L11: 6D Pose Estimation

Hao Su

Ack: Minghua Liu and Jiayuan Gu for helping to prepare slides

We aren’t talking about human pose

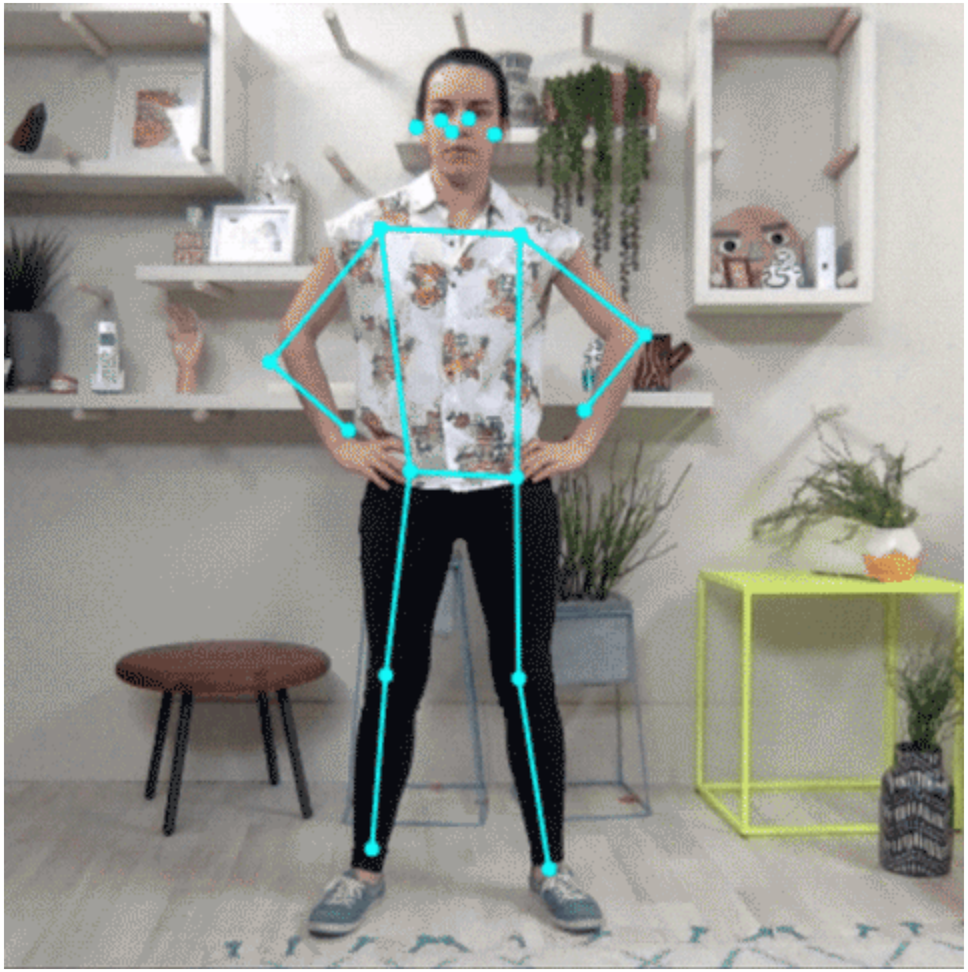
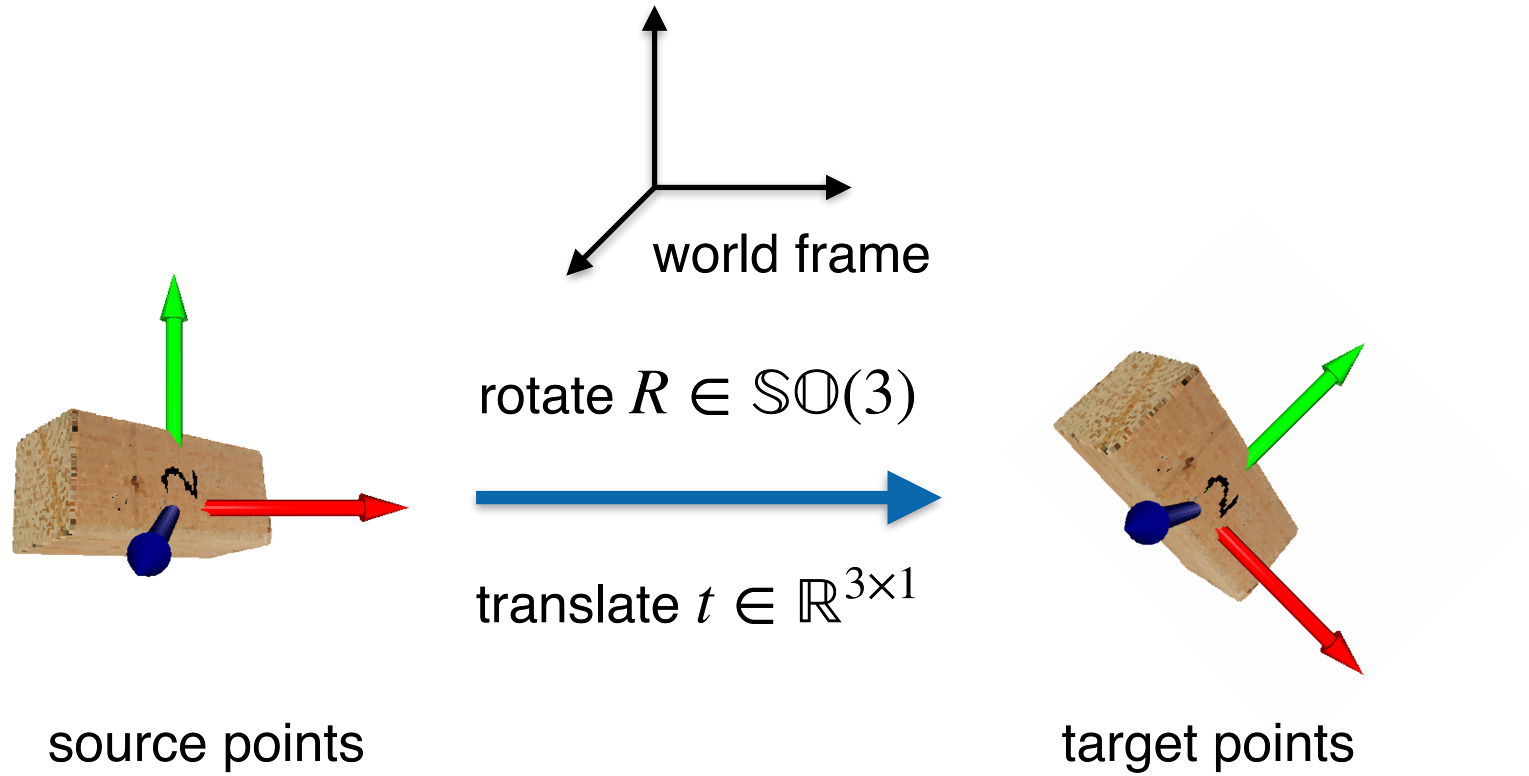


Figure from https://www.tensorflow.org/lite/models/pose_estimation/overview

We are talking about object pose

Figure from https://paperswithcode.com/task/6d-pose-estimation

Rigid Transformation

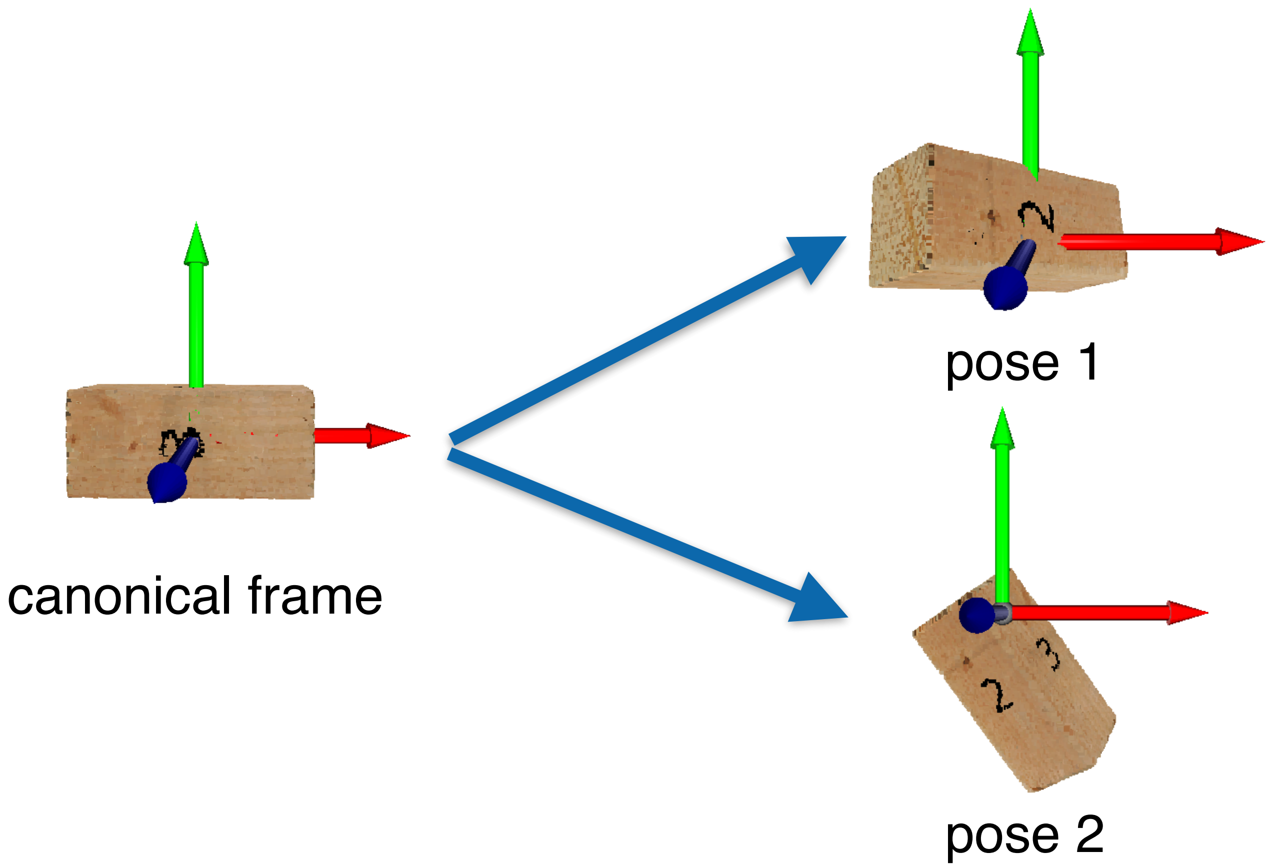


Transformation is relative!

