Semi-Weakly Supervised Object Kinematic Motion Prediction Supplymentary Materials

Gengxin Liu[1] Qian Sun[1] Haibin Huang[2] Chongyang Ma[2] Yulan Guo[3]
Li Yi[4] Hui Huang[1] Ruizhen Hu[1]*
[1]Shenzhen University [2]Kuaishou Technology [3]Sun Yat-sen University [4]Tsinghua University
{gengxin.v.liu,qiansun1006,jackiehuanghaibin,chongyangm,yulan.vision,
ericyi0124,hhzhiyan,ruizhen.hu}@gmail.com

1. Dataset Preparation

In this section, we provide more details about the training data. For GNN and axis selection network, we select 20 categories from the PartNet-Mobility dataset and split the dataset into train and test set by 8:2. For training, we randomly move the mobile parts according to their corresponding ground-truth motion attributes, resulting in a large number of motion variations.

For ANCSH network, we select 5 categories, including refrigerator, dishwasher, table, door_set and scissors from PartNet-Mobility dataset. We use the same train and test sets as described above for training and testing. Since ANCSH requires each category shares the same kinematic chain, we filter out unqualified objects according to their part definitions. Table 1 shows the part definitions of each category, and the number of train and test instances. We use the ground-truth articulation information and object geometry of PartNet-Mobility dataset, and follow the rendering pipeline of [3] to generate train and test data. Specifically, each instance is rendered under 30 different camera views and 31 joints, resulting in a total of 930 depth images for each instance. We also filter out the depth images where some parts are completely occluded. For our semi-weakly supervised setting, we use a threshold to automatically filter out those samples with poor prediction quality, the number of instances of each category that is finally retained are shown in Table 1 (denoted as Augmentation). We see that the number of train sets has been greatly increased due to the addition of PartNet dataset, and thus lead to the significant performance boosts of ANCSH.

2. Details of Axis Selection Network

The axis selection network is shown in Figure 1, which consists of a DGCNN encoder and a fully connected layer. The DGCNN encoder of the axis selection network is illustrated in Figure 2, which is mainly composed of the Edge-Conv [6] module. The Fully Connected Layer of the axis selection network is illustrated in Figure 3, which is composed of the Linear layer, BatchNorm [1] and Dropout [5] modules.

*Ruizhen Hu is the corresponding author.

Category	Part0	Part1	Part2	Train	Test	Augmentation
refrigerator	Base	Left/Upper door	Right/Lower door	22	3	67
dishwasher	Base	Door	-	34	5	91
table	Base	Drawer	-	25	5	90
door_set	Base	Door	-	8	3	7
scissors	Handle	Handle	-	37	6	62

Table 1. Details about the number of samples per class.

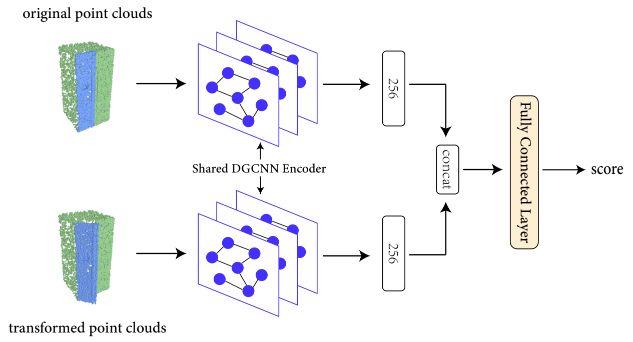

Figure 1. The input to the network includes the original point clouds and the transformed point clouds after applying the candidate motion, associated with point-wise one-hot label indicating whether the point belongs to the mobile or reference part. The output is a feasibility score for the relative motion.

3. Implementation details

We implement our method with the PyTorch [4] framework. We train our graph neural network for 400 epochs with learning rate 1e-4. We train the axis selection network

