URDF-Anything+: End-to-End Generation for Simulation-Ready Articulated Assets

Zhuangzhe Wu[*,1] **Yue Xin**[*,1] **Chengkai Hou**[1] **Minghao Chen**[2†]
Yaoxu Lyu[1] **Jieyu Zhang**[1] **Shanghang Zhang**[1,✉]

[1] State Key Laboratory of Multimedia Information Processing,
School of Computer Science, Peking University
[2] Oxford University

[*] Equal contribution. [✉] Corresponding author

Project page: `https://urdf-anything-plus.github.io/`

Abstract

Articulated objects are fundamental for robotics, simulation of physics, and interactive virtual environments. However, recovering them from visual observations is inherently challenging, as images provide only partial and ambiguous cues about both part geometry and their underlying kinematic structure. Existing approaches typically rely on multi-stage pipelines, retrieval from asset libraries, or explicit part segmentation. We present *URDF-Anything+*, an end-to-end autoregressive diffusion framework that generates simulation-ready URDF models directly from a single RGB image. A pretrained image-to-3D model supplies whole-object geometry, which, together with image features, conditions the joint generation of part geometry and articulation. Specifically, the model sequentially predicts each articulated part together with its associated joint parameters, while a termination token dynamically determines the number of parts. The predicted links and joints can be assembled into executable URDFs without separate part segmentation, asset retrieval, or joint fitting. Experiments on large-scale articulated object benchmarks demonstrate that URDF-Anything+ outperforms prior methods in geometric reconstruction quality, joint parameter estimation, and physical executability, while being substantially more efficient than existing multi-stage approaches. In the evaluated manipulation tasks, the generated URDFs support Real-Follow-Sim: policies trained in simulation generate trajectories in the registered simulator that the real robot executes without real-world fine-tuning.

1 Introduction

Robotic manipulation, physics-based simulation, and interactive virtual environments rely on simulation-ready articulated object models [41, 31]. These models must capture both geometry and kinematic structure to support physical interaction and control [22]. However, constructing such assets (e.g., in URDF or XML formats) remains largely manual and labor-intensive. Recovering articulated object models directly from visual observations is therefore a highly desirable but challenging problem.

A model has to solve two coupled subproblems at once: it must decompose the object into rigid parts, and for every pair of connected parts it must recover the joint specification, including the joint origin, rotation axis, joint type, and motion limits. It requires jointly inferring part geometry and their underlying kinematic relationships from incomplete and ambiguous visual cues. Faced with

[†] Participated in this work while interning at Peking University.

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

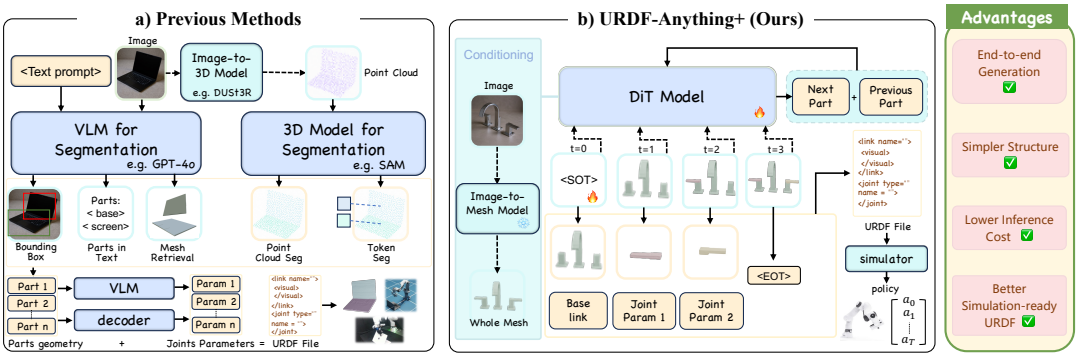


Figure 1: **Comparison between previous methods and our method.** In (a), we summarize previous generation paradigms for articulated objects, including different segmentation approaches and joint parameter generation methods. In (b), we briefly illustrate our method and its advantages.

this difficulty, prior work has largely converged on multi-stage pipelines that decouple segmentation and articulation, as illustrated in Fig. 1(a). Some methods retrieve part meshes from an asset library and fit articulation on top [7, 16], bounding geometry to the library's coverage. Others delegate part segmentation to a vision-language model before synthesizing geometry and joints [16, 4, 48]. Another line first segments part-level point clouds and then generates joint parameters with a separate code head [30]. Such pipelines compound error across stages or generalize poorly beyond categories well represented in retrieval libraries.

