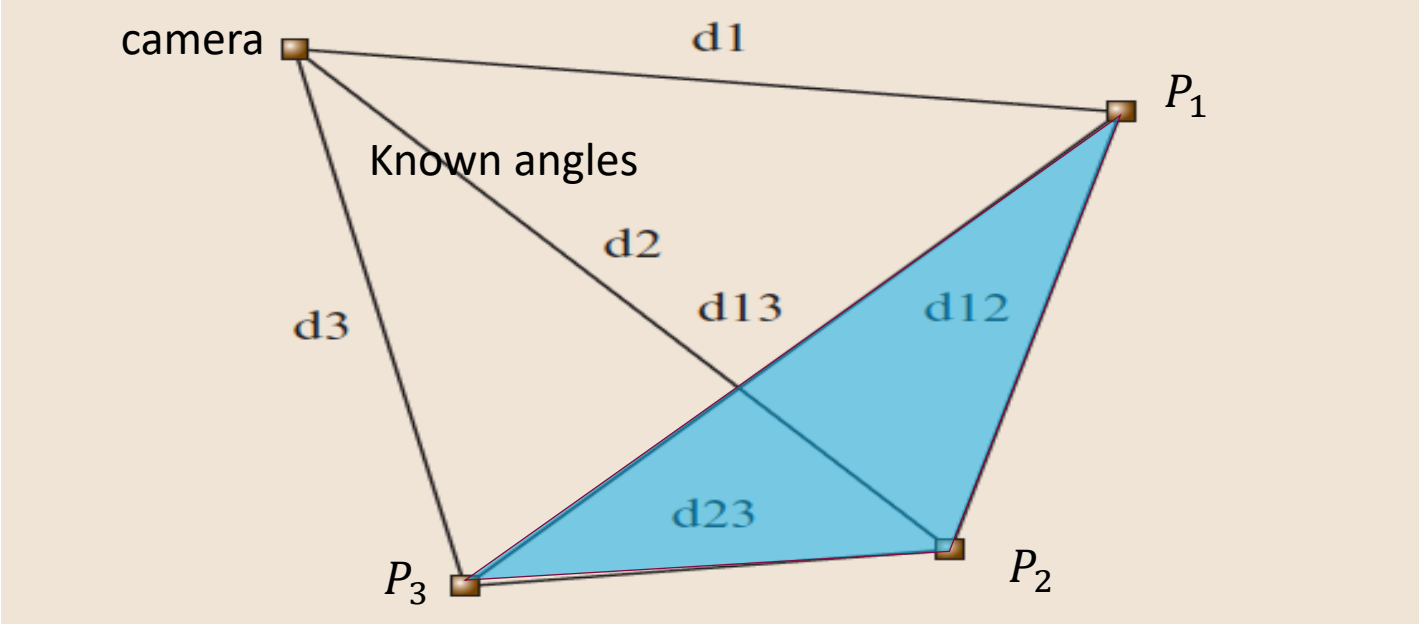
https://www.mathsisfun.com/algebra/trig-cosine-law.html

P3P Step 1: Finding depths d_i of triangle vertices

Let δ_{ij} denote the observed angle between the calibrated coordinates p_i and p_j

Then cosine law reads

$$d_i^2 + d_j^2 - 2d_i d_j \cos\delta_{ij} = d_{ij}^2$$



The cosine law

$$d_i^2 + d_j^2 - 2 d_i d_j \cos \delta_{ij} = d_{ij}^2$$

applies for each point pair. With 3 points we could solve 3 quadratic equations for $d_{i=1\ldots3}$.

