Structure from Action: Learning Interactions for 3D Articulated Object Structure Discovery

Neil Nie Samir Yitzhak Gadre Kiana Ehsani Shuran Song

***Abstract*— We introduce Structure from Action (SfA), a framework to discover 3D part geometry and joint parameters of unseen articulated objects via a sequence of inferred interactions. Our key insight is that 3D interaction and perception should be considered in conjunction to construct 3D articulated CAD models, especially for categories not seen during training. By selecting informative interactions, SfA discovers parts and reveals occluded surfaces, like the inside of a closed drawer. By aggregating visual observations in 3D, SfA accurately segments multiple parts, reconstructs part geometry, and infers all joint parameters in a canonical coordinate frame. Our experiments demonstrate that a SfA model trained in simulation can generalize to many unseen object categories with unknown kinematic structures and to real-world objects. Empirically, SfA outperforms a pipeline of state-of-the-art components by 25.0 3D IoU points. Code and data will be publicly available.**

I. Introduction

For robots to be useful out-of-the-box, they must handle a variety of objects—even those that are unfamiliar. Beyond rigid objects, articulated objects, like drawers and microwaves, are of particular interest [1], [29], [12], especially in household use-cases. For tasks involving novel articulated objects, recovering 3D articulated CAD models (e.g., URDFs) is a promising starting point, as they are immediately useful in task-specific planning pipelines [42], [5], [6], [7], [28]. For instance, recovering models of kitchen drawers can lay the foundation for downstream planning to retrieve objects within them. To discover the structure of objects beyond training categories, there is evidence that interaction is critical [12], [51]. Informative interaction allows an agent to expose kinematic constraints (e.g., prismatic or revolute joints) and observe occluded part geometry.

Inferring joints, kinematic constraints, and the full 3D structure of articulated objects is a complex task that involves tackling a diverse set of challenges:

- **Inferring informative interactions.** Given unstructured point clouds, an agent must act intentionally to expose structures. Random actions are insufficient as they may move the object rigidly or not at all, giving no signal about articulation. Similarly, repetitive actions on single parts can lead to an incomplete recovery of parts and joints.
- **Persistent parts aggregation in 3D.** From an observed sequence of interactions, it is necessary to discover new parts and track existing parts, even in the presence of severe occlusion. If an agent closes a drawer, the part should persist within the object representation, even when it is not directly visible in the following steps.
- **Cross-category generalization.** The algorithm should handle object categories unseen during training, with unknown kinematic structures.

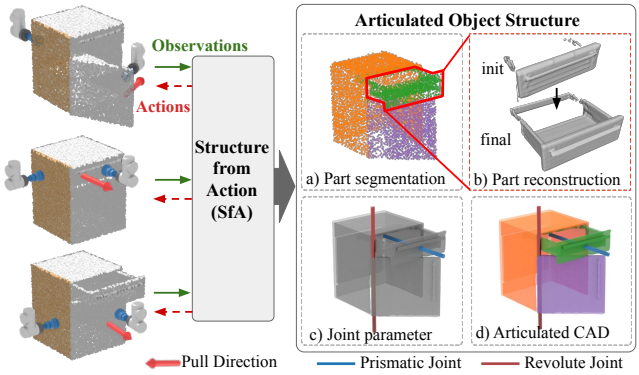

Fig. 1. **Structure from Action.** Our framework discovers an object's structure through a sequence of 3D interactions. The resulting structure includes a) part segmentation, b) 3D reconstruction for each part, and c) joint parameters, together describing d) a 3D articulated CAD model.

These challenges have motivated simplifying assumptions in prior works. For example, Gadre, *et al.*[12] relaxes the problem to a 2D setting, not attempting to reconstruct 3D structure. Jiang, *et al.*[17] consider a single heuristic interaction step, which is insufficient to recover multiple parts and joints. In this work, we relax these assumptions and introduce an approach for the problem of constructing articulated CAD models of 3D objects using interactions.

To address these challenges comprehensively, we introduce *Structure from Action (SfA)* to expose the kinematic structure of objects through interaction. Our *key insight* is that 3D interaction and perception must be considered in conjunction to construct 3D articulated CAD models. Specifically, SfA learns 1) a sequential interaction policy to expose the object's hidden part geometry and kinematics, 2) a dynamic part reconstruction module that segments and completes the object parts by aggregating visual observations, and 3) a joint estimation module that infers object joint types and parameters from the observed motion. The final output of the system is a 3D articulated CAD model of the observed object as seen in Fig. 1.

We evaluate SfA on unseen object instances and categories from the PartNet-Mobility [8], [30], [49] dataset. Our results validate the following contributions:

- An interaction policy that is capable of learning informative interaction strategies in 3D to recover 3D articulated object structure.
- A learnable perception module that aggregates visual observations on-the-fly to improve the accuracy for part reconstruction and joint estimation.

TABLE I

COMPARISON TO PRIOR WORK.

Method	Actions		Object Structure			
	Inferred	3D	Recon.	Seg.	Joint	Multi.
Where2Act [29]	✓	✓	✗	✗	✗	✗
UMP-Net [51]	✓	✓	✗	✗	✓	✗
Ditto [17]	✗	–	✓	✓	✓	✗
AtP [12]	✓	✗	✗	✓	✗	✓
SfA (Ours)	✓	✓	✓	✓	✓	✓

- A single SfA model (both the interaction and perception modules) trained in simulation can generalize to many unseen object categories with unknown kinematic structures, and to real-world objects.

II. Related Work

Recently, interactive perception with articulated objects has gained renewed interest. As an overview, we contrast our method with recent work in Tab. I. Our goal is to recover objects' articulation structure, including objects' part reconstruction (Recon.), segmentation (Seg.), and joints estimation (Joint). An algorithm should also handle objects with multiple (Multi.) parts. While prior work tackles some of these challenges, SfA presents a comprehensive framework addressing all facets of the problem.

Articulated object manipulation. Articulated objects are an important class of objects for manipulation, and the community has come a long way to make datasets and benchmarks to facilitate research in this direction [8], [30], [27], [49], [32], [22]. There is a line of work tackling the problem of interacting with articulated objects to move their parts [29], [48], [28], [40]. Some work [24], [3] uses dual-arm manipulators to enable a more complex interaction. This work mostly focuses on interacting with the purpose of completing a high-level task (such as opening cabinets [40], etc.). Our goal is to learn to interact with objects with the goal of discovering joints, parts, and dynamics of articulation. Eisner, *et al.* [11] propose a vision-based method to predict the flow and articulated motions of an object. However, they do not infer part segmentation or joints. Xu, *et al.* [51] propose a single image-based policy network to recover joint axes, but they do not attempt to recover parts.

