Articulate AnyMesh: Open-Vocabulary 3D Articulated Objects Modeling

Xiaowen Qiu* UMass Amherst
Jincheng Yang* Shanghai Jiao Tong University
Yian Wang UMass Amherst
Zhehuan Chen UMass Amherst

Yufei Wang CMU
Tsun-Hsuan Wang MIT
Zhou Xian CMU
Chuang Gan UMass Amherst, MIT-IBM Watson AI Lab

Abstract: 3D *articulated* objects modeling has long been a challenging problem, since it requires to capture both accurate surface geometries and semantically meaningful and spatially precise structures, parts, and joints. Existing methods heavily depend on training data from a limited set of handcrafted articulated object categories (*e.g.*, cabinets and drawers), which restricts their ability to model a wide range of articulated objects in an open-vocabulary context. To address these limitations, we propose ARTICULATE ANYMESH, an automated framework that is able to convert any rigid 3D mesh into its articulated counterpart in an open-vocabulary manner. Given a 3D mesh, our framework utilizes advanced Vision-Language Models and visual prompting techniques to extract semantic information, allowing for both the segmentation of object parts and the construction of functional joints. Our experiments show that ARTICULATE ANYMESH can generate large-scale, high-quality 3D articulated objects, including tools, toys, mechanical devices, and vehicles, significantly expanding the coverage of existing 3D articulated object datasets. Additionally, we show that these generated assets can facilitate the acquisition of new articulated object manipulation skills in simulation, which can then be transferred to a real robotic system. Our Github website is https://articulateanymesh.github.io/.

1 Introduction

Figure 1: **ARTICULATE ANYMESH** turns 3D meshes (*e.g.* retrieved from Objaverse [1]) into articulated objects. Our pipeline is capable of processing a wide range of objects, including daily necessities, furniture, vehicles and even fictional objects.

The gathering of data on a large scale is an emerging research trend in embodied AI and robotics. Large foundation models built for robotics and embodied AI are extremely data-hungry, intensifying

9th Conference on Robot Learning (CoRL 2025), Seoul, Korea.

the need for large-scale data collection. Compared to collecting data in the real world, gathering data in simulations is significantly faster and more convenient, making it easier to capture various types of ground truth information such as physical states, segmentation masks, depth maps, etc. Existing works have explored many different aspects of how to carry out large-scale data collection in simulations, ranging from asset generation [2], scene generation [3, 4], task design [5], demonstration collection [6, 7], reward design [8], etc.

Despite these efforts, a key challenge remains: collecting diverse and realistic articulated objects essential for everyday life, which is vital for producing diverse data that can be generalized to real-world applications. One intuitive approach to achieve this is to use generative models. While there has been substantial progress in 3D asset generation, few available methods are capable of addressing the demand for the collection of articulated objects. Most 3D generation approaches, utilizing a forward pass of a trained network [9, 10, 11, 12, 13], or SDS loss optimization [14, 15, 16], produce only the surface of the object. These objects can only be manipulated as a whole body. For instance, a closet produced through these 3D generation methods cannot be opened or used to store clothes. Part-aware 3D generation approaches [17, 18, 19, 20, 21, 22, 23] generate 3D objects together with their part-specific details. Although these methods are more sensitive to structure, the objects produced are still restricted to whole-object manipulation for the lack of motion parameters. Articulated object creation approaches [24, 25, 2, 26, 27, 28, 29, 30] are capable of producing articulated objects with several interactive parts, demonstrating functionality. However, such methods require dense observation of a to be reconstructed articulated object in multiple joint states [24, 25, 28, 31], or are restricted to the limited-scale data and object categories used to train their network [2, 26, 27]. As a result, current methods struggle to automatically generate a wide variety of articulated objects, particularly those from underrepresented or absent categories in existing datasets [32, 33, 34].

Consequently, the collection of diverse articulated objects presents extra challenges compared to non-articulated 3D assets: (1) accurate semantic structures must accompany geometry and appearance, (2) articulation parameters such as joint orientation and position are required, and (3) the relatively small-scale articulated object datasets, compared to 3D object datasets, are insufficient to support the training of a generalizable model. To overcome these challenges, we introduce ARTICULATE ANYMESH, an automated pipeline that converts any 3D mesh into a corresponding articulated asset. In order to go beyond existing articulated object datasets, our pipeline leverages prior knowledge from visual and language foundation models [35, 36, 37] and generalizable geometric clues, rather than relying on existing labeled articulated object data. Our pipeline starts with a 3D mesh, either generated or handcrafted, followed by the stages of **Movable Part Segmentation**, **Articulation Estimation** and **Post-Processing**.

In **Movable Part Segmentation**, the goal is to identify all movable parts and determine their semantics for subsequent articulation estimation. Recent open-vocabulary 3D part segmentation methods [38, 39, 40, 41, 42, 23] utilize an open-vocabulary segmentation model (namely SAM) and a multi-modal foundation model on rendered 2D images, integrating the 2D information to achieve 3D part segmentation. In this work, we adopt the pipeline proposed by PartSlip++ [39] for 3D part segmentation.

After obtaining the 3D segmentation of non-fixed parts and their semantics, the next stage in our proposed pipeline focuses **Articulation Estimation**. Unlike previous approaches [43, 31, 44], which rely on training datasets to predict articulation parameters. To overcome the limitations of existing datasets and extent to open-vocabulary manner, we extract geometric clues inherent in articulated objects instead of relying on learned models. Specifically, the joint connecting two segmented parts is inherently related to the geometry of the connected area. For example, in cases where the connection follows a straight line, such as a laptop hinge or an open door, this line effectively represents the joint since the hinge structures align with it. We provide GPT-4o with such information through visual prompting, allowing it to leverage common-sense knowledge geometric reasoning to infer joint parameters. This approach allows generalization to unseen objects with accurate segmentation, and the performance will continue to improve as the vision-language model becomes more advanced.

At this stage, a functional articulated object has been created. However, if the input mesh is a generated or scanned surface mesh that contains only surface geometry and texture, the segmented 3D parts often suffer from occlusion and may exhibit holes. For example, in the closed state, only the outer surface of a microwave may be segmented, with the geometry of its internal cavity entirely missing. To address these issues and complete the pipeline, we incorporate an optional final stage: **Post-Processing**. Here, we leverage 3D generative models [45, 46] to perform shape completion (filling holes and recovering missing geometry) and texture generation, making the pipeline self-contained for surface meshes.

We conducted experiments to compare joint parameter estimation accuracy with various baselines. Experimental results demonstrate that while our method matches state-of-the-art articulated object modeling methods for selected categories that they are trained on within PartNet-Mobility dataset [32], it also exhibits the capacity to model a broader range of articulated objects beyond these categories. We summarize our contributions as follows:

- We introduce ARTICULATE ANYMESH, a pipeline capable of creating diverse, realistic and complex articulated objects in an open-vocabulary manner.
- We propose a novel articulation parameter estimation method that leverages geometric clues and a tailored visual prompting method to estimate joint parameters in an open-vocabulary manner.
- We conducted extensive experiments to evaluate the performance of our pipeline. We compared our method with others in joint parameter estimation and outperformed them in both in-domain and out-of-domain settings. We also carried out manipulation policy learning experiments to demonstrate the usefulness of the data collected by our pipeline.

2 Related Work

2.1 Articulated object modeling

One line of work focuses on the perception end, including reconstruction of articulated objects from observations, joint parameter estimation, etc. [24] and [31] train a network to reconstruct an articulated object from input multi-frame point clouds. Some other works [47, 30, 29, 48] learn joint parameters from active interaction. [25], [49], [50] and [28] reconstruct the geometry and joint parameters of an articulated object from multiview images obtained at distinct states of the articulated object. [2] reconstructs articulated objects from a single real-world RGB image through a network trained on large-scale synthetic data. [43, 51, 52, 53, 54, 55, 34] estimates joint parameters and part segments given the poincloud or image of an articulated object. However, these approaches either demand extensive observation of the same object or rely on existing articulated object data to train neural networks, with limited ability to generalize across only a small range of categories.