Set $d_2 = u d_1$ and $d_3 = v d_1$ and solve all three equations for d_1:

$$\begin{aligned} d_1^2 &= \frac{d_{23}^2}{u^2 + v^2 - 2uv \cos \delta_{23}} \\ d_1^2 &= \frac{d_{13}^2}{1 + v^2 - 2v \cos \delta_{13}} \\ d_1^2 &= \frac{d_{12}^2}{u^2 + 1 - 2u \cos \delta_{12}} \end{aligned}$$

P3P Step 1: The algebraic drudgery

Reduces to two quadratic equations in u and v.

$$d_{13}^2(u^2+v^2-2uv\cos\delta_{23}) = d_{23}^2(1+v^2-2v\cos\delta_{13}) \quad (1)$$

$$d_{12}^2(1+v^2-2v\cos\delta_{13}) = d_{13}^2(u^2+1-2u\cos\delta_{12}) \quad (2)$$

a) Solve Eqn (1) for u^2 in terms of u, v, v^2 (and constants).
b) Plug this solution into Eqn (2), so that it has no u^2 term. Solve for u in terms of v, v^2, and constants.
c) Plug this solution for u back into Eqn (1), so that it has no more u or u^2. Instead, it is a 4th degree equation in v. *Get the 4 real solutions analytically.*
d) Then plug back into the solution found in step b) above, to get u.
e) Then get d_1 from the quadratic equations on the last page.
f) Then plug back into $d_2 = ud_1$ and $d_3 = vd_1$ from the last page to get d_2, d_3.

Reference for full version of P3P algebraic drudgery

- Grunert 1841, summarized in Haralick et al, 1993 which is linked in supplementary readings.

Review and Analysis of Solutions of the Three Point Perspective Pose Estimation Problem

ROBERT M. HARALICK
Intelligent Systems Laboratory, Department of Electrical Engineering FT-10, University of Washington, Seattle, WA 98195, USA

CHUNG-NAN LEE
Institute of Information Engineering, National Sun Yat-Sen University, Kaohsiung, Taiwan 80424, ROC

KARSTEN OTTENBERG
Philips-Forschungslabor, Vogt-Kölln Straße 30, D-2000 Hamburg 54, Germany

MICHAEL NÖLLE
Technische Universität Hamburg-Harburg, Technische Informatik I, Harburger Schloßstraße 20, D-2100 Hamburg 90, Germany

Received October 30, 1990; revised March 27, 1992 and October 29, 1993

End result of algebraic drudgery

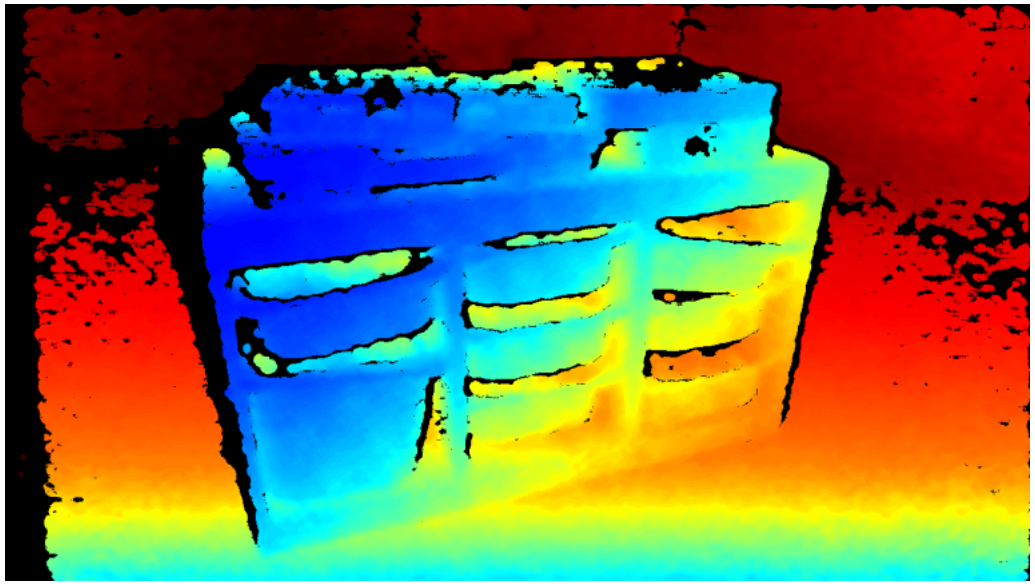
- We know the distances from camera to all three points! This means that we have effectively recovered “depth”, which combined with the calibrated coordinates we had before, means that we have:

Camera-centric 3D coordinates for triangle vertices!

