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

Neil Nie\(^1\) Samir Yitzhak\(^1\) Kiana Ehsani\(^2\) Shuran Song\(^1\)
\(^1\) Columbia University \(^2\) Allen Institute for AI
https://sfa.cs.columbia.edu

Abstract: Articulated objects make up a significant portion of our environment. Discovering their parts, joints, and kinematics is crucial for robots to interact with these objects. We introduce Structure from Action (SfA), a framework that discovers the 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 in the case of categories not seen during training. By selecting informative interactions, SfA discovers parts and reveals initially 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 single SfA model trained in simulation can generalize to many unseen object categories with unknown kinematic structures and to real-world objects. Code and data will be publicly available.

Keywords: Interactive Perception, Articulated Objects, 3D Perception

1 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, 2, 3], especially in household use-cases. For tasks involving novel articulated objects, recovering 3D articulated CAD models (e.g., URDFs) is a promising starting point. These models are immediately useful in task-specific planning pipelines [4, 5, 6, 7, 8]. For instance, in a kitchen, recovering models of drawers can enable downstream planning to retrieve objects within them. To discover the structure of objects beyond training categories, there is mounting evidence that interaction is critical [3, 9]. Informative interactions allow 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 via interaction is a complex task that involves tackling a diverse set of challenges, such as:

- **Inferring informative sequential 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, which gives 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.** Articulated object modeling should work in diverse household environments, even with unseen object categories with novel kinematic structures.

These challenges have dictated a variety of simplifying assumptions in prior works. For example, Gadre et al. [3] relaxes the problem to a 2D setting, not attempting to reconstruct the full 3D structure. Jiang et al. [10] consider only 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 to tackle 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 thereby aggregating visual observations in a spatially consistent manner, 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 [11, 12, 13] 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.
- 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.

### 2 Related Work

Recently, interactive perception with articulated objects has gained renewed interest. As an overview, we contrast our method with recent work in Tab. 1.

**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 [11, 12, 14, 13, 15, 16]. There is a line of work tackling the problem of interacting with articulated objects to move their parts [2, 17, 8, 18]. Some work [19, 20] 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 [18], etc.). Our goal is to learn to interact with objects with the goal of discovering joints, parts, and dynamics of the articulation. Eisner et al. [21] propose a vision-based method to predict the flow and articulated motions of an object. However, they do not infer knowledge of parts or joints. Xu et al. [9] propose a single image-based policy network to recover joint axes, but they do not attempt to recover parts.

**Perception from passive observation.** Prior works have used a variety of methods to recover the articulation of objects, such as using dense pose fitting [22], adapting neural radiance field [23], inferring kinematic graphs [24], using depth point cloud [25], and semantic segmentation [26]. Mu et al. [27] propose a model to generate shapes of articulated objects at unseen angles. These works 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) [28, 29, 30, 31, 1, 32, 33, 34, 35, 36].
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 [37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 10, 48]. 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 [49, 50, 51]. In contrast, we use a generalizable approach to predict the actions that works on unseen categories. Similarly, more modern approaches focus on the perception, using scripted robot actions [10]. This method also only applies for one stage and fails to identify more than one part. Kumar et al. [52] recover the mass distribution of the articulated objects using interaction, but they do not recover joints or parts. Recently, Gadre et al. [3] 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, perception and history, we are able to consider both revolute and prismatic joints and relax restrictions on camera positioning.

## 3 Structure From Action

We introduce Structure from Action (SfA), a learning framework for 3D active perception that discovers object parts and joints, and is capable of generalizing to unseen object categories. Given a raw RGB point cloud, SfA infers and executes informative actions to construct an articulated CAD model, which consists of *multiple* 3D part meshes and the revolute, and prismatic joints connecting them. We start by formally describing our task setting (§ 3.1). We then detail the SfA framework, consisting of four components: an interaction policy (§ 3.2), which chooses *informative* actions that move parts, a part aggregation module (§ 3.3), which tracks part discoveries over a sequence of interactions, a joint estimation module (§ 3.4), which predicts joint parameters and kinematic constraints of the articulation, and finally, the pipeline for the construction of the articulated CAD model (§ 3.5). Our method is summarized in Fig. 2.