Alongside perception works, generation works including [26], [27] and [56], represents articulated objects as graphs and employs a diffusion model to fit the distribution. The shape, position, and semantic information of the parts are encoded in the vertices, while joint parameters are stored in the edges. Similar to perception methods, these generation methods also depend on existing articulated object datasets to train their diffusion models and are unable to generalize beyond the training data.

Most related to our work is Articulate-Anything[57], which uses VLMs to generate articulated objects from text, images, or videos. However, it requires a dataset to retrieve part meshes, and thus cannot operate in an open-vocabulary manner. Overall, none of the existing methods can automatically create articulated objects in an open-vocabulary manner.

2.2 Part aware 3D generation

Instead of generating an 3D object as a whole, part aware 3D generation methods generate objects with part-level information. [17] learns two separate VAEs to model global structural information and detailed part geometries. [18] encodes 3D objects into a representation that disentangles structure and geometry. [19] utilizes a graph VAE to encode shapes into hierarchical graph representations.

[20] applies a Seq2Seq generative network for part assembly and generation. [21] independently models part geometry and global part transformation, and conditioned on both of them, a diffusion model synthesizes the final shape. [22], on the other hand, generates part geometry latents with one diffusion model and synthesizes the global geometry with another diffusion model conditioned on these latents. [23] learns a Neus model from generated multi-view images and corresponding 2D segmentation maps provided by [36], and extracts 3D segmentation masks using a clustering algorithm. [59] starts from multi-view images, applies diffusion-based networks for consistent part segmentation, completes each part across views, and reconstructs 3D parts using a pre-trained model..

While these methods can produce shapes with plausible structures and part geometries, they frequently depend on object datasets with part-level annotations and fail to generalize beyond the datasets used for training. Furthermore, they do not generate articulation parameters for individual parts, causing the generated parts to be individually non-manipulatable in simulation, which limits their applicability in downstream tasks for robot learning.

3 Method

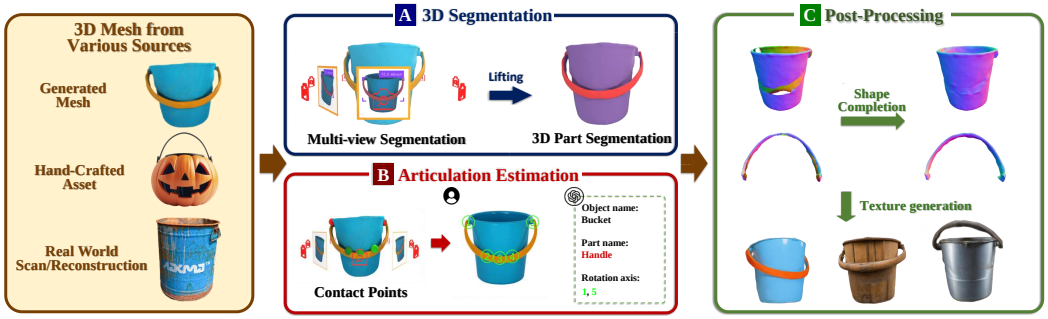


Figure 2: ARTICULATE ANYMESH converts 3D meshes from various sources to high quality articulated objects via three main parts: **A.** Movable Part Segmentation, **B.** Articulation Estimation, **C.** Post-Processing.

3.1 Overview

Our pipeline converts any mesh into its articulated counterpart, including hand-crafted meshes with part geometry, surface meshes generated through text-to-3D or image-to-3D methods and meshes reconstructed from real-world objects. The process involves three main steps, including **Movable Part Segmentation**, **Articulation Estimation** and **Post-Processing**, as depicted in Figure 2. Specifically, the **Movable Part Segmentation** step employs the method proposed by PartSlip++ [38] to segment movable parts from the input mesh. **Articulation Estimation** extracts the articulation parameters for each movable part by leveraging geometric cues and prior knowledge from visual and language foundation models. The optional **Post-Processing** stage enhances geometry and texture using an off-the-shelf 3D shape completion model [45] and an off-the-shelf texture generation model [46].

3.2 Movable Part Segmentation via VLM assistant

As shown in Figure 2A, this step segments all movable parts from a 3D mesh and identifies their semantics in an open-vocabulary manner. We primarily adopt PartSlip++ [39] for segmentation. Given a 3D object and a list of part labels, PartSlip++ applies an open-vocabulary grounding model [60, 61] and SAM to 2D rendered images to extract bounding boxes and segmentation masks for the relevant semantic labels. These multi-view 2D detections are then fused to produce the final 3D part segmentation. In our pipeline, the label list is generated by feeding an image of the input mesh into a VLM to identify all movable parts.

3.3 Articulation Estimation via Geometry-aware Visual Prompting

Achieving an open-vocabulary scope for articulation estimation requires going beyond the limited-scale datasets of articulated objects. To address this, we propose a *Geometry-aware Visual Prompting* approach that generalizes across a diverse range of articulated objects by leveraging shared features in geometry, mechanical structure, and functionality. Specifically, we observe that joints connecting two parts are strongly relevant to the geometry in areas where the parts are in close proximity, which we will refer to as the *connecting area*.

Denote the point clouds of two neighboring parts a and b as P_a and P_b respectively. To define the connecting area, we select the points in P_a whose distance to the closest point in P_b is below a predefined threshold and vice versa for points in P_b. The selected points from P_a and P_b together form the connecting area between the parts. If no point is selected, the threshold is doubled, and the process is repeated until a connecting area is identified. We focus primarily on two dominant joint types: prismatic joints and revolute joints. These are estimated using distinct methods tailored to their respective mechanical structures.

Revolute joints connect two parts of an articulated object through physical hinges or other similar hinge-like mechanisms, like the lid of a cardboard box. The rotation axis of a revolute joint aligns with the length of its hinge, meaning that identifying hinges in 3D space allows for straightforward estimation of the rotation axis. Since a hinge physically connects two parts, it is typically located within the region where the parts are joined. Driven by this observation, we propose the following visual prompting procedure. First, we divide the connecting area into multiple clusters, with the cluster centers serving as candidate points. These candidate points are then projected onto 2D rendered images of the input mesh and labeled with numbers, as illustrated in Figure 2B. This image is used to prompt GPT4o to select two or more points that define the hinge, thereby establishing the rotation axis. The rendering viewpoint is chosen to maximize the number of pixels of the part under consideration. K-means is run multiple times with varying cluster counts, and the final count is selected to maximize the number of candidate points while avoiding overlapping labels on the rendered image.

In some cases, a hinge may not be positioned on the object's surface, resulting in only one end of the rotation axis identifiable on the object's surface, such as with knobs. In these cases, we assume the rotation axis to be perpendicular to the plane fitted to the connected area. The axis is then determined by the normal vector of this plane and the single selected point. When the hinge mechanism is hidden beneath the surface, the connecting structure cannot be assumed to be entirely within the connecting area. Thus, in addition to candidate points from the connecting area, we sample additional points from the part surface. Specifically, we use the l_0-cut pursuit [62] algorithm to generate super points from the part's point cloud, using point normals and color as features, and derive candidate points from these super points. This ensures the new candidate points are distributed across different geometric and textured regions of the part.

Prismatic joints connect two parts of an articulated object through sliding mechanisms, enabling linear motion along a single axis. If the input mesh includes physically accurate inner geometry, sampling a collision-free sliding direction is likely to yield an accurate prismatic joint axis. However, most meshes, particularly surface meshes, lack the ideal geometry needed for this collision-based approach. Additionally, the sliding mechanism is often concealed beneath the object's surface, making it less identifiable compared to hinge structures for revolute joints and making the strategy used for revolute joints impractical for prismatic joints.

