PARTICULATE: Feed-Forward 3D Object Articulation

Ruining Li[1*] Yuxin Yao[2*] Chuanxia Zheng[1,3] Christian Rupprecht[1]
Joan Lasenby[2†] Shangzhe Wu[2†] Andrea Vedaldi[1†]

[1]University of Oxford [2]University of Cambridge [3]Nanyang Technological University

https://ruiningli.com/particulate

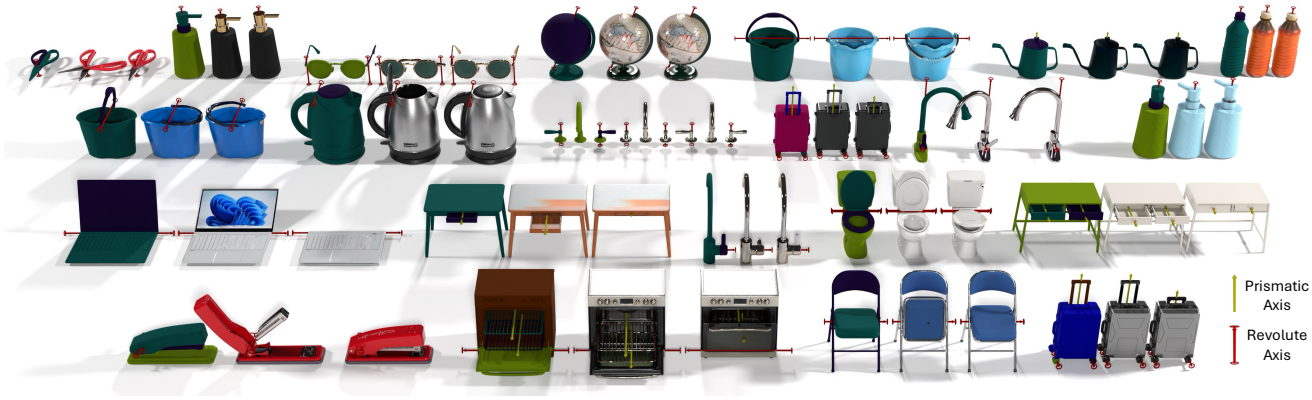

Figure 1. **Articulated 3D objects predicted by PARTICULATE.** Our model infers articulated structures directly from static 3D meshes in a single feed-forward pass, enabling fast inference across diverse objects, including those synthesized by 3D generative models.

Abstract

We introduce PARTICULATE*, a feed-forward model that, given a 3D mesh of an object, infers its articulations, including its 3D parts, their kinematic structure, and the motion constraints. The model is based on a transformer network, the Part Articulation Transformer, which predicts all these parameters for all joints. We train the network end-to-end on a diverse collection of articulated 3D assets from public datasets. During inference,* PARTICULATE *maps the output of the network back to the input mesh, yielding a fully articulated 3D model in seconds, much faster than prior approaches that require per-object optimization.* PARTICULATE *also works on AI-generated 3D assets, enabling the generation of articulated 3D objects from a single (real or synthetic) image when combined with an off-the-shelf image-to-3D model. We further introduce a new challenging benchmark for 3D articulation estimation curated from high-quality public 3D assets, and redesign the evaluation protocol to be more consistent with human preferences. Empirically,* PARTICULATE *significantly outperforms state-of-the-art approaches.*

*Equal contribution. Correspondence to ruining@robots.ox.ac.uk and yy561@cam.ac.uk. †Equal advising.

1. Introduction

Most objects are defined not only by their shape, but also by their ability to move and deform. Cabinets, for example, have doors and drawers that open through rotational and translational motion, constrained by hinges and sliding tracks. Humans can effortlessly understand the *articulated structure* of such objects. Such capabilities are needed to allow robots to manipulate everyday objects [22, 47, 64], and can simplify the creation of interactive digital twins for gaming and simulation [17, 60].

In this paper, we consider the problem of estimating the articulated structure of a single static 3D mesh. Prior work has attempted to model the rich variety of articulated objects via procedural generation [28, 44]. However, scaling such rule-based approaches to the long tail of real-world objects remains extremely challenging. We posit that methods that can *learn* articulations from large collections of 3D assets offer greater potential for generality.

Several authors have already proposed learning-based approaches for closely related tasks. These works [34–36, 58] focus on learning part segmentation on 3D point clouds, but primarily predict *semantic* part segmentation on *static* objects, without modeling the articulations between parts. Other learning-based methods [11, 21, 32, 33] aim to

generate 3D articulated objects directly. While promising, these models are typically trained on only a few object categories, assume key attributes such as the kinematic structure are known a priori, and often rely on part retrieval to assemble fully articulated 3D assets, which limits their ability to produce accurate and diverse articulated objects.

Seeking to address these limitations, we introduce PARTICULATE, a learning-based framework that takes an input 3D mesh and outputs its articulated structure. PARTICULATE predicts a full set of articulation attributes, including *articulated* 3D part segmentation, the kinematic structure, and the parameters of the articulated motion. It does so in a single forward pass, taking only seconds. We focus on *analyzing* existing 3D assets rather than *synthesizing* them from scratch due to the rapid advances in 3D generative models [7, 52–54, 65]. By focusing on the *complementary* task of articulation prediction, PARTICULATE can leverage increasingly high-fidelity 3D generators to enable one-stop creation of articulated 3D objects with realistic geometry and appearance (Fig. 1). The resulting assets can be seamlessly imported into physics engines for simulation.

PARTICULATE builds on the Part Articulation Transformer, a flexible and scalable network that takes as input a point cloud approximating the mesh, a representation applicable to virtually any 3D shape. The network consists of standard attention blocks paired with multiple decoder heads to predict different articulation attributes. This design allows us to train the network on a diverse collection of articulated 3D assets from public datasets, including PartNet-Mobility [64] and GRScenes [56]. To further enhance generalization to novel objects, we augment the raw point cloud with 3D semantic part features obtained from PartField [35] as input to the network. The resulting model predicts the articulated structure of new objects in arbitrary articulated poses, including those generated by off-the-shelf 3D generators (*e.g.*, [54]), in a feed-forward manner.

To evaluate our model, we introduce a new challenging benchmark dataset of 243 high-quality 3D assets with accurate articulation annotations, crafted by Lightwheel under the CC-BY-NC license [29]. We also establish a comprehensive evaluation protocol for the 3D articulation estimation task, with new metrics that more faithfully reflect prediction quality than those used in prior work [33, 43].

In summary, our key contributions are: (1) PARTICULATE, a feed-forward model that, given the 3D mesh of an object, infers its full articulated structure, including articulated part segmentation, kinematic structure, and motion parameters, which can then be directly exported to physics simulators. (2) We show that PARTICULATE generalizes well to unseen objects, including AI-generated 3D models. (3) We release a challenging benchmark for articulation estimation. Experiments show PARTICULATE's significant gains via metrics that are consistent with human preferences.

