

# SELF-SUPERVISED CATEGORY-LEVEL ARTICULATED OBJECT POSE ESTIMATION WITH PART-LEVEL SE(3) EQUIVARIANCE

## Supplementary Materials

**Anonymous authors**

Paper under double-blind review

The appendix is organized as follows:

- Proofs and further explanations on the method
  - Proof of part-Level equivariant property of the pose-aware point convolution module (sec. A.2).
  - Further explanations on some method components (sec. A.3).
- Experiments
  - Data preparation (sec. B.1).
  - Implementation details (sec. B.2).
  - Implementation details for baselines (sec. B.3).
  - Experiments on partial point clouds (sec. B.4).
  - Additional comparisons and applications (sec. B.5).
  - Robustness to input data noise (sec. B.6).
  - Visualization of part-level equivariant features (sec. B.7).
  - Evaluation strategy for category-level articulated object poses (sec. B.8).
- Discussion on part symmetry (sec. C)

## A METHOD DETAILS

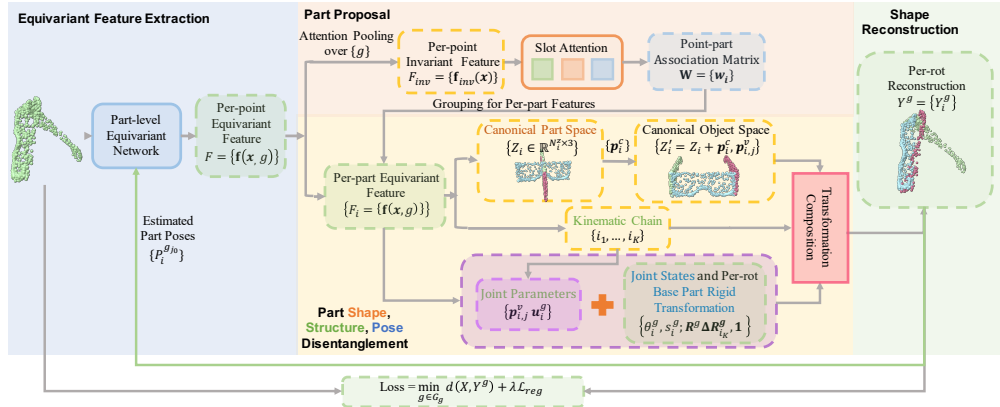


Figure 1: Overview of the proposed self-supervised articulated object pose estimation strategy. The method takes a complete or partial point cloud of an articulated object as input, factorizes canonical shapes, object structure, and the articulated object pose from it. The network is trained by a shape reconstruction task. Part-level SE(3) equivariant features are learned by iterating between part pose estimation and pose-aware equivariant point convolution. Green lines ( $\leftarrow$ ) denote procedures for feeding the estimated part poses back to the pose-aware point convolution module.

### A.1 OVERVIEW

We provide an detailed diagram of our self-supervised learning strategy in Figure 1.

## A.2 PROOF OF PART-LEVEL EQUIVARIANT PROPERTY OF THE POSE-AWARE POINT CONVOLUTION MODULE

In this section, we prove the part-level equivariant property of the designed pose-aware point convolution module:

$$(\mathcal{F} * h_1)(x_i, g) = \sum_{P_j^{-1}x_j \in \mathcal{N}_{P_i^{-1}x_i}^c} \mathcal{F}(x_j, g\mathbf{R}_i\mathbf{R}_j^{-1})h_1(g(x_i - P_iP_j^{-1}x_j)), \quad (1)$$

where  $P_i$  and  $P_j$  are the (estimated) pose of  $x_i$  and  $x_j$  from the canonical object space to the camera space respectively,  $\mathbf{R}_i$  and  $\mathbf{R}_j$  are the (estimated) rotations of point  $x_i$  and point  $x_j$  from the canonical object space to the camera space respectively,  $\mathcal{N}_{P_i^{-1}x_i}^c$  denotes the set of point  $x_i$ 's neighbours in the canonical object space. Note that the neighbourhood set  $\mathcal{N}_{x_i}$  in the Equation ?? represents the neighbourhood of  $x_i$  in the camera space with points in which belong to  $x_i$ 's neighbourhood in the canonical object space, i.e.  $\mathcal{N}_{P_i^{-1}x_i}^c \cdot \mathcal{N}_{x_i}$  would vary as  $x_i$ 's pose changes, while  $\mathcal{N}_{P_i^{-1}x_i}^c$  keeps the same.  $\{x_j | x_j \in \mathcal{N}_{x_i}\} = \{x_j | P_j^{-1}x_j \in \mathcal{N}_{P_i^{-1}x_i}^c\}$ .

To prove the part-level equivariance of  $(\mathcal{F} * h_1)(x_i, g)$ , we need to prove 1)  $(\mathcal{F} * h_1)(x_i, g)$  is invariant to the rigid transformation of point  $x_i$ 's each neighbouring point  $x_j$ ; 2)  $(\mathcal{F} * h_1)(x_i, g)$  is equivariant to the rigid transformation of  $x_i$  itself.

We then prove those properties for the continuous convolution operations,  $(\mathcal{F} * h_1)(x_i, g) = \int_{x_j \in \mathbb{R}^3} \mathcal{F}(x_j, g\mathbf{R}_i\mathbf{R}_j^{-1})h_1(g(x_i - P_iP_j^{-1}x_j))$ .

**Theorem 1.** *The continuous operation  $(\mathcal{F} * h_1)(x_i, g) = \int_{x_j \in \mathbb{R}^3} \mathcal{F}(x_j, g\mathbf{R}_i\mathbf{R}_j^{-1})h_1(g(x_i - P_iP_j^{-1}x_j))$  is invariant to each arbitrary rigid transformation  $\Delta P_j = (\Delta \mathbf{R}_j \in SO(3), \Delta \mathbf{t}_j \in \mathbb{R}^3)$  of  $(x_j \forall x_j \in \mathbb{R}^3, x_j \neq x_i)$  of  $x$ 's neighbouring point  $x_j$ .*

