



Introduction to Embodied AI

Lecture 5 - Vision and Grasping I

Prof. He Wang



Logistics

Logistics

- Assignment 1
 - Due on 03/22 at 11:59PM (Beijing time, this Saturday night).

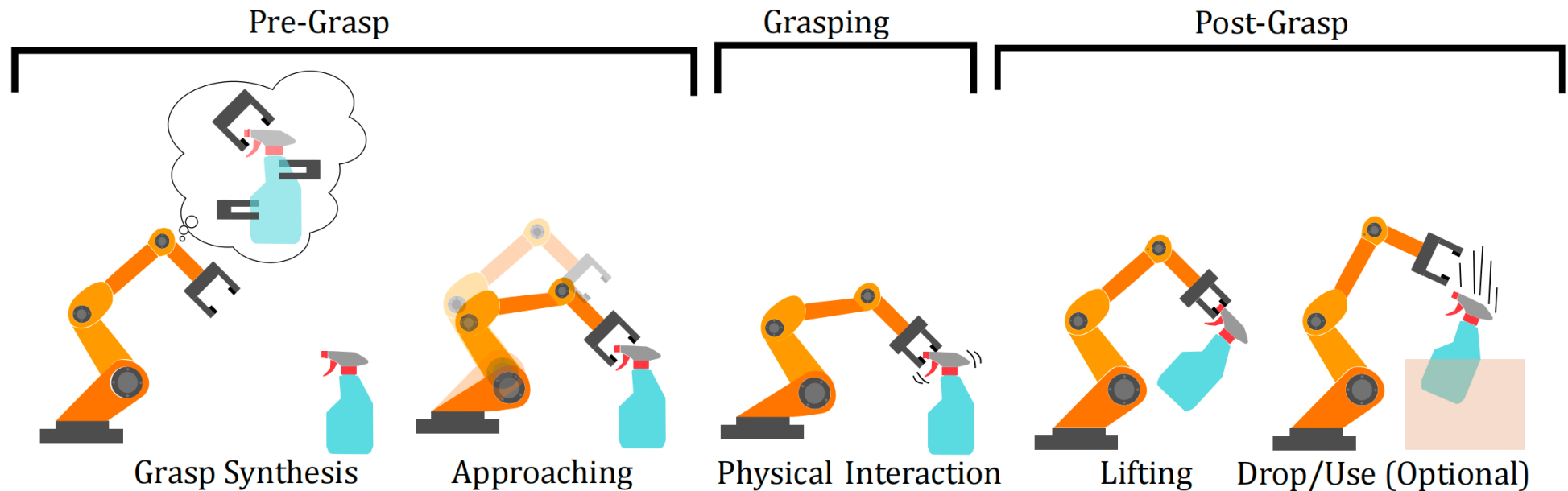
Grasping

From the Basics of Robotics to Trajectory Generation

- Kinematics: conversion between Cartesian space and joint space
- Motion Planning: get a geometrically valid path
- Control: ensures the motion follow the planned trajectory
- If we can provide where to move, a.k.a., end-effector pose in Cartesian space, this already forms a pipeline.

What is Grasping?

- **Grasping** is the process of restraining an object's motion in a desired way by applying forces and torques at a set of contacts
- **Grasp Synthesis** is a high-dimensional search or optimization problem to **find gripper poses** or joint configurations

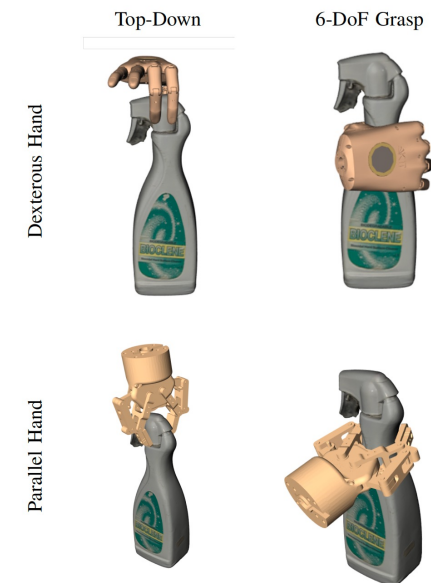


Terminology

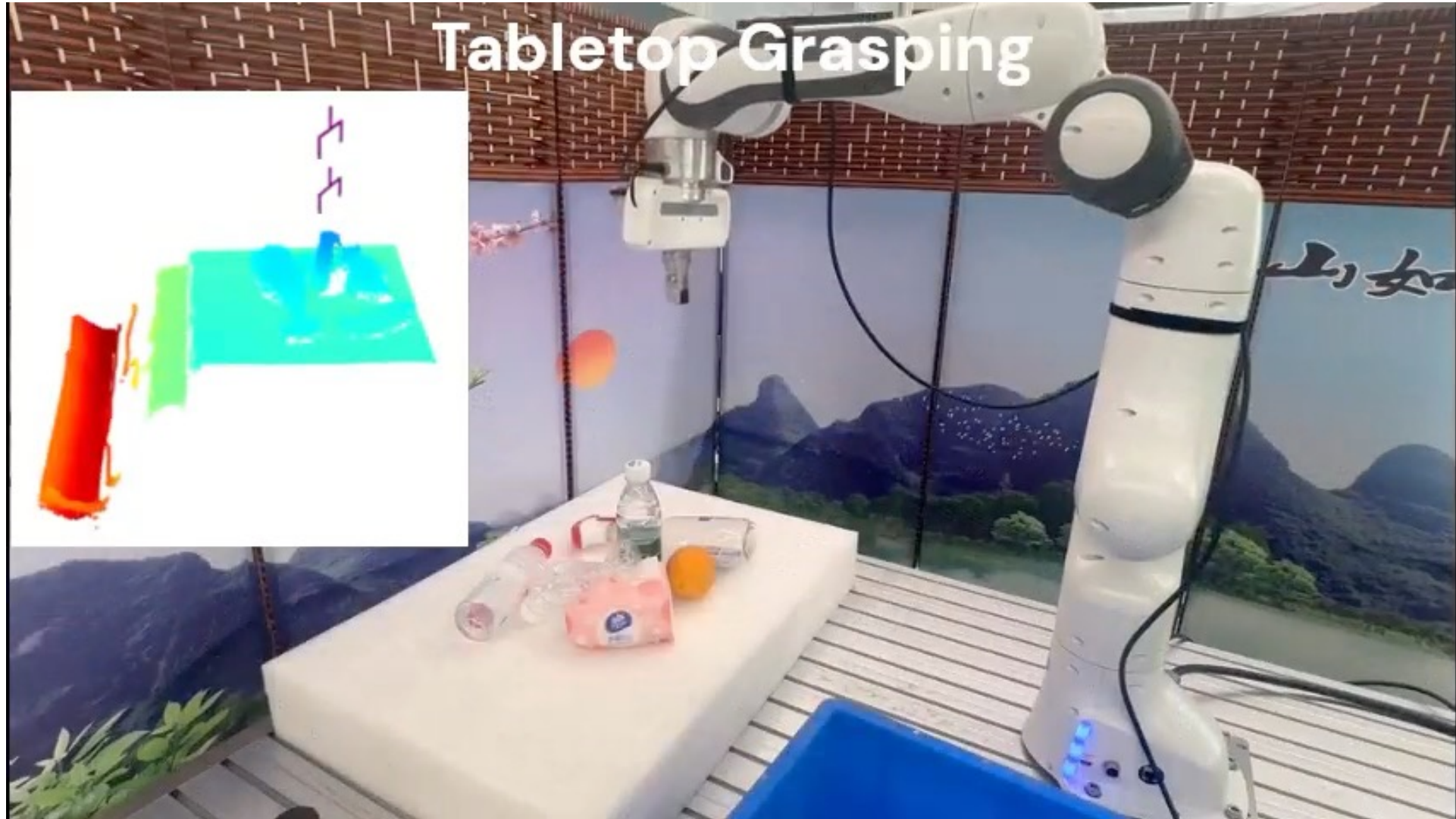
- **Grasp Pose** defines the position, orientation and articulation of a hand
 - position, orientation
 - **4-DoF grasp**: a 3D position and 1D hand orientation aligned with the direction of gravity, a.k.a. “top-down grasping”
 - **6-DoF grasp**: defined by a 3D position and 3D orientation

+

- Articulation (finger DoFs)
 - Parallel gripper: 1
 - Dex Hand: up to 21



Open-Loop Grasping



Closed-Loop Grasping



GraspVLA. In submission.

Closed-Loop Grasping

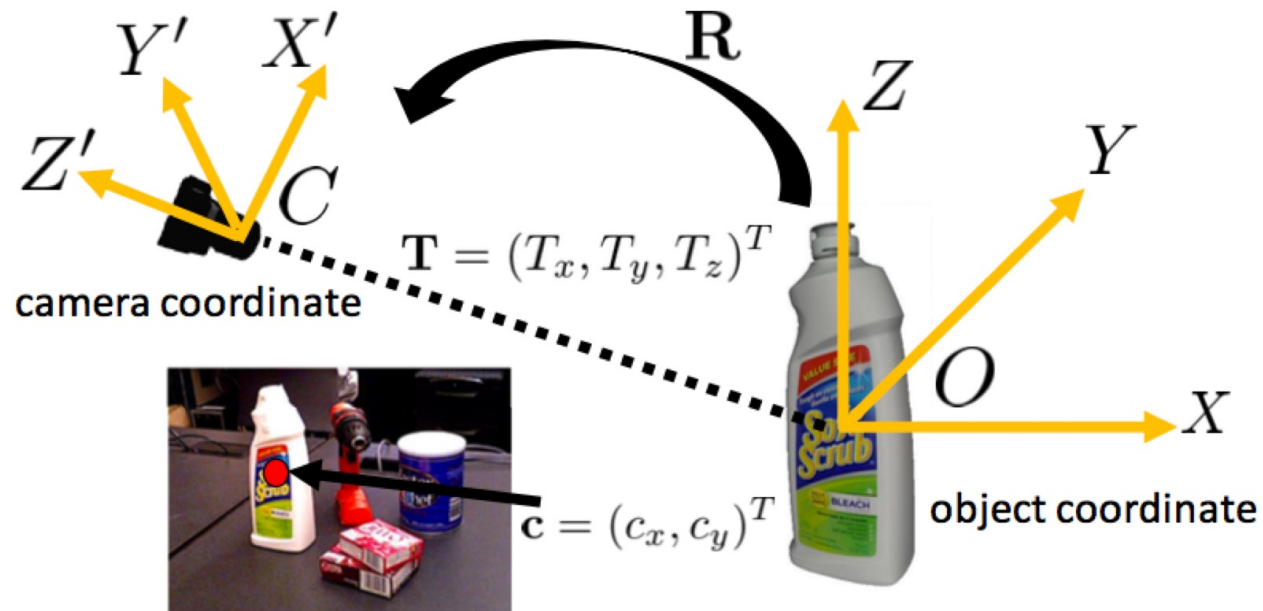


Two Paths for Open-Loop Grasping

- For known objects,
 - pose estimation would be useful
- For unknown and general objects
 - directly predict grasping pose

6D Pose Estimation

6D Object Pose



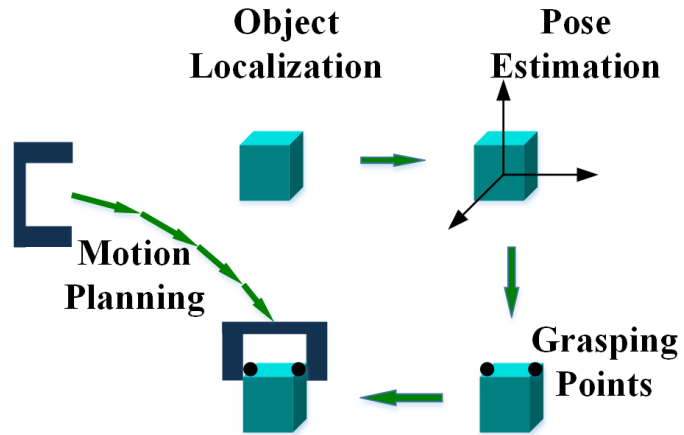
Definition:

6D transformation from object to camera space

- 3DoF translation
- 3DoF rotation

$$\begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} = R \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + T$$