Based on their functionality, prismatic joints can be categorized into two types:

1. **Inward & Outward sliding joints**: These joints allow the child component to slide in and out of the object it is connected to, as seen in parts like drawers and push buttons. Although the translation direction may vary, the motion to pull the part out is predominantly outward from the object. Consequently, the sliding direction is characterized by the normal vector of the plane fitted to the connecting area.

2. **Surface sliding joints**: These joints enable the child component to slide along the surface of the object, as seen in parts like sliding windows and toaster levers. For such joints, the potential translation direction is constrained to two dimensions along the surface. This allows us to annotate a 2D rendered image with arrows and prompt GPT-4 to select the most appropriate direction.

Given an articulated object and its segmented parts, we prompt GPT-4 to classify all prismatic joints into these two categories. Based on the classification, the translation directions of the prismatic joints are then estimated accordingly.

3.4 Geometry and Texture Post-Processing

After completing 3D segmentation and joint estimation, we obtain an articulated object with the correct articulation structure. However, when the input is a surface mesh, the result often remains incomplete. Parts segmented from surface meshes—lacking internal geometry—are prone to occlusions, resulting in incomplete meshes with holes. To address this, we introduce an optional post-processing step for surface meshes (primarily generated ones), where off-the-shelf 3D generation models are used for shape completion and texture generation, as illustrated in Section C of Figure 2.

We use HoloPart[45] for shape completion, which takes in a mesh with 3D segmentation labels and outputs mesh parts with completed geometry. As shown in the pipeline figure, HoloPart repairs issues such as holes in the bucket base (caused by occlusion from the handle) and a broken handle. For texture generation, we use Meshy[46] to apply textures to the completed parts. As shown in the figure, Meshy can generate high-quality textures with diverse materials, such as wood, plastic, and metal.

4 Experiments

Implementation In the **movable part segmentation** stage, we use Partslip++[39] for segmentation and replace the grounding model with DINO-X[60] to achieve better performance. In addition, GPT4o is prompted to extract part labels of the input mesh and some of its joint configurations like joint type, parent link, joint limit, etc. In **articulation estimation** stage, we visually prompt GPT4o for the rest of the joint configurations. In the **Post-Processing** stage, we employ HoloPart [45] for 3D shape completion and Meshy [46] for texture generation.

Baselines To the best of our knowledge, this is the first work to tackle the challenge of converting meshes into articulated objects in an open-vocabulary manner. No existing baseline processes the exact same input and output as ARTICULATE ANYMESH. Therefore, we compare against prior works that perform joint parameter estimation only on object categories seen during training.

URDFormer [2] takes a front-view image as input and sequentially outputs part segmentation, joint parameters, and part geometry. Real2Code [28] takes multi-view images as input and performs part segmentation, shape completion, and joint parameter estimation through code generation. For these two methods, we compare ARTICULATE ANYMESHon the PartNet-Mobility dataset [32]. Additional comparisons with estimation methods such as ANCSH[43], OPD[51], and OPD-Multi[52], as well as generative methods like NAP[26] and CAGE[27], are provided in Appendix section A. We also provide an ablation study analyzing the impact of different vision-language models (VLMs), including GPT, Gemini, and Claude, in Appendix section B.

Metrics For articulation parameter estimation, we evaluate joint directional accuracy using angular error and joint positional accuracy using the distance between lines.

4.1 Quantitative Experiments

We compare ARTICULATE ANYMESHto URDFormer and Real2Code on the PartNet-Mobility dataset. To ensure a comprehensive evaluation, we assess joint estimation performance on both in-domain (ID) and out-of-domain (OOD) object categories. The in-domain categories include the five object types used to train URDFormer: oven, fridge, cabinet, washer, and dishwasher. For Real2Code, we fine-tune a CodeLlama model on objects from the same categories. We align the inputs of all

methods as closely as possible to ensure a fair comparison: URDFormer takes a front-view image and is additionally provided with ground-truth part bounding boxes to predict joint parameters per part; Real2Code takes ground-truth part meshes and performs joint estimation via code generation; ARTICULATE ANYMESHtakes the ground-truth mesh, performs part segmentation, and estimates joint parameters via visual prompting. Results in table 1 show that although URDFormer and Real2Code perform just slightly worse on in-domain categories, their performance drops significantly on out-of-domain objects. In contrast, ARTICULATE ANYMESHmaintains low error across both ID and OOD settings. This highlights the key advantage of our approach: it is not constrained by the biases of existing articulated object datasets and generalizes well beyond them.

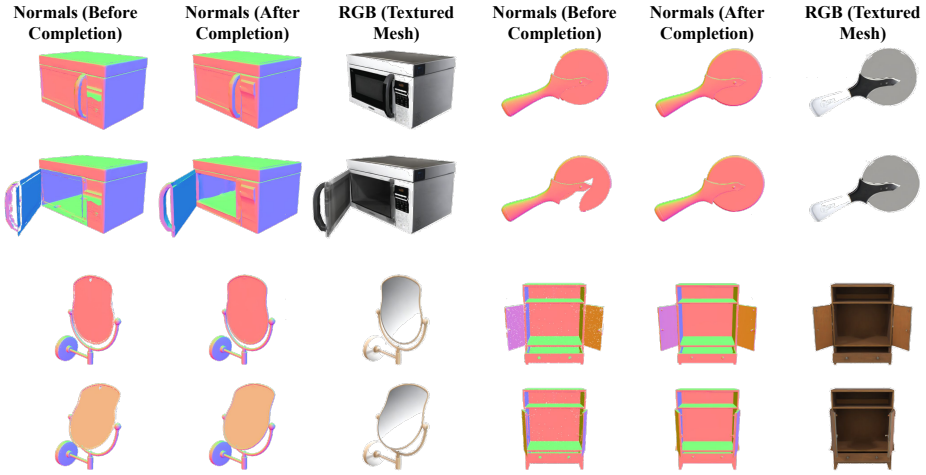


Figure 3: Results of articulated object generation. Each example shows (from left to right): the raw surface mesh generated by Meshy, the completed geometry, and the textured object, all rendered under two different poses.

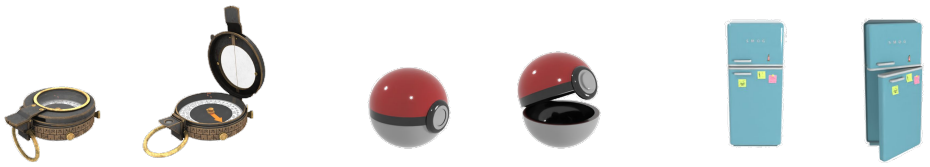


Figure 4: Results of ARTICULATE ANYMESH applied to hand-crafted meshes retrieved from Objaverse.

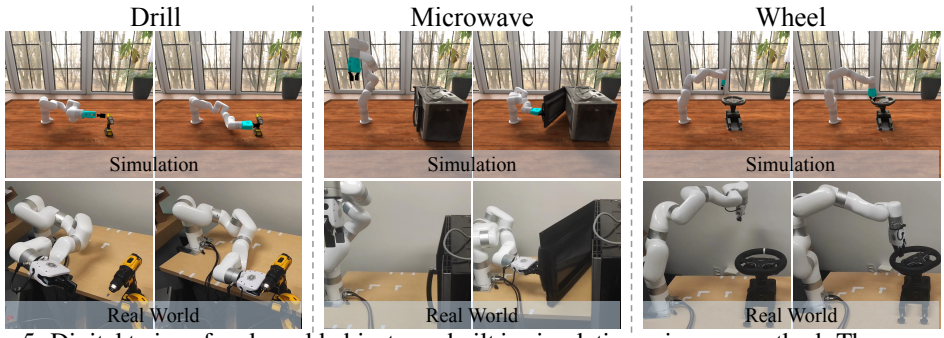


Figure 5: Digital twins of real-world objects are built in simulation using our method. Then we sample trajectories completing the tasks in simulation and replicate those trajectories in the real-world.