### 3.1 Articulated Object Structure Discovery

The task is to capture the articulation of an object consisting of multiple parts connected by prismatic and revolute joints. We do not assume constraints on the semantic categories of articulated objects. Furthermore, the joints and the number of parts are *not* known a priori. Given an initial input point cloud \( P_0 \) capturing the object the task is to reconstruct and segment all parts as 3D meshes, discover all prismatic and revolute joints, and estimate their parameters.

We give the agent a budget of \( T = 5 \) interactions to expose parts and joints as not all articulated structures are necessarily pronounced in \( P_0 \). Inspired by Gadre et al., we assume a bimanual embodied agent, which simultaneously holds a part of the articulated object and pushes some other part at every timestep. This kind of action allows the agent to isolate a single part of the object and is particularly useful for small objects without a fixed base. For example, with scissors an agent might fix one blade and articulate the other. In contrast to Gadre et al. we consider a full 3D action space. Concretely, the agent holds the object
at a 3D location and pushes it at a 3D location in a 3D direction. After the interaction, the perception pipeline needs to discover, track, and update the 3D parts geometry of the object, as well as estimate the parameters of the joint that connects the moved part to the rest of the object.

**Notation.** We require ways to represent both the input observation and current 3D part segmentation of objects at every timestep $t$. For input observations, we consider point-clouds $P_t \in \mathbb{R}^{2048 \times 9}$, which are constructed by sampling points from five differently-posed RGB-D images. We also consider a voxelized version of the point-cloud $V_t \in \mathbb{R}^{7 \times 96 \times 96 \times 96}$. Each voxel contains contains an occupancy channel (1 vs. 0), three channels for color, and three channels for the surface normal. For part segmentation, we consider another volume $H_t \in \mathbb{R}^{6 \times 96 \times 96 \times 96}$. Each voxel in $H_t$ contains an $N$-dim probability vector, which is a distribution over parts. We consider $N = 6$ as the max number of parts. Critically, $H_t$ is aligned to $V_t$ and hence describes the current part segmentation of the observation.

### 3.2 Learning to Interact with Articulated Parts

Inferring interaction over several timesteps allows us to create 3D representations of unseen articulated objects with many parts and joints. Take a door-like structure on a kitchen console, for instance; without interaction, it is hard to determine if the door swings like a cabinet or slides like a drawer. Furthermore, by moving the door, occluded surfaces behind the door are revealed, allowing for more accurate 3D reconstruction. However, interaction without an adequate strategy is not always useful. For example, for less bulky objects like scissors that do not have a fixed base, stabilizing some parts is necessary to prevent the whole object from sliding rigidly. Moreover, given a multi-part object, interacting with only one part repeatedly, may not expose all parts. These examples elucidate a set of requirements for an interaction module (Fig. 3). It should (1) trigger motion, (2) isolate part motion, and (3) prioritize interacting with new parts. We discuss our algorithm designed to fulfill these requirements.

**Model.** Our interaction module consists of two sub-networks to determine hold and push actions. Both networks are modeled by point transformers [53], and take the observation $P_t$. Because we want interactions to be aware of previous part discoveries, we also input part labels stored in $H_t$ for each point. Unlike Gadre et al. [3] that is limited to 2D, we consider a 3D observation space and 3D actions. The hold sub-network infers a hold affordance value for each point, representing the probability that the point is a good hold candidate. The system chooses and executes a hold action by sampling uniformly among the hold locations with the 50 highest affordance values. We want the push sub-network to condition it prediction on the hold action. Therefore, we can generate dense action supervision by computing the scene flow in simulation. For the push network, we supervise the model with forward 3D scene flow, which gives regression targets for push directions and magnitudes, optimized via MSE loss. For the hold network, points without flow should have an affordance values of one (i.e., good holding locations), while other points should have a target value of zero. We use binary cross-entropy loss for training. For more details, please see the supp. material.
3.3 Learning Persistent Part Aggregation for Dynamic Part Reconstruction

To construct the full 3D structure of an articulated object, it is critical to keep track of parts when they are discovered and as they move over multiple timesteps. This is a non-trivial task for many reasons. Take for example the agent opens a previously undiscovered drawer. The model must recognize that a new part of the object has been discovered, and complete the part geometry as more surfaces are observed. At the next timestep if the agent closes the drawer, the model is tracking an already discovered part as it is being moved into occlusion. The model must recognize that the drawer has moved, which is particularly difficult when the motion is small. Furthermore, it is necessary to establish reliable correspondences between the drawer before and after the movement. Point-to-point correspondences are not sufficient as large portions of the drawer may disappear as it closes. Successful strategy should preserve the previously seen geometry even when the drawer is shut and becomes occluded (Fig. 4). To tackle these demonstrative challenge, we propose a learning-based part aggregation module.