Method	Input: text	Input: image	Input: kinematic chain	Input: 3D	Output: bounding box	Output: 3D	Output: part segmentation	Output: kinematic chain	Output: motion axis	Output: motion range	Output: 1^+	Inference time
PARIS [31]		✓				✓	✓	✓	✓	✓		N/A
NAP [21]					✓	(✓)*	✓	✓	✓	✓	✓	N/A
MeshArt [11]		$(✓)^o$		$(✓)^o$	✓	✓	✓		✓	✓	✓	N/A
ArtFormer [50]	✓	✓			✓	✓	✓	✓	✓	✓	✓	N/A
ArtiLatent [5]	$(✓)^o$	✓				✓	✓	✓	✓	✓	✓	~30 sec
CAGE [32]	✓		✓		✓	(✓)*	✓	✓	✓		✓	N/A
SINGAPO [33]		✓			✓	(✓)*	✓	✓	✓	✓	✓	~10 sec
Articulate-Anything [20]	$(✓)^o$	$(✓)^o$				(✓)*	✓	✓	✓	✓		~10 min
GEOPARD [13]				$✓^\dagger$		✓		✓	✓		✓	N/A
DreamArt [38]		✓				✓	✓		✓	✓		N/A
FreeArt3D [4]		✓	✓			✓	✓	✓	✓			~10 min
Kinematify [57]	$(✓)^o$	$(✓)^o$				✓	✓	✓	✓		✓	~20 min
Articulate AnyMesh [43]				✓		✓	✓	✓	✓		✓	~15 min
PARTICULATE (ours)				✓		✓	✓	✓	✓	✓	✓	~10 sec

Table 1. **Related work overview on partly-rigid articulated object modeling.** We summarize existing methods' required inputs and outputs, and their inference time for a single object. Inputs include text: text, image: image or multi-view images, kinematic chain: kinematic chain, **3D**: 3D geometry (*e.g.*, mesh or point cloud). Outputs include bounding box: 3D bounding box, **3D**: 3D geometry (*e.g.*, mesh or point cloud), part segmentation: part segmentation, kinematic chain: kinematic chain, motion axis: motion axis, motion range: motion range, $\mathbf{1}^+$: supports multi-joint articulation. SINGAPO [33], NAP [21], and CAGE [32] perform part-based retrieval from a 3D part database to obtain articulated 3D objects (✓*). MeshArt [11], ArtiLatent [5], Articulate-Anything [20], and Kinematify [57] all perform unconditional generation and optionally take additional inputs for conditional generation such as texts, images, and 3D point clouds ($✓^o$). GEOPARD [13] further requires segmented 3D point clouds as input ($✓^\dagger$). N/A indicates the inference time is not reported in the original paper.

2. Related Work

3D part segmentation. Recent 3D part segmentation methods increasingly target zero-shot settings [1, 34, 51, 66, 69]. Most approaches leverage 2D foundation models such as SAM [19] and GLIP [23] by rendering 3D assets into multiple views and lifting the resulting 2D masks back to 3D. However, these lift-from-2D approaches, which infer 3D structure from visible surface masks in rendered views, are inherently limited by view coverage and struggle to recover occluded or internal parts.

To alleviate these issues, recent work learns native 3D part segmentation models [35, 39, 40, 67]. These models are trained on large datasets of 3D objects with (pseudo) ground-truth part segmentation, and hence acquire open-world capabilities. However, they mostly predict *semantic* parts, which often are *not* meaningful for articulation. By contrast, our PARTICULATE jointly predicts *articulated* parts with their kinematic structure and motion constraints, producing fully articulated 3D objects from static meshes.

Articulated object modeling. Prior work has explored reconstructing articulated objects from multi-view images

across different articulation states. A common paradigm is per-instance optimization using neural radiance fields (NeRFs [41]) [31, 42, 48, 59, 60, 63] or Gaussian splatting (GS [18]) [37, 61]. These methods capture object-specific articulation well but are slow and require densely sampled, posed images that are difficult to acquire in practice.

To improve scalability, generative models of articulated objects train diffusion [14] or autoregressive models on 3D datasets with annotated articulations [12, 64] [11, 21, 32, 33, 50]. However, they are usually trained on only a few categories, assume known attributes such as the kinematic structure, and often rely on part retrieval to assemble articulated assets.

More recent methods leverage foundation models and are training-free or require only light fine-tuning. DragA-Part [24] and DreamArt [38] fine-tune pre-trained 2D generators to synthesize articulation videos [26]; FreeArt3D [4] probes a pre-trained 3D generator [65]; and Articulate-Anything [20] and Articulate AnyMesh [43] prompt vision-language models (VLMs) to reason about part articulation. Despite their broad generalization across categories, these methods still suffer from slow per-object inference, difficulty with internal or subtle parts, and limited multi-joint support. Our problem is also related to automatic rigging [10, 16, 27, 30, 49, 62]; however, these methods focus primarily on characters, humanoids, animals, or animation-centric assets, and do not directly target the partly-rigid everyday objects we study.

In contrast, we propose a data-driven approach that utilizes a flexible and scalable network to infer all articulation attributes in a single feed-forward pass, enabling recovery of articulated objects in seconds rather than hours and achieving superior performance across diverse categories.

3. Method

We introduce PARTICULATE, a feed-forward approach that predicts the articulated structure, including the articulated parts, kinematic tree, and motion constraints, from a single static 3D mesh of an object. We formulate the problem in Sec. 3.1, followed by details of our network architecture in Sec. 3.2 and of the decoder heads in Sec. 3.3. We then describe training and inference in Secs. 3.4 and 3.5.

3.1. Problem Definition

Given a 3D mesh $\mathcal{M} = (V, F)$ with vertices V and faces F, our goal is to predict its (partly rigid) *articulated structure* $\mathcal{A}$. This specifies (1) the segmentation S of the mesh faces F into multiple articulated parts; (2) the kinematic tree K of the parts; and (3) the rigid motion constraints M of the joints. Formally, $\mathcal{A}$ is a 4-tuple (P, S, K, M), where $P \in \mathbb{N}$ is the number of parts and $S : [|F|] \to [P]$ assigns each face to a part, segmenting the mesh $\mathcal{M}$ into P articulated parts[1]. The kinematic tree K is a collection of edges $K \subseteq [P] \times [P]$. While kinematic trees are undirected by default, for most objects, it is possible and convenient to define a base part (*e.g.*, the casing for a microwave). We can then orient the edges from the base part outward. With this, $(p, c) \in K$ indicates a joint connecting part c with its parent part p. We assume a single base part b for each object, and hence K forms an arborescence rooted at b.

The motion constraint M is parameterized by $(M_{\text{tp}}, M_{\text{pd}}, M_{\text{ra}}, M_{\text{pr}}, M_{\text{rr}})$, where each component M_* specifies one aspect of the rigid motion of each part relative to its parent. Specifically, $M_{\text{tp}} : [P] \to \{\text{fixed}, \text{pri}, \text{rev}, \text{both}\}$ tells the **type** of motion: fixed, prismatic (allowing linear sliding), revolute (allowing rotation around a single axis), or both. $M_{\text{pd}} : [P] \to \mathbb{S}^2$ gives the direction of each part p's prismatic motion[2] (*i.e.*, the **p**rismatic **d**irections). $M_{\text{ra}} : [P] \to \mathbb{S}^2 \times \mathbb{R}^3$ specifies the rotation axis of each part's revolute motion (*i.e.*, the **r**evolute **a**xes). Both M_{pd} and M_{ra} are defined in the mesh $\mathcal{M}$'s coordinate system. $M_{\text{pr}} : [P] \to \mathbb{R}^2$ and $M_{\text{rr}} : [P] \to \mathbb{R}^2$ define the range $[-l_{\min}, l_{\max}]$ and $[-\theta_{\min}, \theta_{\max}]$ of each part's **p**rismatic **r**anges and **r**evolute motion **r**anges with respect to the mesh $\mathcal{M}$'s articulation state.

The parameterization (P, S, K, M) of $\mathcal{A}$ captures all articulation attributes, and can be easily converted to the Universal Robot Description Format (URDF), allowing PARTICULATE's predictions to be used in physics simulators.

