CAPTRA: CAtegory-level Pose Tracking for Rigid and Articulated Objects from Point Clouds

Yijia Weng[1,2*] He Wang[3*†] Qiang Zhou [4] Yuzhe Qin[5] Yueqi Duan[3]
Qingnan Fan[3] Baoquan Chen[1,2] Hao Su[5] Leonidas J. Guibas[3]
[1]CFCS, Peking University [2]AICFVE, Beijing Film Academy
[3]Stanford University [4]Shandong University [5]UCSD

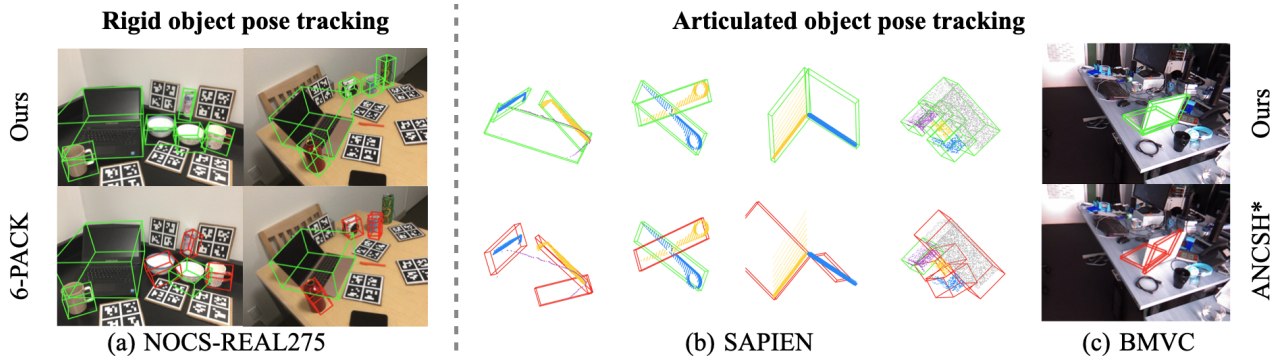

Figure 1. Our method tracks 9DoF category-level poses (3D rotation, 3D translation, and 3D size) of novel rigid objects as well as parts in articulated objects from live point cloud streams. We demonstrate: (a) our method can reliably track rigid object poses from the challenging NOCS-REAL275 dataset [29]; (b) our method can perfectly track articulated objects with big global and articulated motions from the SAPIEN datasets[34]; (c) trained only on SAPIEN, our model can directly generalize to real laptops from BMVC dataset[18]. In all cases, our method significantly outperforms the previous state-of-the-arts and baselines. Here we visualize the estimated 9DoF poses as 3D amodal bounding boxes: green boxes indicate in tracking whereas red boxes indicate off tracking.

Abstract

In this work, we tackle the problem of category-level online pose tracking of objects from point cloud sequences. For the first time, we propose a unified framework that can handle 9DoF pose tracking for novel rigid object instances as well as per-part pose tracking for articulated objects from known categories. Here the 9DoF pose, comprising 6D pose and 3D size, is equivalent to a 3D amodal bounding box representation with free 6D pose. Given the depth point cloud at the current frame and the estimated pose from the last frame, our novel end-to-end pipeline learns to accurately update the pose. Our pipeline is composed of three modules: 1) a pose canonicalization module that normalizes the pose of the input depth point cloud; 2) RotationNet*, a module that directly regresses small interframe delta rotations; and 3)* CoordinateNet*, a module that predicts the normalized coordinates and segmentation, enabling analytical computation of the 3D size and translation. Leveraging the small pose regime in the pose-canonicalized point clouds, our method integrates the best of both worlds by combining dense coordinate prediction and direct rotation regression, thus yielding an end-to-end differentiable pipeline optimized for 9DoF pose accuracy (without using non-differentiable RANSAC). Our extensive experiments demonstrate that our method achieves new state-of-the-art performance on category-level rigid object pose (NOCS-REAL275 [29]) and articulated object pose benchmarks (SAPIEN [34], BMVC [18]) at the fastest FPS $\sim$ 12.*

1. Introduction

Object pose estimation is crucial for a variety of computer vision and robotics applications, such as 3D scene understanding, robotic manipulation and augmented reality. The majority of object pose estimation works, *e.g.*, [35, 24], mainly lie in instance-level estimation, where the task is to estimate poses for objects from a small set of a priori known instances, thus preventing them from perceiving the poses of the vast diversity of objects in our daily life. To mitigate this limitation, Wang *et al.* [29] proposed to generalize the instance-level 6DoF (Degree of Freedom) object pose esti-

*: equal contributions, †: corresponding author
Project page: https://yijiaweng.github.io/CAPTRA

mation problem to a category-level 9DoF pose estimation problem that takes into account the traditional 6D object pose (rotation, translation) as well as 3D object size. The proposed method in [29] can handle novel object instances in known categories without requiring CAD models of the objects. Going beyond rigid objects and in the same spirit, Li *et al.* [14] proposed to estimate category-level per-part 9DoF poses for articulated objects, such as laptops, drawers and eyeglasses.

While most of the existing category-level pose estimation works focus on single-frame estimation, we believe that temporally smooth pose tracking is more useful for many robotics applications, *e.g.*, instant feedback control, as well as AR applications. In this work, we tackle a problem named CAPTRA — *CAtegory-level Pose Tracking for Rigid and Articulated Objects*, from a live point cloud stream. Given an initial object pose at the first frame, our task is to continuously track the 9DoF pose for rigid objects or each individual rigid part of an articulated object. The most related work to ours is 6-PACK [28], which tackles the problem of category-level 6D pose tracking only for rigid objects (see the related work section for detailed comparisons).

To accurately track 9DoF poses, we consider two types of approaches: coordinate-based approaches widely used in object pose [2, 29, 14] and camera pose estimation [4, 4] and direct pose regression as in [35, 32]. These two approaches both have pros and cons. Coordinate-based methods, which predict dense object coordinates followed by a RANSAC-based pose fitting, are generally more accurate and robust, especially on rotation estimation [27], benefiting from outlier removal in RANSAC. However, RANSAC-based pose fitting is non-differentiable and time-consuming, which often leads to a bottleneck in its running speed. In contrast, direct pose regression performs an end-to-end pose prediction, thus can achieve very high running speed, at the cost of being more error-prone.

In this work, we seek to take the best of both worlds and build **an end-to-end differentiable pipeline for accurate and fast pose tracking**. To enable highly accurate pose estimation, we propose to jointly canonicalize the input and output spaces of this estimation problem by transforming the point clouds using the inverse poses from the previous frame. The produced **pose-canonicalized point clouds** feature near identical poses regarding the object/part, whose poses are more regression-friendly. We thus propose **RotationNet**, a PointNet++ [23] based neural network, that directly regresses the small remained rotations. Due to the ambiguity between occlusion and center translation in the partial depth observations, we fouthe nd scale and translation regression still challenging. We instead propose to build **CoordinateNet** to predict dense normalized coordinates, which contain more accurate information about translation and object size due to their awareness of the category-level shape prior. Combining the outputs from both networks, we can then analytically compute the size and translation, yielding an end-to-end differentiable pipeline optimized for 9DoF pose accuracy without using non-differentiable RANSAC.

By harnessing both approaches, our proposed method gains significant performance improvement on the category-level rigid object pose benchmark and articulated object pose benchmarks. On the NOCS-REAL275 dataset [29], we outperform 6-PACK [28], the previous state-of-the-art, by 40.03% absolute improvement on the mean accuracy of $5°5$cm and 10.52% absolute improvement on the mean IoU metric. On the SAPIEN articulated object dataset [34], we are the first to perform tracking and outperform the single-frame articulated pose estimation baseline, constructed using ANCSH [14] and ground truth segmentation masks, by a large margin, *e.g.*, around 20 points on mean accuracy $5°5$cm in the challenging eyeglasses category. On novel real laptop trajectories from the BMVC dataset [18], we achieve the best performance directly generalized from SAPIEN without further fine-tuning. Finally, our extensive experiments further demonstrate the robustness of our tracking method to pose errors and achieve the fastest speed ($\sim$ 12 FPS) among all methods.

2. Related Works

Category-Level Object Pose Estimation To define category-level poses of novel object instances, Wang *et al.* [29] proposed Normalized Object Coordinate Space (NOCS) as a category-specific canonical reference frame for rigid objects. The objects from the same category in NOCS are consistently aligned to a category-level canonical orientation. These objects are further zero-centered and uniformly scaled so that their tight bounding boxes are centered at the origin of NOCS with a diagonal length of 1. Li *et al.* [14] extended the definition of NOCS to rigid parts in articulated objects and proposed Normalized Part Coordinate Space (NPCS), which is a part-level canonical reference frame(see appendix A for a detailed introduction). Several works have been improving [29] via leveraging analysis-by-synthesis and shape generative models as in [7, 6] and learnable deformation as in [26]. Most of these methods leverage RANSAC for pose fitting, which prohibits their pipelines from being end-to-end differentiable, potentially rendering those methods sub-optimal. Although several works have proposed differentiable RANSAC layers to bridge this gap, *e.g.*, DSAC [2], DSAC++ [3], we will show that our method performs better than these methods without using RANSAC.

Category-Level Object Pose Tracking As the only existing work in this field, Wang *et al.* [28] proposed a 6D Pose Anchor-based Category-level Keypoint tracker (6-PACK) by tracking keypoints in RGB-D videos. 6-PACK first em-

ploys an attention mechanism over anchors and then generates keypoints in an unsupervised manner, which are used to compute interframe pose changes. It is important to note several key differences between 6-PACK and our work: 1) 6-PACK is designed only for rigid objects and cannot handle articulated objects; 2) 6-PACK only estimates the 6D pose containing rotation and translation and omits the important 3D size estimation required to obtain the 3D amodal object bounding boxes.

As special cases of category-level articulated object pose tracking, hand and human pose tracking problems are very popular due to their broad applications [21, 31, 20, 12, 11, 36, 1]. However, the developed methods leverage domain-specific knowledge of hand and human body, thus prevent them from being applied to generic articulated objects.

Instance-Level 6D Pose Tracking Instance-level pose tracking works track the poses of known object instances. Classic methods, *e.g.*, ICP-based tracking [38], explicitly rely on the object CAD models. Some recent works [8, 9, 33, 13, 9] use particle filtering to estimate the posterior of object poses. Other methods measure the discrepancy between the current observation and the previous state, and perform tracking via optimization [25, 22]. The most relevant works to ours are delta pose based methods [15, 32], which perform tracking by regressing the pose change between consecutive frames using deep neural networks.