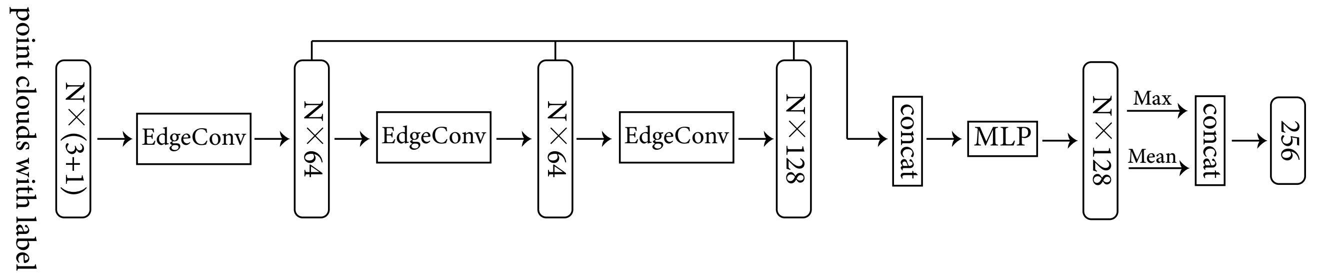

Figure 2. DGCNN encoder used in axis selection network

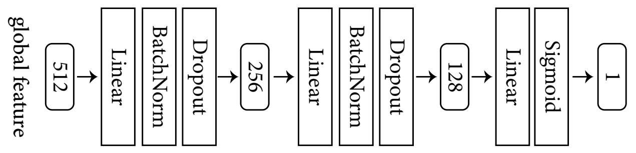

Figure 3. Fully connected layer used in axis selection network

for 2000 epochs with learning rate 3e-4. Both network are trained using the Adam [2] optimizer. For semi-weakly supervised learning, we use the code from [3], and use the same default hyperparameters for different setting.

To train the axis selection network, we use the ground-truth motion parameters and sample the motion amount according to the ground-truth motion limit to generate positive transformed point clouds. We sample a random edge of the movable part's OBB that is different from the GT axis for negative examples generation. We show some positive and negative transformed point clouds in Figure 4. Note that, for *PR* type joint (such as the bottle in the second row), we found that it is better to set the rotation angle to 60∼90° to generate the transformed point clouds which make the difference between the original point clouds and the transformed point clouds larger, making it easier for the network to distinguish.

Testing. For testing, we use the candidate motion parameters to transform the point clouds. However, there is no ground-truth motion limit for test data, we heuristically set a motion amount list for different joints, as shown in Table 2. For each candidate motion parameter, we traverse the corresponding motion amount list and use the motion amount to transform the point clouds, and the feasibility score of the motion parameter is the average of feasibility score of all the motion amount.

4. More Qualitative Results

Figure 5 shows more comparison results on pre-segmented shapes. We see that *OBB_fine* always predict high quality motion parameters compare to other method,

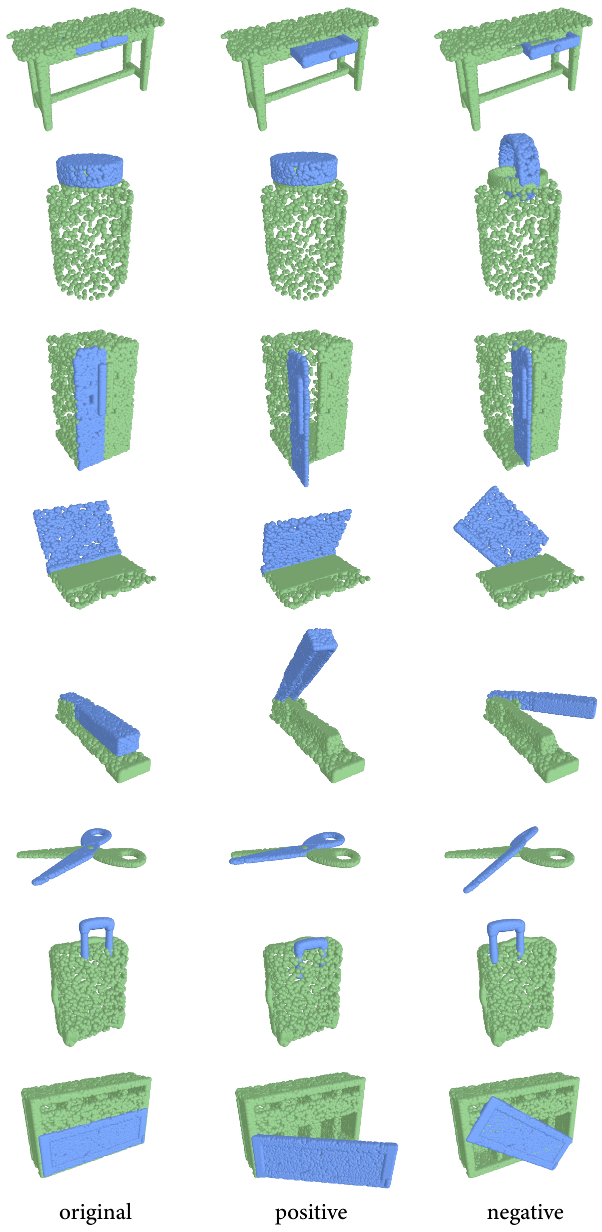

Figure 4. Example of positive and negative transformed point clouds for training axis selection network.

joint	motion amounts
P	-0.38, -0.27, -0.16, 0.16, 0.27, 0.38
R	-80°, -60°, -45°, -30°, 30°, 45°, 60°, 80°
PR	(-0.02, 60°), (-0.02, 60°), (0.02, 90°), (0.02, 90°)

Table 2. Motion amount list for different types of joints. *P*, *R* and *PR* represent the prismatic, revolute and both joints.

which demonstrates that the fine characterization of the motion type and the OBB priors can help to refine the motion parameters. Figure 6 shows some pseudo labels predicted by our method on PartNet. The first and third row are the input point clouds with fine-grained segmentation, and the second and last row are the merge segmentation and motion parameters predicted by our method. Our method correctly merge scattered parts into one movable part and predict reasonable motion parameters, which can be used to boost the performance of previous method designed for kinematic motion prediction.