So, we can now do what we planned all along, and just solve the 3D->3D 3-Point Problem.

Aside: Depth cameras

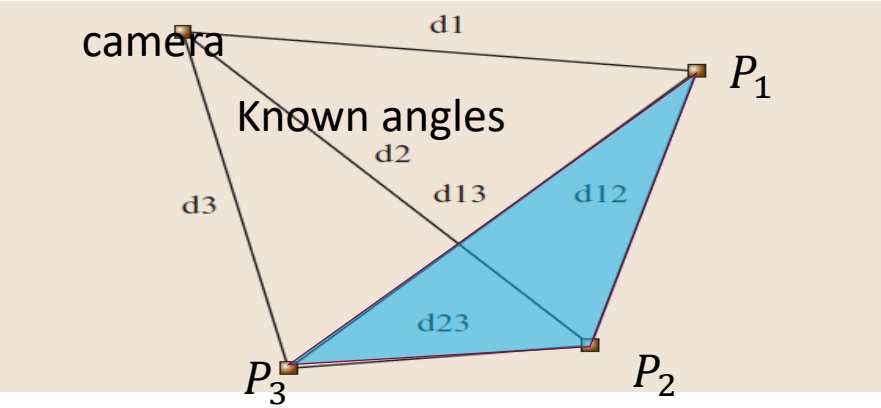
If we had started with depth images, we could have directly skipped to this point!



Note: Depth cameras commonly work from “stereo”: 2-view geometry!

Intel RealSense: https://www.intelrealsense.com/stereo-depth-vision-basics/

P3P Step 2: 3D->3D Pose/ 3D Registration. Find R&T!



The P3P problem has reduced to:

Find R, T such that

$$\frac{d_i}{||p_i||_2} p_i = RP_i + T, \qquad \forall i = 1,2,3$$

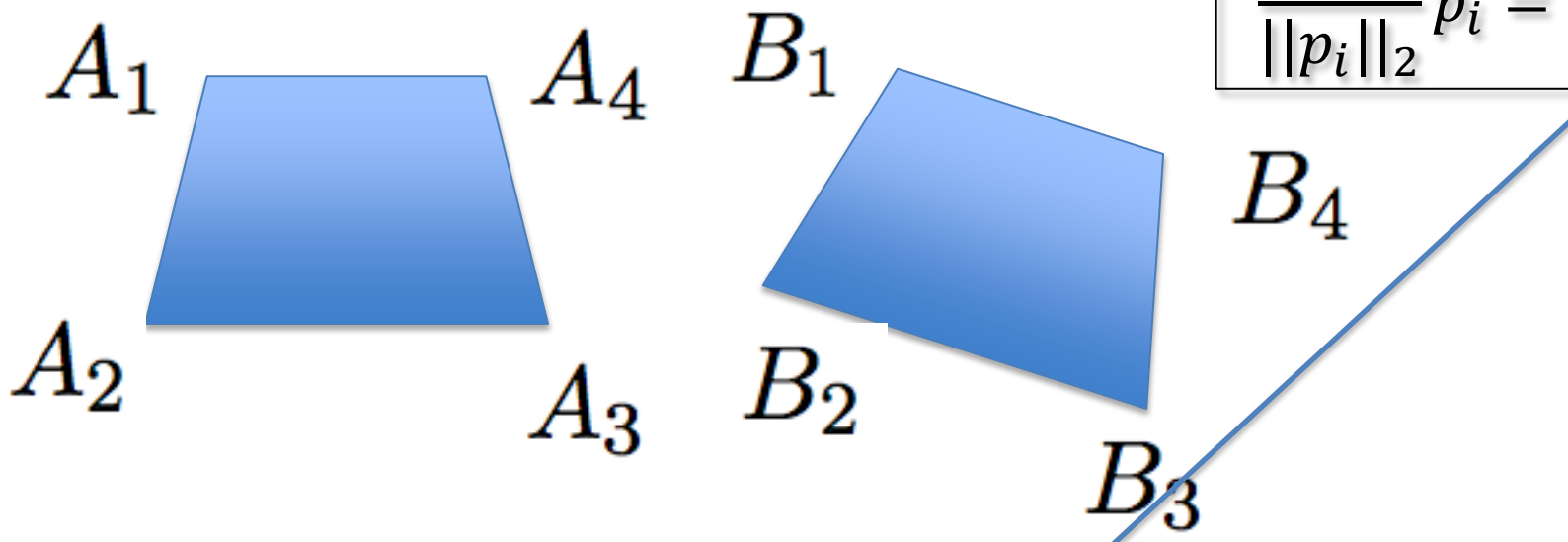
But naïve direct solution of the linear system is perilous, because rotation matrix R might not be valid.