3. Problem Definition and Notations

In this paper, we target at the problem of tracking the 9DoF poses of rigid or articulated objects from known categories. We follow the category-level rigid object and part pose definition in [29, 14] and adopt the assumption in [14] that the number of rigid parts M is known and constant for all the objects in a known category, where $M > 1$ indicates an articulated object category, and $M = 1$ indicates a rigid object category. Without loss of generality, we only describe the notations for articulated object pose tracking. For a point cloud $X = \{x_i \in \mathbb{R}^3\}_{i=1}^{N}$ containing object instance $O = \{C^{(j)}\}_{j=1}^{M}$, where N is the number of points and $C^{(j)} \subset X$ represents points of the j-th rigid part, we denote category-level part pose as $\mathtt{P}^{(j)} = \{d^{(j)}, R^{(j)}, T^{(j)}\}$, where $d^{(j)} \in \mathbb{R}^3$ is 3D size, $R^{(j)} \in SO(3)$ is rotation, and $T^{(j)} \in \mathbb{R}^3$ is translation.

Our problem is then defined as follows: Given a live stream of depth point clouds $\{X_t\}_{t\geq 0}$ containing object instance O along with its per-part pose initialization $\{\mathtt{P}_0^{(j)}\}_j$, our objective is to track its part poses $\{\mathtt{P}_t^{(j)}\}_j$ in an online manner, where we process the point clouds and estimate the poses for all the frames $t > 0$. In other words, at frame $t+1$, given the estimated $\{\mathtt{P}_t^{(j)}\}_j$ from frame t and the depth point cloud X_{t+1}, our system needs to estimate $\{\mathtt{P}_{t+1}^{(j)}\}_j$.

4. End-to-end Differentiable Pose Tracking

In this section, we introduce our approach in detail. We present the pose canonicalization module in Section 4.1, and pose tracking in Section 4.2, which includes the proposed RotationNet module and CoordinateNet module, finally, we describe our training protocol in Section 4.3. The entire framework is differentiable and end-to-end trained, without any pre- or post- processing.

4.1. Pose Canonicalization

Inspired by [29], we factorize the 9DoF pose $\mathtt{P}^{(j)}$ prediction into a 7DoF similarity transformation $\mathcal{T}_t^{(j)} \in \mathrm{Sim}(3)$ estimation problem and a 3D aspect ratio $\hat{d}^{(j)}$ estimation problem. Formally, we define the per-part 1D uniform scale as $s^{(j)} = ||d^{(j)}||$ and 3D aspect ratio as $\hat{d^{(j)}} = d^{(j)}/s^{(j)}$ so that $d^{(j)} = s^{(j)}\hat{d^{(j)}}$. We can then represent $\mathcal{T}^{(j)} = \{s^{(j)}, R^{(j)}, T^{(j)}\}$.

To estimate $\mathcal{T}_{t+1}^{(j)}$ from the observed point cloud X_{t+1}, there are two types of approaches. One way is to train a neural network for direct pose regression. Another way is to estimate the normalized coordinates $Y_{t+1}^{(j)}$ of $C_{t+1}^{(j)}$, which satisfy $C_{t+1}^{(j)} = s_{t+1}^{(j)} R_{t+1}^{(j)} Y_{t+1}^{(j)} + T_{t+1}^{(j)}$, and then compute the $\mathcal{T}_{t+1}^{(j)}$ using the Umeyama algorithm [27] along with RANSAC, thus the 3D aspect ratios $\hat{d^{(j)}}$ can be estimated using the axis range $(|x|_{max}, |y|_{max}, |z|_{max})$ of $Y_{t+1}^{(j)}$.

In our framework, to simplify the learning problem of mapping the input X_{t+1} to the output $\mathcal{T}_{t+1}^{(j)}$, we propose to canonicalize both its input and output spaces using $\mathcal{T}_t^{(j)}$, which allows to further combine the two aforementioned methods.

Definition (Pose-canonicalized point cloud)**.** *Pose-canonicalized point cloud $Z_{t+1}^{(j)}$ with respect to part j in input point cloud X_{t+1} is defined as the product of the inverse transformation of $\mathcal{T}_t^{(j)}$ and X_{t+1}, namely* $Z_{t+1}^{(j)} = \left(R_t^{(j)}\right)^{-1}\left(X_{t+1} - T_t^{(j)}\right)/s_t^{(j)}$.

Theorem. *For the learning problem that maps X_{t+1} to $\mathcal{T}_{t+1}^{(j)}$, by canonicalizing its input space X_{t+1} to pose-canonicalized point cloud $Z_{t+1}^{(j)}$, its output space $\mathcal{T}_{t+1}^{(j)}$ will be canonicalized to $\hat{\mathcal{T}}_{t+1}^{(j)} = \{\hat{s}_{t+1}^{(j)}, \hat{R}_{t+1}^{(j)}, \hat{T}_{t+1}^{(j)}\}$ correspondingly, where $\hat{s}_{t+1}^{(j)} \approx 1, \hat{R}_{t+1}^{(j)} \approx I, \hat{T}_{t+1}^{(j)} \approx 0$.*

Proof. See appendix B. □

Note that $\mathcal{T}_{t+1}^{(j)}$ can be expressed using $\hat{\mathcal{T}}_{t+1}^{(j)}$ and $\mathcal{T}_t^{(j)}$, namely, $s_{t+1} = s_t^{(j)}\hat{s}_t^{(j)}$, $R_{t+1}^{(j)} = R_t^{(j)}\hat{R}_t^{(j)}$, $T_{t+1}^{(j)} = s_t^{(j)}R_t^{(j)}\hat{T}_t^{(j)} + T_t^{(j)}$. Now the pose estimation problem has been transformed and canonicalized to learning a mapping from $Z_{t+1}^{(j)}$ to $\hat{\mathcal{T}}_{t+1}^{(j)}$. The input $Z_{t+1}^{(j)}$ contains $\hat{C}_{t+1}^{(j)} = \left(R_t^{(j)}\right)^{-1}\left(C_{t+1}^{(j)} - T_t^{(j)}\right)/s_t^{(j)}$ that align well across different frames and the output space is quite constrained

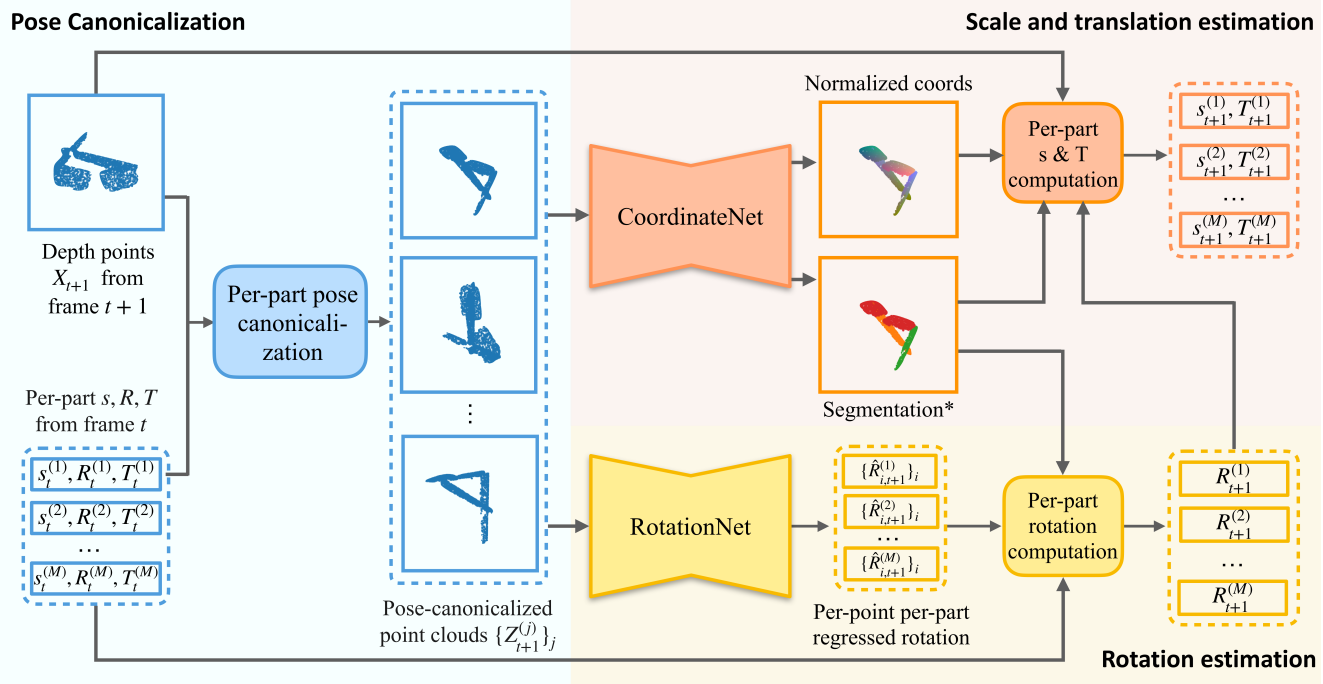

Figure 2. **Our end-to-end differentiable pose tracking pipeline** takes as inputs a depth point cloud of an M-part object along with its per-part scales, rotations, and translations estimated from the last frame. We first adopt per-part pose canonicalization to transform the depth points using the inverse estimated pose and generate M pose canonicalized point clouds. The canonicalized point clouds will be fed into RotationNet for per-part rotation estimation, as well as CoordinateNet for part segmentation and normalized coordinate predictions, which are used to compute the updated scales and translations. When RGB images are available, segmentation can be replaced by the results from the off-the-shelf image detectors for better accuracy. Such a pipeline can be naturally adapted to rigid objects when $M = 1$.

around an identical similarity transformation. In this way, we simultaneously canonicalize the input point cloud space and the output pose space. By doing so, we significantly simplify the regression task, yielding much improved pose estimation accuracy and better generalizability across different instances. Note that turning the estimation of $\mathcal{T}_{t+1}^{(j)}$ into $\hat{\mathcal{T}}_{t+1}^{(j)}$ is closely related to estimating the interframe delta 6D pose widely used in instance-level 6D pose estimation and tracking works [15, 32]. To be more specific, we estimate interframe 7D delta transformations in a category-level canonical frame, *i.e.*, NOCS for rigid objects and NPCS for parts, whereas none of the input and output spaces of delta pose estimation in [15, 32] is canonical.

4.2. Pose Tracking

Taking the pose-canonicalized point cloud $Z_{t+1}^{(j)}$ as input, we learn a RotationNet (see Section 4.2.1) that directly regresses $\hat{R}_{t+1}^{(j)}$ with high accuracy and then recovers $R_{t+1}^{(j)} = R_t^{(j)} \hat{R}_{t+1}^{(j)}$. However, we experimentally find directly regressing $\hat{s}_t^{(j)}$ and $\hat{T}_t^{(j)}$ still difficult, due to the translation ambiguity caused by incompleteness of the partial observations $Z_{t+1}^{(j)}$. Imagine a pencil with one end occluded, the length of the pencil cannot be determined, thus making prediction of its center translation highly ambiguous. Although certain cues, *e.g.*, object symmetry, may help relieve the ambiguity, our experiments show that given partial observations of asymmetric objects, regressing $\hat{T}_t^{(j)}$ still remains challenging. In contrast, our experiments reveal that predicting normalized coordinates $Y_{t+1}^{(j)}$ from $Z_{t+1}^{(j)}$ is quite successful on all rigid and articulated objects, which implicitly estimates $\hat{s}_t^{(j)}$ and $\hat{T}_t^{(j)}$. The reason for this success is that the normalized coordinates $Y_{t+1}^{(j)}$ capture the category-wise prior and enforces a zero-centered frame, thus making the translation estimation more well-considered and accurate than direct regression. We therefore devise a CoordinateNet to segment $C_{t+1}^{(j)}$ from $X_{t+1}^{(j)}$ and predict $Y_{t+1}^{(j)}$ (see Section 4.2.2).