Perception from passive observation. Prior work has used a variety of methods to recover object joint constraints, such as using dense pose fitting [10], adapting neural radiance field [34], inferring kinematic graphs [2], and semantic segmentation [47]. Mu, *et al.* [31] propose a model to generate shapes of articulated objects at unseen angles. These methods require prior knowledge of the object or are category-dependent. Moreover, researchers have addressed the part segmentation and structure recovery from non-sequential data (e.g., a single view or point cloud) [55], [45], [44], [14], [20], [1], [21], [36], [37], [46], [13]. In contrast, our method uses a sequence of data, which enables discovering parts of unseen object categories without prior knowledge. The community has tried to recover and track object structures from motion cues between sequential observations [15], [16], [23], [4], [52], [50], [54], [39], [26], [**?**], [41], [33], [53], [38], [17], [55]. However, these methods rely on motion existing in the scene to provide sufficient perceptual cues of part articulation. Our method uses previous observations to predict actions that result in informative motions.

Perception from the interaction. Classical approaches use hand-tuned actions to create informative motion for downstream perception [43], [18], [35]. In contrast, we use a generalizable approach to predict the actions even on novel categories. Similarly, more modern approaches focus on perception, using scripted robot actions [17]. This method also only applies to one stage and fails to identify more than one part. Kumar, *et al.* [19] recover the mass distribution of the articulated objects using interaction, but they do not recover joints or parts. Gadre, *et al.* [12] proposed a method that learns both interaction and perception. However, they only consider 2D perception, limiting the approach to work only for revolute joints visible from the top-down view. In contrast, by using 3D actions and perception, we are able to consider both revolute and prismatic joints and relax restrictions on camera positioning. Recently, Lv, *et al.* [25] proposed SAGCI, an interactive perception method for articulated object structure discovery using a differentiable physics engine. However, it does not explicitly completes part geometry using history observations, nor persistently represent the part geometry during occlusion, which are functionalities supported by SfA.

III. Structure From Action

We introduce Structure from Action (SfA), a learning framework to interact with articulated objects to discover their parts and joints. Our framework is agnostic to object category and to the number of parts and joints that constitute an object. SfA, then, can generalize to novel categories. Given an observation P_0, the initial RGB point cloud before any interaction, SfA infers actions to reveal objects' parts and joint structure (§ III-A). Then by observing the object motion, SfA discovers and reconstructs the object part using a part aggregation module (§ III-B) and infers joint parameters using a joint estimation module (§ III-C). Over several timesteps, the output of the algorithm is an articulated CAD model consisting of 3D part meshes along with the revolute and prismatic joints that connect them (§ III-D). Fig. 2 give an overview of our approach.

A. Learning to Interact with Articulated Parts

The first step of SfA is to infer informative actions to reveal an object's kinematic structure. An action is informative if it isolates an individual part, instead of moving the whole object or multiple parts. Furthermore, an informative action should attempt to move new parts, instead of interacting with the same part repeatedly over many timesteps.

Action representation. Inspired by AtP [12], we assume a bimanual embodied agent, which uses two suction grippers to simultaneously hold and push different parts of the object. These interactions allow the agent to isolate a single part of the object and are particularly useful for small objects

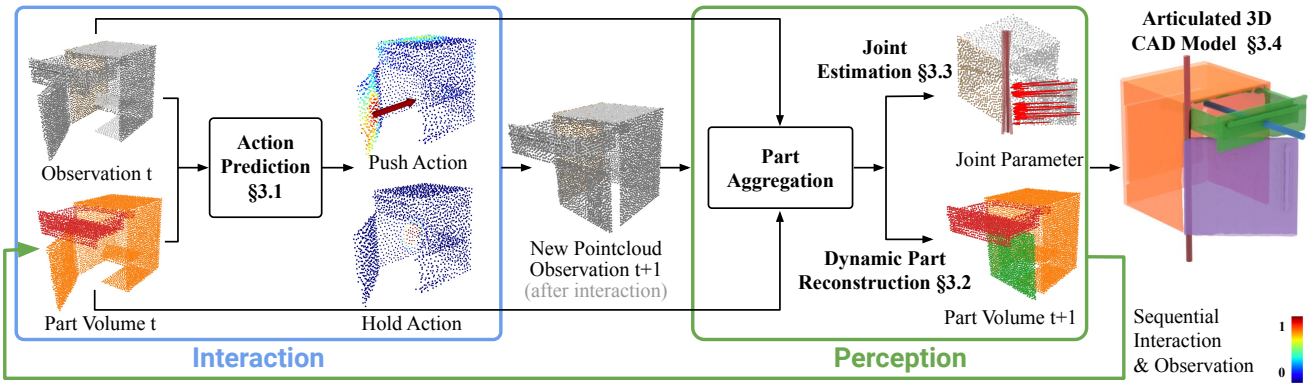

Fig. 2. **Approach overview.** Given an RGB point cloud observation of an unknown articulated object, SfA infers and executes a sequence of informative actions (§ III-A), discovers and reconstructs parts (§ III-B), estimates joint parameters (§ III-C), and outputs an articulated 3D CAD model of the object (§ III-D).

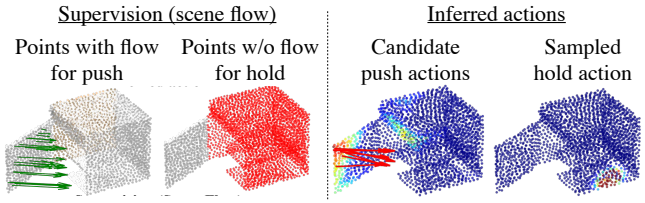

Fig. 3. **Learning Interaction Policy.** *(left)* During training, the 3D scene flow is used to supervise the action directions (green). For a timestep, areas where flow is zero are assumed to be good hold locations (red). *(right)* Inferred candidate push actions conditioned on a sampled hold action.

without a fixed base. However, unlike AtP, we consider a *continuous 3D action space* instead of a discrete 2D action space. This provides the flexibility to handle parts rotating and sliding about arbitrary axes. We represent a hold as a 3D point location. We define a push to be a 3D point location and 3D direction along which an agent applies a fixed force. This formulation makes no explicit distinction between pulling and pushing. For consistency, we refer to the actions as push. In terms of the mechanics of pulling, we assume that the agent has access to a suction gripper that can be used, for example, to pull out a drawer.

