Category-Level Articulated Object Pose Estimation Supplementary Material

Xiaolong Li[1*] He Wang[2*] Li Yi[3] Leonidas Guibas [2] A. Lynn Abbott [1] Shuran Song[4]
[1]Virginia Tech [2]Stanford University [3]Google Research [4]Columbia University

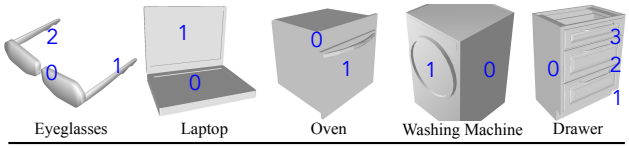
1. Implementation Details

We use Tensorflow 1.10 to build our models and run the experiments for all categories. The input is uniformly sampled points from the whole back-projected depth point cloud, with points number N set to 1024. We train our model on a single Nvidia V100 GPU with batch size of 16 across the experiments. The initial learning rate is set to 0.001, with a decay factor of 0.7 every 200k steps. From the observations of our experiments, the loss will usually converge well after $> 150k$ steps in less than one day.

2. Data generation and statistics

We render synthetic depth images using the object 3D model provided in the Shape2Motion dataset [4] and SAPIEN dataset [5]. Both datasets provide the descriptions of the object geometry and articulation information, which we leverage for generating ground truths. During rendering, the program automatically generates random joint states for each object instance, according to its joint motion ranges. Then the depth images and corresponding ground truth masks are rendered from a set of random camera viewpoints. We also filter out camera poses where some parts of the object are completely occluded. Figure 1 shows the index definitions of parts for each object category used in the main paper, together with number of object instances splitted for training and testing.

We use real data from ICCV2015 Articulated Object Challenge [3], which contains RGB-D data with 4 articulated objects: laptop, cabinet, cupboard and toy train. This dataset provides 2 testing sequences for each object. Each sequence contains around 1000 images captured by having a RGB-D camera slowly moving around the object. Objects maintain the same articulation state within each sequence. Each part of the articulated object is annotated with its 6D pose with respect to the known CAD model. Since no training data is provided, we use the provided CAD models to render synthetic depth data, with 10 groups of random articulation status considered. We render object masks for the testing sequences with Pybullet[2].



Category	Part definitions				Data statistics
	Part 0	Part 1	Part 2	Part 3	train/test
Eye-glasses	base	left temple	right temple	-	39/3
Oven	base	door	-	-	35/3
Washing Machine	base	door	-	-	42/2
Laptop	base	display	-	-	78/3
Drawer	base	lowest	middle	top	30/2

Figure 1. **Synthetic data statistics.** We list part definitions for each object category tested in our experiments on synthetic data, together with the numbers of object instances used for training and testing.

3. Handling severe occlusion cases

We carefully examine how ANCSH performs under different levels of occlusion. Compared to our NPCS baseline, our proposed method is still capable of improving the pose estimation under severe occlusion, as shown in Figure 2. The occlusion level is defined according to the ratio of visible area with respect to the total mesh surface per part.

4. Real-world instance-level benchmark.

We have also tested our algorithm's ability to generalize to real-world depth images on the dataset provided in [3]. Table 1 shows quantitative comparison of AD accuracy.

5. Additional results

Figure 4 shows additional qualitative results on the synthetic dataset. More qualitative results on real-world

* indicates equal contributions.

Object	Sequence		Brachmann et al.[1]	Frank et al.[3]	ANCSH (Ours)
Laptop	1	all	8.9%	64.8%	**94.1%**
		parts	29.8% 25.1%	65.5% 66.9%	**97.5% 94.7%**
	2	all	1%	65.7%	**98.4%**
		parts	1.1% 63.9%	66.3% 66.6%	**98.9% 99.0%**
Cabinet	3	all	0.5%	**95.8%**	90.0%
		parts	86% 46.7% 2.6%	98.2% 97.2% **96.1%**	**98.9% 97.8%** 91.9%
	4	all	49.8%	**98.3%**	94.5%
		parts	76.8% 85% 74%	98.3% 98.7% **98.7%**	**99.5% 99.5%** 94.9%
Cupboard	5	all	90%	**95.8%**	93.9%
		parts	91.5% 94.3%	95.9% **95.8%**	**99.9%** 93.9%
	6	all	71.1%	99.2%	**99.9%**
		parts	76.1% 81.4%	99.9% 99.2%	**100% 99.9%**
Toy train	7	all	7.8%	**98.1%**	68.4%
		parts	90.1% 17.8% 81.1% 52.5%	**99.2% 99.9% 99.9%** 99.1%	92.0% 68.5% 99.3% **99.2%**
	8	all	5.7%	**94.3%**	91.1%
		parts	74.8% 20.3% 78.2% 51.2%	**100% 100%** 97% **94.3%**	**100% 100% 100%** 91.1%