*Proof.* To prove the invariance of  $(\mathcal{F} * h_1)(x_i, g)$ , we need to prove that  $\forall x_j \in \mathbb{R}^3, x_j \neq x_i, \forall \Delta P_j \in SE(3), \mathbf{R}_j' = \Delta \mathbf{R}_j \mathbf{R}_j$ , we have

$$\Delta P_j(\mathcal{F} * h_1)(x_i, g) = (\mathcal{F} * h_1)(x_i, g).$$

Let  $x_j' = \Delta P_j x_j, P_j' = \Delta P_j P_j$ , then we have,

$$\begin{aligned} \Delta P_j(\mathcal{F} * h_1)(x_i, g) &= \int_{x_j' \in \mathbb{R}^3} \mathcal{F}(x_j', g\mathbf{R}_i\mathbf{R}_j'^{-1})h_1(g(x_i - P_iP_j'^{-1}x_j')) \\ &= \int_{x_j \in \mathbb{R}^3} \mathcal{F}(\Delta P_j x_j, g\mathbf{R}_i\mathbf{R}_j^{-1}\Delta \mathbf{R}_j^{-1})h_1(g(x_i - P_iP_j^{-1}\Delta P_j^{-1}\Delta P_j x_j)) \\ &= \int_{x_j \in \mathbb{R}^3} \mathcal{F}(\Delta \mathbf{R}_j x_j, g\mathbf{R}_i\mathbf{R}_j^{-1}\Delta \mathbf{R}_j^{-1})h_1(g(x_i - P_iP_j^{-1}x_j)) \\ &= \int_{x_j \in \mathbb{R}^3} \mathcal{F}(x_j, g\mathbf{R}_i\mathbf{R}_j^{-1})h_1(g(x_i - P_iP_j^{-1}x_j)) \\ &= (\mathcal{F} * h_1)(x_i, g). \end{aligned}$$

**Theorem 2.** *The continuous operation  $(\mathcal{F} * h_1)(x_i, g) = \int_{x_j \in \mathbb{R}^3} \mathcal{F}(x_j, g\mathbf{R}_i\mathbf{R}_j^{-1})h_1(g(x_i - P_iP_j^{-1}x_j))$  is equivariant to the rigid transformation  $\Delta P_i = (\Delta \mathbf{R}_i \in SO(3), \Delta \mathbf{t}_i \in \mathbb{R}^3)$  of  $x_i$ .*

*Proof.* To prove that  $(\mathcal{F} * h_1)(x_i, g)$  is equivariant to the rigid transformation of  $x_i$ , we need to prove that  $\forall \Delta P_i \in SE(3)$ , we have

$$\Delta P_i(\mathcal{F} * h_1)(x_i, g) = (\Delta \mathbf{R}_i \mathcal{F} * h_1)(x_i, g).$$

It can be proved by

$$\begin{aligned}
\Delta P_i(\mathcal{F} * h_1)(x_i, g) &= (\mathcal{F} * h_1)(\Delta P_i x_i, g \Delta \mathbf{R}_i) \\
&= \int_{x_j \in \mathbb{R}^3} \mathcal{F}(x_j, g \Delta \mathbf{R}_i \mathbf{R}_j \mathbf{R}_j^{-1}) h_1(g(\Delta P_i x_i - \Delta P_i P_j^{-1} x_j)) \\
&= \int_{x_j \in \mathbb{R}^3} \mathcal{F}(x_j, (g \Delta \mathbf{R}_i) \mathbf{R}_j \mathbf{R}_j^{-1}) h_1((g \Delta \mathbf{R}_i)(x_i - P_j^{-1} x_j)) \\
&= (\Delta \mathbf{R}_i \mathcal{F} * h_1)(x_i, g).
\end{aligned}$$

### A.3 FURTHER EXPLANATIONS ON SOME METHOD COMPONENTS

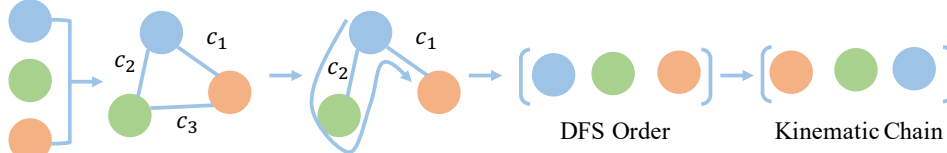


Figure 2: Kinematic chain prediction procedure (an example of the object containing three parts).

**Kinematic Chain Prediction.** The kinematic chain is predicted as an invariant property from per-part invariant features to describe part articulation transformation order. It is predicted through the following four steps: 1) Predict an adjacency confidence value  $c_{i,j}$  for each part pair  $(i, j)$ ; 2) Construct an fully-connected adjacency confidence graph  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$  based on predicted confidence values, with all parts as its nodes and predicted confidence values as edge weights; 3) Find a maximum spanning tree from the constructed graph  $\mathcal{G}$ :  $\mathcal{T} = (\mathcal{V}, \mathcal{E}_{\mathcal{T}})$ ; 4) Calculate the DFS visiting order of  $\mathcal{T}$  and take the inverse visiting order as the predicted kinematic chain. We draw the prediction procedure in Figure 2.

**Invariant/Equivariant Features for Prediction.** Given per-part equivariant feature output from the feature backbone  $F_i$ , its equivariant feature for equivariant properties prediction is further calculated from an SO(3)-PointNet, *i.e.*  $\hat{F}_i = \text{SO(3)-PointNet}(X_i, F_i)$ . Its invariant feature for invariant properties prediction is then computed through a max-pooling operation:  $F_i^{\text{inv}} = \text{Max-Pooling}(\hat{F}_i)$ .

**Joint Axis Orientation.** We assume that all joints’ axis orientations in one shape are consistent. Thus, in practice, we set the orientation of all joints to the same predicted orientation, *i.e.*  $\mathbf{u}_i^g \leftarrow \mathbf{u}_{i_m}^g, \forall (i, j) \in \mathcal{E}_{\mathcal{T}}, \forall g \in G_g$ , where  $(i_m, j_m)$  is set to the part pair connected to the tree root. Saying  $(i, j) \in \mathcal{E}_{\mathcal{T}}$ , we mean a directional edge from part  $i$  to part  $j$ . In the node pair  $(i, j) \in \mathcal{E}_{\mathcal{T}}$ , node  $i$  is deeper than node  $j$  in the tree  $\mathcal{T}$ . It indicates that node  $i$ ’s subtree should rotate around the joint  $\mathbf{u}_i^g$  passing through the joint between  $i$  and  $j$ .

**Iterative Pose Estimation.** Our pose-aware equivariant point convolution module requires per-point pose as input. Due to our self-supervised setting where input poses are not assumed, we adopt an iterative pose estimation strategy. Through this design, we can improve the quality of part-level equivariant features gradually by feeding back estimated poses in the last iteration to the pose-aware point convolution module in the current iteration. It is because that more accurate input per-point poses would lead to better “part-level” SE(3) equivariant features considering the nature of our pose-aware point convolution. In practice, we set per-point poses to identity values in the first iteration due to the lack of estimated poses, *i.e.*  $P_0 = (\mathbf{R}_0, \mathbf{t}_0)$ .

