CAPT: Category-level Articulation Estimation from a Single Point Cloud Using Transformer

Lian Fu[1], Ryoichi Ishikawa[1], Yoshihiro Sato[2] and Takeshi Oishi[1]

***Abstract*— The ability to estimate joint parameters is essential for various applications in robotics and computer vision. In this paper, we propose CAPT: category-level articulation estimation from a point cloud using Transformer. CAPT uses an end-to-end transformer-based architecture for joint parameter and state estimation of articulated objects from a single point cloud. The proposed CAPT methods accurately estimate joint parameters and states for various articulated objects with high precision and robustness. The paper also introduces a motion loss approach, which improves articulation estimation performance by emphasizing the dynamic features of articulated objects. Additionally, the paper presents a double voting strategy to provide the framework with coarse-to-fine parameter estimation. Experimental results on several category datasets demonstrate that our methods outperform existing alternatives for articulation estimation. Our research provides a promising solution for applying Transformer-based architectures in articulated object analysis.**

I. Introduction

Accurate articulation estimation is essential for a wide range of applications in robotics. Articulation estimation is the task of obtaining the joint parameters and joint states of the objects from visual input, as depicted in Fig. 1. With the development of Deep Learning, related methods have now shifted from instance level toward estimating category-level articulation parameters from videos [1] or pairs of images [2].

Even so, category-level articulation estimation based on a single static point cloud remains a challenging task. The category prior provides only an ambiguous indication of an object's kinematic constraints, instead of clear information about the spatial structure as in a CAD model. Moreover, as dynamic properties, articulation parameters are not explicitly stored in static point clouds. Previous research has commonly approached this task by dividing it into multiple stages [3] or incorporating post-optimization algorithms [4]. However, such methods require relatively complex training procedures. The performance of a latter-stage model relies on a corresponding former-stage model, where a poor result in the former stage usually causes an even worse final estimation [2].

To address these challenges, we propose CAPT: category-level articulation estimation based on a single point cloud using Transformer. We introduce an end-to-end Transformer-based architecture for articulation estimation based on a single point cloud. To emphasize the dynamic nature of articulated objects, we also propose a motion loss approach to restore the dynamic features. Additionally, we design a high-accuracy double voting strategy to determine the final predicted parameter values. To the best of our knowledge, this is the first work that utilizes a Transformer-based architecture for the estimation of joint parameters of articulated objects.

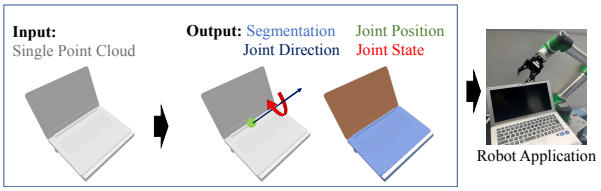

Fig. 1. Articulation estimation aims to estimate joint parameters and states from visual information. In our case, we propose to infer from only a single static point cloud. This task could be applied in virtual/augmented reality, robot interaction, etc.

Experimental results from several category datasets demonstrate that our methods achieve better performance than previously published alternatives for articulation estimation. Our methods accurately estimate the joint parameters of various articulated objects with high precision and robustness. This work opens new opportunities for the application of Transformer-based architectures in the field of articulated object analysis and control.

The main contributions of this paper are summarized as follows:

- Proposal of an end-to-end category-level articulation estimation model using Transformer
- Proposal of a motion loss approach that improves articulation estimation performance by emphasizing the dynamic character of articulated objects
- Design of a high-accuracy double voting strategy to decide the final predicted parameters
- Experiments on a synthetic dataset to demonstrate the high accuracy of our methods

II. Related Work

A. Articulation Estimation

Existing articulation estimation methods can be grouped into three categories based on the input type: interaction-based, multi-view-based, and single-view-based. Interaction methods provide sufficient dynamic information for articulation estimation, as made clear in numerous studies [5][6][7][8]. Related methods focus on using the differences

[1]The authors are with The Institute of Industrial Science, The University of Tokyo, Japan. {lianfu, ishikawa, oishi}@cvl.iis.u-tokyo.ac.jp

[2]The author is with Faculty of Engineering, Kyoto University of Advanced Science, Japan. sato.yoshihiro@kuas.ac.jp

in the observations before and after the interaction to help estimate an initial guess of the articulation model [9] or build up an implicit neural-representation of the object [2]. Multi-view methods infer the articulations from multiple observations [10][11][12], with recent works including ScrewNet [1], CAPTRA [13], DUST-net [14] and CLA-NeRF [15]. Compared with the other two types of methods, single-view methods require as little information as a single point cloud or a depth image [16][17][18]. To achieve single-view estimation, existing methods usually approach this task with multi-stage networks. RPM-Net [16] first predicts a temporal sequence of pointwise displacements using recurrent neural network before estimating articulation. ANCSH [4] predicts the articulation parameters with the help of articulation-aware normalized coordinate space hierarchy and post-optimization. Related research also combines manipulation with articulation estimation, including FlowBot3D [19]. Recently, Liu et al. [18] proposed a semi-weakly supervised approach using a Graph Neural Network (GNN) which takes pre-segmented point clouds as input. While single-view methods seem to be the most useful and practical, their performance remains in need of improvement.