Action inference. The input to the action inference module is the current object observation P_t and part history voxel volume $\mathcal{H}_t$. The object observation is represented as a point cloud $P_t \in \mathbb{R}^{2048\times 9}$, formed by farthest point sampling over posed multi-view RGB-D images after ground plane removal. The 9 channels encode the point's XYZ location, 3D surface normal, and RGB color. A part history volume $\mathcal{H}_t$ encodes the agent's current belief about the object's part segmentation and is spatially aligned with P_t (see §III-B for more details on $\mathcal{H}_t$). We wish to associate each point with its current segmentation prediction. Hence, we concatenate each point in P_t with its corresponding value from $\mathcal{H}_t$, before passing the points into the action inference module. Action inference is hence conditioned on the current belief about the part segmentation. Intuitively, we want inferred actions to push parts that are not already confidently segmented so that the downstream perceptual model (§III-B) is able to discover these new parts.

The action inference module is composed of two point transformer encoder-decoders [56], the first to infer a hold score for each point and the second to infer a push action for each point conditioned on a sampled hold location. To predict the hold action, the network infers a score for every point. A higher score indicates a better hold location. To predict push prediction conditioned on the selected hold action, we compute the point-wise distance to the selected hold location and use it as additional input to the push network. The push network output the flow vector for each input point, where the vector directions indicate the inferred push directions and the magnitude indicates the push score of the action. The action with the highest score is selected for execution. At inference, the hold and push are executed in tandem by the agent.

Supervision and dataset creation. We observe that 3D scene flow on a part implies effective push actions on that part. The direction of a good push action is aligned with flow vectors, while the magnitude of each flow vector gives a notion of how effective a push is. Take for instance a door on a cabinet that swings open. Locations with larger flow magnitudes correspond to points farther away from the revolute axis. Interacting with such points is more likely to create discernible motion given a push action with a fixed force. Similarly, points with no movements (i.e., no flow) can be used as candidates for the hold action. While all points without flow are not always equally good for holding, our results suggest that this approximate supervision is sufficient in practice. To ensure push action affordance consistency, we normalize the 3D scene flow vectors before using them for supervision during training.

Based on this intuition, we generate a supervised dataset using the PyBullet [9] simulator and URDF assets from PartNet-Mobility [30]. We move a single part per step by changing its simulation joint state directly. Once a part has moved we consider it *discovered*. We repeat this process for five timesteps per object, moving parts that have not been discovered before moving parts that have already moved. At each timestep, we save the point cloud generated from posed RGBD views, scene flow per point, and the ground truth part labels, with a single label for undiscovered parts and unique labels for each discovered part. In essence, once a part has

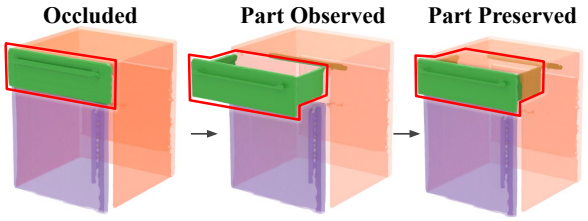

Fig. 4. **Dynamic part reconstruction.** completes the part geometry by aggregating all past observations in a spatially consistent manner.

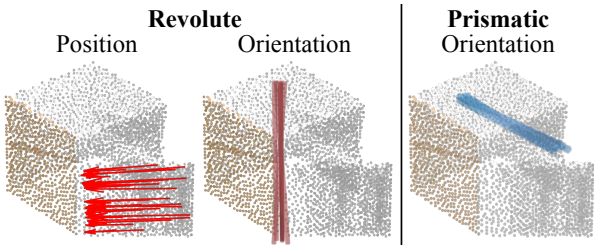

Fig. 5. **Joint Inference.** (Left) Revolute joint position and axis orientation votes. (Right) Prismatic joint orientation votes.

moved, we generate a categorical label for it.

Training. By using 3D flow as supervision and ground truth part labels as input, the interaction model learns to first interact with parts that have not been moved before interacting with discovered parts. With this supervision, the push network is trained to predict 3D scene flow using MSE loss, and the hold network is trained to label points without motion, which can be learned with binary cross-entropy loss.

B. Learning Persistent Part Aggregation

The goal of the part aggregation module is to construct a history volume $\mathcal{H}_t$ that encodes the agent's current belief of the object structure (i.e., segmentation and geometry) from all the past observations. Performing such part aggregation is challenging since it requires the algorithm to establish reliable correspondences between the part before and after the movement. Here, point-to-point correspondences are insufficient as large portions of the surface may disappear (e.g., a drawer as it closes). To tackle these challenges, we propose a learning-based part aggregation module.

We choose to use volumetric representation to allow the network better leverage the spatial alignment between different observations and the history volume. We represent $\mathcal{H}_t$ as a 3D volume $\mathcal{H}_t \in \mathbb{R}^{N\times 96\times 96\times 96}$, which is aligned to the current observation in the world frame. Different from the point representation used in Action inference, which requires high precision of surface locations, the history aggregation module should leverage the spatial alignment between different observations and should infer the occupancy of parts across interaction steps. Hence a volumetric representation that tags regions of space is beneficial. The N channel dimensions per spatial location store a distribution over part indices, with the first channel representing free space. A large probability in the i-th channel of a voxel indicates a high likelihood of this voxel belonging to the i-th part. We allow the max number of parts to be six.

$\mathcal{H}_0$ is initialized with all occupied voxels from the initial point cloud observation P_0 assigned to the first part with probability 1 and all other channels 0. We track assigned and unassigned channels across interaction steps with a pointer k that indexes the next channel to be assigned. Over a few interactions $\mathcal{H}_0$ is refined to more accurately capture the various parts that make up the object. When a new part is discovered, the voxels associated with the new part are assigned to channel k, k is incremented. If a discovered part (say i-th part) gets moved again, the part aggregation module updates the occupancy of i-th channel to reflect new observations, such as to complete its geometry as more surfaces are observed (Fig. 4 2nd step) or preserve its shape when being moved into occlusion (Fig. 4 3rd step). The model also learns to copy over labels of the stationary parts to maintain parts' permanence across interaction steps.