We propose **URDF-Anything+**, a framework for generating simulation-ready articulated URDFs from a single RGB image. A pretrained image-to-3D model first reconstructs whole-object geometry. The articulated generator then uses this geometry and image features to generate links and joints autoregressively.

We build on the compositional generation of AutoPartGen [6] by jointly predicting each rigid link and its articulation parameters. In particular, the model predicts the joint origin and axis in the same object coordinate frame as the link geometry, so the predictions can be assembled directly into a URDF.

A shared diffusion model generates the link geometry and joint parameters at each autoregressive step in a structured 3D latent space. The articulated generation stage requires no separate part segmentation, asset retrieval, or joint fitting.

We first evaluate URDF-Anything+ on large-scale articulated object benchmarks, where our method significantly outperforms previous approaches in both geometric reconstruction and joint parameter prediction, and achieves a substantially higher rate of physically executable URDFs on held-out object instances. We further introduce **Real-Follow-Sim**, a deployment scheme in which a simulation-trained policy runs in a scene registered to the current trial and the real robot executes the resulting trajectory.

Our contributions are threefold:

1. We introduce URDF-Anything+, an autoregressive diffusion framework that generates articulated URDFs link by link from a single image and its reconstructed whole-object geometry. Parts are synthesized in a unified coordinate system, simplifying URDF assembly. The model is lightweight and runs in 34.70 s on average, considerably faster than prior multi-stage pipelines.

2. We demonstrate **Real-Follow-Sim**: the generated URDFs support simulation rollouts whose trajectories are executed by the real robot without real-world fine-tuning.

3. Extensive experiments demonstrate state-of-the-art performance in geometry, joint accuracy, and physical executability, while running considerably faster than prior multi-stage methods.

2 Related Work

3D Generation Method. Researchers have developed strong geometric priors for synthesizing shapes from text, images, or multi-view observations, enabling scalable reconstruction and synthesis of 3D content. Single-image-to-3D generation addresses the challenge of recovering full 3D geometry from sparse visual input [44, 18, 11, 47, 25, 24, 26, 28]. Beyond optimization-based pipelines, generative 3D models explore alternative representations and latent structures, including implicit generative models such as Shap·E [15] and GET3D [10], as well as structured latent formulations exemplified by Trellis [43, 42] and CLAY [49]. Gaussian-based 3D generation further improves efficiency and rendering quality through explicit point-based representations [38, 45]. Complementary to holistic shape synthesis, part-level 3D generation explicitly models semantic parts to improve controllability and compositionality, through either diffusion-based multi-view part reconstruction or autoregressive part generation and discovery [5, 6].

Articulated Object Models. Currently, they focus on recovering part decomposition, kinematic structure, and motion constraints of human-made objects from visual observations, forming a foundation for robotic interaction and simulation [22]. Early work primarily studied articulated structure inference from 3D geometry or partial observations, supported by part-level benchmarks and interactive simulation platforms such as PartNet [31] and SAPIEN [41]. These efforts explored generalizable kinematic modeling, movable or openable part detection, part-level pose estimation, and articulated shape representation, including self-supervised category-level articulation learning and disentangled implicit formulations [1, 14, 21, 27, 32]. Subsequent approaches shift toward vision-conditioned and controllable modeling, leveraging visual input to recover articulated structure directly from images, including neural implicit representations for articulated digital twins and vision- or vision-language-based articulation reasoning [39, 16, 7, 20, 23]. Alongside vision-conditioned articulation modeling, growing attention has been directed toward generating simulation-ready articulated assets with executable representations, including code-based reconstruction of articulated objects [30], symbolic URDF construction via multimodal language models [19], and physically grounded asset synthesis compatible with physics engines [4]. NAP [17] learns a diffusion prior over articulation graphs, while MeshArt [9] synthesizes meshes from an autoregressively generated articulated structure. ArtFormer [37] jointly generates geometry codes and kinematic relations for controllable synthesis. Our task is to reconstruct a specific object from a single RGB image, jointly predicting its link meshes and continuous joint parameters for URDF assembly.