Applications: 6D Pose for Robotic Manipulation

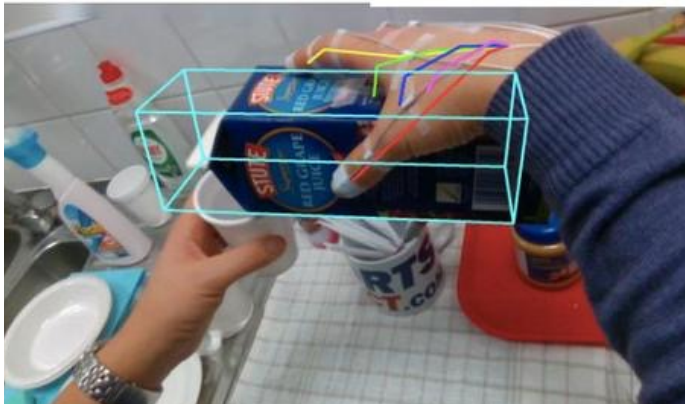
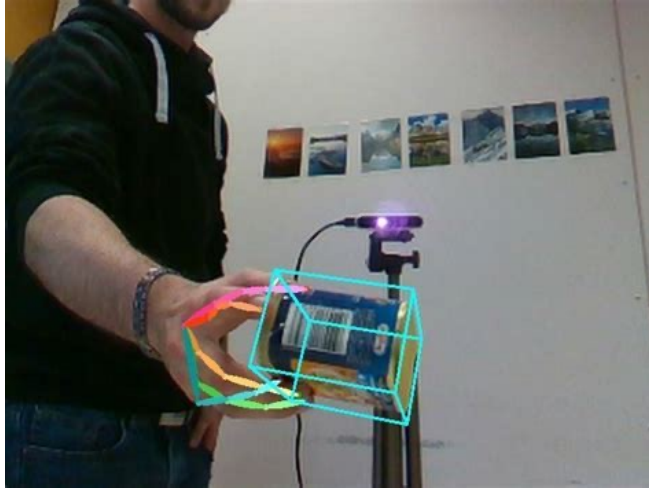


- Object pose helps to transfer annotation from CAD model to the object in the camera space
- Pose tracking serves as visual closed-loop control

Object Pose based Grasping



Applications: 6D Pose for Human Object Interaction



- **Vision** perspective: to perceive and understand human object interaction
- **Robotic** perspective: to extract pose trajectories for imitation learning

Hampali, S. , et al. "HONnotate: A method for 3D Annotation of Hand and Objects Poses." *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* , 2020.

Doosti, B. , et al. "HOPE-Net: A Graph-based Model for Hand-Object Pose Estimation." *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* , 2020.

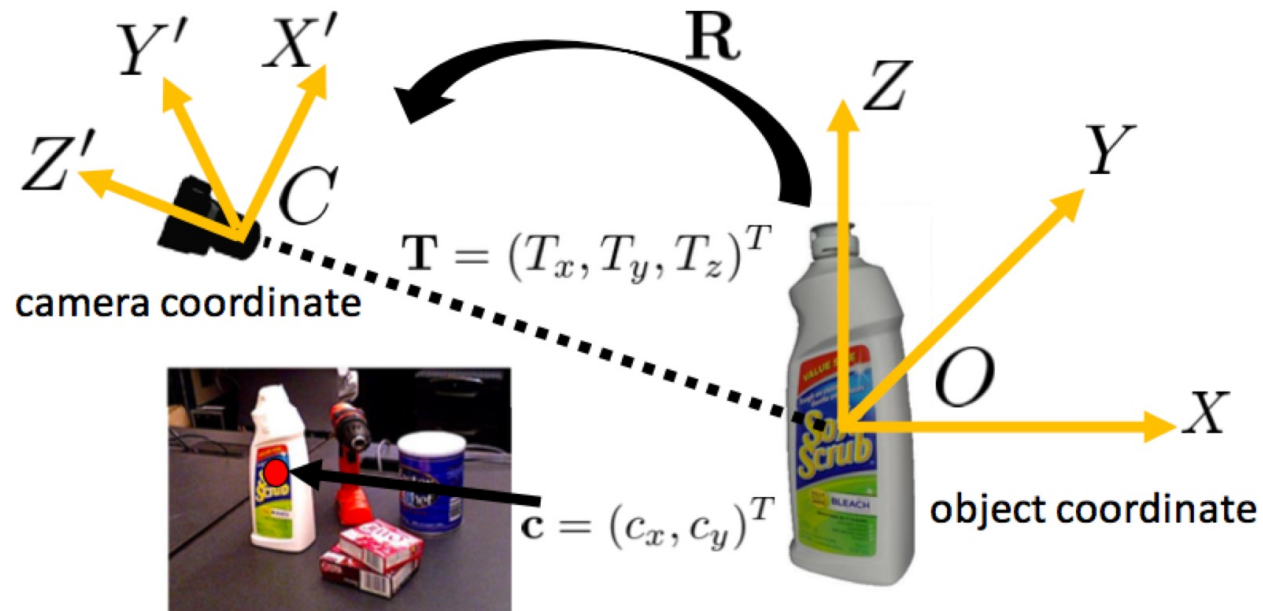
Applications: Augmented Reality



- **AR:** immersive interaction for **metaverse**
- visual guidance and E-learning
- mobile apps and entertainment

Marchand, E. , H. Uchiyama , and F. Spindler. "Pose Estimation for Augmented Reality: A Hands-On Survey." *IEEE Transactions on Visualization & Computer Graphics* 22.12(2016):2633-2651.

6D Object Pose



Definition:

6D transformation from object to camera space

- 3DoF translation
- **3DoF rotation**

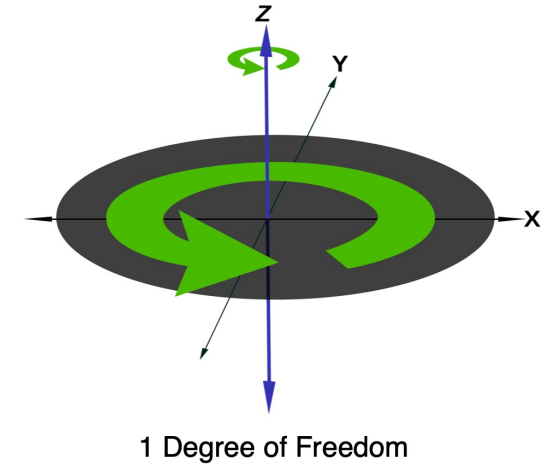
How to predict a rotation?

Rotation Regression

Rotation

- 3D Rotation matrix R is a 3×3 matrix that satisfies $R^T R = I$ and $\det R = +1$.
- All rotation matrices form a group $SO(3)$. Here SO stands for Special Orthogonal.

2D rotation



3D rotation



Rotation Representations

- 3D rotation only has 3 degree of freedom, however rotation matrix contains 9 elements, which makes neural network harder to predict. Any other way to represent a rotation?
- Euler Angle: 3D
- Axis Angle: 3D
- Quaternion: 4D

Euler Angle

- Rotation about principal axis is represented as:

$$R_x(\alpha) := \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \alpha & -\sin \alpha \\ 0 & \sin \alpha & \cos \alpha \end{bmatrix}$$

$$R_y(\beta) := \begin{bmatrix} \cos \beta & 0 & \sin \beta \\ 0 & 1 & 0 \\ -\sin \beta & 0 & \cos \beta \end{bmatrix}$$

$$R_z(\gamma) := \begin{bmatrix} \cos \gamma & -\sin \gamma & 0 \\ \sin \gamma & \cos \gamma & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$R = R_x(\alpha)R_y(\beta)R_z(\gamma)$$

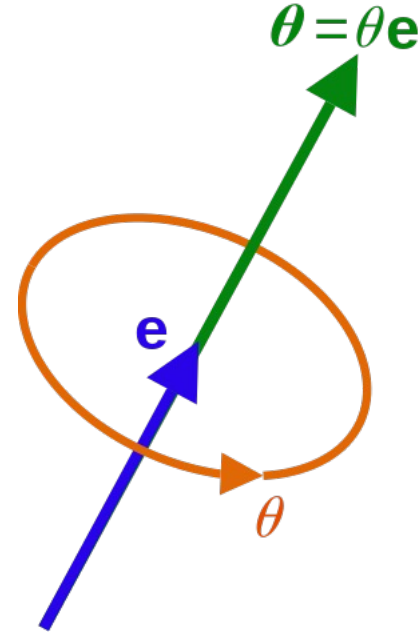
Slides credit: Hao Su



For the conventions of Euler angles, refer to
https://en.wikipedia.org/wiki/Euler_angles

Axis Angle

$$(\text{axis, angle}) = \left(\begin{bmatrix} e_x \\ e_y \\ e_z \end{bmatrix}, \theta \right)$$



- Use Rodrigues' rotation formula to transform axis–angle into rotation matrix:

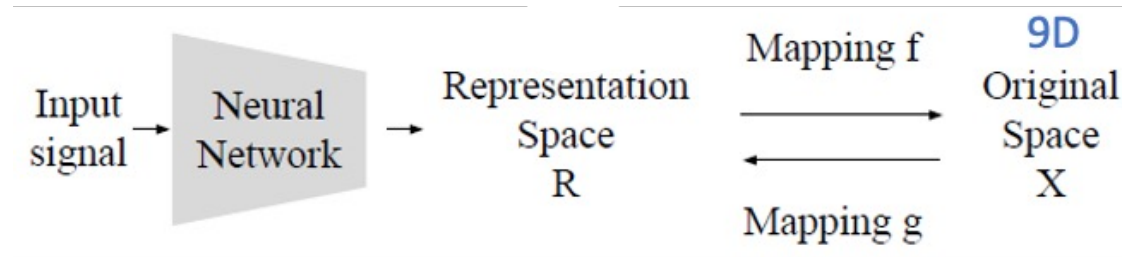
$$R = I + (\sin \theta) \mathbf{K} + (1 - \cos \theta) \mathbf{K}^2, \quad K = [e]_{\times} = \begin{bmatrix} 0 & -e_z e_y \\ e_z & 0 & -e_x \\ -e_y & e_x & 0 \end{bmatrix}$$

Quaternion

- Recall the complex number $a + b\mathbf{i}$
- Quaternion is a more generalized complex number:
 $q = w + x\mathbf{i} + y\mathbf{j} + z\mathbf{k}$
 - w is the real part and $\vec{v} = (x, y, z)$ is the imaginary part
 - Imaginary: $\mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = \mathbf{ijk} = -1$
 - anti-commutative :
 $\mathbf{ij} = \mathbf{k} = -\mathbf{ji}, \mathbf{jk} = \mathbf{i} = -\mathbf{kj}, \mathbf{ki} = \mathbf{j} = -\mathbf{ik}$

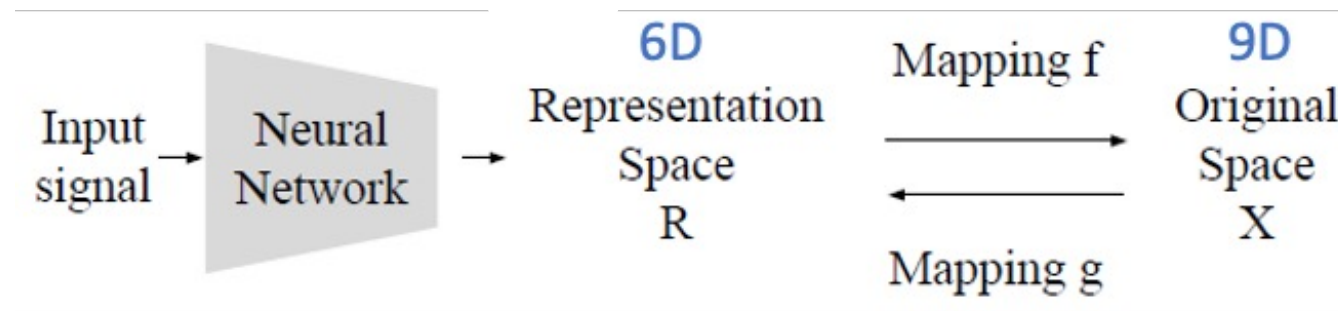
How to Estimate Rotation using Neural Network

- Method 1: Use a neural network to regress a rotation representation



How to Estimate Rotation using Neural Networks

- Method 1: Use a neural network to regress a rotation representation



Quaternion was very popular to regress.
Still an active research area.