4.2 Applications

Articulated object generation By integrating an existing 3D generation model (Meshy[46]) to generate surface meshes as inputs to ARTICULATE ANYMESH, our pipeline exploits the open-

	ID		OOD	
	angle↓	pos↓	angle↓	pos↓
URDFormer	13.674	0.144	33.864	0.299
Real2Code	7.950	0.167	24.035	0.194
Ours	**6.257**	**0.076**	**6.582**	**0.083**

Table 1: Comparison of joint estimation accuracy—including angle and position error—between ARTICULATE ANYMESH, URDFormer and Real2Code.

	Laptop		Bucket	
	original	aug	original	aug
succ. rate ↑	0.408	**0.513**	0.339	**0.479**

Table 2: Success rates of manipulation, comparing fine-tuning on original vs. augmented training sets. Evaluated on the augmented test set.

vocabulary 3D generation capabilities of the 3D generation model and inherits its generative paradigm. As in Figure 3, our extended pipeline generates high-quality articulated objects of various categories.

Annotate 3D object datasets Other than generating from scratch, ARTICULATE ANYMESHcan also start from existing artist designed meshes. For example, we annotate part segmentation and joint structures on objects from Objaverse [1]. Such objects usually have high-quality mesh and textures and also with inner structures. Thus, we only execute the 3D segmentation part and articulation estimation part of our pipeline to annotate these objects. The results are demonstrated in Figure 4. Objects in the teaser (Figure 1) are also produced in this way.

Real-to-Sim-to-Real In this experiment, we first reconstruct the 3D surface mesh of real-world objects using 2DGS [63]. We then apply ARTICULATE ANYMESH to convert the reconstructed mesh into an articulated object represented in URDF format, making it compatible with simulation environments. Next, we sample action targets and use motion planning [64] to avoid collisions, generating a trajectory to successfully complete the task in simulation. Finally, we replicate the trajectory in the real world and observe that the robot arm can effectively execute the task.

We focus on three tasks involving different articulated objects: **pushing a drill trigger**, **opening a microwave door**, and **rotating a steering wheel**, as shown in Figure 5. Our results show that the generated trajectories can be successfully transferred to the real world, demonstrating minimal error in part segmentation and joint prediction.

Policy learning in simulation We follow DexArt [65] for articulated object manipulation policy learning and evaluate two tasks: opening laptop lid and lifting bucket. To augment the training set for the "bucket" and "laptop" experiments, we include additional articulated objects of the same categories generated by ARTICULATE ANYMESH. Additionally, we diversify the testing set by adding some generated articulated objects. We fine-tune the checkpoints trained exclusively on the original dataset and evaluate their performance on the augmented testing set. The results, presented in Table 2, demonstrate that the data generated by ARTICULATE ANYMESH effectively enhances robot learning. The articulated objects generated by ARTICULATE ANYMESH are sourced from Objaverse. For visualizations of the data used and implementation details, please refer to Appendix Section C.

5 Conclusion

In this paper, we introduce ARTICULATE ANYMESH, a novel framework for the critical task of converting open-vocabulary 3D meshes into their articulated counterparts. Our pipeline first segments 3D objects based on part-level semantics, estimates the articulation structure among object parts, and then refines the geometry and texture to repair flaws and enhance visual quality. By leveraging advancements in vision-language models and visual prompting techniques, we effectively segment parts and estimate articulation structures using common geometric cues. Our pipeline creates high-quality textured articulated objects from generated 3D meshes, hand-crafted 3D assets, and reconstructed meshes. The resulting assets can support the learning of articulated object manipulation skills in simulation, which can then be transferred to real-world robots.

6 Limitations

Although our method creates articulated objects with semantically correct parts, the articulation parameters are not always accurate or physically grounded. This limitation arises from three key challenges: (1) existing and generated 3D meshes often lack correct physical structures, (2) current 3D segmentation methods occasionally produce incorrect results, and (3) GPT4o exhibits biases in certain scenarios. Additionally, some uncommon cases are not covered by our method—for example, when a drawer is designed to be pulled out up-front.

If future advancements lead to the development of large VLMs with enhanced 3D reasoning capabilities, our pipeline could leverage these improvements to become more robust and reliable. Addressing these challenges to create fully accurate and physically grounded articulation parameters represents an exciting direction for future research.

References

[1] M. Deitke, R. Liu, M. Wallingford, H. Ngo, O. Michel, A. Kusupati, A. Fan, C. Laforte, V. Voleti, S. Y. Gadre, et al. Objaverse-xl: A universe of 10m+ 3d objects. *Advances in Neural Information Processing Systems*, 36, 2024.

[2] Z. Chen, A. Walsman, M. Memmel, K. Mo, A. Fang, K. Vemuri, A. Wu, D. Fox, and A. Gupta. Urdformer: A pipeline for constructing articulated simulation environments from real-world images. *arXiv preprint arXiv:2405.11656*, 2024.

[3] Y. Yang, B. Jia, P. Zhi, and S. Huang. Physcene: Physically interactable 3d scene synthesis for embodied ai. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16262–16272, 2024.

[4] Y. Yang, F.-Y. Sun, L. Weihs, E. VanderBilt, A. Herrasti, W. Han, J. Wu, N. Haber, R. Krishna, L. Liu, et al. Holodeck: Language guided generation of 3d embodied ai environments. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16227–16237, 2024.

[5] Y. Wang, Z. Xian, F. Chen, T.-H. Wang, Y. Wang, K. Fragkiadaki, Z. Erickson, D. Held, and C. Gan. Robogen: Towards unleashing infinite data for automated robot learning via generative simulation. *arXiv preprint arXiv:2311.01455*, 2023.

[6] M. Dalal, A. Mandlekar, C. Garrett, A. Handa, R. Salakhutdinov, and D. Fox. Imitating task and motion planning with visuomotor transformers. *arXiv preprint arXiv:2305.16309*, 2023.

[7] H. Ha, P. Florence, and S. Song. Scaling up and distilling down: Language-guided robot skill acquisition. In *Conference on Robot Learning*, pages 3766–3777. PMLR, 2023.

[8] Y. J. Ma, W. Liang, G. Wang, D.-A. Huang, O. Bastani, D. Jayaraman, Y. Zhu, L. Fan, and A. Anandkumar. Eureka: Human-level reward design via coding large language models. *arXiv preprint arXiv:2310.12931*, 2023.

[9] X. Long, Y.-C. Guo, C. Lin, Y. Liu, Z. Dou, L. Liu, Y. Ma, S.-H. Zhang, M. Habermann, C. Theobalt, et al. Wonder3d: Single image to 3d using cross-domain diffusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9970–9980, 2024.

[10] Y. Hong, K. Zhang, J. Gu, S. Bi, Y. Zhou, D. Liu, F. Liu, K. Sunkavalli, T. Bui, and H. Tan. Lrm: Large reconstruction model for single image to 3d. *arXiv preprint arXiv:2311.04400*, 2023.

[11] J. Xu, W. Cheng, Y. Gao, X. Wang, S. Gao, and Y. Shan. Instantmesh: Efficient 3d mesh generation from a single image with sparse-view large reconstruction models. *arXiv preprint arXiv:2404.07191*, 2024.

[12] R. Shi, H. Chen, Z. Zhang, M. Liu, C. Xu, X. Wei, L. Chen, C. Zeng, and H. Su. Zero123++: a single image to consistent multi-view diffusion base model. *arXiv preprint arXiv:2310.15110*, 2023.

[13] Y. Shi, P. Wang, J. Ye, M. Long, K. Li, and X. Yang. Mvdream: Multi-view diffusion for 3d generation. *arXiv preprint arXiv:2308.16512*, 2023.

