This CVPR paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the accepted version;
the final published version of the proceedings is available on IEEE Xplore.

Command-driven Articulated Object Understanding and Manipulation

Ruihang Chu[1] Zhengzhe Liu[1] Xiaoqing Ye[2] Xiao Tan[2]
Xiaojuan Qi[3*] Chi-Wing Fu[1] Jiaya Jia[1,4]
[1]CUHK [2]Baidu Inc. [3]HKU [4]SmartMore

Abstract

We present Cart, a new approach towards articulated-object manipulations by human commands. Beyond the existing work that focuses on inferring articulation structures, we further support manipulating articulated shapes to align them subject to simple command templates. The key of Cart is to utilize the prediction of object structures to connect visual observations with user commands for effective manipulations. It is achieved by encoding command messages for motion prediction and a test-time adaptation to adjust the amount of movement from only command supervision. For a rich variety of object categories, Cart can accurately manipulate object shapes and outperform the state-of-the-art approaches in understanding the inherent articulation structures. Also, it can well generalize to unseen object categories and real-world objects. We hope Cart could open new directions for instructing machines to operate articulated objects. Code is available at https://github.com/dvlab-research/Cart.

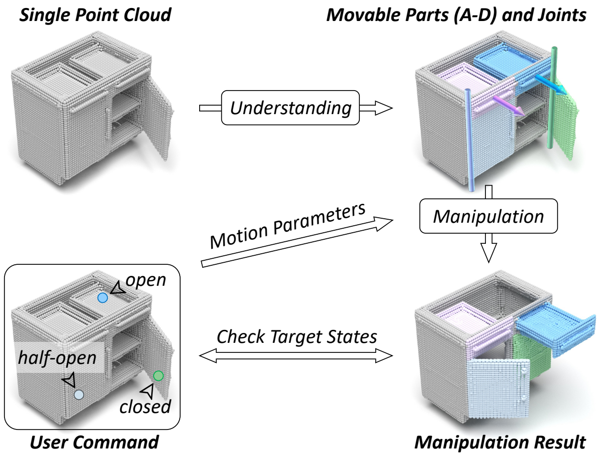


Figure 1. Given a single point cloud of an articulated object and a user command, Cart accordingly manipulates the object shape, based on understanding object structure and connecting the command to predict the motion parameters. The command gives the desired state and we use it to check if the manipulation succeeds.

1. Introduction

Articulated objects such as doors, cabinets, and laptops are ubiquitous in our daily life. Enabling machines to manipulate them under human instructions will pave the way for many applications, such as robotic work, human-computer interaction, and augmented reality. Recent research majorly focuses on understanding the structural properties of articulated objects, *e.g.*, discovering the movable parts [51, 61] and estimating the articulations between connected parts [19, 28]. Despite their success, the way from understanding objects to user-controllable manipulation remains challenging.

First, the model should be able to interpret the user instruction and determine the corresponding object part for performing the manipulation action. To do so requires understanding not only the object's articulated structure but also the operational properties (see Fig. 1). If a part is movable, the action type and current state should be recognized ahead. Second, to achieve successful manipulation, the model should know the target configuration or state of the object so as to determine the associated manipulation parameters, such as the rotation orientation and angle, for achieving the user instruction (see Fig. 1).

In this work, we study the above challenges by formulating a new task called *command-driven articulated object manipulation*. Given a visual observation of an articulated object and a simple user command that specifies how to manipulate the object, our objective is to manipulate the associated object part to reach the target articulation state given by the command. Specifically, this task combines the needs of understanding the object structure (which part to move), recognizing the associated articulation property (how it moves), and predicting the amount of motion subject to the user command (how much to move). To this end, we propose a novel learning-based framework called **C**ommand-driven **art**iculation modeling, or Cart for short; see Fig. 1.

In particular, Cart is powerful in both structure understanding and shape manipulation for articulated objects. The former acts as a critical building block, where we jointly

[*]Corresponding Author

segment individual object parts, predict their motion joints, and model the current articulation state, given only a single visual observation. Based on the structural prediction, we further connect the user commands with the inherent object geometry for effective object manipulation.

In our approach, we choose structured templates to create user-friendly commands. This design allows users to select a single point on object's point cloud to identify the part to operate and specify its desired target state. To associate the user-selected point to an intact object part, we leverage model segmentation prediction to locate the matched one. Subsequently, we transform the target states to a learnable embedding and fuse it with the visual features in a part-aware manner. Since the fused features encapsulate twofold information, it can be further applied to estimate the command-related motion parameters.

After the manipulation, to ensure the target object state satisfies the user command, we formulate the novel Test-time State Adaptation (TTSA) algorithm for state regularization. It automatically generates the state label from the command input and uses it to optimize the movement parameters, thereby adjusting the final object status. Essentially, it is self-supervised and works without requiring ground-truth supervision. With little additional computation cost, TTSA is able to robustly improve the quality of the object manipulation, especially for unseen object categories.

We extensively evaluate Cart on two widely-used datasets [51, 57] of articulated objects. Compared to the recent work [19, 28, 61], Cart yields superior predictive performance on both static object structure and dynamic motion parameters used for manipulation. We specifically show that each previous method can only partially achieve our task, and their simple augmentations to fit our setting underperform as well. Then, we apply our method to real-world scenarios. We provide examples of manipulating real articulated objects' shape subject to the user command. Further, we spawn our predictions in a virtual environment to simulate robotic manipulations. Our overall contributions are listed as follows.

- We propose a command-driven shape manipulation task for articulated objects. It enables a simple interaction way for users to control the manipulation procedure.
- We present an integrated framework Cart towards this systematic task. It succeeds by understanding the articulated object structure and further connecting user commands to predict movement to the desired state.
- Cart works on both articulation understanding and object manipulation on synthetic and real-world data.

2. Related Work

Analyzing articulated objects. Articulated man-made objects have diverse attributes to explore. Facilitated by the many publicly available datasets [1, 30, 33, 37, 51, 57, 61, 62] and physical simulators [8, 27, 43, 45, 57], research has made great progresses in estimating their 6D poses [28, 50], discovering part-level structures [19, 28, 51, 61, 62], recognizing the articulation relation and joint parameters between parts [13, 15, 16, 19, 22, 31, 38, 41, 47, 51, 58, 61, 64], and predicting manipulation in a robotic system [9, 10, 34, 52, 54, 59].

For vision-based research, a long-standing goal is to allow the machine to interact with articulated objects. Ditto [19] facilitates the virtual interaction by creating an operational twin of physical object in the virtual world. It takes a pair of point clouds as input for estimating the articulated structure of CAD models. Unlike Ditto, we take only a single visual observation plus a simple user command as our input. Also, our target is to control the manipulation of articulated objects by user command, which has great potential for supporting human-instructed robotic object manipulation. There also exists other research [36, 53] on object manipulation. They are based on implicit models, while we explicitly model the articulation and thus support physical operations.

Sensory observation for articulation modeling. To perceive articulated objects, previous work tries different visual inputs, including generating 3D models from multi-view RGB(-D) images [10, 11, 14–16, 49, 53, 64, 66] and processing 3D point clouds [1, 13, 19, 28, 31, 34, 38, 46, 47, 51, 61, 62].