Rigid Transformation

- Rigid transformation $T(p) = Rp + t$, where $p \in \mathbb{R}^{3\times 1}$
- Rotation $R \in \mathbf{SO}(3)$, and a translation $t \in \mathbb{R}^{3\times 1}$
- All the rigid transformations $\{T\}$ form the special Euclidean group, denoted by $\mathbf{SE}(3)$

6D Pose Estimation



recognize the 3D location and orientation of an object relative to a canonical frame

6D Pose & Rigid Transformation

- 6D pose: object-level rigid transformation, associated with a canonical frame
- rigid transformation: can be object-level or scene-level, no predefined canonical frame

Agenda

- Introduction

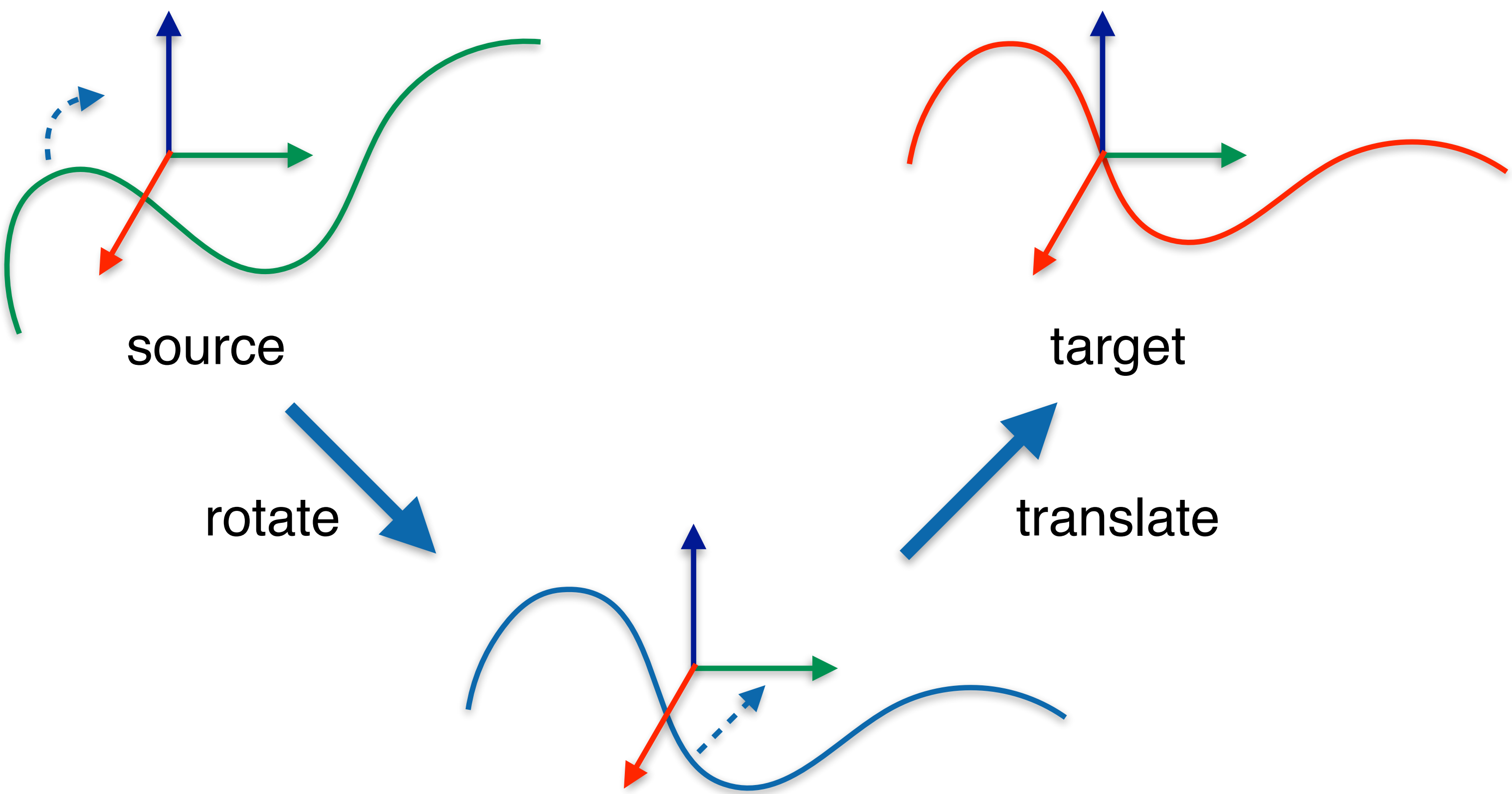
- Rigid transformation estimation
  - Closed-form solution given correspondences
  - Iterative closet point (ICP)

- Learning-based approaches
  - Direct approaches
  - Indirect approaches

Rigid Transformation Estimation

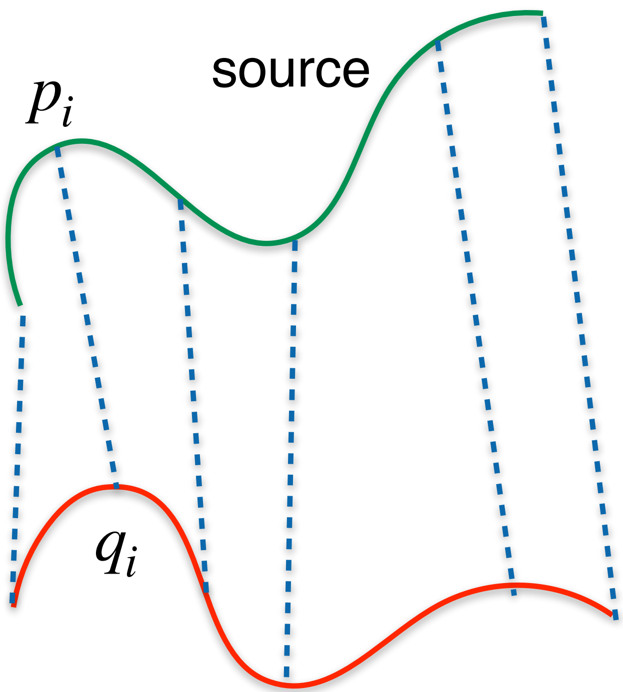
Rigid Transformation Estimation

Rigid transformation $T(p) = Rp + t$



Correspondence

Rigid transformation $T(p) = Rp + t$



$$q_1 = T(p_1) = Rp_1 + t$$

$$q_2 = T(p_2) = Rp_2 + t$$

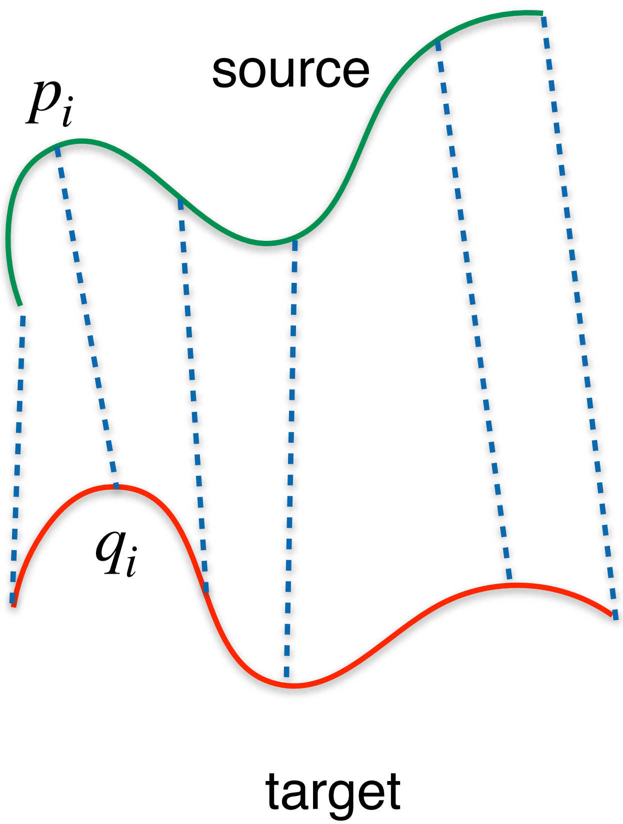
$$\vdots$$

$$q_n = T(p_n) = Rp_n + t$$

n equations from
n pairs of points

Estimate from Correspondence

Rigid transformation $T(p) = Rp + t$



How many pairs of points are required to uniquely define a rigid transformation?