[14] B. Poole, A. Jain, J. T. Barron, and B. Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. *arXiv preprint arXiv:2209.14988*, 2022.

[15] Z. Wang, C. Lu, Y. Wang, F. Bao, C. Li, H. Su, and J. Zhu. Prolificdreamer: High-fidelity and diverse text-to-3d generation with variational score distillation. *Advances in Neural Information Processing Systems*, 36, 2024.

[16] L. Qiu, G. Chen, X. Gu, Q. Zuo, M. Xu, Y. Wu, W. Yuan, Z. Dong, L. Bo, and X. Han. Richdreamer: A generalizable normal-depth diffusion model for detail richness in text-to-3d. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9914–9925, 2024.

[17] L. Gao, J. Yang, T. Wu, Y.-J. Yuan, H. Fu, Y.-K. Lai, and H. Zhang. Sdm-net: Deep generative network for structured deformable mesh. *ACM Transactions on Graphics (TOG)*, 38(6):1–15, 2019.

[18] J. Yang, K. Mo, Y.-K. Lai, L. J. Guibas, and L. Gao. Dsg-net: Learning disentangled structure and geometry for 3d shape generation. *ACM Transactions on Graphics (TOG)*, 42(1):1–17, 2022.

[19] K. Mo, P. Guerrero, L. Yi, H. Su, P. Wonka, N. Mitra, and L. J. Guibas. Structurenet: Hierarchical graph networks for 3d shape generation. *arXiv preprint arXiv:1908.00575*, 2019.

[20] R. Wu, Y. Zhuang, K. Xu, H. Zhang, and B. Chen. Pq-net: A generative part seq2seq network for 3d shapes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 829–838, 2020.

[21] G. K. Nakayama, M. A. Uy, J. Huang, S.-M. Hu, K. Li, and L. Guibas. Difffacto: Controllable part-based 3d point cloud generation with cross diffusion. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14257–14267, 2023.

[22] J. Koo, S. Yoo, M. H. Nguyen, and M. Sung. Salad: Part-level latent diffusion for 3d shape generation and manipulation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14441–14451, 2023.

[23] A. Liu, C. Lin, Y. Liu, X. Long, Z. Dou, H.-X. Guo, P. Luo, and W. Wang. Part123: Part-aware 3d reconstruction from a single-view image. In *ACM SIGGRAPH 2024 Conference Papers*, pages 1–12, 2024.

[24] Z. Jiang, C.-C. Hsu, and Y. Zhu. Ditto: Building digital twins of articulated objects from interaction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5616–5626, 2022.

[25] J. Liu, A. Mahdavi-Amiri, and M. Savva. Paris: Part-level reconstruction and motion analysis for articulated objects. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 352–363, 2023.

[26] J. Lei, C. Deng, W. B. Shen, L. J. Guibas, and K. Daniilidis. Nap: Neural 3d articulated object prior. *Advances in Neural Information Processing Systems*, 36:31878–31894, 2023.

[27] J. Liu, H. I. I. Tam, A. Mahdavi-Amiri, and M. Savva. Cage: Controllable articulation generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17880–17889, 2024.

[28] Z. Mandi, Y. Weng, D. Bauer, and S. Song. Real2code: Reconstruct articulated objects via code generation. *arXiv preprint arXiv:2406.08474*, 2024.

[29] N. Nie, S. Y. Gadre, K. Ehsani, and S. Song. Structure from action: Learning interactions for 3d articulated object structure discovery. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1222–1229. IEEE, 2023.

[30] S. Y. Gadre, K. Ehsani, and S. Song. Act the part: Learning interaction strategies for articulated object part discovery. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15752–15761, 2021.

[31] J. Huang, H. Wang, T. Birdal, M. Sung, F. Arrigoni, S.-M. Hu, and L. J. Guibas. Multibodysync: Multi-body segmentation and motion estimation via 3d scan synchronization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7108–7118, 2021.

[32] F. Xiang, Y. Qin, K. Mo, Y. Xia, H. Zhu, F. Liu, M. Liu, H. Jiang, Y. Yuan, H. Wang, et al. Sapien: A simulated part-based interactive environment. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11097–11107, 2020.

[33] X. Wang, B. Zhou, Y. Shi, X. Chen, Q. Zhao, and K. Xu. Shape2motion: Joint analysis of motion parts and attributes from 3d shapes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8876–8884, 2019.

[34] H. Geng, H. Xu, C. Zhao, C. Xu, L. Yi, S. Huang, and H. Wang. Gapartnet: Cross-category domain-generalizable object perception and manipulation via generalizable and actionable parts. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7081–7091, 2023.

[35] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[36] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, et al. Segment anything. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4015–4026, 2023.

[37] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022.

[38] M. Liu, Y. Zhu, H. Cai, S. Han, Z. Ling, F. Porikli, and H. Su. Partslip: Low-shot part segmentation for 3d point clouds via pretrained image-language models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 21736–21746, 2023.

[39] Y. Zhou, J. Gu, X. Li, M. Liu, Y. Fang, and H. Su. Partslip++: Enhancing low-shot 3d part segmentation via multi-view instance segmentation and maximum likelihood estimation. *arXiv preprint arXiv:2312.03015*, 2023.

[40] A. Umam, C.-K. Yang, M.-H. Chen, J.-H. Chuang, and Y.-Y. Lin. Partdistill: 3d shape part segmentation by vision-language model distillation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3470–3479, 2024.

[41] Y. Xue, N. Chen, J. Liu, and W. Sun. Zerops: High-quality cross-modal knowledge transfer for zero-shot 3d part segmentation. *arXiv preprint arXiv:2311.14262*, 2023.

[42] Y. Yang, Y. Huang, Y.-C. Guo, L. Lu, X. Wu, E. Y. Lam, Y.-P. Cao, and X. Liu. Sampart3d: Segment any part in 3d objects. *arXiv preprint arXiv:2411.07184*, 2024.

[43] X. Li, H. Wang, L. Yi, L. J. Guibas, A. L. Abbott, and S. Song. Category-level articulated object pose estimation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3706–3715, 2020.

[44] H. Zeng, P. Zhang, C. Wu, J. Wang, T. Ye, and F. Li. Mars: Multimodal active robotic sensing for articulated characterization. *arXiv preprint arXiv:2407.01191*, 2024.

[45] Y. Yang, Y.-C. Guo, Y. Huang, Z.-X. Zou, Z. Yu, Y. Li, Y.-P. Cao, and X. Liu. Holopart: Generative 3d part amodal segmentation. *arXiv preprint arXiv:2504.07943*, 2025.

[46] Meshy ai, 2025. URL https://www.meshy.ai.

[47] C.-C. Hsu, Z. Jiang, and Y. Zhu. Ditto in the house: Building articulation models of indoor scenes through interactive perception. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3933–3939. IEEE, 2023.

[48] Y. Wang, R. Wu, K. Mo, J. Ke, Q. Fan, L. J. Guibas, and H. Dong. Adaafford: Learning to adapt manipulation affordance for 3d articulated objects via few-shot interactions. In *European conference on computer vision*, pages 90–107. Springer, 2022.

[49] Y. Weng, B. Wen, J. Tremblay, V. Blukis, D. Fox, L. Guibas, and S. Birchfield. Neural implicit representation for building digital twins of unknown articulated objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3141–3150, 2024.

[50] F. Wei, R. Chabra, L. Ma, C. Lassner, M. Zollhöfer, S. Rusinkiewicz, C. Sweeney, R. Newcombe, and M. Slavcheva. Self-supervised neural articulated shape and appearance models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15816–15826, 2022.

[51] H. Jiang, Y. Mao, M. Savva, and A. X. Chang. Opd: Single-view 3d openable part detection. In *European Conference on Computer Vision*, pages 410–426. Springer, 2022.