Part aggregation network. The aggregation network is modeled as 3D CNN. It takes the history $\mathcal{H}_{t-1}$ and voxelized point cloud V_{t-1}, V_t as input, and outputs a new history $\mathcal{H}_t$. V_t carries the same information as the point could observation P_t, but in a volumetric format. This 3D volume $V_t \in \mathbb{R}^{96\times 96\times 96\times 7}$ is centered around the first observation point could P_0, encompass $2\times 2\times 2$ 3D space (unit up to scale). The 7 channel encodes the object's occupancy (1D), surface normal (3D) and color (3D). The network and trained with voxel-wise cross-entropy loss between the predicted and target volume.

Supervision. We construct the target history volume $\mathcal{H}_t^{\text{gt}}$ together with the offline data generation process described in §III-A. At each step t, the target volume includes channels for the parts moved by the agent and allocates new channels if new parts are observed. For each part channel, the target volume will include all surfaces that the camera has observed in any of the past and current steps $\in (0, t]$, including surfaces that get occluded in this step. A voxel that has never been observed will be ignored. Since $\mathcal{H}_t^{\text{gt}}$ is generated with a consistent part index across steps, it naturally encourages the network to keep track of part identity over time after the discovery step. However, we do not require it to perform voxel-level correspondence tracking. Moreover, since $\mathcal{H}_t^{\text{gt}}$ introduces part geometry incrementally for each step (only after the surface is observed). It allows the network to learn how to "aggregate" existing observations without the need to "guess" the unobserved part geometry, which allows the network to generalize to new object categories. Finally, since the $\mathcal{H}_t^{\text{gt}}$ preserves the part geometry once it is observed, it allows the network to learn object permanence during occlusion. As a result, this part aggregation module is able to discover, track, and reconstruct the object part geometry using a single network.

C. Joint Inference

Apart from the part information, it is also critical to infer the object's joint parameters to fully recover its kinematic structure. To do so, we designed a joint inference module that infers the object's joint type and parameters from two consecutive object observations P_{t-1} and P_t with object

motion. If no part has moved, this interaction step will be ignored for joint prediction.

With the learned action policy (i.e., simultaneously holding and pushing different parts), the agent tries to move a single part at each step. This interaction strategy greatly simplifies the joint inference model, which only needs to consider the case of one part being moved and linked to the rest of the object via one joint. If more than one part is moved, the model will treat all moving parts as one common part and predict one set of joint parameters, this error could be fixed with future interaction steps. Lastly, we assume that all movable parts are connected to the base link via a joint, with the base link always labeled 1 in the parts segmentation volume.

Joint inference. The joint inference module (modeled by a 3D CNN) is inspired by [17], and the joint parameter representation is inspired by [21]. The inferred joint parameters are represented as one volumetric output J with three components: 1) $J_{\text{type}} \in \mathbb{R}^{1\times 96\times 96\times 96}$ for joint type trained with BCE loss. 2) $J_{\text{axis}} \in \mathbb{R}^{3\times 96\times 96\times 96}$, gives per voxel predictions of the joint axis direction, trained with cosine similarity loss with groundturth value. 3) $J_{\text{pos}} \in \mathbb{R}^{1\times 96\times 96\times 96}$, gives per voxel predictions of the position of the revolute joint axis, which is represented using the distance between each voxel to its corresponding joint axis position, trained with MSE loss.

During training, we use the ground truth volumetric parts labels and only supervise on the output voxels of the moved part. From these predictions, we can compute the joint parameters by averaging the predictions over all voxels labeled as the moving part inferred by the part aggregation module. To track multiple joints over several steps, we maintain a dictionary where the key is the part label inferred by the part aggregation model and the value is a list of $\{J_{\text{type}},\ J_{\text{axis}},\ J_{\text{pos}}\}$. If a part is seen more than once, the inferred joint parameters will be appended to the existing list in the dictionary. The final joint parameters will be the median of all inferred values over several interaction steps.

D. Constructing an Articulated CAD Model

Given the part volume $\mathcal{H}_t$, the last step is to extract the 3D mesh for each part. Each entry in $\mathcal{H}_t$ encodes a probability distribution over parts. We observe that computing an $\texttt{argmax}$ over $\mathcal{H}_t$ can result in artifacts. To circumvent this problem, we directly deal with the continuous probability values to extract a smoother surface. First, we compute the inverted probability volume $\hat{\mathcal{H}_t} = 1 - \mathcal{H}_t$, where a value closer to 0 indicates higher probabilities of the surface. Treating $\hat{\mathcal{H}_t}$ as a distance volume, we can apply marching cubes to extract the zero-crossing surface. Since $\hat{\mathcal{H}_t}$ consists of continuous value, we can further upsample the volume (i.e., from 96^3 to 288^3) to improve the mesh quality. Finally, by combining the 3D part mesh with the estimated joint parameters (§III-C), we can generate a consolidated URDF file describing the articulated 3D CAD model as visualized in Fig. 1 d). For more details please refer to the supp. material.

IV. Experiments

We train a *single* perception and interaction model and evaluate it on 48 unseen instances from 10 categories and 77 instances from 7 unseen categories. When evaluating our method in simulation, an agent executes actions directly in our PyBullet [9] environment – following the definition of interactive perception. For the real-world proof of concept, we generate qualitative results for the perception component of our pipeline. Please refer to the supp. material for more details on the simulation environment and dataset.

Real-world Implementation: To demonstrate the feasibility of SfA in the real-world setting, we set up a single-arm tabletop environment, as shown in Fig 6. The robot arm is equipped with a cylindrical pusher, which moves the object parts based on the inferred actions. The environment has 4 Intel RealSense RGBD cameras, together capturing a RGB point cloud of the object. The interaction and 3D reconstruction results are shown in this video: https://sites.google.com/view/sfa-rebuttal-site/home. We believe the real-world results show that SfA is a promising step in future interactive perception and robotics research.

Metrics: We first evaluate the the effectiveness of the interaction policies independent from the perception model by measuring **optimal action ratio** = # optimal action/ # total action. Following Gadre, *et al.* [12]'s definition, action is optimal if it successfully moves a part that has not been discovered. If all parts are discovered, moving any part is considered optimal. For a single-step policy (i.e., UMP-Net [51]), the action is considered optimal as long as it triggers part movement.