B. Transformer for Point Clouds

Ever since its proposal by Vaswani et al. [20], Transformer has become the most popular neural model not only in natural language processing (NLP) but also in computer vision (CV). Vision-Transformer (ViT) [21] brought the Transformer into vision tasks and pushed performance in many CV tasks to new heights. The permutation-equivariance of the self-attention mechanism fits the inherently unordered point cloud data structure well. Transformer allows for the capture of global contextual information, compared with the limited receptive field in convolutional methods. Such global capture can be useful in understanding the overall structure of the point cloud. Guo et al. [22] and Zhao et al. [23] proposed Point Cloud Transformer and Point Transformer, respectively; both demonstrate the potential of Transformer in the field of Deep Vision. Transformer has been adopted in various point cloud processing tasks, including point cloud completion [24], denoising [25], and registration [26][27]. We believe that it is appropriate to apply an attention-based approach to articulation estimation.

III. Category-level Articulation Estimation Framework

In this section, we first clarify the problem formulation. Next, we introduce the structure of our framework, including a Transformer-based encoder and a multi-branch decoder. The encoder is identical to that of PCT [22], but for better understanding, we briefly explain it in Sec. III-B. The overall architecture of CAPT is shown in Fig. 2.

A. Problem Formulation

Given an articulated object O from a known category with n_J joints and n_L links, a partial point cloud with n points $Q = \{\mathbf{p}_i \in \mathbb{R}^d | i = 1, \ldots, n\}$ may be obtained from a single observation. The dimension $d = 3$ if the space coordinates x, y, z are used for each point, which is the case in our method. However, d could certainly be higher if more information is added to each point, such as color space r, g, b (where $d = 6$) [28] or normal vector space v^x, v^y, v^z (where $d = 6$) [29].

The expected outputs of the method are the segmentation of Q into n_L links, parameters, and states for all n_J joints. Point-wise segmentation predicts a segmentation label for each point $C = \{c(\mathbf{p}_i) \in 1, \ldots, n_\mathrm{L} | i = 1, \ldots, n\}$. Here, joints with one degree of freedom (DoF) are considered, since they are the most common joints. For joint $J_k (k = 1, ..., n_\mathrm{J})$, one property is joint direction, denoted by unit vector $\phi_k^{\mathrm{dir}} \in \mathbb{R}^3$. Although joint axes are usually considered to be directionless, we manually define a direction for each joint under a consistent rule. We also consider joint position, denoted as a point on the axis named pivot $\phi_k^{\mathrm{pivot}} \in \mathbb{R}^3$. Joint state is represented in radians, $\phi_k^{\mathrm{s}} \in (-\pi, \pi]$, and assigned a predefined zero state for each category.

B. Input embedding and Encoder

As a disordered data structure, a point cloud needs no position embedding as does Transformer in NLP [20] or patch embedding as does ViT [21]. However, as discussed in [30], adopting local feature embedding and neighbor feature embedding improves feature extraction performance. The encoder aims to extract high-dimensional features from 3D point clouds, so as to provide sufficient information for attached decoders to estimate targets. Our model adopts the same encoder as [22]. The encoder consists of four self-attention layers, which are used to process the point cloud data in a hierarchical manner. A feature map of $F^{\mathrm{e}} \in \mathbb{R}^{n \times 4d_{\mathrm{e}}}$ is finally outputted by the encoder and fed into the decoder.

C. Multi-branch Decoders

Multi-branch decoders, attached downstream from the encoder, execute several tasks. The decoders consist of articulation branch $\Omega_\theta^{\mathrm{arti}}$ and segmentation branch $\Omega_\theta^{\mathrm{seg}}$. The output channel number is determined by the category property, including the part number, joint number, and joint type for each sample in the category dataset. Within a given category, the link number and joint number are not always fixed.

Here we adopt a fully prepared strategy, which considers the number of parts n_L and the number of joints n_J for the output channels to represent the maximum number for the whole category dataset. For those samples whose numbers of parts or joints are below the maximum, we set the output of the extra channels to zero.

The segmentation branch predicts a point-wise segmentation mask for each link, i.e.,

$$\Omega_\theta^{\mathrm{seg}}(F_{\mathrm{e}}) \rightarrow \hat{H}(Q) \in \mathbb{R}^{n \times n_\mathrm{L}}, \tag{1}$$

where $\hat{H}(Q)$ is the possibility distribution for each point in Q belonging to each part. The joint axis direction may be defined by a unit direction vector $\phi_k^{\mathrm{dir}} \in \mathbb{R}^3$. The joint position may be represented by any point in the point cloud $\mathbf{p}_i \in Q$; a unit vector $\phi_k^{\mathrm{pdir}}(\mathbf{p}_i) \in \mathbb{R}^3$ starting from $\mathbf{p}_i$ and perpendicular