Since point clouds provide rich geometry information, [16, 22, 51, 61, 64] take a single point cloud as input to support part articulation prediction. Another line of work tends to utilize sequential observations [2, 15, 19, 32, 41, 44, 48, 60, 62], as changing the articulation states reveals explicit motion patterns. Yet, collecting sequential data for perception needs active interactions either from human-hand-tuned actions [19] or policy-based robotic manipulation [12, 21, 38]. In this work, we take only a single point cloud as input, yet achieving decent accuracy on articulation modeling. We even show comparable performance with Ditto [19], which takes a point cloud pair as input. Such a result manifests that our approach can mitigate the need for tedious paired data collection and point pair registration.

Parts segmentation of articulated objects. Parts segmentation [6, 7, 20, 24, 35, 63, 65] is crucial for understanding the structures of articulated objects, particularly for supporting part-level manipulation. This task is distinct from classic 3D semantic segmentation [17, 25, 26, 55, 56, 67] or 3D object detection [3, 4, 29] and requires separating different part entities of the same object category at the mask level. The work of [13, 58] directly applies segmentation ground truths to predict articulation attributes.

Recent methods [19, 28, 51, 61] turn to jointly separate the articulated parts and infer the mobility attributes. Several solutions [19, 28] simplify the segmentation issue by assuming a fixed number of object parts. To allow unknown parts

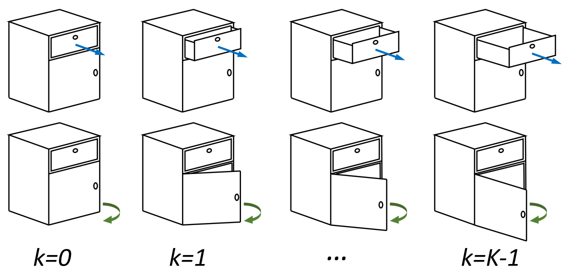


Figure 2. Object shapes under diverse state nodes $\{T^k\}_{k=0}^{K-1}$. A larger k denotes a higher opening extent of articulation joints. We show examples of the prismatic (top) and revolute (bottom) joints.

structure, RPM-Net [61] and Shape2Motion [51] apply point-wise motion prediction to separate object parts. Compared with these methods [51, 61], we formulate a segment-then-predict paradigm that first segments the individual object parts and then leverages this prior for producing more accurate predictions on the articulated structure.

3. Task Definition

We make a system to manipulate articulated objects given (i) a single 3D point cloud of unknown articulated structure and (ii) a manipulation command patternized as "Move <part> to <target state>", where <> is user-given. Users can pick one or more points in the point cloud to indicate the <part> to be manipulated, as well as the desired <target state> of this part after manipulation; see, *e.g.*, Fig. 1.

While user commands can naturally be given in natural human languages, doing so may complicate this task. For articulated objects, see, *e.g.*, the small cabinet shown in Fig. 2, the motion space of its movable parts should range from "fully-closed" to "fully-open". Hence, we indicate the <target state> by a state node T^k, where k is an integer ranged from 0 to K-1 for various opening extents of the part's articulated joint. As Fig. 2 shows, T^0 denotes the fully-closed state, T^{K-1} denotes the fully-open state, whereas others are uniformly-sampled intermediate states. By then, language instructions can be translated into these states through a natural language processing—which is not the focus of this paper.

Our task is formulated as follows. From a point cloud observation $P = \{p_i \in \mathbb{R}^3\}_{i=1}^N$ of an unknown articulated object composed of an arbitrary number of movable parts $L = \{l_j\}_{j=1}^M$, we first predict the articulation parameters $[\{\tau_j, \psi_j\}]_{j=1}^M$ for all the object parts, where N is the point number and M is the movable part number. Given a user command that provides a 3D point (suppose it belongs to the j^{th} part) and the corresponding state node (*e.g.*, T_j^k), we infer the motion parameter φ_j to make the j^{th} part move from its current articulation state to the target. Here, τ_j is the binary joint type (1D prismatic or 1D revolute joint following [19, 28]) and ψ_j is the joint parameter.

For prismatic joints, the joint parameter ψ includes the translation direction $u^p \in \mathbb{R}^3$ and the motion parameter φ is a signed ($\pm$) translation distance along u^p. For revolute joints, ψ consists of the rotational axis $u^r \in \mathbb{R}^3$ and a pivot point $v^r \in \mathbb{R}^3$ on the axis, and φ is a signed ($\pm$) rotation angle, clockwise or counterclockwise about rotation axis u^r.

4. Our Method

We present Cart to achieve articulated object understanding and command-controllable manipulation. It consists of two neural networks, namely Seg-Net and Art-Net. As shown in Fig. 3, Seg-Net is pre-trained first for part segmentation and articulation state modeling. Then, Art-Net makes use of these predictions for predicting articulated models.

Given a user command, Art-Net applies the segmentation results to localize the object part for conducting manipulation and merge the command message into a visual embedding. As the fused part feature includes both geometry characteristics and command meaning, we use it to reason articulation patterns and command-related motion parameters, which together support physical shape manipulations. Then, we propose a Test-time State Adaptation algorithm to regularize the changed articulation states. It employs the state prediction model of Seg-Net to optimize the motion parameters from command supervision. By iterative optimization, TTSA robustly achieves high manipulation performance.

4.1. Seg-Net

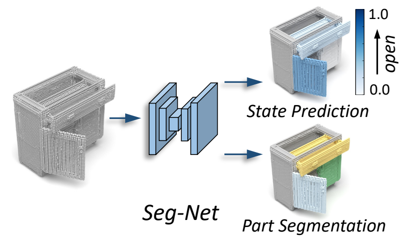
Seg-Net segments objects into multiple rigid parts and predicts per-part articulation states, as shown in Fig. 3a. Given a point cloud $P = \{p_i\}_{i=1}^N$ of an articulated object with unknown articulation states, we extract point-wise feature by a Sparse 3D U-Net [5] then process the feature with three MLP-based branches.

The first branch is to predict per-point class $\hat{c}_i \in \{0, 1\}$ of either movable parts or static one. The second branch predicts per-point state value $\hat{s}_i \in \mathbb{R}$ only for points of movable parts. To mitigate object scale, $\hat{s}_i$ is defined as the normalized pose (with angle or translation distance) compared to a pre-defined rest state. For example, the fully-closed state is 0 and the fully-open one is 1. The third branch produces a collection of per-point offset vectors $\hat{o}_i \in \mathbb{R}^3$. We employ them to shift points of an object part close to the part centroid, so as to separate points from different parts via a clustering.