**Canonical Part Spaces, Canonical Object Space, Camera Space.** For each part, the canonical part space normalizes its pose. For each object, the canonical object space normalizes its object orientation, articulation states. Camera space denotes the observation space. Each part shape in the canonical part space is its canonical part shape. Each object shape with articulation states canonicalized is called its canonical object shape. Canonical *spaces* are category-level concepts, while canonical shapes are instance level concepts. Figure 3 draws the relationship between such three spaces mentioned frequently in our method.

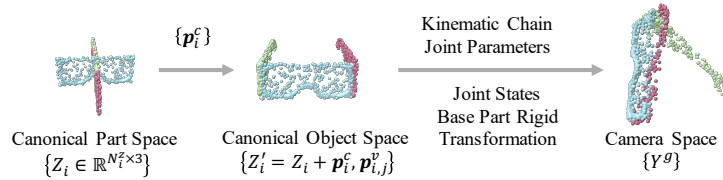


Figure 3: The relationship between our three crucial spaces: the canonical part spaces, the canonical object space, and the camera space.

**Partial Point Clouds.** The loss function used in Li et al. (2021) for partial point clouds is the unidirectional Chamfer Distance. Using this function can make the network aware of complete shapes by observing partial point clouds from different viewpoints. Such expectation can be achieved for asymmetric objects if the viewpoint could cover the full  $SO(3)$  space. However, we restrict the range of the viewpoint when rendering partial point clouds for articulated objects to make each part visible. Such restriction would result in relatively homogeneous occlusion patterns. Therefore, we choose to use unidirectional Chamfer Distance only for certain categories such as Safe when tested on partial point clouds.

**Equivariant/Invariant Properties of the Designed Modules.** The designed method wishes to use part-level  $SE(3)$ -equivariance to reduce the difficulty of factorizing part pose and part shape. Exact part-level equivariant features can make those modules meet our expectations. However, due to the approximate  $SE(3)$ -equivariance of the employed feature backbone and the estimated part pose that may not be very accurate, we cannot expect such invariance/equivariance for them. For instance, if we do not consider part kinematic constraints, the part shape reconstruction module and the part-assembling parameters prediction module should be invariant to  $K$  rigid transformations in the quotient group  $(SE(3)/G_A)(SE(3))^{K-1}$  if using global equivariant features, while it should be invariant to the rigid transformation in the quotient group  $SE(3)/G_A$  if using part-level equivariant features given correct part pose estimation. Similarly, the pivot point prediction module should be invariant to two rigid transformations in the quotient group  $(SE(3)/G_A)^2$  if using part-level equivariant features. Part-level equivariance design could reduce the difficulty of a network doing factorization, which may count as a reason for its effectiveness.

## B EXPERIMENTS

### B.1 DATA PREPARATION

**Data Collection.** We choose seven categories from three different datasets, namely Oven, Washing Machine, Eyeglasses, Laptop (S) with revolute parts from Shape2Motion Wang et al. (2019), Drawer with prismatic parts from SAPIEN Xiang et al. (2020), Safe and Laptop (R) with revolute parts from HOI4D Liu et al. (2022).

The first five datasets are selected according to previous works on articulated object pose estimation or part decomposition Li et al. (2020); Kawana et al. (2021). To further test the effectiveness of our method on objects collected from the real world, we choose two more categories (Safe and Laptop (R)) from a real dataset Liu et al. (2022).

**Data Splitting.** We split our data according to the per-category data split approach introduced in Li et al. (2020). Note that not all shapes in a category are used for training/testing. Incomplete shapes or instances whose canonical articulation states are inconsistent with other shapes are excluded from experiments. Per-category train/test splits are listed in Table 1.

**Data Preprocessing.** For each shape, we generate 100 posed shapes in different articulation states.

Then for complete point clouds, we randomly generate 10 rotated samples from each articulated posed object. When generating articulated posed objects, we would add restrictions on valid articulation state ranges. For Oven, Safe, and Washing Machine, the valid degree range of their lids is  $[45^\circ, 135^\circ)$ .



Table 1: Per-category data splitting.

	Oven	Washing Machine	Eyeglasses	Laptop (S)	Safe	Laptop (R)	Drawer
#Total	32	41	42	82	30	50	30
#Train	28	36	37	73	26	44	24
#Test	4	5	5	9	4	6	6

For Eyeglasses, the range of the degree between two legs and the frame is set to  $[0^\circ, 81^\circ]$ . For Laptop (S) and Laptop (R), the range of the degree between two parts is set to  $[9^\circ, 99^\circ]$ .

For partial point clouds, we render depth images of complete object instances using the same rendering method described in Li et al. (2021). The difference is that we manually set a viewpoint range for each category to ensure that all parts are visible in the rendered depth images. For each articulated posed shape, we render 10 depth images for it. The dataset will be made public.

**Data samples visualization.** In Figure 4, we provide samples of training and test shapes for some categories for an intuitive understanding w.r.t. intra-category shape variations. Such variations mainly come from part geomtry (*e.g.* Eyeglasses frames, Oven bodies, Laptop) and part size (*e.g.* Washing Machine, Laptop).

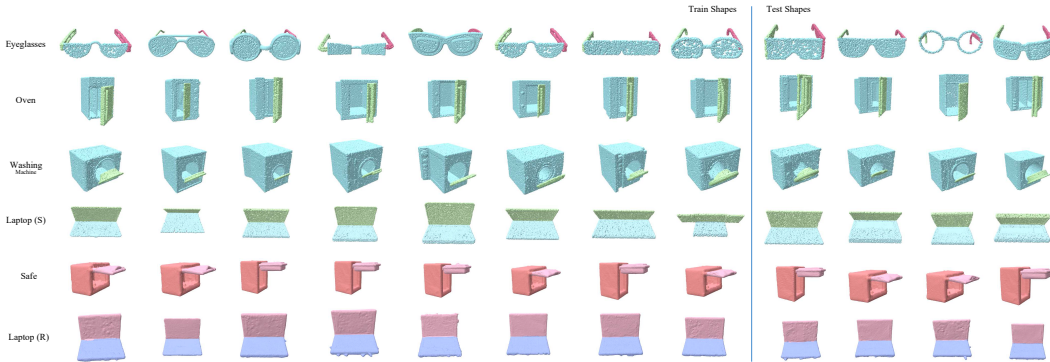


Figure 4: Samples of training and test shapes.

## B.2 IMPLEMENTATION DETAILS

**Architecture.** For point convolution, we use a kernel-rotated version kernel point convolution (KPConv Thomas et al. (2019)) proposed in EPN Chen et al. (2021). The size of the (one) convolution kernel is determined by the number of anchor points and the feature dimension. In our implementation, we use 24 anchor points. Feature dimensions at different convolution blocks are set to 64, 128, and 512 respectively.