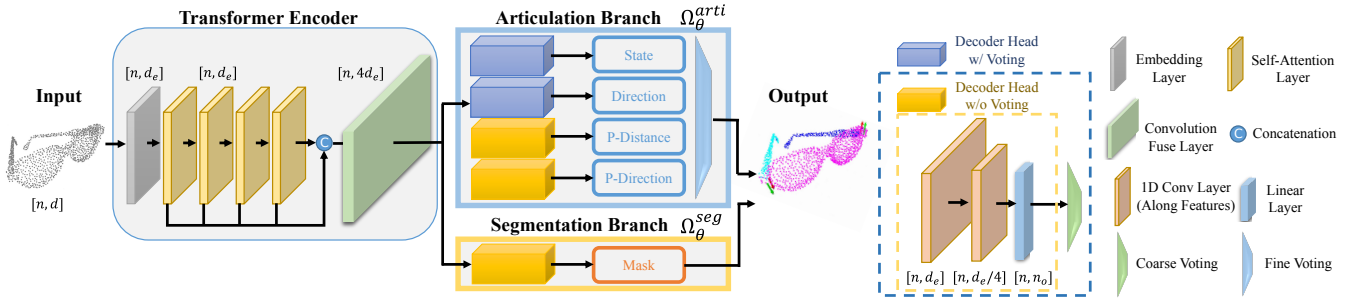

Fig. 2. Our CAPT (category-level articulation estimation from a point cloud using Transformer) architecture. n is the number of points, d is the origin feature dimension of each point and d_e is the embedded feature dimension. In the output, as in all figures in this paper, the red arrow represents the predicted joint, while the green arrow represents the ground truth joint.

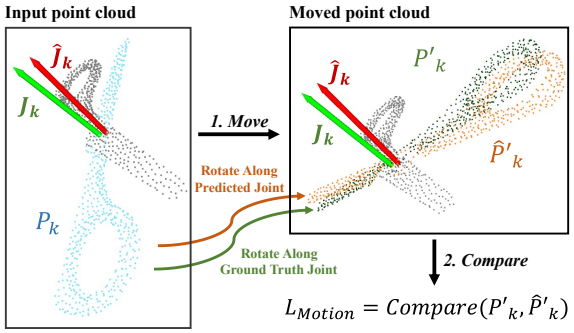

Fig. 3. Diagram of motion loss calculation. Motion loss of k^{th} joint is calculated in two steps. (1) Move: Moving the part point cloud P_k along predicted joint $\hat{J}_k$ and ground truth joint J_k to obtain rotated point clouds $\hat{P}'_k$ and P'_k, respectively. (2) Compare: Get motion loss by comparing $\hat{P}'_k$ and P'_k. The total motion loss is the sum of each joint's motion loss.

to the joint J_k^{r}; and the distance $\phi_k^{\mathrm{pdir}}(\mathbf{p}_i) \in \mathbb{R}^3$ between $\mathbf{p}_i$ and J_k^{r}.

To take better advantage of the global receptive field of Transformer, the articulation branch conducts a point-wise prediction of all articulation parameters:

$$\Omega_\theta^{\mathrm{arti}}(F_{\mathrm{e}}) \rightarrow \hat{\Phi}_k(\mathbf{p}), \tag{2}$$

$$\hat{\Phi}_k(\mathbf{p}) = \{\hat{\phi}_k^{\mathrm{dir}}(\mathbf{p}), \hat{\phi}_k^{\mathrm{dist}}(\mathbf{p}), \hat{\phi}_k^{\mathrm{pdir}}(\mathbf{p}), \hat{\phi}_k^{\mathrm{s}}(\mathbf{p})\}. \tag{3}$$

$\hat{\Phi}_k(\mathbf{p})$ consists of joint direction $\hat{\phi}_k^{\mathrm{dir}}(\mathbf{p}) \in \mathbb{R}^3$, point-to-joint distance $\hat{\phi}_k^{\mathrm{dist}}(\mathbf{p}) \in \mathbb{R}$, point-to-joint direction $\hat{\phi}_k^{\mathrm{pdir}}(\mathbf{p}) \in \mathbb{R}^3$, and joint state $\hat{\phi}_k^{\mathrm{s}}(\mathbf{p}) \in \mathbb{R}^3$.

The predicted joint pivot of each point is computed as

$$\hat{\phi}_k^{\mathrm{pivot}}(\mathbf{p}) = \mathbf{p} + \hat{\phi}_k^{\mathrm{dist}}(\mathbf{p}) \cdot \hat{\phi}_k^{\mathrm{pdir}}(\mathbf{p}). \tag{4}$$

IV. Loss Design and Optimization

We next explain the proposed motion loss approach, which recovers dynamic features of joints from static input. We also describe the total loss design and the double voting strategy, which inherits the coarse-to-fine inference paradigm in estimating articulation model parameters.

A. Motion Loss

Articulation is a dynamic property of objects that is not explicitly expressed by a single static point cloud. However, restoring explicit dynamic information is made possible by using additional constraints. The intuitive idea behind motion loss is to move the point cloud of parts separately according to the predicted and ground-truth joints and then compare the moved point clouds. A diagram of this calculation is shown in Fig. 3.

Formally, given point cloud $Q = \{\mathbf{p}_i \in \mathbb{R}^3 | i = 1, \ldots, n\}$, the model provides a prediction of the joint direction $\hat{\phi}_k^{\mathrm{dir}}$ and pivot $\hat{\phi}_k^{\mathrm{pivot}}$ of a joint J_k. We also know the ground truth direction ϕ_k^{dir}, pivot position ϕ_k^{pivot} of J_k, and segmentation label C for loss computation. We first obtain the part point cloud for the k_{th} link $P_k = \{\mathbf{p}_i \text{ if } C(\mathbf{p}_i) = k\}$, which is the child link of J_k. Denote the point number of P_k as n_k. A rotated point cloud P'_k may be obtained by using the Rodrigues rotation formula [31], which rotates points P_k along an arbitrary 3D axis decided by direction ϕ_k^{dir} and any point on the axis ϕ_k^{pivot}, by an amount α, i.e.,

$$P'_k = \Psi_{\mathrm{Rodrigues}}(P_k, \phi_k^{\mathrm{dir}}, \phi_k^{\mathrm{pivot}}, \alpha). \tag{5}$$