The performance of object structure discovery is measured by following two aspects: 1) **Part segmentation and reconstructions.** Evaluated by part-wise 3D Intersection over Union (IoU) between predicted and ground truth part geometry. 2) **Joint Inference.** The accuracy of joint estimation is evaluated by 1) classification accuracy (between prismatic or revolute). 2) axis orientation error in degree. 3) axis position error in normalized scale (revolute joint only). All objects are scaled to fit in a $2 \times 2 \times 2$ cube in this dataset, and position error is evaluated with respect to this scale.

Baselines and Ablations: We test and compare with the following alternative interaction or perception module to study the efficacy of our system design:

- GT-Act (Oracle): to evaluate the perception module's performance upper bound, we test our perception module with the optimal interaction policy computed based on the ground truth state.
- UMP-Net [51], an interaction policy that aims to change the objects' joint state but not discover parts. This method might fail to produce an effective action when the interactable part is not observed in view.
- Ditto [17], a perception network that infers object's part segmentation and joint parameters from a single-step interaction. We combine Ditto with the other interaction policy to form a full pipeline.

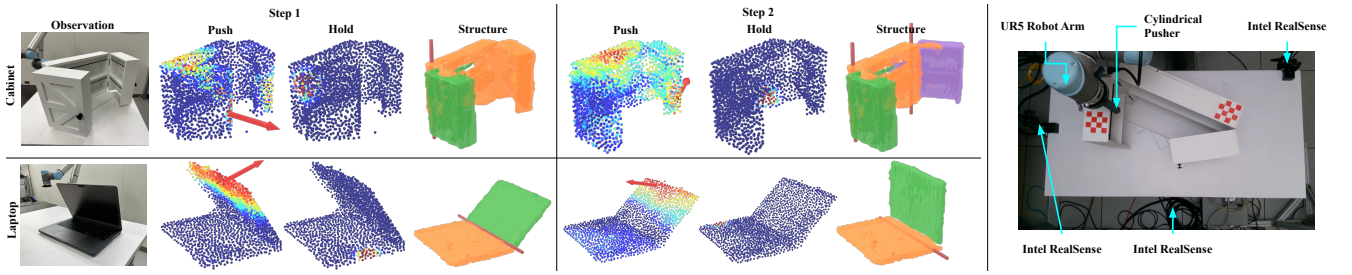

Fig. 6. **Realworld Result.** We evaluate the SfA pipeline on real-world point cloud constructed from multiple RGBD frames. The model performs well on previously unseen instances in the real world despite challenging noise artifacts from the real RGBD camera.

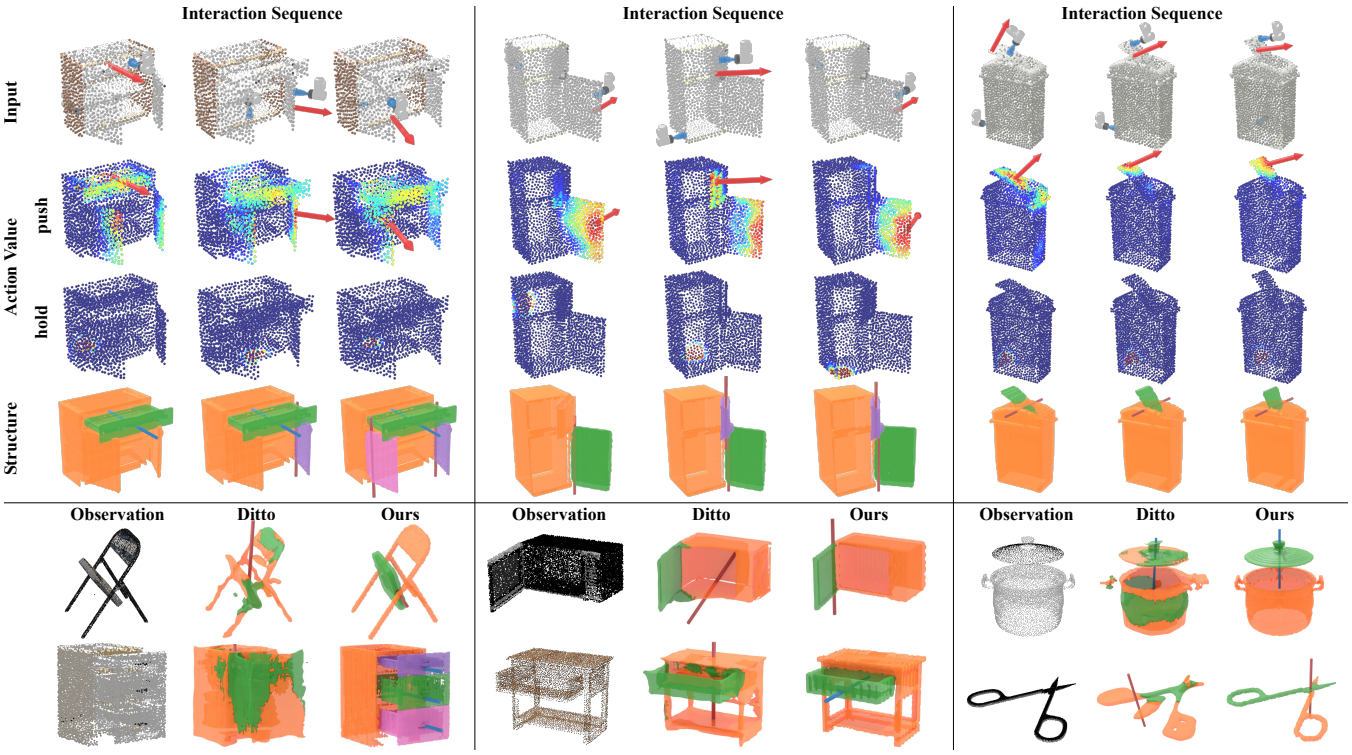

Fig. 7. **Qualitative Result in Simulation.** We show the step-by-step results from the SfA pipeline. The inferred actions prioritize new parts discovery and expose articulations. Our method outperforms the Ditto [17] on both parts reconstruction and joints estimation (revolute: red, prismatic: blue).