3.2. Part Articulation Transformer

Unlike per-shape optimization methods [38, 43, 57], which rely on handcrafted heuristics or distil 2D priors from vision-language foundation models (VLMs), we train a *feed-forward* network f_θ, dubbed Part Articulation Transformer, on a repository of diverse articulated 3D assets to directly predict the articulated structure $\mathcal{A} = f_\theta(\mathcal{M})$ from an input mesh $\mathcal{M}$. This makes inference fast and better handles small or internal parts that are difficult to view externally and thus difficult to handle by a VLM. Because of these advantages, we can train on *all* articulated objects in existing public datasets. These span multiple categories and lead to a model more generalizable than prior works [32, 33] that only handled a few similar categories.

To this end, following recent works on 3D deep learning on meshes [35, 39, 68], our model operates on a point cloud $\mathcal{P} = \{\mathbf{p}_i \in \mathbb{R}^3\}_{i=1}^N$. f_θ processes $\mathcal{P}$, together with a set of learnable part queries, to produce latent point and part vectors using multiple attention blocks. The latent vec-

[1] Note $[n] \coloneqq \{1, 2, \ldots, n\}$ is the set of positive integers up to n.

[2] Strictly speaking, M_{pd} is only defined on the parts with prismatic motion, *i.e.*, $\{p \in [P] : M_{\text{tp}}(p) \in \{\text{pri}, \text{both}\}\}$. We annotate its domain as $[P]$ for simplicity (similar for M_{ra}, M_{pr} and M_{rr} below).

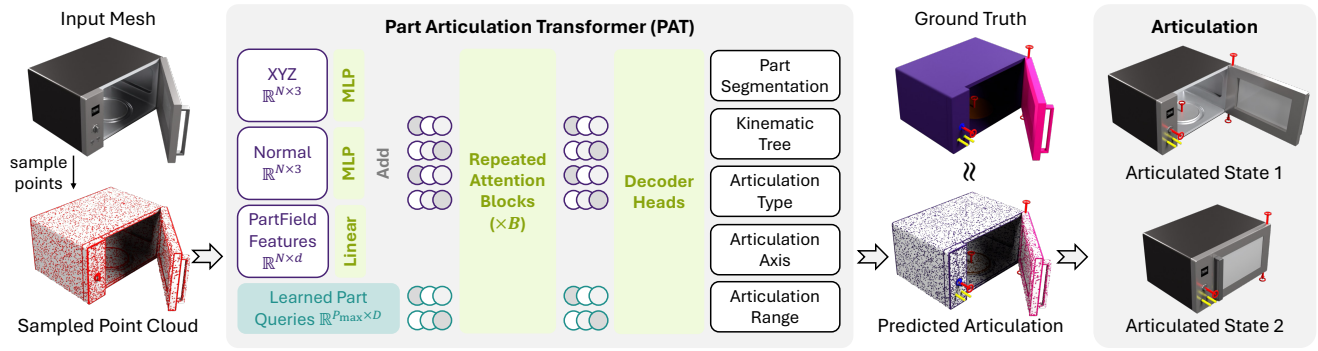

Figure 2. **Overview of PARTICULATE.** Our model consumes a point cloud sampled from the input mesh and predicts its articulated structure. The backbone consists of B standard attention blocks that operate on the point vectors and a set of learnable part query vectors. The resulting latent vectors are then processed by specialized decoder heads dedicated to different components of the articulated structure. The entire model is trained end-to-end in a fully supervised manner. At inference time, PARTICULATE projects the point-level segmentation predictions back onto the original mesh to reconstruct a complete articulated 3D model.

tors are later consumed by the decoder heads (Sec. 3.3) to predict the articulation attributes S, K and M. The overall architecture is illustrated in Fig. 2.

Inputs. For each point $\mathbf{p}_i \in \mathcal{P}$, we also provide f_θ 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 [35]. The normal captures further geometric information, and PartField, which is trained to extract 2D *semantic* parts, captures information. We use separate MLPs to map these per-point inputs $\mathbf{p}_i$, $\mathbf{n}_i$ and $\mathbf{f}_i$ to vectors of equal dimension D and sum them to obtain each point's token $\tilde{\mathbf{p}}_i \in \mathbb{R}^D$.

Part queries. Since the number P of articulated parts in $\mathcal{M}$ is unknown a priori, inspired by [3], we initialize a set of $P_{\max}$ learnable part queries $\mathcal{Q} = \{\mathbf{q}_j \in \mathbb{R}^D\}_{j=1}^{P_{\max}}$, where $P_{\max}$ is set to be much larger than P in a typical mesh $\mathcal{M}$.

Attention blocks. The backbone of f_θ is a standard transformer [55]. Inspired by [2], we apply a sequence of self-attention and cross-attention modules between the point tokens $\{\tilde{\mathbf{p}}_i\}_{i=1}^N$ and the part tokens $\{\mathbf{q}_j\}_{j=1}^{P_{\max}}$. Specifically, f_θ consists of B attention blocks alternating query self-attention and query-to-point cross-attention. Due to their large number $N \gg P_{\max}$, we do *not* apply self-attention across the point tokens $\{\tilde{\mathbf{p}}_i\}_{i=1}^N$. This way we can operate on dense point clouds while maintaining a small memory footprint. The attention blocks output the processed point tokens $\{\tilde{\mathbf{p}}_i\}_{i=1}^N$ and part tokens $\{\tilde{\mathbf{q}}_j\}_{j=1}^{P_{\max}}$.

3.3. Decoder Heads

We use several prediction heads to map point and part tokens $\{\tilde{\mathbf{p}}_i\}_{i=1}^N$ and $\{\tilde{\mathbf{q}}_j\}_{j=1}^{P_{\max}}$ to part segmentation, kinematic structure and motion constraints. Note that the number $P_{\max}$ of part tokens is larger than the number of actual parts P. We will later show how to match the $P_{\max}$ part tokens to the P ground-truth parts for supervision during training (Sec. 3.4) and predict articulated structures $\mathcal{A}$ with $P < P_{\max}$ parts during inference (Sec. 3.5). Next, we describe these decoder heads in detail.

Part segmentation. f_θ predicts a matrix of logits $\tilde{\mathbf{S}} \in \mathbb{R}^{N\times P_{\max}}$ telling which points belong to which parts using an MLP h_S as $\tilde{\mathbf{S}}_{i,j} = h_S(\tilde{\mathbf{p}}_i, \tilde{\mathbf{q}}_j)$.

Kinematic tree. The MLP h_K takes a pair of part tokens $(\tilde{\mathbf{q}}_i, \tilde{\mathbf{q}}_j)$ and outputs the log-probability that part i is the parent of part j, outputting a soft adjacency matrix $\tilde{\mathbf{K}} \in \mathbb{R}^{P_{\max}\times P_{\max}}$ that captures the kinematic tree:

$$\tilde{\mathbf{K}}_{i,j} := \log \mathbb{P}[i \text{ is the parent of } j] = h_K(\tilde{\mathbf{q}}_i, \tilde{\mathbf{q}}_j). \quad (1)$$

Motion types, motion ranges & prismatic directions. Separate MLPs h_{tp}, h_{pr}, h_{rr} and h_{pd} take individual part tokens $\tilde{\mathbf{q}}_i$ to predict the motion type, prismatic range, revolute range and prismatic direction of the corresponding part:

$$\tilde{\mathbf{M}}^i_{\text{tp}} = h_{\text{tp}}(\tilde{\mathbf{q}}_i) \in \mathbb{R}^4, \quad (2)$$
$$\tilde{\mathbf{M}}^i_{\text{pr}} = h_{\text{pr}}(\tilde{\mathbf{q}}_i) \in \mathbb{R}^2, \quad (3)$$
$$\tilde{\mathbf{M}}^i_{\text{rr}} = h_{\text{rr}}(\tilde{\mathbf{q}}_i) \in \mathbb{R}^2, \quad (4)$$
$$\tilde{\mathbf{M}}^i_{\text{pd}} = h_{\text{pd}}(\tilde{\mathbf{q}}_i)/\|h_{\text{pd}}(\tilde{\mathbf{q}}_i)\|_2 \in \mathbb{S}^2. \quad (5)$$

Here $\tilde{\mathbf{M}}^i_{\text{tp}} \in \mathbb{R}^4$ are the logits of query part i having one of the *four* motion types and $h_{\text{pd}}(\tilde{\mathbf{q}}_i) \in \mathbb{R}^3$ is normalized to a unit direction vector.