We set rotation angle α as $\pi/2$. $\hat{P}'_k$ may be obtained similarly by rotating the part point cloud P_k along predicted joint axis. The motion loss is then calculated as

$$L_{\mathrm{motion}}(P'_k, \hat{P}'_k) = \sum_k^{n_{\mathrm{L}}} \sum_i^{n_k} \frac{\|\mathbf{p}'_{k,i} - \hat{\mathbf{p}}'_{k,i}\|}{n_{\mathrm{L}} \cdot n_k}, \tag{6}$$

where $\mathbf{p}'_{k,i} \in P'_k, \hat{\mathbf{p}}'_{k,i} \in \hat{P}'_k$. In our case, we choose the least squared error loss (L2 loss) as the per-point space distance function *Compare*.

B. Total Loss

The total loss is composed of the losses for segmentation, joint direction, joint position, and joint state. For segmentation, we compute the cross-entropy loss as

$$L_{\mathrm{seg}} = -\sum_i^n \sum_k^{n_{\mathrm{L}}} y_i \cdot \log(\hat{H}(\mathbf{p}_i)), \tag{7}$$

where $y_i = 1$ if $C(\mathbf{p}_i) = k$, otherwise $y_i = 0$.

For joint direction prediction, we compute the cosine similarity distance as

$$L_{\text{dir}} = \sum_{k}^{n_{\text{J}}} \sum_{i}^{n_k} \frac{1 - \Psi_{\text{cossim}}(\hat{\phi}_k^{\text{dir}}(\mathbf{p}_i), \phi_k^{\text{dir}}(\mathbf{p}_i))}{n_{\text{J}} \cdot n_k}, \tag{8}$$

where $\Psi_{\text{cossim}}(u, v)$ represents the cosine similarity between *d-dim* vectors u and v as

$$\Psi_{\text{cossim}}(u, v) = \frac{\sum_{i=1}^{d} u_i \cdot v_i}{\sqrt{\sum_{i=1}^{d} u_i^2} \cdot \sqrt{\sum_{i=1}^{d} v_i^2}}. \tag{9}$$

For joint position prediction, we again compute the cosine similarity distance as

$$L_{\text{pdir}} = \sum_{k}^{n_{\text{J}}} \sum_{i}^{n_k} \frac{1 - \Psi_{\text{cossim}}(\hat{\phi}_k^{\text{pdir}}(\mathbf{p}_i), \phi_k^{\text{pdir}}(\mathbf{p}_i))}{n_{\text{J}} \cdot n_k}, \tag{10}$$

and the L2 loss for point-to-joint distance as

$$L_{\text{dist}} = \sum_{k}^{n_{\text{J}}} \sum_{i}^{n_k} \frac{\|\hat{\phi}_k^{\text{dist}}(\mathbf{p}_i) - \phi_k^{\text{dist}}(\mathbf{p}_i)\|}{n_{\text{J}} \cdot n_k}, \tag{11}$$

Joint state prediction loss is computed as least absolute deviations loss (L1 loss), i.e.,

$$L_{\text{s}} = \sum_{k}^{n_{\text{J}}} \sum_{i}^{n_k} \frac{|\hat{\phi}_k^{\text{s}}(\mathbf{p}_i) - \phi_k^{\text{s}}(\mathbf{p}_i)|}{n_{\text{J}} \cdot n_k}. \tag{12}$$

The overall loss is computed as

$$L = [L_{\text{seg}}, L_{\text{dir}}, L_{\text{pdir}}, L_{\text{dist}}, L_{\text{s}}, L_{\text{motion}}] \cdot \mathbf{W}^{\text{T}}. \tag{13}$$

We roughly set the weight combination $\mathbf{W}$ used in this paper to be $\mathbf{W} = [1, 1, 1, 1, 1, 0.1]$, considering a balanced learning rate for each target.

C. Double Voting

The decoder gives per-point prediction $\hat{\Phi}_k(\mathbf{p})$ for the parameters and state of joint J_k. Points at different distances from J_k usually contain different amounts of information; such differences should not be ignored. Furthermore, points around or on the axis give poor predictions, since the normals of such points may be unstable due to the articulation structure. In order to rule these points out, we propose double voting, a coarse-to-fine voting strategy for accurately and robustly deciding the target estimated parameter values. As in the operation described in [2], a coarse voting is first conducted, i.e.,

$$\bar{\Phi}_k = \sum_{i}^{n} \frac{\hat{\Phi}_k(\mathbf{p}_i)}{n}, \tag{14}$$

where all points are weighted equally. Coarse voting assumes that all points contain the same amount of information about J_k, but this assumption holds only if the object lies fully along the joint axis. To focus on the points that probably contain more information and are more likely to give good predictions of joint parameter values, we conduct a subsequent round of fine voting. We first compute the distance to the coarsely voted joint J_k for each point as

$$\hat{\phi}_k^{\text{dist}}(\mathbf{p}) = \Psi_{\text{dist}}(\mathbf{p}, \hat{\phi}_k^{\text{pivot}}, \hat{\phi}_k^{\text{dir}}), \tag{15}$$

where

$$\Psi_{\text{dist}}(\mathbf{p}, \mathbf{q}, \mathbf{u}) = \|((\mathbf{p} - \mathbf{q}) \cdot \mathbf{u}^{\text{T}}) \cdot \mathbf{u} - (\mathbf{p} - \mathbf{q})\|. \tag{16}$$