Zhou, Yi, et al. "On the continuity of rotation representations in neural networks." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2019.

Levinson, Jake, et al. "An analysis of svd for deep rotation estimation." Advances in Neural Information Processing Systems 33 (2020): 22554-22565.

Peretroukhin, Valentin, et al. "A Smooth Representation of Belief over SO (3) for Deep Rotation Learning with Uncertainty."

Chen, Jiayi, et al. "Projective Manifold Gradient Layer for Deep Rotation Regression." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2022.

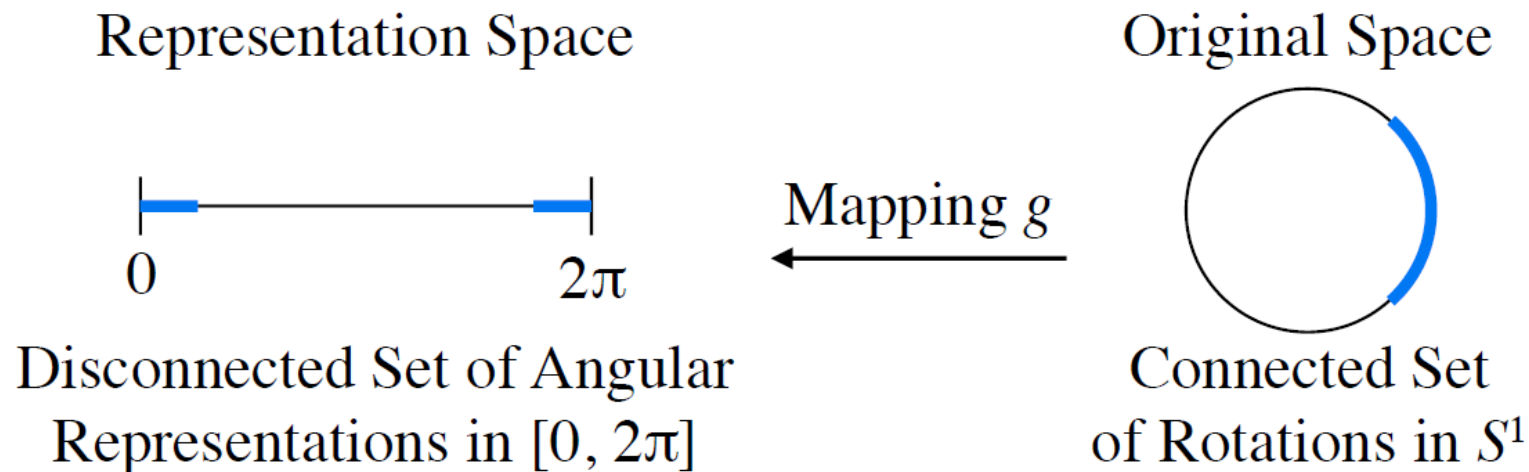
Singularities and Discontinuity

Singularities: In mathematics, a singularity is a point at which a given mathematical object is not defined or not well-behaved (ambiguity, non-differentiation, etc.)

Several rotation representations suffer from singularities and discontinuities.

Euler Angles – Discontinuity

- Representing $SO(3)$ with angular representation is discontinuous.
- Take one direction of rotation as an example, the limit of identity rotation is undefined: 0 or 2π



Axis-angle

- If restrict $\theta \in (0, \pi)$, a unique parameterization exists:

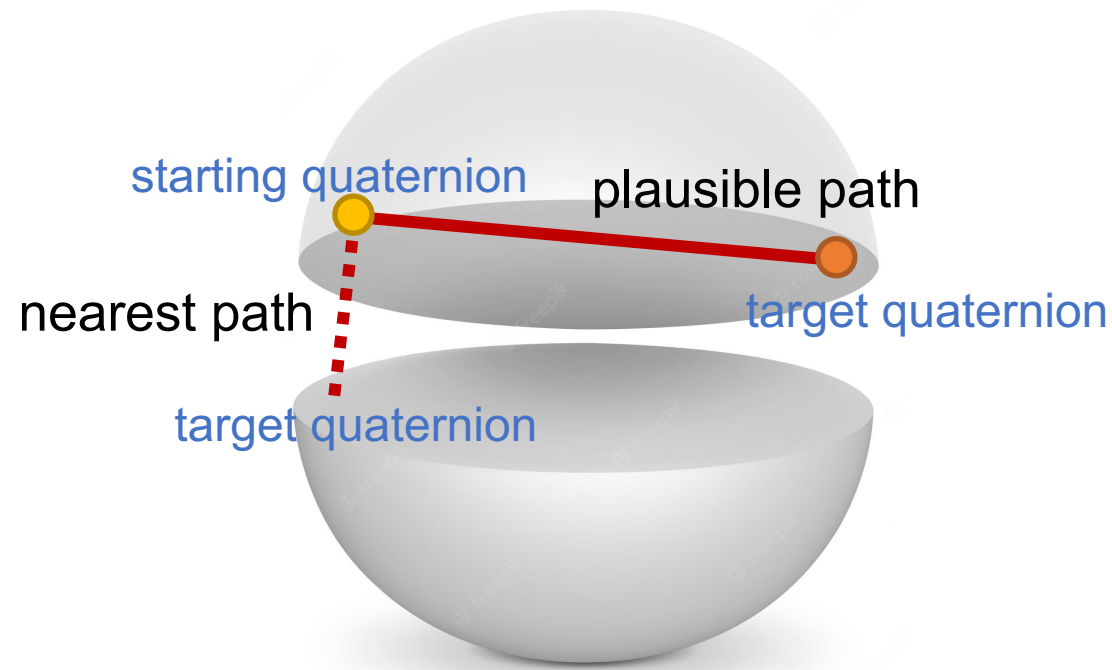
- $$\theta = \arccos \frac{1}{2} [\text{tr}(R) - 1], \quad [\hat{\omega}] = \frac{1}{2\sin\theta} (R - R^T)$$

- At $\theta = 0$, each axis represents identity rotation
- At $\theta = \pi$, each axis and its negation represent the same rotation, which become discontinuous

$$R = I + (\sin \theta) \mathbf{K} + (1 - \cos \theta) \mathbf{K}^2, \quad K = [e]_{\times} = \begin{bmatrix} 0 & -e_z e_y \\ e_z & 0 & -e_x \\ -e_y & e_x & 0 \end{bmatrix}$$

Quaternions

- Double coverage: q and $-q$ represent the same rotation R
- When using quaternion for predicting R , we assume a one-to-one mapping, making it discontinuous
- Assume we constrain q to be at the upper semi-sphere



Continuous Rotation Representations: 6D Representation

- Rotation matrix is a continuous representation in $SO(3)$, but has excessive number of parameters
- 6D representation simply eliminates the last column of rotation matrix
- 6D representation applies Gram-Schmidt orthogonalization to map the Euclidean output of neural networks to $SO(3)$

6D Representation

- Given a rotation matrix A as

$$\begin{bmatrix} | & | & | \\ a_1 & a_2 & a_3 \\ | & | & | \end{bmatrix}$$

6D representation is given by the first two columns of A , i.e., 6D representation is:

$$\begin{bmatrix} | & | \\ a_1 & a_2 \\ | & | \end{bmatrix}$$

6D Representation

- The mapping from 6D representation to rotation matrix is done by Gram-Schmidt process
 - Column 1 is normalized
 - Column 2 only preserves the perpendicular component to column 1 and is then normalized
 - Column 3 is determined by cross product of column 1&2

$$f_{GS} \left(\begin{bmatrix} | & | \\ a_1 & a_2 \\ | & | \end{bmatrix} \right) = \begin{bmatrix} | & | & | \\ b_1 & b_2 & b_3 \\ | & | & | \end{bmatrix} \quad b_i = \begin{bmatrix} N(a_1) & \text{if } i = 1 \\ N(a_2 - (b_1 \cdot a_2)b_1) & \text{if } i = 2 \\ b_1 \times b_2 & \text{if } i = 3 \end{bmatrix}^T$$

Continuous Rotation Representations: 9D Representation

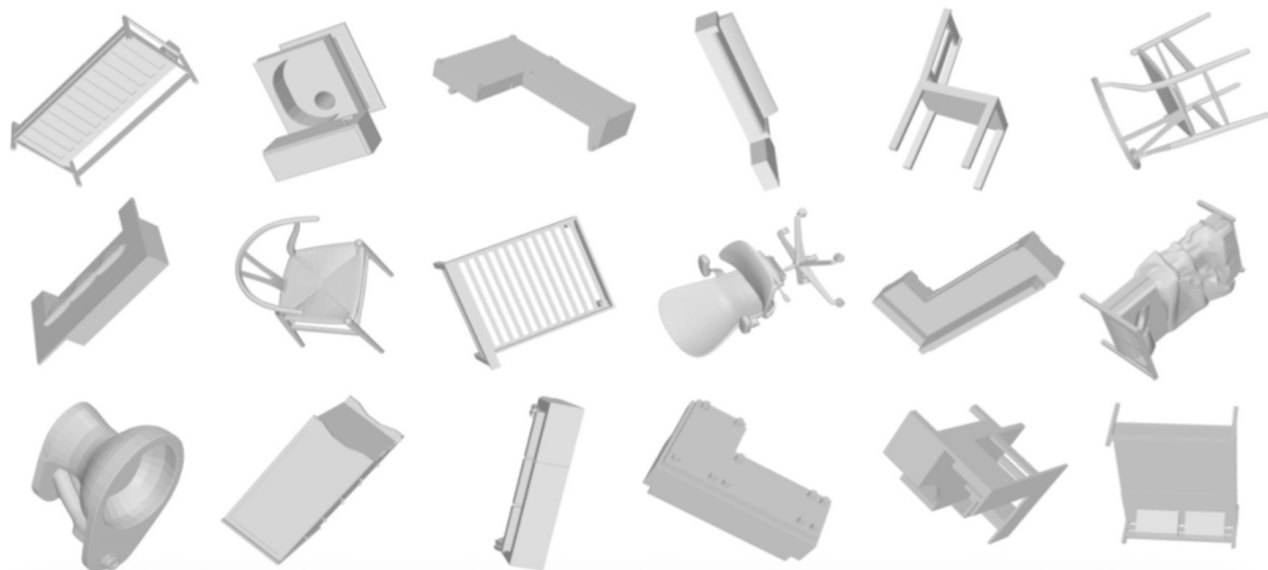
- 9D representation simply correspond to rotation matrix
- 9D representation applies singular value decomposition (SVD) orthogonalization to map the Euclidean output of neural networks to $SO(3)$

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

Comparisons

Task: Single image rotation regression

Dataset: ModelNet10-SO3



3D rotations (*ModelNet10-SO3*)

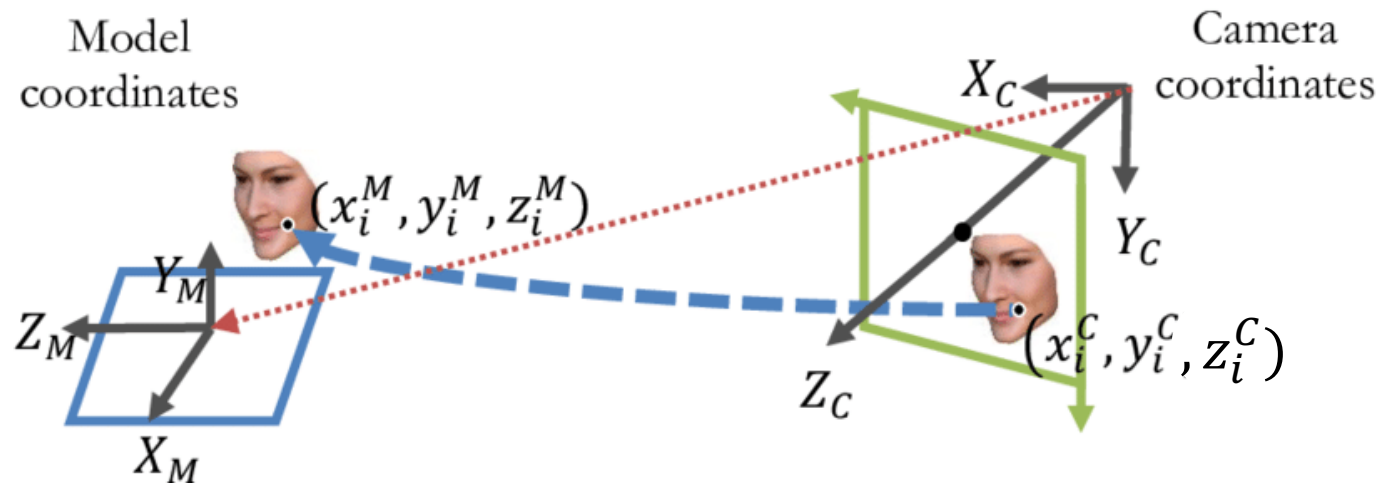
	Mean (°)	Med	Std
3D-RCNN	35.50	13.21	46.55
M_G	31.60	16.70	41.86
Euler	41.35	27.44	37.73
Axis-Angle	32.30	19.74	34.70
Quaternion	26.92	14.39	32.92
S^2 -Reg	27.36	15.41	33.17
5D	25.18	13.40	32.10
6D	22.60	11.51	31.24
SVD-Train	21.25	11.14	30.28