By combining the RotationNet and CoordinateNet's outputs and knowing $C_{t+1}^{(j)} = s_{t+1}^{(j)} R_{t+1}^{(j)} Y_{t+1}^{(j)} + T_{t+1}^{(j)}$, we can analytically compute $s_{t+1}^{(j)}$ and $T_{t+1}^{(j)}$ via the Umeyama algorithm [27] (assume R is given). Usually a non-differentiable RANSAC is needed when using Umeyama algorithm as in [29, 14] due to the multi-modal noises in the predicted $Y_{t+1}^{(j)}$. Thanks to the pose canonicalization, we find that our $Y_{t+1}^{(j)}$ predictions are very successful and RANSAC only brings limited improvement to our predictions (see Sec. 5.5).

Being free from the non-differentiable RANSAC step,

our end-to-end differentiable pose tracking pipeline can be straightly optimized for pose accuracy, enforce pose losses (*e.g.*, IoU loss) directly at its outputs (see Section 4.3), and improve its running speed.

4.2.1 Rotation Estimation

RotationNet To predict $\{\hat{R}^{(j)}_{t+1}\}_j$ for each individual part, we devise a point cloud based deep neural network, RotationNet, that takes as inputs the pose-canonicalized point clouds $\{Z^{(j)}_{t+1}\}_j$ with respect to each individual part j. Built upon PointNet++ [23], RotationNet regresses per-point per-part rotations $\{\hat{R}^{(j)}_{i,t}\}_{i,j}$ in the form of the 6D continuous rotation representation [39]. Note that the PointNet++ is not deterministic since it uses random further point sampling in both set abstraction and ball query operations, thus resulting in difficulties achieving convergence on accurate regression tasks. To suppress noise, we average across the rotation predictions using the Euclidean mean [19] from points on part j to obtain the final prediction $\hat{R}^{(j)}_{t+1}$.

Rotation Supervision for Symmetric Objects Unseen instances from symmetric object categories, like bowls or bottles contain a rotation ambiguity around their symmetric axis $\hat{q}$ [29]. Due to this rotation ambiguity, only two degrees of freedom in the rotation are unique and should be supervised. We propose to regress the 3D end-point position $p_{\hat{R}}$ of its unit rotation axis $\hat{q}$. Similar to [39], the redundancy in the representation renders a continuous and regression-friendly rotation representation for symmetric objects. On the other hand, articulated objects rarely have rotational ambiguities for their rigid parts, since their kinematic structures usually help to disambiguate. Therefore, we only use the symmetric rotation representation for the bowl, bottle, and can categories in the NOCS-REAL275 dataset [29].

Training and Inference At training time, we enforce a per-point mean square loss for points inside the ground truth mask $m^{(j)}_{t+1}$. At test time, the mask comes from the predicted part segmentation from CoordinateNet.

4.2.2 Scale and translation estimation

CoordinateNet To estimate $\{Y^{(j)}_{t+1}\}_j$, we devise CoordinateNet that takes as input the pose-canonicalized point cloud $Z^{(1)}_{t+1}$ with respect to the first part ($j = 1$) and predicts its per-point part segmentation and per-point per-part normalized coordinates $\{Y^{(j)}_{i,t+1}\}_{i,j}$. Note that pose-canonicalized point clouds with respect to different parts share the same segmentation and normalized coordinates; thus, we only need to take $Z^{(1)}_{t+1}$ as the input.

Built upon a PointNet++ segmentation network, CoordinateNet branches into two heads after the final feature propagation layers: one head for segmentation and the other for normalized coordinate prediction. For the segmentation head, we use relaxed IoU loss [37]. For the normalized coordinate head, we predict class-aware normalized coordinates, similar to [29, 14]. During training, we enforce an RMSE loss on the points inside the ground truth part masks. At test time, we use predicted masks to select coordinate predictions from M parts.

Coordinate Supervision for Symmetric Objects For a symmetric object, *e.g.*, a bowl, its normalized coordinates contain ambiguities: one can freely rotate them together along its symmetric axis. Note that point pairwise distances are invariant under the rotation as are their y and $\sqrt{x^2+z^2}$ values (y is the symmetric axis). To supervise coordinate predictions for symmetric objects, we propose to jointly enforce an L2 loss on the pairwise distance matrix and a symmetric coordinate loss $\sqrt{|x^2+z^2-\hat{x}^2-\hat{z}^2|+(y-\hat{y}^2)}$ on the normalized coordinates.

Per-part Scale and Translation Computing With knowing the dense correspondence between $Y^{(j)}_{t+1}$ and $C^{(j)}_{t+1}$ and assuming $R^{(j)}_{t+1}$ is given by RotationNet, we can analytically compute $s^{(j)}_{t+1}$ and $T^{(j)}_{t+1}$ via the Umeyama algorithm [27]. See appendix C.2 for further detail and how we handle symmetric objects.

4.3. Training Protocol

Training Data Generation To train CoordinateNet and RotationNet, we need paired data of pose-canonicalized point clouds and their corresponding ground truth poses. We propose to generate the training data on-the-fly without using any real temporal data. For a depth point cloud X and part j in it, we perturb its per-part ground truth scale, rotation, and translation by adding random Gaussian noise to them, namely $s'^{(j)} = s^{(j)}(1+n_s)$, $R'^{(j)} = R^{(j)}R_{\text{rand}}$, $T'^{(j)} = T^{(j)} + n_T$, where $n_s \sim \mathcal{N}(0, \sigma_s)$, R_{rand} is a rotation matrix with a random axis and a random angle $n_\theta \sim \mathcal{N}(0, \sigma_\theta)$, and n_T is a 3D vector with a random direction and a random length $t \sim \mathcal{N}(0, \sigma_T)$. We then generate the pose-canonicalized point clouds and compute their corresponding ground truth.

Pose Losses for RotationNet and CoordinateNet For $s^{(j)}_{t+1}$, $R^{(j)}_{t+1}$, $T^{(j)}_{t+1}$, their predictions are end-to-end differentiable; we thus propose to additionally enforce pose losses directly on these predictions. We use RMSE loss for supervising scale error L_{scale} and translation error L_{trans}. To directly improve the final 3D IoU predictions, we incorporate a corner loss L_{corner} [16], defined as the corresponding per-vertex distances between the ground truth bounding box in the camera frame and the ground truth bounding box in the normalized coordinate space transformed by our predicted $s^{(j)}_{t+1}$, $R^{(j)}_{t+1}$, $T^{(j)}_{t+1}$. For symmetric objects, we enforce the corner loss on the two intersection points of the symmetric axis and the bounding box surface. The total loss $L_{\text{total}} = \lambda_{\text{seg}}L_{\text{seg}} + \lambda_{\text{coord}}L_{\text{coord}} + \lambda_{\text{rot}}L_{\text{rot}} + \lambda_{\text{scale}}L_{\text{scale}} + \lambda_{\text{translation}}L_{\text{translation}} + \lambda_{\text{corner}}L_{\text{corner}}$.

5. Experiment

5.1. Datasets and Evaluation Metrics

NOCS-REAL275 For rigid object pose tracking, we evaluate our methods on the NOCS dataset [29] that contains six categories: bottle, bowl, camera, can, laptop, and mug, where bottle, bowl, and can are symmetric. The training set contains: 1) the train split of the CAMERA dataset [29], composed of 300K mixed reality data with synthetic object models from ShapeNetCore [5] as foregrounds and real backgrounds captured in IKEA; and 2) seven real videos capturing challenging cluttered scenes composed of rigid objects and containing three object instances in total for each object category. The testing set, NOCS-REAL275, has six real videos depicting in total three different (unseen) instances for each object category totaling 3200 frames.

Articulated Objects from SAPIEN For evaluating articulated object pose tracking, we create a synthetic dataset using SAPIEN [34]. Our dataset contains 183 object instances from four categories: glasses, scissors, laptop, and drawers, where drawers have prismatic joints and the others have revolute joints. The training set contains 98K depth images of 164 standalone object instances with random joint states and viewpoints. The testing set contains 190 depth videos of 19 unseen instances with a total length of 19K frames, in which unseen articulated instances keep moving and changing their joint states. See appendix E for more information.

Real-World Laptop Test Trajectories from the BMVC dataset [18] We also test our model on the real articulated object trajectories, where the objects maintain the same joint state and only viewpoint changes. The dataset only contains 4 instances: a laptop, a cupboard, a toy train, and a cabinet. We can only evaluate our method on the laptop for which we have category-level synthetic training data from SAPIEN. Its two laptop depth sequences contain a total of 1765 frames.

Evaluation Metrics We report the following metrics for both rigid and articulated object pose tracking: 1) **5°5cm accuracy**, the percentage of pose predictions with rotation error $< 5^\circ$ and translation error $<$ 5cm; 2) **mIoU**, the average 3D intersection over union (3D IoU) of ground-truth and predicted bounding boxes; 3) $\boldsymbol{R_{err}}$, average rotation error in degrees; and 4) $\boldsymbol{T_{err}}$, average translation error in centimeters. For articulated objects, we additionally report the average joint state accuracy: 5) $\boldsymbol{\theta_{err}}$ rotation error in degrees for revolute joints; and 6) $\boldsymbol{d_{err}}$ translation error in centimeters for prismatic joints. For real-world laptop trajectories, we follow the original paper [18] and use pose tolerance, namely the Averaged Distance (AD) accuracy with 10% of the object part diameter as the threshold.

5.2. Category-Level Rigid Object Pose Tracking

Experiment Setting To track an object in the cluttered scenes from the NOCS-REAL275 dataset, we propose to first crop from the scene a ball of depth points enclosing the object of interest. We set the center and the radius of the ball according to the previous frame's 9DoF pose estimation. To generate training data, we jitter the ground-truth pose with Gaussian noises ($\sigma_{scale} = 0.02$, $\sigma_{rot} = 5^\circ$, and $\sigma_{trans} = 3$cm) to mimic interframe pose changes and crop balls accordingly. At test time, we randomly sample an initial pose around the ground-truth pose for the first frame from the same distribution.

Results Table 1 summarizes the quantitative results for rigid object pose tracking. We report the performance of our method using only depth and using RGBD where object segmentation masks can be obtained by running off-the-shelf detectors on RGB images as in CASS[6]. We compare our method with: 6-PACK [28], a tracking based method that is initialized with the same pose error distributions or ground-truth poses (6-PACK originally only initializes with translation errors); Oracle ICP, which leverages the ground truth object models; and several single-frame based method, including NOCS [29], CASS [6] and CPS++ [17].