References

[1] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015. 1

[2] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 2

[3] Xiaolong Li, He Wang, Li Yi, Leonidas J Guibas, A Lynn Abbott, and Shuran Song. Category-level articulated object pose estimation. In *CVPR*, pages 3706–3715, 2020. 1, 2

[4] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019. 1

[5] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014. 1

[6] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *Acm Transactions On Graphics (tog)*, 38(5):1–12, 2019. 1

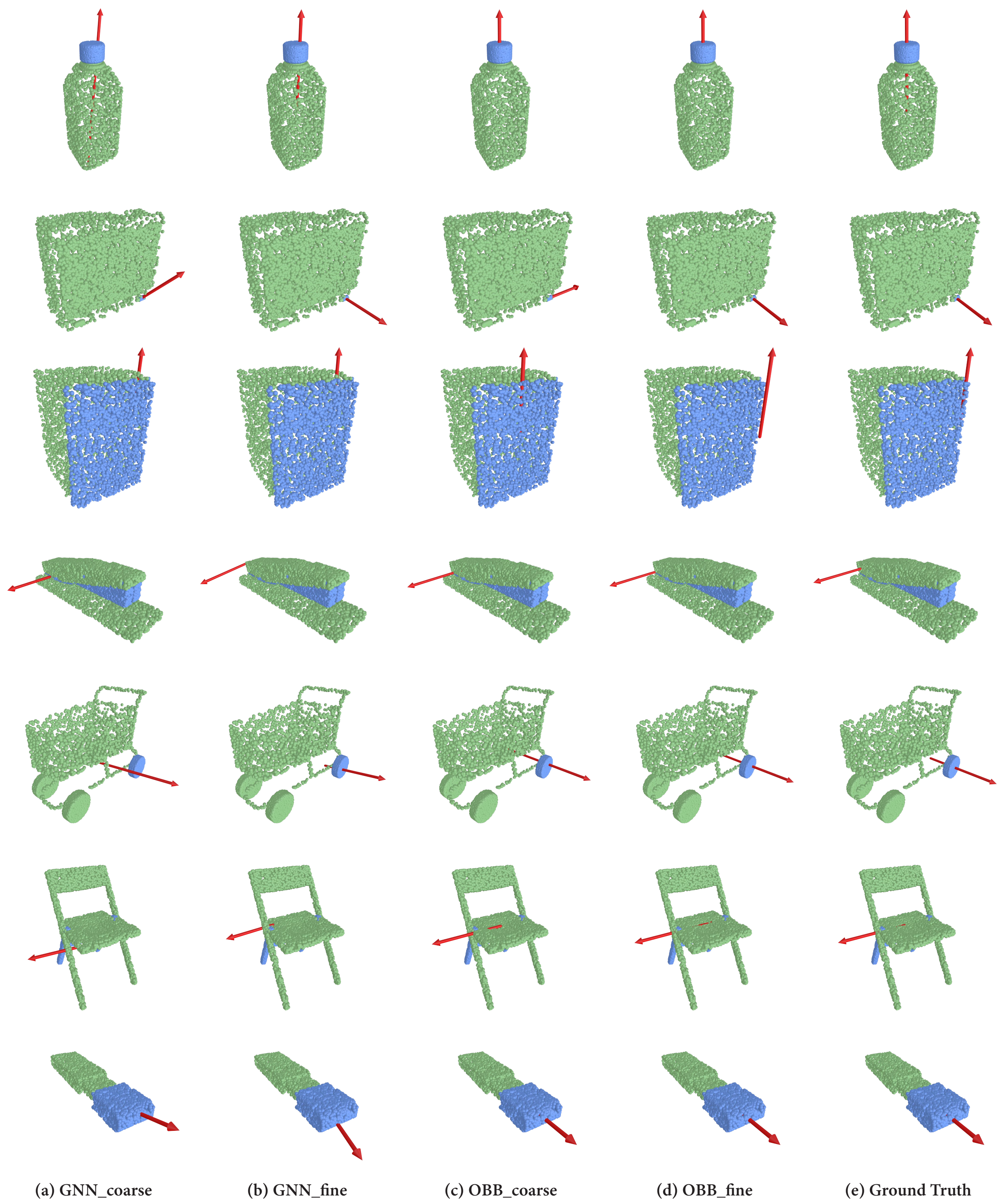

Figure 5. Visual comparison of results obtained by *GNN_coarse*, *GNN_fine*, *OBB_coarse* and *OBB_fine* on pre-segmented point clouds. *OBB_fine* always predict high quality motion parameters compare to other method owing to the fine characterization of the motion type and the use of OBB priors and introduction of the interaction region.