**Training Protocol.** In the training stage, the learning rate is set to 0.0001, which is decayed by 0.7 every 1000 iterations. The model is trained for 10000 steps with batch size 8 on all datasets. We use the self-supervised reconstruction loss to train the network, with the weight for joint regularization  $\lambda$  set to 1.0 empirically. We use Adam optimizer with  $\beta = (0.9, 0.999)$ ,  $\epsilon = 10^{-8}$ .

**Software and Hardware Configurations.** All models are implemented by PyTorch version 1.9.1, torch\_cluster version 1.5.1, torch\_scatter version 2.0.7, pyrender version 0.1.45, trimesh version 3.2.0, and Python 3.8.8. All the experiments are conducted on a Ubuntu 20.04.3 server with 8 NVIDIA GPUs, 504G RAM, CUDA version 11.4.

## B.3 IMPLEMENTATION DETAILS FOR BASELINES

**NPCS Li et al. (2020).** The NPCS’s original version Li et al. (2020) trains a network for category-level articulated object pose estimation in a supervised manner. It utilizes a PointNet++ Qi et al.

(2017) to regress three kinds of information and a set of pre-defined normalized part coordinate spaces. Then in the evaluation process, the RANSAC algorithm is leveraged to calculate the rigid transformation of each part from its predicted normalized part coordinates to the shape in the camera space. To apply NPCS in our experiments, we make the following two modifications: 1) We change the backbone used in NPCS from PointNet++ to EPN. We further add supervision on its rotation mode selection process for the major rotation matrix prediction as does in Chen et al. (2021). 2) We add a joint axis orientation prediction branch and a pivot point regression branch for joint parameters estimation. Such two prediction branches act on the global shape feature corresponding to the selected rotation mode and predict a residual rotation and a translation for estimation. By applying the major rotation matrix of the selected mode, we could then arrive at the joint axis orientations and pivot points in the camera space.

**Oracle ICP.** To apply ICP on the articulated object pose estimation, we introduce Oracle ICP. Oracle ICP registers each ground-truth part from the template shape to the observed shape iteratively. We randomly select 5 segmented shapes from the train set to register on each test shape. For complete point clouds, we first centralize the part shape from both of the template shape and the observed shape, we then iteratively register the template part shape to the observed part shape under 60 initial hypotheses. For partial point clouds, we iteratively register the template part shape to the observed part shape under 60 initial hypotheses together with 10 initial translation hypotheses. The one that achieves the smallest inlier RMSE value is selected as the registration result. After each registration, we assign per-point segmentation label as the label of the nearest part. Therefore, we also treat Oracle ICP as one of our segmentation baseline.

**BSP-Net Chen et al. (2020).** BSP-Net reconstructs an input shape using implicit fields as the representation by learning to partition the shape using three levels of representations, from planes to convexes, and further to the concave. Indices of reconstructed convexes are consistent across different shapes in the same category. Thus, we can map from each convex index to a ground-truth segmentation label. The relationship can then help us get segmentations for test shapes. The mapping can then help us segment each test shape by assigning each convex to its corresponding part segment. The intra-category convex partition consistency further provide cross-instance aligned part segmentations. However, due to the global pose variations of our data, such convex index consistency may not be observed when directly applying BSP-Net on our data. Thus, we propose to improve the evaluation process of BSP-Net to mitigate this problem. Specifically, for each test shape, we find a shape from the train set that is the most similar to the current test shape. Then, we directly use its convex-segmentation mapping relationship to get segments for the test shape. The segments are then used to calculate the segmentation IoU for the test shape.

**NSD Kawana et al. (2020).** Neural Star Domain Kawana et al. (2020) decomposes shapes into parameterized primitives. To test its part segmentation performance on our data with arbitrary global pose variations, we adopt the evaluation strategy similar to that used for BSP-Net.

#### B.4 EXPERIMENTS ON PARTIAL POINT CLOUDS

In this section, we present the experimental results on rendered partial point clouds of our method and baseline methods.

**Articulated Object Pose Estimation.** In Table 2, we present the part pose estimation and joint parameter prediction results of our method and baseline methods. Compared to the pose estimation results of different models achieved on complete point clouds, our model can sometimes outperform the **supervised** NPCS baseline (using EPN as the backbone), such as better part pose estimations and joint parameter prediction results on the Laptop (S) dataset.

In Figure 6, we draw some samples for a qualitative evaluation. Moreover, we also provide the visualization of all categories for complete point clouds in Figure 5. In the figure, the point cloud distance function used for Safe is unidirectional Chamfer Distance, while that used for others is still bidirectional Chamfer Distance. Using unidirectional Chamfer Distance can relieve the problem of joint regularization on partial point clouds to some extent. It is because that in this way the point cloud completion could be naturally enforced. For instance, reconstruction results for the Safe category are drawn in Figure 6. However, points that are not mapped to any point in the input shape will also

affect the point-based joint regularization. For simple shapes, using bidirectional Chamfer Distance could also sometimes make the decoder decode complete part shapes, e.g. reconstructions for Laptop (R). As for the reconstructed reference shape in the canonical object space, better joint predictions would lead to better global shape alignment. For instance, we can observe that the angle between two parts of Laptop (R) and Laptop (S) is relatively consistent across shapes with different articulation states. Joint regularization enforcing the connectivity between two adjacent parts both before and after the articulated transformation in the canonical object space. It could then help make the joint behave like a real joint, based on which we the “lazy” network tends to decode part shapes with consistent orientations. However, there is a degenerated solution where the decoded rotation angle is put near to zero. In that case, the joint regularization term could be satisfied by decoding “twisted” part shapes. Since the decoded angle is near zero, the connectivity between two parts will not be broken when rotating along the decoded joint. Sometimes, decoding angles near to zeros is a local minimum that the optimization process gets stuck in. At that time, the regularization loss term is a large one but the decoded joint parameters are not optimized in the correct direction by the network. The reconstructed shapes in the canonical object space do not have consistent angles, e.g. Washing Machine and Safe drawn in Figure 6.