Our method achieves the best performance and significantly outperforms the previous state-of-the-art method, 6-PACK, under both initialization settings. We are especially competitive under the rotation error and 5°5cm metrics, showing less than a third of the rotation error and a 105% higher 5°5cm percentage compared to 6-PACK. Using only depth, our method generates relatively weaker performance regarding translation error, however, this is only due to the failure to segment out cameras on the real test depth images. The three test cameras are either too reflective or too dark (being completely black), thus yielding very noisy depth point clouds and making object segmentation difficult. Excluding this camera category, our method will be the best under all metrics (see appendix F.1). It is worth noting that while our method tracks the full 9DoF pose and predicts the bounding boxes, 6-PACK only tracks the 6DoF rigid transformation and has to use a ground-truth box scale to compute 3D IoU, which unfairly advantages 6-PACK during the comparison. Figure 1 further shows the qualitative comparison between our method and 6-PACK. Our method loses track less often and gives better pose estimations.

5.3. Category-Level Articulated Pose Tracking

Results on SAPIEN In Table 2 and Figure 1, we show the results of our articulated pose tracking on the held-out test instances from SAPIEN datasets. We compare our method to 1) ANCSH* (oracle ANCSH), where we provide ground-truth object segmentation masks to the state-of-the-art single frame articulated object pose estimation work, ANCSH [14]. The original ANCSH fails drastically on part segmentation on our dataset due to the part ambiguity of textureless object depth point clouds rendered from arbitrary viewpoints; and 2) oracle ICP, where we assume available

Method	NOCS[29]	CASS[6]	CPS++[17]	Oracle ICP	6-PACK[28]	6-PACK [28]	Ours	Ours+RGB seg.
Input	RGBD	RGBD	RGB	Depth	RGBD	RGBD	Depth	RGBD
Setting	Single frame			Tracking				
Initialization	N/A	N/A	N/A	GT.	GT.	Pert.	Pert.	Pert.
5°5cm ↑	16.97	29.44	2.24	0.65	28.92	22.13	62.16	**63.60**
mIoU↑	55.15	55.98	30.02	14.69	55.42	53.58	64.10	**69.19**
R_{err} ↓	20.18	14.17	25.32	40.28	19.33	19.66	**5.94**	6.43
T_{err} ↓	4.85	12.07	21.62	7.71	**3.31**	3.62	7.92	4.18

Table 1. **Results of category-level rigid object pose tracking on NOCS-REAL275**. The results are averaged over all 6 categories.

Method	5° 5cm↑	mIoU↑	R_{err} ↓	T_{err} ↓	θ_{err} ↓	d_{err} ↓
ANCSH* [14]	92.55	68.69	2.18	0.48	1.62	0.64
Oracle ICP	62.87	56.61	8.95	3.04	7.21	1.05
Ours	**98.35**	**74.00**	**1.03**	**0.29**	**1.38**	**0.34**
C-sRT regression	21.69	34.21	20.48	11.46	6.08	7.57
C-CoordinateNet	95.06	71.99	2.09	0.40	1.52	0.75
C-Crd. + DSAC++ [3]	95.68	68.21	1.80	0.47	1.61	0.56
Ours w/o L_c, L_s, L_t	97.63	72.09	1.24	0.35	1.43	0.36
Ours + Rot. Proj.	98.74	74.17	0.97	0.29	1.37	0.34

Table 2. **Experiment results and ablation studies of articulated object pose tracking on the held-out instances from SAPIEN.** θ_{err} is averaged over all revolute joints of glasses, scissors, laptops, while d_{err} is averaged over all prismatic joints of drawers. Other results are averaged over parts and categories. See appendix F.2 for per-part, per-category results. Ours + Rot. Proj leverages kinematic constraints, see Section 5.7.

Method		Michel et al.	ANCSH	ANCSH*	Ours
Setting		Known instance		Category-level	
1	all	64.8	94.1	74.7	**95.5**
	parts	65.5 66.9	97.5 94.7	89.1 78.5	**99.8 95.7**
2	all	65.7	98.4	97.0	**98.9**
	parts	66.3 66.6	98.9 **99.0**	98.0 97.6	**100.0** 98.9

Table 3. **Results on two real sequences of an unseen laptop** are measured in pose tolerance (the higher, the better, see [18]). The left two columns reported by [18, 14] are directly trained on the instance, whereas ANCSH*(with GT part mask) and ours are only trained on SAPIEN and have never seen the instance.

ground-truth part labels and object part models, then track each part using ICP.

Note that our articulated SAPIEN dataset is depth-only while RGB-D input is essential for 6-PACK, we thus did not run per-part 6-PACK tracking as a baseline.

We outperform the baselines under all metrics. Although ANCSH* uses ground-truth labels and regulates its predictions with joint constraints, our per-part scheme still beats it with exceptionally precise rotation estimations.

Testing on Real-World Laptop Trajectories In Table 3 and Figure 1, we compare our method to **Michel et al.** [18], ANCSH [14], and ANCSH* on two real-world laptop trajectories from [18]. We follow [14] and use their rendered object masks for test sequences. While Michel et al. and ANCSH are trained on the exact same instance used in testing, we directly test ANCSH* and our model trained on the synthetic dataset rendered from SAPIEN objects. Despite facing a Sim2Real gap and a category-level generalization gap, our model can outperform all other methods.

Method	CrdNet	C-Crd.	C-Crd.+ DSAC++	C-sRT	Ours w/o L_c, L_s, L_t	Ours
5°5cm ↑	14.93	46.74	54.77	25.99	60.48	**62.16**
mIoU↑	49.48	59.99	53.89	32.86	58.80	**64.10**
R_{err} ↓	53.63	35.08	8.88	34.74	6.41	**5.94**
T_{err} ↓	9.48	12.97	9.95	21.84	12.64	**7.92**

Table 4. **Ablation study of rigid object pose tracking on NOCS-REAL275.** The results are averaged over all 6 categories. Here C represents canonicalized.

5.4. Ablation Study

To demonstrate the effectiveness of our multi-component design, we construct several variants of our network: 1) CoordinateNet, where we directly regress the NOCS/NPCS coordinates from X without pose canonicalization; 2) canonicalized CoordinateNet (C in Table 4 represents canonicalized), where we perform pose canonicalization but don't have RotationNet; 3) canonicalized CoordinateNet with DSAC++, where we follow [3] and train our CoordinateNet with a differentiable pose estimation module; 4) canonicalized sRT regression, where we extend our RotationNet to further regress scale $\hat{s}_{t+1}^{(j)}$ and translation $\hat{T}_{t+1}^{(j)}$ based on canonicalized point clouds without using CoordinateNet; and 5) Ours w/o L_c, L_s, L_t losses, where we discard the pose losses $L_{scale}, L_{trans}, L_{corner}$ during training. For 1), 2), and 3) we take the coordinate predictions from CoordinateNet and use RANSAC-based pose fitting.

We test the variants on NOCS-REAL275 for rigid object tracking and SAPIEN synthetic dataset for articulated object tracking. The results are summarized in Table 4 and Table 2, where our method outperforms all ablated versions by successfully combining the benefits of pose canonicalization, coordinate prediction, and pose regression. Note that we did not test CoordinateNet without canonicalization on articulated objects due to the part ambiguity of uncolored, arbitrarily posed synthetic objects.

Canonicalized CoordinateNet significantly outperforms CoordinateNet, demonstrating the benefit brought by pose canonicalization. With additional RotationNet, our method further improves canonicalized CoordinateNet and beats the differentiable pipeline CoordinateNet + DSAC++ which also includes explicit pose losses, proving direct regression of small $\hat{R}_{t+1}^{(j)}$ to be a better choice in the tracking scenario. In contrast, due to ambiguities and insufficient visual cues about scale and translation in the input, the pure regression

Dataset	Metric	Orig.	Init.×1	Init.×2	All×1	All×2
Rigid	5°5cm	62.16	59.64	55.94	59.83	58.69
	mIoU	64.10	61.40	57.56	61.30	60.40
	R_{err}	5.94	5.95	5.93	5.81	5.89
	T_{err}	7.92	10.23	10.78	9.82	13.08
Arti.	5°5cm	98.35	98.40	97.75	98.45	97.68
	mIoU	74.00	74.00	73.68	74.05	75.53
	R_{err}	1.03	1.03	1.18	1.01	1.39
	T_{err}	0.29	0.29	0.32	0.29	0.39

Table 5. **Robustness to pose errors.** Init.×m means adding m times train-time errors in pose initialization, on top of the 1× train-time error already used in our original setting (denoted Orig.), and All×m means adding m times the errors to all estimated poses.

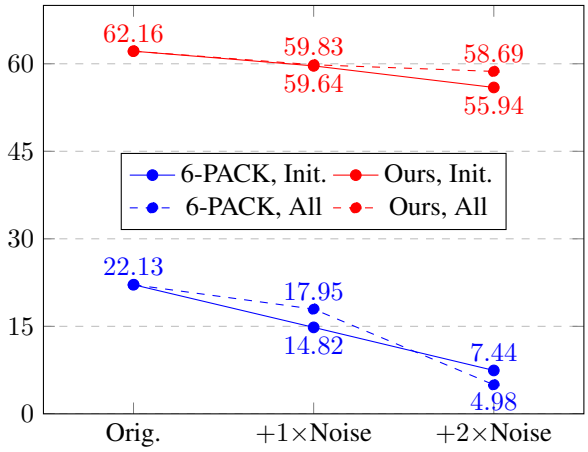

Figure 3. **5°5cm (%) w.r.t. Additional Noise.** +m×Noise means adding m times train-time errors to 1) the initial pose (denoted **Init.**), which already contains 1× train-time error; or 2) every frame during training (denoted **All**).

pipeline, canonicalzied sRT regression, produces the worst results. Finally, explicit scale, translation, and corner losses effectively improve our performance compared to ours w/o L_c, L_s, L_t losses.

5.5. Tracking Robustness

Robustness to Pose Errors Our pose prediction is conditioned on the pose from the previous frame, either the initial pose or an estimated pose. It is therefore worth testing the tracking robustness of our method against noisy pose inputs. As described in Sec. 5.2 and Sec. 5.3, the initial pose errors are randomly sampled from Gaussian distributions. As shown in Table 5, we directly test our model under the following settings: (1) increasing the intial pose error by 1 or 2 times, denoted as Init.×1 and Init.×2, to examine the robustness to pose initialization; and (2) adding 1 or 2 times pose error to every previous frame's prediction, denoted as All.×1 and All.×2, to examine the robustness to pose estimation errors. For rigid objects, we test 6-PACK under the same settings and plot the degradation of 5°5cm accuracy under each setting in figure 3. Our method is significantly more robust to pose noises than 6-PACK. For articulated objects, our method is very robust with less than 1 point drop on both 5°5cm and mIoU metrics.