We set a simple distance threshold, i.e.,

$$\mathbf{p}_i \rightarrow Q_k^{\text{f}} \text{ if } \hat{\phi}_k^{\text{dist}}(\mathbf{p}_i) \in [\omega_0, \omega_1] \cdot \beta(\hat{\Phi}_k^{\text{dist}}), \tag{17}$$

where Q_k^{f} is a subset of Q after removing points far from estimated joint $\hat{J}_k$. $\beta(\mathbf{x})$ indicates the median value of $\mathbf{x}$. ω_0 and ω_1 are hyperparameters for adjusting the range of points involved in fine voting. A larger ω_0 value means that more distant points also contribute to the final result, while a smaller value means that only very close points participate in the fine-voting round. ω_1 performs just the opposite role, controlling how close a point may be and still being counted in the fine voting round. Finally, the estimated parameters after fine voting are computed as

$$\bar{\Phi}_k^{\text{f}} = \sum_{q \in Q_k^{\text{f}}} \frac{\hat{\Phi}_k(\mathbf{q})}{n_k^{\text{f}}}, \tag{18}$$

where n_k^{f} is the number of points in Q_k^{f}.

V. Experiments

A. Datasets

Several articulated object datasets have been proposed recently [17][32][33][34], but Shape2Motion [3] is still the most used and most appropriate for articulation estimation tasks. We used four subsets for quantitative evaluation. We also conducted a qualitative evaluation of the other subsets to demonstrate that our methods could achieve good performance on other synthetic categories.

Here we briefly summarize the data used in our experiment. After considering the diversity, representativeness, and feasibility of the test set, we selected **laptop**, **washingmachine**, **oven**, and **eyeglasses** for quantitative evaluation experiments. **Scissors** and **bike** were also used for qualitative experiments. The joint numbers of these objects (1, 1, 1, 2, 2, and 4, respectively) cover various kinematic structures. The numbers of models in each category are 86, 62, 42, 43, 26, and 63, respectively.

Point clouds of articulated objects with annotation labels were generated through simulation in Pybullet [35]. Each joint was set to random initial states within the motion limits. Data were further augmented through random rotation along the x-axis, y-axis, and z-axis, with $R_x, R_y, R_z \in [-\pi, \pi]$; random translation along each axis with range $[-1, 1]$; and scaling by ratios with range $[0.8, 1.2]$. We split the dataset into training, validation, and test data sets with a volume ratio of 7:2:1, respectively. We ensured that all inputs were unseen objects with random states and points of view during testing.

B. Baselines

We compared our methods with a simple PCT approach that combines CAPT-plain and ANCSH [4]. This simple PCT approach shares the same encoder structure as our method but uses a single-branch multilayer perceptron as

the decoder and directly regresses the joint parameters. This approach was used as a control to determine whether a powerful Transformer encoder alone is sufficient for satisfactory completion of the articulation estimation task. ANCSH also estimates category-level articulation parameters from single-point-cloud input. However, ANCSH requires a post-optimization process, unlike our end-to-end method. To determine the effects of motion loss and double voting, we also used CAPT-plain for comparison, which is CAPT without motion loss and double voting.

C. Evaluation Metrics

We used the following metrics to evaluate our method. We used per-point accuracy (PA) and mean intersection over union (mIoU) to evaluate the segmentation result for each point. For joint position, we use average Euclidean distance (AED) between the predicted joint axis and the ground truth joint axis. For joint direction, we used mean error in degrees (MED) between the predicted joint axis direction and the ground truth joint axis direction. We again used mean error in degrees to evaluate joint state estimation accuracy. To assess the stability of the evaluated method, we used the proportions of results with joint position error values below 0.05 and below 0.01 (reported as AP5 and AP10, respectively, under the joint position columns), the proportions of results with joint direction error below 5 degrees and below 10 degrees (reported as AP5 and AP10, respectively, under the joint direction columns), and the proportions of results with joint state error values below 5 degrees and below 10 degrees (reported as AP5 and AP10, respectively, under the joint state columns).

D. Experimental Results

Evaluation results are summarized in Table I. Qualitative results are visualized in Fig. 6.

For segmentation, our methods achieved a high segmentation accuracy: >98% in all subsets. Moreover, our methods achieved top performance in three of the four estimation categories. Even in the eyeglasses category, our methods obtained results comparable with those of the best-performing alternative method.

We also determined the influence of motion loss and double voting: for joint parameter estimation, a direct regression method such as PCT might achieve a result slightly better than that of ANCSH, a PointNet- and RANSAC-based method. The comparison of the results between PCT and CAPT-plain shows that a voting-based multi-head decoder is necessary in order to obtain joint parameter estimation of high accuracy and stability. However, CAPT with motion loss and double voting further enhanced precision, especially for joint direction prediction. Double voting also boosted the exactness of the estimation to some extent.

The advantage of double voting was strengthened in some hard cases, as shown in Fig. 4. It was particularly difficult to conduct successful articulation estimation on objects with such tiny links. These tiny links resulted in (1) few points in the link, thus few structural features, and (2) degenerate surface normals, since each link was reduced to a line with no effective surface. PCT failed in this case, while our method achieved good estimation despite the difficulty. Double voting here filtered out some of the less trustable points, thus obtaining a better result than CAPT-plain. However, we note that the two double voting hyperparameters $\omega_0 and \omega_1$ sometimes needed to be carefully examined before inference in order to avoid loss of accuracy.