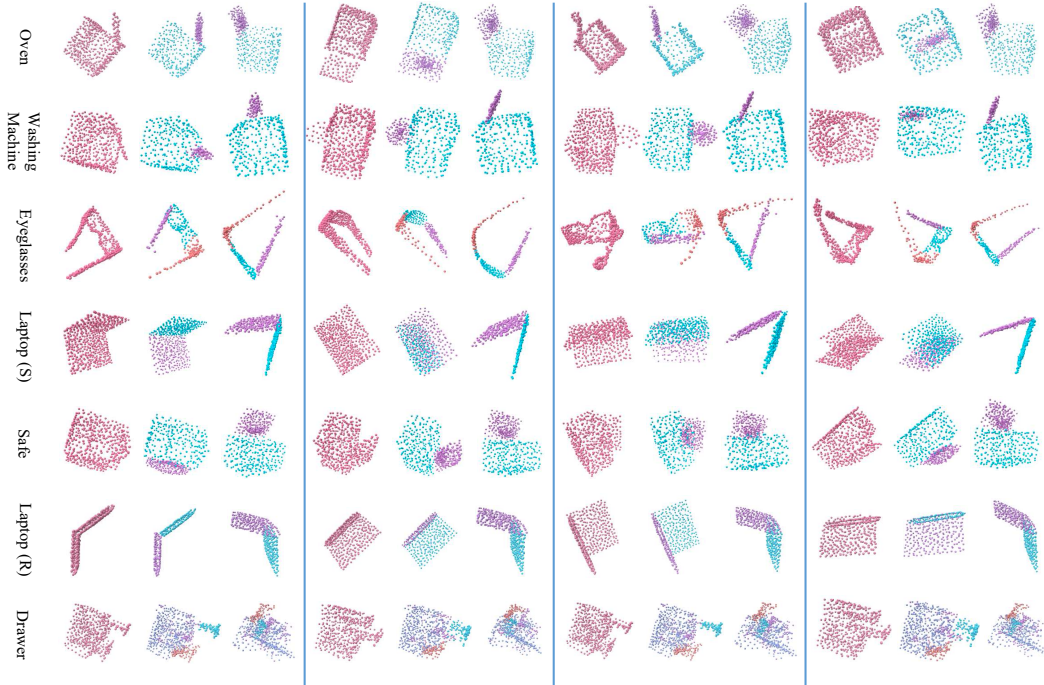


Figure 5: Visualization for experimental results on complete point clouds. Shapes drawn for every three shapes from the left side to the right side are the input point cloud, reconstructions, and the predicted canonical object shape. **We put drawers in an aligned space just for better visualization.** Their global pose may vary when feeding into the network. Please zoom in for details.

**Part Segmentation.** In Table 3, we evaluate the segmentation performance of our method. BSP-Net is not compared here since it requires mesh data for pre-processing, which is not compatible with the rendered partial point clouds. Oracle ICP uses real segmentation labels to register each part from the example shape to the observed shape. Despite this, it can still not achieve satisfactory estimation results due to shape occlusions and part-symmetry-related pose ambiguity issues.

**Shape Reconstruction.** In Table 4, we evaluate the shape reconstruction performance of our method. The part-by-part reconstruction strategy used by our method can outperform the EPN-based whole shape reconstruction strategy in most of those categories except for Washing Machine. One possible reason is the poor segmentation performance of our model on shapes in the Washing Machine category.

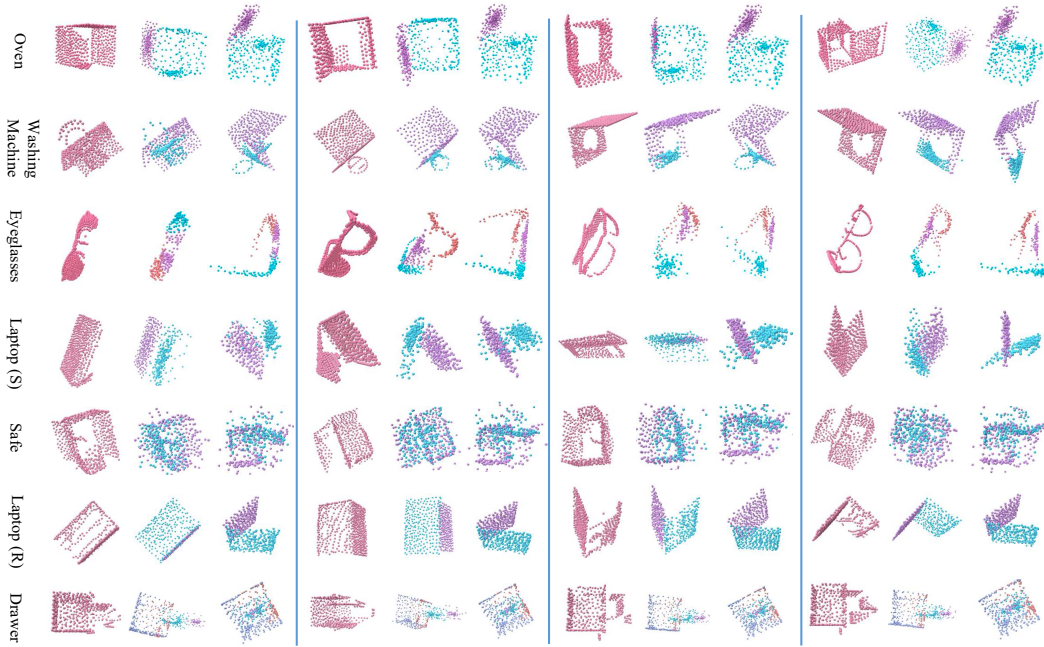


Figure 6: Visualization for experimental results on **partial point clouds**. Shapes drawn for every three shapes from the left side to the right side are the input point cloud, reconstructions, and the predicted canonical object shape. Please zoom in for details.

## B.5 ADDITIONAL COMPARISONS AND APPLICATIONS

**Comparison with Other Baselines.** We compare our method with other two baselines that are not discussed in the main body in Table 5.

Firstly, we use KPConv Thomas et al. (2019) as NPCS’s feature backbone (denoted as “NPCS-KPConv”) and test its performance on our data with arbitrary global pose variation. We can see that NPCS of this version performs terribly compared to our unsupervised method. NPCS estimates part poses by estimating the transformation from estimated NPCS coordinates and the observed shape. It therefore requires invariant NPCS predictions to estimate category-level part poses. However, such prediction consistency may not be easily achieved for input shapes with various global pose variations.

The second one is Oracle EPN, where we assume ground-truth part segmentation labels and use EPN to estimate the pose for each individual part. Despite in such oracle setting, EPN cannot infer joint parameters since it estimates per-part poses individually. Besides, the part symmetry problem will also hinder such strategy from getting good performance to some extent, which will be discussed in the next section C.

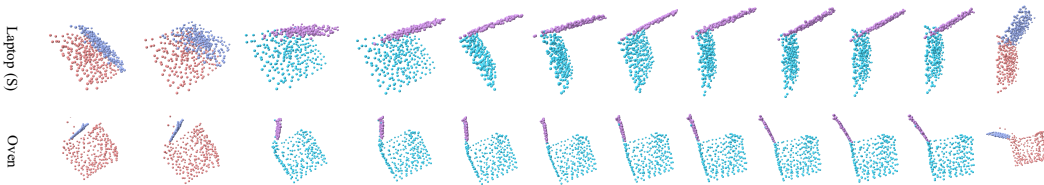


Figure 7: Reconstruction for shapes in different articulation states and manipulations to change their states. Shapes (in blue and orange) drawn on the two sides are manipulated shapes from their nearest reconstructions. Others are reconstructions (in purple and green). Please zoom in for details.