Model. Our part aggregation model is auto-regressive. It takes the last part volume $H_{t-1}$ along with voxelized representation of the previous and current observations $V_{t-1}, V_t$. The model is a 3D encoder-decoder U-Net, which outputs part labels at the same voxel resolution as the input. The output represents an updated part volume $H_t$. Similar to image segmentation representations in vision, each voxel in $H_t$ is a distribution over all possible part segmentation labels.

Supervision. We supervise this module with the groundtruth volume computed from simulation $H_{gt}$. At each timestep $t$, the groundtruth target only includes the voxels that have been moved by the agent and observed by the camera, in any one of the previous steps. If a voxel has never been on a static part or occluded surface, it will be ignored. The correspondence between predicted and groundtruth part is found using Hungarian matching.

3.4 Joint Inference

Learning the dynamics and kinematics of the joints of the object is critical for uncovering how parts move. Hence, we define a joint inference module to predict the joint parameters of the object. Our module must (1) predict the type of the joint (i.e., prismatic vs. revolute), (2) infer the joint axis, and (3) infer the joint position in the case of revolute joints. For example joint inference see Fig. 5.

Model and supervision. Given the voxelized input of previous $V_{t-1}$ and current $V_t$ observations, we employ a 3D U-Net encoder-decoder to predict joint parameters for the most recently moved part. Hence, we mask the model output using the part segmentation for the most recently moved part—as predicted by the part aggregation module (Sec. 3.3). More specifically, we decode the model output to an eight-channel output voxel grid. The first channel specifies whether the voxel is connected to another part via a revolute or prismatic joint. The following three channels give per voxel votes of the joint axis direction. For revolute joints, the position of the joint axis matters. The last four channels of the model output provide the revolute joint axis position. Of the last four channels, the first three provide a per voxel vector pointing in the direction of the joint axis. The last channel provides the per voxel closest distance to the revolute joint. We supervise this model with groundtruth joint parameters corresponding to the part that is moved between timestep $t-1$ and $t$. The joint type is trained with BCE loss, joint axis using cosine loss and distance to the revolute joint using MSE loss. Since the joint parameters are independently inferred at each interaction step, we need to combine all estimations into one coherent output. The details on how we consolidate multiple estimations for the same joint and associate discovered joints with parts can be found in the supp. material.
Figure 6: **Realworld Result.** We evaluate the SfA pipeline on realworld 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.

Figure 7: **Qualitative Result in Simulation.** We show the step-by-step results from the SfA pipeline (top). The inferred actions prioritize new parts discovery and exposes articulations. Our method outperforms the Ditto [10] on both parts reconstruction and joints estimation (revolute: red, prismatic: blue).

### 3.5 Constructing a Complete Articulated CAD Model

Given the part volume $H_t$, the last step is to extract 3D mesh for each part. Each channel of $H_t$ encodes the occupancy of one identified part, where the value stored at each voxel is predicted occupancy probability. We empirically observe that computing an argmax over $H_t$ can result in artifacts. To circumvent this problem, we deal with the continuous probability values directly to extract a smoother surface. First, we compute the inverted probability volume $\hat{H}_t = 1 - H_t$, where a value closer to 0 indicates higher probabilities of surface. Treating $\hat{H}_t$ as a distance volume, we can apply marching cubes to extract the zero-crossing surface. Furthermore, since $\hat{H}_t$ consists of continuous value, we can further upsample the volume (i.e., from $96^3$ to $288^3$) before running marching cube to improve the mesh quality, without making training more expensive. Finally, by combining the 3D part mesh with the estimated joint parameters (§3.4), we can generate a combined URDF file describing the articulated 3D CAD model as visualized in Fig. 1 d).

### 4 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. Each instance’s start position, orientation, joint configuration, and scale are randomly generated.

**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.’s
The performance of object structure discovery is measured from the following two aspects: 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 × 2 × 2 cube in this dataset, and position error is evaluated with respect to this scale.

