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

Yijia Weng[1*] He Wang[1,2,5*†] Qiang Zhou [4] Yuzhe Qin[3] Yueqi Duan[2]
Qingnan Fan[2,6] Baoquan Chen[1] Hao Su[3] Leonidas J. Guibas[2]
[1]CFCS, Peking University [2]Stanford University [3]UCSD
[4]Shandong University [5]Beijing Institute for General AI [6]Tencent AI Lab

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 [5]. 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.* [5] 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 [4] 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.*[1] 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 observation

Proof. Denote the corresponding points of $C^{(j)}_{t+1}$ in $Z^{(j)}_{t+1}$ as $\hat{C}^{(j)}_{t+1}$, then $C^{(j)}_{t+1} = s^{(j)}_{t+1}R^{(j)}_{t+1}Y^{(j)}_{t+1} + T^{(j)}_{t+1}$, $\hat{C}^{(j)}_{t+1} = \hat{s}^{(j)}_t \hat{R}^{(j)}_t Y^{(j)}_{t+1} + \hat{T}^{(j)}_t$. Given that $\hat{C}^{(j)}_{t+1} = (R^{(j)}_t)^{-1}(C^{(j)}_{t+1} - T^{(j)}_t)/s^{(j)}_t$, we can obtain $\hat{s}^{(j)}_t = (s^{(j)}_t)^{-1}s_{t+1} \approx 1$, $\hat{R}^{(j)}_t = (R^{(j)}_t)^{-1}R^{(j)}_{t+1} \approx I$, $\hat{T}^{(j)}_t = (s^{(j)}_t)^{-1}(R^{(j)}_t)^{-1}(T^{(j)}_{t+1} - T^{(j)}_t) \approx 0$, 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 [2] 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 $m^{(j)}_{t+1}$, our final prediction is given by $\hat{R}^{(j)}_{t+1} = \text{EuclideanMean}(\{\hat{R}^{(j)}_{i,t+1} | i \in m^{(j)}_{t+1}\}_i)$.

C.2. Rotation Supervision for Symmetric Objects

Unseen instances from symmetric object categories, like bowls or bottles contain a rotation ambiguity around their symmetric axis $\hat{q}$ [5]. 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 endpoint position $p_{\hat{R}}$ of its unit rotation axis $\hat{q}$. Similar to [7], 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 [5].

C.3. 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.

C.4. 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 [4] 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 have made our code public at https://github.com/halfsummer11/CAPTRA

D.3. Network Architecture

Both CoordinateNet and RotationNet use PointNet++ [3] MSG segmentation network as their backbone. The detailted architecture is as follows:

Backbone:
SA($\texttt{num\_points} = 512, \texttt{radius} = [0.05, 0.1, 0.2]$, $\texttt{mlps}=[[32,32,64],[64,64,128],[64,96,128]]$) $\rightarrow$
SA($\texttt{num\_points} = 128, \texttt{radius} = [0.2, 0.4]$, $\texttt{mlps}=[[128,128,256],[128,196,256]]$) $\rightarrow$
GlobalSA($\texttt{mlp}=[256,512,1024]$) $\rightarrow$

FP($\texttt{mlp}=[256,256]$) $\rightarrow$
FP($\texttt{mlp}=[256,128]$) $\rightarrow$
FP($\texttt{mlp}=[128,128]$)
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 [6]. In Table 1, we summarize for

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 1. **Statistics of our synthetic articulated object dataset.**

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.

F. Experiment Details of Real Drawers under Robot-Object Interaction

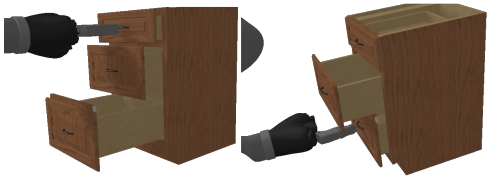

Figure 1. **Synthetic training data for real drawers tracking.** We use SAPIEN environment to render depth sequences where a Kinova Jaco2 robotic arm manipulates one drawer. Here color images are for visualization only.