Table 1. **Instance-level real-world depth benchmark.** While not designed for instance-level articulated object pose estimation, our algorithm is able to achieve comparable performance compare to the state-of-the-art approach and improves the performance for challenging cases such as laptops. AD accuracy is evaluated for both the whole kinematic chain(all) and different parts(parts).

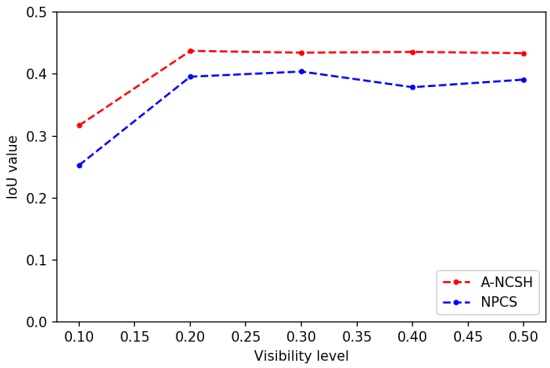


Figure 2. **Performance under different occlusion levels.** Data is collected from part 2 of unseen eyeglasses.

dataset are visualized in Figure 5.

6. Limitation and failure cases

Figure 3 shows typical failure cases of our algorithm. A typical failure mode of our algorithm is the inaccurate prediction under heavy occlusion where one of the object parts is almost not observed. Figure 3 shows one of such cases where one of the eye-glasses temples is almost completely occluded. Also, under the situation of heavy occlusion for prismatic joints, there is considerate ambiguity for ANCSH prediction on the size of the heavily occluded parts, as shown in Figure 3. However, NAOCS representation does not suffer from the size ambiguity, thus leading to a more reliable estimation of the joint state (relative translation distance compare to the rest state) and joint parameters (translation axis).

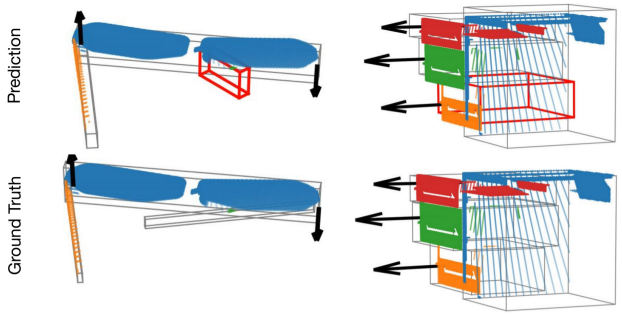


Figure 3. **Failure cases.** Left column shows failure cases on unseen eyeglasses instances, when a part is under heavy occlusion and barely visible. Right column shows the failure case on unseen drawers, when there are shape variations on parts and only the front area of the drawer is visible. The predicted drawer size is bigger than the real size. Although the box prediction is wrong, our method can reliably predict the joint state and joint parameters by leveraging the NAOCS representation.