[52] X. Sun, H. Jiang, M. Savva, and A. Chang. Opdmulti: Openable part detection for multiple objects. In *2024 International Conference on 3D Vision (3DV)*, pages 169–178. IEEE, 2024.

[53] Z. Yan, R. Hu, X. Yan, L. Chen, O. Van Kaick, H. Zhang, and H. Huang. Rpm-net: recurrent prediction of motion and parts from point cloud. *arXiv preprint arXiv:2006.14865*, 2020.

[54] L. Liu, H. Xue, W. Xu, H. Fu, and C. Lu. Toward real-world category-level articulation pose estimation. *IEEE Transactions on Image Processing*, 31:1072–1083, 2022.

[55] X. Liu, J. Zhang, R. Hu, H. Huang, H. Wang, and L. Yi. Self-supervised category-level articulated object pose estimation with part-level se (3) equivariance. *arXiv preprint arXiv:2302.14268*, 2023.

[56] J. Liu, D. Iliash, A. X. Chang, M. Savva, and A. Mahdavi-Amiri. Singapo: Single image controlled generation of articulated parts in objects. *arXiv preprint arXiv:2410.16499*, 2024.

[57] L. Le, J. Xie, W. Liang, H.-J. Wang, Y. Yang, Y. J. Ma, K. Vedder, A. Krishna, D. Jayaraman, and E. Eaton. Articulate-anything: Automatic modeling of articulated objects via a vision-language foundation model. *arXiv preprint arXiv:2410.13882*, 2024.

[58] J. Liu, M. Savva, and A. Mahdavi-Amiri. Survey on modeling of articulated objects. *arXiv preprint arXiv:2403.14937*, 2024.

[59] M. Chen, R. Shapovalov, I. Laina, T. Monnier, J. Wang, D. Novotny, and A. Vedaldi. Partgen: Part-level 3d generation and reconstruction with multi-view diffusion models. *arXiv preprint arXiv:2412.18608*, 2024.

[60] T. Ren, Y. Chen, Q. Jiang, Z. Zeng, Y. Xiong, W. Liu, Z. Ma, J. Shen, Y. Gao, X. Jiang, et al. Dino-x: A unified vision model for open-world object detection and understanding. *arXiv preprint arXiv:2411.14347*, 2024.

[61] L. H. Li, P. Zhang, H. Zhang, J. Yang, C. Li, Y. Zhong, L. Wang, L. Yuan, L. Zhang, J.-N. Hwang, et al. Grounded language-image pre-training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10965–10975, 2022.

[62] L. Landrieu and G. Obozinski. Cut pursuit: Fast algorithms to learn piecewise constant functions on general weighted graphs. *SIAM Journal on Imaging Sciences*, 10(4):1724–1766, 2017.

[63] B. Huang, Z. Yu, A. Chen, A. Geiger, and S. Gao. 2d gaussian splatting for geometrically accurate radiance fields. In *ACM SIGGRAPH 2024 conference papers*, pages 1–11, 2024.

[64] I. A. Sucan, M. Moll, and L. E. Kavraki. The open motion planning library. *IEEE Robotics & Automation Magazine*, 19(4):72–82, 2012.

[65] C. Bao, H. Xu, Y. Qin, and X. Wang. Dexart: Benchmarking generalizable dexterous manipulation with articulated objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21190–21200, 2023.

A More quantitative results on joint parameter estimation

In this section, we provide additional comparisons with methods not covered in the main paper for joint parameter estimation. Specifically, we include two generative approaches—NAP and CAGE—that leverage diffusion models to synthesize articulated objects and can generate joint configurations conditioned on object parts. In addition to these generative baselines, we also compare against discriminative methods such as ANCSH[43], OPD[51], and OPDmulti[52]. These methods take a single observation (e.g., RGB, depth, or both) as input and jointly predict part segmentation and articulation parameters. Together, these methods provide a broader context for evaluating joint estimation performance.

We first evaluate performance in the single-observation setting, using the OPD-Synth dataset. In this experiment, we compare ARTICULATE ANYMESHagainst ANCSH, OPD, and OPDmulti for articulation parameter estimation from a single-view input. ANCSH uses a single-view point cloud as input, while OPD and OPDmulti operate on RGB images, optionally supplemented with depth. All three methods jointly predict part segmentation and joint parameters. To ensure a fair comparison, we adapt ARTICULATE ANYMESHto this setting by restricting segmentation to a single image, which is then projected into a point cloud with part labels for articulation parameter estimation. As reported in Table 3, ARTICULATE ANYMESH consistently outperforms all three methods within their respective training domains, despite being designed as a training-free, open-vocabulary approach.

We also evaluate articulation parameter estimation using ground-truth shapes of object parts as input, from PartNet-Mobility. Here, we compare ARTICULATE ANYMESHwith the generative diffusion-based models NAP and CAGE, both of which represent articulated objects as part graphs. NAP predicts joint parameters conditioned on part-level geometry descriptors such as bounding boxes, spatial locations, and learned shape latents. CAGE, by contrast, generates joint axes and ranges given attributes including part bounding boxes, semantic labels, and joint types. Following the experimental setup in CAGE, we use the PartNet-Mobility dataset and retrain NAP on the same train-test split. We evaluate both in-domain performance on test categories and generalization to held-out categories. As shown in Table4, while all three methods perform comparably in-domain, ARTICULATE ANYMESH substantially outperforms NAP and CAGE on previously unseen object categories, demonstrating superior generalization.

	ANCSH	OPD	OPDmulti	**Ours**
angle error ↓	6.74	10.73	9.66	**5.37**
position error ↓	0.065	0.117	0.108	**0.049**

Table 3: The results of single observation joint estimation. The dataset for training and testing is the one-door dataset (a subset of OPDsynth), which is used to train ANCSH as provided in its PyTorch version Github repository.

	angle error		position error	
	ID ↓	OOD ↓	ID ↓	OOD ↓
NAP	31.89	42.23	0.53	0.225
CAGE	8.96	58.64	**0.136**	0.192
Ours	**7.90**	**4.81**	0.190	**0.075**

Table 4: The results for object part mesh joint estimation are evaluated on two sets of testing object categories: **in-domain (ID)**, consisting of categories included in CAGE's training set, and **out-of-domain (OOD)**, consisting of some categories from PartNet-Mobility that are not part of CAGE's training set.

	ID		OOD	
	angle↓	pos↓	angle↓	pos↓
URDFormer	13.674	0.144	33.864	0.299
Real2Code	7.950	0.167	24.035	0.194
Ours (Claude)	9.291	0.160	7.541	0.096
Ours (Gemini)	6.477	0.239	6.612	0.081
Ours (GPT)	**6.257**	**0.076**	**6.582**	**0.083**

Table 5: Comparison of joint estimation accuracy (angle and position error) using different vision-language models.

B Ablation study of VLM

To investigate the effect of different vision-language model (VLM) choices, we conducted joint parameter estimation experiments using three VLMs: GPT-4o, Claude 3.7 Sonnet, and Gemini 2.0 Flash. Ground-truth part meshes from selected PartNet-Mobility categories (consistent with the setting used in the comparison with URDFormer and Real2Code, Table 1) were used as input for part segmentation and joint prediction. The results, presented in Table 5, indicate that GPT-4o slightly outperforms Claude and Gemini, although the performance gap is relatively small. Notably, all three variants significantly outperform URDFormer and Real2Code across both in-domain and out-of-domain settings.

C Implementation of policy learning experiment

In this experiment, we evaluate two tasks proposed by DexArt [65]. The first task is lifting a bucket, where a dexterous hand must grasp the bucket handle and lift it. In the original implementation, the bucket handle is thickened to prevent severe penetration, and we adhere to this setting. The second task is opening a laptop lid, where a dexterous hand must open the lid to a 90-degree angle.