The accuracy of joint estimation is evaluated by optimal action ratio. The accuracy of joint estimation is evaluated by optimal action ratio. Evaluated by part-wise 3D Intersection over Union (IoU) between predicted and groundtruth 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 × 2 × 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 provide the perception module’s performance upper bound, we evaluate our perception module with an interaction policy that takes optimal action in every step computed based on the ground truth state.
- **UMP-Net [9],** a single-view based interaction policy that targets at changing the objects’ joint state, but not discovering parts. This method might fail to produce an effective action when the interactable part is not observed in view.
- **Ditto [10],** 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.
- **Heuristic:** Heuristic baseline for joint inference with ICP. Details can be found in Supp.
- **AtP [3]:** has both interaction and perception module, however, considers only 2D sequential action and 2D part segmentation.
- **NoHistory, our method without history aggregation for part reconstruction.**
4.1 Experimental Results

**Does sequential interaction help with discovering parts?** Based on the results in Fig. 8, we can observe that our method can discover and segment parts better than Ditto [10], 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).

**Does history aggregation help in part reconstruction?** 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). SfA’s performance in parts reconstruction is supported by the results in Tab. 3 compared with the single-step baseline Ditto [10].

**Generalization to unseen objects and categories.** Our method makes no category-level assumptions, allows it to generalize across categories. Tab. 2, 3, 4, and Fig. 7, show that SfA is able to achieve similar performance for unseen categories compared to the training ones, and outperforms alternative methods of for majority of the 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.

**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 impressive part and joint discovery and part tracking, thereby validating the sim2real adaption of the perception module. See the video in the supplementary for examples.

**Are 3D actions necessary?** Observing AtP’s performance in Tab. 2, we can see that while 2D action space can be sufficient for simple objects like scissors, it is not effective for complex objects with different joint types, and results in close to zero effective action 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.

**Does history inform actions?** Comparing SfA and SfA-NoHistory in Tab. 2, we can see whether the history information could inform the interaction policy. The result shows that, while using history does not influence the action’s efficiency for many two-part objects, it improves the optimal action ratio for the objects with more than two parts (e.g., furniture) as well as all unseen object categories.

4.2 Limitation and assumptions

Our pipeline has made the assumption 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. In those cases, the algorithm needs to consider full object kinematic structure for joint estimation. In addition, limited by the sensory input (visual inputs only) and simulation fidelity, our algorithm does not estimate the physical parameters of the joints and parts (e.g., friction), which can be useful for applications like robot manipulation.

5 Conclusion

We present Structure from Action (SfA), a learning framework that discovers 3D parts geometry and joint parameters of unseen 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 (e.g., URDFs) of objects from unseen categories and with unknown kinematic structures. These results and findings substantiate SfA’s potential to enable robots to interact with and to reconstruct 3D articulated CAD models of unknown articulated objects autonomously.
Acknowledgments

This work was supported in part by National Science Foundation under 2143601, 2037101, and 2132519. We would like to thank Google for the UR5 robot hardware. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the sponsors.

References

[1] B. Abbatematteo, S. Tellex, and G. Konidaris. Learning to generalize kinematic models to novel objects. CoRL, 2019.
[2] K. Mo, L. Guibas, M. Mukadam, A. Gupta, and S. Tulsiani. Where2act: From pixels to actions for articulated 3d objects. arXiv, 2021.
[3] S. Y. Cadre, K. Ehsani, and S. Song. Act the part: Learning interaction strategies for articulated object part discovery. ICCV, 2021.
[4] J. Sturm, A. Jain, C. Stachniss, C. C. Kemp, and W. Burgard. Operating articulated objects based on experience. IROS, 2010.
[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] A. Capitanelli, M. Maratea, F. Mastrogiovanni, and M. Vallati. On the manipulation of articulated objects in human-robot cooperation scenarios. Robotics Auton. Syst., 2018.
[8] 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.
[9] Z. Xu, H. Zhanpeng, and S. Song. Umpnet: Universal manipulation policy network for articulated objects. IEEE Robotics and Automation Letters, 2022.
[10] Z. Jiang, C.-C. Hsu, and Y. Zhu. Ditto: Building digital twins of articulated objects from interaction. arXiv, 2022.
[11] 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.
[12] 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.
[13] 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.
[14] R. Martín-Martín, C. Eppner, and O. Brock. The rbo dataset of articulated objects and interactions. URR, 2019.
[15] 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.
[16] 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.
[17] 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.
[18] H. Shen, W. Wan, and H. Wang. Learning category-level generalizable object manipulation policy via generative adversarial self-imitation learning from demonstrations. *arXiv*, 2022.

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

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

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

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

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

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

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

[26] 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.