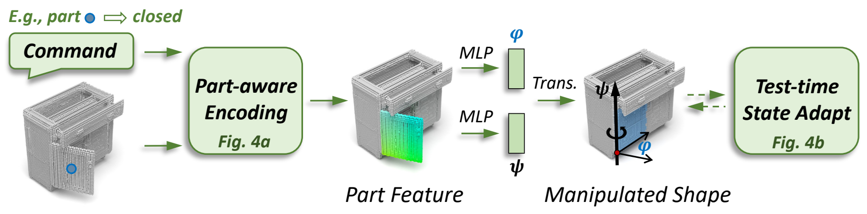
During pre-training, Seg-Net predictions are jointly optimized with a combination of the loss function of

$$\mathcal{L}_{seg} = \frac{1}{N}\sum_{i}^{N}(\mathcal{L}_c(\hat{c}_i, c_i) + \mathcal{L}_s(\hat{s}_i, s_i) + \mathcal{L}_o(\hat{o}_i, o_i)). \quad (1)$$

where c_i, s_i and o_i are ground-truth values, and o_i is an offset vector from the point coordinate p_i to the associated part's centroid coordinate. $\mathcal{L}_c$ is the standard cross-entropy

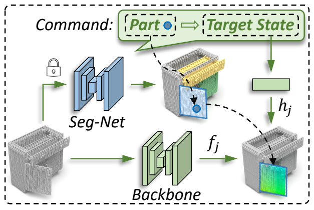


(a) Seg-Net Learning Pipeline

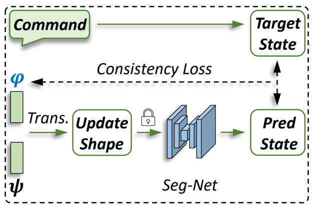


(b) Art-Net Learning Pipeline

Figure 3. Overview of Cart framework. It is formed by Seg-Net and Art-Net. (a) First, we pre-train Seg-Net to segment each movable object part and predict its articulation state. (b) Taking as input a point cloud shape and a command, Art-Net employs a part-aware encoding module (see Fig. 4a) to localize which part(s) to manipulate and obtain the corresponding part(s) feature. Then, we further predict joint parameter ψ and motion parameter φ, thus enabling a concrete manipulation operation. The Test-time State Adaptation (see Fig. 4b) regularizes the object status after manipulation. The pre-trained Seg-Net helps the part-aware encoding module and Test-time State Adaptation.



(a) Part-aware Encoding



(b) Test-time State Adaptation

Figure 4. (a) We first localize the j^{th} part to manipulate by matching the Seg-Net segmentation results with the point coordinate offered by the command. Then, we jointly encode its visual feature f_j and command feature h_j. (b) After manipulation, we predict the articulation state of object (from Seg-Net) and compare it with the target one (from command). The consistency loss, in a backward manner, optimizes the predicted value of motion parameter ψ. For (a) and (b), the network parameters of Seg-Net are fixed.

loss, $\mathcal{L}_s$ is the L1 distance loss, and $\mathcal{L}_o$ imposes constraints on both the point offset L1 distance and direction. It is formulated as

$$\mathcal{L}_o(\hat{o}_i, o_i) = ||o_i - \hat{o}_i|| + (-\frac{o_i}{||o_i||_2} \cdot \frac{\hat{o}_i}{||\hat{o}_i||_2}). \quad (2)$$

When training Art-Net, we make all parameters of Seg-Net *not updated* for feature inference. Specifically, the points of movable parts are first selected by binary classification. Then, they are shifted by offset vectors to form a more compact 3D distribution $\{(p_i + \hat{o}_i)\}$ where the intra-part points are spatially closer. We adopt a breadth-first search [18] to find neighboring points within a sphere of radius r and group these points into a part cluster. The predicted articulation state of each part is a voting result of per-point prediction. Seg-Net output is utilized for the following process.

4.2. Art-Net

The architecture of Art-Net is shown in Fig. 3b. The input to Art-Net is a single point cloud P with a user command. The objective is to manipulate part(s) to the target articulation states specified in the command. For the sake of clarity, we take single-part manipulation as an example. Note that multiple parts can be manipulated in parallel.

Part-aware encoding. This module finds the part to manipulate. As shown in Fig. 4a, as users provide the point coordinate inside an object part, we use the segmentation results of Seg-Net to localize the intact part matched with this point, denoted as $\hat{l}_j$. Then, we represent the target state T_j^k from the command by a K-dimensional one-hot code. To make it learnable, this code will be forwarded by an MLP layer and propagated in part-level to form the command feature $h_j \in \mathbb{R}^{N_j \times d}$. N_j is the point number of part $\hat{l}_j$.

Meanwhile, we employ a PointNet++ [40] network to extract per-point geometrical features, where the point feature of part $\hat{l}_j$ is denoted as $f_j \in \mathbb{R}^{N_j \times d}$. h_j and f_j are concatenated to acquire the fused feature $\overline{f}_j \in \mathbb{R}^{N_j \times 2d}$.

Articulation model estimation. We use $\overline{f}_j$ to estimate the associated articulation model, *i.e.*, joint type τ, joint parameter ψ, and command-related motion parameter φ. Points of $\overline{f}_j$ are from the same object part, which share the consistent articulation attributes and are naturally suitable for predicting global articulation joint. Compared with alternative strategies, this module yields superior performance as validated in Sec. 5.2.

We represent the parameters to be predicted in a per-point representation, so as to enable robust estimation by voting during the inference. Specifically, we form the joint type as $\tau_j \in \{0, 1\}^{N_j}$, the direction of translation and rotation axis as $u_j^p, u_j^r \in \mathbb{R}^{N_j \times 3}$, and the motion parameter as $\varphi_j \in \mathbb{R}^{N_j}$. To estimate the position of revolute joints u^r, we follow the general paradigm of [19, 28] to alternatively predict the projection of each point into the related axis. Thus, v_j^r is represented as $[z_j^r \in \mathbb{R}^{N_j \times 3}, d_j^r \in \mathbb{R}^{N_j}]$, where z^r stands for the unit vector for the projection direction and d^r is the projection distance scalar. We decode the above parameters by different one-layer MLP heads as

$$\begin{aligned} \tau_j = \theta_{type}(\overline{f}_j) \quad & and \quad \varphi_j = \theta_{mot}(\overline{f}_j). \\ u_j^p = \theta_{para}^p(\overline{f}_j) \quad & and \quad [z_j^r, d_j^r] = \theta_{para}^r(\overline{f}_j). \end{aligned} \tag{3}$$

Note that parameters of prismatic and revolute joints are learned individually due to their different motion properties.

Training objective. To supervise the joint type prediction, we employ a binary cross entropy loss $\mathcal{L}_{type}$. To supervise the motion parameter estimation, we use the L1 distance loss $\mathcal{L}_{mot}$ to minimize the difference between the ground-truth and predicted ones. For parameters of the prismatic joint, we use a cosine distance loss, defined as $\mathcal{L}_{para}^p$=arccos$(\hat{u}_j^p \cdot u_j^p)$, to restrict the translation axis direction. Since the parameters of revolute joints are more complicated, we jointly penalize the direction and position difference between the predicted revolute joints and ground-truth with loss

$$\mathcal{L}_{para}^r = \mathcal{L}_{ori}^r(\hat{u}_j^r, u_j^r) + \mathcal{L}_{pos}^r(\hat{z}_j^r, \hat{d}_j^r, z_j^r, d_j^r) + \mathcal{L}_{rot}^r, \tag{4}$$

where $\mathcal{L}_{ori}^r$ =arccos$(\hat{u}_j^r \cdot u_j^r)$ is the loss for axis direction. $\mathcal{L}_{pos}^r$ is the addition of three components: the direction loss arccos$(\hat{z}_j^r \cdot z_j^r)$ for the projection vector, the L1 loss $||\hat{d}_j^r - d_j^r||$ for the projection distance, and a point-axis distance loss.