(Does this remind you of something?)

Procrustes Problem and Its Kabsch Algorithm

The P3P problem: Find R, T such that

$$\frac{d_i}{||p_i||_2} p_i = RP_i + T, \qquad \forall i = 1,2,3$$



How do we solve for R, T from n point correspondences?

$$A_i = RB_i + T$$

(Note: the method we discuss for P3P step 2 is the same for >3 points too. So the rest of this discussion applies to more points too.)

(Nothing new on this slide except change of notation)

We solve a minimization problem for $N > 3$ point correspondences:

$$\min_{R,T} \sum_{i}^{N} \|A_i - RB_i - T\|^2$$

After differentiating with respect to T we observe that the translation is the difference between the centroids:

$$T = \frac{1}{N} \sum_{i}^{N} A_i - R \frac{1}{N} \sum_{i}^{N} B_i = \bar{A} - R\bar{B}$$

Notes:

1. If we find the right rotation matrix R, we will be able to find T afterwards.
2. If the centroids $\bar{A}$ and $\bar{B}$ had both been zero, then we could have set $T = 0$.

3. Rotation matrix between point sets is unaffected by translating the point sets, so can centroid-subtract the point sets before finding rotation.

4. And after this centroid-subtraction, we know $T = 0$ (by note#2 above), so:

$$\min_R \|A - RB\|_F^2$$

where

$$A = \begin{pmatrix} A_1 - \bar{A} & \dots & A_N - \bar{A} \end{pmatrix}$$

and

$$B = \begin{pmatrix} B_1 - \bar{B} & \dots & B_N - \bar{B} \end{pmatrix}$$

Reducing to Procrustes problem

This problem can be shown to be equivalent to the Procrustes problem we have seen last class for finding 2D homographies!

$$\underset{R \in SO(3)}{\operatorname{argmin}} ||A - RB||_F^2 = \underset{R \in SO(3)}{\operatorname{argmin}} ||R - AB^T||_F^2$$

Kabsch algorithm for Procrustes (proof in supp readings)

$$\underset{R \in SO(3)}{\arg\min} \|R - \begin{pmatrix} h_1' & h_2' & h_1' \times h_2' \end{pmatrix}\|_F^2$$