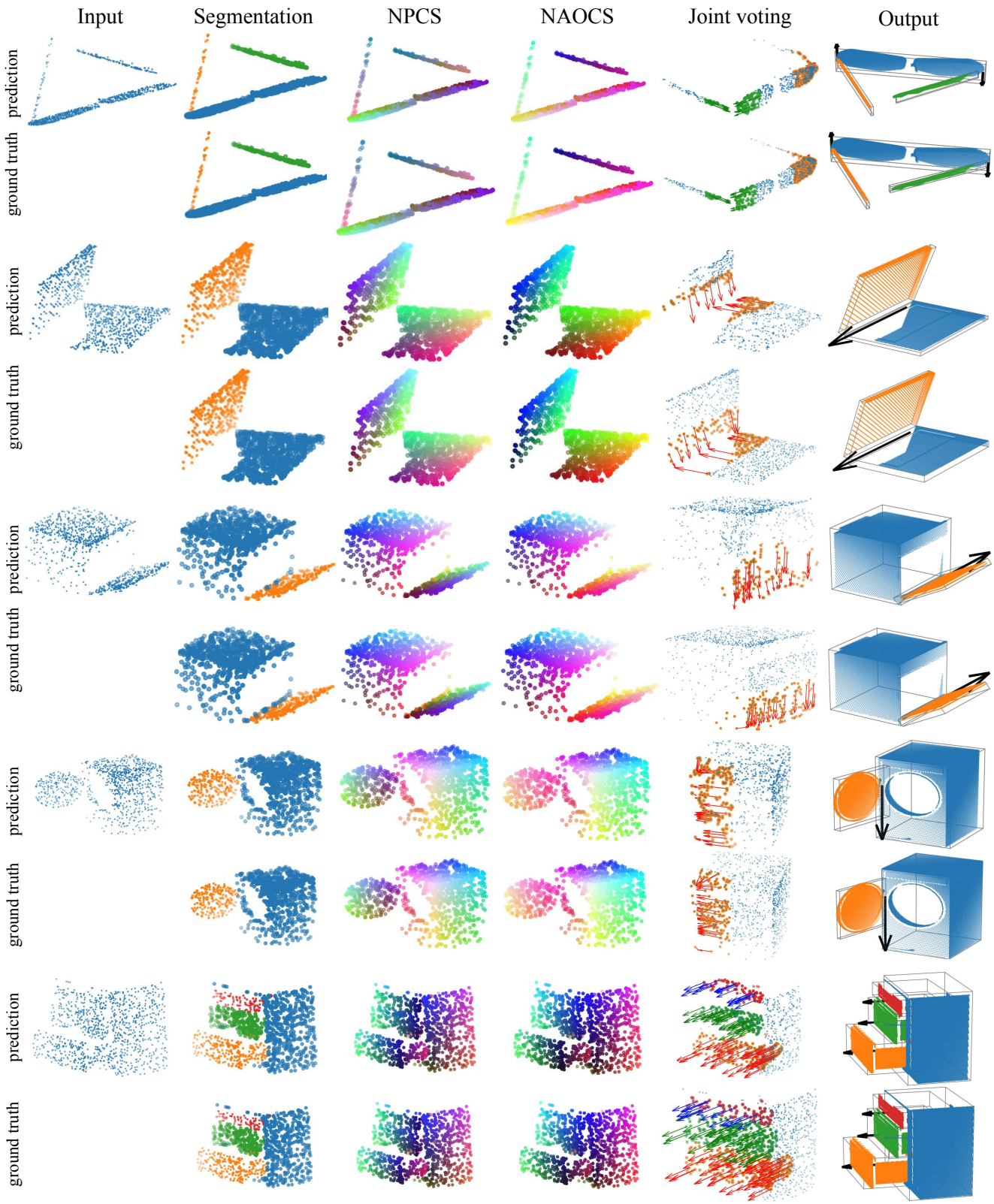


Figure 4. **Additional results on category-level synthetic dataset.** The first column shows the input point clouds; the second column shows our prediction and ground truth part segmentation mask; the third and fourth column show our prediction and ground truth NPCS and NAOCS, where the RGB channels encode the coordinates; the fifth column visualizes joint voting, where the arrows represent offset vectors to rotational hinge for revolute joints and the direction of joint axis for prismatic joints; the sixth column visualizes per part 3D bounding boxes, together with joint parameters.