Two Key Steps

- Find the correspondences between source and target
  - combinatorial problem
  - greedy heuristic or exhaustive search
- Estimate the rigid transformation given the correspondence
  - constrained (for rotation) optimization
  - closed-form solution

Two Key Steps

- Find the correspondence between source and target
  - combinatorial problem
  - greedy heuristic or exhaustive search

- Estimate the rigid transformation given the correspondence
  - constrained (for rotation) optimization
  - closed-form solution

Math is coming

Least-square Estimation of Rigid Transformation

- Given source points $P = \{p_i\}$, and target points $Q = \{q_i\}$, the objective (least-square error) is:

$$L = \sum_{i=1}^{n} \|Rp_i + t - q_i\|^2$$

Optimization Problem

- Parameters: $R \in \mathbb{SO}(3)$ and $t \in \mathbb{R}^{3\times1}$
- Target point cloud: $Q = \{q_i\}$
- Source point cloud: $P = \{p_i\}$
- Objective: $\min\limits_{R,t} \sum\limits_{i=1}^{n} \|Rp_i + t - q_i\|^2$

Step I: Representing t by R

- Assuming $R \in \mathbb{SO}(3)$ is known,
  - Recall the objective $L = \sum_{i=1}^{n} \|Rp_i + t - q_i\|^2$
  - Calculate the gradient $\frac{\partial L}{\partial t} = \sum_{i=1}^{n} (Rp_i + t - q_i)$
  - Solve $\frac{\partial L}{\partial t} = 0$:

$$\boxed{t = \frac{\sum_{i=1}^{n} q_i}{n} - \frac{\sum_{i=1}^{n} Rp_i}{n}}$$

Step I: Representing t by R

- Substituting the $t \in \mathbb{R}^{3\times 1}$ expressed by R, the objective can be simplified as

$$L = \sum_{i=1}^{n} \|R\bar{p}_i - \bar{q}_i\|^2,$$

where

$$\bar{p}_i = p_i - \frac{\sum_{i=1}^{n} p_i}{n},$$

$$\bar{q}_i = q_i - \frac{\sum_{i=1}^{n} q_i}{n}$$

Step II: Solve R

Objective: $$L = \sum_{i=1}^{n} \|R\bar{p}_i - \bar{q}_i\|^2$$

$$\Rightarrow R^* = \text{argmin}_R \|RP - Q\|_F,$$

subject to

$$R^T R = \mathbf{I},$$
$$\det(R) = 1$$

where $P = [p_1, p_2, \cdots, p_n] \in \mathbb{R}^{3\times n}$,

$$Q = [q_1, q_2, \cdots, q_n] \in \mathbb{R}^{3\times n}$$

Step II: Solve R

- **Orthogonal Procrustes Problem**

  $$\text{argmin}_R \|RP - Q\|_F, \text{ subject to } R^T R = \mathbf{I}$$

- Notice: No determinant constraint
- This problem has a closed form solution!
  - Project $M = QP^T$ to the space of orthogonal matrices
  - Numerically, the magical SVD comes!
    - If $M = U\Sigma V^T$,
    - then $R = UV^T$
- The proof can be found on the wiki

Step II: Solve R

- How to satisfy the determinant constraint?
- Assume $R = UV^T$
  - If $\det(R) = -1$, then flip the sign of the last column of V
- The proof can be found in Umeyama's paper

Summary of Closed-Form Solution (Known Correspondences)

- Known: $P = \{p_i\}, Q = \{q_i\}$
- Objective: $\min_{R,t} \sum_{i=1}^{n} \|Rp_i + t - q_i\|^2$
- Solution
  - $\sum_{i=1}^{n} (q_i - \bar{q}_i)(p_i - \bar{p}_i)^T = U\Sigma V^T$ (SVD)
  - $R = UV^T$ (flip the sign of the last column of V if $\det(R) = -1$)
  - $t = \frac{\sum_{i=1}^{n} q_i}{n} - \frac{\sum_{i=1}^{n} Rp_i}{n} := \bar{q} - R\bar{p}$

Two Key Steps

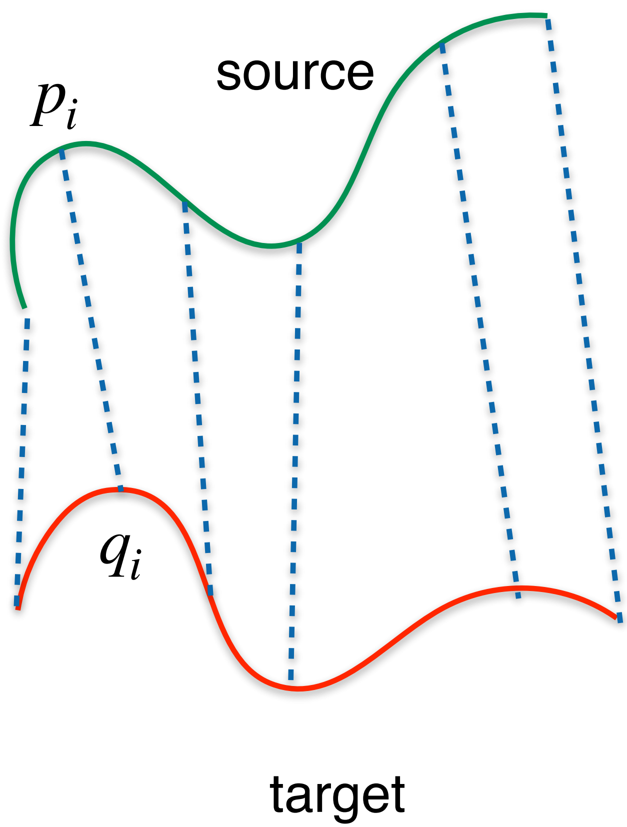
- Find the correspondence between source and target
  - combinatorial problem
  - greedy heuristic or exhaustive search
- Estimate the rigid transformation given the correspondence
  - constrained (for rotation) optimization
  - closed-form solution

Iterative Closet Point (ICP)

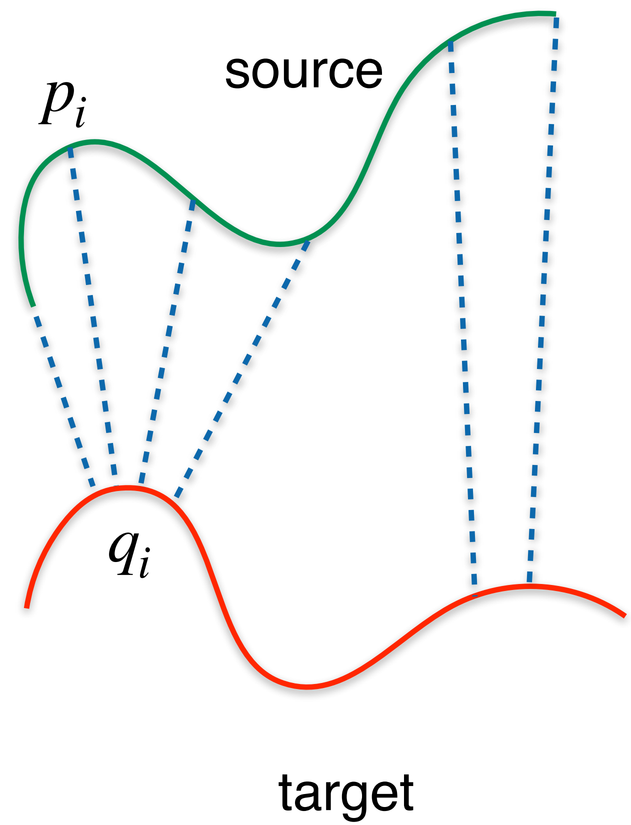
Heuristic

- The closest point might be the corresponding point
- If starting from a transformation close to the actual one, we can iteratively improve the estimation