Over-parameterization of revolute axes. Next, we consider predicting each part's revolute axis $\tilde{\mathbf{M}}^i_{\text{ra}} := (\tilde{\mathbf{d}}^i_{\text{ra}}, \tilde{\mathbf{x}}^i_{\text{ra}}) \in \mathbb{S}^2 \times \mathbb{R}^3$ where $\tilde{\mathbf{d}}^i_{\text{ra}}$ denotes the direction and $\tilde{\mathbf{x}}^i_{\text{ra}}$ a fixed point on the axis. We could use another MLP to do so, but found empirically that this would result in slightly shifted axes.[3]

[3]Following [21], we did so by using Plücker coordinates. We hypothesize that this is an overfitting problem, as the size of our training data is modest. The location of the axis $\tilde{\mathbf{x}}^i_{\text{ra}}$ is much more difficult to learn than the direction $\tilde{\mathbf{d}}^i_{\text{ra}}$, probably because the latter is often axis aligned.

Because this problem primarily affects location, we do use an MLP h_{rd} to predict (only) the direction of the revolute axis from the part token $\tilde{\mathbf{q}}_i$ as:

$$\tilde{\mathbf{d}}^i_{\text{ra}} = h_{\text{rd}}(\tilde{\mathbf{q}}_i)/\|h_{\text{rd}}(\tilde{\mathbf{q}}_i)\|_2. \quad (6)$$

However, we *over-parameterize* the location by letting each 3D point $\mathbf{p}_j$ that belongs to the part *vote* for the location of the axis, using an MLP that takes both the point token $\tilde{\mathbf{p}}_j$ and the part token $\tilde{\mathbf{q}}_i$ as input, and maps the point $\mathbf{p}_j$ to its orthogonal projection on the axis:

$$\tilde{\mathbf{x}}^i_j := \underset{\mathbf{x} \text{ on the revolute axis } \tilde{\mathbf{M}}^i_{\text{ra}}}{\arg\min} \|\mathbf{x} - \mathbf{p}_j\|_2 = h_{\text{cp}}(\tilde{\mathbf{p}}_j, \tilde{\mathbf{q}}_i). \quad (7)$$

This leads to a more accurate estimate of $\tilde{\mathbf{M}}^i_{\text{ra}} = (\tilde{\mathbf{d}}^i_{\text{ra}}, \tilde{\mathbf{x}}^i_{\text{ra}})$ where $\tilde{\mathbf{d}}^i_{\text{ra}}$ is predicted directly and $\tilde{\mathbf{x}}^i_j$ is aggregated as explained in Sec. 3.5.

3.4. Training

The network f_θ is trained end-to-end on public datasets of articulated 3D objects. For each training object, we sample a point cloud $\mathcal{P}$ and obtain the ground-truth point-to-part segmentation $\mathbf{S} \in \{0,1\}^{N\times P}$, kinematic tree $\mathbf{K} \in \{0,1\}^{P\times P}$ and motion constraints $\mathbf{M} = (\mathbf{M}_{\text{tp}} \in \{0,1\}^{P\times 4}, \mathbf{M}_{\text{pd}} \in \mathbb{R}^{P\times 3}, \mathbf{M}_{\text{pr}} \in \mathbb{R}^{P\times 2}, \mathbf{M}_{\text{rr}} \in \mathbb{R}^{P\times 2}, \mathbf{d}_{\text{ra}} \in \mathbb{R}^{P\times 3}, \mathbf{x} \in \mathbb{R}^{N\times 3})$ to supervise f_θ.

Matching predicted parts with ground-truth. For training, we need to map each ground-truth part $i \in [P]$ to a distinct part query $j = \pi(i) \in [P_{\max}]$. Following DETR [3], the assignment $\hat{\pi} : [P] \to [P_{\max}]$ (an injection) is inferred by aligning the ground-truth and predicted point-to-part assignments $\mathbf{S}_{j,i}$ and $\tilde{\mathbf{S}}_{j,k}$ as

$$\hat{\pi} = \arg\max_{\pi} \sum_{i=1}^{P}\sum_{j=1}^{N} \mathbf{S}_{j,i} \log \frac{\exp(\tilde{\mathbf{S}}_{j,\pi(i)})}{\sum_{k=1}^{P_{\max}} \exp(\tilde{\mathbf{S}}_{j,k})}. \quad (8)$$

The assignment $\hat{\pi}$ is computed on the fly using the Hungarian algorithm and used to map the ground truth articulation attributes of each part to the corresponding query for supervision, as explained next.

Training losses. We train the network f_θ end-to-end using a multi-task loss: $\mathcal{L} = \mathcal{L}_S + \mathcal{L}_K + \mathcal{L}_M$, where $\mathcal{L}_M = \mathcal{L}_{M_{\text{tp}}} + \mathcal{L}_{M_{\text{pr}}} + 0.1\mathcal{L}_{M_{\text{rr}}} + \mathcal{L}_{M_{\text{pd}}} + \mathcal{L}_{d_{\text{ra}}} + \mathcal{L}_{x_{\text{ra}}}$ is the combined loss for motion constraints. $\mathcal{L}_S = \text{CE}(\tilde{\mathbf{S}}^*, \mathbf{S})$ is the cross-entropy loss for the part segmentation, where $\tilde{\mathbf{S}}^* \in \mathbb{R}^{N\times P}$ is obtained by column-wise permuting $\tilde{\mathbf{S}}$ according to $\hat{\pi}$ (*i.e.*, $\tilde{\mathbf{S}}^*_{j,i} = \tilde{\mathbf{S}}_{j,\hat{\pi}(i)}$, and analogously for the starred notations below). $\mathcal{L}_K = \text{BCE}(\tilde{\mathbf{K}}^*, \mathbf{K})$ is the binary cross-entropy loss for the kinematic tree. The terms in $\mathcal{L}_M$ supervise the motion parameters, comparing each part query's prediction with its matched ground truth: $\mathcal{L}_{M_{\text{tp}}} = \text{CE}(\tilde{\mathbf{M}}^*_{\text{tp}}, \mathbf{M}_{\text{tp}})$, $\mathcal{L}_{M_{\text{pr}}} = \|\tilde{\mathbf{M}}^*_{\text{pr}} - \mathbf{M}_{\text{pr}}\|_1$, $\mathcal{L}_{M_{\text{rr}}} = \|\tilde{\mathbf{M}}^*_{\text{rr}} - \mathbf{M}_{\text{rr}}\|_1$, $\mathcal{L}_{M_{\text{pd}}} = \|\tilde{\mathbf{M}}^*_{\text{pd}} - \mathbf{M}_{\text{pd}}\|_1$, $\mathcal{L}_{d_{\text{ra}}} = \|\tilde{\mathbf{d}}^*_{\text{ra}} - \mathbf{d}_{\text{ra}}\|_1$, and $\mathcal{L}_{x_{\text{ra}}} = \|\tilde{\mathbf{x}} - \mathbf{x}\|_1$.