Method	NOCS	6-PACK	ANCSH	Ours
FPS	4.05	3.53	0.80	**12.66**

Table 6. **Tracking speed in FPS**

Why no RANSAC? Most coordinate-based pose estimation approaches heavily rely on RANSAC during pose fitting, because rotations estimation done by orthogonal Procrustes is very sensitive to outliers. In our pipeline, the pose canonicalization significantly simplifies the rotation regression and reduce the noise in the coordinate prediction, thus freeing us from the need to use RANSAC. Our experiment shows that incorporating RANSAC to our scale and translation computation only bring very little improvements, i.e., increasing 5°5cm accuracy and mIoU by 0.86% and 1.85% respectively for rigid objects from NOCS-REAL275, 0.06% and 0.06% respectively for articulated objects from SAPIEN. In contrast, when we remove RANSAC, C-CoordinateNet drops 3.22% on 5°5cm accuracy and 4.37% on mIoU, due to the aforementioned rotation sensititity.

5.6. Tracking speed

Aside from the state-of-the-art performance, our method also has the highest tracking speed among all others, as summarized in Table 6. All methods are tested on the same device. NOCS and ANCSH are slow due to RANSAC and optimizations, which we don't need. We found that 6-PACK's actual speed is much slower than what they claim in their paper (>10 FPS) since their network used in their officially released code [30] is forwarded 27 times at a grid of potential object centers at each frame to achieve their reported performance.

5.7. Discussions

Tracking Scale Although the actual scale of the object is constant during tracking, we still track the scale in our framework to deal with inaccurate initial scale. Compared to fixing the scale as the noisy initial scale throughout the tracking, our scale tracking scheme decreases the average scale error from 1.09% to 0.30% and increases mIoU from 71.70% to 74.00% on articulated objects; and improve mIoU from 73.43% to 76.42% on rigid objects (excluding camera where both schemes fail).

Leveraging Kinematic Chain Constraints For articulated objects, our method focuses on per-part tracking without explicitly leveraging joint constraints at test time, though the constraints are implicitly enforced in training losses. Prior works leverage these constraints in instance-level tracking [18, 10] and category-level tracking [14]. [18] and [10] assume perfect knowledge of joint parameters (axis orientation and pivot point positions) and treat them as a hard

constraint. In the category-level setting, however, joint parameters are unknown and difficult to predict due to occlusions, especially for pivot point predictions. Empirically, we find even 1cm prediction error in pivot point predictions will make these constraints harmful. ANCSH [14] offers an alternative where energy constructed using only estimated joint axis orientations is incorporated into pose fitting as soft constraints for per-part rotation predictions at the cost of slower speed. Note that without leveraging the constraints, our method already significantly outperforms ANCSH [14] under all settings. Without sacrificing speed, we examine the usage of ground truth joint axis orientations as hard constraints but only gain little improvement as shown in Table 2 (Ours + Rot. Proj.). We will leave leveraging kinematic chain constraints to future works.

6. Conclusion

In this paper, for the first time, we tackle the problem of category-level 9DoF pose tracking for both rigid and articulated objects. To achieve this goal, we propose an end-to-end differentiable pose tracking framework consisting of three modules: pose canonicalization, RotationNet, and CoordinateNet. Our algorithm achieves state-of-the-art performance on both category-level rigid and articulated pose benchmarks and runs comparably fast for evaluation.

Acknowledgement: This research is supported by a grant from the SAIL-Toyota Center for AI Research, a grant from the Samsung GRO program, NSF grant IIS-1763268, a Vannevar Bush Faculty fellowship, the support of the Stanford UGVR program, and gifts from Kwai and Qualcomm.

References

[1] Mykhaylo Andriluka, Umar Iqbal, Eldar Insafutdinov, Leonid Pishchulin, Anton Milan, Juergen Gall, and Bernt Schiele. Posetrack: A benchmark for human pose estimation and tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5167–5176, 2018. 3

[2] Eric Brachmann, Alexander Krull, Sebastian Nowozin, Jamie Shotton, Frank Michel, Stefan Gumhold, and Carsten Rother. Dsac-differentiable RANSAC for camera localization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6684–6692, 2017. 2

[3] Eric Brachmann and Carsten Rother. Learning less is more-6D camera localization via 3D surface regression. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4654–4662, 2018. 2, 7

[4] Tommaso Cavallari, Stuart Golodetz, Nicholas A Lord, Julien Valentin, Luigi Di Stefano, and Philip HS Torr. On-the-fly adaptation of regression forests for online camera relocalisation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4457–4466, 2017. 2

[5] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015. 6

[6] Dengsheng Chen, Jun Li, Zheng Wang, and Kai Xu. Learning canonical shape space for category-level 6D object pose and size estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 11973–11982, 2020. 2, 6, 7

[7] Xu Chen, Zijian Dong, Jie Song, Andreas Geiger, and Otmar Hilliges. Category level object pose estimation via neural analysis-by-synthesis. In *European Conference on Computer Vision*, pages 139–156. Springer, 2020. 2

[8] Changhyun Choi and Henrik I Christensen. RGB-D object tracking: A particle filter approach on GPU. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1084–1091. IEEE, 2013. 3

[9] Xinke Deng, Arsalan Mousavian, Yu Xiang, Fei Xia, Timothy Bretl, and Dieter Fox. PoseRBPF: A rao-blackwellized particle filter for 6D object pose tracking. *Robotics: Science and Systems*, 2019. 3

[10] Karthik Desingh, Shiyang Lu, Anthony Opipari, and Odest Chadwicke Jenkins. Factored pose estimation of articulated objects using efficient nonparametric belief propagation. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 7221–7227. IEEE, 2019. 8

[11] Varun Ganapathi, Christian Plagemann, Daphne Koller, and Sebastian Thrun. Real-time human pose tracking from range data. In *Proceedings of the European Conference on Computer Vision*, pages 738–751, 2012. 3

[12] Shangchen Han, Beibei Liu, Randi Cabezas, Christopher D Twigg, Peizhao Zhang, Jeff Petkau, Tsz-Ho Yu, Chun-Jung Tai, Muzaffer Akbay, Zheng Wang, et al. MEgATrack: monochrome egocentric articulated hand-tracking for virtual reality. *ACM Transactions on Graphics*, 39(4):87–1, 2020. 3

[13] Alexander Krull, Frank Michel, Eric Brachmann, Stefan Gumhold, Stephan Ihrke, and Carsten Rother. 6-DOF model based tracking via object coordinate regression. In *Asian Conference on Computer Vision*, pages 384–399, 2014. 3

[14] Xiaolong Li, He Wang, Li Yi, Leonidas J Guibas, A Lynn Abbott, and Shuran Song. Category-level articulated object pose estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3706–3715, 2020. 2, 3, 4, 5, 6, 7, 8, 9, 11

[15] Yi Li, Gu Wang, Xiangyang Ji, Yu Xiang, and Dieter Fox. Deepim: Deep iterative matching for 6D pose estimation. In *Proceedings of the European Conference on Computer Vision*, pages 683–698, 2018. 3, 4

[16] Fabian Manhardt, Wadim Kehl, and Adrien Gaidon. Roi-10d: Monocular lifting of 2d detection to 6d pose and metric shape. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2069–2078, 2019. 5

[17] Fabian Manhardt, Gu Wang, Benjamin Busam, Manuel Nickel, Sven Meier, Luca Minciullo, Xiangyang Ji, and Nassir Navab. Cps++: Improving class-level 6d pose and shape estimation from monocular images with self-supervised learning, 2020. 6, 7

[18] Frank Michel, Alexander Krull, Eric Brachmann, Michael Ying Yang, Stefan Gumhold, and Carsten Rother.

Pose estimation of kinematic chain instances via object coordinate regression. In *BMVC*, pages 181–1, 2015. 1, 2, 6, 7, 8

[19] Maher Moakher. Means and averaging in the group of rotations. *SIAM Journal on Matrix Analysis and Applications*, 24(1):1–16, 2002. 5, 11

[20] Franziska Mueller, Florian Bernard, Oleksandr Sotnychenko, Dushyant Mehta, Srinath Sridhar, Dan Casas, and Christian Theobalt. GANerated hands for real-time 3D hand tracking from monocular RGB. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 49–59, 2018. 3

[21] Iason Oikonomidis, Nikolaos Kyriazis, and Antonis A Argyros. Efficient model-based 3D tracking of hand articulations using kinect. In *BmVC*, volume 1, page 3, 2011. 3

[22] Karl Pauwels and Danica Kragic. Simtrack: A simulation-based framework for scalable real-time object pose detection and tracking. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1300–1307. IEEE, 2015. 3

[23] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. PointNet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in Neural Information Processing Systems*, pages 5099–5108, 2017. 2, 5, 11

[24] Mahdi Rad and Vincent Lepetit. Bb8: A scalable, accurate, robust to partial occlusion method for predicting the 3d poses of challenging objects without using depth. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3828–3836, 2017. 1

[25] Tanner Schmidt, Richard A Newcombe, and Dieter Fox. DART: Dense articulated real-time tracking. In *Robotics: Science and Systems*, 2014. 3

[26] Meng Tian, Marcelo H Ang, and Gim Hee Lee. Shape prior deformation for categorical 6d object pose and size estimation. In *European Conference on Computer Vision*, pages 530–546. Springer, 2020. 2

[27] Shinji Umeyama. Least-squares estimation of transformation parameters between two point patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(4):376–380, 1991. 2, 3, 4, 5, 11

[28] C. Wang, R. Martín-Martín, D. Xu, J. Lv, C. Lu, L. Fei-Fei, S. Savarese, and Y. Zhu. 6-pack: Category-level 6d pose tracker with anchor-based keypoints. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 10059–10066, 2020. 2, 6, 7

[29] He Wang, Srinath Sridhar, Jingwei Huang, Julien Valentin, Shuran Song, and Leonidas J Guibas. Normalized object coordinate space for category-level 6D object pose and size estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2642–2651, 2019. 1, 2, 3, 4, 5, 6, 7, 10, 12

[30] Jeremy Wang. 6-pack. https://github.com/j96w/6-PACK, 2020. 8

[31] Robert Y Wang and Jovan Popović. Real-time hand-tracking with a color glove. *ACM Transactions on Graphics*, 28(3):1–8, 2009. 3

[32] Bowen Wen, Chaitanya Mitash, Baozhang Ren, and Kostas E Bekris. se(3)-TrackNet: Data-driven 6D pose tracking by calibrating image residuals in synthetic domains. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2020. 2, 3, 4

[33] Manuel Wüthrich, Peter Pastor, Mrinal Kalakrishnan, Jeannette Bohg, and Stefan Schaal. Probabilistic object tracking using a range camera. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3195–3202, 2013. 3

[34] Fanbo Xiang, Yuzhe Qin, Kaichun Mo, Yikuan Xia, Hao Zhu, Fangchen Liu, Minghua Liu, Hanxiao Jiang, Yifu Yuan, He Wang, Li Yi, Angel X. Chang, Leonidas J. Guibas, and Hao Su. SAPIEN: A simulated part-based interactive environment. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 1, 2, 6, 12

[35] Yu Xiang, Tanner Schmidt, Venkatraman Narayanan, and Dieter Fox. Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes. In *Robotics: Science and Systems (RSS)*, 2018. 1, 2

[36] Bin Xiao, Haiping Wu, and Yichen Wei. Simple baselines for human pose estimation and tracking. In *Proceedings of the European Conference on Computer Vision*, pages 466–481, 2018. 3

[37] Li Yi, Haibin Huang, Difan Liu, Evangelos Kalogerakis, Hao Su, and Leonidas Guibas. Deep part induction from articulated object pairs. In *SIGGRAPH Asia 2018 Technical Papers*, page 209, 2018. 5

[38] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Open3d: A modern library for 3d data processing. *arXiv preprint arXiv:1801.09847*, 2018. 3

[39] Yi Zhou, Connelly Barnes, Jingwan Lu, Jimei Yang, and Hao Li. On the continuity of rotation representations in neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5745–5753, 2019. 5

A. Coordinate-based Method Review

Normalized Object Coordinate Space (NOCS) NOCS is a category-level canonical reference frame defined within a unit 3D cube introduced in [29]. The objects from the same category in NOCS are consistently aligned to a category-level canonical orientation. These objects are further zero-centered and uniformly scaled so that their tight bounding boxes are centered at the origin of NOCS with a diagonal length of 1.