TABLE II

INTERACTION POLICY EVALUATION. THE INTERACTION POLICIES IS EVALUATED BY THE OPTIMAL ACTION RATIO = # OPTIMAL ACTIONS / # TOTAL ACTIONS. AN ACTION IS OPTIMAL IF IT SUCCESSFULLY MOVES AN UNDISCOVERED PART.

	Unseen Instances in Training Categories										Unseen Categories						
AtP [12]	0.0	25.0	0.0	50.0	20.0	25.0	0.0	20.0	0.0	0.0	20.0	0.0	20.0	20.0	20.0	20.0	0.0
UMP-Net [51]	0.0	50.0	10.0	0.0	18.2	28.5	0.0	0.0	70.0	83.3	6.2	16.6	**22.7**	26.3	5.5	60	5.3
SfA	**60.0**	**61.6**	**77.5**	**100**	**56.6**	**95.0**	**90.0**	66.6	86.0	73.3	**83.7**	**51.6**	19.1	**65.4**	**86.6**	**70.0**	**22.2**

TABLE III

PART SEGMENTATION AND RECONSTRUCTION RESULTS. PERFORMANCE IS EVALUATED BY PART-WISE 3D IOU BETWEEN PREDICTED AND GROUND TRUTH PART GEOMETRY. PERFORMANCE IS COMPARABLE ON BOTH UNSEEN INSTANCES AND UNSEEN CATEGORIES.

	Unseen Instances in Training Categories										Unseen Categories						
SfA-Percep + GT-Act*	79.5	79.6	92.5	94.6	91.2	94.7	82.3	87.3	73.4	80.4	71.5	87.9	92.2	86.4	92.9	83.8	78.7
Ditto[17]+UMP-Act[51]	30.9	37.0	43.8	40.3	52.1	36.6	40.8	44.7	43.7	42.5	52.2	37.3	30.7	41.7	52.1	39.3	30.0
Ditto[17]+SfA-Act	24.4	40.4	48.2	43.6	36.0	66.6	43.4	50.8	70.5	52.5	54.0	41.2	33.0	42.1	60.9	36.6	31.4
SfA-NoHistory	48.8	**75.9**	86.1	82.4	66.1	89.8	**68.4**	**86.7**	64.6	87.5	**95.6**	69.8	**49.6**	**62.7**	83.9	**72.1**	43.6
SfA	**71.5**	70.1	**93.1**	**87.0**	**68.9**	**92.2**	61.6	85.2	**75.0**	**95.7**	89.1	**78.8**	49.1	58.6	**85.3**	67.0	**49.3**

TABLE IV

JOINT EVALUATION. THE QUALITY OF JOINT ESTIMATION IS EVALUATED BY 1) CLASSIFICATION ACCURACY BETWEEN PRISMATIC AND REVOLUTE JOINT CATEGORIES (MACC), 2) AXIS ROTATION ERROR IN DEGREE, 3) AXIS POSITION ERROR IN NORMALIZED SCALE (REVOLUTE JOINT ONLY). DITTO IS EVALUATED WITH SFA INFERRED ACTIONS.

	Revolute joint															Prismatic joint				Type
	Unseen Instances in Training Categories									Unseen Categories						Unseen Ins.		Unseen Cat.		mAcc
	Rotation error (in degree) ↓																			↑
Heuristic	40.0	89.8	75.1	89.2	15.4	10.4	47.8	59.64	9.17	81.7	40.5	88.3	84.8	89.2	79.4	56.8	85.9	81.9	69.7	52.7
Ditto [17]	0.83	0.82	12.7	3.17	15.6	32.8	**0.36**	75.83	89.63	0.76	**3.02**	**1.20**	8.08	2.98	35.6	85.4	3.63	2.93	1.27	68.9
SfA	**0.39**	**0.79**	**11.43**	**1.02**	**5.42**	**8.61**	0.44	**2.11**	**3.72**	**0.49**	3.74	1.77	**7.52**	**1.94**	**35.3**	**1.49**	**0.27**	**2.82**	**3.34**	**86.7**
	Position error for revolute joint (in normalized scale) ↓																			
Heuristic	0.79	0.76	0.71	0.51	0.67	0.46	0.65	0.57	0.48	0.82	0.67	0.76	1.17	0.73	0.42					
Ditto [17]	0.22	0.61	0.19	0.37	**0.14**	0.23	0.25	0.32	0.44	0.34	0.39	**0.13**	0.77	0.46	1.05					
SfA	**0.06**	**0.12**	**0.03**	**0.26**	0.24	**0.07**	**0.07**	**0.01**	**0.05**	**0.04**	**0.05**	0.17	**0.41**	**0.13**	**0.41**					

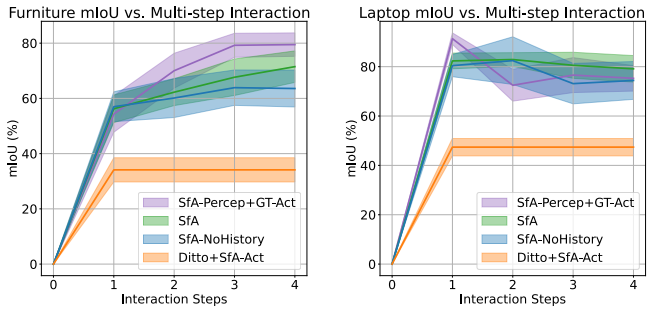

Fig. 8. **IoU w.r.t steps.** SfA can better discover parts with sequential interactions compared to single-step baseline [Ditto+SfA-Act][17]., especially on multi-part objects such as furniture. SfA can discover the full structure of two-part objects in one interaction step.

- Heuristic: Heuristic baseline for joint inference with ICP. Details can be found in Supp.
- AtP [12]: has both interaction and perception module, however, considers only 2D sequential action and 2D part segmentation.
- NoHistory: to evaluate the perception module's performance when multi-step parts aggregation is not used for interaction and perception.

A. Experimental Results

Comparison with baselines SfA goes beyond combining state-of-the-art components; Tab. 2 illustrates this empirically. SfA, on average, outperforms the combination of existing interaction and perception modules (Ditto+UMP-Act) by over 25 percentage points on the 3D reconstruction task with unseen objects. We conjecture the superior performance of SfA over the competing approach is due to BLAH BLAH, which the baseline is not able to properly exploit.