Table 2: Comparison between the part pose estimation performance of different methods on all test categories (**partial point clouds**). “R” denotes rotation errors with the value format “Mean  $R_{err}$ /Median  $R_{err}$ ”. “T” denotes translation errors with the value format “Mean  $T_{err}$ /Median  $T_{err}$ ”. “J” denotes joint parameters estimation results with the value format “Mean  $\theta_{err}$ /Mean  $d_{err}$ ”. **ICP could not predict joint parameters**. Therefore, only the results of supervised NPCS and our method on joint prediction are presented. For all metrics, the smaller, the better. **Bold** numbers for best values, while *blue* values represent second best ones.

	Method	Oven	Washing Machine	Eyeglasses	Laptop (S)	Safe	Laptop (R)	Drawer	Avg.
R	NPCS-EPN (supervised)	<b>2.51/2.27</b> , <b>2.93/2.64</b>	<i>4.71/3.84</i> , <b>8.56/7.46</b>	<i>7.26/6.08</i> , <i>23.39/17.33</i> , <i>20.86/18.76</i>	<i>21.40/23.56</i> , <i>29.90/32.71</i>	<b>6.64/5.76</b> , <b>5.43/5.19</b>	<i>9.39/8.75</i> , <i>6.75/6.14</i>	<b>21.74/10.80</b> , <b>22.92/10.18</b> , <b>25.10/14.16</b> , <b>7.34/6.83</b>	<b>13.34/10.73</b>
	Oracle ICP	21.53/10.80, 20.68/20.50	32.42/17.82, 19.39/16.99	73.24/78.73, 68.74/74.09, 69.23/74.53	67.48/73.01, 63.22/68.23	38.72/34.44, 52.28/42.16	30.78/28.674, 42.06/39.25	82.93/82.64, 61.31/59.51, 54.39/52.82, 26.88/29.66	48.55/47.29
	Ours	<i>11.77/7.87</i> , <i>10.83/9.15</i>	<b>1.61/1.52</b> , <i>12.81/12.51</i>	<b>4.69/3.77</b> , <i>9.56/5.36</i> , <b>7.53/6.12</b>	<b>10.18/5.30</b> , <b>11.10/5.22</b>	<i>15.38/14.46</i> , <i>21.91/19.04</i>	<b>8.50/6.85</b> , <b>6.92/5.66</b>	<i>2.60/1.79</i> , <i>2.60/1.79</i> , <i>2.06/1.79</i> , <i>2.06/1.79</i>	<i>8.36/6.47</i>
T	NPCS-EPN (supervised)	<b>0.028/0.030</b> , <b>0.028/0.023</b>	<b>0.034/0.030</b> , <b>0.033/0.028</b>	<b>0.085/0.075</b> , <b>0.056/0.052</b> , <b>0.057/0.049</b>	<i>0.263/0.253</i> , 0.286/0.236	<b>0.022/0.021</b> , <b>0.034/0.034</b>	<b>0.048/0.043</b> , <b>0.047/0.044</b>	<b>0.441/0.365</b> , <b>0.367/0.343</b> , <b>0.549/0.299</b> , <b>0.081/0.065</b>	<b>0.145/0.117</b>
	Oracle ICP	0.324/0.321, <i>0.169/0.171</i>	0.322/0.311, <i>0.136/0.144</i>	<i>0.092/0.097</i> , 0.188/0.197, 0.185/0.193	0.265/0.278, <i>0.267/0.277</i>	0.281/0.280, 0.246/0.248	0.280/0.289, 0.305/0.306	<i>0.193/0.197</i> , <i>0.161/0.170</i> , <i>0.159/0.164</i> , <i>0.129/0.132</i>	0.218/0.222
	Ours	<i>0.071/0.065</i> , 0.204/0.120	<i>0.179/0.164</i> , 0.253/0.254	0.219/0.226, <i>0.169/0.166</i> , <i>0.177/0.171</i>	<b>0.044/0.034</b> , <b>0.031/0.025</b>	<i>0.030/0.030</i> , <i>0.100/0.104</i>	<i>0.088/0.082</i> , <i>0.070/0.067</i>	0.046/0.046, 0.047/0.050, 0.122/0.131, 0.172/0.142	<i>0.119/0.110</i>
J	NPCS-EPN (supervised)	28.62/0.092	<b>8.05/0.194</b>	<b>20.11/0.221</b> , <b>20.11/0.239</b>	10.91/0.155	<b>11.23/0.084</b>	<b>12.25/0.134</b>	11.21/-	<b>15.31/0.160</b>
	Ours	<b>5.24/0.105</b>	22.30/0.212	26.96/0.087, 26.96/0.260	<b>10.83/0.142</b>	55.16/0.170	18.02/0.170	<b>7.43/-</b>	21.61/0.164

Table 3: Comparison between the part segmentation performance of different methods (**partial point clouds**). The metric used for this task is Segmentation MIoU, calculated on 4096 points for each shape. Values presented in the table are scaled by 100. Larger values indicate better performance.

	Oven	Washing Machine	Eyeglasses	Laptop (S)	Safe	Laptop (R)	Drawer
Oracle ICP	75.83	<b>73.07</b>	<b>68.92</b>	54.01	<b>66.90</b>	59.96	<b>58.38</b>
Ours	<b>87.07</b>	51.73	56.80	<b>84.94</b>	44.64	<b>86.04</b>	45.45

**Shape Reconstruction and Manipulation.** The predicted joints can enable us to manipulate the reconstruction by changing the value of predicted rotation angles. We then arrive at shapes in new articulation states different from input shapes. In Figure 7, we draw some examples for Laptop (S) and Oven.

## B.6 ROBUSTNESS TO INPUT DATA NOISE

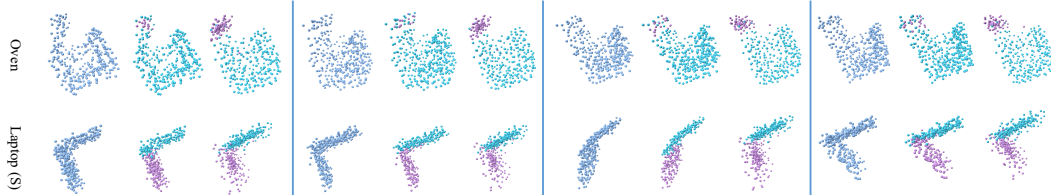


Figure 8: Visualization for the model performance on input data with random noise. Shapes for each three drawn from left to the right are input data corrupted by random normal noise, segmentation, and reconstruction, respectively. We align shapes here just for a better visualization, while they may be put into arbitrary poses for input.