	Mean (°)	Med	Std
3D-RCNN	34.80	7.32	55.73
M_G	31.41	13.93	48.48
Euler	49.31	32.03	43.47
Axis-Angle	31.82	17.31	37.19
Quaternion	29.60	14.56	37.00
S^2 -Reg	25.99	12.11	37.67
5D	26.23	11.52	38.91
6D	20.25	7.84	36.85
SVD-Train	18.01	7.31	33.96

Rotation Fitting

How to Estimate Rotation using Neural Networks

- Method 2: Predict object coordinate or correspondence and then solve rotation:
 - Given an image, for each pixel on the object, predicts the 3D coordinate of this pixel on the object CAD model (object coordinates)
 - This builds 2D–3D dense correspondence, if you have depth, then you have 3D–3D dense correspondence.



Orthogonal Procrustes Problem

- Given 3D–3D dense correspondence, use Orthogonal Procrustes to solve/fit R.

The orthogonal Procrustes problem is a matrix approximation problem that can be stated as follows: for $M \in \mathbb{R}^{n \times 3}$ and $N \in \mathbb{R}^{n \times 3}$, solve

$$\hat{A} = \operatorname{argmin}_{A \in \mathbb{R}^{3 \times 3}} \|M^T - AN^T\|_F^2 \text{ subject to } A^T A = I$$

where $\|\cdot\|_F$ represents the Frobenius norm.

$$\|X\|_F = \sqrt{\operatorname{trace}(X^T X)} = \sqrt{\sum_{i,j} x_{ij}^2}$$

Analytical Solution to Orthogonal Procrustes

- The solution can be expressed in terms of the SVD of a special matrix
- If we have the SVD: $\mathbf{M}^T \mathbf{N} = \mathbf{U} \mathbf{D} \mathbf{V}^T$, then

$$\hat{A} = \mathbf{U} \mathbf{V}^T$$

Proof

$$\begin{aligned}\|M^T - AN^T\|_F^2 &= \|\mathbf{M} - \mathbf{N}\mathbf{A}^T\|_F^2 = \text{tr}(\mathbf{M}^T\mathbf{M}) - 2\text{tr}(\mathbf{M}^T\mathbf{N}\mathbf{A}^T) + \text{tr}(\mathbf{A}\mathbf{N}^T\mathbf{N}\mathbf{A}^T) \\ &= \text{tr}(\mathbf{M}^T\mathbf{M}) - 2\text{tr}(\mathbf{M}^T\mathbf{N}\mathbf{A}^T) + \text{tr}(\mathbf{A}^T\mathbf{A}\mathbf{N}^T\mathbf{N}) \\ &= \text{tr}(\mathbf{M}^T\mathbf{M}) - 2\text{tr}(\mathbf{M}^T\mathbf{N}\mathbf{A}^T) + \text{tr}(\mathbf{N}^T\mathbf{N}).\end{aligned}$$

Only the second term depends on $\mathbf{A}$, so we want to find $\mathbf{A}$ that maximizes $\text{tr}(\mathbf{M}^T\mathbf{N}\mathbf{A}^T)$. If we have the SVD $\mathbf{M}^T\mathbf{N} = \mathbf{U}\mathbf{D}\mathbf{V}^T$, then

$$\begin{aligned}\text{tr}(\mathbf{M}^T\mathbf{N}\mathbf{A}^T) &= \text{tr}(\mathbf{U}\mathbf{D}\mathbf{V}^T\mathbf{A}^T) = \text{tr}(\mathbf{U}\mathbf{D}(\mathbf{A}\mathbf{V})^T) = \text{tr}((\mathbf{A}\mathbf{V})^T\mathbf{U}\mathbf{D}) \\ &= \sum_{i=1}^d [(\mathbf{A}\mathbf{V})^T\mathbf{U}]_{ii}d_i,\end{aligned}$$

Proof (Cont'd)

where d_i is the i th entry on the diagonal of $\mathbf{D}$, and d is the number of non-zero entries in $\mathbf{D}$ (also the number of non-zero singular values of $\mathbf{M}^T \mathbf{N}$). Hence, to maximize the sum above, we want the diagonal entries of $(\mathbf{A}\mathbf{V})^T \mathbf{U}$ to be as large as possible. But since it is an orthogonal matrix, its diagonal entries cannot exceed 1 (if not the L_2 norm of that corresponding column/row will exceed 1). Thus, the sum is maximized when all the diagonal entries of $(\mathbf{A}\mathbf{V})^T \mathbf{U}$ are equal to 1. This occurs when

$$\begin{aligned}(\mathbf{A}\mathbf{V})^T \mathbf{U} &= \mathbf{I}, \\ \mathbf{A}\mathbf{V} &= \mathbf{U}, \\ \mathbf{A} &= \mathbf{U}\mathbf{V}^T.\end{aligned}$$

Constrained Orthogonal Procrustes Problem

- If we only allow A to be a rotation matrix, *i.e.*, $A \in SO(3)$, then $\det A = +1$.
- We can simply let

$$\hat{A} = U \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \det UV^T \end{bmatrix} V^T.$$

Jointly Solve R and T

- for $M \in \mathbb{R}^{n \times 3}$ and $N \in \mathbb{R}^{n \times 3}$, solve

$$\hat{R}, \hat{T} = \operatorname{argmin}_{R \in SO(3), T \in \mathbb{R}^{1 \times 3}} \|M^T - RN^T - T\|_F^2$$

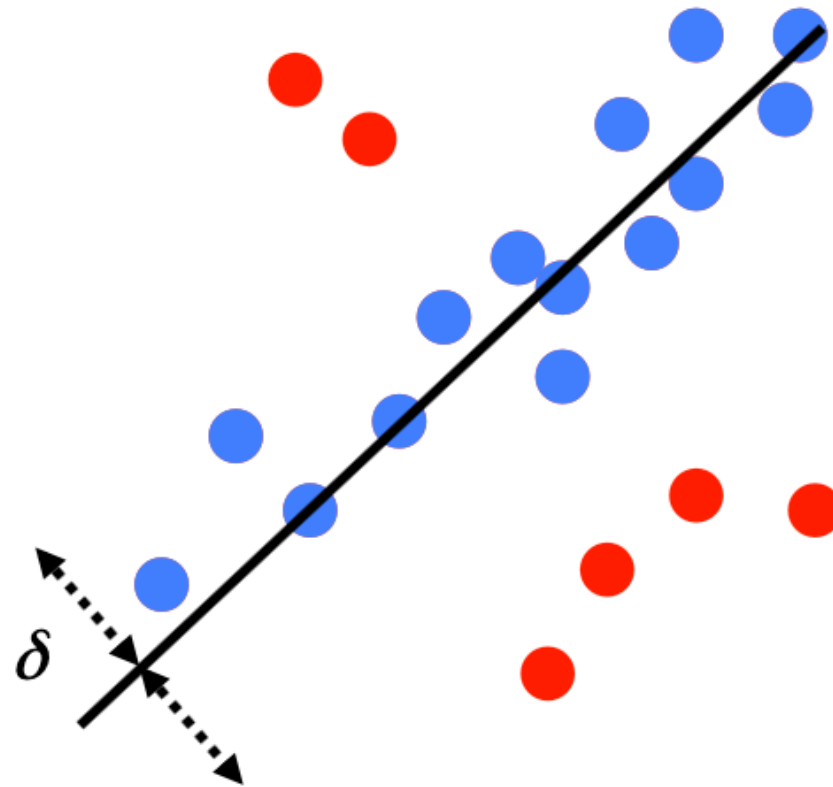
- First step: $\tilde{M} = M - \bar{M}$, $\tilde{N} = N - \bar{N}$, then we assume $\tilde{M}^T \approx \hat{R}\tilde{N}^T$
- Second step: using constrained Orthogonal Procrustes to get $\hat{R}$
- Third step: $\hat{T} = \overline{M^T - \hat{R}N^T}$

Rotation Fitting

- SVD is very sensitive to outliers.
- For fitting rotations, we need to use RANSAC.
 - How many pairs of 3D-3D correspondence do we need for hypothesis generation?
 - How to judge inlier/outliers?

RANSAC: **RAN**dom **SA**mples **C**onsensus

- Idea: we need to find a line that has the largest supporters (or inliers)

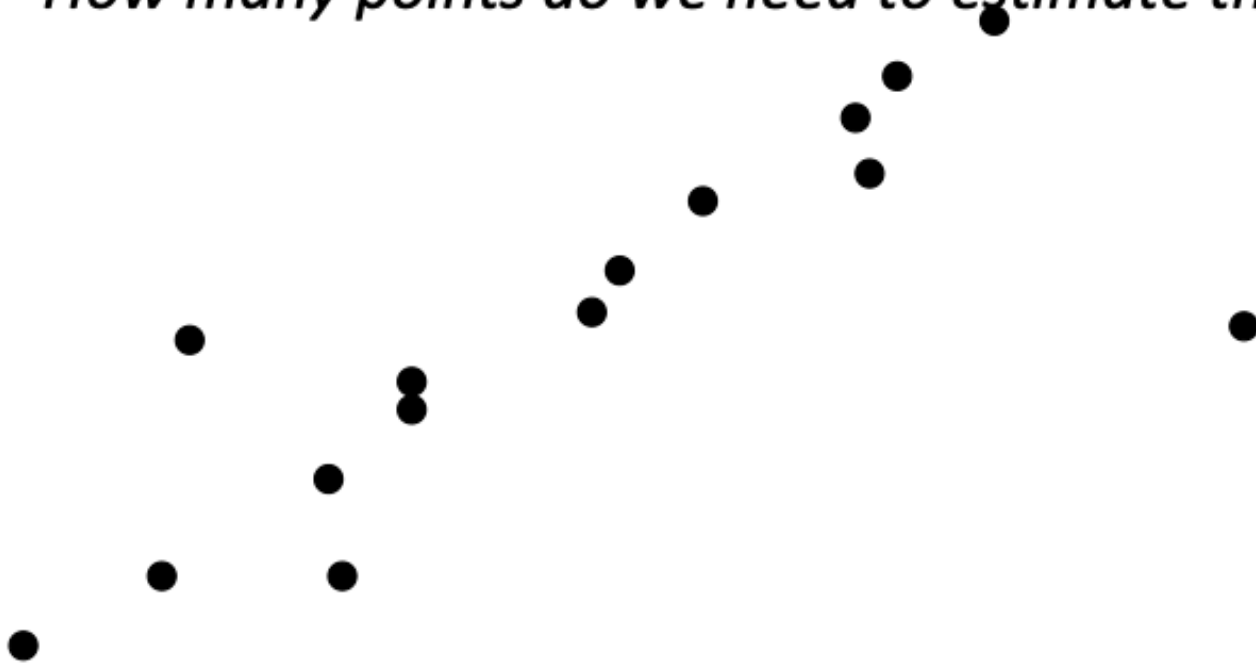


Fischler & Bolles in '81.