For the last one, we shift the original point coordinate of part $\hat{l}_j$ towards the rotation axis by adding $(\hat{d}_j^r \cdot \hat{z}_j^r)$. Then, we apply the L1 loss on the distance between these shifted points and ground-truth revolute axis. $\mathcal{L}_{rot}^r$ is an additional loss following work of [15, 19] to confirm the orthogonality of rotation matrix. It is formulated as $\mathcal{L}_{rot}^r = ||\mathbf{I}_{3,3} - \hat{R}_j \cdot R_j||$, where $\hat{R}_j$ and R_j are the predicted and ground-truth rotation matrices, respectively. They are computed from the axis direction and rotation angle based on the Rodirgues's Rotation Formula [42]. By taking into consideration all the articulation constraints, we formulate the loss as

$$\mathcal{L}_{art} = \mathcal{L}_{type} + \mathcal{L}_{mot} + \mathbb{I}^r \mathcal{L}_{para}^r + \mathbb{I}^p \mathcal{L}_{para}^p, \tag{5}$$

where $\mathbb{I}^p$ and $\mathbb{I}^r$ are binary indicators that are dynamically activated depending on the ground-truth joint type.

4.3. Inference

Basic inference. At test time, Seg-Net first clusters the object points into individual movable parts. Art-Net receives the command and localizes the part(s) to manipulate by part-aware encoding module. It uses the part-level point feature to vote for the joint type τ, joint parameters ψ, and command-related motion parameter φ. As φ is a signed value including the movement amplitude and direction, we refer to ψ and φ to translate or rotate the selected part(s) accordingly.

Test-time State Adaptation inference. We observe that the predicted motion parameter φ may not be accurate, *i.e.*, the object state after manipulation do not satisfy the user command. We design a Test-time State Adaptation (TTSA) algorithm to self-refine the motion parameters at test time.

As shown in Fig. 4b, after operating point cloud P to P', we send P' to Seg-Net again to estimate the actual per-part state $\hat{s}'$. For j^{th} part, we compare $\hat{s}_j'$ with the desired state node $\{T_j^k\}$. $\{T_j^k\}$ can be transformed to a normalized state value $s_j' = k/(K-1)$ to allow the quantitative comparison with $\hat{s}_j'$. Then, we use their difference as the loss to update the motion parameter decoder θ_{mot} (defined in Eq. (3)) to correct the motion predictions:

$$\hat{\theta}_{mot} = \underset{\theta_{mot}}{\arg\min} \frac{1}{M'} \sum_{j=1}^{M'} ||\hat{s}_j - s_j||, \tag{6}$$

where M' is the number of manipulated object parts. We use the new $\hat{\theta}_{mot}$ to re-predict the motion parameter φ and update part state $\hat{s}_j'$. We perform iterative optimization several times to make the final state close to the target goal. Thanks to the efficiency of gradient back-propagation, it introduces minimal additional computation costs. Experiments in Sections 5.2 and 5.3 show that TTSA effectively reduces final state errors, especially for data under distribution shift.

5. Experiments

Datasets. We employ two widely-used datasets of articulated objects, Shape2Motion [51] and PartNet-Mobility [57]. We evaluate on nine common object categories from two datasets, with each category comprising 30∼100 instances. For each object category, we generate 10K, 1K, and 1K samples for training, validation, and testing, respectively. Each sample consists of a command and a 3D point cloud of the object. Each object contains one or more movable rigid parts without prior knowledge on structure and articulation state. To generate point clouds with various articulation states, we follow Ditto [19] to import objects into the PyBullet[1] simulator for modifying their articulation states.

Setting on state nodes. In main experiments, we set K=3 by default. For each articulated part, T^0 is the fully-closed state, T^1 is the half-open one, and T^2 is for fully-open notation. For prismatic joints, the fully-open state means the upper limit of part motion range defined in the dataset, and for revolute joints, we uniformly set the fully-open status to 90°.

Evaluation metrics. We adopt two kinds of metrics. The static ones measure the object-structure-understanding performance, and the dynamic metrics measure the command-related performance. The former includes the widely-used part segmentation mAP and the error of the predicted joint type and joint parameter, i.e., τ and ψ in Sec. 3. The latter is the error of predicted motion parameter according to the command, i.e., φ in Sec. 3. More dataset statistics and training details are given in the supplementary material.

[1] https://pybullet.org/wordpress/

			Revolute Joint								Prismatic Joint		
Input	Method	Avg	cabinet	oven	laptop	eyeglass	micro	fridge	stapler	table	cabinet	storage	table
					(Static) Errors of Axis Direction ↓								
Single	RPM-Net [61]	8.37°	4.57°	3.45°	8.27°	15.68°	4.91°	4.11°	17.57°	13.36°	6.32°	6.40°	7.38°
	ANCSH [28]	3.70°	-	2.10°	1.94°	3.86°	1.70°	-	7.53°	-	-	5.08°	-
	Associated Points	2.47°	2.63°	1.42°	1.95°	1.77°	1.33°	0.95°	2.47°	0.94°	4.21°	4.02°	5.45°
	Cart (Ours)	**1.36°**	**0.80°**	0.72°	**0.94°**	**1.16°**	**0.84°**	**0.66°**	**1.05°**	**0.89°**	2.57°	**2.75°**	**2.31°**
Pair	Ditto [19]	1.43°	0.92°	**0.70°**	1.00°	1.25°	0.87°	0.80°	1.10°	0.94°	**2.19°**	3.16°	2.84°
					(Static) Errors of Axis Position ↓								
Single	RPM-Net [61]	0.12	0.12	0.14	0.08	0.14	0.10	0.10	0.21	0.09	-	-	-
	ANCSH [28]	0.06	-	0.08	0.05	0.04	0.04	-	0.08	-	-	-	-
	Associated Points	0.05	0.06	0.03	0.04	0.04	0.04	0.06	0.05	0.07	-	-	-
	Cart (Ours)	**0.03**	**0.04**	**0.02**	**0.02**	**0.03**	0.03	**0.04**	**0.02**	0.04	-	-	-
Pair	Ditto [19]	0.03	0.04	0.02	0.03	0.04	**0.02**	0.04	0.03	**0.03**	-	-	-
					(Dynamic) Errors of Motion Parameter ↓								
Single	ANCSH* [28]	9.92° (0.21)	-	5.98°	7.78°	10.72°	8.77°	-	16.36°	-	-	0.21	-
	State Difference	9.03° (0.11)	14.72°	6.42°	7.31°	6.53°	4.84°	8.83°	13.21°	10.40°	0.12	0.11	0.10
	Cart w/o TTSA	6.02° (0.07)	8.83°	5.29°	6.46°	5.21°	3.49°	4.11°	7.56°	7.25°	0.08	0.07	0.06
	Cart (Ours)	**3.34° (0.02)**	**4.15°**	**2.13°**	**2.42°**	**4.57°**	**2.19°**	**3.82°**	**4.32°**	**3.13°**	**0.03**	**0.03**	**0.02**
Pair	Ditto* [19]	5.19° (0.04)	5.73°	2.58°	2.91°	5.15°	4.50°	4.03°	10.25°	6.43°	0.05	0.05	0.03

Table 1. Quantitative results of articulation estimation including both *static* and *dynamic* parameters. On all object classes, our method achieves the best results when fairly compared to others. It even yields slightly better performance than Ditto [19] where the latter uses stronger prior input. "*" denotes our modification to support predicting *dynamic* motion parameters.