If the SVD of

$$\begin{pmatrix} h_1' & h_2' & h_1' \times h_2' \end{pmatrix} = USV^T$$

then the solution is

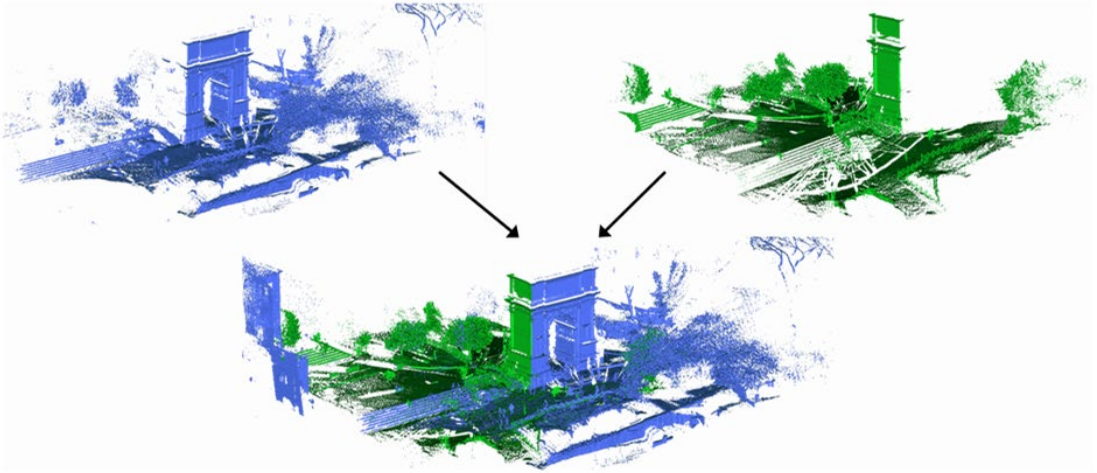
$$R = U \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \det(UV^T) \end{pmatrix} V^T$$

Kabsch Algorithm for 3D->3D

- Compute centroids $\bar{A}$ and $\bar{B}$ of the two sets of 3D points.
- Create matrices $A_{3\times n},\ B_{3\times n}$ after subtracting $\bar{A}$ and $\bar{B}$ from all points in the two sets.
- To find R, we must solve: $\underset{R\in SO(3)}{\operatorname{argmin}} ||A - RB||_F = \underset{R\in SO(3)}{\operatorname{argmin}} ||R - AB^T||_F^2$
  - First set $\hat{R} = (AB^T)_{3\times 3}$
  - Then decompose $\hat{R} = \mathrm{U}\Sigma V^T$
  - Set $R = U\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \det(VU^T) \end{pmatrix} V^T$
- Set $T = \bar{A} - R\bar{B}$

Sidenote: Extensions of 3D Point Set registration: Iterative Closest Point (ICP)

- Widely used for 3D point set registration ***without prior knowledge of correspondences***. Just needs 2 unordered point sets A and B. Jointly finds correspondences and transformation.



- Basic pseudocode:
  - **Step 1:** First, find "closest points" in set A for each point in set B. Treat these as pseudocorrespondences.
  - **Step 2:** Then solve for R, T treating these correspondences as given. E.g. using Kabsch, but several other alternatives.
  - **Step 3:** Transform set B using R, T, and go back to step 1.

Sidenote: Extensions of 3D Point Set registration: Iterative Closest Point (ICP)

- Widely used for 3D point set registration ***without prior knowledge of correspondences***. Just needs 2 unordered point sets A and B. Jointly finds correspondences and transformation.
- Basic pseudocode:
  - **Step 1:** First, find "closest points" in set A for each point in set B. Treat these as pseudocorrespondences.
  - **Step 2:** Then solve for R, T treating these correspondences as given. E.g. using Kabsch, but several other alternatives.
  - **Step 3:** Transform set B using R, T, and go back to step 1.
- **Disadvantages:**
  - Much slower because of having to iterate between correspondences and transformation.
  - Requires a good initialization of R and T.

PnP produces non-unique solutions $(n > 3)$?