Mathematically, for an object point cloud $X \in \mathbb{R}^{N\times 3}$, its corresponding point-wise normalized object coordinates are denoted as $Y \in \mathbb{R}^{N\times 3}$. The transformation between Y in the NOCS frame and X in the camera frame is a 7D similarity transformation $\mathcal{T}^{(j)} = \{s^{(j)}, R^{(j)}, T^{(j)}\}$, which satisfies $X = sRY + T$. This 7D transformation defines the category-level 6D pose and 1D uniform scale of rigid objects.

Given the input object point cloud X, Wang *et. al.* [29] trained a deep neural network to directly regress Y. The

7D similarity transformation can then be computed with the 3D-3D point correspondence established between X and Y using the Umeyama algorithm [27] along with RANSAC. Knowing the 1D scale s, the actual object size $d \in \mathbb{R}^3$ can be estimated as $d = s \times (|x|_{max}, |y|_{max}, |z|_{max})$.

Normalized Part Coordinate Space (NPCS) Li *et. al.*[14] extended the definition of NOCS to rigid parts in articulated objects, and proposed a part-level canonical reference frame, namely NPCS. Similar to NOCS, each individual part has canonical orientation, zero translation, and normalized scale in its NPCS. Leveraging per-point NPCS estimations, per-part 9DoF poses can be estimated in the same way as NOCS.

B. Proof of the pose canonicalization theorem

Proof. Denote the corresponding points of $C_{t+1}^{(j)}$ in $Z_{t+1}^{(j)}$ as $\hat{C}_{t+1}^{(j)}$, then $C_{t+1}^{(j)} = s_{t+1}^{(j)} R_{t+1}^{(j)} Y_{t+1}^{(j)} + T_{t+1}^{(j)}$, $\hat{C}_{t+1}^{(j)} = \hat{s}_t^{(j)} \hat{R}_t^{(j)} Y_{t+1}^{(j)} + \hat{T}_t^{(j)}$. Given that $\hat{C}_{t+1}^{(j)} = (R_t^{(j)})^{-1}(C_{t+1}^{(j)} - T_t^{(j)})/s_t^{(j)}$, we can obtain $\hat{s}_t^{(j)} = (s_t^{(j)})^{-1} s_{t+1} \approx 1$, $\hat{R}_t^{(j)} = (R_t^{(j)})^{-1} R_{t+1}^{(j)} \approx I$, $\hat{T}_t^{(j)} = (s_t^{(j)})^{-1}(R_t^{(j)})^{-1}(T_{t+1}^{(j)} - T_t^{(j)}) \approx 0$, as a natural result of temporal continuity. □

C. Per-part Rotation, Scale and Translation Computation

C.1. Euclidean Mean of Rotations

For averaging over rotations, we adopt the euclidean mean [19] of the multiple rotation predictions, which converts the 6D rotation representation back to matrix format, takes the mean matrix, and then project back to $SO(3)$. Taking a binary segmentation mask $m_{t+1}^{(j)}$, our final prediction is given by $\hat{R}_{t+1}^{(j)} = \text{EuclideanMean}(\{\hat{R}_{i,t+1}^{(j)} | i \in m_{t+1}^{(j)}\}_i)$.

C.2. Scale and Translation Computation

For part j, given segmentation mask prediction $\tilde{m}_{t+1}^{(j)}$ and normalized coordinate predictions from the CoordinateNet, we can obtain input points from part j, namely $\tilde{C}_{t+1}^{(j)} = \{X_{i,t+1} | i \in \tilde{m}_{t+1}^{(j)}\}$ and their corresponding normalized coordinate predictions $\tilde{Y}_{t+1}^{(j)}$.

Based on RotationNet predictions, we can compute absolute per-part rotation prediction $\tilde{R}_{t+1}^{(j)} = \tilde{R}_t^{(j)} \hat{R}_{t+1}^{(j)}$.

For asymmetrical objects, let $\tilde{W}_{t+1}^{(j)} = \tilde{R}_{t+1}^{(j)} \tilde{Y}_{t+1}^{(j)}$, $\tilde{W}_{t+1}^{(j)}$ and $\tilde{C}_{t+1}^{(j)}$ only differs by a scaling and a translation, namely

$$\tilde{C}_{t+1}^{(j)} = \tilde{s}_{t+1}^{(j)} \tilde{W}_{t+1}^{(j)} + \tilde{T}_{t+1}^{(j)}$$

We compute the scale and translation of part j as follows:

$$\tilde{s}_{t+1}^{(j)} = \sum_i \tilde{W}_{i,t+1}^{(j)\top} \tilde{C}_{i,t+1}^{(j)} / \sum_i \tilde{W}_{i,t+1}^{(j)\top} \tilde{W}_{i,t+1}^{(j)}$$

$$\tilde{T}_{t+1}^{(j)} = \text{avg}_i (\tilde{C}_{i,t+1}^{(j)} - \tilde{s}_{t+1}^{(j)} \tilde{W}_{i,t+1}^{(j)})$$

For symmetrical objects, Let $\tilde{U}_{t+1}^{(j)} = \tilde{R}_{t+1}^{(j)\top} \tilde{C}_{t+1}^{(j)}$. Besides a scaling and a translation, $\tilde{Y}_{t+1}^{(j)}$ and $\tilde{U}_{t+1}^{(j)}$ may further differ by a rotation $R(l, \theta)$ around the axis of symmetry l, namely

$$\tilde{U}_{t+1}^{(j)} = \tilde{s}_{t+1}^{(j)} R(l, \theta) \tilde{Y}_{t+1}^{(j)} + \tilde{R}_{t+1}^{(j)\top} \tilde{T}_{t+1}^{(j)}$$

This is because for ground-truth normalized coordinates $Y_{t+1}^{(j)*}$, $R(l, \theta) Y_{t+1}^{(j)*}$ will also be a correct set of predictions due to the object symmetry.

To simplify the problem, we assume l overlaps with the y-axis, then $R(l, \theta)$ becomes a 2D rotation in xz-plane.

We propose to take the xz-plane projection of everything and use the 2D version of Umeyama algorithm [27] to compute $R(l, \theta)$. Then we have:

$$R(l, \theta)^\top \tilde{U}_{t+1}^{(j)} = \tilde{s}_{t+1}^{(j)} \tilde{Y}_{t+1}^{(j)} + (\tilde{R}_{t+1}^{(j)} R(l, \theta))^\top \tilde{T}_{t+1}^{(j)}$$

This is the same case as asymmetrical objects, we can compute $\tilde{s}_{t+1}^{(j)}$ and $\tilde{T}_{t+1}^{(j)}$ similarly.

D. Implementation Details

D.1. Network Input

For synthetic articulated data, the input to our network is a partial depth point cloud projected from a single-view depth image, downsampled to $N = 4096$ points using FPS. For NOCS-REAL275 data, our pipeline crops a ball centered at estimated position of the object center, with a radius 1.2 times the object's estimated radius. Scene points within the ball are then downsampled to $N = 4096$ points.

D.2. Training Details

Our network is implemented using PyTorch and optimized by the Adam optimizer, with a learning rate starting at 10^{-3} and decay by half every 20 epochs. It takes around 20 and 100 epochs for our model to converge on the rigid object dataset and the articulated object dataset, respectively. We will release our code to facilitate research in this field.

D.3. Network Architecture

Both CoordinateNet and RotationNet use PointNet++ [23] MSG segmentation network as their backbone. The detailted architecture is as follows:

Backbone:

SA(`num_points` $= 512$, `radius` $= [0.05, 0.1, 0.2]$,
`mlps`$=[[32, 32, 64], [64, 64, 128], [64, 96, 128]]) \rightarrow$
SA(`num_points` $= 128$, `radius` $= [0.2, 0.4]$,
`mlps`$=[[128, 128, 256], [128, 196, 256]]) \rightarrow$
GlobalSA(`mlp`$=[256, 512, 1024]) \rightarrow$

Category	Part definitions				Data statistics		Training Pose Perturbation Distribution $\mathcal{N}(0, \sigma)$		
	Part 0	Part 1	Part 2	Part 3	Train/Test	Average joint state change	σ_{scale}	$\sigma_{rot}(^\circ)$	σ_{trans}(cm)
glasses	right temple	left temple	base	-	47/8	19.19°	0.02	5	2
scissors	right half	left half	-	-	33/3	34.32°	0.01	3	1
laptop	base	display	-	-	49/6	26.13°	0.015	3	2
drawers	lowest	middle	top	base	28/2	3.72cm	0.02	3	2

Table 7. **Statistics of our synthetic articulated object dataset.**

FP(mlp=$[256, 256]$) $\rightarrow$

FP(mlp=$[256, 128]$) $\rightarrow$

FP(mlp=$[128, 128]$)

CoordinateNet:

Backbone $\rightarrow$

Coordinate Head:MLP($[128, 3P]$) $\rightarrow$ Sigmoid()

Segmentation Head:FC($[P + 1]$) $\rightarrow$ Softmax()

RotationNet:

Backbone $\rightarrow$ MLP($[512, 512, 256, 6P]$)

We use LeakyReLU and group normalization for each FC layer in set abstraction (SA) and feature propagation (FP) layers.

E. SAPIEN Articulated Objects Data Generation and Statistics

We render our synthetic articulated object pose tracking dataset using SAPIEN [34]. In Table 7, we summarize for each object category 1) the part definitions; 2) the train-test split; 3) the average joint state change over all test sequences (each consisting of 100 frames); and 4) the variance of Gaussian noise distributions from which we sample input pose perturbations during training. **Note that both the global pose and the joint states of the articulated objects are changing in the trajectories.**

We use different amount of noise for different object categories depending on the difficulty of pose estimation, e.g. the poses of thin glasses temples are difficult to predict, therefore we train the model with a larger perturbation to handle larger prediction error during tracking.