Method									
S2M [51]	48.7	51.9	67.7	59.9	54.3	52.2	60.6	46.6	45.5
RPM-Net [61]	55.5	58.2	69.6	65.2	70.0	69.3	57.9	46.4	45.3
ANCSH [28]	-	73.2	90.5	78.4	69.4	-	82.0	-	61.2
Ditto [19]	-	69.4	89.2	75.1	65.7	-	73.0	-	62.7
Cart	**73.1**	**75.8**	**95.6**	**80.1**	**71.1**	**73.4**	**84.9**	**78.5**	**69.8**

Table 2. Comparison on part segmentation mAP on nine object categories. ANCSH and Ditto assume fixed object structure.

Class	Method	mAP ↑	Direction ↓	Position ↓	Motion ↓
Oven	Ditto	62.1	6.5°	0.08	12.7°
	Ours w/o TTSA	72.0	2.0°	0.03	9.0°
	Ours	**72.0**	**2.0°**	**0.03**	**3.6°**
Fridge	Ditto	-	8.9°	0.09	15.6°
	Ours w/o TTSA	70.5	2.5°	0.04	8.7°
	Ours	**70.5**	**2.5°**	**0.04**	**4.2°**

Table 3. Model performance on unseen object classes. The model is trained on *cabinet* class and tested on *oven* and *fridge* classes.

5.1. Baselines

S2M, RPM-Net, and ANCSH. We select three recent methods that take a single point cloud as input for fair comparison. S2M [51] and RPM-Net [61] first predict possible point-wise displacement, and utilize the inferred mobility information for part and joint estimation. ANCSH [28] transforms a set of object parts to canonical space, so that articulation attributes can be more readily learned. ANCSH relies on a simplified assumption that the number of object parts is fixed, which limits the practical applicability.

Ditto. Ditto [19] is the SOTA approach for articulated object understanding. We note that Cart works on a single input, while Ditto needs a pair of observations of an articulated object before and after interaction. This introduces much more cues for articulation understanding than ours. To predict motion parameters, Ditto cannot receive user commands, so we construct an input pair of the current point cloud and ground-truth after manipulation for Ditto, and use its predicted state difference as the motion value, termed Ditto*.

Associated Points. Compared to our method, it alters the articulation estimation scheme using different point features. These points are no longer from the same part proposal, and instead are spatially close to the related joints. We follow ANCSH [28] to learn the point-joint association scheme.

State Difference. Compared with Cart, we do not directly learn the motion parameter by networks. Instead, we use Seg-Net to predict the initial normalized articulation state and compare it with required one indicated by commands. For manipulation, we convert their state difference to actual motion distance and predict a moving direction, *e.g*., clockwise or counterclockwise for rotation joints.

Cart w/o TTSA. We remove the Test-time State Adaptation algorithm (TTSA) for ablation study.

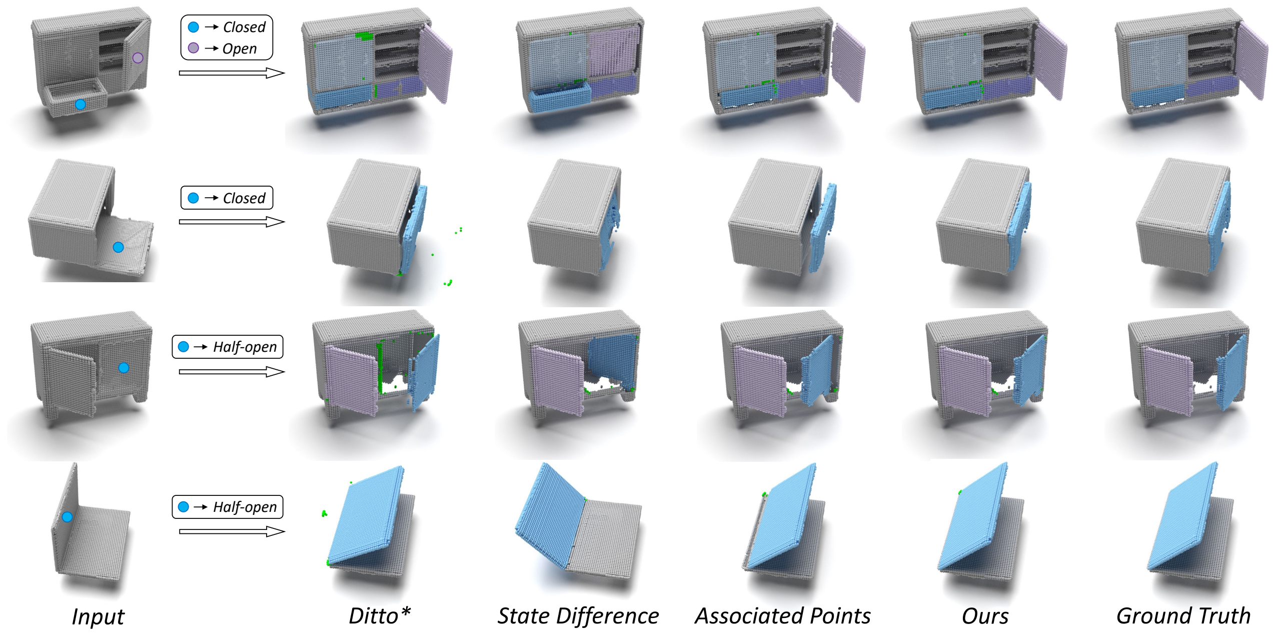


Figure 5. Comparison on visualized results. We use different colors to represent distinct object parts predicted by the models.

5.2. Main Results

Comparison on static parameters. Table 1 shows quantitative evaluations on joint parameter prediction. Cart outperforms two SOTA methods RPM-Net and ANCSH significantly on all object categories. It even achieves comparable performance with Ditto whose input gains extra advantages.

Our superior accuracy is attributed to we leveraging the segmentation prior for voting. After segmentation, the points in each segmented part share the consistent articulation attributes. When they further vote for a global articulation joint, our framework yields better performance. [19, 28, 61] do not utilize this. The comparison between Cart and Associated Points further validates our choice.

Table 2 presents the results of part segmentation, where Cart surpasses other approaches by a large margin in terms of mAP. ANCSH and Ditto both presume a fixed structure for articulated objects. Thus, they can test only on certain object classes. Instead, our clustering-based segmentation is a structure-agnostic paradigm. It allows to apply Cart in a large variety of articulation scenarios.

Comparison on dynamic parameters. All baselines and Cart use command input for fairness. As existing works do not directly support command-based manipulation, they must manually convert commands to the compilable goal states and derive the motion value and direction from the state difference; see Sec. 5.1 for details.