Find Correspondence

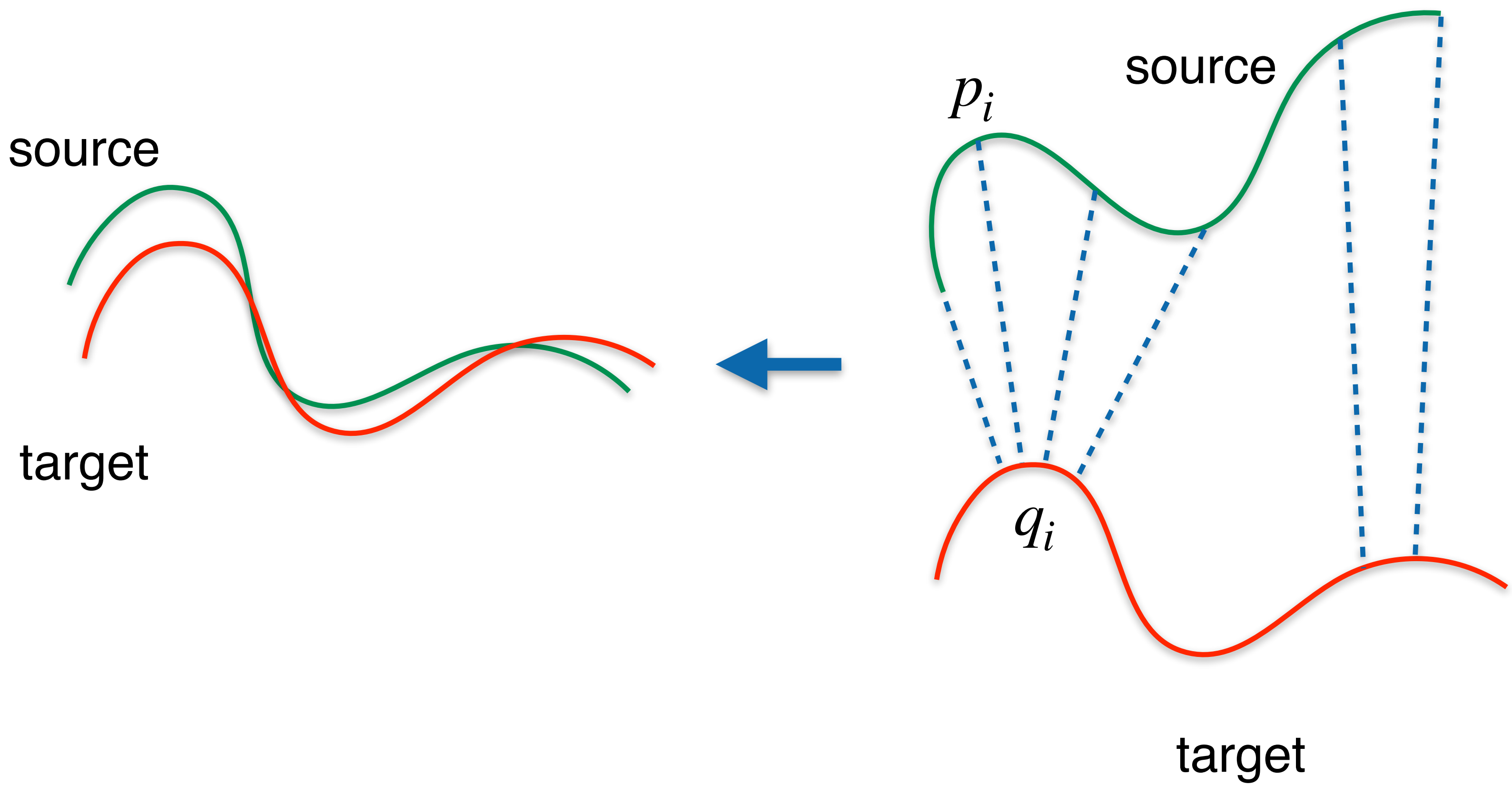


GT correspondence



ICP correspondence

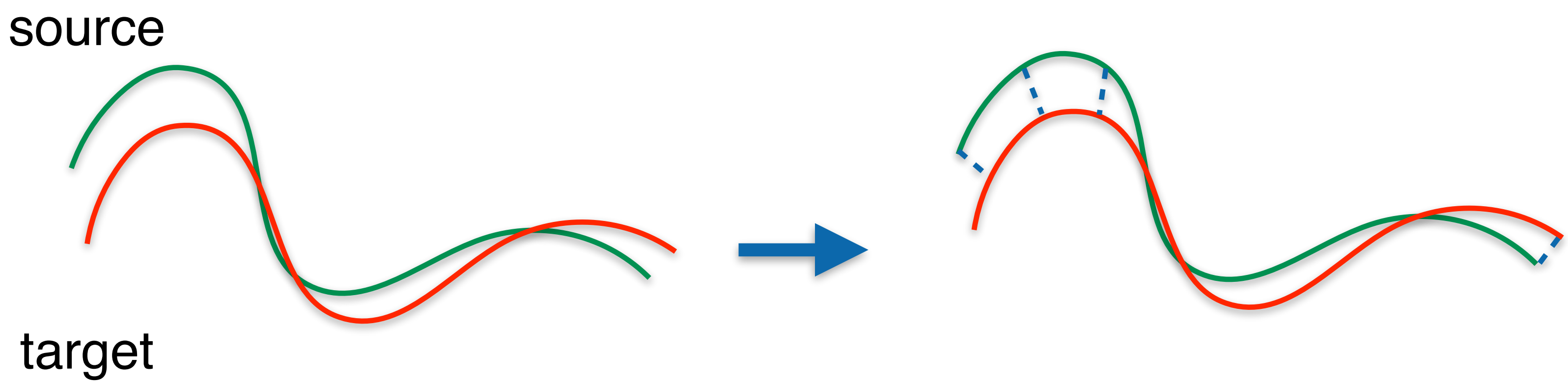
Update Transformation



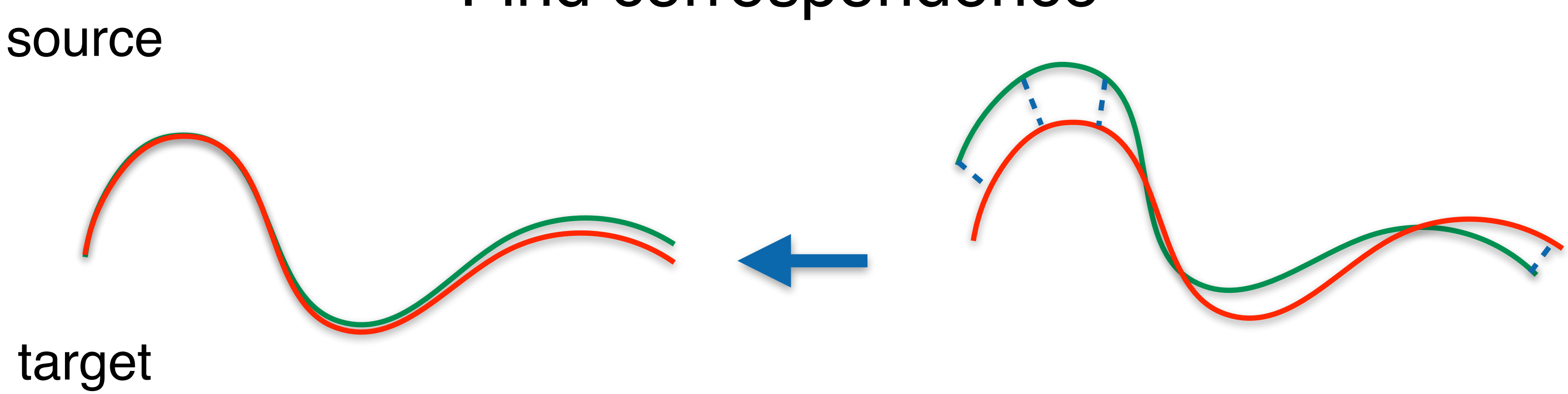
Update transformation by minimizing the least square error

ICP correspondence

Iterate to Refine



Find correspondence



Update transformation

General ICP Algorithm

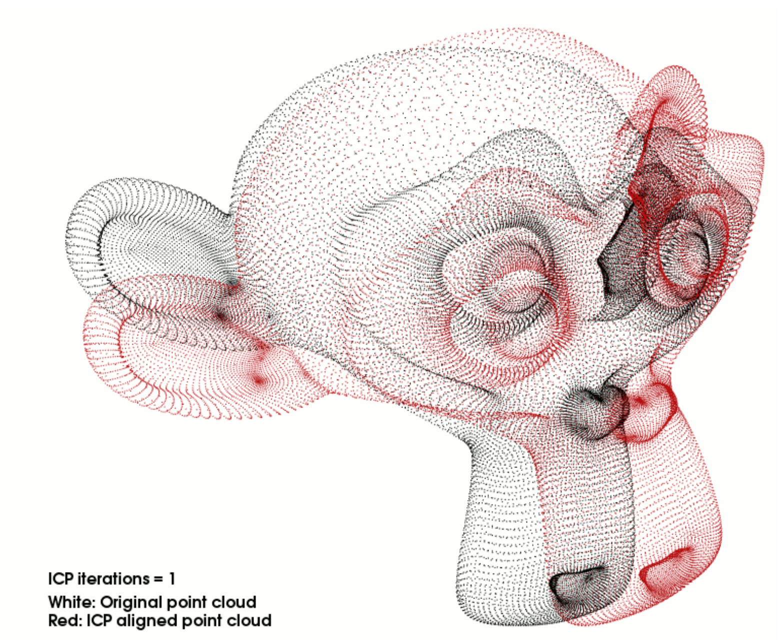
Starting from an initial transformation $T = (R, t)$

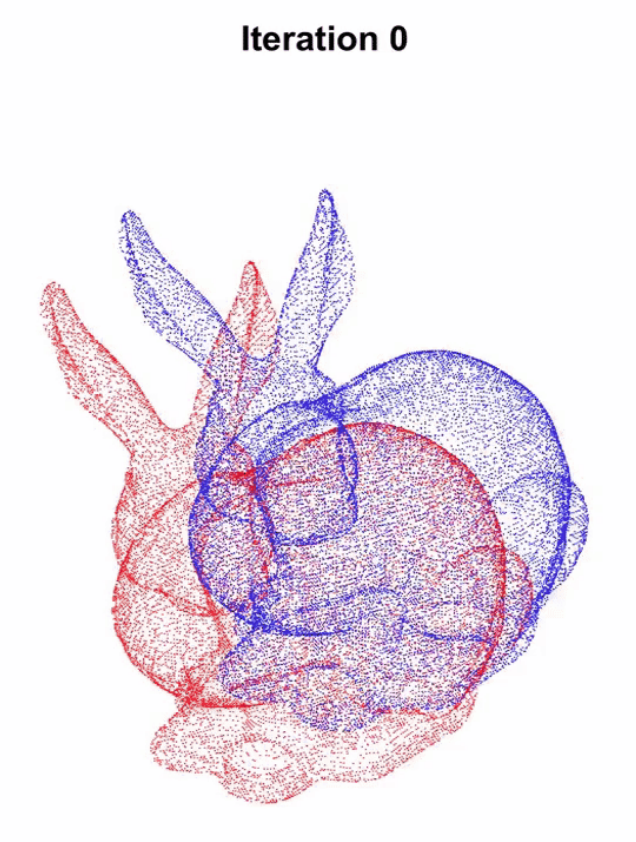
1. Find correspondence: for each point in the source point cloud transformed with current transformation, find the nearest neighbor in the target point cloud

2. Update the transformation by minimizing an objective function $E(T)$ over the correspondence

3. Go to step 1 until the transformation is not updated

Illustration





Animation from https://github.com/yassram/iterative-closest-point

Animation from https://github.com/pglira/simpleICP

Improve ICP

- Objective functions
  - PointToPoint
  - PointToPlane (faster convergence, but requires normal computation)
- Outlier removal: abandon pairs of points with too large distance