3.5. Inference

We now describe how PARTICULATE infers the 3D articulated structure $\mathcal{A}$ from a static mesh $\mathcal{M}$.

Feed-forward prediction. Given the input mesh $\mathcal{M} = (V, F)$, we first sample a point cloud $\mathcal{P}$, ensuring that each face has at least one point $\mathbf{p}_i \in \mathcal{P}$ sampled on it. We then feed $\mathcal{P}$ into f_θ to map each point $\mathbf{p}_i$ to a corresponding part index $\tilde{\mathbf{s}}_i = \arg\max_{j\in[P_{\max}]} \tilde{\mathbf{S}}_{i,j}$. Similarly, we assign each face $f \in [|F|]$ to the part index $S(f)$ of its majority points (breaking ties randomly).

Note that the number of actual parts P in $\mathcal{A}$ is often much smaller than $P_{\max}$. The set of active part indices is simply the *range* $\mathcal{Q}_\mathcal{A} = \{S(f) : f \in [|F|]\} \subseteq [P_{\max}]$ of the face assignments S, and the number of active parts is $P = |\mathcal{Q}_\mathcal{A}|$. Next, we compute the kinematic tree K from f_θ's prediction $\tilde{\mathbf{K}} \in \mathbb{R}^{P_{\max}\times P_{\max}}$, where $\tilde{\mathbf{K}}_{i,j}$ represents the log likelihood of part query i being the parent of part query j. To do so, we run Edmonds' algorithm to extract the *maximum* spanning arborescence (MSA) of $\tilde{\mathbf{K}}$. The induced arborescence of this MSA on the present parts $\mathcal{Q}_\mathcal{A}$ forms K.

We can directly read off the network predictions $\tilde{\mathbf{M}}^{\mathcal{Q}_\mathcal{A}}_{\text{tp}}$, $\tilde{\mathbf{M}}^{\mathcal{Q}_\mathcal{A}}_{\text{pd}}$, $\tilde{\mathbf{M}}^{\mathcal{Q}_\mathcal{A}}_{\text{pr}}$ and $\tilde{\mathbf{M}}^{\mathcal{Q}_\mathcal{A}}_{\text{rr}}$ to get the motion types M_{tp}, prismatic directions M_{pd}, prismatic ranges M_{pr} and revolute ranges M_{rr} of the present parts $\mathcal{Q}_\mathcal{A}$. For the revolute axes, we first infer their directions as $\tilde{\mathbf{d}}^{\mathcal{Q}_\mathcal{A}}_{\text{ra}}$. Then, for each point $\mathbf{p}_j \in \mathcal{P}$, we use the MLP h_{cp} to predict the closest point on the axis of its part $\tilde{\mathbf{s}}_j$, so that each point effectively *votes* for the axis location. We take the (coordinate-wise) *median* of these votes as the axis location Formally, $M_{\text{ra}}(i) \in \mathbb{S}^2 \times \mathbb{R}^3$ is computed as:

$$M_{\text{ra}}(i) = \Big(\tilde{\mathbf{d}}^i_{\text{ra}}, \text{median}(\{h_{\text{cp}}(\tilde{\mathbf{p}}_j, \tilde{\mathbf{q}}_i) : \tilde{\mathbf{s}}_j = i\})\Big) \quad (9)$$

for $i \in \mathcal{Q}_\mathcal{A}$. Together, the inferred tuple (S, K, M) where $M = (M_{\text{tp}}, M_{\text{pd}}, M_{\text{ra}}, M_{\text{pr}}, M_{\text{rr}})$ defines the 3D articulated structure $\mathcal{A}$.

Refinement with connected components. Many handcrafted 3D assets have well-defined connected components, which we can leverage to refine the segmentation. For such input meshes, after obtaining the preliminary per-face part segmentation S, we ensure that all faces within the same connected component $\mathcal{C}$ are assigned to the same part, selected as the one with the largest surface coverage in $\mathcal{C}$.

4. Experiments

We evaluate PARTICULATE on recovering articulated structures from static 3D meshes and compare it with prior work that addresses this task in full or in part (Sec. 4.2). We structure the evaluation around two sub-tasks: articulated part segmentation (Sec. 4.2.2) and articulated motion prediction (Sec. 4.2.3), so we can compare our all-in-one approach

with methods that handle only one of them. In Sec. 4.3, we discuss the contribution of data and network design choices.

4.1. Experiment Details

Training datasets. PARTICULATE is trained on the PartNet-Mobility [64] and GRScenes [56] datasets. For [64], we follow the train-test split by [33]. For [56], we use all its articulated assets. We set $P_{\max}$=16 and discard objects with more than 16 articulated parts. This yields $3,800$ objects across 50 categories.

Evaluation datasets. We evaluate PARTICULATE on the test split of PartNet-Mobility [64], which contains 7 common categories. In addition, we introduce a new challenging benchmark for the 3D articulation estimation task, which contains 243 high-quality articulated objects spanning 14 categories crafted by Lightwheel and released under a CC-BY-NC license [29].

Implementation details. During training, our model is given point clouds of a fixed size N=2048, among which 50% are sampled uniformly on the mesh surface and the remaining 50% are sampled from the *sharp edges* where dihedral angles are larger than 30° [6]. We sample a random articulated state for each training iteration and compute the PartField [35] features on the fly. All point clouds are first normalized to $[-0.5, 0.5]^3$, followed by data augmentation. Each point cloud is scaled by a factor randomly drawn from $\mathcal{U}(0.95, 1.05)$ and translated by a vector from $\mathcal{N}(0, 0.02)^3$. Our model consists of B=8 attention blocks with a total of 150M parameters. The final model is optimized for 100K iterations using AdamW with a global batch size of 128 on 8 H100 GPUs. At inference time, the model receives point clouds of size $102,400$ sampled from the input mesh using the same procedure to ensure coverage of all mesh faces.

4.2. Results and Comparisons

4.2.1. Qualitative Results on Synthesized 3D Meshes

In Fig. 3, we visualize the articulated 3D objects of a variety of categories predicted by PARTICULATE from static 3D meshes generated by Hunyuan3D [15], an off-the-shelf 3D generator. Our model recovers articulated parts faithful to the kinematic structure of the input objects, with plausible motion constraints for the movable parts. Notably, despite being trained only on artist-created assets, our model generalizes well to AI-generated meshes, yielding a fully automatic pipeline for generating articulated 3D assets directly from text prompts or images.

4.2.2. Comparisons on Articulated Part Segmentation

Baselines. We compare against four state-of-the-art 3D part segmentation methods: **PartField** [35], **P3SAM** [39], **SINGAPO** [33], and **Articulate AnyMesh** [43]. For reference, we also include a **Naive Baseline**, which simply treats the entire object as a single (fixed) part. As PartField, P3SAM, and Articulate AnyMesh leverage face connectivity for part segmentation, we likewise refine SINGAPO and our outputs using mesh connected components as described in Sec. 3.5, and report metrics both with and without this refinement. PartField allows specifying the number of output parts, so we provide it with the number of ground-truth (GT) parts. P3SAM additionally requires point prompts, so we supply one point randomly sampled from each GT part. SINGAPO does *not* take a 3D mesh as input, and instead generates bounding boxes for the parts conditioned on a single object-centric image and retrieves their geometries from a repertoire of part meshes. To reduce any disadvantage from not having access to the original mesh, we make the following adjustments when evaluating SINGAPO: first, instead of one-shot generation, we generate 10 samples per instance using different rendered views for conditioning, and report a "1@10" metric computed using the best sample among the 10; second, on the Lightwheel evaluation set, we extract part geometries directly as sub-meshes of the original mesh within the generated bounding boxes, rather than retrieving them from the part repository curated from the PartNet-Mobility training set. We use the official implementations of all baselines and the model weights publicly released by the authors without further finetuning.