Method		NOCS	CASS	CPS++	Oracle ICP	6PACK	6PACK	Ours
Input		RGBD	RGBD	RGB	Depth	RGBD	RGBD	Depth
Setting		Single frame			Tracking			
Initialization		N/A	N/A	N/A	GT.	GT.	Pert.	Pert.
bottle	5°5cm ↑	5.50	11.49	2.90	0.28	14.11	17.48	**79.46**
	mIoU↑	33.73	34.72	27.91	10.72	59.77	49.98	**72.11**
	$R_{err}\downarrow$	25.60	18.39	14.81	44.03	21.45	12.83	**3.29**
	$T_{err}\downarrow$	14.40	26.66	32.67	8.28	3.36	4.97	**2.60**
bowl	5°5cm ↑	62.20	33.50	5.61	0.45	40.46	34.30	**79.20**
	mIoU↑	78.78	62.36	32.07	11.54	56.29	56.15	**79.64**
	$R_{err}\downarrow$	4.70	5.98	12.39	30.31	5.83	6.78	**3.50**
	$T_{err}\downarrow$	**1.20**	4.76	19.97	6.65	1.64	1.67	1.43
can	5°5cm ↑	7.10	22.24	3.22	0.49	28.07	21.51	**64.70**
	mIoU↑	49.56	59.43	33.20	10.50	50.32	49.48	**62.47**
	$R_{err}\downarrow$	16.90	12.08	13.99	43.85	11.66	16.58	**3.43**
	$T_{err}\downarrow$	**4.00**	9.08	19.75	8.48	5.03	5.82	5.69
camera	5°5cm ↑	0.60	**12.73**	0.29	0.60	6.89	0.97	0.41
	mIoU↑	58.13	**60.84**	36.18	19.62	52.10	51.55	2.50
	$R_{err}\downarrow$	33.80	**14.70**	30.22	36.09	49.96	57.65	17.82
	$T_{err}\downarrow$	**3.10**	7.29	16.12	7.23	6.06	5.65	35.53
laptop	5°5cm ↑	25.50	82.81	0.51	1.60	64.09	36.31	**94.03**
	mIoU↑	52.59	63.98	19.58	22.11	49.76	49.79	**87.20**
	$R_{err}\downarrow$	8.60	5.89	30.85	14.39	5.03	6.12	**2.24**
	$T_{err}\downarrow$	2.40	3.89	13.47	8.41	2.57	2.44	**1.48**
mug	5°5cm ↑	0.90	13.85	0.90	0.48	19.90	22.23	**55.17**
	mIoU↑	58.08	54.56	31.15	13.63	64.26	64.54	**80.70**
	$R_{err}\downarrow$	31.50	27.97	49.65	73.02	22.06	17.99	**5.36**
	$T_{err}\downarrow$	4.00	20.76	27.73	7.21	1.19	1.17	**0.79**
all	5°5cm ↑	16.97	29.44	2.24	0.65	28.92	22.13	**62.16**
	mIoU↑	55.15	55.98	30.02	14.69	55.42	53.58	**64.10**
	$R_{err}\downarrow$	20.18	14.17	25.32	40.28	19.33	19.66	**5.94**
	$T_{err}\downarrow$	4.85	12.07	21.62	7.71	**3.31**	3.62	7.92
all w/o cam.	5°5cm ↑	20.24	32.78	2.63	0.66	33.33	26.37	**74.51**
	mIoU↑	54.55	55.01	28.78	13.70	56.08	53.99	**76.42**
	$R_{err}\downarrow$	17.46	14.06	24.34	41.12	13.20	12.06	**3.56**
	$T_{err}\downarrow$	5.20	13.03	22.72	7.81	2.76	3.21	**2.40**

Table 8. **Per-category results of category-level rigid object pose tracking on NOCS-REAL275**

.

F. Additional Results

F.1. Per-Category Results for Rigid Objects

Table 8 summarizes the per-category results for rigid object pose tracking on NOCS-REAL275. Our method only fails on the camera category and outperforms the previous state-of-the-arts under most metrics on all other categories. This is mainly due to the huge domain gap between our mostly synthetic training data and real camera test instances - 2 out of 3 are black and hence yield a larger sensor noise. Our method purely relying on depth points is not designed to cope with this issue. Consequently, our CoordinateNet fails to segment out the camera instances. To ameliorate this issue, we use 2D segmentation masks from RGB-based Mask-RCNN detection predictions provided in [29]. When there are multiple RoIs of the camera category, we choose the one having the biggest overlap with a predicted 2D bounding box computed from our previous pose estimation. We also test our method with ground-truth segmentation masks. The results are shown in Table 10. Our performance significantly improves with better segmentation predictions.

Method		ANCSH	Oracle ICP	Ours	C-sRT	C-CrdNet	C-Crd.+ DSAC++	Ours w/o L_c, L_s, L_t
glasses	5°5cm↑	72.6, 75.8, 81.9	46.9, 46.1, 78.4	**97.7**, 95.3, **99.6**	27.1, 22.6, 25.3	91.6, 89.2, 91.5	81.9, 84.0, 92.4	97.4, **96.0**, 99.2
	mIoU↑	73.7, 74.3, 47.7	65.8, 67.2, 56.0	**81.8, 81.4, 57.2**	12.6, 11.5, 1.7	81.2, 80.8, 56.7	67.7, 71.2, 41.5	80.8, 80.8, 55.0
	$R_{err}\downarrow$	4.17, 3.86, 3.58	11.00, 10.22, 4.66	**1.72, 1.93, 1.22**	5.80, 5.86, 2.98	2.78, 3.06, 1.90	3.43, 3.17, 2.00	1.87, 2.14, 1.47
	$T_{err}\downarrow$	0.47, 0.50, 0.23	2.10, 2.82, 1.98	0.27, 0.26, **0.14**	11.43, 12.56, 12.67	**0.25, 0.24, 0.14**	0.55, 0.39, 0.22	0.27, 0.33, 0.17
	$\theta_{err}\downarrow$	1.40, 1.43	5.25, 4.25	0.94, 0.97	3.26, 3.64	**0.65**, 0.74	0.73, **0.65**	0.93, 1.14
scissors	5°5cm↑	98.7, 98.8	25.7, 28.3	99.0, 99.4	3.1, 2.7	96.6, 98.7	**99.5, 99.9**	98.4, 99.4
	mIoU↑	64.0, 64.4	19.9, 26.8	65.6, 71.9	1.1, 1.6	64.9, 72.5	**65.7**, 71.4	63.0, **72.2**
	$R_{err}\downarrow$	1.82, 1.77	19.85, 17.30	1.60, **1.17**	55.17, 59.08	2.25, 1.88	1.56, 1.77	**1.48**, 1.23
	$T_{err}\downarrow$	0.16, 0.21	7.80, 4.82	0.12, 0.14	7.63, 7.62	**0.10, 0.12**	0.13, 0.14	0.14, 0.15
	$\theta_{err}\downarrow$	1.96	13.14	**1.85**	11.11	1.93	1.96	1.89
laptop	5°5cm↑	**97.5, 99.1**	81.4, 92.4	97.1, 97.2	38.8, 57.3	96.1, 98.3	96.5, 98.4	96.6, 95.9
	mIoU↑	70.3, 50.6	52.5, **62.7**	**76.2**, 53.5	45.9, 40.4	74.3, 54.0	47.5, 42.7	73.2, 47.8
	$R_{err}\downarrow$	1.72, 1.08	6.85, 1.70	**0.62, 1.22**	4.86, 2.61	3.02, 1.92	2.14, 1.49	1.18, 1.31
	$T_{err}\downarrow$	0.58, 0.49	2.00, 0.90	**0.32, 0.35**	9.63, 6.18	0.58, 0.52	1.31, 1.00	0.43, 0.42
	$\theta_{err}\downarrow$	1.48	3.74	**1.33**	3.69	1.94	2.17	1.35
drawers	5°5cm↑	94.3, 93.5, 98.1, 99.6	65.8, 79.7, 79.9, 96.4	**99.6, 99.6, 99.6, 99.7**	6.7, 11.2, 14.1, 11.3	92.2, 91.7, 97.2, 97.4	97.2, 96.7, 97.8, 97.5	97.4, 97.3, 98.0, 98.5
	mIoU↑	80.7, 83.3, 84.4, 91.1	73.8, 80.8, 82.3, 93.3	**85.1, 86.4, 89.8, 94.2**	26.2, 30.3, 30.9, 41.2	83.5, 84.7, 88.8, 93.0	84.2, 85.2, 88.2, 88.2	84.9, 86.3, 88.9, 92.0
	$R_{err}\downarrow$	2.11, 2.21, 1.67, 0.69	8.45, 5.40, 2.69, 0.80	**0.18, 0.18, 0.19, 0.23**	22.07, 15.82, 16.20, 23.39	2.22, 2.20, 1.23, 0.63	1.18, 1.21, 0.83, 0.70	0.55, 0.65, 0.44, 0.51
	$T_{err}\downarrow$	1.15, 0.85, 0.68, 0.51	3.33, 2.51, 1.48, 1.07	**0.59, 0.60, 0.38, 0.29**	22.99, 17.56, 13.07, 18.77	0.91, 0.93, 0.46, 0.57	0.70, 0.62, 0.40, 0.66	0.74, 0.73, 0.51, 0.36
	$d_{err}\downarrow$	0.72, 0.62, 0.58	1.33, 1.00, 0.82	**0.37, 0.36, 0.28**	5.31, 6.91, 10.48	0.75, 0.83, 0.68	0.46, 0.65, 0.58	0.39, 0.37, 0.32

Table 9. **Per-part, per-category results of category-level articulated object pose tracking on held-out instances from SAPIEN.**

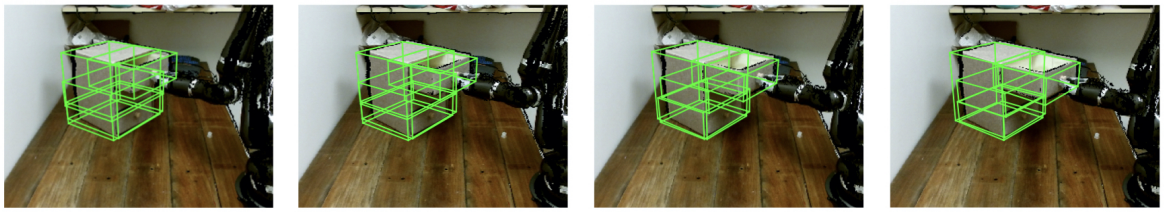

Figure 4. **Qualitative evaluation on the real drawers trajectory.** Color images are shown for better reference.

F.2. Per-Part, Per-Category Results for Articulated Object Tracking

Table 9 shows per-part, per-category articulated object pose tracking results on our synthetic dataset. In most cases, we perform better than the baseline methods and the ablated versions. We also achieve the best overall performance as shown in the main paper.