Useful Libraries

- Open3D
  - http://www.open3d.org/docs/release/tutorial/pipelines/icp_registration.html
  - http://www.open3d.org/docs/release/tutorial/geometry/kdtree.html

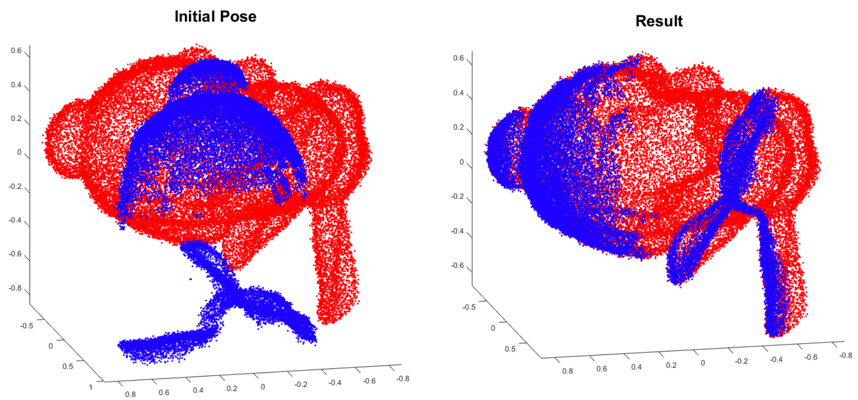
- PCL: https://pointclouds.org/documentation/group__registration.html

Limitation of ICP

- However, even with improvement
  - Easy to get stuck in local minima
  - Require a good initialization to work

Acquire the Initialization for ICP

- Go-ICP (correspondences based on features)
  - http://www.open3d.org/docs/release/tutorial/pipelines/global_registration.html
  - https://github.com/yangjiaolong/Go-ICP



- Teaser (more robust to outliers)
  - https://github.com/MIT-SPARK/TEASER-plusplus

Read by yourself

Learning-based Approach

Two Categories of Approaches

- Direct: predict (R, t) directly
- Indirect: predict corresponding pairs $\{(p_i, q_i)\}$
  - points in the canonical frame $\{p_i\}$
  - points in the camera frame $\{q_i\}$
  - estimate (R, t) by solving

$$\min_{R,t} \sum_{i=1}^{n} \|Rp_i + t - q_i\|^2$$

Direct Approaches

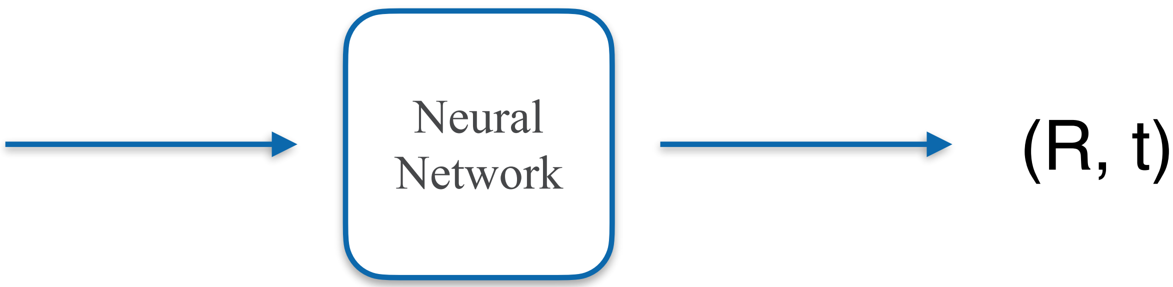
Direct Approaches

- Input: cropped image/point cloud/depth map of a single object
- Output: (R, t)

Image

Depth map

Point cloud



Challenges for Direct Approaches

- The choice of the representation of R
  - Recall Lec 5: rotation matrix, Euler angles, quaternion, axis-angle, … (more will be introduced in this lecture)
  - Which is more learnable for neural networks?

Direct Approaches

Representation of rotation

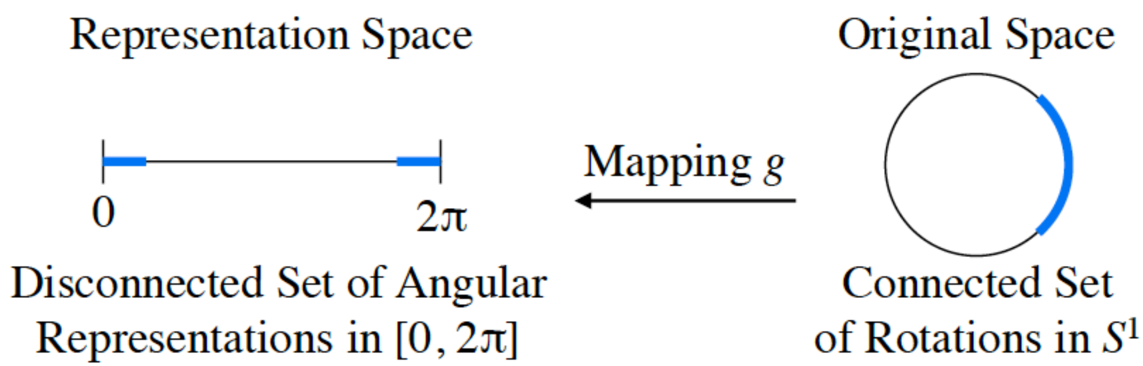
Loss for rotation

Example: DenseFusion

Continuity: 2D Rotation Example

$$M = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}$$

2D rotation can be parameterized by θ



However, the mapping is discontinuous due to the topology difference

Zhou, Yi, et al. "On the continuity of rotation representations in neural networks." CVPR 2019

Rotation Representations

Representation	Continuous parameterization	Unique for a rotation
Rotation Matrix	✔	✔
Euler Angle	No	No (gimbal lock)
Angle-axis	No	No (singularity)
Quaternion	No	No (double covering)

Note that neural networks are generally continuous

Fitting a discontinuous function is
not friendly to neural networks

Continuous Representation

- Next, we will introduce two continuous representations: 6D (vector) and 9D (vector)

6D Representation

- 6D representation: $[a_1^T, a_2^T]$, where $a_1, a_2 \in \mathbb{R}^{3x1}$
- Convert $x = [a_1^T, a_2^T]$ to a rotation matrix $R = [b_1, b_2, b_3]$ through the Gram-Schmidt process, where $b_1, b_2, b_3 \in \mathbb{R}^{3x1}$

$$b_1 = \frac{a_1}{\|a_1\|}, b_2 = \frac{a_2}{\|a_2\|} - \langle b_1, \frac{a_2}{\|a_2\|} \rangle b_1, b_3 = b_1 \times b_2$$

- Convert a rotation matrix to 6D: $x = [b_1^T, b_2^T]$

Zhou, Yi, et al. "On the continuity of rotation representations in neural networks." CVPR 2019

9D Representation

- 9D representation (rotation matrix): $X \in \mathbb{R}^{3x3}$
- Find the closest rotation matrix $R \in SO(3)$
  - Recall the Orthogonal Procrustes Problem
    $\text{argmin}_R \|RP - Q\|_F$, subject to $R^T R = \mathbf{I}$
  - equivalent to $P = I$ and $Q = X$
- Implementation

Levinson, Jake, et al. "An analysis of svd for deep rotation estimation." NeurIPS 2020

Direct Approaches

Representation of rotation

Loss for rotation

Example: DenseFusion

Loss for Direct Approaches

- shape-agnostic: distance between (R, t) and (R_{GT}, t_{GT})
- shape-aware: distance between the same shape $X \in \mathbb{R}^{3 \times n}$ transformed by (R, t) and (R_{GT}, t_{GT}) respectively

Shape-agnostic Loss

- Rotation
  - Geodesic distance (angle difference, relative rotation error) on $SO(3)$: $\arccos[\frac{1}{2}(tr(R_{GT}R^T) - 1)]$
  - Mean square error over $\mathbb{R}^{3\times3}$: $\|R - R_{GT}\|_F^2$
  - L_p distance on representation x: $\|x - x_{GT}\|_p$

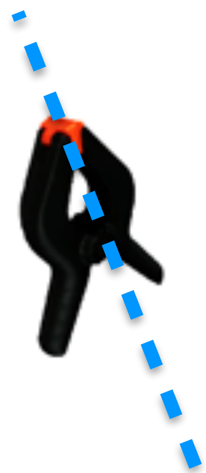
- Translation
  - L_p distance: $\|t - t_{GT}\|_p$

Challenges: Symmetry

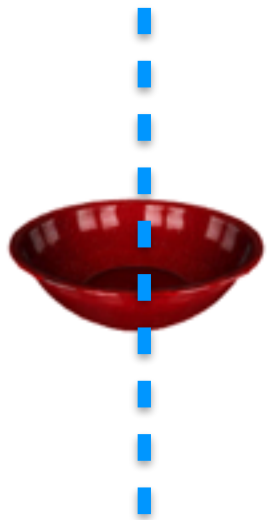
- Symmetry introduces ambiguities of GT labels
- Require specially designed losses to tackle it

Examples of symmetric objects in YCB dataset

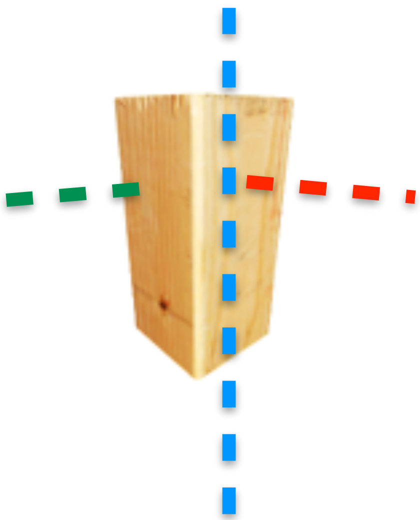
Rotational Symmetry



1 symmetry axis
symmetry order 2



1 symmetry axis
symmetry order infinite



3 symmetry axis
symmetry order (4, 2, 2)