Metrics. We report three metrics for part segmentation: generalized Intersection over Union (**gIoU**) [45], mean Intersection over Union (**mIoU**), and bidirectional part-wise Chamfer distance (**PC**), computed between predicted and GT parts. Since the predicted and GT parts have unknown correspondence and may differ in count, we first perform Hungarian matching between the two sets based on pairwise part-centroid distances. Prior work [21, 32, 33] computes metrics only on matched part pairs and ignores unmatched parts. However, as we show in Sec. 6.1, metrics computed in this manner do *not* sufficiently penalize missing small parts: under this protocol, the Naive Baseline outperforms *all* baseline methods on *all* metrics, making these scores uninformative for assessing part-segmentation quality. To better assess the full set of predictions, we introduce a penalty for any predicted or GT part that remains unmatched. Specifically, we compute each metric $\mathcal{D}_d$ over all predicted parts $\{p_i^{\text{pred}}\}_{i=1}^N$ and GT parts $\{p_j^{\text{gt}}\}_{j=1}^M$:

$$\mathcal{D}_d = \frac{1}{2}\left(\frac{1}{N}\sum_{i=1}^{N}\tilde{d}(p_i^{\text{pred}}) + \frac{1}{M}\sum_{j=1}^{M}\tilde{d}(p_j^{\text{gt}})\right), \qquad (10)$$

where $\tilde{d}(p_i) = d(p_i, \mathcal{H}(p_i))$ if (predicted or GT) part p_i is matched to a (GT or predicted) part $\mathcal{H}(p_i)$, and is otherwise set to a penalty value $\epsilon = -1$ for gIoU, 0 for mIoU, and half the diagonal length of the input mesh's bounding box for PC. Here, d selects the metric (gIoU, mIoU, or PC), and $\mathcal{D}_d$ reports that metric with penalties applied to unmatched parts. We report $\mathcal{D}_d$ averaged over all instances in the test

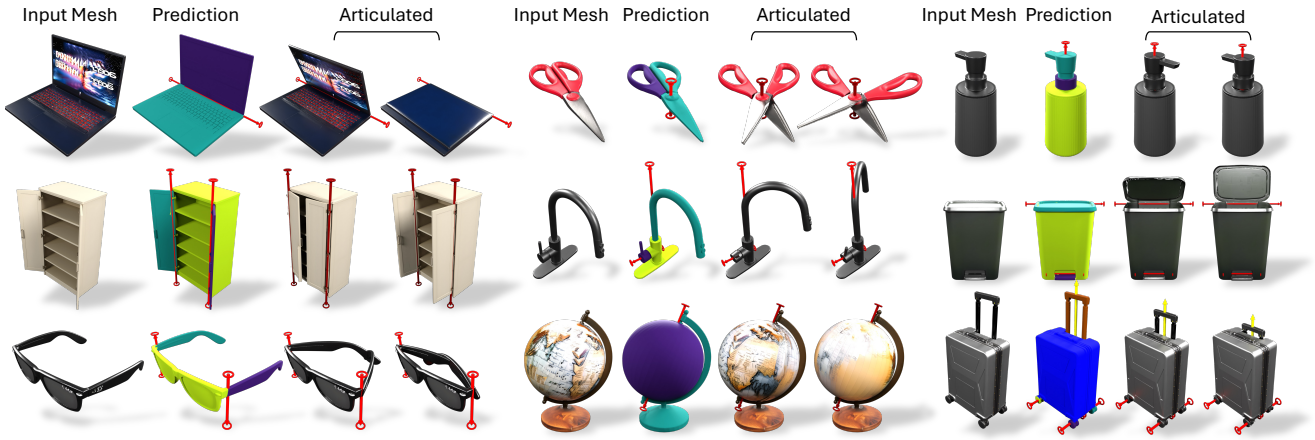

Figure 3. **Qualitative results** of PARTICULATE, illustrating the input meshes and the predicted articulated parts with their motion constraints. We also show each 3D object in two different articulated states. All input meshes are *generated* using an off-the-shelf 3D generator.

Method	Lightwheel			PartNet-Mobility		
	gIoU↑	PC↓	mIoU↑	gIoU↑	PC↓	mIoU↑
Naive Baseline	0.018	0.285	0.413	0.296	0.210	0.612
SINGAPO [33]	-0.116	0.201	0.272	–	–	–
SINGAPO [33] (1@10)	-0.096	0.190	0.277	–	–	–
PARTICULATE (ours)	**0.183**	**0.163**	0.430	**0.879**	**0.003**	**0.883**
PartField [35]†	0.079	0.106	0.264	0.183	0.123	0.361
P3SAM [39]†	0.122	0.177	0.411	-0.116	0.261	0.267
SINGAPO [33]†	-0.097	0.234	0.273	0.262	0.124	0.468
SINGAPO [33] (1@10)†	-0.050	0.221	0.297	0.271	0.117	0.471
Articulate AnyMesh [43]†	0.172	0.190	**0.452**	0.383	0.104	0.542
PARTICULATE (ours)†	**0.332**	**0.168**	**0.576**	**0.880**	**0.003**	**0.884**

Table 2. **Part segmentation results**. We report generalized IoU (gIoU), part-wise Chamfer distance (PC), and mean IoU (mIoU), all computed with penalties applied to unmatched parts. †: leveraging mesh connectivity. –: SINGAPO retrieves part geometries from a part library on the PartNet-Mobility test set, which assumes mesh connectivity. Therefore, its metrics without leveraging connectivity are left empty. 1@10: computed using the best sample out of 10 predictions per instance. Colors: **best** and **second best**.

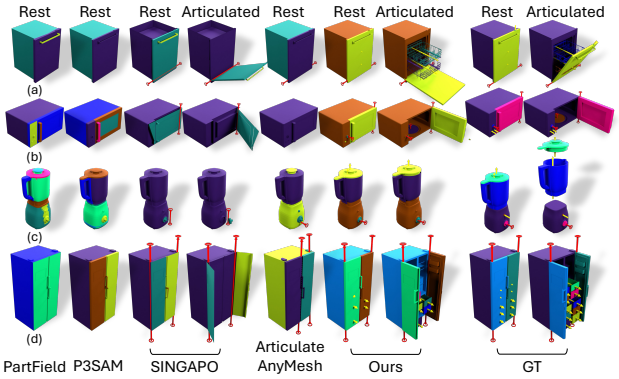

Figure 4. **Qualitative comparison** on the Lightwheel benchmark. Compared with *semantic* part segmentation methods (PartField [35] and P3SAM [39]), the segments of PARTICULATE better correspond to *articulated* parts. In contrast to Articulate AnyMesh [43] and SINGAPO [33], which fail to predict internal structures invisible from external views (*e.g.*, a, b and d), PARTICULATE recovers most of these parts and faithfully infers their motion constraints.

set as the final score, where d specifies the metric (gIoU, mIoU, or PC). All metrics are evaluated based on 10^6 points uniformly sampled from the input mesh.

Results. We report the quantitative results on the PartNet-Mobility test set and our Lightwheel benchmark in Tab. 2, where PARTICULATE consistently outperforms all baselines on both datasets. Note that PartField and P3SAM are given privileged information at test time including the exact number of GT parts, therefore avoiding any penalty for unmatched parts. All methods except P3SAM experience a performance drop on the more challenging Lightwheel dataset, which contains more diverse assets with substantially finer part annotations.