- Recall that P3P Step 1 produced non-unique solutions for the distances d_i.
- But if we have $n > 3$ point correspondences, we only need Step 2. This is the **"Perspective-N-Point" problem, or simply PnP.**

Direct solution for PnP ($n > 3$): Steps

Again, first switch to calibrated coordinates:

Pose from N points in space given intrinsic parameters K and correspondences $(X_i, Y_i, Z_i, x_i, y_i)_{i=1...N}$
where

$$\begin{pmatrix} x_i \\ y_i \\ 1 \end{pmatrix} \sim K^{-1} \begin{pmatrix} u_i \\ v_i \\ 1 \end{pmatrix}$$

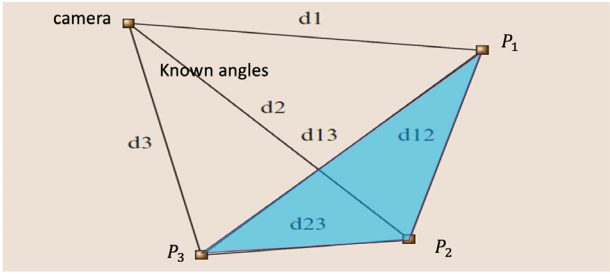
where u_i, v_i are pixel coordinates.

Direct solution for PnP: Steps

Given $(X_i, Y_i, Z_i, x_i, y_i)_{i=1\ldots N}$
find R, T such that

$$\lambda_i \begin{pmatrix} x_i \\ y_i \\ 1 \end{pmatrix} = R \begin{pmatrix} X_i \\ Y_i \\ Z_i \end{pmatrix} + T$$

P3P from Pixels



A triangle's world coordinates P_i are known, and its camera-centric calibrated coordinates p_i are known

The P3P problem: Find λ_i, R, T such that

$$\lambda_1 p_1 = RP_1 + T$$
$$\lambda_2 p_2 = RP_2 + T$$
$$\lambda_3 p_3 = RP_3 + T$$

Identical to the stage we reached with P3P, shown above.

But now, we are after a *direct* solution.

No need to solve explicitly for depths as we did in P3P, so need to find unit vectors etc.

Direct solution for PnP: Steps

Instead substitute λ and get 2 linear equations per point correspondence

$$\lambda_i \begin{pmatrix} x_i \\ y_i \\ 1 \end{pmatrix} = R \begin{pmatrix} X_i \\ Y_i \\ Z_i \end{pmatrix} + T \quad \Rightarrow \quad \begin{aligned} x_i &= \frac{r_{11}X_i + r_{12}Y_i + r_{13}Z_i + T_1}{r_{31}X_i + r_{32}Y_i + r_{33}Z_i + T_3} \\ y_i &= \frac{r_{21}X_i + r_{22}Y_i + r_{23}Z_i + T_2}{r_{31}X_i + r_{32}Y_i + r_{33}Z_i + T_3} \end{aligned}$$

Get linear equations by cross-multiplying ...

$$A_{2n\times 12}\boldsymbol{x}_{12} = 0,$$

where $\boldsymbol{x}$ is a vector of all the unknowns including R and T.

Need at least 11 equations = at least 6 point correspondences to solve.
(just like for homography $A_{2n\times 9}\boldsymbol{h}_{9\times 1} = 0$, we needed at least 4 point correspondences = 8 equations.)

Q: Why do we need more points for the direct method, when P3P was able to work with 3 points?

We meet Procrustes and Kabsch yet again

Again, just solving the linear system won't give good rotations. So full steps:

1. Solve $A_{2n\times 12}\boldsymbol{x}_{12} = 0$ from $n \geq 6$ correspondences.
2. Then, assemble the rotation matrix $\hat{R}$ from the solution for $\boldsymbol{x}$.
3. Then, find the closest valid rotation matrix by Kabsch!

Decompose $\hat{R} = \mathrm{U}\Sigma V^T$

Then, set $R = U\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \det(VU^T) \end{pmatrix} V^T$

4. May now need to adjust translation T to be consistent with new R, e.g., by solving $A_{2n\times 12}\boldsymbol{x}_{12} = 0$, for only t_1, t_2, t_3 (elements of $\boldsymbol{x}$), holding R fixed.