INSTRUCT-PARTICULATE: Scaling Feed-Forward 3D Object Articulation with Kinematic Control

Ruining Li[1*] **Yuxin Yao**[2*] **Matt Zhou**[2] **Chuanxia Zheng**[3]
Christian Rupprecht[1] **Joan Lasenby**[2] **Shangzhe Wu**[2] **Andrea Vedaldi**[1]
[1]University of Oxford [2]University of Cambridge [3]Nanyang Technological University
instruct-particulate.github.io

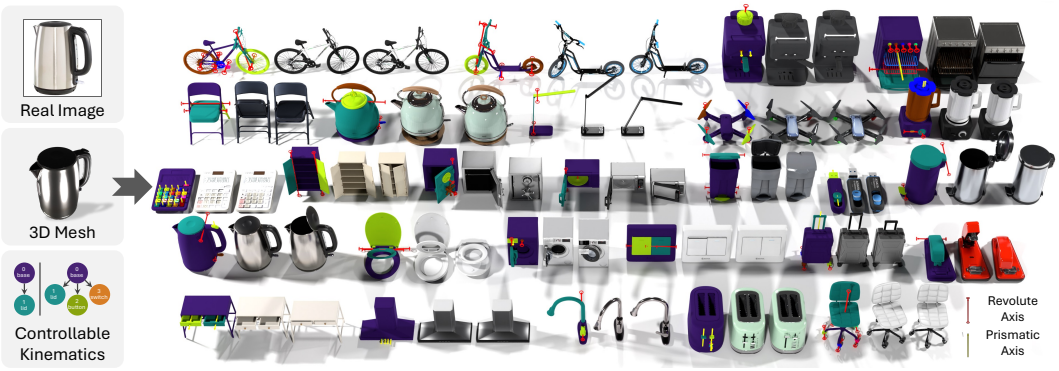

Figure 1: **Articulated 3D objects predicted by INSTRUCT-PARTICULATE from real-world images**. Our model infers articulated structures from static 3D assets, including outputs from off-the-shelf 3D generators, and supports optional kinematic prompting. This allows for generating diverse, realistic articulated 3D objects directly from real-world images. We encourage the reader to view the 3D results on the supplementary website.

Abstract

Reconstructing articulated 3D objects is important for animation, gaming, and robotic simulations. Recent neural networks can estimate the articulated structure of 3D objects, but their generalization remains limited by the scarcity of annotated data for this task. To address this gap, we introduce INSTRUCT-PARTICULATE, a model that takes a 3D mesh together with a target kinematic specification, including part descriptions, connectivity, joint types, and optional point prompts, and predicts the corresponding kinematic part segmentation and joint motion parameters. The kinematic specification disambiguates the task and allows the model to target annotations of different granularity, thereby making it possible to use more abundant heterogeneous training data. At test time, the kinematic specification can be obtained automatically from large-scale vision-language models, so the model can be applied to any input mesh. To train our model at scale, we construct a heterogeneous dataset of more than 150,000 articulated 3D objects, extending existing publicly available collections with data obtained by partially labelling other 3D models (monolithic or already decomposed into parts) with kinematic labels by means of vision-language models. Experiments show that our model generalizes better across categories and to AI-generated meshes, enabling articulated asset reconstruction from real-world images via image-to-3D models.

∗ Equal contribution.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

1 Introduction

Understanding and manipulating articulated objects is often necessary to interact with the physical world, and thus a key capability that physical agents must acquire. To this end, we consider the problem of reconstructing the articulated structure of a 3D object, decomposing it into parts with their articulation parameters (Fig. 1). Prior work on this problem [32, 35, 5, 26, 73] has so far relied only on small training datasets [79], thus missing the benefits of large-scale pre-training already demonstrated in text [4], image [15, 60, 59, 56], video [3, 67], and 3D [14, 13] understanding.

The lack of high-quality articulated 3D data remains a major bottleneck for further progress in this area. In particular, while recent works have increased the number of available articulated 3D models via procedural generation [23, 36, 5], part permutation [5, 35], and manual annotation [19, 6], the diversity of the resulting data is lacking. As a consequence, models trained on such data struggle to generalize to novel objects and categories, as we demonstrate in Section 5.

In this work, we introduce INSTRUCT-PARTICULATE, an approach that significantly boosts generalization in articulated object understanding. To achieve this, we first focus on scaling both data *size* and *diversity*. We are motivated by the observations that (1) there are several orders of magnitude more generic 3D assets [14, 13] than assets paired with articulation annotations, and (2) frontier vision-language models (VLMs) [54, 21, 2] possess strong 2D understanding of object articulation. We thus propose a category-agnostic data engine that uses the reasoning and generation capabilities of an off-the-shelf VLM [57] to pseudo-label (real or generated) 3D assets with articulated parts. Using this engine, we extract articulated part segmentations for 27k synthetic 3D assets of everyday objects spanning 432 categories. We further incorporate 120k curated 3D assets with generic part decompositions and 10k articulated 3D assets across 200 categories produced by a coding agent specialized for articulated 3D design [84].

This expanded data mixture improves the potential for generalization beyond prior work, but it also introduces a new challenge: often there is no single correct way to assign an articulated structure to a given 3D object, and greater data diversity therefore introduces inconsistencies in part granularity and semantics. A naive model would "average" over multiple plausible annotations and produce suboptimal results.

We address this problem by proposing a new formulation in which the model is *instructed* to extract a particular articulated structure, specified by an explicit kinematic structure (i.e., a list of parts and their connectivity), the joint types (e.g., revolute or prismatic), and optional 3D point prompts. All of these pieces of information, which can be provided manually or extracted automatically by a VLM at test time, disambiguate the target articulated structure and enable the model to predict crisp, coherent 3D part segmentations.

This new model is implemented using a scalable encoder-decoder architecture. Given a point cloud approximating the mesh together with kinematic prompts, the model predicts part labels for arbitrary surface query points and estimates the corresponding joint motion parameters. At test time, it infers the articulated structure of new objects efficiently, in a feed-forward manner. Furthermore, while we focus on the *analysis* of given 3D objects, we also consider *synthesizing* them from scratch by leveraging off-the-shelf 3D generators [65, 80] in combination with our model.

We summarize our contributions as follows: (1) We propose INSTRUCT-PARTICULATE, a state-of-the-art model that takes as input a static 3D object together with kinematic instructions, and outputs a corresponding fully articulated 3D object in a feed-forward manner; (2) We design a pipeline to pseudo-label a large library of 3D models with 3D articulated parts; (3) We show that our model enables the generation of diverse articulated 3D objects from real-world images, producing assets that are directly exportable to physics simulators.

2 Related Work

Reconstruction and generation of articulated 3D objects. Early approaches for reconstructing 3D articulated objects assume dense multi-view inputs and use test-time optimization [39, 61, 74, 52, 8, 46, 45], which makes them scale poorly. More recent approaches train feed-forward models that generate articulated 3D assets from few images [25, 11, 40, 5, 33, 6, 75, 71, 35, 44, 78, 26, 50], but their effectiveness is often constrained by limited training data. To improve generalization, several

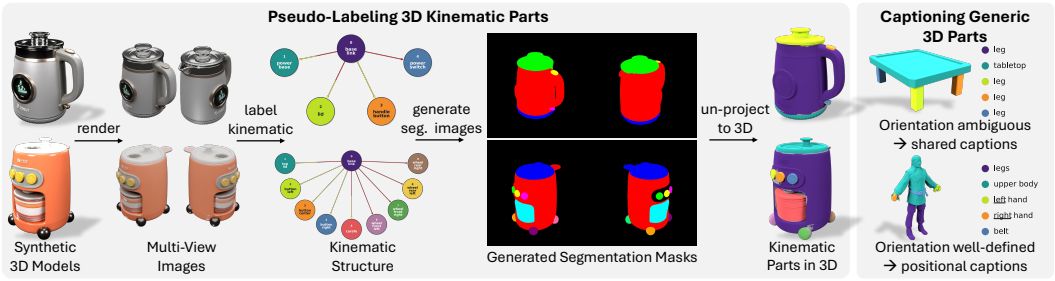

Figure 2: **Data pipelines of INSTRUCT-PARTICULATE**. *Left*: synthetic 3D assets are first rendered from multiple views. A vision-language model then extracts their kinematic structures and generates segmentation masks according to a fixed color scheme. The 2D segmentation results are then unprojected to 3D. Certain occluded regions are left unlabeled (visualized in gray). *Right*: for assets with existing part decompositions, we generate part captions that avoid arbitrary spatial labels when the object orientation is ambiguous, but use positional cues when a canonical orientation is available.

researchers have proposed to leverage foundation vision models [28, 31, 47] to infer plausible part decompositions and motion constraints. However, these methods often struggle with small parts and cannot predict multiple parts in parallel. In this work, inspired by [63], we scale the size and diversity of the training data, obtaining a model which is much more robust and general.

Feed-forward 3D part segmentation and articulation estimation. We are motivated by recent progress in 3D part segmentation, where state-of-the-art methods have moved beyond lifting masks from 2D foundation models such as SAM [24] and GLIP [27] in [42, 85, 81, 1, 64] toward data-driven models that operate directly in 3D [9, 10, 43, 48, 49, 82]. This shift has been enabled by increasingly large and diverse 3D datasets with part-level annotations. Several works extend these models to jointly predict articulation and rigging [76, 22, 34, 38, 16, 62, 32]. However, applicable datasets are dominated by humanoid and animal assets commonly used in games. As a result, these methods transfer poorly to other common objects, which are the focus of this work.

3D datasets. Existing 3D object datasets have been obtained by scanning real objects [18, 12, 77] or downloading manually-authored 3D assets from the web [7, 14, 13]. While some of these collections are large, they generally lack information about articulation. Some works exploit 'accidental metadata', such as the fact that manually-authored 3D assets are already organized into different components, to derive part annotations, but such decompositions rarely align with kinematic parts. Conversely, articulated 3D datasets [79, 41, 20, 70, 5, 53] are typically annotated by hand, which limits their scale and diversity. Procedural generation has also been used to expand articulated 3D asset collections [23, 36, 5]. However, these rule-based pipelines are difficult to scale to the long tail of real-world objects, leaving them practical only for a limited set of categories. In this work, we introduce complementary strategies for scaling this data.