[27] 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.

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

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

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

[31] 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.

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

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

[34] 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.

[35] 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.

[36] 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.

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

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

We use the PyBullet physics simulator for generating the training data and evaluating the SfA pipeline. Five depth cameras with known positions and camera parameters are placed around the object: one at the top with a birds-eye-view, and one on each of the four sides of the object. The observation is generated by fusing the back-projected depth maps from each of the cameras. When an object is loaded, the object will be resized to fit in a $2^3$ cube. The scaling factor, initial position, and orientation are all randomly sampled from a normal distribution during training and testing. The state of each joint is randomized each time the object is loaded into the simulator.
B  Ground Truth Generation

**Action Supervision from Motion.** In Sec. 3.2, we describe using the observed 3D scene flow as the supervision signal for the interaction module. To generate the supervision, we load the training objects from the PartNet-Mobility dataset [13] in the PyBullet physics simulator. We allow the ground truth generation algorithm to interact with each training object \( N = 5 \) times, since all objects in the dataset have five parts or fewer. At each timestep, we use the joint parameters specified in the object URDF file to randomly generate a new target joint state for a single non-fixed joint in the object. Then, we change that joint state to the new target state. By changing the joint state, we have effectively moved one part of the object. We capture observations before and after the part movement and use the ground truth correspondence provided by the simulation environment to generate the forward 3D scene flow, only during training. Within \( N \) steps, the algorithm will move each part at least once by altering the joint states, exposing the full kinematic structure of the object. Given new URDF files of other object categories and objects with novel kinematic structures, this method of ground truth generation is easily cheap to execute and scaleable.

**Persistent Part Aggregation.** We use the same method described above to generate the ground truth parts labels used to train the persistent part aggregation module described in Sec. 3.3. At the initial timestep, before a joint state has changed, the part label 1 is given to all points on the object, as no part has been discovered yet. Once a joint state has changed, a new part has been discovered. This newly discovered part will be assigned a new part label 2, while the rest of the object will keep the part label 1. In the following steps, all undiscovered parts will be moved at least once, and they will each receive a unique integer part label. In the case of movement of a discovered part, the part and the rest of the object will maintain their parts labels.

Initially, the unobserved occluded surfaces in the object is not included in the ground truth parts segmentation. As more surfaces and parts of the objects are observed by the five-camera array, the new observations are aggregated with the most up-to-date parts segmentation, building up a more and more complete representation of the object parts geometry. If an already seen part of the object becomes occluded after some interaction, the parts segmentation will maintain the part integrity, thus encouraging the parts aggregation model to learn parts permanence.

C  Network Architecture

**Action Network.** The action network is trained using the SGD optimizer with an initial learning rate of 0.5 over 50 epochs, and a multi-step learning rate scheduler. All hyperparameters are selected based on the original Point Transformer [53] implementation. Both the hold prediction and push prediction networks have 4 blocks, 256 transformer dimensions, progressive downsampling by 4x, and consider 16 points for nearest neighbors. We use farthest point sampling to downsample the observation point cloud to 2048 points.

**Part Aggregation Network.** The parts aggregation network is trained using the Adam optimizer with 2e-4 learning rate, batch size 9, over 20 epochs. The network follows the 3D UNet [54] architecture with residual connections similar to those proposed by He et al. [55]. The input of the network is: the voxelized observation at time \( t \) before the interaction, the voxelized parts segmentation at time \( t \), and the voxelized observation after the interaction at \( t + 1 \). The output of the network is a voxel volume with each voxel representing the parts label using 1-hot encoding. The network is comprised of an encoder and decoder sub-network. The encoder network has 8 residual blocks, each block consists of 2 3D convolutional layers with kernel size \( 3 \times 3 \times 3 \), batch normal layers, and leaky ReLU activation layers. The decoder network consists of 4 ResNet 3D upsampling blocks. Within each block, there is one trilinear upsampling layer followed by a convolutional layer and 2 ResNet blocks. After the last ResNet 3D upsampling block, we apply another 3D convolution layer with \( 1 \times 1 \times 1 \) kernel size and channel dimension 8, which is the final output channel dimension. There are skip connections between the decoder and encoder network, based on the original UNet [54] architecture.