Read by Yourself

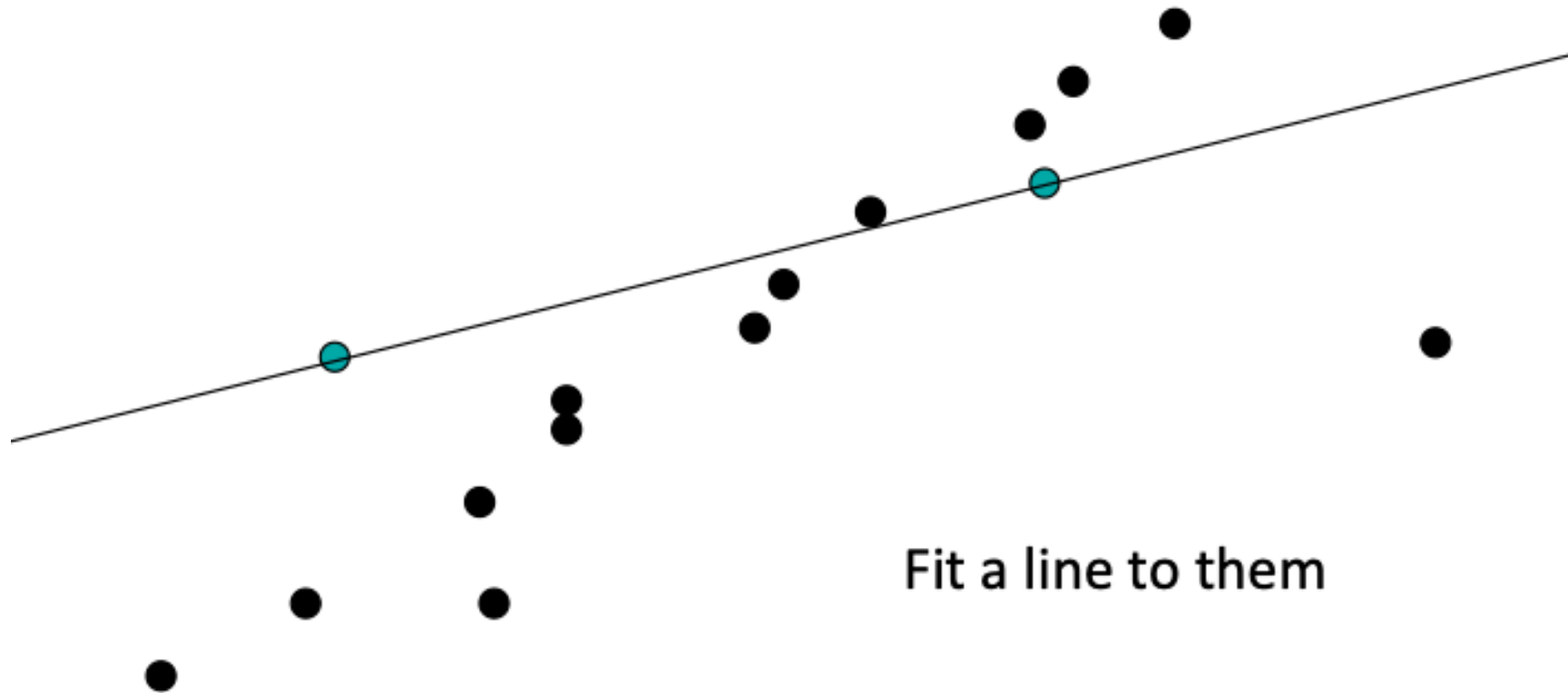
RANSAC Line Fitting

- Task: Estimate the best line
 - *How many points do we need to estimate the line?*