For both tasks, we fine-tune the "no-pretrain" policy checkpoint provided by DexArt's official GitHub repository using a combined training set consisting of DexArt's original samples (sourced from PartNet-Mobility [32]) and additional samples (sourced from Objaverse [1]) generated by ARTICULATE ANYMESH. As a baseline, we fine-tune the policy for the same number of steps but only use the original dataset. During evaluation, the new policies are tested on both the original test set and the additional test samples generated by ARTICULATE ANYMESH. We conducted 100 evaluations on the test set, where each evaluation included all test set objects and included randomization in position and rotation. The final success rate was obtained by averaging across all evaluations. Visualizations of the datasets used are provided in Figures 6, 7, and 8.

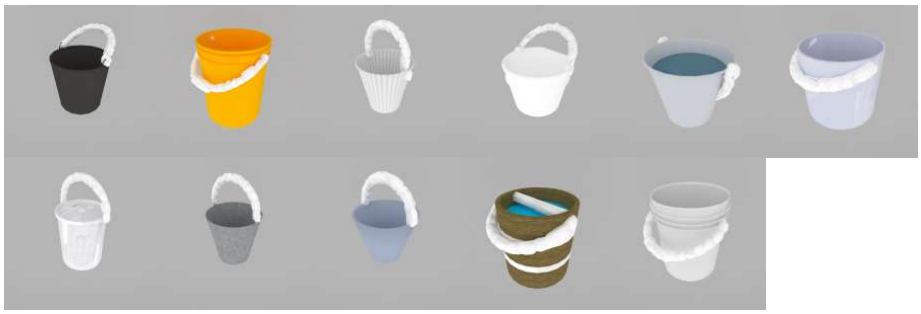
D Prompting details

The prompt for recognizing parts in an rendered image of a input mesh:

```
You have a good understanding of the structure of articulated
objects. Your job is to assist the user to analyze the structure of
an object. Specifically, the user will give you an image of an
articulated object, and your task is to recognize the main parts of
that object. You should give your answer in the following format:

‘‘‘part_list
(1) part_name: name of the part; description: a brief description
about the part, and how it moves
(2) part_name: name of the part; description: a brief description
about the part, and how it moves
...
```

DexArt's original train set (Bucket)



Additionally generated train set (Bucket)

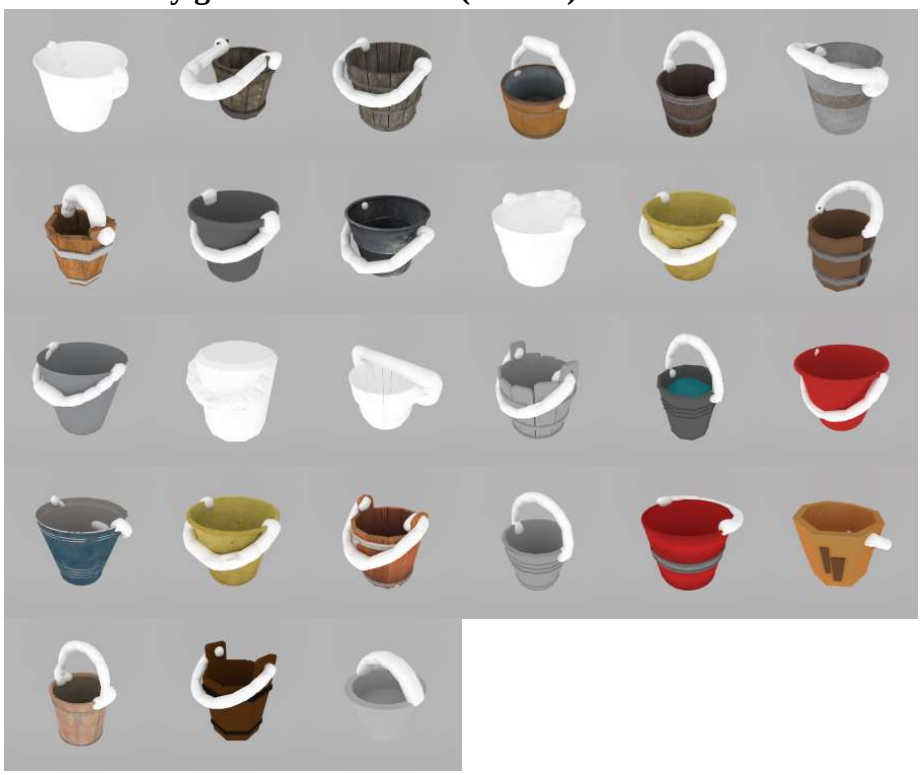


Figure 6: Visualization of the original training set for the bucket task from DexArt [65] and the augmented training set collected using ARTICULATE ANYMESH.

DexArt's original train set (Laptop)

Additionally generated train set (Laptop)

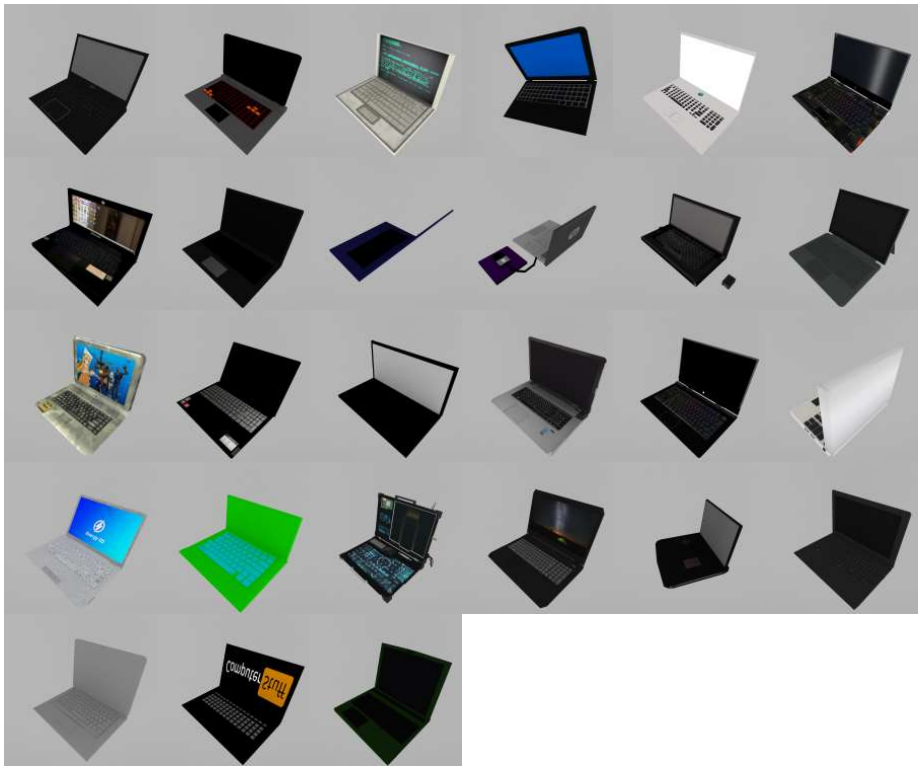
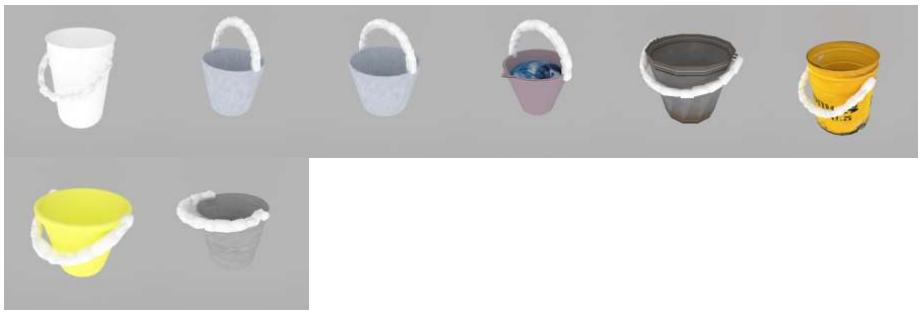


Figure 7: Visualization of the original training set for the laptop task from DexArt [65] and the augmented training set collected using ARTICULATE ANYMESH.