In Fig. 4, we visualize the part segmentation results of PARTICULATE and the baselines on the Lightwheel benchmark. PartField and P3SAM are trained for *semantic* part segmentation, where part definitions differ from those of *articulated* parts. Consequently, their predicted segments often fail to coincide with the GT parts (Fig. 4b). SINGAPO is trained only on a limited number of categories, and hence fails to generalize to more diverse objects (Fig. 4c). Articulate AnyMesh's VLM-based pipeline cannot handle internal parts which are invisible under the rest state (Fig. 4a and Fig. 4b). By contrast, our data-driven approach accurately recovers articulated parts, including internal ones, across a wide variety of object categories.

4.2.3. Comparisons on Articulated Motion Prediction

We now compare our method's predicted motion constraints (*i.e.*, types, axes and ranges) with those of the baselines.

Baselines. Out of the four baselines we compare for part segmentation, **SINGAPO** [33] and **Articulate**

Method	Lightwheel			PartNet-Mobility		
	gIoU↑	PC↓	OC↓	gIoU↑	PC↓	OC↓
Naive Baseline	0.016	0.293	0.017	0.296	0.216	0.027
SINGAPO [33]	-0.121	0.299	0.011	–	–	–
SINGAPO [33] (1@10)	-0.100	0.238	0.012	–	–	–
PARTICULATE (ours)	**0.165**	**0.200**	**0.008**	**0.842**	**0.024**	**0.003**
SINGAPO [33]†	-0.102	0.329	0.018	0.255	0.184	0.046
SINGAPO [33] (1@10)†	-0.056	0.261	0.019	0.264	0.168	0.041
Articulate AnyMesh [43]†	0.158	0.237	0.010	0.378	0.251	0.022
PARTICULATE (ours)†	**0.305**	**0.208**	**0.009**	**0.843**	**0.022**	**0.003**

Table 3. **Results evaluated using fully articulated geometries**, where every part is moved to its maximum extent. We report part-wise gIoU and PC as in Tab. 2, and whole-object Chamfer distance (OC), which measures the overall bidirectional Chamfer distance between the *whole* predicted geometry and its GT counterpart. Notations †, –, 1@10: same as Tab. 2. Colors: **best** and **second best**.

AnyMesh [43] also predict articulated motion, which we compare here. Note that Articulate AnyMesh predicts the motion type and axis for each part but does not estimate its range. For quantitative evaluation, we assign each predicted part the motion range of its matched GT part, using the same Hungarian matching based on part-centroid distances.

Metrics. Since the predicted parts may differ substantially from the GT parts, directly comparing the predicted motion parameters with those of the matched GT parts is not always sensible. Following prior work [32, 33], we instead compare the resulting articulated geometries. Specifically, given the predicted articulated parts and their motion constraints, we fully articulate the predicted articulated asset, where every part is moved to its maximum extent. We then compare the resulting fully articulated geometry with the ground-truth counterpart also in its fully articulated state, and report the part-wise **gIoU** and Chamfer distance (**PC**) from Sec. 4.2.2 as proxies for motion accuracy. As in the segmentation metrics, unmatched parts incur a penalty. Furthermore, we introduce a new metric, *whole-object* Chamfer distance (**OC**), which measures the overall bidirectional Chamfer distance between the entire predicted geometry and its GT counterpart in the fully articulated state.

Results. The results are reported in Tab. 3, where PARTICULATE significantly outperforms all baselines across all metrics on both datasets. This includes Articulate AnyMesh, which is given the ground-truth motion range for each predicted part. These results demonstrate the superiority of our end-to-end approach in estimating articulated motion constraints.

4.3. Ablations

Data. We train a separate model, using identical hyperparameters, on the PartNet-Mobility [64] dataset *only*. In Tab. 4 ($\mathbb{A}$ *vs.* $\mathbb{B}$), reducing the training set leads to a slightly worse model. The model still outperforms the baselines by a wide margin, indicating that ours gains mainly come from the simple, data-efficient design.

	Data	Part-Field	Axis over-param.	Articulated state			Rest state		
				gIoU↑	PC↓	OC↓	gIoU↑	PC↓	mIoU↑
$\mathbb{A}$	PM+GRS	✓	✓	**0.259**	**0.215**	**0.008**	**0.286**	**0.184**	**0.551**
$\mathbb{B}$	PM	✓	✓	0.231	0.227	0.013	0.252	**0.175**	**0.504**
$\mathbb{C}$	PM+GRS	✓	×	**0.236**	**0.224**	**0.008**	**0.262**	0.186	0.411
$\mathbb{D}$	PM+GRS	×	✓	0.174	0.245	**0.009**	0.195	0.201	0.491

Table 4. **Ablations** on training data (PM: PartNet-Mobility, GRS: GRSCenes), revolute axis over-parameterization and pre-trained PartField features, evaluated on (a subset of) the Lightwheel [29] benchmark. Colors: **best** and **second best**.

Over-parameterization of revolute axes. We evaluate the effectiveness of the proposed over-parameterization of revolute axes by comparing it with an alternative design in which an MLP directly regresses each axis in $\mathbb{R}^6$ following [21]. In Tab. 4 ($\mathbb{A}$ *vs.* $\mathbb{C}$), over-parameterization improves performance, as it mitigates overfitting and averages out errors in individual location estimates.

Pre-trained semantic features. In Tab. 4 ($\mathbb{A}$ *vs.* $\mathbb{D}$), PartField features improve performance, as they provide global shape context and richer semantic cues.

5. Conclusion

We have presented PARTICULATE, a feed-forward approach that directly predicts all attributes of the underlying articulated structure from a single static 3D mesh. Our approach departs from prior methods by training an end-to-end network with a simple and scalable architecture on a diverse collection of articulated 3D assets. The resulting model is significantly faster and more accurate at recovering articulated structures than prior methods. PARTICULATE also excels on AI-generated objects, enabling the creation of articulated assets compatible with physics simulators directly from images or text prompts.

Limitations. Despite the promising results, PARTICULATE has some limitations. First, while PARTICULATE generalizes well to unseen instances, it fails to recover articulated objects with very different kinematic structures from those seen during training. Fundamentally, this is because the available training data remains several orders of magnitude smaller than datasets in other domains (*e.g.*, LAION [46] for images or Objaverse [8, 9] for static 3D meshes). Second, although combining PARTICULATE with an off-the-shelf 3D generator enables the creation of diverse articulated assets, these assets often exhibit inter-part penetrations, due to both artifacts in generated meshes and imperfect motion predictions. Enhancing the physical plausibility of the generated articulated assets (*e.g.*, via post-training [25]) to enable large-scale sim-to-real training for robotics is a promising direction for future work.

Acknowledgments. 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. This work is partially supported by the UKRI AIRR programme (ID: u5ex) 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.

[2] Sergio Arnaud, Paul McVay, Ada Martin, Arjun Majumdar, Krishna Murthy Jatavallabhula, Phillip Thomas, Ruslan Partsey, Daniel Dugas, Abha Gejji, Alexander Sax, et al. Locate 3D: Real-world object localization via self-supervised learning in 3d. In *ICML*, 2025.

[3] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *ECCV*, 2020.

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

[5] Honghua Chen, Yushi Lan, Yongwei Chen, and Xingang Pan. ArtiLatent: Realistic articulated 3d object generation via structured latents. In *SIGGRAPH Asia*, 2025.

[6] Rui Chen, Jianfeng Zhang, Yixun Liang, Guan Luo, Weiyu Li, Jiarui Liu, Xiu Li, Xiaoxiao Long, Jiashi Feng, and Ping Tan. Dora: Sampling and benchmarking for 3d shape variational auto-encoders. In *CVPR*, 2025.

[7] Deemos. Rodin text-to-3D gen-1 (0525) v0.5, 2024.