RANSAC Line Fitting

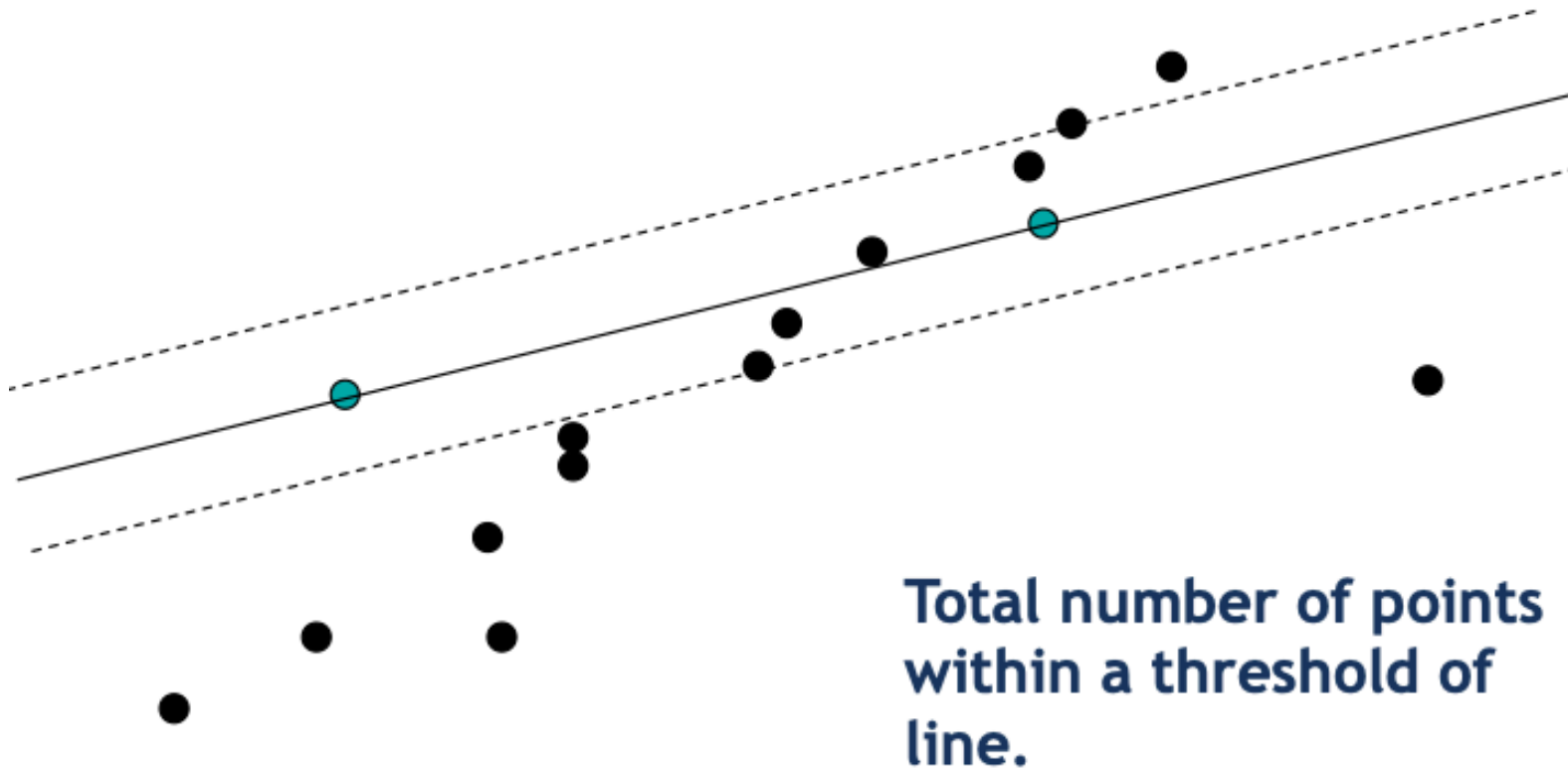
- Task: Estimate the best line



Fit a line to them

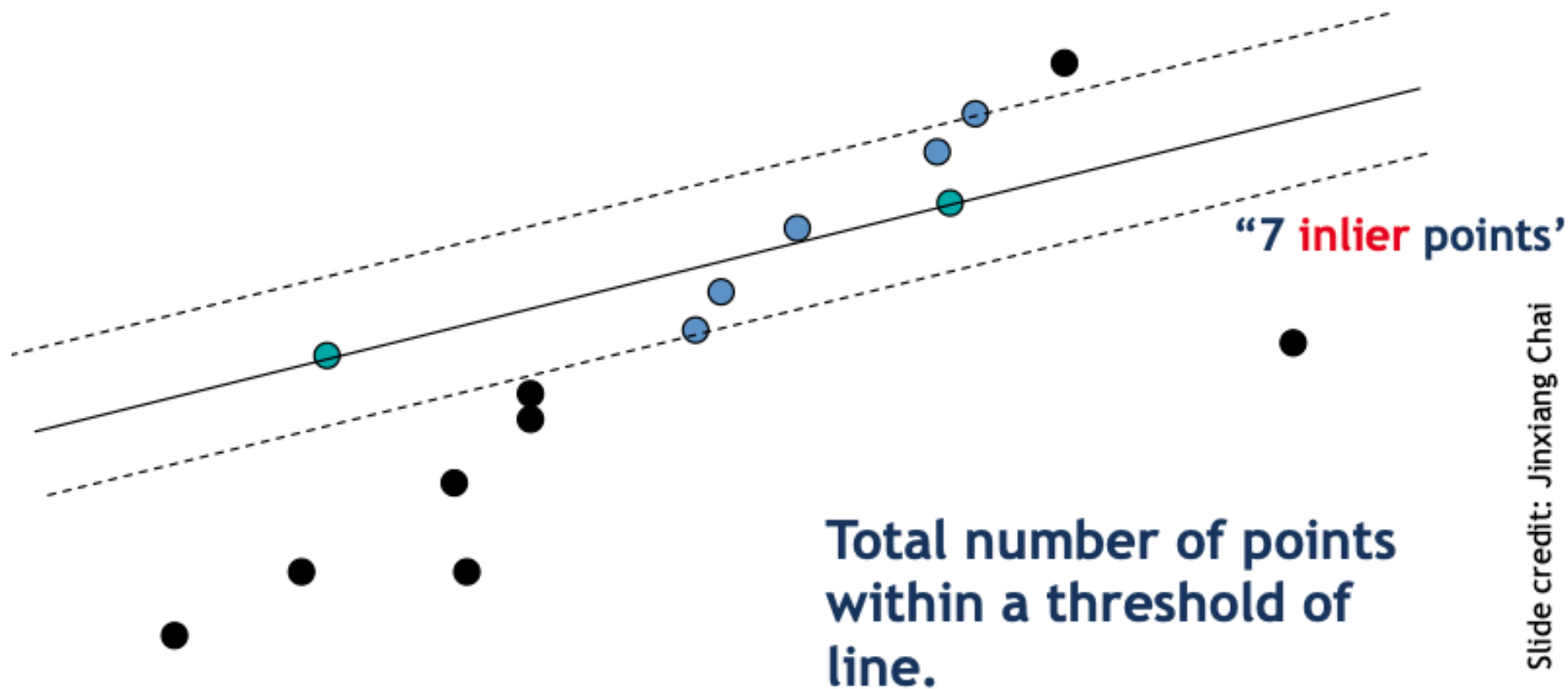
RANSAC Line Fitting

- Task: Estimate the best line



RANSAC Line Fitting

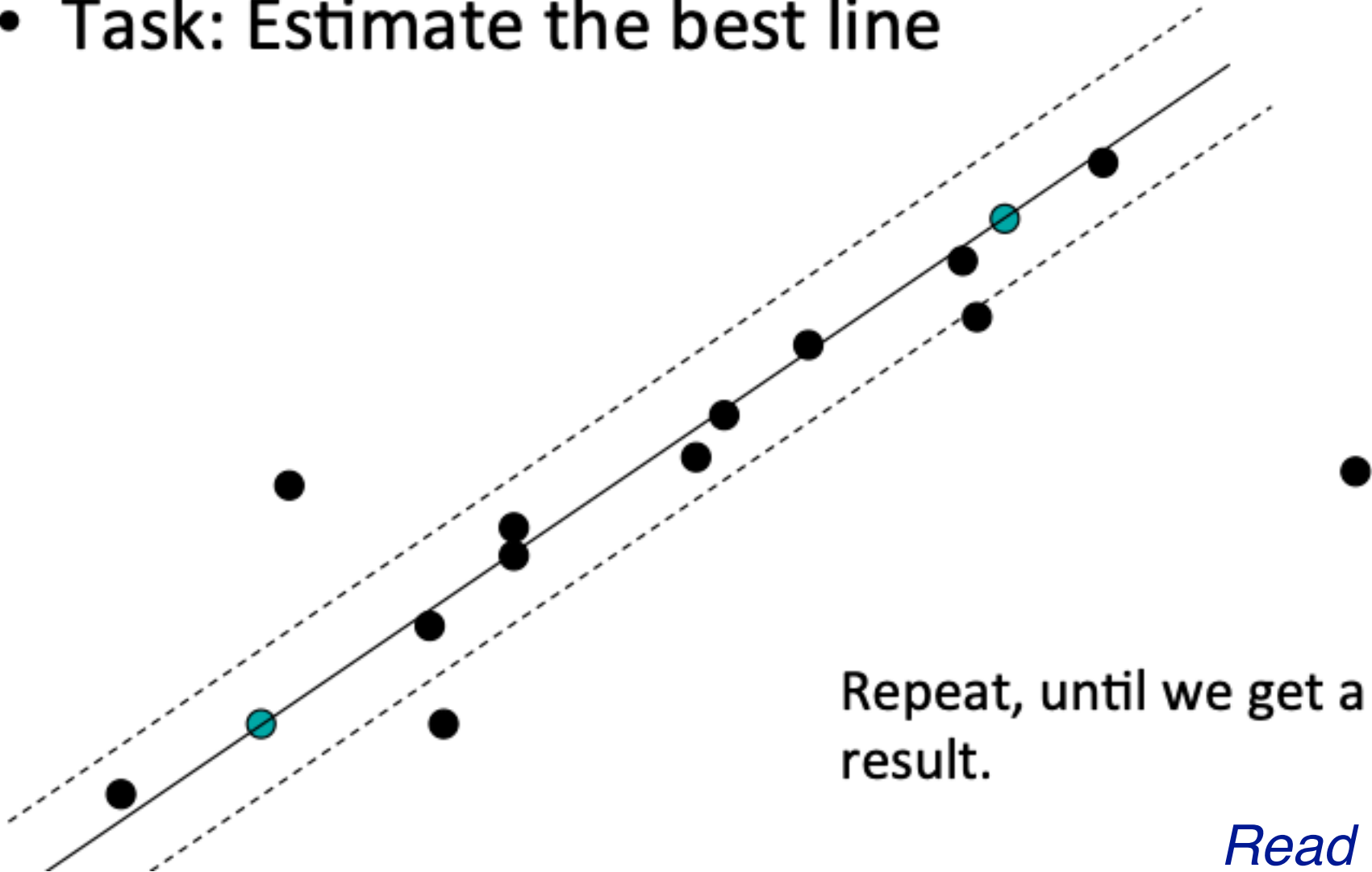
- Task: Estimate the best line



Slide credit: Jinxiang Chai

RANSAC Line Fitting

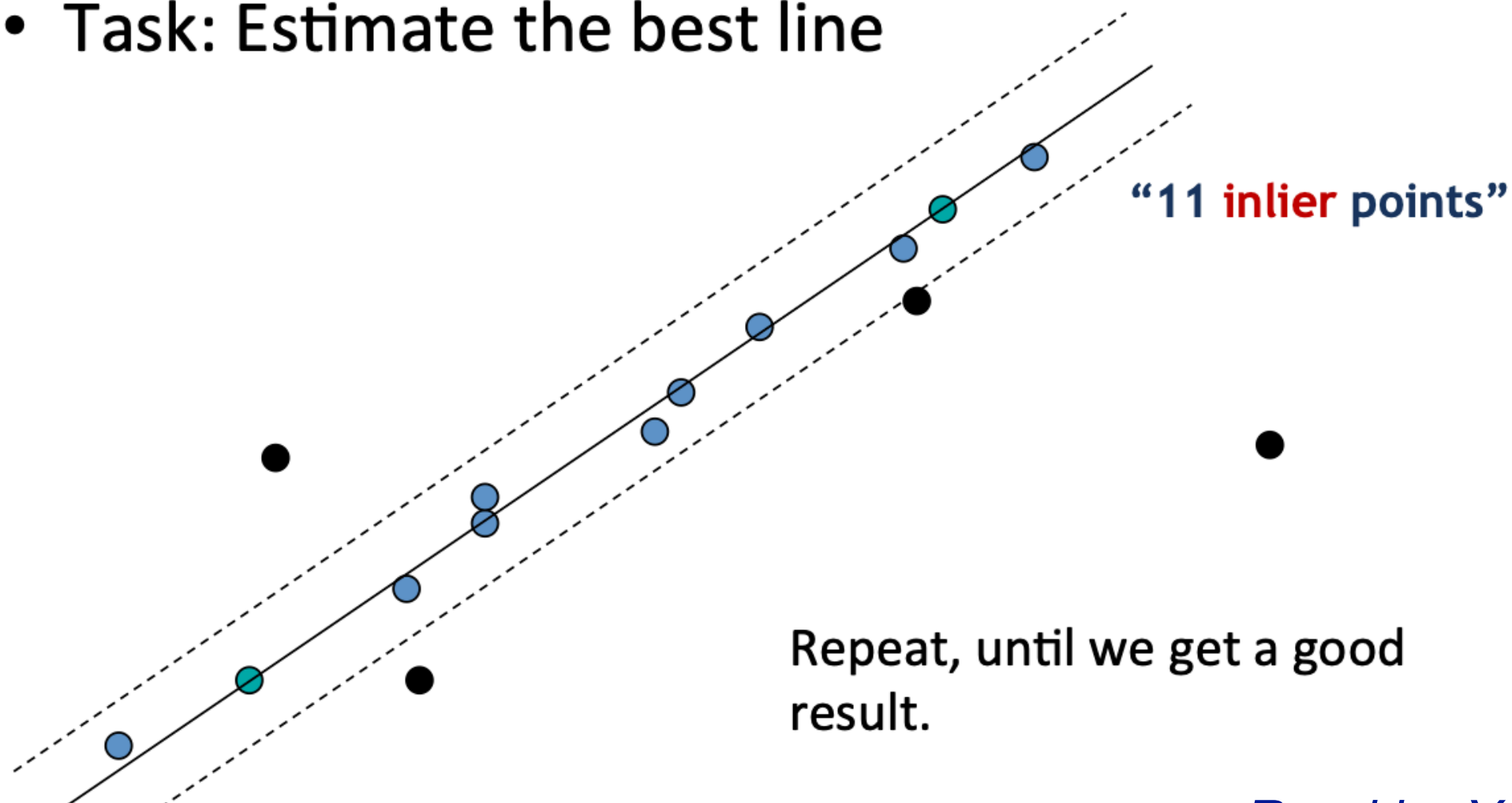
- Task: Estimate the best line



Repeat, until we get a good result.

RANSAC Line Fitting

- Task: Estimate the best line



RANSAC Line Fitting

RANSAC loop:

1. Randomly select a *seed group* of points on which to base transformation estimate (e.g., a group of matches)
 2. Compute transformation from seed group
 3. Find *inliers* to this transformation
 4. If the number of inliers is sufficiently large, re-compute least-squares estimate of transformation on all of the inliers
- Keep the transformation with the largest number of inliers

Perspective-n-Point (PnP) Algorithm

- For 2D–3D dense correspondence, if camera intrinsics K is known, use PnP algorithm to solve/fit R and T .

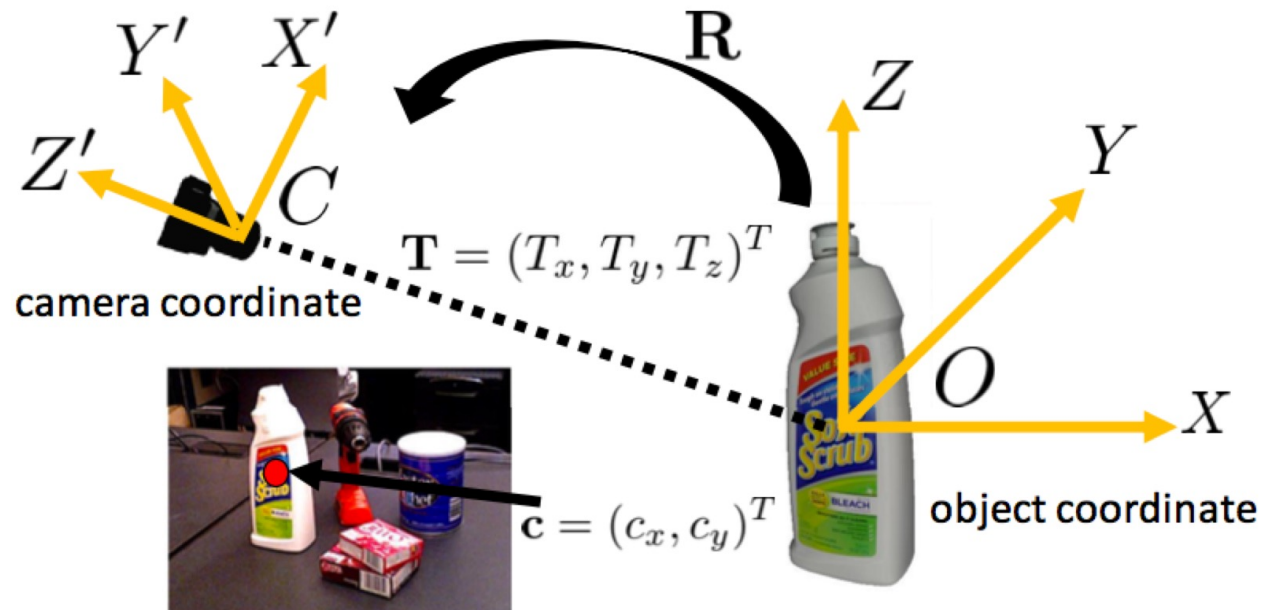
$$s p_c = K [R | T] p_w$$

where p_c and p_w are the homogeneous coordinates in the image and world frame, respectively, s is a scale factor for the image point.

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & \gamma & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

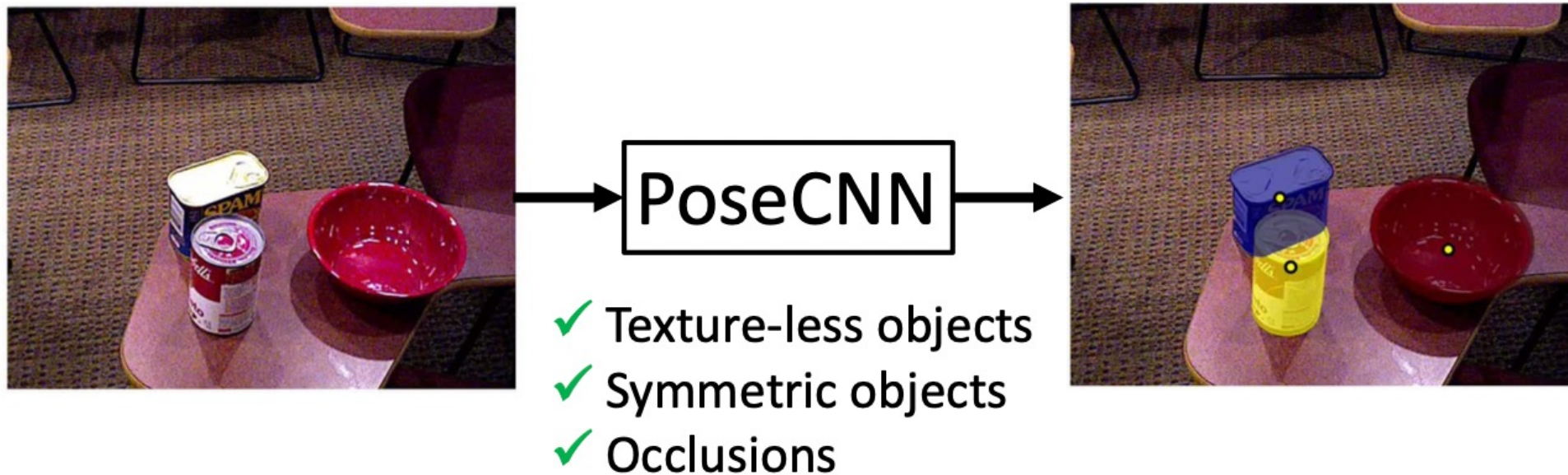
Instance-Level 6D Object Pose Estimation

Instance-Level 6D Pose Estimation



- Instance-level: a small set of known instances
- Pose is defined for each instance according to their CAD model.
- Input: RGB/RGBD
- Output: 6D pose(s) of an object or all known objects