Shape-agnostic Loss for Symmetric Objects

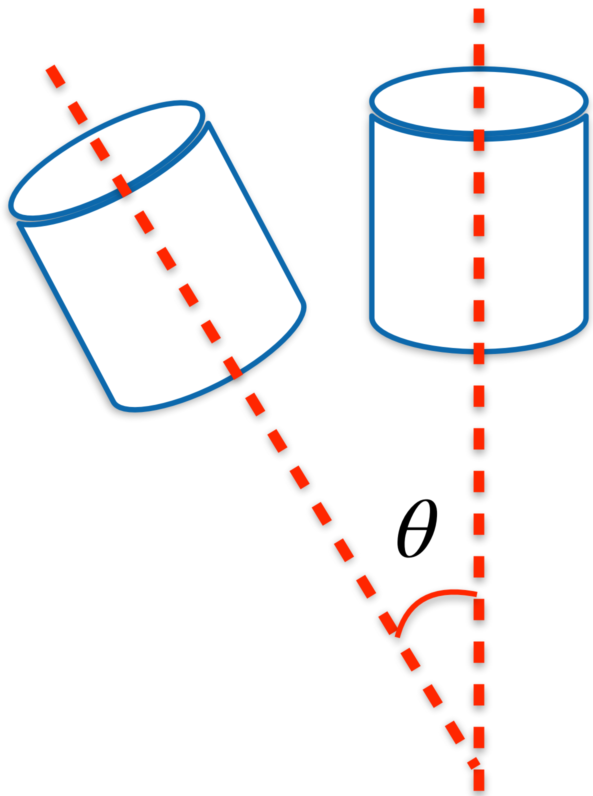
- Symmetry group: Multiple symmetry-equivalent GT rotations $\mathscr{R} = \{R_{GT}^1, R_{GT}^2, \cdots, R_{GT}^n\}$ (n is the order of symmetry)

- *Min of N* loss (for finite order of symmetry)

$$\min_{R_{GT}^i \in \mathscr{R}} L(R_{GT}^i, R)$$

Q: How to deal with infinite order?

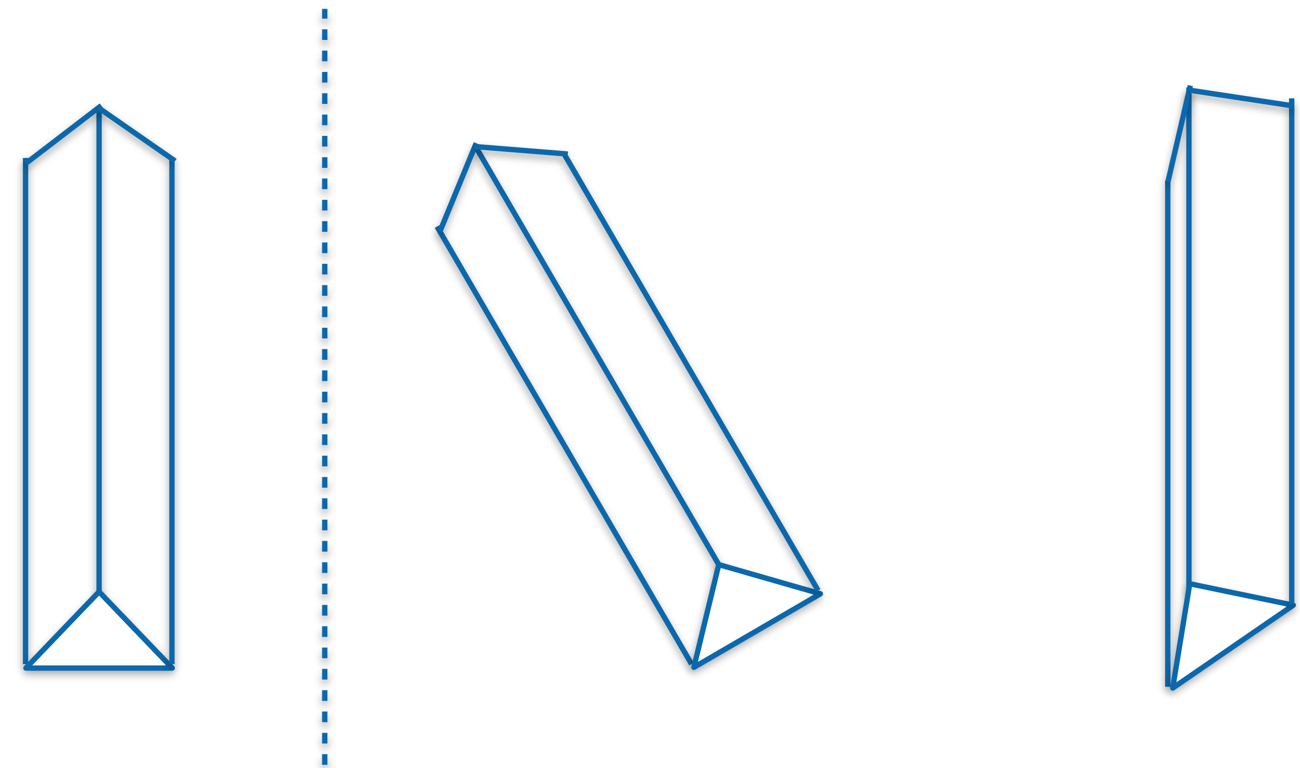
Shape-agnostic Loss for Symmetric Objects (Infinite Order)



Use the angle between two symmetry axes

Read by yourself

Motivation of Shape-aware Loss



target pose   predicted pose1   predicted pose2

Similar angle difference, but very different perception error

Shape-aware Loss

- Given a shape $X \in \mathbb{R}^{3 \times n}$, the loss for rotation can be the distance between RX and $R_{GT}X$
- e.g., $L = \|RX - R_{GT}X\|_F^2$ (per-point Mean Square Estimation (MSE))

Shape-aware Loss for Symmetry Objects

- With $\|RX - R_{GT}X\|_F^2$ as the distance metric, we can also apply the *Min of N* loss

$$L = \min_{R_{GT}^i \in \mathscr{R}} \|RX - R_{GT}^i X\|_F^2$$

Shape-aware Loss for Symmetry Objects

- Recall the distance metrics for point clouds in Lec7
  - Chamfer distance
  - earth mover distance

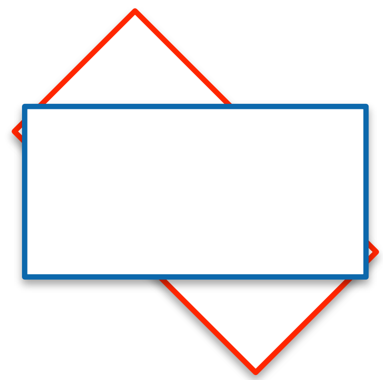
- Those distance metrics are compatible with symmetry (even infinite order)

- But requires a shape and might get stuck in local minima

Xiang et al., “PoseCNN: A Convolutional Neural Network for 6D Object Pose Estimation in Cluttered Scenes”, RSS 2018

Example of Local Minima for Chamfer Distance

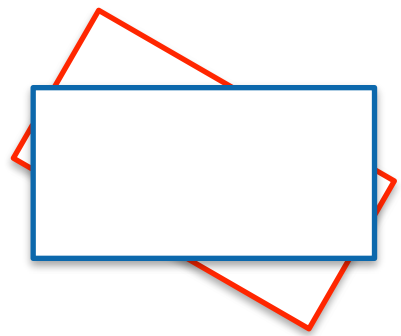
Rotate the red left a bit?



CD=2.02

No local minima for Min of N

Example of Local Minima for Chamfer Distance

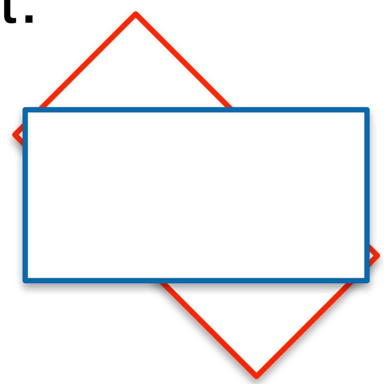


CD=2.05

No local minima for Min of N

Example of Local Minima for Chamfer Distance

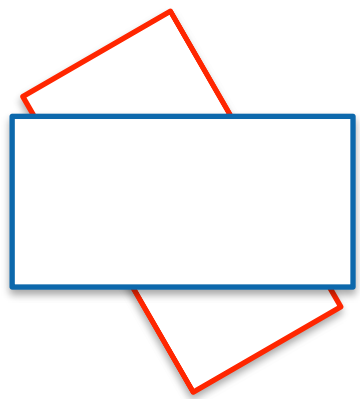
Reset, and try rotate right:



CD=2.02

No local minima for Min of N

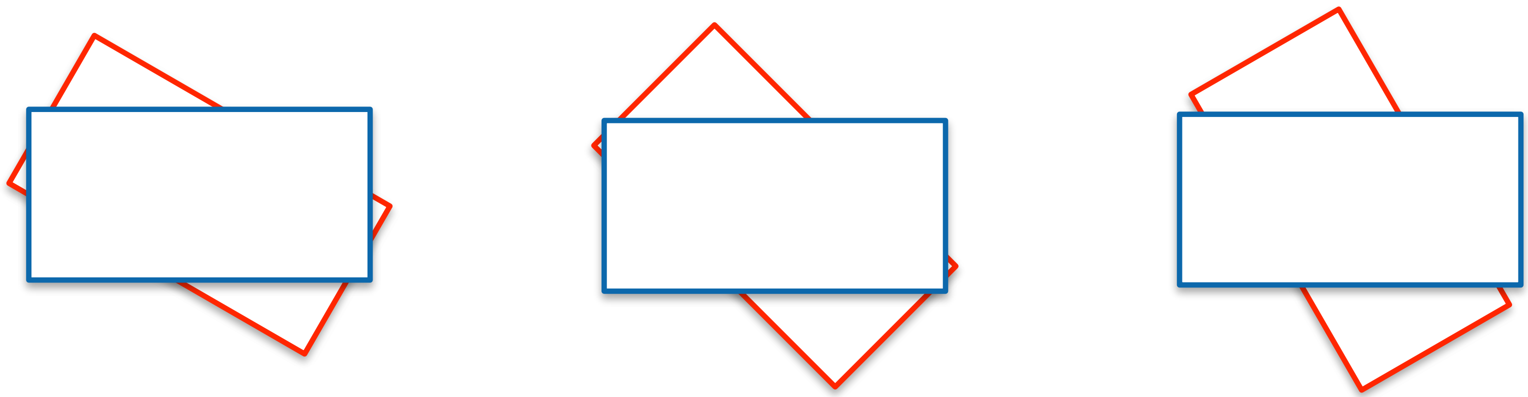
Example of Local Minima for Chamfer Distance



CD=2.50

No local minima for Min of N

Example of Local Minima for Chamfer Distance



CD=2.05 CD=2.02 CD=2.50

Note: No local minima for Min of N

Direct Approaches

Representation of rotation

Loss for rotation

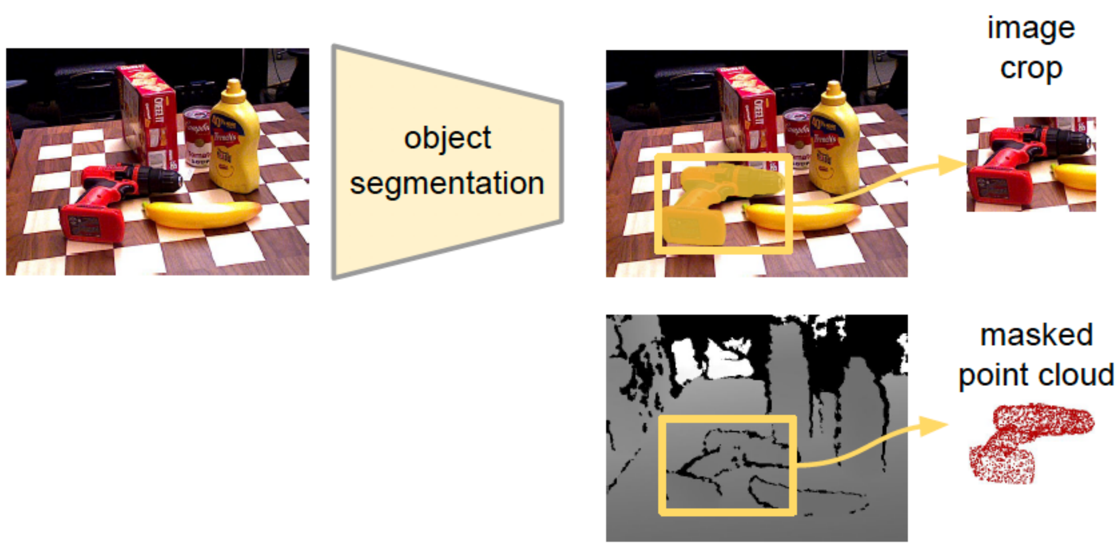
Example: DenseFusion

Example: DenseFusion

Segment objects

Wang et al., “DenseFusion: 6D Object Pose Estimation by Iterative Dense Fusion”, CVPR 2019

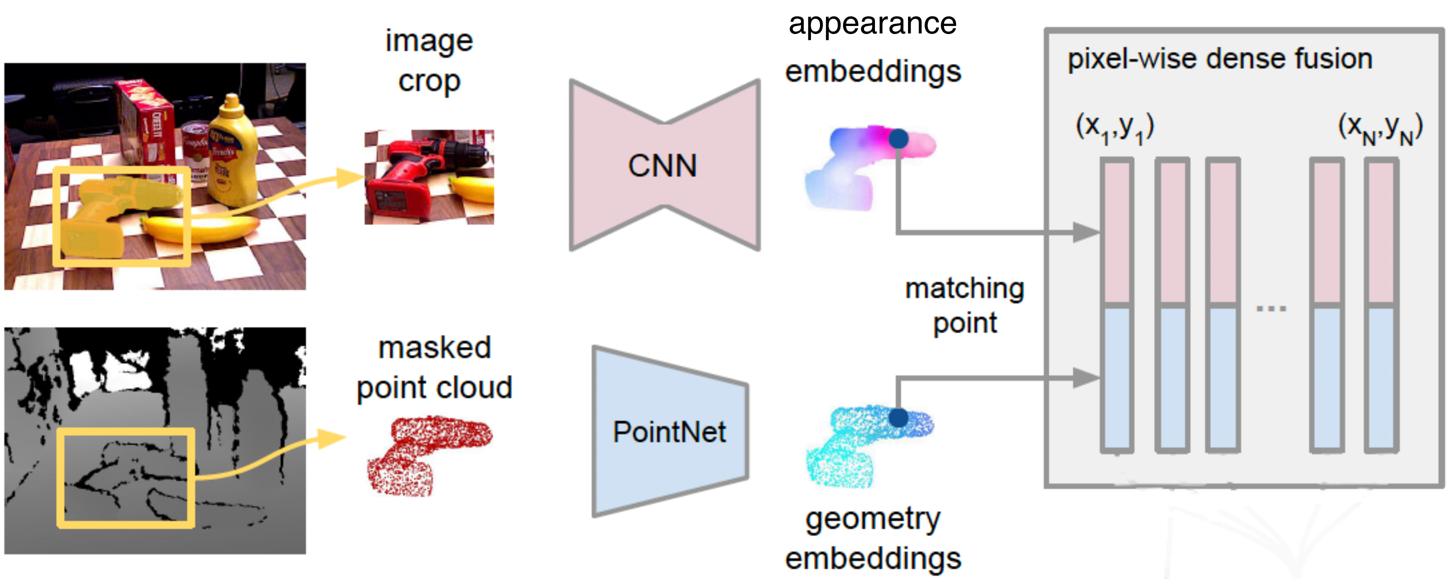
Example: DenseFusion



Get the image crop and the point cloud for an object

Wang et al., "DenseFusion: 6D Object Pose Estimation by Iterative Dense Fusion", CVPR 2019

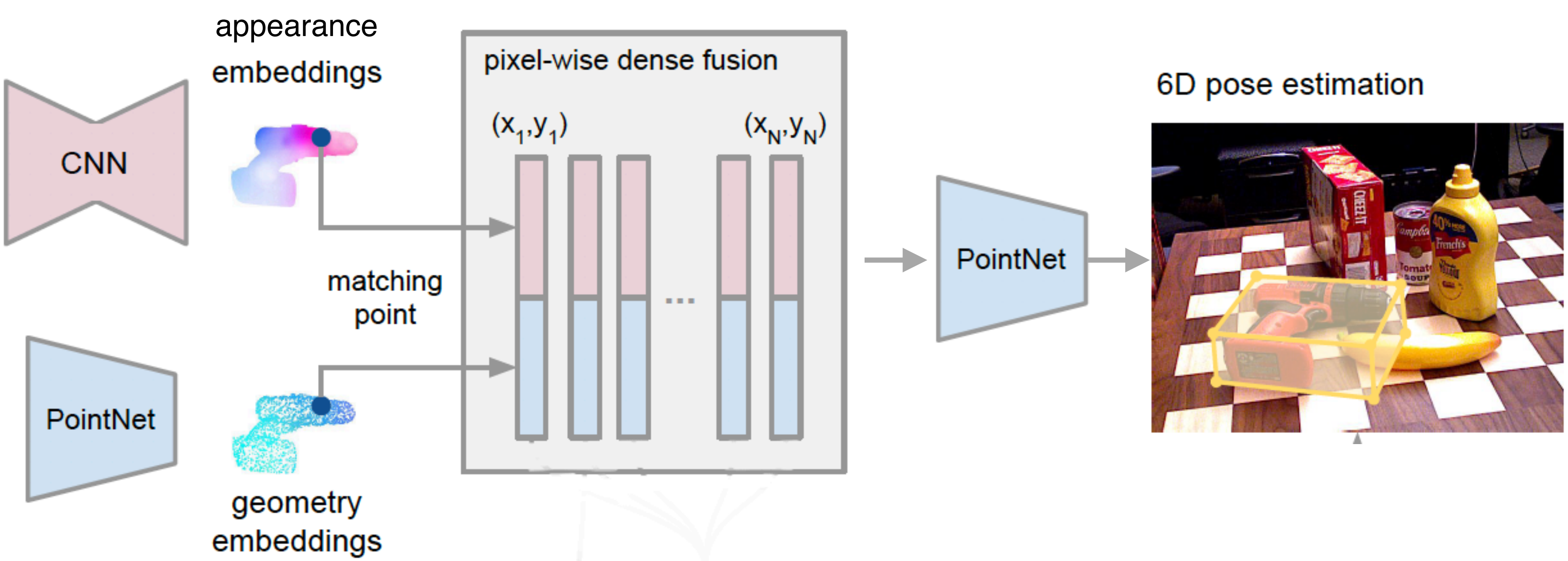
Example: DenseFusion



Encode appearance (image) with CNN
and geometry (point cloud) with PointNet

Wang et al., "DenseFusion: 6D Object Pose Estimation by Iterative Dense Fusion", CVPR 2019

Example: DenseFusion



Predict poses from features

Wang et al., "DenseFusion: 6D Object Pose Estimation by Iterative Dense Fusion", CVPR 2019

Example: DenseFusion

6D pose estimation

non-symmetric objects:
per-point MSE

symmetric objects:
chamfer distance

Min-of-N as alternative?