Generalization to unseen objects and categories. Our method makes no category-level assumptions, and allows it to generalize across categories. Tab. II, III, IV, show that SfA is able to achieve similar performance on unseen categories when compared to training categories, and outperforms alternative methods for the majority of the categories. Specifically, SfA interaction model beats the closest baseline by about 8 percent on the unseen categories. For objects with novel kinematics structures such as glasses, the pipeline performance is slightly worse than categories such as microwave, but still outperforms the best competing methods by 16% in the mIoU evaluation as seen in Tab III.

Are 3D actions necessary? Observing AtP's performance in Tab. II, we see that while 2D action space is sufficient for simple objects like scissors, it is not effective for complex objects with different joint types, and results in close to zero effective actions for many object categories. AtP's performance drops considerably when the object is not planar (e.g., kitchen pots). The structure of such objects cannot be observed from the top-down view. In contrast, our interaction policy is able to effectively infer informative 3D actions for a wide variety of objects. Furthermore, we also compare extensively against baselines that employ 3D continuous action spaces. Specifically, we compare to baselines that employ UMP-Net (see Tab.II and Tab.III). SfA outperforms the 3D action space baselines in nearly all categories for action inference and parts segmentation.

Does sequential interaction help? Based on the results in Fig. 8, we can observe that our method can not only discover new parts, but also segment parts better than Ditto [17], a single-step interaction baseline as well as our ablated SfA-NoHistory baseline. The improvement is more salient for objects with more than two parts (e.g., furniture and refrigerators). Comparing SfA and SfA-Percep + GT-Act in Tab. III, SfA outperformed the model with GT interactions. This result indicates that for those object categories, the inferred actions from SfA is effective in revealing all parts of the object, the discrepancies in IoU are due to inaccuracies in the perception module inference.

Does history aggregation help? By using informative interactions and aggregating visual observations in 3D, SfA could reveal and track the surfaces that are initially occluded and thereby better reconstruct the part geometry (e.g., the inside of a closed drawer). Comparing SfA and SfA-NoHistory in Tab. III, we can see that for most categories, action inference with parts labels and multi-step part aggregation outperforms the ablated baseline. The major advantage of

SfA over SfA-NoHistory is when objects have more than three parts, which helps contextualize the similar results for many objects that have two parts. In certain two-part object categories, the NoHistory baseline beats SfA This might be caused by the accumulation of perception errors in the multi-step part aggregation process.

Generalization to real-world observations. To validate the generalization of our approach to real-world data, we implement a capture pipeline that uses a 6DoF robot arm with a wrist-mounted camera to capture registered RGBD images of real-world articulated objects. For interactions, we allow a human to move parts. Fig. 6 demonstrates part and joint discovery and part tracking. This results validate the feasibility of our perception module (decoupled from the interaction module) is able to recover CAD models from real world RGB-D observations. See the video in the supplementary for examples.

Limitation and assumptions. Our pipeline assumes that only one joint is activated at each interaction step. While this assumption is mainly satisfied by our learned interaction policy (with both hold and push action), there are still cases violating this assumption. Additionally our algorithm does not estimate parameters like friction, which can be useful for robot manipulation.

V. Conclusion

We present SfA, a learning framework that discovers 3D parts geometry and joint parameters of novel articulated objects through a sequence of inferred interactions. Our results show that by coupling interactions and perception, the model can discover and reconstruct 3D articulated CAD models of objects from novel categories and with unknown kinematic structures. These results substantiate SfA's potential to enable robots to interact and reconstruct 3D articulated CAD models of unknown articulated objects autonomously.

References

[1] B. Abbatematteo, S. Tellex, and G. Konidaris, "Learning to generalize kinematic models to novel objects," *CoRL*, 2019.

[2] H. Abdul-Rashid, M. Freeman, B. Abbatematteo, G. D. Konidaris, and D. Ritchie, "Learning to infer kinematic hierarchies for novel object instances," *arXiv*, 2021.

[3] R. Bertolucci, A. Capitanelli, C. Dodaro, N. Leone, M. Maratea, F. Mastrogiovanni, and M. Vallati, "Manipulation of articulated objects using dual-arm robots via answer set programming," *Theory Pract. Log. Program.*, 2021.

[4] M. J. Black and A. D. Jepson, "Eigentracking: Robust matching and tracking of articulated objects using a view-based representation," *IJCV*, 1998.

[5] F. Burget, A. Hornung, and M. Bennewitz, "Whole-body motion planning for manipulation of articulated objects," *ICRA*, 2013.

[6] A. Capitanelli, M. Maratea, F. Mastrogiovanni, and M. Vallati, "Automated planning techniques for robot manipulation tasks involving articulated objects," *AI*IA*, 2017.

[7] ——, "On the manipulation of articulated objects in human-robot cooperation scenarios," *Robotics Auton. Syst.*, 2018.

[8] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, *et al.*, "Shapenet: An information-rich 3d model repository," *arXiv*, 2015.

[9] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation for games, robotics and machine learning," 2016.

[10] K. Desingh, S. Lu, A. Opipari, and O. C. Jenkins, "Efficient non-parametric belief propagation for pose estimation and manipulation of articulated objects," *Science Robotics*, 2019.

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

[12] S. Y. Gadre, K. Ehsani, and S. Song, "Act the part: Learning interaction strategies for articulated object part discovery," *ICCV*, 2021.

[13] J. Huang, H. Wang, T. Birdal, M. Sung, F. Arrigoni, S.-M. Hu, and L. Guibas, "Multibodysync: Multi-body segmentation and motion estimation via 3d scan synchronization," *arXiv*, 2021.

[14] W.-C. Hung, V. Jampani, S. Liu, P. Molchanov, M.-H. Yang, and J. Kautz, "Scops: Self-supervised co-part segmentation," *CVPR*, 2019.

[15] A. Jain, S. Giguere, R. Lioutikov, and S. Niekum, "Distributional depth-based estimation of object articulation models," *CoRL*, 2021.

[16] A. Jain, R. Lioutikov, C. Chuck, and S. Niekum, "Screwnet: Category-independent articulation model estimation from depth images using screw theory," in *ICRA*, 2021.

[17] Z. Jiang, C.-C. Hsu, and Y. Zhu, "Ditto: Building digital twins of articulated objects from interaction," *arXiv*, 2022.

[18] D. Katz, M. Kazemi, J. A. Bagnell, and A. Stentz, "Interactive segmentation, tracking, and kinematic modeling of unknown 3d articulated objects," *ICRA*, 2013.