3 Building a Large Dataset of Articulated 3D Objects

A key contribution of our paper is to build a large dataset of articulated 3D objects for training. To this end, we experiment with three complementary approaches. First, we build a VLM-based pipeline to partially label a large library of static 3D objects (Section 3.1). Second, for 3D objects that are already decomposed into parts, we adapt the VLM-based pipeline to assign textual captions to each part (Section 3.2). Third, we leverage a recent 3D coding agent to generate additional articulated objects (Section 3.3) with full supervision of their joint parameters.

Articulated 3D objects. We first specify what we mean by an articulated 3D object. An object is given by a 3D surface $M \subset \mathbb{R}^3$ (a mesh in practice). The mesh has P *kinematic parts* defined by a mapping $m : M \mapsto [P] = \{1, \dots, P\}$ that assigns each point $\mathbf{x} \in M$ of the surface to a part $m(\mathbf{x}) \in [P]$. Parts compose in a *kinematic structure* $\mathcal{K} := \{(u_e, v_e, \tau_e)\}_{e=1}^{P-1}$, which is a collection of $P-1$ joints where part $u_e \in [P]$ connects to part $v_e \in [P]$ with a joint of type $\tau_e \subset \{\text{pri}, \text{rev}\}$ (i.e., fixed, revolute, prismatic, or both). The kinematic structure forms a directed tree, and only

specifies the connectivity and motion types of the parts. We further associate to each movable joint e (i.e., $\tau_e \neq \varnothing$) an *axis* $\mathbf{a}_e = (\mathbf{d}_e, \mathbf{p}_e) \in \mathbb{S}^2 \times \mathbb{R}^3$ comprising a unit direction $\mathbf{d}_e$ and a pivot point $\mathbf{p}_e$ with motion bounds $[\theta_e^-, \theta_e^+] \subset \mathbb{R}$. A joint that is both revolute and prismatic can have separate axes for rotation and translation. An articulated 3D object is then defined as a tuple $(M, m, \mathcal{K}, \mathbf{a}, \theta)$ comprising a mesh, part assignments, kinematic structure, and joint parameters.

To specify instructions to our model, we further define the *kinematic condition* as $C := \left(\{(t_p, \mathbf{x}_p)\}_{p=1}^P, \mathcal{K}\right)$, where $\mathcal{K}$ is the kinematic structure defined above, and each part p is associated with a text prompt t_p and an optional point prompt $\mathbf{x}_p \in M$ such that $m(\mathbf{x}_p) = p$ (i.e., a 3D point that belongs to the part).

3.1 Pseudo-Labeling 3D Articulated Parts with Vision-Language Models

Our first approach to obtain articulated objects is to start with a large library of synthetic 3D models M and use an off-the-shelf vision-language model (VLM) to pseudo-label them with articulated part segmentations, obtaining a corresponding part segmentation map m, kinematic structure $\mathcal{K}$, part captions $\{t_p\}_{p=1}^P$, and point prompts $\{\mathbf{x}_p\}_{p=1}^P$ (which can be easily obtained by sampling m).

Given a 3D object M (i.e., a textured mesh), the first step is to extract 3D parts from it. While there are numerous 3D part segmentation methods [10, 9, 82, 43, 48] and datasets [51, 72, 17, 66], these focus on *semantic* part decompositions, which often do not correspond well to object articulation.

Instead, we use a VLM [21] to extract the names of articulated parts and their connectivity from multi-view renderings of each 3D object M, thus defining $\mathcal{K}$. We then prompt an image generator [57] to assign a different color to each kinematic part, thus obtaining an instance segmentation map for each view. Since the available parts are known a priori from the kinematic labeling, we instruct the model to use a fixed color scheme of our choice, which makes it easy to identify the parts in all views. We then obtain 2D segmentations by assigning each pixel to the part with the closest color according to the prompt. Finally, we project the 2D segmentations back to the 3D model to obtain the 3D articulated part segmentations m. The entire pipeline is illustrated in Fig. 2 (left).

We label AI-generated assets from HY3D-Bench-Synthetic [66] using this pipeline, excluding objects that are either fully rigid or exhibit soft, non-rigid deformations. This yields 27k synthetic 3D assets spanning 432 categories.

3.2 Augmenting Part-Segmented 3D Models

In addition to starting from static 3D models M, we also consider datasets like HY3D-Bench-Part-Level [66] that contain part-segmented objects (M, m). While the parts annotated in these datasets are not necessarily kinematic, we posit that they still provide useful supervision for aligning the part text prompts $\{t_p\}_{p=1}^P$ with the geometry M.

We obtain these prompts t_p by captioning each of the P parts using a VLM, following a pipeline similar to Section 3.1. The VLM is provided with a rendering of the textured 3D model and a segmentation map rendered from the same viewpoint, and outputs a set of candidate captions for each visible part. A key design choice is how to deal with semantically identical parts that are hard to distinguish with text alone (e.g., the four legs of a dining table). Inconsistent positional labeling (e.g., left vs. right, front vs. back) would likely confuse the model. As shown in Fig. 2 (right), we address this by carefully prompting the VLM to (1) use spatial cues only when the object has a meaningful canonical orientation (e.g., the humanoid character); or otherwise (2) output the exact same set of captions for all semantically identical parts (e.g., the table). For parts with identical captions, we therefore keep their point prompts $\mathbf{x}_p$ (i.e., no random dropout) during training, enabling the model to distinguish them geometrically.

When selecting assets from HY3D-Bench-Part-Level [66], we filter out assets with more than 10 visible parts, as dense segmentation maps make it difficult for the VLM to reliably distinguish colors and assign accurate captions. We caption the remaining 117k objects and include them in training.

3.3 Generating Articulated 3D Models with Coding Agents

The datasets above are labeled with part segments and part prompts $(M, m, \{t_p\}_{p=1}^P)$, but they do *not* contain the joint parameters $J = (\mathbf{a}, \theta)$. To supply our model with this supervision, we further

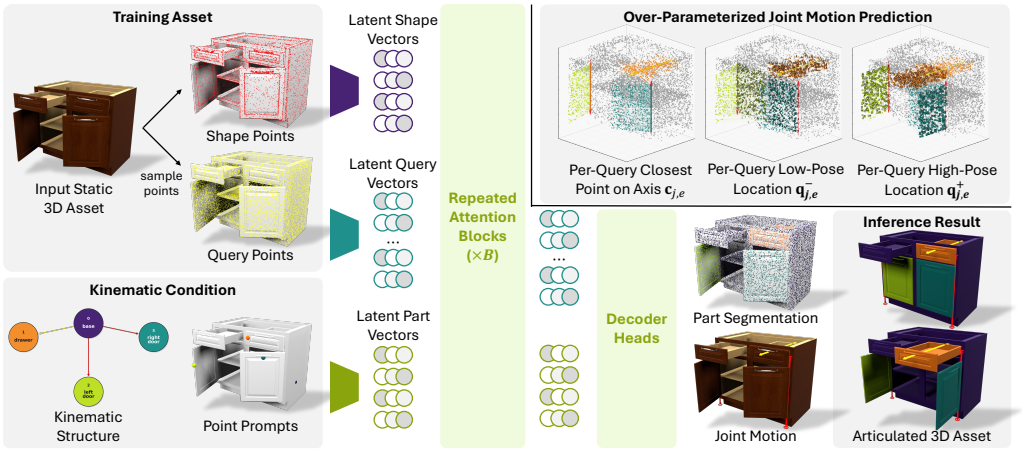

Figure 3: **Architectural overview of INSTRUCT-PARTICULATE.** Given a 3D mesh represented as a surface point cloud and a description of the desired kinematic structure, the model predicts part associations for arbitrary surface query points and motion parameters for the joints. Internally, shape, part, and query tokens are processed by B attention blocks before lightweight decoder heads recover segmentation and joint parameters. *Top-Right*: Joint motion is decoded from over-parameterized predictions of each query point's closest point on the joint axis and its locations at the joint limits.

consider 10k articulated 3D objects spanning 200 categories generated by Articraft [84], a recently introduced 3D coding agent.

4 Model Architecture

Having introduced our data, we now present the architecture of our model INSTRUCT-PARTICULATE. The goal of the model is to predict the articulated structure of a given 3D object following the directives given as an additional kinematic prompt. Specifically, the input is a mesh M together with a kinematic condition C that specifies the target kinematic tree $\mathcal{K}$ and the part prompts $\{(t_p, x_p)\}_{p=1}^{P}$. The output is a map m that assigns each point on the surface of M to a part, as well as the motion parameters $J = (\mathbf{a}, \theta)$ of the movable joints in $\mathcal{K}$.

Next, we describe the architecture, illustrated in Fig. 3. We begin by explaining how the model encodes the various inputs as tokens (Section 4.1), followed by the attention blocks used to process these tokens (Section 4.2), and finally the decoder heads that predict the part segmentation and joint motion parameters (Section 4.3).

4.1 Encoders

Shape Tokens. The model takes as input a shape M which, inspired by [43, 48, 83, 32], is represented by sampling N points $S := \{\mathbf{x}_i \in \mathbb{R}^3\}_{i=1}^{N} \subset M$ on it. As in [32], we encode each point with its associated surface normal $\mathbf{n}_i \in \mathbb{R}^3$ and feature vector $\mathbf{f}_i \in \mathbb{R}^d$ obtained using PartField [43], and sum them to obtain the point token $\tilde{\mathbf{x}}_i = \text{embed}(\mathbf{x}_i, \mathbf{n}_i, \mathbf{f}_i) = \phi_x(\mathbf{x}_i) + \phi_n(\mathbf{n}_i) + \phi_f(\mathbf{f}_i) \in \mathbb{R}^D$. Here, ϕ_x, ϕ_n, and ϕ_f are separate MLP embedders. Following VecSet [83], these N point tokens are then reduced to a much smaller number $L \ll N$ of *shape tokens* $\mathbf{Z}^S \in \mathbb{R}^{L \times D}$. This is achieved by letting a fixed set of learnable embeddings $\mathbf{Z}^0 \in \mathbb{R}^{L \times D}$ cross attend the point tokens $\{\tilde{\mathbf{x}}_i\}_{i=1}^{N}$, i.e., $\mathbf{Z}^S = \text{CrossAttn}\left(\mathbf{Z}^0, \{\tilde{\mathbf{x}}_i\}_{i=1}^{N}\right)$.