To mimic the interaction scenario, we simulate a Kinova Jaco2 robotic arm in the SAPIEN [6] environment, make it push synthetic drawers from the SAPIEN dataset, and render depth images. Figure 1 shows two examples of our simulated data. We train our model only on these synthetic depth images following the same protocol as in Section 5.3.

G. Discussions on not Using 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

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

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

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,

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

when we remove RANSAC, C-CoordinateNet drops 3.22% on 5°5cm accuracy and 4.37% on mIoU, due to the aforementioned rotation sensitivity.

H. Additional Results

H.1. Per-Category Results for Rigid Objects

Table 2 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 [5]. 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 3. Our performance significantly improves with better segmentation predictions.

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

Table 4 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.

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.

H.3. Robustness to Pose Errors

Table 5 shows the performance of our model w.r.t. different amount of pose errors. We 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; 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 initialization and estimation errors, respectively. For rigid objects, our performance degrades gracefully. And as shown in Fig.4 of the main paper, it is significantly more robust to 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.

H.4. Additional Visualization

See Fig.2 - 4 for our visual tracking results. For more visual results, please refer to our supplementary videos.

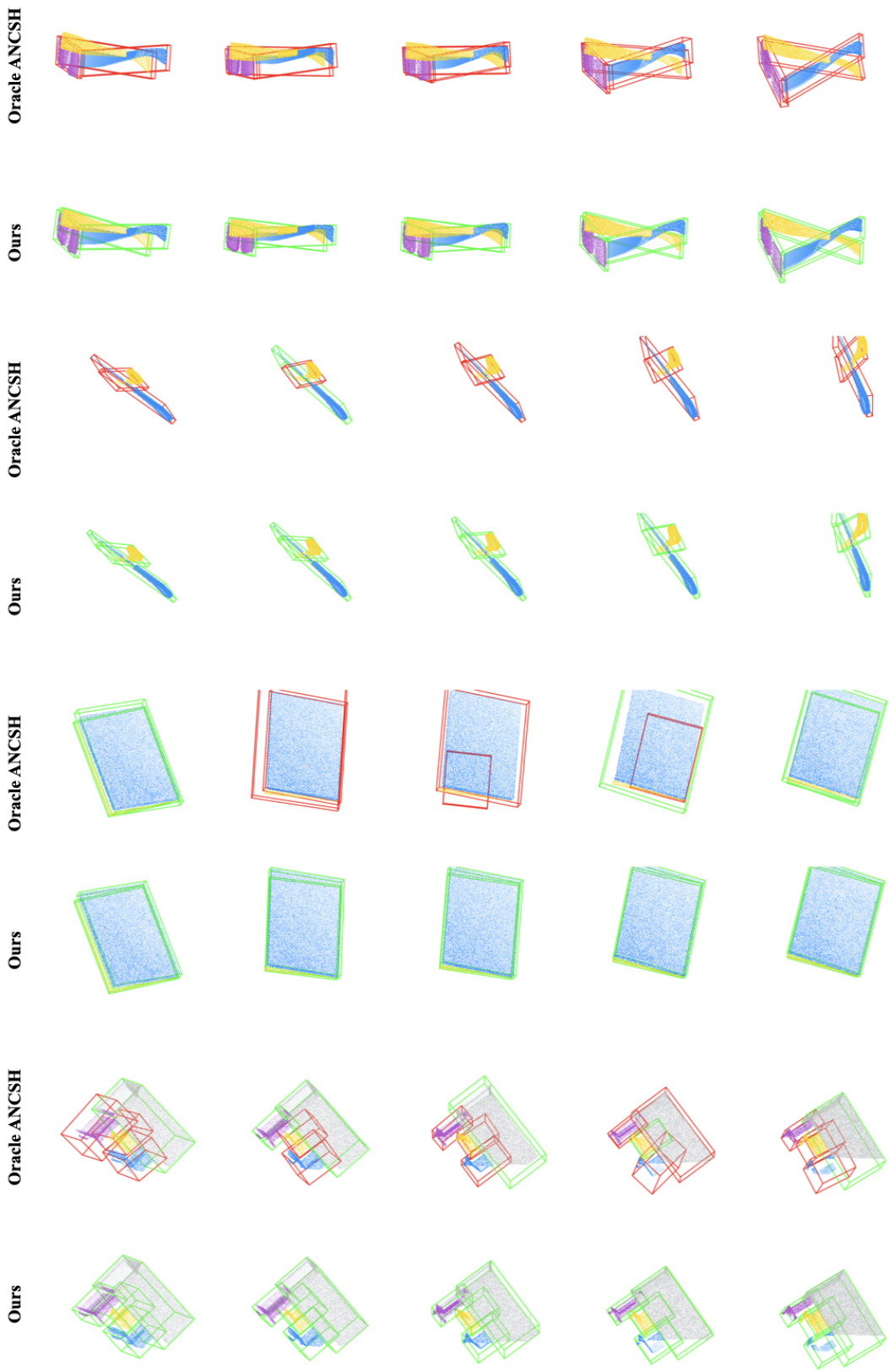

Figure 2. **Result visualization on the SAPIEN articulated object dataset.** Here we compare our method with oracle ANCSH, which assumes the availability of ground truth part masks.

Figure 3. **Result visualization on the NOCS-REAL275 dataset.** Here we compare our method with 6-PACK initialized with the same pose noise as ours.

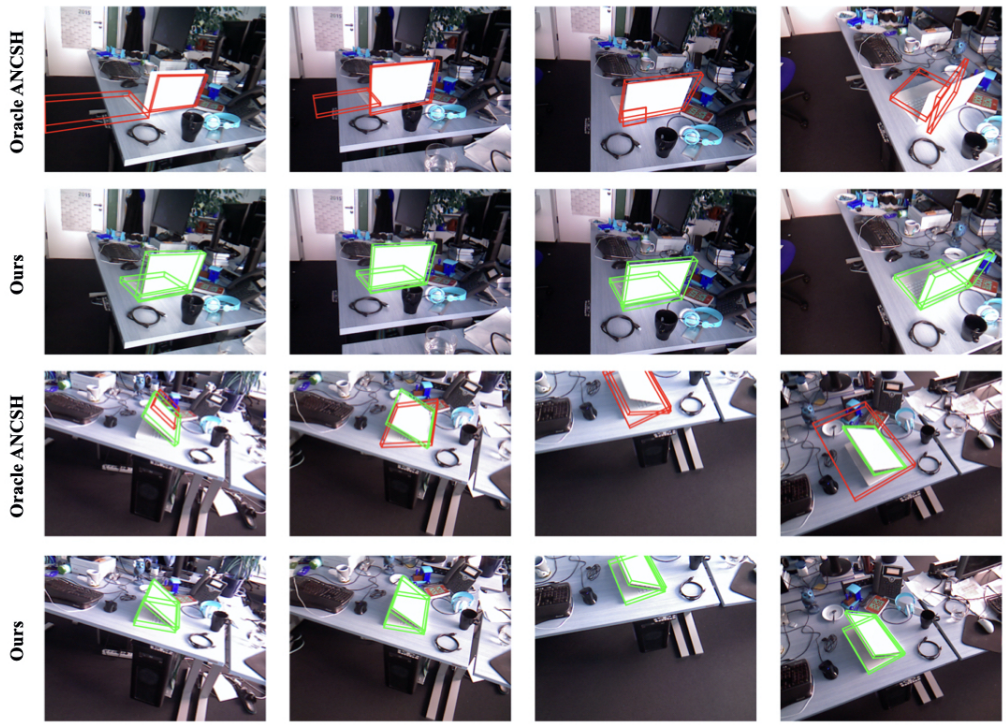

Figure 4. **Result visualization on the real laptop trajectories from the BMVC dataset.** Here we compare our method with Oracle ANCSH under the category-level setting, where the two methods only see synthetic data from SAPIEN during training and directly test on the real data without finetuning.

References

[1] 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. 1

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

[3] 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

[4] 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. 1, 2

[5] 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, 4

[6] 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. 2, 3

[7] 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. 1