As shown in the green region of Table 1, We try this strategy on ANCSH and our method (*i.e.*, State Difference), because both are able to predict initial articulation states. However, it yields inferior performance. There are two potential reasons. First, the states are normalized values. When converted to actual distances, the prediction errors increase due to the object scale uncertainty. Second, additional estimation on motion direction exposes the shape manipulation to a higher failure risk, which is showcased in Fig. 5.

In contrast, Cart learns a signed motion value directly from the object and command features, which allows for a differentiable optimization paradigm. Moreover, we show that applying Test-time State Adaption (TTSA) effectively reduces prediction errors, indicating that TTSA contributes to better manipulation aligned with the command. With TTSA, Cart attains the best performance, even surpassing Ditto* that takes as input the ground-truth clues.

Visualization results. Fig. 5 shows quantitative visualization comparison to examine the manipulation ability of Cart. We first exhibit the performance of Ditto* that uses enhanced input. It is observed that a number of unsegmented points (highlighted in green color) do not get involved in the manipulation procedure due to the segmentation error, resulting in undesired noise. The predicted axis and motion parameters are also not so accurate, especially for objects with complicated structures.

State difference mainly fails in predicting both correct motion amplitude and direction, *e.g.*, sometimes in opposite direction. Associate Point struggles to accurately localize articulation joints. Compared with them, Cart performs better part segmentation, joint parameter estimation, and motion parameter prediction. These fundamental elements work jointly to guarantee the final manipulation quality.

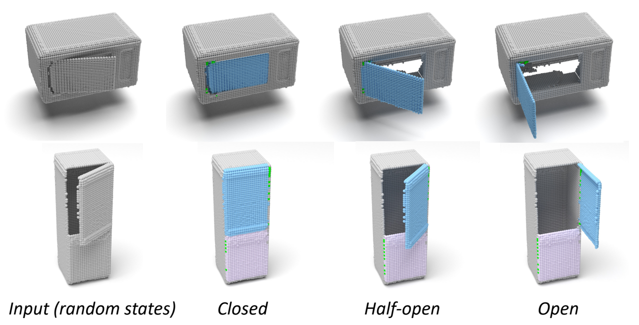


Figure 6. Manipulation effect on unseen object classes. The model is trained on the *cabinet* class and tested on *oven* and *fridge* classes.

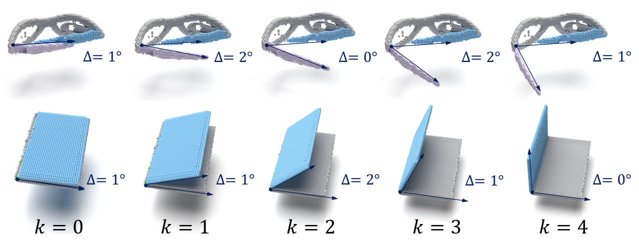


Figure 7. Manipulation experiments driven by more fine-grained state nodes $\{T^k\}_{k=0}^4$. Δ is the angle difference between manipulated states and ground truths.

5.3. Generalization to Unseen Categories

To verify the generalization ability, we test the model on two unseen object categories. Table 3 summarizes the comparison regarding a series of metrics. Our method again outperforms Ditto by a notable margin when the latter adopts a stronger input. The key to achieve inter-class generalization is our structure-agnostic part segmentation. It makes the model easily fit to unseen data to accurately separate different parts.

These results will facilitate downstream articulation estimations. It is noteworthy that our TTSA algorithm works especially well on reducing motion errors. Under distribution shift, this technique makes substantial improvement by learning from unlabeled data. We show the decent manipulation outcome of Cart in Fig. 6.

5.4. More Fine-grained State Nodes

To accommodate more manipulation options, we conduct experiments on a larger group of state nodes. Here, we set K=5 to construct five state nodes $\{T^k\}_{k=0}^4$, which have a finer-grained division of the part motion range. For network training, we encode the state nodes to 5D one-hot codes and integrate them into our pipeline. The manipulation performance is visualized in Fig. 7, where we also denote the angle difference between the states of our results and of ground truths by Δ. The negligible angle errors reveal that Cart is robust to different settings of state nodes.

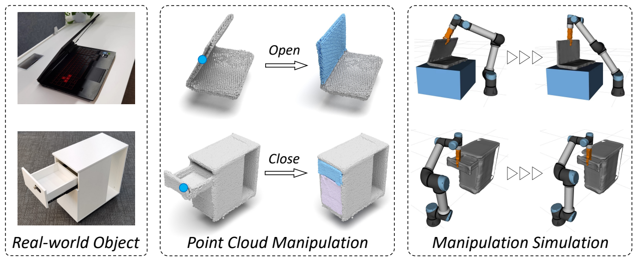


Figure 8. Real-world experiment results on a laptop and a cabinet.

5.5. Real-world Experiments

We apply our method on real-world objects to verify its generalization ability. As shown in Fig. 8, we choose two articulated products, *i.e.*, laptop and cabinet. We scan the point clouds by an iPhone 13Pro RGB-D camera and infer the articulation structures. Then, we give specified commands to close the lid of a laptop, and to push back the drawer of a cabinet. Despite the noisy input points, Cart still manipulates the object shape accurately both for revolute (for laptop) and prismatic (for cabinet) joints.

To simulate a physical manipulation in virtual environment, we further spawn the digital twin of actual objects in Gazebo [23], a robot simulation platform. Following Ditto [19], we employ a recent method [39] to reconstruct the per-part meshes from point clouds and write them together with the articulation parameters into a URDF format, which can represent a robot model and virtually move the object parts. We use a simulated robot arm to manipulate the specified parts with our predicted motion parameters for action planning. By setting other physical properties to default values, the manipulation generally satisfies our instructions.

6. Conclusion

We present Cart, a new approach for understanding the structure properties of articulated objects and, more importantly, manipulating the object shape in response to user commands. Our critical insight is to make user commands connect to visual observations to ensure consistency between manipulated geometry and command requirement. Towards this goal, we develop a part-aware encoding module to aggregate visual and command messages, and a Test-time State Adaptation algorithm for shape regularization. Due to our command-driven paradigm, we expect that the work will empower human-instructed embodied AI research and human-machine interaction applications. The limitations and discussions on work extension are given in the supplementary file for readers to explore these insights.

Acknowledgements. The work is supported by the Research Grants Council of the HKSAR, China [CUHK 14201921] and the Shenzhen Science and Technology Program (KQTD20210811090149095).