PoseCNN



PoseCNN

- 3D Translation $\mathbf{T} = (T_x, T_y, T_z)^T$



2D center

$$\mathbf{c} = (c_x, c_y)^T$$

Distance T_z

2D Center Localization

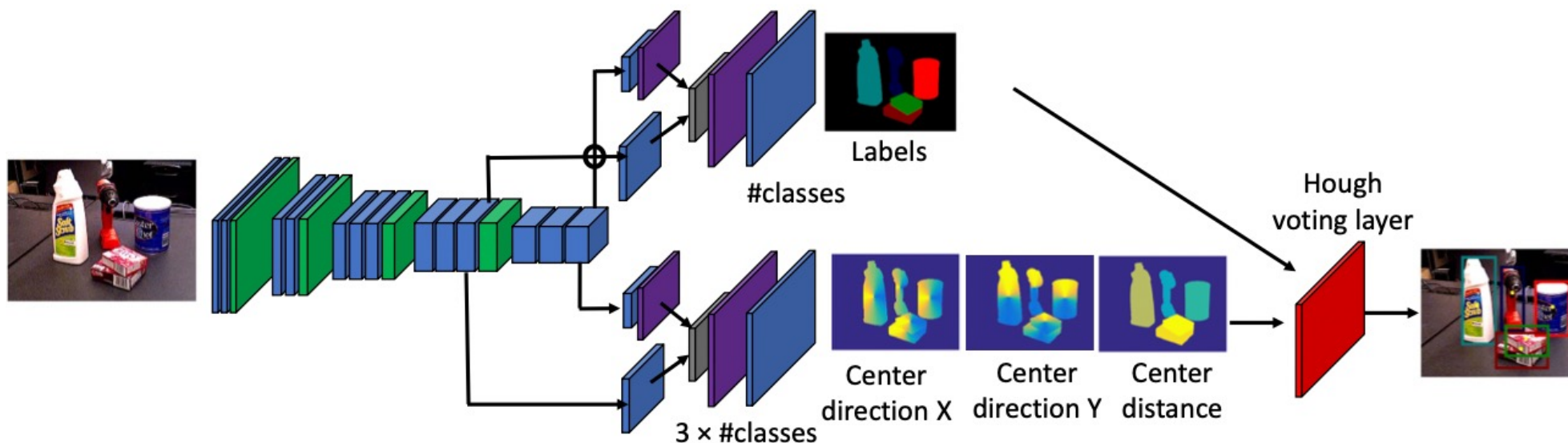
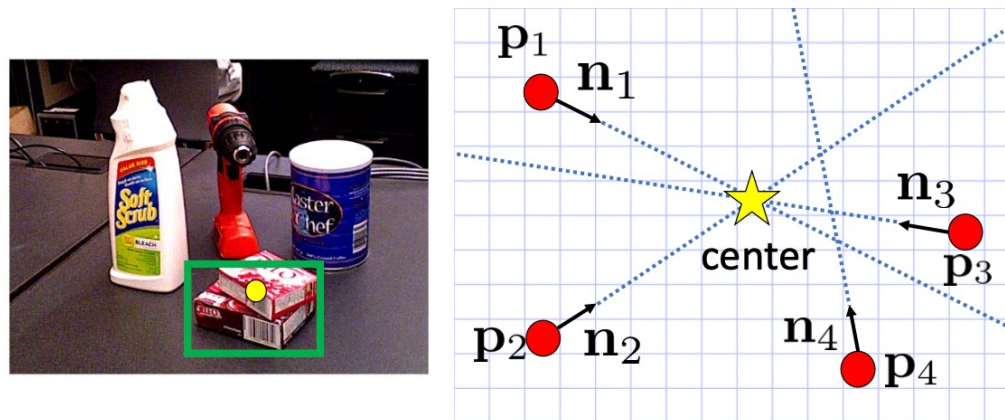
- 3D Rotation $\mathbf{R}$



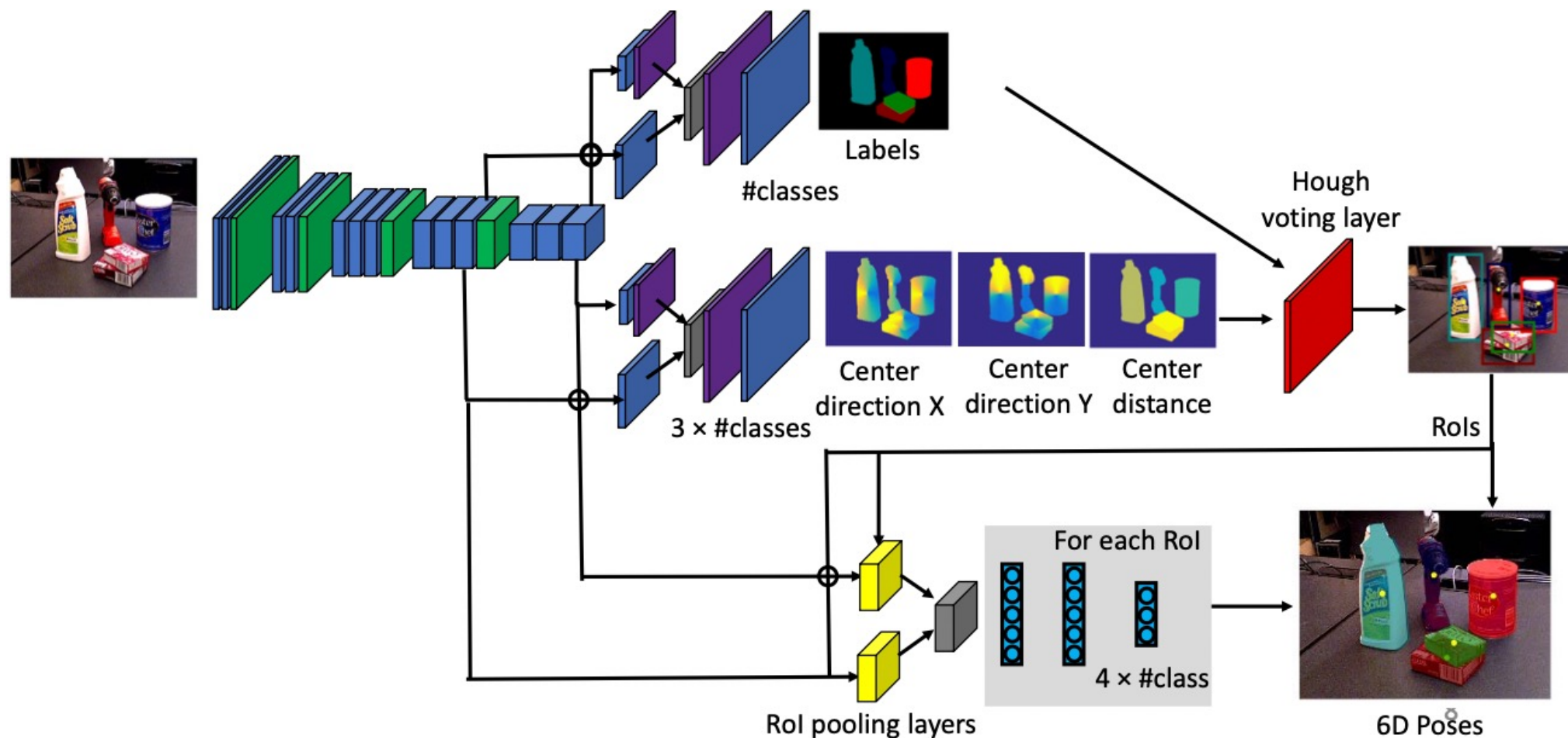
$\mathbf{R}$

3D Rotation Regression

PoseCNN: Translation Estimation



PoseCNN: Rotation Estimation



Regress quaternion.

PoseCNN: Result

Our YCB-Video Dataset

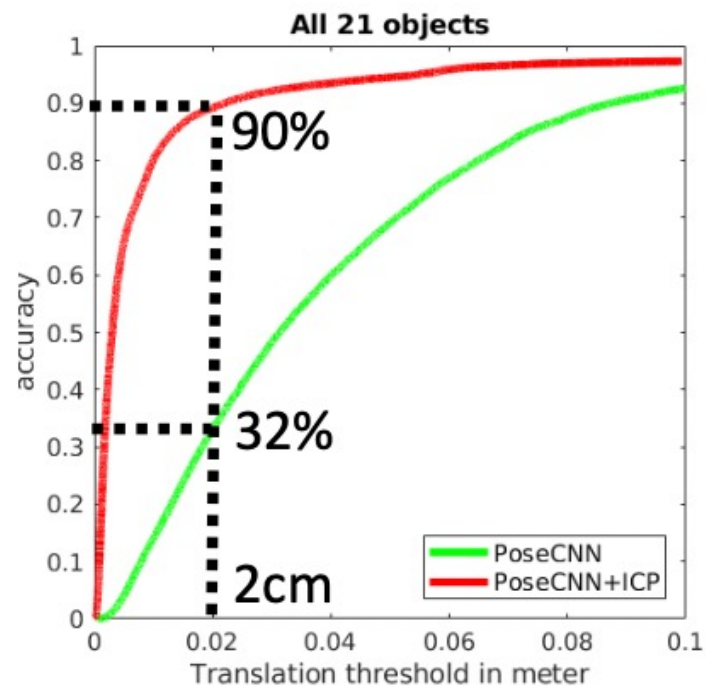


21 YCB Objects

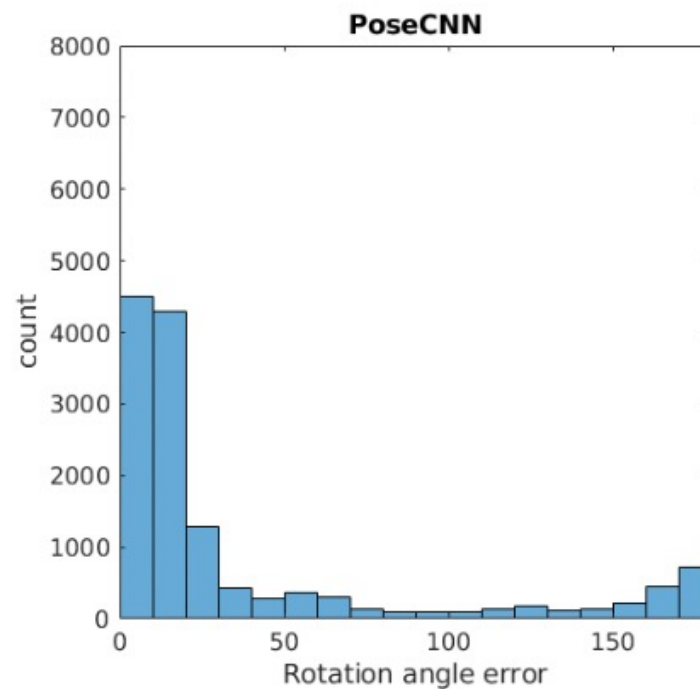


92 Videos, 133,827 frames₀

PoseCNN: Result



3D Translation Error Analysis

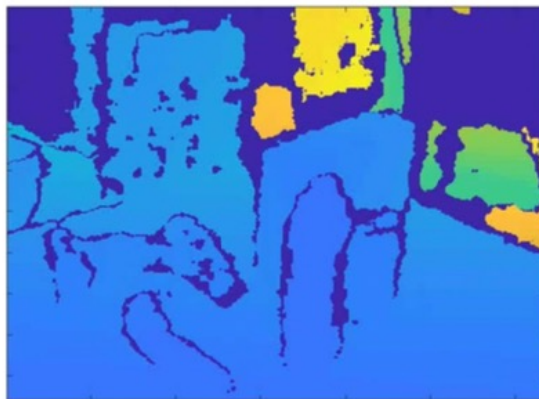


3D Rotation Error Analysis

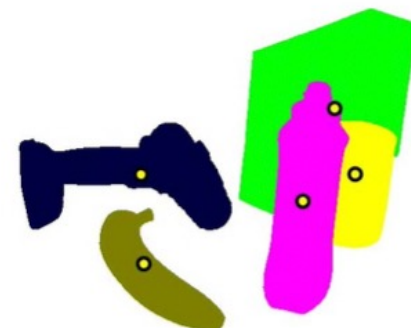
PoseCNN: Result Visualization



RGB



Depth



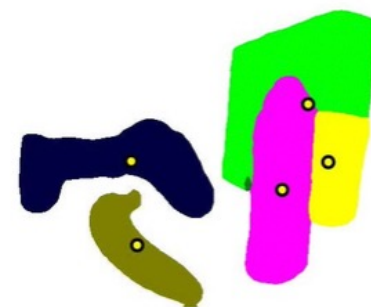
Groundtruth Labels



PoseCNN (RGB only)