Point Query Tokens. In order to express the part segmentation, as well as some of the articulation parameters, we further consider additional *query points* $Q := \{\mathbf{q}_j \in \mathbb{R}^3\}_{j=1}^{K} \subset M$ sampled on the shape and embed them in the same way as before, namely $\tilde{\mathbf{q}}_j^0 = \text{embed}(\mathbf{q}_j, \mathbf{n}_j, \mathbf{f}_j)$.

Part Tokens. The model is also given a sequence of part descriptors $\{(t_p, \mathbf{x}_p)\}_{p=1}^{P}$, where t_p is a text prompt describing part p and $\mathbf{x}_p$ is an optional point that belongs to that part. We associate a

token $\tilde{\mathbf{l}}_p^0 \in \mathbb{R}^D$ to each part p by encoding the text prompt t_p with an MLP applied to its CLIP [56] embedding, and again reusing the point embedder $\text{embed}(\cdot)$ for the point prompt $\mathbf{x}_p$, so that $\tilde{\mathbf{l}}_p^0 = \phi_t(\text{CLIP}(t_p)) + \text{embed}(\mathbf{x}_p, \mathbf{n}_p, \mathbf{f}_p) \in \mathbb{R}^D$.

4.2 Attention

The shape, query, and part tokens are then processed by a transformer [69] using a custom attention pattern. The part tokens are meant to infer the articulation parameters of each part with respect to its parent. To this end, each of B transformer blocks updates the part tokens $\tilde{\mathbf{l}}_p$ via self-attention and by cross-attending the shape tokens $\mathbf{Z}^S$ to obtain the necessary information on the shape of the object:

$$\{\tilde{\mathbf{l}}_p^b\}_{p=1}^P = \text{SelfAttn}\left(\text{CrossAttn}(\{\tilde{\mathbf{l}}_p^{b-1}\}_{p=1}^P, \mathbf{Z}^S)\right), \quad b = 1, \ldots, B.$$

The query tokens, on the other hand, are meant to infer the association of each query surface point to the corresponding part. They are updated by cross-attending the shape tokens $\mathbf{Z}^S$ to obtain information on the object shape, and to the (updated) part tokens $\tilde{\mathbf{l}}_p$ to obtain information on the parts:

$$\{\tilde{\mathbf{q}}_j^b\}_{j=1}^K = \text{CrossAttn}\left(\text{CrossAttn}(\{\tilde{\mathbf{q}}_j^{b-1}\}_{j=1}^K, \mathbf{Z}^S), \{\tilde{\mathbf{l}}_p^b\}_{p=1}^P\right), \quad b = 1, \ldots, B.$$

Crucially, we do not use self-attention between query tokens, for speed, and to avoid making the result dependent on the number of query points decoded together [29].

4.3 Decoders

Once the final versions of the part $\tilde{\mathbf{l}}_p^B$ and query $\tilde{\mathbf{q}}_j^B$ tokens are obtained, decoder heads extract the part segmentation and joint motion parameters.

Part segmentation. For part segmentation, an MLP ψ_{part} scores each query-part pair, producing logits $\tilde{\mathbf{S}} \in \mathbb{R}^{K \times P}$ with $\tilde{\mathbf{S}}_{j,p} = \psi_{\text{part}}(\tilde{\mathbf{q}}_j^B, \tilde{\mathbf{l}}_p^B)$.

Motion parameters. For each movable joint $(u_e, v_e, \tau_e \neq \varnothing)$ in $\mathcal{K}$, we predict its axis of motion $\mathbf{a}_e = (\mathbf{d}_e, \mathbf{p}_e) \in \mathbb{S}^2 \times \mathbb{R}^3$, comprising a unit direction $\mathbf{d}_e$ and a pivot point $\mathbf{p}_e$, along with lower and upper bounds $[\theta_e^-, \theta_e^+] \subset \mathbb{R}$. Following [32], we predict these motion parameters in a per-query-point, over-parameterized manner. Concretely, for each query point $\mathbf{q}_j$ of the child part v_e and each allowed motion type $\tau \in \tau_e$, an MLP ψ_{joint} predicts an over-parameterized motion target

$$\left(\tilde{\mathbf{d}}_{j,e}^\tau, \tilde{\mathbf{c}}_{j,e}^\tau, \tilde{\mathbf{q}}_{j,e}^{\tau,-}, \tilde{\mathbf{q}}_{j,e}^{\tau,+}\right) = \psi_{\text{joint}}\left(\tilde{\mathbf{q}}_j^B, \tilde{\mathbf{l}}_{u_e}^B, \tilde{\mathbf{l}}_{v_e}^B, \mathbf{e}_\tau\right) \in \mathbb{S}^2 \times \mathbb{R}^3 \times \mathbb{R}^3 \times \mathbb{R}^3,$$

where $\mathbf{e}_\tau$ is a learnable embedding that encodes the motion type τ. The output tuple of the MLP consists of a local axis-direction estimate $\tilde{\mathbf{d}}_{j,e}^\tau$, the closest point $\tilde{\mathbf{c}}_{j,e}^\tau$ on the joint axis to $\mathbf{q}_j$, and the query point's limit-pose locations $\tilde{\mathbf{q}}_{j,e}^{\tau,-}$ and $\tilde{\mathbf{q}}_{j,e}^{\tau,+}$ obtained by articulating only this joint to its lower and upper limits. These quantities are visualized in Fig. 3. The decoder ψ_{joint} is trained in a teacher-forcing style: during training, query points are assigned to joints using their ground-truth part labels, while during inference we use the segmentation prediction $\tilde{s}_j = \arg\max_p \tilde{\mathbf{S}}_{j,p}$. Per-query results are aggregated by a geometric fitting step to recover one shared motion axis $\tilde{\mathbf{a}}_e$ and range $[\tilde{\theta}_e^-, \tilde{\theta}_e^+]$ for each joint in the kinematic tree $\mathcal{K}$. We refer the reader to Section A for more details.

4.4 Training

The model is trained end-to-end using a multi-task loss: $\mathcal{L} = \mathcal{L}_{\text{part}} + \mathcal{L}_{\text{joint}}$, where $\mathcal{L}_{\text{part}} = \frac{1}{K}\sum_{j=1}^K \text{CE}(\tilde{S}_j, m(\mathbf{q}_j))$ is the cross-entropy loss for the part segmentation, and $\mathcal{L}_{\text{joint}} = \frac{1}{P-1}\sum_{e=1}^{P-1}\sum_{\tau\in\tau_e}\sum_{m(\mathbf{q}_j)=v_e}\big(\lambda_d\|\tilde{\mathbf{d}}_{j,e}^\tau - \mathbf{d}_e^\tau\|_1 + \lambda_c\|\tilde{\mathbf{c}}_{j,e}^\tau - \mathbf{c}_{j,e}^\tau\|_1 + \lambda_q\|\tilde{\mathbf{q}}_{j,e}^{\tau,-} - \mathbf{q}_{j,e}^{\tau,-}\|_1 + \lambda_q\|\tilde{\mathbf{q}}_{j,e}^{\tau,+} - \mathbf{q}_{j,e}^{\tau,+}\|_1\big)$ supervises the over-parameterized joint targets.

5 Experiments

We organize our experiments around three questions: (1) Section 5.1: How does INSTRUCTPARTICULATE compare with existing methods for 3D articulation estimation and generation? (2)

Table 1: **Quantitative comparison**. We compare against 8 existing methods on the Lightwheel dataset [37]. For fairness, we evaluate INSTRUCT-PARTICULATE under the same input mode as each baseline. INSTRUCT-PARTICULATE substantially outperforms all baselines. 📷: image; 📦: mesh; 🌳: kinematic condition (where # means that the number of parts is given but not the full kinematic structure); 📍: point prompts. Colors: **best** and **second best**.

	Input				Part Match		Rest-Pose Seg.			Articulated Geom.			Joint Axes	
Method	📷	📦	🌳	📍	Prec.↑	Rec.↑	gIoU↑	PC↓	mIoU↑	gIoU↑	PC↓	OC↓	AE↓	LE↓
SINGAPO [40]	✓				37.6%	25.4%	−0.259	0.200	0.182	−0.263	0.295	0.030	30.3	0.077
PAct [44]	✓				50.6%	35.3%	−0.078	0.154	0.256	−0.084	0.201	0.019	37.5	0.120
PhysX-Anything [6]	✓				20.3%	19.2%	−0.458	0.247	0.093	−0.459	0.334	0.064	38.8	0.123
URDF-Anything+ [78]	✓				70.7%	23.9%	−0.133	0.259	0.260	−0.138	0.305	0.033	50.4	0.128
INSTRUCT-PARTICULATE	✓				73.4%	57.6%	0.177	0.102	0.405	0.164	0.161	0.015	18.1	0.139
Articulate AnyMesh [55]		✓			89.2%	42.5%	0.172	0.190	0.452	0.158	0.237	0.010	21.7	0.043
PartField [43]		✓	#		61.2%	62.6%	0.079	0.106	0.264	—	—	—	—	—
URDF-Anything+ [78]	(✓)	✓			77.5%	27.4%	−0.104	0.241	0.267	−0.110	0.298	0.026	49.9	0.111
Particulate [32]		✓			89.9%	51.5%	0.332	0.168	0.576	0.305	0.208	0.009	20.9	0.040
INSTRUCT-PARTICULATE		✓			94.3%	74.8%	0.583	0.091	0.724	0.542	0.108	0.004	13.9	0.018
P3SAM [48]		✓	#	✓	41.6%	79.7%	0.122	0.177	0.411	—	—	—	—	—
INSTRUCT-PARTICULATE		✓	✓	✓	97.3%	95.9%	0.799	0.010	0.821	0.747	0.039	0.006	9.5	0.015

Section 5.2: How much does each component of the curated data mixture, and each conditioning modality, contribute to performance? (3) Section 5.3: What capabilities does our model enable?

5.1 Comparisons

Baselines. We compare our model against eight existing methods with different input modalities: SINGAPO [40], PAct [44], PhysX-Anything [6], and URDF-Anything+ [78] take a single image as input and generate an articulated 3D object; Articulate AnyMesh [55] and Particulate [32] take a 3D mesh as input and predict its articulated structure; in addition, PartField [43] and P3SAM [48] are two feed-forward 3D part segmentation methods. Since URDF-Anything+ can optionally condition on a 3D mesh, we additionally compare against it using the ground-truth 3D object as input. PartField and P3SAM are given the exact number of ground-truth parts at test time.