**Joints Network.** The joint network is trained using the Adam optimizer with 5e-4 learning rate over 20 epochs. The network architecture is nearly identical to the parts aggregation network. The two major
Table 5: Training category icon names.

| Category Name | Storage Furniture | Scissor Folding Chair | Kitchen Pot | Refrigerator | Safe | Knife | Trashcan | Laptop | Box |
|---------------|-------------------|-----------------------|------------|-------------|------|-------|----------|--------|-----|

Table 6: Testing category icon names.

| Category Name | Dishwasher | Kettle | Eyeglasses | Lighter | Microwave | Plier | Table |

Differences are: 1) joints network only considers the before and after observation, without the previous history, 2) the joints network has 8 channels that represent the joint parameters. The joint network and the part aggregation network are trained separately.

D Details on Joint Inference Algorithm

In Sec. 3.4, we present the joint inference model which predicts the joint parameters of one single joint that connects the most recently moved part to the rest of the object. During multiple interaction steps, we aggregate the joint parameters of many different joints using the following method. The key idea is to build a dictionary containing part’s segmentation label to joint parameter pairs. At each interaction step, assuming one part has been moved, the part aggregation model returns the moved part segmentation label \( \text{id}_{moved} \). The joints model returns the joint parameters, \( J_{\text{type}}, J_{\text{axis}}, J_{\text{position}} \) denoting the joint type, axis, and position, associated with \( \text{id}_{moved} \). Given this information, we add the key-value pair \( \text{id}_{moved}: \{ J_{\text{type}}, J_{\text{axis}}, J_{\text{position}} \} \) to our dictionary. After all interaction steps, we use all key-value pairs in the dictionary to generate the 3D articulated CAD model (i.e. the URDF file).

E Heuristics Joint Estimation Baseline

To validate our learned joint estimation module, we developed a heuristic-based joint parameter estimation algorithm using the parts segmentation masks predicted by the parts aggregation model. The key idea is to obtain two point clouds of only the moved part before and after the interaction. By using ICP, we can estimate a SE(3) transform of the part movement. By using some heuristics, we can determine based on the SE(3) transform, the joint type, joint axis, and joint position. To compute the two point clouds of the move part, we exploit the segmentation mask of the moved part before and after the interaction.

F Dataset

We use the articulated objects provided by the PartNet-Mobility [13] dataset. There are 48 unseen instances from 10 training categories listed in Tab 1 and 77 instances from 7 testing categories listed in Tab 2. We sample the storage furniture category more than other categories to achieve a good balance of revolute joints and prismatic joints and also objects with more than 2 parts. Additionally, we excluded any instances from the original PartNet-Mobility data with missing meshes or unstable simulation results.
Algorithm 1: Heuristic Based Joint Estimation Algorithm

Data: \( V_{t-1}, V_t, H_{t-1} \)

Result: \( J_{\text{type}}, J_{\text{axis}}, J_{\text{position}} \)

1. \( H_t \leftarrow \text{part\_model.\_forward}(H_{t-1}, V_{t-1}, V_t) \)
2. \( H'_t \leftarrow \text{part\_model.\_forward}(H_t, V_t, V_{t-1}) \)
3. \( \text{id}_{\text{moved}} \leftarrow \max_{\text{occurrence}}(\text{abs}(H'_t - H_t)) \)
4. \( P_t \leftarrow \text{where } H_t \text{ is id}_{\text{moved}} \)
5. \( P_{t-1} \leftarrow \text{where } H_{t-1} \text{ is id}_{\text{moved}} \)
6. \( T \leftarrow \text{ICP}(P_t, P_{t-1}); \quad /* \ T \text{ is a SE(3) transform } */ \)
7. \( \theta, \vec{a} \leftarrow \text{axis\_rotation}(T) \)
8. if \( \theta > k\text{Threshold} \) then
9. \( J_{\text{type}} \leftarrow \text{revolute} \)
10. else
11. \( J_{\text{type}} \leftarrow \text{prismatic} \)
12. \( J_{\text{axis}} \leftarrow \vec{a} \)
13. \( J_{\text{position}} \leftarrow \text{center\_of\_part}(\text{id}_{\text{moved}}) \)
14. end
15. if \( J_{\text{type}} \text{ is revolute} \) then
16. \( \bar{P} \leftarrow \text{find\_overlap\_points}(P_t, P_{t-1}) \)
17. \( J_{\text{axis}} \leftarrow \text{fit\_line}(\bar{P}) \)
18. \( J_{\text{position}} \leftarrow \text{center}(\bar{P}) \)
19. end