Besides testing the performance of the proposed method on partial point clouds with occlusion patterns caused by viewpoint changes, we also test its effectiveness on noisy data. Specifically,



Table 4: Comparison between the shape reconstruction performance of different methods (**partial point clouds**). The metric used in this task is unidirectional Chamfer L1 from the original input shape to the reconstructed shape. The smaller, the better.

Method	Oven	Washing Machine	Eyeglasses	Laptop (S)	Safe	Laptop (R)	Drawer
EPN Li et al. (2021)	0.040	<b>0.043</b>	0.044	0.032	0.020	0.026	0.079
Ours	<b>0.035</b>	0.062	<b>0.041</b>	<b>0.025</b>	<b>0.019</b>	<b>0.024</b>	<b>0.061</b>

Table 5: Comparison between the part pose estimation performance of different methods. Backbone used for NPCCS is KPConv. “R” denotes rotation errors with the value format “Mean  $R_{err}$ /Median  $R_{err}$ ”. “T” denotes translation errors with the value format “Mean  $T_{err}$ /Median  $T_{err}$ ”. “J” denotes joint parameters estimation results with the value format “Mean  $\theta_{err}$ /Mean  $d_{err}$ ”. For all metrics, the smaller, the better. **Bold** numbers for best values.

	Method	Oven	Washing Machine	Eyeglasses	Laptop (S)	Safe	Laptop (R)	Drawer	Avg.
R	NPCCS-KPConv (supervised)	44.16/43.09, 60.58/63.35	56.20/56.22, 50.16/51.38	51.99/53.97, 42.48/38.08, 42.29/38.11	55.67/66.44, 55.63/61.33	11.68/11.10, 43.48/42.22	49.98/68.43, 73.40/83.55	62.73/69.42, 56.16/60.34, 57.23/63.90, 48.76/46.82	50.74/53.99
	Oracle EPN	<b>7.07/6.88</b> , 16.33/9.17	7.97/7.60, 33.56/20.49	54.01/13.09, 86.12/65.07, 116.56/119.23	18.33/9.73, 18.98/12.75	45.85/48.59, 38.03/27.67	20.46/14.03, 21.08/19.30	47.88/47.03, 30.84/25.23, 35.79/37.17, 43.83/39.46	37.81/30.73
	Ours	7.74/7.35, <b>4.07/3.97</b>	<b>7.49/7.37</b> , <b>19.27/19.19</b>	<b>8.16/8.21</b> , <b>12.29/10.89</b> , <b>12.53/9.88</b>	<b>7.34/5.16</b> , <b>10.41/9.34</b>	<b>9.03/9.09</b> , <b>13.83/13.59</b>	<b>5.71/3.61</b> , <b>3.64/2.84</b>	<b>3.18/2.73</b> , <b>3.18/2.73</b> , <b>3.18/2.71</b> , <b>3.18/2.71</b>	<b>7.90/7.14</b>
T	NPCCS-KPConv (supervised)	0.133/0.121, 0.104/0.091	0.146/0.142, 0.066/0.065	0.401/0.326, 0.418/0.257, 0.396/0.263	0.233/0.203, 0.217/0.169	<b>0.055/0.052</b> , 0.098/0.091	0.179/0.226, 0.161/0.174	0.791/0.742, 0.694/0.640, 1.005/0.942, 0.271/0.240	0.316/0.279
	Oracle EPN	<b>0.031/0.030</b> , <b>0.058/0.052</b>	<b>0.046/0.044</b> , 0.059/0.053	0.197/0.129, 0.128/0.118, 0.334/0.292	0.132/0.128, 0.117/0.090	0.157/0.157, 0.158/0.151	<b>0.092/0.086</b> , <b>0.094/0.082</b>	0.204/0.187, 0.177/0.166, 0.161/0.146, 0.290/0.282	0.143/0.129
	Ours	0.054/0.052, 0.067/0.046	<b>0.082/0.083</b> , <b>0.042/0.034</b>	<b>0.054/0.039</b> , <b>0.086/0.088</b> , <b>0.070/0.055</b>	<b>0.040/0.037</b> , <b>0.046/0.042</b>	0.066/0.069, <b>0.037/0.035</b>	<b>0.021/0.019</b> , <b>0.027/0.026</b>	<b>0.096/0.096</b> , <b>0.097/0.092</b> , <b>0.108/0.105</b> , <b>0.109/0.100</b>	<b>0.065/0.060</b>
J	NPCCS-KPConv (supervised)	55.62/0.194	55.01/0.149	60.58/0.329, 60.59/0.379	41.40/0.259	54.07/0.055	57.04/0.070	52.48/-	54.60/0.205
	Ours	<b>20.30/0.089</b>	<b>28.40/0.118</b>	<b>17.75/0.045</b> , <b>17.75/0.129</b>	<b>30.31/0.122</b>	<b>4.36/0.031</b>	<b>17.17/0.169</b>	<b>38.86/-</b>	<b>21.86/0.100</b>

we add noise for each point in the shape by sampling offsets for its x/y/z coordinates from normal distributions, e.g.  $\Delta x \sim \mathcal{N}(0, \sigma^2)$ , where we set  $\sigma = 0.02$  here. Results on Oven and Laptop (S) are presented in Table 6. From the table, we can see the degenerated segmentation IoU on Oven’s noisy data, while still relatively good part pose estimation performance. Another discovery is the even better joint axis orientation prediction, but larger offset prediction perhaps due to the poor segmentation. Besides, the shape reconstruction quality also drops a lot, probably due to the randomly shifted point coordinates. We can observe a similar phenomenon on Laptop (S). In Figure 8, we draw some examples for a qualitative understanding w.r.t. model’s performance on noise data.

## B.7 VISUALIZATION OF PART-LEVEL EQUIVARIANT FEATURES

Aiming for an intuitive understanding w.r.t. the property output by the designed part-level equivariant network, we draw features output by the global equivariant network and part-level equivariant network for some laptop samples in Figure 9. From the figure, we can see that the point features of the non-motion part (base) do not change a lot when the moving part (display) rotates an angle. That echoes the wish for the part-level equivariance design to disentangle other parts’ rigid transformation from the current part’s feature learning.

## B.8 EVALUATION STRATEGY FOR CATEGORY-LEVEL ARTICULATED OBJECT POSES

To evaluate the category-level part pose estimation performance of our model, we adopt the evaluation strategy used in Li et al. (2021).

Table 6: Performance comparison of the proposed method on clean data and data corrupted by random normal noise.