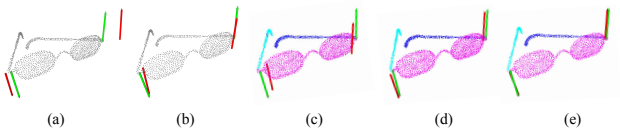

Fig. 4. Comparisons between (a) ANCSH (b) PCT (c) CAPT-plain (CAPT without double voting and motion loss) (d) CAPT without double voting and (e) CAPT with double voting. Here the object has thin arms, which can make naive PCT unmanageable. On the other hand, our method successfully predicted joint parameter values with relatively high accuracy whether or not double voting was used. Double voting yielded an even better result.

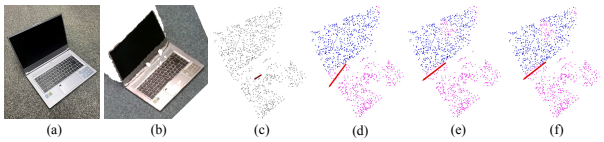

Fig. 5. Direct sim-to-reality result. (a) Real scene, (b) extracted point cloud, (c) naive PCT, (d) without motion loss, (e) without double voting, and (f) our methods. The results indicate that our category-level articulation estimation from a single point cloud using Transformer (CAPT) methods successfully captured the category features of noisy real-world articulated objects despite being trained with only a synthetic dataset.

E. Time Consumption

The average time Consumption for each input is shown in Table II. Since CAPT's end-to-end structure required no further post-optimization processing, it took a relatively shorter time compared with the multi-stage method. Notably, the inference time consumption of CAPT was relatively insensitive to the joint number of the subset. The increase in inference time between the single-joint laptop subset and the multiple-joint eyeglasses subset was only 13% for CAPT but was up to 2 times more for ANCSH.

F. Direct Simulation-to-reality Result

Direct simulation-to-real experiments indicated that the CAPT model exhibits promising performance in real-world articulation estimation, even in the absence of fine-tuning on real-world datasets. This finding is demonstrated by the direct simulation-to-reality results presented in Fig. 5. Prior to inputting the real-world point cloud data into the model, we conducted a series of pre-processing steps, including the use of RANSAC to fit and remove the ground plane; statistical outlier removal to denoise the point cloud; and point cloud normalization [36].

TABLE I

EVALUATION RESULTS FOR FOUR TEST DATA SETS: **PA** refers to per-point accuracy. **mIoU** refers to mean intersection over union. **AP***x* refers to average precision (the proportion of results with error below *x*). **MED** refers to mean error in degrees. **AED** refers to average Euclidean distance. Values for each joint are represented as multiple values in a single cell for the multiple joints category.

Dataset	Method	Segmentation		Joint Direction			Joint Position			Joint State		
		AP↑	mIoU↑	MED↓	AP5↑	AP10↑	AED↓	AP1↑	AP5↑	MED↓	AP5↑	AP10↑
Eyeglasses	PCT	-	-	7.74, 6.98	0.29, 0.38	0.73, 0.79	0.05, 0.04	0.12, 0.14	0.53, 0.60	11.3, 11.1	0.27, 0.26	0.52, 0.50
	ANCSH	0.89	0.73	6.35, 6.37	0.48, 0.49	0.84, 0.84	0.09, 0.11	0.08, 0.01	0.37, 0.27	13.9, 13.2	0.04, 0.05	0.11, 0.12
	CAPT-Plain	**0.97**	**0.90**	6.86, 5.83	0.35, 0.48	0.83, 0.90	0.04, 0.04	0.15, 0.17	0.62, 0.70	10.14, 10.49	0.26, 0.26	0.52, 0.49
	CAPT (ours)	0.95	0.87	**5.15, 4.26**	**0.54, 0.72**	**0.95, 0.96**	**0.03, 0.03**	**0.21, 0.20**	**0.81, 0.82**	**6.58, 6.84**	**0.51, 0.49**	**0.79, 0.79**
Laptop	PCT	-	-	5.40	0.54	0.94	0.02	0.30	0.92	10.90	0.26	0.51
	ANCSH	0.52	0.33	9.06	0.48	0.71	0.50	0.01	0.01	12.10	0.02	0.03
	CAPT-Plain	0.98	0.95	**2.13**	**0.95**	**0.99**	0.01	0.50	0.98	**9.38**	0.30	**0.55**
	CAPT (ours)	**0.98**	**0.95**	2.20	0.95	0.99	**0.01**	**0.53**	**0.98**	9.80	**0.59**	0.54
Oven	PCT	-	-	4.64	0.73	0.92	0.03	0.21	0.73	7.46	0.49	0.72
	ANCSH	0.69	0.46	7.11	0.51	0.74	0.31	0.02	0.02	26.50	0.08	0.17
	CAPT-Plain	0.98	0.94	4.27	0.64	0.92	0.03	**0.22**	0.75	6.99	0.49	0.77
	CAPT (ours)	**0.98**	**0.94**	**4.22**	**0.74**	**0.94**	**0.03**	0.20	**0.76**	**5.58**	**0.60**	**0.87**
Washing Machine	PCT	-	-	6.77	0.44	0.85	0.04	0.22	0.77	10.40	0.36	0.63
	ANCSH	0.83	0.56	19.10	0.21	0.35	0.30	0.04	0.05	22.60	0.11	0.23
	CAPT-Plain	0.99	0.97	**5.84**	0.52	**0.91**	0.03	0.24	**0.95**	**8.35**	**0.49**	**0.75**
	CAPT (ours)	**0.99**	**0.97**	5.93	**0.53**	0.90	**0.03**	**0.25**	0.81	8.85	0.44	0.73
Mean	PCT	-	-	6.31	0.48	0.85	0.04	0.20	0.71	10.23	0.22	0.37
	ANCSH	0.73	0.52	9.60	0.43	0.70	0.26	0.03	0.14	17.66	0.04	0.09
	CAPT-Plain	0.98	0.93	4.99	0.59	0.91	0.03	0.26	0.80	9.07	0.26	0.41
	CAPT (ours)	**0.98**	**0.93**	**4.35**	**0.70**	**0.95**	**0.03**	**0.28**	**0.84**	**7.53**	**0.33**	**0.43**