References

[1] Ben Abbatematteo, Stefanie Tellex, and George Konidaris. Learning to generalize kinematic models to novel objects. In *CoRL*, 2019. 2

[2] Michael J Black and Allan D Jepson. Eigentracking: Robust matching and tracking of articulated objects using a view-based representation. *International Journal of Computer Vision*, 1998. 2

[3] Yukang Chen, Jianhui Liu, Xiangyu Zhang, Xiaojuan Qi, and Jiaya Jia. Fully sparse voxelnet for 3d object detection and tracking. In *CVPR*, 2023. 2

[4] Yukang Chen, Jianhui Liu, Xiangyu Zhang, Xiaojuan Qi, and Jiaya Jia. Scaling up kernels in 3d sparse cnns. In *CVPR*, 2023. 2

[5] Christopher Choy, JunYoung Gwak, and Silvio Savarese. 4d spatio-temporal convnets: Minkowski convolutional neural networks. In *CVPR*, 2019. 3

[6] Ruihang Chu, Yukang Chen, Tao Kong, Lu Qi, and Lei Li. Icm-3d: Instantiated category modeling for 3d instance segmentation. *IEEE Robotics and Automation Letters*, 2021. 2

[7] Ruihang Chu, Xiaoqing Ye, Zhengzhe Liu, Xiao Tan, Xiaojuan Qi, Chi-Wing Fu, and Jiaya Jia. Twist: Two-way inter-label self-training for semi-supervised 3d instance segmentation. In *CVPR*, 2022. 2

[8] Matt Deitke, Winson Han, Alvaro Herrasti, Aniruddha Kembhavi, Eric Kolve, Roozbeh Mottaghi, Jordi Salvador, Dustin Schwenk, Eli VanderBilt, Matthew Wallingford, et al. Robothor: An open simulation-to-real embodied ai platform. In *CVPR*, 2020. 2

[9] Ben Eisner*, Harry Zhang*, and David Held. Flowbot3d: Learning 3d articulation flow to manipulate articulated objects. In *Robotics: Science and Systems (RSS)*, 2022. 2

[10] Samir Yitzhak Gadre, Kiana Ehsani, and Shuran Song. Act the part: Learning interaction strategies for articulated object part discovery. In *ICCV*, 2021. 2

[11] Muzhi Han, Zeyu Zhang, Ziyuan Jiao, Xu Xie, Yixin Zhu, Song-Chun Zhu, and Hangxin Liu. Reconstructing interactive 3d scenes by panoptic mapping and cad model alignments. In *ICRA*, 2021. 2

[12] Karol Hausman, Scott Niekum, Sarah Osentoski, and Gaurav S Sukhatme. Active articulation model estimation through interactive perception. In *ICRA*, 2015. 2

[13] Ruizhen Hu, Wenchao Li, Oliver Van Kaick, Ariel Shamir, Hao Zhang, and Hui Huang. Learning to predict part mobility from a single static snapshot. *TOG*, 2017. 2

[14] Tao Hu, Liwei Wang, Xiaogang Xu, Shu Liu, and Jiaya Jia. Self-supervised 3d mesh reconstruction from single images. In *CVPR*, 2021. 2

[15] Ajinkya Jain, Rudolf Lioutikov, Caleb Chuck, and Scott Niekum. Screwnet: Category-independent articulation model estimation from depth images using screw theory. In *ICRA*, 2021. 2, 5

[16] Hanxiao Jiang, Yongsen Mao, Manolis Savva, and Angel X Chang. Opd: Single-view 3d openable part detection. In *ECCV*, 2022. 2

[17] Li Jiang, Shaoshuai Shi, Zhuotao Tian, Xin Lai, Shu Liu, Chi-Wing Fu, and Jiaya Jia. Guided point contrastive learning for semi-supervised point cloud semantic segmentation. In *ICCV*, 2021. 2

[18] Li Jiang, Hengshuang Zhao, Shaoshuai Shi, Shu Liu, Chi-Wing Fu, and Jiaya Jia. Pointgroup: Dual-set point grouping for 3d instance segmentation. In *CVPR*, 2020. 4

[19] Zhenyu Jiang, Cheng-Chun Hsu, and Yuke Zhu. Ditto: Building digital twins of articulated objects from interaction. In *CVPR*, 2022. 1, 2, 3, 4, 5, 6, 7, 8

[20] Evangelos Kalogerakis, Melinos Averkiou, Subhransu Maji, and Siddhartha Chaudhuri. 3d shape segmentation with projective convolutional networks. In *CVPR*, 2017. 2

[21] Dov Katz and Oliver Brock. Manipulating articulated objects with interactive perception. In *ICRA*, 2008. 2

[22] Yuki Kawana, Yusuke Mukuta, and Tatsuya Harada. Unsupervised pose-aware part decomposition for 3d articulated objects. *arXiv preprint arXiv:2110.04411*, 2021. 2

[23] Nathan Koenig and Andrew Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *ROS*, 2004. 8

[24] Hamid Laga, Michela Mortara, and Michela Spagnuolo. Geometry and context for semantic correspondences and functionality recognition in man-made 3d shapes. *TOG*, 2013. 2

[25] Xin Lai, Yukang Chen, Fanbin Lu, Jianhui Liu, and Jiaya Jia. Spherical transformer for lidar-based 3d recognition. In *CVPR*, 2023. 2

[26] Xin Lai, Jianhui Liu, Li Jiang, Liwei Wang, Hengshuang Zhao, Shu Liu, Xiaojuan Qi, and Jiaya Jia. Stratified transformer for 3d point cloud segmentation. In *CVPR*, 2022. 2

[27] Chengshu Li, Fei Xia, Roberto Martín-Martín, Michael Lingelbach, Sanjana Srivastava, Bokui Shen, Kent Elliott Vainio, Cem Gokmen, Gokul Dharan, Tanish Jain, Andrey Kurenkov, Karen Liu, Hyowon Gweon, Jiajun Wu, Li Fei-Fei, and Silvio Savarese. igibson 2.0: Object-centric simulation for robot learning of everyday household tasks. In *CoRL*, 2021. 2

[28] Xiaolong Li, He Wang, Li Yi, Leonidas J Guibas, A Lynn Abbott, and Shuran Song. Category-level articulated object pose estimation. In *CVPR*, 2020. 1, 2, 3, 4, 6, 7

[29] Jianhui Liu, Yukang Chen, Xiaoqing Ye, Zhuotao Tian, Xiao Tan, and Xiaojuan Qi. Spatial pruned sparse convolution for efficient 3d object detection. In *NeurIPS*, 2022. 2

[30] Liu Liu, Wenqiang Xu, Haoyuan Fu, Sucheng Qian, Qiaojun Yu, Yang Han, and Cewu Lu. Akb-48: A real-world articulated object knowledge base. In *CVPR*, 2022. 2

[31] Liu Liu, Han Xue, Wenqiang Xu, Haoyuan Fu, and Cewu Lu. Toward real-world category-level articulation pose estimation. *IEEE Transactions on Image Processing*, 2022. 2

[32] Zhijian Liu, Jiajun Wu, Zhenjia Xu, Chen Sun, Kevin Murphy, William T. Freeman, and Joshua B. Tenenbaum. Modeling parts, structure, and system dynamics via predictive learning. In *ICLR*, 2019. 2

[33] Roberto Martín-Martín, Clemens Eppner, and Oliver Brock. The rbo dataset of articulated objects and interactions. *The International Journal of Robotics Research*, 2019. 2

[34] Kaichun Mo, Leonidas J Guibas, Mustafa Mukadam, Abhinav Gupta, and Shubham Tulsiani. Where2act: From pixels to actions for articulated 3d objects. In *ICCV*, 2021. 2