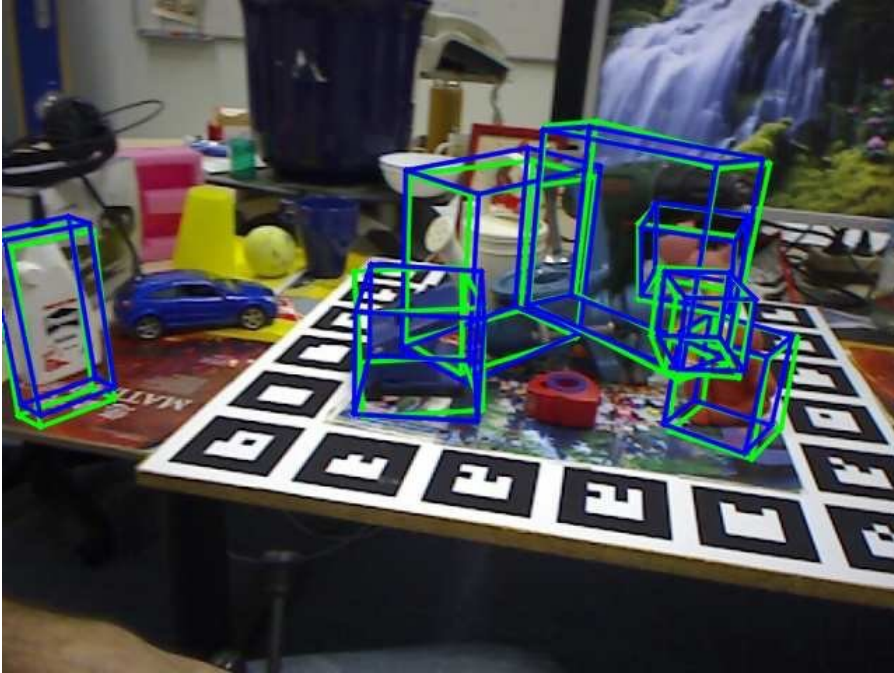


PoseCNN + ICP



Predicted Labels

Instance-Level 6D Pose Estimation

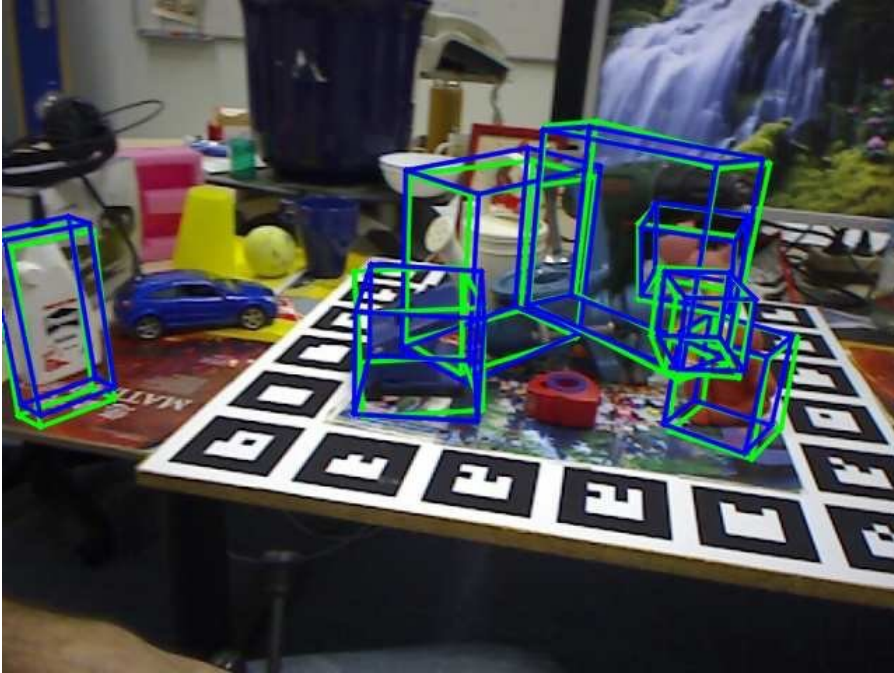


LINEMOD dataset

- A small set of known instances
- Ground truth CAD model available
- Useful in controlled environments

Hinterstoisser, Stefan, et al. "Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes." *Asian conference on computer vision*. Springer, Berlin, Heidelberg, 2012.

Limitations of Instance-Level 6D Pose Est.



LINEMOD dataset

Limitations

- Instance-specific
- Can't generalize to novel objects
- Needs CAD model

Hinterstoisser, Stefan, et al. "Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes." *Asian conference on computer vision*. Springer, Berlin, Heidelberg, 2012.

Category-Level 6D Object Pose Estimation

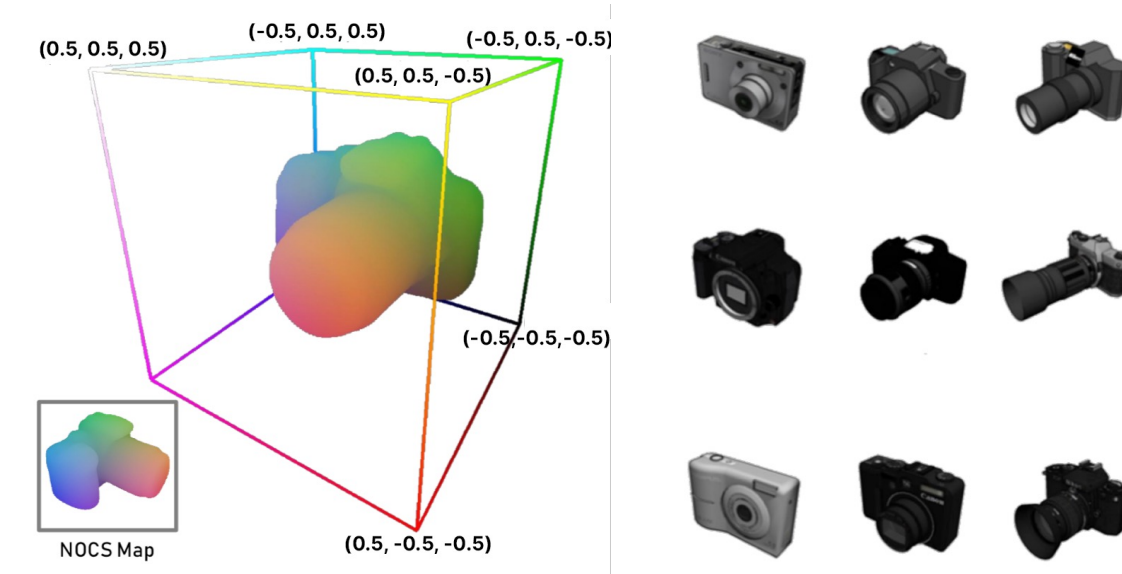
Category-Level 6D Pose



Our goal

- Generalize to novel object instances
- **3D input, full 6D pose output**
- Without the need to use CAD model

Normalized Object Coordinate Space for Category-Level 6D Object Pose and Size Estimation



He Wang, Srinath Sridhar, Jingwei Huang, Julien Valentin, Shuran Song, Leonidas Guibas

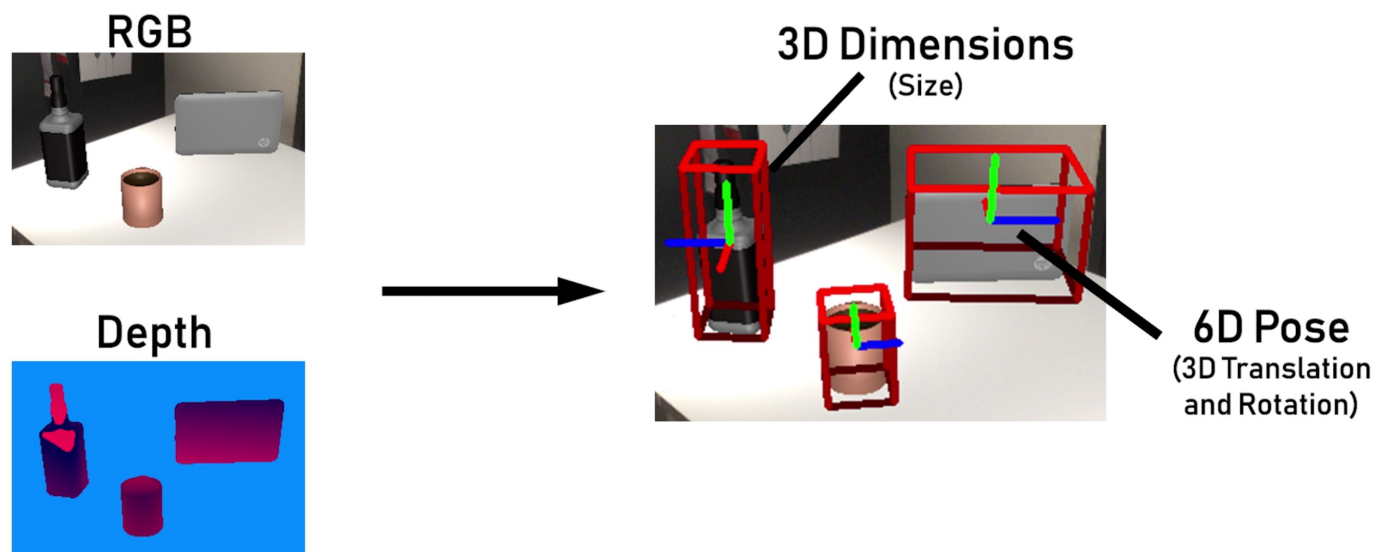


LONG BEACH
CALIFORNIA
June 16-20, 2019

Oral Presentation

Category-Level Object Pose and Size Estimation

Detecting and estimating **6D pose** and **3D size** of **previously unseen** objects from certain categories from RGBD images.



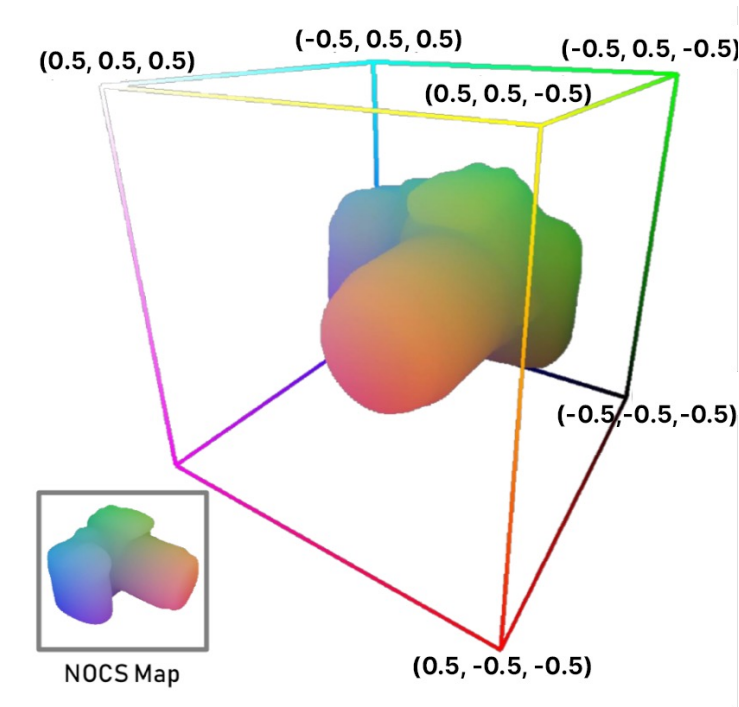
without knowing CAD models

Challenges

- Intra-category shape variations



Representation: Normalized Object Coordinate Space



RGB colors represent XYZ coordinates of shape.



Introduction to Embodied AI

Next week: Lecture 6, Vision and Grasping II