Evaluation protocol. We evaluate INSTRUCT-PARTICULATE on the challenging Lightwheel dataset [37] introduced by [32], which contains 243 high-quality articulated objects spanning 14 categories, including categories absent from prior datasets, such as stand mixers, range hoods, and cooktop stoves. To align the conditioning setup with the baselines, we evaluate INSTRUCT-PARTICULATE under three input modes: (1) *Image Only*, which provides only a rendered image of the object; (2) *Mesh*, which provides only the ground-truth 3D mesh M; and (3) *Mesh + Kinematic*, which additionally provides the ground-truth kinematic structure $\mathcal{K}$, along with per-part text t_p and point prompts $\mathbf{x}_p$. In the *Image Only* setting, we first reconstruct a textured 3D mesh from the input image using the off-the-shelf 3D generator HY3D-3.1 [68]. For the *Image Only* and *Mesh* settings, we infer the kinematic condition C with a VLM [21], following a pipeline similar to Section 3.1. We also prompt the VLM to localize a 2D point on each identified kinematic part, and unproject these points onto the corresponding input mesh (generated for *Image Only*, ground-truth for *Mesh*) to construct the point prompts. Predicted and ground-truth parts have unknown correspondence and may differ in number (i.e., all baselines and our method in the *Image Only* and *Mesh* settings). We thus perform Hungarian matching between the two sets based on pairwise part-centroid distances.

Metrics. We organize the evaluation metrics into four groups. *Part Match* assesses whether the predicted parts match the ground-truth parts in both granularity and spatial layout, using *precision* and *recall*. We consider a matched pair of predicted and ground-truth parts to be valid if its generalized Intersection over Union [58] (gIoU) is at least 0. *Rest-Pose Segmentation* measures segmentation quality in the rest pose using part-wise *gIoU*, mean Intersection over Union (*mIoU*), and bidirectional Chamfer distance (*PC*), following [32]. *Articulated Geometry* evaluates the geometry after fully articulating the predicted asset by moving every movable joint to its (predicted) upper-limit pose. Following [32], we report part-wise *gIoU*, part-wise Chamfer distance (*PC*), and whole-object Chamfer distance (*OC*), where these metrics provide a heuristic assessment of both part segmentation

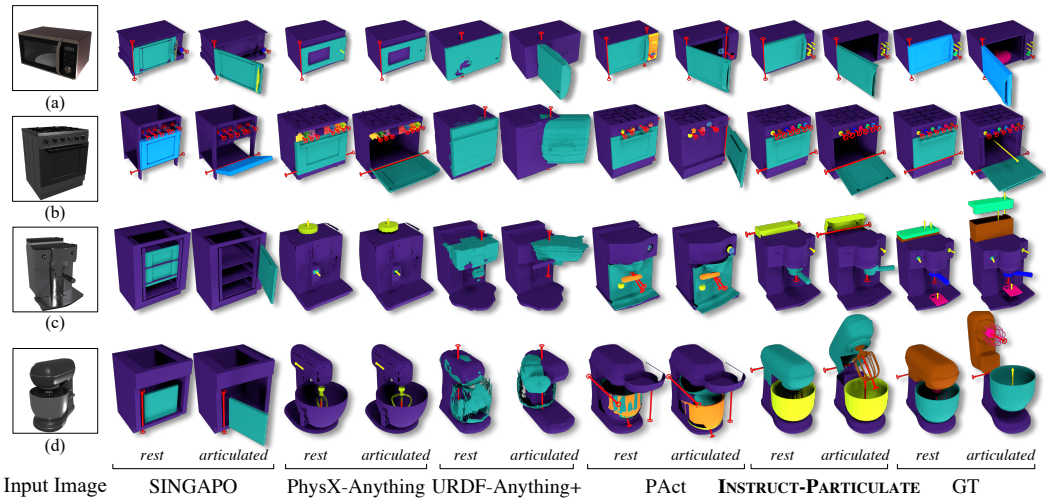

Figure 4: **Qualitative comparison** (*Image Only* mode). Given a single image as input, INSTRUCT-PARTICULATE (combined with an off-the-shelf 3D generator) can generate more realistic articulated 3D objects than the baselines.

Table 2: **Data ablations**. We incrementally add curated data sources from Section 3 (PM: PartNet-Mobility, GRS: GRScenes). The added sources yield complementary gains in part segmentation and joint motion estimation. Colors: **best** and **second best**.

	Training Data				Part Match		Rest-Pose Seg.			Articulated Geom.			Joint Axes	
	PM+GRS	S. 3.2	S. 3.1	S. 3.3	Prec.↑	Rec.↑	gIoU↑	PC↓	mIoU↑	gIoU↑	PC↓	OC↓	AE↓	LE↓
𝔸	✓				89.3%	71.2%	0.477	0.085	0.620	0.446	0.124	0.009	13.0	0.036
𝔹	✓	✓			95.3%	89.2%	0.692	0.032	0.743	0.652	0.070	0.008	12.3	0.025
ℂ	✓	✓	✓		96.8%	95.3%	0.791	0.009	0.813	0.745	0.049	0.008	13.1	0.020
𝔻	✓	✓	✓	✓	96.9%	95.4%	0.803	0.010	0.825	0.757	0.041	0.007	11.0	0.018

and joint motion estimation. Finally, *Joint Axes* directly measures joint-axis accuracy using angle error (*AE*) and location error (*LE*) between matched predicted and ground-truth joints.

Results. We report quantitative results in Table 1, where INSTRUCT-PARTICULATE consistently outperforms all baselines across all input settings. Figure 4 shows image-conditioned articulated 3D generation results from our approach and the baselines on the Lightwheel benchmark. Even for common categories, such as the microwave oven in (a) and the stove in (b), most baselines fail to reliably recover small articulated parts, including buttons and knobs. On more complex objects, such as the coffee machine in (c) and the stand mixer in (d), methods such as SINGAPO and URDF-Anything+ fail entirely. By contrast, INSTRUCT-PARTICULATE is robust to synthetic meshes, allowing us to offload geometry reconstruction to an off-the-shelf 3D generator, while recovering small articulated parts and estimating their joint motion accurately. We provide qualitative comparisons in the *Mesh* setting in Section B.

5.2 Ablation Studies

Data. We train separate models using identical hyperparameters on different data mixtures, starting from the existing articulated 3D datasets PartNet-Mobility [79] and GRScenes [70] used in [32], and incrementally adding the curated data sources described in Sections 3.1 to 3.3. In Table 2, we report the results obtained by the best checkpoint from each training run. Data generated by the coding agent with full joint parameter supervision (Section 3.3) mainly contributes to the *Joint Axes* metrics (ℂ vs. 𝔻), while the larger-scale part-segmented data from Sections 3.1 and 3.2 improves part segmentation quality (𝔸 vs. 𝔹 vs. ℂ). The model trained with only existing datasets (i.e., 𝔸) severely overfits to the available shapes and kinematic structures, and struggles to generalize to new categories.

Table 3: **Conditioning modality ablations**. We compare a model trained without kinematic conditioning ($\mathbb{A}$) with variants that disable point prompts x_p ($\mathbb{B}$) or text prompts t_p ($\mathbb{C}$) at inference time. Both prompt types improve performance, while kinematic conditioning disambiguates the target structure and enables learning from diverse annotations. 🔤: text prompts, 📍: point prompts. Colors: **best** and **second best**.

	Train	Inference		Part Match		Rest-Pose Seg.			Articulated Geom.			Joint Axes	
	C	🔤	📍	Prec.↑	Rec.↑	gIoU↑	PC↓	mIoU↑	gIoU↑	PC↓	OC↓	AE↓	LE↓
$\mathbb{A}$	×	—	—	64.1%	74.7%	0.249	0.123	0.463	0.226	0.163	0.011	22.9	0.048
$\mathbb{B}$	✓	✓		87.7%	66.5%	0.425	0.096	0.576	0.398	0.132	0.007	13.8	0.079
$\mathbb{C}$	✓		✓	95.6%	91.1%	0.626	0.028	0.663	0.594	0.064	0.007	18.8	0.035
$\mathbb{D}$	✓	✓	✓	96.9%	95.4%	0.803	0.010	0.825	0.757	0.041	0.007	11.0	0.018

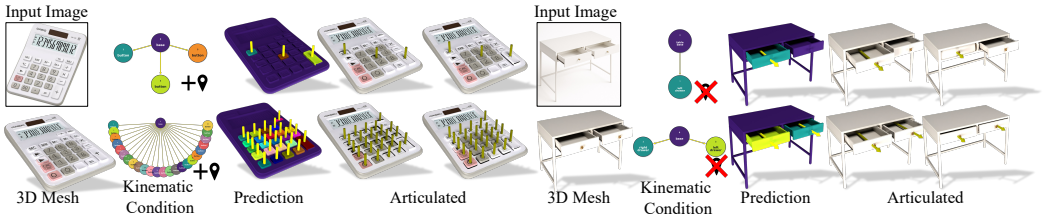

Figure 5: **Qualitative results with kinematic prompting**. We show articulated structures predicted by our model given different kinematic conditions. The model follows the specification faithfully.

Conditioning. We further ablate the conditioning modalities in Table 3. Using the same checkpoint trained on the full data mixture, we disable point prompts $\mathbf{x}_p$ ($\mathbb{B}$) or text prompts t_p ($\mathbb{C}$) at inference time. Both modalities provide useful context for articulation estimation, with point prompts providing the larger gain. We also train a separate model without kinematic conditioning ($\mathbb{A}$), using the architecture of [32] and supervising parts after Hungarian matching between predicted and ground-truth parts. Its much weaker performance suggests that, without explicit conditioning, variation in part granularity and semantics across datasets encourages the model to average over multiple plausible annotations and produce suboptimal results.

5.3 Articulated 3D Object Generation and Kinematic Prompting

Image-conditioned articulated 3D object generation. While our model takes existing 3D assets as input, it is robust to AI-generated meshes, and thus enables image-conditioned articulated 3D object generation via an off-the-shelf 3D generator. All results shown in Fig. 1 are obtained in this manner. This pipeline can support the creation of diverse simulation assets for embodied AI training.