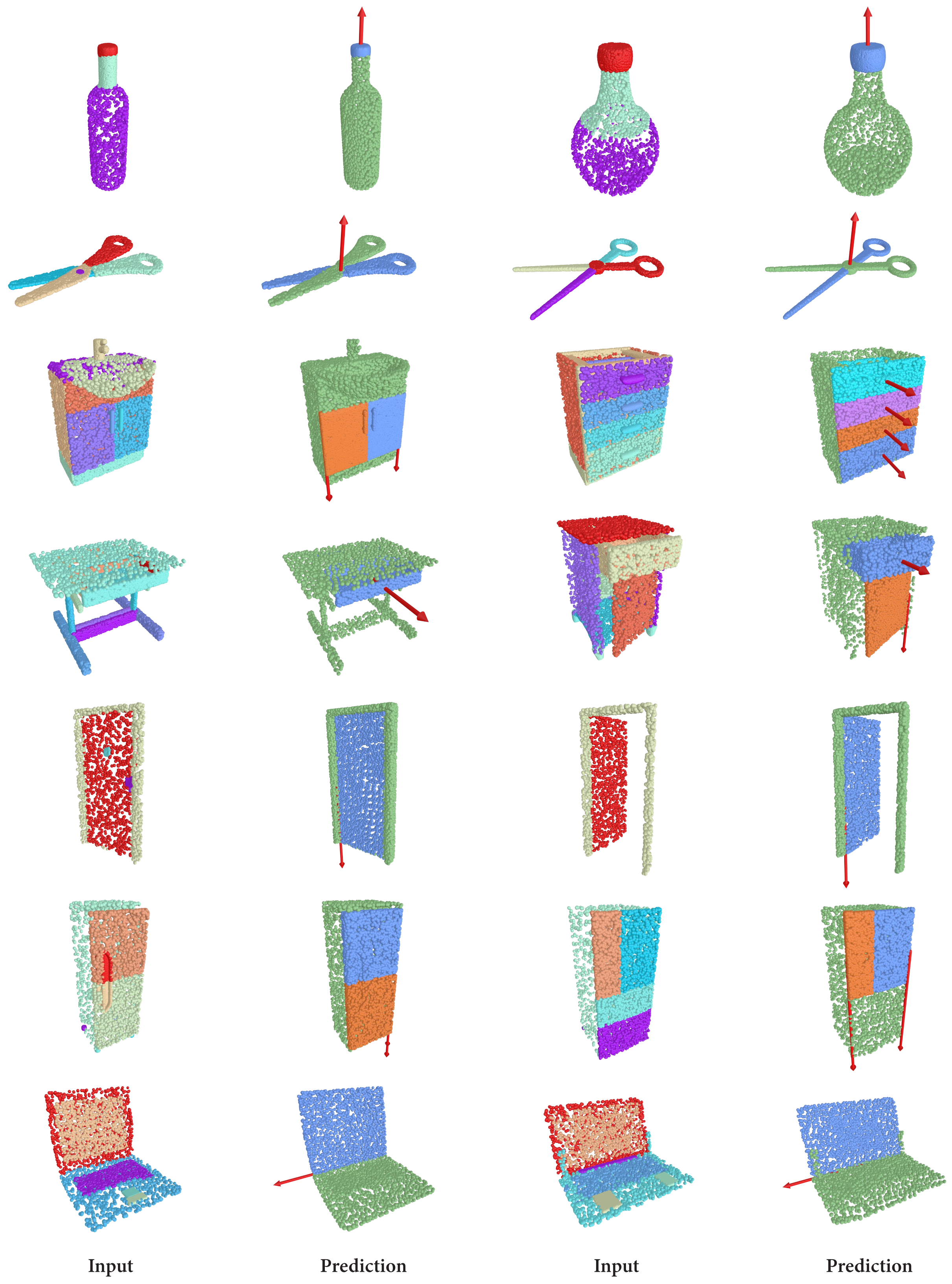

Figure 6. Pseudo labels predicted by our method on PartNet. The first and third column are the input point clouds with fine-grained segmentation, and the second and last column are the merge segmentation and motion parameters predicted by our method