[35] Kaichun Mo, Shilin Zhu, Angel X Chang, Li Yi, Subarna Tripathi, Leonidas J Guibas, and Hao Su. Partnet: A large-scale benchmark for fine-grained and hierarchical part-level 3d object understanding. In *CVPR*, 2019. 2

[36] Jiteng Mu, Weichao Qiu, Adam Kortylewski, Alan Yuille, Nuno Vasconcelos, and Xiaolong Wang. A-sdf: Learning disentangled signed distance functions for articulated shape representation. In *ICCV*, 2021. 2

[37] Tongzhou Mu, Zhan Ling, Fanbo Xiang, Derek Cathera Yang, Xuanlin Li, Stone Tao, Zhiao Huang, Zhiwei Jia, and Hao Su. Maniskill: Generalizable manipulation skill benchmark with large-scale demonstrations. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021. 2

[38] Neil Nie, Samir Yitzhak Gadre, Kiana Ehsani, and Shuran Song. Structure from action: Learning interactions for articulated object 3d structure discovery. *arxiv*, 2022. 2

[39] Songyou Peng, Michael Niemeyer, Lars Mescheder, Marc Pollefeys, and Andreas Geiger. Convolutional occupancy networks. In *ECCV*, 2020. 8

[40] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *NeurIPS*, 2017. 4

[41] Shengyi Qian, Linyi Jin, Chris Rockwell, Siyi Chen, and David F Fouhey. Understanding 3d object articulation in internet videos. In *CVPR*, 2022. 2

[42] Olinde Rodrigues. Des lois géométriques qui régissent les déplacements d'un système solide dans l'espace, et de la variation des coordonnées provenant de ces déplacements considérés indépendamment des causes qui peuvent les produire. *J. Math. Pures Appl*, 1840. 5

[43] Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, Devi Parikh, and Dhruv Batra. Habitat: A platform for embodied ai research. In *ICCV*, 2019. 2

[44] Tanner Schmidt, Richard A Newcombe, and Dieter Fox. Dart: Dense articulated real-time tracking. In *Robotics: Science and Systems*, 2014. 2

[45] Bokui Shen, Fei Xia, Chengshu Li, Roberto Martín-Martín, Linxi Fan, Guanzhi Wang, Claudia Pérez-D'Arpino, Shyamal Buch, Sanjana Srivastava, Lyne Tchapmi, Micael Tchapmi, Kent Vainio, Josiah Wong, Li Fei-Fei, and Silvio Savarese. igibson 1.0: A simulation environment for interactive tasks in large realistic scenes. In *IROS*, 2021. 2

[46] Yahao Shi, Xinyu Cao, Feixiang Lu, and Bin Zhou. P^3-net: Part mobility parsing from point cloud sequences via learning explicit point correspondence. In *AAAI*, 2022. 2

[47] Yahao Shi, Xinyu Cao, and Bin Zhou. Self-supervised learning of part mobility from point cloud sequence. In *Computer Graphics Forum*, 2021. 2

[48] Aliaksandr Siarohin, Oliver J Woodford, Jian Ren, Menglei Chai, and Sergey Tulyakov. Motion representations for articulated animation. In *CVPR*, 2021. 2

[49] Wei-Cheng Tseng, Hung-Ju Liao, Yen-Chen Lin, and Min Sun. Cla-nerf: Category-level articulated neural radiance field. In *ICRA*, 2022. 2

[50] He Wang, Srinath Sridhar, Jingwei Huang, Julien Valentin, Shuran Song, and Leonidas J Guibas. Normalized object coordinate space for category-level 6d object pose and size estimation. In *CVPR*, 2019. 2

[51] 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 *CVPR*, 2019. 1, 2, 3, 5, 6

[52] Yian Wang, Ruihai Wu, Kaichun Mo, Jiaqi Ke, Qingnan Fan, Leonidas Guibas, and Hao Dong. Adaafford: Learning to adapt manipulation affordance for 3d articulated objects via few-shot interactions. In *ECCV*, 2022. 2

[53] Fangyin Wei, Rohan Chabra, Lingni Ma, Christoph Lassner, Michael Zollhöfer, Szymon Rusinkiewicz, Chris Sweeney, Richard Newcombe, and Mira Slavcheva. Self-supervised neural articulated shape and appearance models. In *CVPR*, 2022. 2

[54] Ruihai Wu, Yan Zhao, Kaichun Mo, Zizheng Guo, Yian Wang, Tianhao Wu, Qingnan Fan, Xuelin Chen, Leonidas Guibas, and Hao Dong. VAT-mart: Learning visual action trajectory proposals for manipulating 3d ARTiculated objects. In *ICLR*, 2022. 2

[55] Xiaoyang Wu, Yixing Lao, Li Jiang, Xihui Liu, and Hengshuang Zhao. Point transformer v2: Grouped vector attention and partition-based pooling. In *NeurIPS*, 2022. 2

[56] Xiaoyang Wu, Xin Wen, Xihui Liu, and Hengshuang Zhao. Masked scene contrast: A scalable framework for unsupervised 3d representation learning. In *CVPR*, 2023. 2

[57] 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 *CVPR*, 2020. 2, 5

[58] Xianghao Xu, Yifan Ruan, Srinath Sridhar, and Daniel Ritchie. Unsupervised kinematic motion detection for part-segmented 3d shape collections. In *SIGGRAPH*, pages 1–9, 2022. 2

[59] Zhenjia Xu, He Zhanpeng, and Shuran Song. Umpnet: Universal manipulation policy network for articulated objects. *IEEE Robotics and Automation Letters*, 2022. 2

[60] Jingyu Yan and Marc Pollefeys. A general framework for motion segmentation: Independent, articulated, rigid, non-rigid, degenerate and non-degenerate. In *ECCV*, 2006. 2

[61] Zihao Yan, Ruizhen Hu, Xingguang Yan, Luanmin Chen, Oliver van Kaick, Hao Zhang, and Hui Huang. Rpm-net: Recurrent prediction of motion and parts from point cloud. *TOG*, 2019. 1, 2, 3, 6, 7

[62] Li Yi, Haibin Huang, Difan Liu, Evangelos Kalogerakis, Hao Su, and Leonidas J. Guibas. Deep part induction from articulated object pairs. *TOG*, 2018. 2

[63] Li Yi, Hao Su, Xingwen Guo, and Leonidas J Guibas. Syncspeccnn: Synchronized spectral cnn for 3d shape segmentation. In *CVPR*, 2017. 2

[64] Vicky Zeng, Tabitha Edith Lee, Jacky Liang, and Oliver Kroemer. Visual identification of articulated object parts. In *IROS*, 2021. 2

[65] Biao Zhang and Peter Wonka. Point cloud instance segmentation using probabilistic embeddings. In *CVPR*, 2021. 2

[66] Ge Zhang, Or Litany, Srinath Sridhar, and Leonidas Guibas. Strobenet: Category-level multiview reconstruction of articulated objects. *arXiv preprint arXiv:2105.08016*, 2021. 2

[67] Zhisheng Zhong, Jiequan Cui, Yibo Yang, Xiaoyang Wu, Xiaojuan Qi, Xiangyu Zhang, and Jiaya Jia. Understanding imbalanced semantic segmentation through neural collapse. *arXiv preprint arXiv:2301.01100*, 2023. 2