Kinematic prompting. Beyond disambiguating outputs, kinematic conditioning also enables test-time prompting. In Fig. 5, we show predictions for the same input mesh under different kinematic conditions. The model follows these specifications faithfully, including a challenging case with 24 button joints. It also supports text-guided spatial control: for the desk example, the model receives no point prompts to distinguish the two drawers, but instead uses spatial text prompts (i.e., "left drawer" and "right drawer"). This reflects our data-curation effort to disambiguate spatial descriptors (Section 3.2), allowing reliable spatial reasoning when the object orientation is well defined.

6 Conclusions

We have presented INSTRUCT-PARTICULATE, a feed-forward model for recovering the articulated structure of a static 3D object, and endowed it with kinematic control. Our model is built on an efficient encoder-decoder architecture and is trained on more than 150k 3D objects, for which we developed a new data annotation engine that can label 3D articulated parts using a VLM. In this way, INSTRUCT-PARTICULATE generalizes much better to novel and more diverse objects than prior works. Furthermore, when paired with an off-the-shelf 3D generator, INSTRUCT-PARTICULATE provides a practical pipeline to generate articulated 3D assets directly from real-world images.

Acknowledgements

Ruining Li is supported by a Toshiba Research Studentship. Chuanxia Zheng is supported by NTU SUG-NAP and National Research Foundation, Singapore, under its NRF Fellowship Award NRF-NRFF17-2025-0009. Christian Rupprecht is supported by an Amazon Research Award and ERC StG Volute (grant no. 101222037). This work is partially supported by the UKRI AIRR programme (ID: u6en) and ERC CoG 101001212-UNION.

References

[1] Ahmed Abdelreheem, Ivan Skorokhodov, Maks Ovsjanikov, and Peter Wonka. Satr: Zero-shot semantic segmentation of 3d shapes. In *ICCV*, 2023. 3

[2] Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, Wenbin Ge, Zhifang Guo, Qidong Huang, Jie Huang, Fei Huang, Binyuan Hui, Shutong Jiang, Zhaohai Li, Mingsheng Li, Mei Li, Kaixin Li, Zicheng Lin, Junyang Lin, Xuejing Liu, Jiawei Liu, Chenglong Liu, Yang Liu, Dayiheng Liu, Shixuan Liu, Dunjie Lu, Ruilin Luo, Chenxu Lv, Rui Men, Lingchen Meng, Xuancheng Ren, Xingzhang Ren, Sibo Song, Yuchong Sun, Jun Tang, Jianhong Tu, Jianqiang Wan, Peng Wang, Pengfei Wang, Qiuyue Wang, Yuxuan Wang, Tianbao Xie, Yiheng Xu, Haiyang Xu, Jin Xu, Zhibo Yang, Mingkun Yang, Jianxin Yang, An Yang, Bowen Yu, Fei Zhang, Hang Zhang, Xi Zhang, Bo Zheng, Humen Zhong, Jingren Zhou, Fan Zhou, Jing Zhou, Yuanzhi Zhu, and Ke Zhu. Qwen3-VL technical report. *arXiv preprint arXiv:2511.21631*, 2025. 2

[3] Tim Brooks, Bill Peebles, Connor Holmes, Will DePue, Yufei Guo, Li Jing, David Schnurr, Joe Taylor, Troy Luhman, Eric Luhman, Clarence Ng, Ricky Wang, and Aditya Ramesh. Video generation models as world simulators. Technical report, OpenAI, 2024. 2

[4] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *NeurIPS*, 2020. 2

[5] Ziang Cao, Zhaoxi Chen, Liang Pan, and Ziwei Liu. PhysX-3D: Physical-grounded 3D asset generation. In *NeurIPS*, 2025. 2, 3

[6] Ziang Cao, Fangzhou Hong, Zhaoxi Chen, Liang Pan, and Ziwei Liu. Physx-anything: Simulation-ready physical 3d assets from single image. In *CVPR*, 2026. 2, 7

[7] Angel X. Chang, Thomas A. Funkhouser, Leonidas J. Guibas, Pat Hanrahan, Qi-Xing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. ShapeNet: An information-rich 3D model repository. *arXiv preprint arXiv:1512.03012*, 2015. 3

[8] Chuhao Chen, Isabella Liu, Xinyue Wei, Hao Su, and Minghua Liu. FreeArt3D: Training-free articulated object generation using 3d diffusion. In *SIGGRAPH Asia*, 2025. 2

[9] Minghao Chen, Roman Shapovalov, Iro Laina, Tom Monnier, Jianyuan Wang, David Novotny, and Andrea Vedaldi. PartGen: Part-level 3D generation and reconstruction with multi-view diffusion models. In *CVPR*, 2025. 3, 4

[10] Minghao Chen, Jianyuan Wang, Roman Shapovalov, Tom Monnier, Hyunyoung Jung, Dilin Wang, Rakesh Ranjan, Iro Laina, and Andrea Vedaldi. AutoPartGen: Autogressive 3D part generation and discovery. In *NeurIPS*, 2025. 3, 4

[11] Qiuyu Chen, Aaron Walsman, Marius Memmel, Kaichun Mo, Alex Fang, Dieter Fox, and Abhishek Gupta. URDFormer: A pipeline for constructing articulated simulation environments from real-world images. In *RSS*, 2024. 2

[12] Jasmine Collins, Shubham Goel, Kenan Deng, Achleshwar Luthra, Leon Xu, Erhan Gundogdu, Xi Zhang, Tomas F. Yago Vicente, Thomas Dideriksen, Himanshu Arora, Matthieu Guillaumin, and Jitendra Malik. ABO: Dataset and benchmarks for real-world 3D object understanding. In *CVPR*, 2022. 3

[13] Matt Deitke, Ruoshi Liu, Matthew Wallingford, Huong Ngo, Oscar Michel, Aditya Kusupati, Alan Fan, Christian Laforte, Vikram Voleti, Samir Yitzhak Gadre, Eli VanderBilt, Aniruddha Kembhavi, Carl Vondrick, Georgia Gkioxari, Kiana Ehsani, Ludwig Schmidt, and Ali Farhadi. Objaverse-XL: A universe of 10m+ 3d objects. In *NeurIPS Datasets and Benchmarks Track*, 2023. 2, 3

[14] Matt Deitke, Dustin Schwenk, Jordi Salvador, Luca Weihs, Oscar Michel, Eli VanderBilt, Ludwig Schmidt, Kiana Ehsani, Aniruddha Kembhavi, and Ali Farhadi. Objaverse: A universe of annotated 3D objects. In *CVPR*, 2023. 2, 3

[15] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Fei-Fei Li. ImageNet: A large-scale hierarchical image database. In *CVPR*, 2009. 2

[16] Yufan Deng, Yuhao Zhang, Chen Geng, Shangzhe Wu, and Jiajun Wu. Anymate: A dataset and baselines for learning 3d object rigging. In *SIGGRAPH*, 2025. 3

[17] Shaocong Dong, Lihe Ding, Xiao Chen, Yaokun Li, Yuxin Wang, Yucheng Wang, Qi Wang, Jaehyeok Kim, Chenjian Gao, Zhanpeng Huang, Zibin Wang, Tianfan Xue, and Dan Xu. From one to more: Contextual part latents for 3D generation. In *ICCV*, 2025. 4

[18] Laura Downs, Anthony Francis, Nate Koenig, Brandon Kinman, Ryan Hickman, Krista Reymann, Thomas B. McHugh, and Vincent Vanhoucke. Google Scanned Objects: A high-quality dataset of 3D scanned household items. In *ICRA*, 2022. 3

[19] Daoyi Gao, Yawar Siddiqui, Lei Li, and Angela Dai. MeshArt: Generating articulated meshes with structure-guided transformers. In *CVPR*, 2025. 2

[20] Haoran Geng, Helin Xu, Chengyang Zhao, Chao Xu, Li Yi, Siyuan Huang, and He Wang. GAPartNet: Cross-category domain-generalizable object perception and manipulation via generalizable and actionable parts. In *CVPR*, 2023. 3

[21] Google DeepMind. Gemini. https://deepmind.google/models/gemini/, 2026. Accessed: 2026-04-24. 2, 4, 7

[22] Tomas Jakab, Ruining Li, Shangzhe Wu, Christian Rupprecht, and Andrea Vedaldi. Farm3D: Learning articulated 3D animals by distilling 2D diffusion. In *3DV*, 2024. 3

[23] Abhishek Joshi, Beining Han, Jack Nugent, Yiming Zuo, Jonathan Liu, Hongyu Wen, Stamatis Alexandropoulos, Tao Sun, Alexander Raistrick, Gaowen Liu, Yi Shao, and Jia Deng. Infinigen-Sim: Procedural generation of articulated simulation assets. In *Procl. CoRL Workshop*, 2025. 2, 3

[24] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross Girshick. Segment anything. In *CVPR*, 2023. 3

[25] Jiahui Lei, Congyue Deng, William B Shen, Leonidas J Guibas, and Kostas Daniilidis. Nap: Neural 3d articulation prior. In *NeurIPS*, 2023. 2

[26] Haitian Li, Haozhe Xie, Junxiang Xu, Beichen Wen, Fangzhou Hong, and Ziwei Liu. MonoArt: Progressive structural reasoning for monocular articulated 3D reconstruction. In *ECCV*, 2026. 2

[27] Liunian Harold Li, Pengchuan Zhang, Haotian Zhang, Jianwei Yang, Chunyuan Li, Yiwu Zhong, Lijuan Wang, Lu Yuan, Lei Zhang, Jenq-Neng Hwang, Kai-Wei Chang, and Jianfeng Gao. Grounded language-image pre-training. In *CVPR*, 2022. 3

[28] Ruining Li, Chuanxia Zheng, Christian Rupprecht, and Andrea Vedaldi. DragAPart: Learning a part-level motion prior for articulated objects. In *ECCV*, 2024. 3

[29] Ruining Li, Gabrijel Boduljak, and Jensen (Jinghao) Zhou. On vanishing variance in transformer length generalization. In *SCOPE Workshop at ICLR 2025*, 2025. 6

[30] Ruining Li, Chuanxia Zheng, Christian Rupprecht, and Andrea Vedaldi. DSO: Aligning 3D generators with simulation feedback for physical soundness. In *ICCV*, 2025. 19