DexArt's original test set (Bucket)



Additionally generated test set (Bucket)

DexArt's original test set (Laptop)

Additionally generated test set (Laptop)

Figure 8: Visualization of the original test set used by DexArt [65] and augmented test set collected using ARTICULATE ANYMESH.

```
```

Remember:
(1) Do not answer anything not asked.
(2) Your answer should be purely based on the input image, do not
imagine anything.
(3) If there are multiple parts with the same semantic, just add
one part to the list. For example, if there are four wheels, just
add one part whose name is wheel.
```

The prompt to generate the link and joint configuration of the articulated object, and exclude non-movable parts:

```
You have a good understanding of the structure of articulated
objects. You are very familiar with URDF format. Your job is to
assist the user to analyze the structure of an articulated object.
Specifically, the user will name an object and then give you the
main parts of that object. You will have to group these parts into
links and then give the joints connecting these links. You should
give your answer in the following format:

```articulation tree
parts:
(1) part_name: name of the recognized part;
...

links:
(1) link_name: name of the link;
...

joints:
(1) joint_name: name of the joint; joint_type: type of the joint;
parent_link: name of the parent link; child_link: name of the child
link; joint_limit: [lower limit, upper limit];
...
```

For example:
```articulation tree
parts:
(1) part_name: Front windshield;
(2) part_name: Doors;
(3) part_name: Headlights;
(4) part_name: Wheels;
(5) part_name: Windows;

links:
(1) link_name: Chasis;
(2) link_name: Doors;
(3) link_name: Wheels;
(4) link_name: Windows;

joints:
(1) joint_name: wheel_chasis_joint; joint_type: continuous;
parent_link: Chasis ; child_link: Wheels; joint_limit: None;
(2) joint_name: door_chasis_joint; joint_type: revolute;
parent_link: Chasis ; child_link: Doors; joint_limit: [0, 90];
(3) joint_name: door_chasis_joint; joint_type: prismatic;
parent_link: Chasis ; child_link: Windows; joint_limit: [0, 1];
```

```
```

Remember:
(1) Do not answer anything not asked.
(2) If a part is actually movable in any direction, its joint type
is floating.
(3) Available joint types are: fixed, prismatic, revolute,
continuous and floating.
(4) For joint_type, only answer one word (among the available
types), do not answer anything else.
(5) For every part that is not fixed, there must be a unique link
for it.
(6) For parts that are fixed, try to group as many as you can.
(7) Joint limit must be given for prismatic joints and revolute
joints. The unit of a revolute joint limit is degrees. The unit of
a prismatic joint limit is the size of its child link in its
translation direction. The joint limit should be two numbers.

Object: OBJECT_NAME
Parts: RECOGNIZED_PARTS
```

The following prompts are for **joint parameter estimation**:

For revolute joints, we first prompt GPT4o to determine whether both points of the joint are on the surface:

```
You are an assistant with a deep understanding of the structure of
objects. Your task is to help users determine the hinge position of
some parts of a given object mesh using common sense. It is
important to note that the object is represented by a mesh, so you
only have access to the object's surface and no access to its inner
structure. The term "hinge" here does not only refer to the
mechanical structure of a hinge, but also has a broader meaning.
For example, the connection between a cardboard box lid and the
body of the box is also considered a hinge.

Specifically, the user will provide you with an object, and the
part for which the hinge position needs to be predicted will be
specified. You will have to decide whether (1) both ends of the
hinge are positioned on the surface of the object or (2) only one
end of the hinge is positioned near the surface of the object, and
the other end is inside the object.

For example, the hinge of a door has both its ends positioned on
the door frame, which is recognizable and falls into the first
category. The hinge of a wheel has one end recognizable on the
center of the wheel, and its other end hidden inside the car
(normally an object mesh of a car will not have detailed mechanical
structures), which falls into the second category.

Please give your answer in the following format:
```hinge_info
description: a brief description of the hinge
choice: (1) or (2)
```
```

Based on GPT4o's selection, we provide a tailored prompt. For choice (1), the specific prompt is as follows. Once GPT4o selects two or more points, we fit a line through these points, defining the rotation axis.
```

```
You are an assistant with a deep understanding of the structure of
objects. Your task is to answer some questions about the input
image of an object. The input image is of a {object_name}. The
image has some points marked, each with an numerical ID as a label.
Please select the points that are on the rotation axis of the
{part_name} of the {object_name}. First provide your analysis, and
then give your answer in the following format:

‘‘‘hinge points
selected IDs: ID of selected point1, ID of selected point2, ...
(for example 1,3)
‘‘‘

Remember:
(1) Do not answer anything not asked for.
(2) Select two or more points.
(3) Give your answer based on the provided image.
```

For choice (2), the prompt is as follows. The rotation axis is determined using the point selected by GPT4o and the normal of the connected area.

```
You are an assistant with a deep understanding of the structure of
objects. Your task is to answer some questions about the input
image of an object. The input image is of a {object_name}. The
image has some points marked, each with an numerical ID as a label.
Please select the point that is on the rotation axis of the
{part_name} of the {object_name}. First provide your analysis, and
then give your answer in the following format:

‘‘‘hinge points
selected IDs: ID of the selected point
‘‘‘

Remember:
(1) Do not answer anything not asked for.
(2) Only select one point that is the most suitable.
(3) Give your answer based on the provided image.
```

For prismatic joints, we prompt GPT4o to determine whether its child link moves in and out or along the surface:

```
You are an assistant with a deep understanding of the structure of
objects. Your task is to help users to determine the translation
direction of some parts of a given object mesh using common sense.
It is important to note that the object is represented by a mesh,
so you only have access to the object’s surface and no access to
its inner structure.

Specifically, the user will provide you with an object and the part
for which the translation direction needs to be predicted will be
specified. You will have to decide whether the translation
direction is outwards from/inwards towards the mesh, or along the
surface of the mesh.

When a part moves outwards, you will see more of that part coming
out from the object. When a part moves inwards, you will see
portions of that part going into the object. When a part moves
along the surface of an object, you still see the exact same part.
```

```
For example, a pressing button can be pressed inwards (when you
press it, the button goes into the object), the telescopic handle
of a suitcase can be pulled outwards (when you pull it, the entire
handle comes out of the suitcase), and a stick shift moves along
the surface of a shift pattern (when you are shifting, the shift
does not go into the transmission or out of the transmission).

Please give your answer in the following format:
‘‘‘translation_axis_info
description: a brief description of the translation axis
choice: outward/inward or surface
‘‘‘
```

If the child link moves inward or outward, the translation direction is defined by the normal vector of the connected area. If it moves along the surface, we draw four arrows (originating from the child link's center and pointing up, down, left, and right) on the image and prompt GPT4o to select the translation direction:

```
You are an assistant with a deep understanding of the structure of
objects. Your task is to answer some questions about the input
image of an object. The input image is of a {object_name}. The
image has some arrows marked, each with a different color. Please
select the arrow that indicate the translation direction of the
{part_name} of the {object_name}. Give your answer in the following
format:

‘‘‘sliding direction
description: a brief description of the direction of the sliding
axis and the selected arrow
selected arrow: color of the arrow (in red, yellow, blue, green)
‘‘‘

Remember:
(1) Do not answer anything not asked for.
(2) Select one arrow.
```

The selected arrow is then projected onto the plane fitted to the connecting area, and the translation direction is defined by the direction of the projected arrow.

4. Validate joint limits. For revolute joints, we incrementally rotate the child link based on the joint parameters and identify the maximum range where penetration remains below a predefined threshold. For prismatic joints, we follow the joint limits provided by GPT4o.