Compute shape-aware loss

Wang et al., “DenseFusion: 6D Object Pose Estimation by Iterative Dense Fusion”, CVPR 2019

Indirect Approaches

Indirect Approaches

- Input: cropped image/point cloud/depth map of a single object

- Output: corresponding pairs $\{(p_i, q_i)\}$
  - points in canonical frame $\{p_i\}$
  - points in camera frame $\{q_i\}$
  - estimate (R, t) by solving

$$\min_{R,t} \sum_{i=1}^{n} \|Rp_i + t - q_i\|^2$$

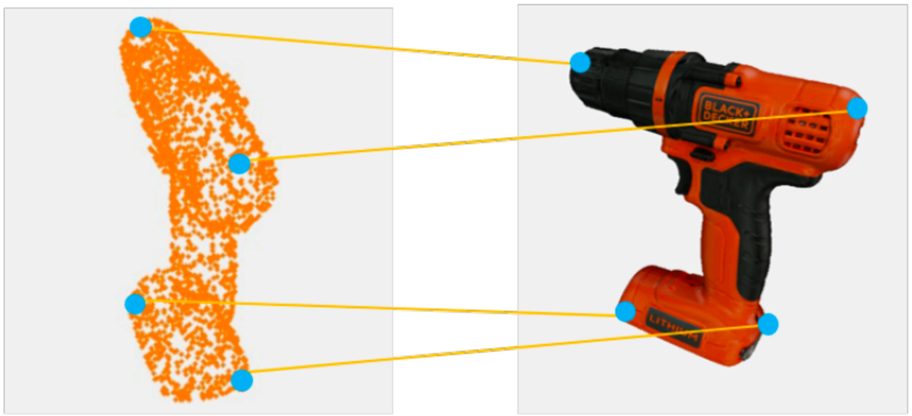
Two Categories of Indirect Approaches

- If points in the canonical frame are known, predict their corresponding locations in the camera frame

- If points in the camera frame are known, predict their corresponding locations in the canonical frame

Given Points in the Canonical Frame, Predict Corresponding Location in the Camera Frame

- Recall: Three correspondences are enough
- Which points in the canonical frame should be given?
  - Choice by PVN3D: keypoints in the canonical frame



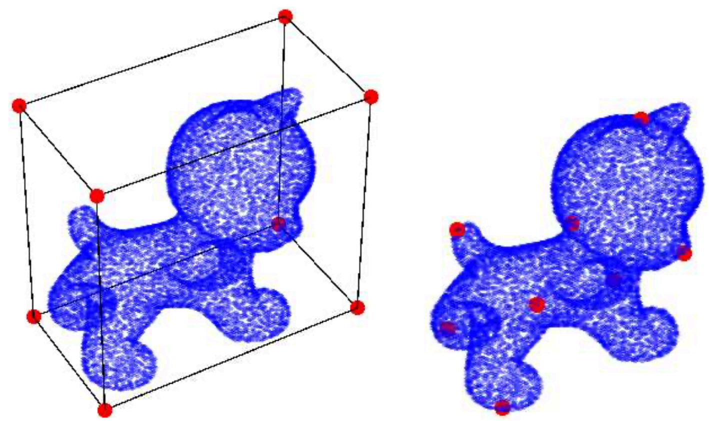
(d) 3D keypoints (cam.) (e) 3D keypoints (obj.)

He et al., "PVN3D: A Deep Point-wise 3D Keypoints Voting Network for 6DoF Pose Estimation", CVPR 2020

Keypoint Selection

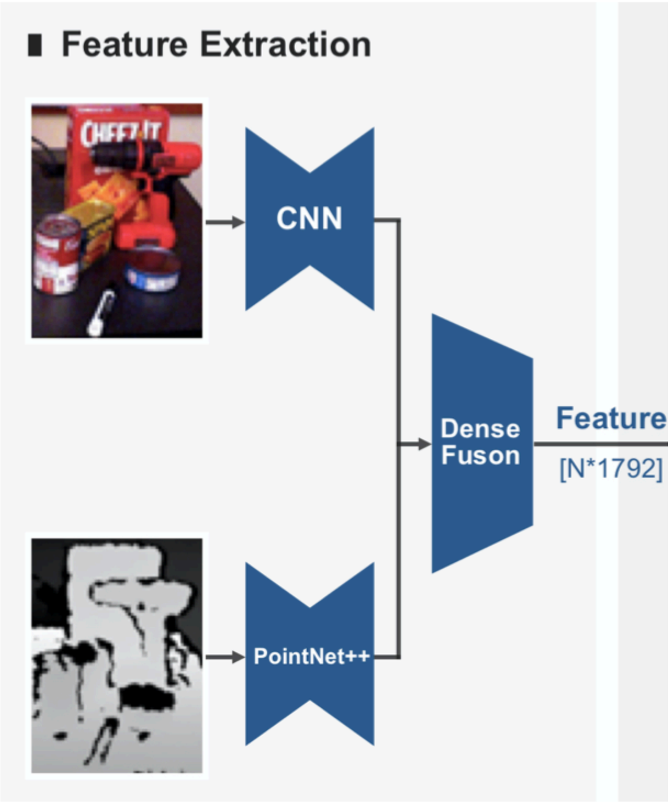
- Option1: bounding box vertices

- Option 2: farthest point sampling (FPS) over CAD object model



Chen et al., “G2L-Net: Global to Local Network for Real-time 6D Pose Estimation with Embedding Vector Features”, CVPR 2020

Example: PVN3D



Get point-wise features
by fusing color and geometry features

Example: PVN3D

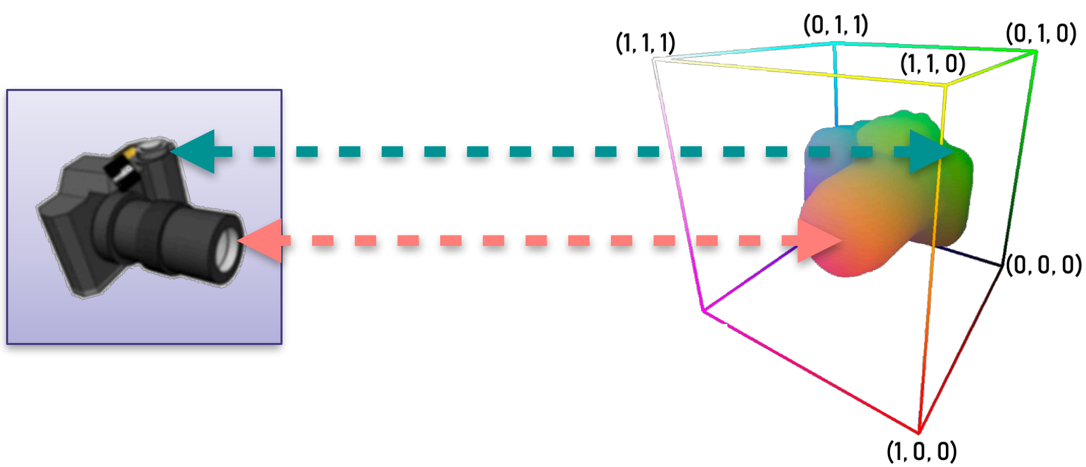
(c) Voting & clustering

For each keypoint:

- **Voting**: for each point in the camera frame, predict its offset to the keypoint (in the camera frame)
- **Clustering**: find one location according to all the candidates

Given Points in the Camera Frame, Predict Corresponding Location in the Canonical Frame

- Which points in the camera frame should be given?
  - Choice by NOCS: every point in the camera frame



3D point in the camera frame (2D visible pixel with depth)

3D point in the (normalized) canonical frame

Wang, He, et al. "Normalized object coordinate space for category-level 6d object pose and size estimation." CVPR 2019.

Example: NOCS

Output NOCS Map

$$H \times W \times 3$$

Input Image

NOCS Map

Note: the object is normalized to have unit diagonal of bounding box in the canonical space, so the canonical space is called “Normalized Object Canonical Space” (NOCS)

Wang, He, et al. "Normalized object coordinate space for category-level 6d object pose and size estimation." CVPR 2019.

Example: NOCS for Symmetric Objects

- Given equivalent GT rotations $\mathscr{R} = \{R_{GT}^1, R_{GT}^2, \cdots, R_{GT}^n\}$ (finite symmetry order n), we can generate n equivalent NOCS maps

- Similar to shape-agnostic loss in direct approaches, we can use *Min of N* loss

Wang, He, et al. "Normalized object coordinate space for category-level 6d object pose and size estimation." CVPR 2019.

Umeyama's Algorithm

- However, the target points in the canonical space of NOCS are normalized, and thus we also need to predict the scale factor

- Similarity transformation estimation (rigid transformation + uniform scale factor)

- Closed-form solution
  - Umeyama algorithm: http://web.stanford.edu/class/cs273/refs/umeyama.pdf
  - Similar to the counterpart without scale

Read by yourself

Tips for Homework 2

- For learning-based approaches
  - Start with direct approaches
  - Crop the point cloud of each object from GT depth map given GT segmentation mask
  - Train a neural network, e.g. PointNet, with shape-agnostic loss
  - Improve the results considering symmetry