[31] Ruining Li, Chuanxia Zheng, Christian Rupprecht, and Andrea Vedaldi. Puppet-master: Scaling interactive video generation as a motion prior for part-level dynamics. In *ICCV*, 2025. 3

[32] Ruining Li, Yuxin Yao, Chuanxia Zheng, Christian Rupprecht, Joan Lasenby, Shangzhe Wu, and Andrea Vedaldi. Particulate: Feed-forward 3d object articulation. In *CVPR*, 2026. 2, 3, 5, 6, 7, 8, 9

[33] Zhe Li, Xiang Bai, Jieyu Zhang, Zhuangzhe Wu, Che Xu, Ying Li, Chengkai Hou, and Shanghang Zhang. URDF-Anything: Constructing articulated objects with 3D multimodal language model. In *NeurIPS*, 2025. 2

[34] Zizhang Li, Dor Litvak, Ruining Li, Yunzhi Zhang, Tomas Jakab, Christian Rupprecht, Shangzhe Wu, Andrea Vedaldi, and Jiajun Wu. Learning the 3D fauna of the Web. In *CVPR*, 2024. 3

[35] Zizhang Li, Cheng Zhang, Zhengqin Li, Henry Howard-Jenkins, Zhaoyang Lv, Chen Geng, Jiajun Wu, Richard Newcombe, Jakob Engel, and Zhao Dong. ART: Articulated reconstruction transformer. In *CVPR*, 2026. 2

[36] Xinyu Lian, Zichao Yu, Ruiming Liang, Yitong Wang, Li Ray Luo, Kaixu Chen, Yuanzhen Zhou, Qihong Tang, Xudong Xu, Zhaoyang Lyu, et al. Infinite mobility: Scalable high-fidelity synthesis of articulated objects via procedural generation. *arXiv preprint arXiv:2503.13424*, 2025. 2, 3

[37] Lightwheel. Simready: Simulation-ready 3d assets. https://simready.com/, 2025. Accessed: 2025. 7

[38] Isabella Liu, Zhan Xu, Yifan Wang, Hao Tan, Zexiang Xu, Xiaolong Wang, Hao Su, and Zifan Shi. RigAnything: Template-free autoregressive rigging for diverse 3d assets. *ACM TOG*, 2025. 3

[39] Jiayi Liu, Ali Mahdavi-Amiri, and Manolis Savva. PARIS: Part-level reconstruction and motion analysis for articulated objects. In *ICCV*, 2023. 2

[40] Jiayi Liu, Denys Iliash, Angel X Chang, Manolis Savva, and Ali Mahdavi Amiri. SINGAPO: Single image controlled generation of articulated parts in objects. In *ICLR*, 2025. 2, 7

[41] Liu Liu, Wenqiang Xu, Haoyuan Fu, Sucheng Qian, Qiaojun Yu, Yang Han, and Cewu Lu. AKB-48: A real-world articulated object knowledge base. In *CVPR*, 2022. 3

[42] Minghua Liu, Yinhao Zhu, Hong Cai, Shizhong Han, Zhan Ling, Fatih Porikli, and Hao Su. PartSLIP: low-shot part segmentation for 3D point clouds via pretrained image-language models. In *CVPR*, 2023. 3

[43] Minghua Liu, Mikaela Angelina Uy, Donglai Xiang, Hao Su, Sanja Fidler, Nicholas Sharp, and Jun Gao. PartField: Learning 3d feature fields for part segmentation and beyond. In *ICCV*, 2025. 3, 4, 5, 7

[44] Qingming Liu, Xinyue Yao, Shuyuan Zhang, Yueci Deng, Guiliang Liu, Zhen Liu, and Kui Jia. PAct: Part-decomposed single-view articulated object generation. *arXiv preprint arXiv:2602.14965*, 2026. 2, 7

[45] Yu Liu, Baoxiong Jia, Ruijie Lu, Chuyue Gan, Huayu Chen, Junfeng Ni, Song-Chun Zhu, and Siyuan Huang. VideoArtGS: Building digital twins of articulated objects from monocular video. *arXiv preprint arXiv:2509.17647*, 2025. 2

[46] Yu Liu, Baoxiong Jia, Ruijie Lu, Junfeng Ni, Song-Chun Zhu, and Siyuan Huang. Building interactable replicas of complex articulated objects via gaussian splatting. In *ICLR*, 2025. 2

[47] Ruijie Lu, Yu Liu, Jiaxiang Tang, Junfeng Ni, Yuxiang Wang, Diwen Wan, Gang Zeng, Yixin Chen, and Siyuan Huang. Generating objects with part-articulation from a single image. In *SIGGRAPH Asia*. ACM, 2025. 3

[48] Changfeng Ma, Yang Li, Xinhao Yan, Jiachen Xu, Yunhan Yang, Chunshi Wang, Zibo Zhao, Zhuo Chen, Yanwen Guo, and Chunchao Guo. P^3-SAM: Native 3D part segmentation. In *ECCV*, 2026. 3, 4, 5, 7

[49] Ziqi Ma, Yisong Yue, and Georgia Gkioxari. Find any part in 3d. In *ICCV*, 2025. 3

[50] Zhao Mandi, Yijia Weng, Dominik Bauer, and Shuran Song. Real2Code: Reconstruct articulated objects via code generation. In *ICLR*, 2025. 2

[51] Kaichun Mo, Shilin Zhu, Angel X. Chang, Li Yi, Subarna Tripathi, Leonidas J. Guibas, and Hao Su. PartNet: A large-scale benchmark for fine-grained and hierarchical part-level 3D object understanding. In *CVPR*, 2019. 4

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

[53] Soroush Nasiriany, Sepehr Nasiriany, Abhiram Maddukuri, and Yuke Zhu. RoboCasa365: A large-scale simulation framework for training and benchmarking generalist robots. In *ICLR*, 2026. 3

[54] OpenAI. Introducing GPT-5.4, 2026. URL https://openai.com/index/introducing-gpt-5-4/. 2

[55] Xiaowen Qiu, Jincheng Yang, Yian Wang, Zhehuan Chen, Yufei Wang, Tsun-Hsuan Wang, Zhou Xian, and Chuang Gan. Articulate AnyMesh: Open-vocabulary 3d articulated objects modeling. In *CoRL*, 2025. 7

[56] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *ICML*, 2021. 2, 6

[57] Naina Raisinghani. Introducing Nano Banana Pro. https://blog.google/innovation-and-ai/products/nano-banana-pro/, 2025. Accessed: 2026-04-24. 2, 4

[58] Hamid Rezatofighi, Nathan Tsoi, JunYoung Gwak, Amir Sadeghian, Ian Reid, and Silvio Savarese. Generalized intersection over union: A metric and a loss for bounding box regression. In *CVPR*, 2019. 7

[59] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, 2022. 2

[60] Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, Patrick Schramowski, Srivatsa Kundurthy, Katherine Crowson, Ludwig Schmidt, Robert Kaczmarczyk, and Jenia Jitsev. LAION-5B: An open large-scale dataset for training next generation image-text models. In *NeurIPS Datasets and Benchmarks Track*, 2022. 2

[61] Chaoyue Song, Jiacheng Wei, Chuan Sheng Foo, Guosheng Lin, and Fayao Liu. Reacto: Reconstructing articulated objects from a single video. In *CVPR*, 2024. 2

[62] Chaoyue Song, Jianfeng Zhang, Xiu Li, Fan Yang, Yiwen Chen, Zhongcong Xu, Jun Hao Liew, Xiaoyang Guo, Fayao Liu, Jiashi Feng, and Guosheng Lin. MagicArticulate: Make your 3d models articulation-ready. In *CVPR*, 2025. 3

[63] Richard S. Sutton. The bitter lesson, 2019. URL http://www.incompleteideas.net/IncIdeas/BitterLesson.html. 3

[64] George Tang, William Zhao, Logan Ford, David Benhaim, and Paul Zhang. Segment any mesh: Zero-shot mesh part segmentation via lifting segment anything 2 to 3d. *arXiv preprint arXiv:2408.13679*, 2024. 3

[65] Tencent Hunyuan3D Team. Hunyuan3d 2.5: Towards high-fidelity 3d assets generation with ultimate details, 2025. URL https://arxiv.org/abs/2506.16504. 2

[66] Team Hunyuan3D, Bowen Zhang, Chunchao Guo, Dongyuan Guo, Haolin Liu, Hongyu Yan, Huiwen Shi, Jiaao Yu, Jiachen Xu, Jingwei Huang, Kunhong Li, Lifu Wang, Linus, Penghao Wang, Qingxiang Lin, Ruining Tang, Xianghui Yang, Yang Li, Yirui Guan, Yunfei Zhao, Yunhan Yang, Zeqiang Lai, Zhihao Liang, and Zibo Zhao. HY3D-Bench: Generation of 3D assets. *arXiv preprint arXiv:2602.03907*, 2026. 4, 15

[67] Team Seedance, De Chen, Liyang Chen, Xin Chen, Ying Chen, Zhuo Chen, Zhuowei Chen, Feng Cheng, Tianheng Cheng, Yufeng Cheng, et al. Seedance 2.0: Advancing video generation for world complexity. *arXiv preprint arXiv:2604.14148*, 2026. 2

[68] Tencent Hunyuan3D Team. Hunyuan3D 3.1. https://3d.hunyuan.tencent.com/, 2026. Accessed: 2026-05-01. 7

[69] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017. 6

[70] Hanqing Wang, Jiahe Chen, Wensi Huang, Qingwei Ben, Tai Wang, Boyu Mi, Tao Huang, Siheng Zhao, Yilun Chen, Sizhe Yang, et al. GRUtopia: Dream general robots in a city at scale. *arXiv preprint arXiv:2407.10943*, 2024. 3, 8, 15

[71] Jiawei Wang, Dingyou Wang, Jiaming Hu, Qixuan Zhang, Jingyi Yu, and Lan Xu. Kinematify: Open-vocabulary synthesis of high-dof articulated objects. In *ICRA*, 2026. 2

[72] Penghao Wang, Yiyang He, Xin Lv, Yukai Zhou, Lan Xu, Jingyi Yu, and Jiayuan Gu. PartNeXt: A next-generation dataset for fine-grained and hierarchical 3D part understanding. In *NeurIPS Datasets and Benchmarks Track*, 2025. 4