TABLE II

AVERAGE TIME CONSUMPTION FOR EACH INPUT: CAPT w/o DV means CAPT without using double voting. Laptop represents single-joint objects, while eyeglasses represent multiple-joint objects.

Dataset	ANCSH (s)	CAPT w/o DV (s)	CAPT (s)
Laptop	1.00	**0.035**	0.036
Eyeglasses	3.33	**0.040**	0.041

VI. CONCLUSION

In this paper, we proposed a set of articulation estimation methods: articulation point transformer, with a motion loss approach to recover dynamic features from a single static point cloud, and coarse-to-fine double voting to achieve high accuracy in category-level articulation estimation. We demonstrated better performance for these methods than for existing alternative methods or for baseline performance on a multi-category articulated object data set. The end-to-end structure of our study distinguishes it from previous work in which post-optimization or multi-stage networks were used. Future work will focus on cross-category articulation estimation with less or no prior kinematic constraint knowledge required. We are also working on the deployment of this articulation estimation system in real-scene robot manipulation, as a promising application.

ACKNOWLEDGMENT

This work was partly supported by the social cooperation program "Technology for IoT sensing and analysis," sponsored by UTokyo and Air Water.

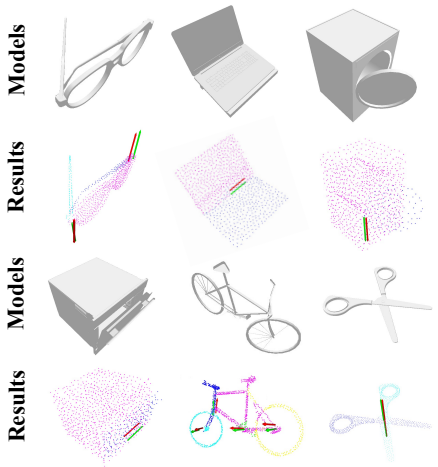

Fig. 6. Qualitative results for six categories: eyeglasses, laptop, washing machine, oven, bike, and scissors. The models are provided for reference only. The point clouds of washing machine and oven are merged from multiple observations to show the inner structures.

REFERENCES

[1] A. Jain, R. Lioutikov, C. Chuck, and S. Niekum, "ScrewNet: Category-Independent Articulation Model Estimation From Depth Images Using Screw Theory," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 13 670–13 677.

[2] Z. Jiang, C.-C. Hsu, and Y. Zhu, "Ditto: Building Digital Twins of Articulated Objects From Interaction," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 5616–5626.

[3] X. Wang, B. Zhou, Y. Shi, X. Chen, Q. Zhao, and K. 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*, 2019, pp. 8876–8884.

[4] X. Li, H. Wang, L. Yi, L. J. Guibas, A. L. Abbott, and S. Song, "Category-level articulated object pose estimation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 3706–3715.

[5] K. Hausman, S. Niekum, S. Osentoski, and G. S. Sukhatme, "Active articulation model estimation through interactive perception," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, May 2015, pp. 3305–3312.

[6] J. Sturm, C. Stachniss, and W. Burgard, "A Probabilistic Framework for Learning Kinematic Models of Articulated Objects," *Journal of Artificial Intelligence Research*, vol. 41, pp. 477–526, Aug. 2011.

[7] J. Bohg, K. Hausman, B. Sankaran, O. Brock, D. Kragic, S. Schaal, and G. Sukhatme, "Interactive Perception: Leveraging Action in Perception and Perception in Action," *IEEE Transactions on Robotics*, vol. 33, no. 6, pp. 1273–1291, Dec. 2017.

[8] D. Katz, M. Kazemi, J. A. Bagnell, and A. Stentz, "Interactive segmentation, tracking, and kinematic modeling of unknown 3d articulated objects," in *2013 IEEE International Conference on Robotics and Automation*. IEEE, 2013, pp. 5003–5010.

[9] R. S. Hartanto, R. Ishikawa, M. Roxas, and T. Oishi, "Hand-Motion-guided Articulation and Segmentation Estimation," in *2020 29th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*. IEEE, 2020, pp. 807–813.

[10] L. Yi, H. Huang, D. Liu, E. Kalogerakis, H. Su, and L. Guibas, "Deep Part Induction from Articulated Object Pairs," *ACM Transactions on Graphics*, vol. 37, no. 6, pp. 1–15, Dec. 2018.