Category	Method	Seg. IoU	Mean $R_{err}(\circ)$	Median $R_{err}(\circ)$	Mean $T_{err}$	Median $T_{err}$	Joint Error	Chamfer L1
Oven	Without noise	<b>76.22</b>	<b>7.74, 4.07</b>	<b>7.35, 3.97</b>	<b>0.054, 0.067</b>	<b>0.052, 0.046</b>	20.30/ <b>0.089</b>	<b>0.025</b>
	With noise	55.35	9.84, 11.05	9.94, 9.99	0.073, <b>0.063</b>	0.073, 0.057	<b>9.28</b> /0.310	0.049
Laptop (S)	Without noise	<b>82.97</b>	<b>7.34, 10.41</b>	<b>5.16, 9.34</b>	<b>0.040, 0.046</b>	<b>0.037, 0.042</b>	<b>30.31</b> /0.122	<b>0.024</b>
	With noise	70.04	16.01, 13.27	11.47, 9.52	0.082, 0.067	0.075, 0.065	32.84/ <b>0.029</b>	0.044

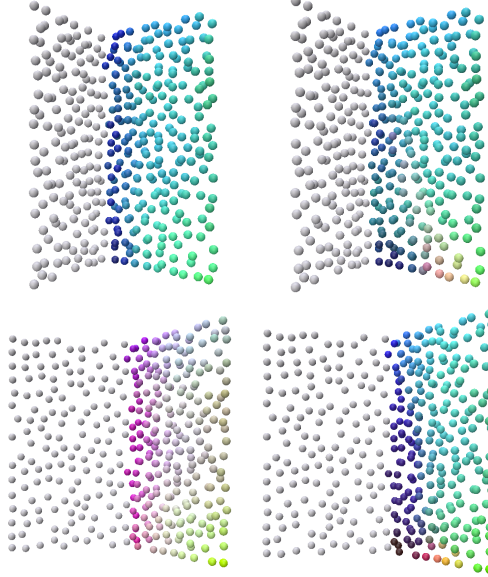


Figure 9: Visualization for an intuitive understanding w.r.t. the difference between the part-level equivariant feature and global equivariant feature of a specific part. Visualized features are obtained by using the PCA algorithm to reduce the feature dimension to 3, which are further normalized to the range of  $[0, 1]$ . We only draw point features of the non-motion part with the moving part in gray. Features drawn on the left global equivariant features while those on the right are from the part-level equivariant network.

For part-based metrics, we first feed a set of train shapes in the canonical articulation states and canonical object pose state to get a set of per-part pose predictions  $\{P_i\}$ . Then we can calculate the residual pose  $\hat{P}_i$  for each part  $i$  from the canonical part space defined by human to the canonical part space defined by the network from the pose prediction set (via RANSAC). After that, predicted pose from the canonical part space defined by human can be computed by applying the inverse residual pose estimation on the estimated per-part pose, e.g.  $P_i \leftarrow \hat{P}_i^{-1} P_i$ . When calculating the rotation and translation from part shape  $X_1$  to  $X_2$ , we first centralize their bounding boxes ( $\overline{X_1}$  and  $\overline{X_2}$ , respectively). Then, the transformation from  $\overline{X_1}$  to  $\overline{X_2}$  is taken as the transformation from  $X_1$  to  $X_2$ .

For joint parameters, we take the angle error between the predicted joint axis orientation and the ground-truth axis orientation as the metric for joint axis orientation prediction. Metric for joint position prediction is set to the minimum line-to-line distance, following Li et al. (2020). Only joint axis orientation prediction error is computed for prismatic joints.

## C DISCUSSION ON PART SYMMETRY

In this section, we discuss the part-symmetry-related problem that one would encounter in the part pose estimation problem. For rigid objects, the pose of a shape is ambiguous for symmetric shapes. To say a shape  $X$  is symmetric, we mean that there is a non-trivial  $SE(3)$  transformation  $S_{A_0}$  such that  $X = S_{A_0}[X]$ . In those cases, the performance of the pose estimation algorithm may degenerate due to ambiguous poses. It is a reasonable phenomenon, however. But for articulated objects, we may have symmetric parts even if the whole shape is not a symmetric one. For those shapes, we still expect for accurate part pose estimation. It indicates that estimating part poses for each part individually is not reasonable due to part pose ambiguity. That’s why we choose to model the relationship between parts,

or specifically, the kinematic chain, joint parameters. Without such object-level inter-part modeling, we cannot get accurate part poses by estimating their pose individually, even using ground-truth segmentation. The comparison between Oracle EPN and our method in Table 5 can demonstrate this point to some extent.

## REFERENCES

- Haiwei Chen, Shichen Liu, Weikai Chen, Hao Li, and Randall Hill. Equivariant point network for 3d point cloud analysis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14514–14523, 2021.
- Zhiqin Chen, Andrea Tagliasacchi, and Hao Zhang. Bsp-net: Generating compact meshes via binary space partitioning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 45–54, 2020.
- Yuki Kawana, Yusuke Mukuta, and Tatsuya Harada. Neural star domain as primitive representation. *Advances in Neural Information Processing Systems*, 33:7875–7886, 2020.
- Yuki Kawana, Yusuke Mukuta, and Tatsuya Harada. Unsupervised pose-aware part decomposition for 3d articulated objects. *CoRR*, abs/2110.04411, 2021. URL <https://arxiv.org/abs/2110.04411>.
- 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/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3706–3715, 2020.
- Xiaolong Li, Yijia Weng, Li Yi, Leonidas Guibas, A. Lynn Abbott, Shuran Song, and He Wang. Leveraging se(3) equivariance for self-supervised category-level object pose estimation, 2021.
- Yunze Liu, Yun Liu, Che Jiang, Zhoujie Fu, Kangbo Lyu, Weikang Wan, Hao Shen, Boqiang Liang, He Wang, and Li Yi. Hoi4d: A 4d egocentric dataset for category-level human-object interaction. *arXiv preprint arXiv:2203.01577*, 2022.
- Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, 30, 2017.
- Hugues Thomas, Charles R Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas J Guibas. Kpconv: Flexible and deformable convolution for point clouds. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 6411–6420, 2019.
- Xiaogang Wang, Bin Zhou, Yahao Shi, Xiaowu Chen, Qinpeng Zhao, and Kai Xu. Shape2motion: Joint analysis of motion parts and attributes from 3d shapes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8876–8884, 2019.
- Fanbo Xiang, Yuzhe Qin, Kaichun Mo, Yikuan Xia, Hao Zhu, Fangchen Liu, Minghua Liu, Hanxiao Jiang, Yifu Yuan, He Wang, et al. Sapien: A simulated part-based interactive environment. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11097–11107, 2020.