[73] Penghao Wang, Siyuan Xie, Hongyu Yan, Xianghui Yang, Jingwei Huang, Chunchao Guo, and Jiayuan Gu. ArtLLM: Generating articulated assets via 3D LLM. In *CVPR*, 2026. 2

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

[75] Ruiqi Wu, Xinjie Wang, Liu Liu, Chunle Guo, Jiaxiong Qiu, Chongyi Li, Lichao Huang, Zhizhong Su, and Ming-Ming Cheng. DIPO: Dual-state images controlled articulated object generation powered by diverse data. In *NeurIPS*, 2025. 2

[76] Shangzhe Wu, Ruining Li, Tomas Jakab, Christian Rupprecht, and Andrea Vedaldi. MagicPony: Learning articulated 3D animals in the wild. In *CVPR*, 2023. 3

[77] Tong Wu, Jiarui Zhang, Xiao Fu, Yuxin Wang, Jiawei Ren, Liang Pan, Wayne Wu, Lei Yang, Jiaqi Wang, Chen Qian, Dahua Lin, and Ziwei Liu. OmniObject3D: Large-vocabulary 3D object dataset for realistic perception, reconstruction and generation. In *CVPR*, 2023. 3

[78] Zhuangzhe Wu, Yue Xin, Chengkai Hou, Minghao Chen, Yaoxu Lyu, Jieyu Zhang, and Shanghang Zhang. URDF-Anything+: End-to-end generation for simulation-ready articulated assets. *arXiv preprint arXiv:2603.14010*, 2026. 2, 7

[79] Fanbo Xiang, Yuzhe Qin, Kaichun Mo, Yikuan Xia, Hao Zhu, Fangchen Liu, Minghua Liu, Hanxiao Jiang, Yifu Yuan, He Wang, Li Yi, Angel X. Chang, Leonidas J. Guibas, and Hao Su. SAPIEN: A simulated part-based interactive environment. In *CVPR*, 2020. 2, 3, 8, 15

[80] Jianfeng Xiang, Xiaoxue Chen, Sicheng Xu, Ruicheng Wang, Zelong Lv, Yu Deng, Hongyuan Zhu, Yue Dong, Hao Zhao, Nicholas Jing Yuan, and Jiaolong Yang. Native and compact structured latents for 3D generation. In *CVPR*, 2026. 2

[81] Yuheng Xue, Nenglun Chen, Jun Liu, and Wenyun Sun. ZeroPS: High-quality cross-modal knowledge transfer for zero-shot 3d part segmentation. In *3DV*, 2025. 3

[82] Yunhan Yang, Yukun Huang, Yuan-Chen Guo, Liangjun Lu, Xiaoyang Wu, Edmund Y Lam, Yan-Pei Cao, and Xihui Liu. SAMPart3D: Segment any part in 3d objects. *arXiv preprint arXiv:2411.07184*, 2024. 3, 4

[83] Biao Zhang, Jiapeng Tang, Matthias Niessner, and Peter Wonka. 3DShape2VecSet: A 3d shape representation for neural fields and generative diffusion models. *ACM TOG*, 2023. 5

[84] Matt Zhou, Ruining Li, Xiaoyang Lyu, Zhaomou Song, Zhening Huang, Chuanxia Zheng, Christian Rupprecht, Andrea Vedaldi, and Shangzhe Wu. Articraft: An agentic system for scalable articulated 3d asset generation. In *NeurIPS*, 2026. 2, 5, 15

[85] Yuchen Zhou, Jiayuan Gu, Xuanlin Li, Minghua Liu, Yunhao Fang, and Hao Su. PartSLIP++: enhancing low-shot 3d part segmentation via multi-view instance segmentation and maximum likelihood estimation. *arXiv preprint arXiv:2312.03015*, 2023. 3

A Implementation Details

A.1 Training Details

Data mixture. Our final INSTRUCT-PARTICULATE model is trained on the available articulated 3D datasets PartNet-Mobility [79] and GRScenes [70], together with the curated data sources introduced in Sections 3.1 to 3.3. The training data mixture is determined empirically and is summarized in Table 4.

Table 4: **Training data mixture** of INSTRUCT-PARTICULATE.

Source	Type	Scale	Weight
PartNet-Mobility [79]	Existing articulated assets	2.3k	0.17
GRScenes [70]	Existing articulated assets	1.8k	0.10
HY3D-Bench-Part-Level [66]	Generic part-segmented assets	117k	0.18
HY3D-Bench-Synthetic [66]	Synthetic assets pseudo-labeled with kinematic parts	27k	0.22
Articraft [84]	Agent-generated articulated assets	10k	0.33

Data augmentation. During training, we sample a random articulated state for each training iteration. All input shapes are first normalized to $[-0.5, 0.5]^3$, and then rotated around the z-axis by a random angle sampled from $\{0, \pi/2, \pi, 3\pi/2\}$. We further apply a random scale factor drawn from $\mathcal{U}(0.95, 1.05)$ and a random translation vector from $\mathcal{N}(0, 0.05)^3$. We randomly remove normals with a probability of 0.3. For each training iteration, we also randomly merge each part to its parent part in the kinematic tree with a probability of 0.15. The text prompt t_p is randomly removed with a probability of 0.2 independently for each part in the (augmented) kinematic tree $\mathcal{K}$, while the point prompt x_p is randomly removed with a probability of 0.25. We make sure the point prompt and text prompt are not both removed for the same part.

Hyperparameters. The main training hyperparameters are summarized in Table 5.

Table 5: **Training configuration** of INSTRUCT-PARTICULATE.

Hyperparameter name	Value
Model	
Number of attention blocks B	6
Number of shape points N	10,240
Number of latent shape tokens L	2,048
Model dimension D	768
Loss weights	
λ_d	5
λ_c	5
λ_q	10
Optimization	
Optimizer	AdamW
Peak learning rate	6.4×10^{-5}
Warmup steps	1,000
Global batch size	128
Number of query points decoded together M	4,096
Training steps	100,000
Training compute	16 NVIDIA Blackwell GPUs
Weight decay	0.01

A.2 Inference Details

Part segmentation inference. We perform inference with a much denser query point cloud ($102,400$ points vs. $4,096$ points during training) to ensure sufficient coverage of the object surface. We obtain the part labels from the predicted logits as $\tilde{s}_j = \arg\max_p \tilde{\mathbf{S}}_{j,p}$.

Optimizing joint motion constraints from over-parameterized targets. Once we obtain the per-query-point over-parameterized joint motion targets $\tilde{\mathbf{d}}^{\tau}_{j,e}$, $\tilde{\mathbf{c}}^{\tau}_{j,e}$, $\tilde{\mathbf{q}}^{\tau,-}_{j,e}$, and $\tilde{\mathbf{q}}^{\tau,+}_{j,e}$ from the decoder, we run an optimization to recover the joint axis $\mathbf{a}_e = (\mathbf{d}_e, \mathbf{p}_e)$ and range $[\theta^-_e, \theta^+_e]$ for each joint e. Specifically, we first aggregate each query point's "vote" for the joint direction, $\tilde{\mathbf{d}}^{\tau}_{j,e}$, and pivot point, $\tilde{\mathbf{c}}^{\tau}_{j,e}$, to obtain the joint axis $(\frac{1}{|\mathcal{Q}(e)|}\sum_{j\in\mathcal{Q}(e)} \tilde{\mathbf{d}}^{\tau}_{j,e}, \frac{1}{|\mathcal{Q}(e)|}\sum_{j\in\mathcal{Q}(e)} \tilde{\mathbf{c}}^{\tau}_{j,e})$, where $\mathcal{Q}(e) = \{j \mid \tilde{s}_j = v_e\}$ is the set of query points predicted to belong to the child part v_e. We then solve the motion bounds $[\theta^-_e, \theta^+_e]$ as $\arg\min_{\theta^-_e,\theta^+_e} \sum_{j\in\mathcal{Q}(e)} \left\|\tilde{\mathbf{q}}^{\tau,-}_{j,e} - F(\mathbf{q}_j, \mathbf{a}_e, \theta^-_e, \tau_e)\right\|_2 + \left\|\tilde{\mathbf{q}}^{\tau,+}_{j,e} - F(\mathbf{q}_j, \mathbf{a}_e, \theta^+_e, \tau_e)\right\|_2$, where $F(\mathbf{q}_j, \mathbf{a}_e, \theta_e, \tau_e)$ is the forward kinematics function that computes the location of a query point $\mathbf{q}_j$ given the joint axis $\mathbf{a}_e$ and motion bounds $[\theta^-_e, \theta^+_e]$. Empirically, we find that first fitting the axis based on $\tilde{\mathbf{d}}^{\tau}_{j,e}$ and $\tilde{\mathbf{c}}^{\tau}_{j,e}$ yields better results than performing global optimization over both $\mathbf{a}_e$ and $[\theta^-_e, \theta^+_e]$.

A.3 Prompt for Kinematic Prediction

We provide the full prompt used for kinematic prediction in Listing 1.