[11] E. Colleoni, S. Moccia, X. Du, E. De Momi, and D. Stoyanov, "Deep Learning Based Robotic Tool Detection and Articulation Estimation With Spatio-Temporal Layers," *IEEE Robotics and Automation Letters*, vol. 4, no. 3, pp. 2714–2721, July 2019.

[12] N. Heppert, T. Migimatsu, B. Yi, C. Chen, and J. Bohg, "Category-Independent Articulated Object Tracking with Factor Graphs," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2022, pp. 3800–3807.

[13] Y. Weng, H. Wang, Q. Zhou, Y. Qin, Y. Duan, Q. Fan, B. Chen, H. Su, and L. J. Guibas, "CAPTRA: CAtegory-Level Pose Tracking for Rigid and Articulated Objects From Point Clouds," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 13 209–13 218.

[14] A. Jain, S. Giguere, R. Lioutikov, and S. Niekum, "Distributional Depth-Based Estimation of Object Articulation Models," in *Proceedings of the 5th Conference on Robot Learning*. PMLR, Jan. 2022, pp. 1611–1621.

[15] W.-C. Tseng, H.-J. Liao, L. Yen-Chen, and M. Sun, "CLA-NeRF: Category-Level Articulated Neural Radiance Field," in *2022 International Conference on Robotics and Automation (ICRA)*, May 2022, pp. 8454–8460.

[16] Z. Yan, R. Hu, X. Yan, L. Chen, O. van Kaick, H. Zhang, and H. Huang, "RPM-Net: Recurrent Prediction of Motion and Parts from Point Cloud," *ACM Transactions on Graphics*, vol. 38, no. 6, pp. 1–15, Dec. 2019.

[17] L. Liu, H. Xue, W. Xu, H. Fu, and C. Lu, "Towards Real-World Category-level Articulation Pose Estimation," *IEEE Transactions on Image Processing*, vol. 31, pp. 1072–1083, 2022.

[18] G. Liu, Q. Sun, H. Huang, C. Ma, Y. Guo, L. Yi, H. Huang, and R. Hu, "Semi-Weakly Supervised Object Kinematic Motion Prediction," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 21 726–21 735.

[19] B. Eisner, H. Zhang, and D. Held, "Flowbot3d: Learning 3d articulation flow to manipulate articulated objects," *arXiv preprint arXiv:2205.04382*, 2022.

[20] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is All you Need," in *Advances in Neural Information Processing Systems*, vol. 30. Curran Associates, Inc., 2017.

[21] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[22] M.-H. Guo, J.-X. Cai, Z.-N. Liu, T.-J. Mu, R. R. Martin, and S.-M. Hu, "Pct: Point cloud transformer," *Computational Visual Media*, vol. 7, no. 2, pp. 187–199, 2021.

[23] H. Zhao, L. Jiang, J. Jia, P. H. Torr, and V. Koltun, "Point transformer," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 16 259–16 268.

[24] J. Lin, M. Rickert, A. Perzylo, and A. Knoll, "PCTMA-Net: Point Cloud Transformer with Morphing Atlas-based Point Generation Network for Dense Point Cloud Completion," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sept. 2021, pp. 5657–5663.

[25] X. Xu, G. Geng, X. Cao, K. Li, and M. Zhou, "TDNet: Transformer-based network for point cloud denoising," *Applied Optics*, vol. 61, no. 6, pp. C80–C88, Feb. 2022.

[26] Y. Wang and J. M. Solomon, "Deep Closest Point: Learning Representations for Point Cloud Registration," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 3523–3532.

[27] X. Yu, L. Tang, Y. Rao, T. Huang, J. Zhou, and J. Lu, "Point-BERT: Pre-Training 3D Point Cloud Transformers With Masked Point Modeling," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 19 313–19 322.

[28] J. Park, Q.-Y. Zhou, and V. Koltun, "Colored Point Cloud Registration Revisited," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 143–152.

[29] W. Wu, Z. Qi, and L. Fuxin, "PointConv: Deep Convolutional Networks on 3D Point Clouds," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 9621–9630.

[30] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 652–660.

[31] H. Cheng and K. C. Gupta, "An Historical Note on Finite Rotations," *Journal of Applied Mechanics*, vol. 56, no. 1, pp. 139–145, Mar. 1989.

[32] F. Michel, A. Krull, E. Brachmann, M. Y. Yang, S. Gumhold, and C. Rother, "Pose estimation of kinematic chain instances via object coordinate regression." in *BMVC*, 2015, pp. 181–1.

[33] F. Xiang, Y. Qin, K. Mo, Y. Xia, H. Zhu, F. Liu, M. Liu, H. Jiang, Y. Yuan, H. Wang, L. Yi, A. X. Chang, L. J. Guibas, and H. Su, "SAPIEN: A SimulAted Part-Based Interactive ENvironment," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 11 097–11 107.

[34] R. Hu, W. Li, O. Van Kaick, A. Shamir, H. Zhang, and H. Huang, "Learning to predict part mobility from a single static snapshot," *ACM Transactions on Graphics*, vol. 36, no. 6, pp. 227:1–227:13, Nov. 2017.

[35] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation for games, robotics and machine learning," http://pybullet.org, 2016–2021.

[36] J. Guo, L. Fu, M. Jia, K. Wang, and S. Liu, "Fast and Robust Bin-picking System for Densely Piled Industrial Objects," in *2020 Chinese Automation Congress (CAC)*, Nov. 2020, pp. 2845–2850.