[19] K. N. Kumar, I. Essa, and C. K. Liu, "Estimating mass distribution of articulated objects through non-prehensile manipulation," *arXiv*, 2019.

[20] T. E. Lee, J. Tremblay, T. To, J. Cheng, T. Mosier, O. Kroemer, D. Fox, and S. Birchfield, "Camera-to-robot pose estimation from a single image," *ICRA*, 2020.

[21] X. Li, H. Wang, L. Yi, L. Guibas, A. L. Abbott, and S. Song, "Category-level articulated object pose estimation," *CVPR*, 2020.

[22] L. Liu, W. Xu, H. Fu, S. Qian, Y.-J. Han, and C. Lu, "Akb-48: A real-world articulated object knowledge base," *arXiv*, 2022.

[23] Q. Liu, W. Qiu, W. Wang, G. D. Hager, and A. L. Yuille, "Nothing but geometric constraints: A model-free method for articulated object pose estimation," *arXiv*, 2020.

[24] X. Liu and K. M. Kitani, "V-mao: Generative modeling for multi-arm manipulation of articulated objects," *CoRL*, 2021.

[25] J. Lv, Q. Yu, L. Shao, W. Liu, W. Xu, and C. Lu, "Sagci-system: Towards sample-efficient, generalizable, compositional, and incremental robot learning," in *ICRA*, 2022.

[26] R. Martín Martín and O. Brock, "Online interactive perception of articulated objects with multi-level recursive estimation based on task-specific priors," *IROS*, 2014.

[27] R. Martín-Martín, C. Eppner, and O. Brock, "The rbo dataset of articulated objects and interactions," *IJRR*, 2019.

[28] M. K. Mittal, D. Hoeller, F. Farshidian, M. Hutter, and A. Garg, "Articulated object interaction in unknown scenes with whole-body mobile manipulation," *arXiv*, 2021.

[29] K. Mo, L. Guibas, M. Mukadam, A. Gupta, and S. Tulsiani, "Where2act: From pixels to actions for articulated 3d objects," *arXiv*, 2021.

[30] K. Mo, S. Zhu, A. X. Chang, L. Yi, S. Tripathi, L. J. Guibas, and H. Su, "PartNet: A large-scale benchmark for fine-grained and hierarchical part-level 3D object understanding," *CVPR*, 2019.

[31] J. Mu, W. Qiu, A. Kortylewski, A. L. Yuille, N. Vasconcelos, and X. Wang, "A-sdf: Learning disentangled signed distance functions for articulated shape representation," *ICCV*, 2021.

[32] T. Mu, Z. Ling, F. Xiang, D. Yang, X. Li, S. Tao, Z. Huang, Z. Jia, and H. Su, "Maniskill: Generalizable manipulation skill benchmark with large-scale demonstrations," *arXiv*, 2021.

[33] A. Noguchi, U. Iqbal, J. Tremblay, T. Harada, and O. Gallo, "Watch it move: Unsupervised discovery of 3d joints for re-posing of articulated objects," *arXiv*, 2021.

[34] A. Noguchi, X. Sun, S. Lin, and T. Harada, "Neural articulated radiance field," *ICCV*, 2021.

[35] S. Pillai, M. Walter, and S. Teller, "Learning articulated motions from visual demonstration," *RSS*, 2014.

[36] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," *CVPR*, 2017.

[37] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," *NeurIPS*, 2017.

[38] S. Qian, L. Jin, C. Rockwell, S. Chen, and D. F. Fouhey, "Understanding 3d object articulation in internet videos," *arXiv*, 2022.

[39] T. Schmidt, R. A. Newcombe, and D. Fox, "Dart: Dense articulated real-time tracking," *RSS*, 2014.

[40] H. Shen, W. Wan, and H. Wang, "Learning category-level generalizable object manipulation policy via generative adversarial self-imitation learning from demonstrations," *arXiv*, 2022.

[41] A. Siarohin, O. J. Woodford, J. Ren, M. Chai, and S. Tulyakov, "Motion representations for articulated animation," *CVPR*, 2021.

[42] J. Sturm, A. Jain, C. Stachniss, C. C. Kemp, and W. Burgard, "Operating articulated objects based on experience," *IROS*, 2010.

[43] J. Sturm, C. Stachniss, and W. Burgard, "A probabilistic framework for learning kinematic models of articulated objects," *JAIR*, 2011.

[44] S. Tsogkas, I. Kokkinos, G. Papandreou, and A. Vedaldi, "Semantic part segmentation with deep learning," *arXiv*, 2015.

[45] J. Wang and A. Yuille, "Semantic part segmentation using compositional model combining shape and appearance," *CVPR*, 2015.

[46] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, "Dynamic graph cnn for learning on point clouds," *TOG*, 2019.

[47] 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," *ICCV*, 2021.

[48] R. Wu, Y. Zhao, K. Mo, Z. Guo, Y. Wang, T. Wu, Q. Fan, X. Chen, L. J. Guibas, and H. Dong, "Vat-mart: Learning visual action trajectory proposals for manipulating 3d articulated objects," *arXiv*, 2021.

[49] 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," *CVPR*, 2020.

[50] Z. Xu, Z. Liu, C. Sun, K. Murphy, W. Freeman, J. Tenenbaum, and J. Wu, "Unsupervised discovery of parts, structure, and dynamics," *ICLR*, 2019.

[51] Z. Xu, H. Zhanpeng, and S. Song, "Umpnet: Universal manipulation policy network for articulated objects," *RA-L*, 2022.

[52] J. Yan and M. Pollefeys, "A general framework for motion segmentation: Independent, articulated, rigid, non-rigid, degenerate and non-degenerate," *ECCV*, 2006.

[53] G. Yang, D. Sun, V. Jampani, D. Vlasic, F. Cole, C. Liu, and D. Ramanan, "Viser: Video-specific surface embeddings for articulated 3d shape reconstruction," *NeurIPS*, 2021.

[54] L. Yi, H. Huang, D. Liu, E. Kalogerakis, H. Su, and L. Guibas, "Deep part induction from articulated object pairs," *TOG*, 2019.

[55] V. Zeng, T. E. Lee, J. Liang, and O. Kroemer, "Visual identification of articulated object parts," in *IROS*, 2021.

[56] H. Zhao, L. Jiang, J. Jia, P. H. S. Torr, and V. Koltun, "Point transformer," *CoRR*, 2020.