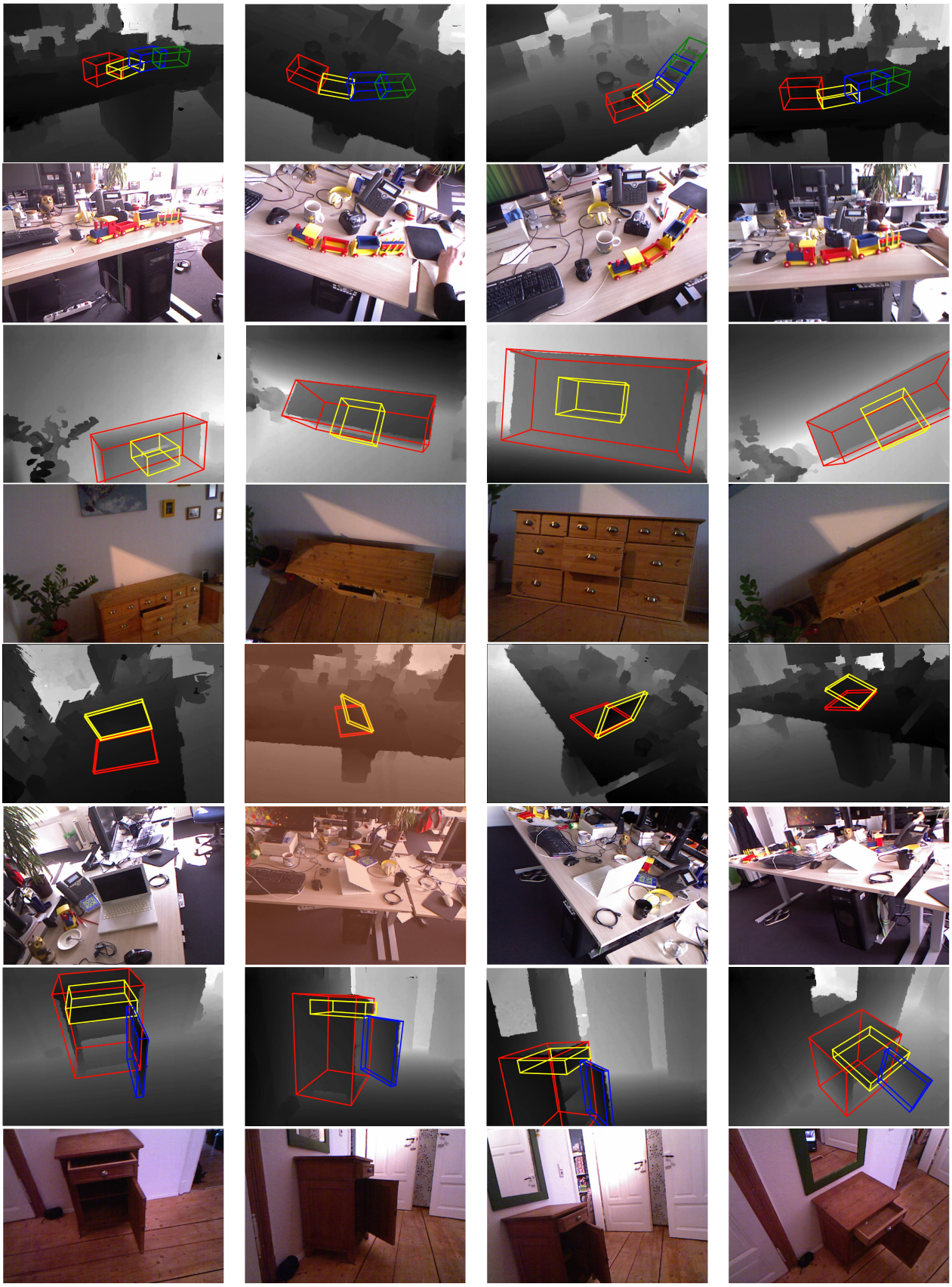


Figure 5. **Additional results on real-world instance-level depth dataset.** More qualitative results on all 4 objects from ICCV2015 Articulated Object Challenge [3] are shown here, with toy train, cupboard, laptop, cabinet from up-pest row to lowest row in order. Only depth images are used for pose estimation, RGB images are shown here for better reference. For each object, we estimate 3D tight bounding boxes to all parts on the kinematic chain, and project the predicted bounding boxes back to the depth image.

References

[1] Eric Brachmann, Alexander Krull, Frank Michel, Stefan Gumhold, Jamie Shotton, and Carsten Rother. Learning 6d object pose estimation using 3d object coordinates. In *European conference on computer vision*, pages 536–551. Springer, 2014. 2

[2] Erwin Coumans and Yunfei Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. http://pybullet.org, 2016–2018. 1

[3] 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, 4

[4] Xiaogang Wang, Bin Zhou, Yahao Shi, Xiaowu Chen, Qinping Zhao, and Kai Xu. Shape2motion: Joint analysis of motion parts and attributes from 3d shapes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8876–8884, 2019. 1

[5] 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, 2020. 1