[8] 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*, 2023.

[9] 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.

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

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

[12] 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.

[13] Pradyumn Goyal, Dmitry Petrov, Sheldon Andrews, Yizhak Ben-Shabat, Hsueh-Ti Derek Liu, and Evangelos Kalogerakis. Geopard: Geometric pretraining for articulation prediction in 3d shapes. In *ICCV*, 2025.

[14] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *NeurIPS*, 2020.

[15] Team Hunyuan3D, Shuhui Yang, Mingxin Yang, Yifei Feng, Xin Huang, Sheng Zhang, Zebin He, Di Luo, Haolin Liu, Yunfei Zhao, Qingxiang Lin, Zeqiang Lai, Xianghui Yang, Huiwen Shi, Zibo Zhao, Bowen Zhang, Hongyu Yan, Lifu Wang, Sicong Liu, Jihong Zhang, Meng Chen, Liang Dong, Yiwen Jia, Yulin Cai, Jiaao Yu, Yixuan Tang, Dongyuan Guo, Junlin Yu, Hao Zhang, Zheng Ye, Peng He, Runzhou Wu, Shida Wei, Chao Zhang, Yonghao Tan, Yifu Sun, Lin Niu, Shirui Huang, Bojian Zheng, Shu Liu, Shilin Chen, Xiang Yuan, Xiaofeng Yang, Kai Liu, Jianchen Zhu, Peng Chen, Tian Liu, Di Wang, Yuhong Liu, Linus, Jie Jiang, Jingwei Huang, and Chunchao Guo. Hunyuan3D 2.1: From images to high-fidelity 3D assets with production-ready PBR material. *arXiv*, 2506.15442, 2025.

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

[17] Zhenyu Jiang, Cheng-Chun Hsu, and Yuke Zhu. Ditto: Building digital twins of articulated objects from interaction. In *CVPR*, 2022.

[18] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3D Gaussian Splatting for real-time radiance field rendering. *SIGGRAPH*, 42(4), 2023.

[19] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. In *ICCV*, 2023.

[20] Long Le, Jason Xie, William Liang, Hung-Ju Wang, Yue Yang, Yecheng Jason Ma, Kyle Vedder, Arjun Krishna, Dinesh Jayaraman, and Eric Eaton. Articulate-Anything: Automatic modeling of articulated objects via a vision-language foundation model. In *ICLR*, 2025.

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

[22] Chengshu Li, Ruohan Zhang, Josiah Wong, Cem Gokmen, Sanjana Srivastava, Roberto Martín-Martín, Chen Wang, Gabrael Levine, Michael Lingelbach, Jiankai Sun, et al. BEHAVIOR-1K: A human-centered, embodied ai benchmark with 1,000 everyday activities and realistic simulation. In *CoRL*, 2023.

[23] 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.

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

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

[26] 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.

[27] 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.

[28] 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.

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

[30] 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*, 44(4), 2025.

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

[32] Jiayi Liu, Hou In Ivan Tam, Ali Mahdavi-Amiri, and Manolis Savva. CAGE: Controllable articulation generation. In *CVPR*, 2024.

[33] 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.

[34] 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.

[35] 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.

[36] Yongcheng Liu, Bin Fan, Shiming Xiang, and Chunhong Pan. Relation-shape convolutional neural network for point cloud analysis. In *CVPR*, 2019.

[37] 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.

[38] Ruijie Lu, Yu Liu, Jiaxiang Tang, Junfeng Ni, Yuxiang Wang, Diwen Wan, Gang Zeng, Yixin Chen, and Siyuan Huang. DreamArt: Generating interactable articulated objects from a single image. *arXiv preprint arXiv:2507.05763*, 2025.

[39] Changfeng Ma, Yang Li, Xinhao Yan, Jiachen Xu, Yunhan Yang, Chunshi Wang, Zibo Zhao, Yanwen Guo, Zhuo Chen, and Chunchao Guo. P3-SAM: Native 3d part segmentation. *arXiv preprint arXiv:2509.06784*, 2025.

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

[41] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020.

[42] 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.

[43] 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.

[44] Alexander Raistrick, Lingjie Mei, Karhan Kayan, David Yan, Yiming Zuo, Beining Han, Hongyu Wen, Meenal Parakh, Stamatis Alexandropoulos, Lahav Lipson, et al. Infinigen Indoors: Photorealistic indoor scenes using procedural generation. In *CVPR*, 2024.

[45] 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.

[46] Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, et al. Laion-5b: An open large-scale dataset for training next generation image-text models. *NeurIPS*, 2022.

[47] Bokui Shen, Fei Xia, Chengshu Li, Roberto Martín-Martín, Linxi Fan, Guanzhi Wang, Claudia Pérez-D'Arpino, Shyamal Buch, Sanjana Srivastava, Lyne Tchapmi, et al. iGibson 1.0: a simulation environment for interactive tasks in large realistic scenes. In *IROS*, 2021.

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

[49] 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.

[50] Jiayi Su, Youhe Feng, Zheng Li, Jinhua Song, Yangfan He, Botao Ren, and Botian Xu. ArtFormer: Controllable generation of diverse 3d articulated objects. In *CVPR*, 2025.

[51] 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 e-prints*, pages arXiv–2408, 2024.

[52] Tencent Hunyuan3D Team. Hunyuan3D 1.0: A unified framework for text-to-3d and image-to-3d generation. *arXiv preprint arXiv:2411.02293*, 2024.

[53] Tencent Hunyuan3D Team. Hunyuan3D 2.0: Scaling diffusion models for high resolution textured 3d assets generation. *arXiv preprint arXiv:2501.12202*, 2025.

[54] Tencent Hunyuan3D Team. Hunyuan3d 2.5: Towards high-fidelity 3d assets generation with ultimate details, 2025.

[55] 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.

[56] 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.

[57] Jiawei Wang, Dingyou Wang, Jiaming Hu, Qixuan Zhang, Jingyi Yu, and Lan Xu. Kinematify: Open-vocabulary synthesis of high-dof articulated objects. *arXiv preprint arXiv:2511.01294*, 2025.

[58] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. Dynamic graph cnn for learning on point clouds. *ACM TOG*, 2019.

[59] 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.

[60] Yijia Weng, Bowen Wen, Jonathan Tremblay, Valts Blukis, Dieter Fox, Leonidas Guibas, and Stan Birchfield. Neural implicit representation for building digital twins of unknown articulated objects. In *CVPR*, 2024.

[61] Di Wu, Liu Liu, Zhou Linli, Anran Huang, Liangtu Song, Qiaojun Yu, Qi Wu, and Cewu Lu. Reartgs: Reconstructing and generating articulated objects via 3d gaussian splatting with geometric and motion constraints. In *NeurIPS*, 2025.

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

[63] Tianhao Wu, Fangcheng Zhong, Andrea Tagliasacchi, Forrester Cole, and Cengiz Oztireli. D$\hat{2}$nerf: Self-supervised decoupling of dynamic and static objects from a monocular video. *NeurIPS*, 2022.

[64] 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.

[65] Jianfeng Xiang, Zelong Lv, Sicheng Xu, Yu Deng, Ruicheng Wang, Bowen Zhang, Dong Chen, Xin Tong, and Jiaolong Yang. Structured 3d latents for scalable and versatile 3d generation. In *CVPR*, 2025.

[66] 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.

[67] 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.

[68] Biao Zhang, Jiapeng Tang, Matthias Niessner, and Peter Wonka. 3DShape2VecSet: A 3d shape representation for neural fields and generative diffusion models. *ACM TOG*, 42 (4):1–16, 2023.

[69] 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.