Listing 1: Full Prompt.

```
## ROLE
You are an expert **Simulation Designer** specializing in Isaac Sim URDF and USD asset
     creation. Your capability is converting 2D visual data of real-world objects into
     high-fidelity, simulation-ready kinematic descriptions.

## OBJECTIVE
The user will provide several **Images** of an object and, optionally, a brief **Text
     Description**.
Your task is to analyze the visual structure of the object and decompose it into a
     **Kinematic Tree** of rigid bodies ("Links") connected by specific physics constraints
     ("Joints").

## CRITICAL RULES

### 1. INTERACTABLE VS. STRUCTURAL (The "Functionality" Rule)
* **Functional Small Parts:** You **MUST** identify parts as separate links if they move
     independently and are used for daily interaction (e.g., buttons, rotating knobs,
     flipping lids, switches).
* **Structural Fasteners:** You **MUST MERGE** parts that are purely structural or
     fasteners. Screws, rivets, bolts, and internal locking latches should be merged with
     their parent link.

### 2. SINGLE LINK OBJECTS
* If the object is a solid, static item with **NO independently moving parts** (e.g., a
     simple plush toy, a solid ceramic mug, a seamless plastic block):
    * **Total Links:** 1
    * **Links List:** Contains only Link 0 (Base Link).
    * **Kinematic Tree:** Must be an **EMPTY LIST** `[]`.

### 3. MULTIPLE INSTANCES = SEPARATE LINKS
* If the object contains multiple instances of the same semantic class (e.g., 3 drawers, 2
     cabinet doors), you **MUST** treat them as **separate links** with unique IDs.
* **Condition:** Only group them into a single link if they are mechanically fused.

### 4. NON-RIGID PARTS (Wires, Cables, Cloth)
* If the object contains significant non-rigid components (e.g., dangling wires, hoses),
     include them as separate links.
* **Flagging:** Set `"is_rigid": false` for these links. Set `"is_rigid": true` for all
     others.

### 5. IDENTIFY THE BASE LINK
* The first link (Link 0) **MUST** be the "Base Link" (static root, frame, chassis).

### 6. INTRINSIC LOCALIZATION
* **NEVER** use image coordinates. **ALWAYS** use object-intrinsic landmarks (e.g., "Next to
     the handle").

### 7. PART IDENTITY VS. ARTICULATION STATE
* When describing a link, focus on the stable identity of the part.
* Do **NOT** define or name a part by its current articulation state (for example "open
     door", "closed drawer", or "extended tray").
* Any state description should avoid articulation-state wording unless absolutely necessary
     for disambiguation.

## JOINT TYPE DEFINITIONS
Assign one of the following types to every edge in the kinematic tree:
* **Revolute:** Rotates around an axis.
* **Prismatic:** Slides along a linear axis.

## VERIFICATION WORKFLOW
Before generating the final JSON, perform these internal checks:
1.  **JSON Validity:** Ensure strictly valid JSON.
2.  **One Link Check:** If `total_links` is 1, verify `kinematic_tree` is empty `[]`.
```

```
3.  **No Fixed Joints:** Scan `joint_type`. If "Fixed", merge the link.
4.  **Instance Check:** Did you split multiple drawers into unique links?
5.  **Rigidity Check:** Did you mark wires as `false`?
6.  **Tree Topology:** Ensure valid structure (No loops, no orphans).
7.  **Continuous ID Labeling:** Ensure that the `link_id` and `joint_id` are labeled
     continuously from 0.

## OUTPUT FORMAT
You must respond with **ONLY** a valid, parseable JSON object. Do not include markdown
     formatting or conversational filler. The JSON must adhere to this schema:

{
  "object_name": "Generic high-level category (less than 10 words) in English.",
  "total_links": "Integer count",
  "links": [
    {
      "link_id": 0,
      "name": "base_link",
      "is_rigid": true,
      "description": {
        "semantics": "What is this part?",
        "appearance": "Material, color, texture details",
        "geometry": "Shape approximation",
        "functionality": "Its role",
        "state": "Current visual state",
        "intrinsic_location": "Location relative to the whole object"
      }
    },
    ...
  ],
  "joints": [
    {
      "joint_id": 0,
      "parent_link_id": 0,
      "child_link_id": 1,
      "joint_type": "prismatic | revolute",
      "joint_description": "Explanation of movement"
    },
    ...
  ]
}

## INPUT PROCESSING INSTRUCTIONS
1.  Analyze the image to find the largest static mass. **This is Link 0.**
2.  Scan for interactable components (knobs, drawers, buttons). Ask: "Does this move
     relative to the base?" -> Create Links.
3.  Scan for non-rigid components (wires, hoses, cloth). -> Create Links (`is_rigid: false`).
4.  Apply **Instance Split**: If you see multiple similar items (e.g., 3 drawers), assign a
     unique Link ID to *each one*.
5.  **Single Link Check:** If no independent moving parts exist (static object), stop here.
     Set `total_links` to 1 and ensure `kinematic_tree` is empty `[]`.
6.  Determine motion types for all identified links (Slide? Rotate?).
7.  **RUN VERIFICATION WORKFLOW** (Check for fixed joints, topology errors, continuous
     labeling, and valid JSON).
8.  Generate the JSON.
```

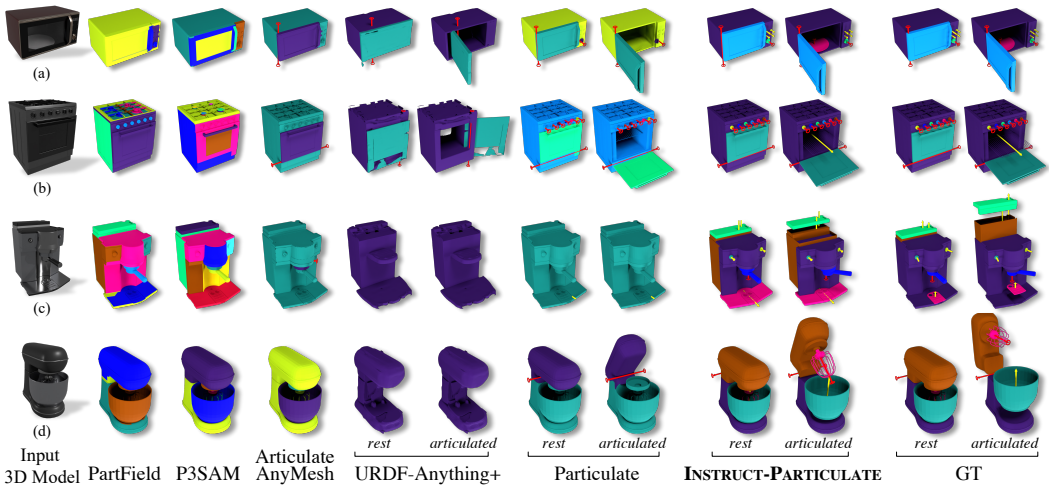

Figure 6: **Qualitative comparison** (*Mesh* mode). Given a 3D mesh as input, INSTRUCT-PARTICULATE (combined with a VLM that labels the kinematic structure) can more reliably segment small and internal parts than the baselines, while generalizing to more complex objects.

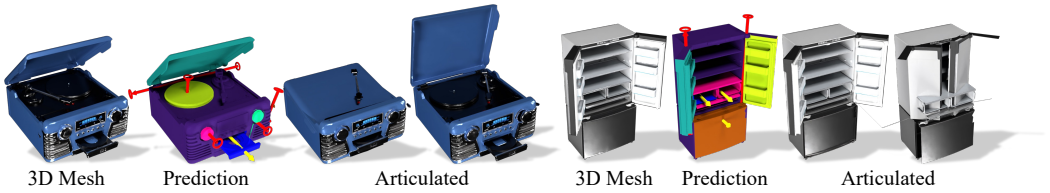

Figure 7: **Failure cases** of INSTRUCT-PARTICULATE.

B Additional Results

B.1 Additional Qualitative Comparisons

In Fig. 6, we present additional qualitative comparisons in the *Mesh* setting, where each method takes an artist-created 3D mesh from the Lightwheel benchmark as input. PartField and P3SAM are designed for *semantic* part segmentation, whose part definitions often do not align with the *articulated* parts required for kinematic reasoning. While baseline methods can produce plausible results on common categories, such as the microwave oven in (a) and the stove in (b), they generalize less reliably to more complex objects, such as the coffee machine in (c) and the stand mixer in (d), and often miss small and internal parts. By contrast, INSTRUCT-PARTICULATE reliably segments them, including the microwave oven's rotating plate and the buttons and knobs of the stove and coffee machine, while also estimating their joint motion accurately.

B.2 Failure Cases and Limitations

Failure cases. We present representative failure cases in Fig. 7. While our model is generally robust to AI-generated meshes, segmentation artifacts can still occur, such as on the right knob of the CD player. Because each joint axis is estimated by aggregating votes from query points on the predicted part, such local errors can propagate to joint-motion estimation. Artifacts in the generated meshes can also degrade prediction quality, as shown by the floating component in the CD player and the missing part separation in the refrigerator.

Limitations. While INSTRUCT-PARTICULATE can support large-scale creation of simulation assets, its outputs are *not* yet simulation-ready. They lack physical properties, and AI-generated meshes may contain incomplete geometry or excessive face counts. Improving the simulation readiness of these assets (e.g., via post-training [30]), remains future work.

C Societal Impacts

INSTRUCT-PARTICULATE can have positive societal impacts by reducing the cost of producing articulated 3D assets for animation, games, education, robotic simulation, and research on assistive physical agents. By improving feed-forward articulation estimation from static assets and image-derived meshes, the method may also reduce the manual annotation effort required to build diverse embodied-AI datasets. At the same time, the same capability could be misused to create more realistic synthetic environments or articulated objects for deceptive visual content, unauthorized replication of proprietary assets, or unsafe embodied-agent training. Incorrect articulation predictions may also propagate to downstream simulators or robots and cause unreliable behavior in safety-critical manipulation settings. We therefore view INSTRUCT-PARTICULATE primarily as a research tool, and recommend that practical deployments validate predictions in the target domain, respect dataset and object licensing and privacy constraints, and apply appropriate access controls or monitoring when models or generated assets are used in higher-risk settings.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction clearly state the contribution is primarily a new model for feed-forward 3D articulation estimation.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The paper includes a separate "Limitations" section.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

Justification: The paper does not include theoretical results.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We describe in detail how the datasets used to train our model are prepared and their exact mixture ratios. We also provide full architectural details of our model and the training hyperparameters. We will make the model accessible to the community.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: The inference code, pretrained model, and interactive demo are publicly available at https://huggingface.co/spaces/rayli/instruct-particulate. The training data are assembled from publicly available source datasets; we describe the data mixture and preparation pipeline in the paper and appendix.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

Answer: [Yes]

Justification: We provide full details of our training settings, including the data splits and hyperparameters.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [No]

Justification: Our main contribution is a new model trained on large-scale datasets. Due to the large amount of compute resources required, performing multiple training runs to obtain error bars is infeasible.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We provide the GPU workers required for model training in the paper.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: The paper conforms with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: This is discussed in the Appendix.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.

- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper poses no such risks.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We use the open-source assets from PartNet-Mobility, GRScenes, and HY3D-Bench in accordance with their respective licenses. We have also properly cited the original papers.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.

- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: The paper does not release new assets.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [Yes]

    Justification: We use LLMs, or more specifically, VLMs, to pseudo-label our data. The details are described in the paper.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.