F.3. Additional Visualization

See Fig. 6 - 8 for our visual tracking results. For more visual results, please refer to our supplementary videos.

Mask Source	CoordNet(Depth)	NOCS(RGB)	GT
5°5cm↑	0.41	9.05	20.09
mIoU↑	2.50	33.00	46.35
$R_{err}\downarrow$	17.82	20.75	10.89
$T_{err}\downarrow$	35.53	13.09	3.67

Table 10. **Results on the camera category from NOCS-REAL275 with different segmentation mask sources.**

F.4. Tracking Real Drawers under Robot Object Interaction

To evaluate our method on articulated objects with prismatic joints, we capture a real drawers trajectory using Kinect2, where a Kinova Jaco2 arm interacts with the drawers and pulls out the top drawer, as shown in 4. We process the real data by removing irrelevant depth points in the background or on the table, and use a ground-truth object mask as in our experiments on real laptop trajectories during evaluation.

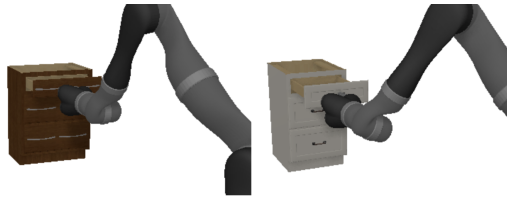

Figure 5. **Synthetic training data for real drawers tracking.** We use SAPIEN environment to render depth sequences where a Kinova Jaco2 robotic arm manipulates one drawer. Here color images are for visualization only.

To generate training data, we simulate the arm pulling synthetic drawers models from SAPIEN and render the depth images. Figure 5 shows two examples of our simulation data. We train our model only on those synthetic depth images following the same protocol as in Section 5.3.

The qualitative results in Figure 4 demonstrate that our model successfully generalizes to real data and tracks both the moving top drawer and the other parts.

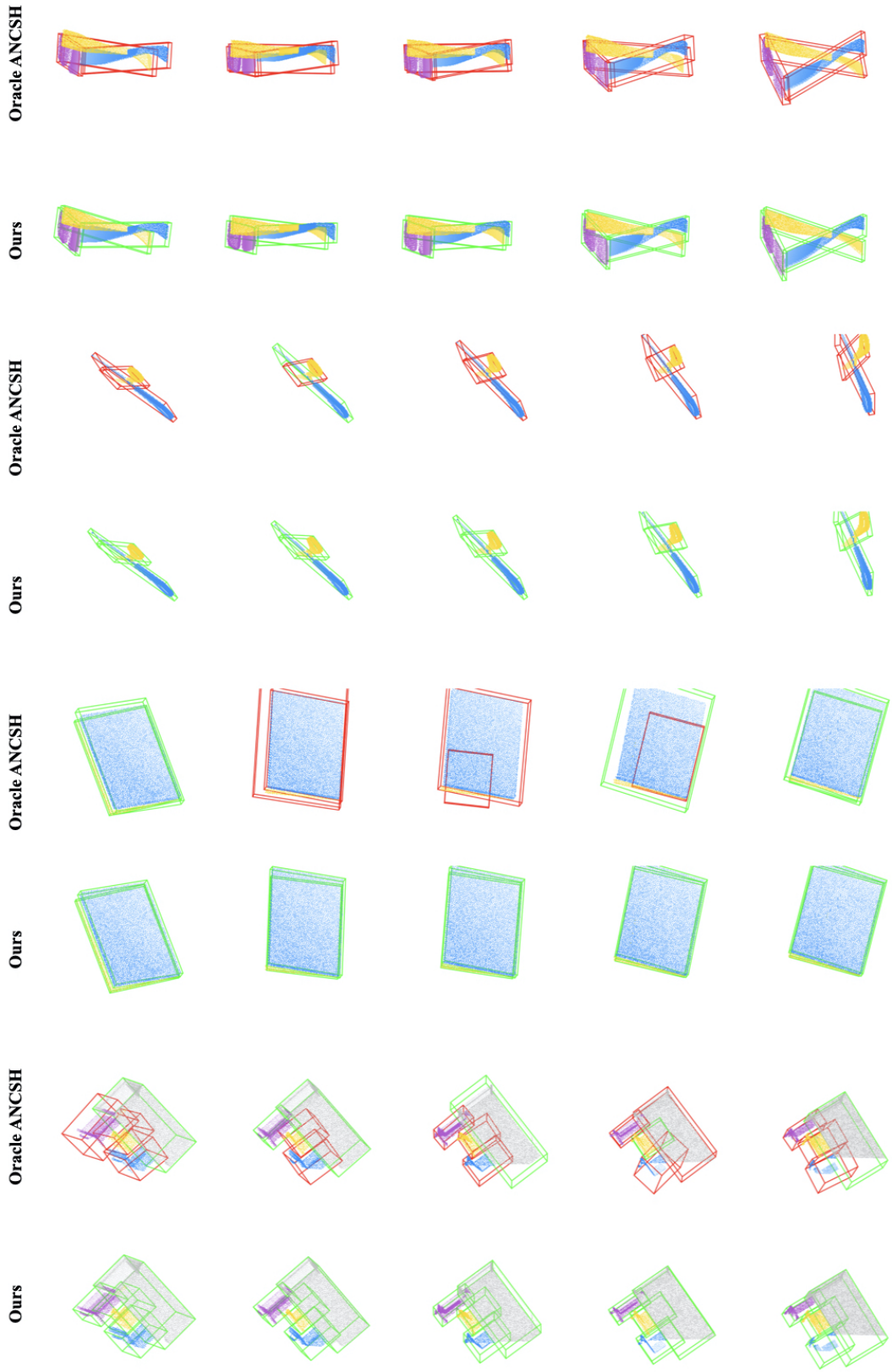

Figure 6. **Result visualization on the SAPIEN articulated object dataset.** Here we compare our method with oracle ANCSH, which assumes the availability of ground truth part masks.

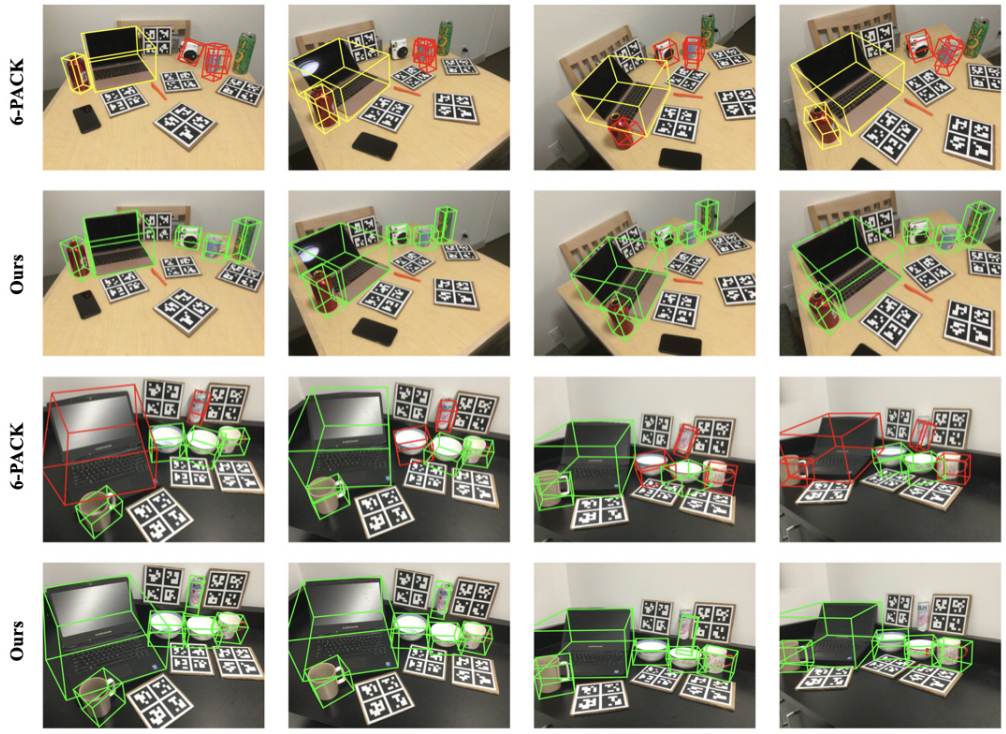

Figure 7. **Result visualization on the NOCS-REAL275 dataset.** Here we compare our method with 6-PACK initialized with the same pose noise as ours.

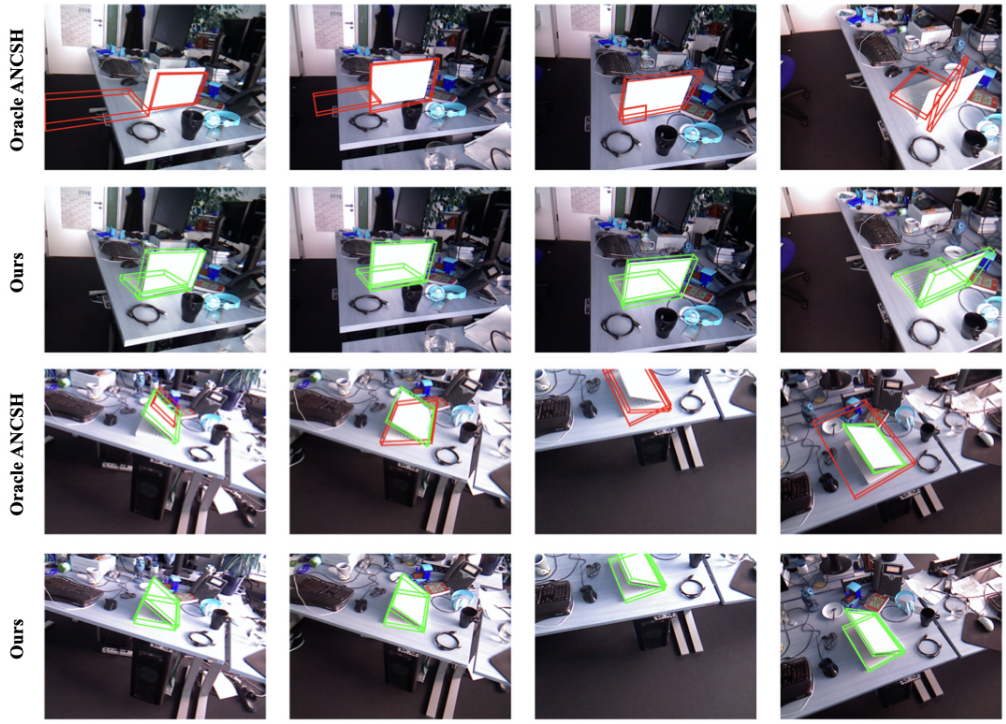

Figure 8. **Result visualization on the real laptop trajectories from the BMVC dataset.** Here we compare our method with Oracle ANCSH under the category-level setting, where the two methods only see synthetic data from SAPIEN during training and directly test on the real data without finetuning.