3 Method

Given a single RGB image, our goal is to infer a complete articulated object model that explicitly represents part geometries and their kinematic relationships. The output is a valid URDF (Unified Robot Description Format) file that can be directly instantiated in standard physics simulators, enabling downstream robot policy learning without manual asset engineering.

Given the input image, we extract visual features with a pretrained DINOv3 encoder [36] and reconstruct the whole-object mesh with a pretrained TripoSG model [18]. Based on these global visual and geometric cues, our model sequentially generates part-level geometries and their associated joint specifications, including joint type, axis, origin, and limits. The predicted components are finally assembled into a fully specified URDF model.

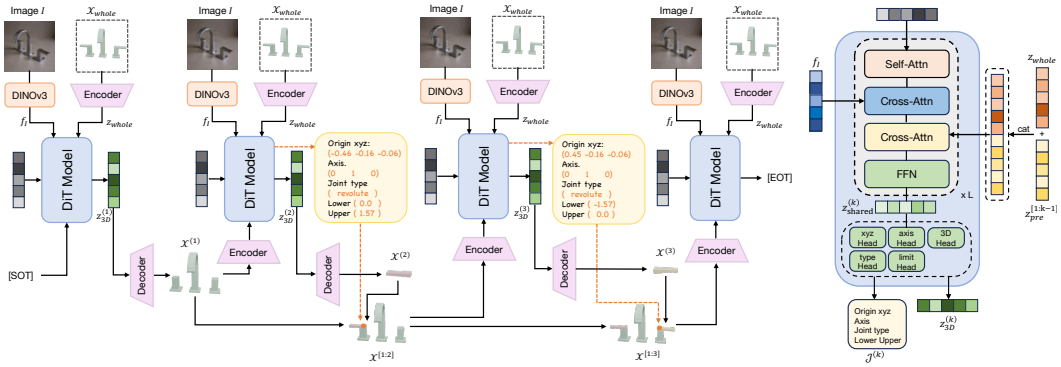
3.1 Task Definition

Let $\mathcal{X}_{\text{whole}}$ denote the complete object-level 3D geometry capturing the external appearance of an articulated object, typically represented as a single mesh. Importantly, $\mathcal{X}_{\text{whole}}$ does not necessarily coincide with the union of its part-level geometries, as articulated parts may contain internal or functional structures that are not directly observable from the exterior surface.

An articulated object is represented by

$$\mathcal{A} = \big(\{\mathcal{X}^{(k)}\}_{k=1}^{K}, \{\mathcal{J}^{(k)}\}_{k=1}^{K}\big), \tag{1}$$

where $\mathcal{X}^{(k)}$ denotes the mesh of the k-th rigid link and $\mathcal{J}^{(k)}$ specifies the joint parameters defining the kinematic relationship between part $\mathcal{X}^{(k)}$ and its parent. Each $\mathcal{J}^{(k)}$ includes the joint origin,



(a) Generation pipeline. (b) DiT and decoders.

Figure 2: **Overview of URDF-Anything+.** (a) The autoregressive generation pipeline that produces 3D parts together with their joint specifications and assembles them into a complete URDF. (b) Structure of the DiT backbone and the geometry / articulation parameter decoders.

axis, type, and motion limits. Together, these components define a URDF model that can be directly instantiated in a physics simulator.

We consider the image-to-articulated-model setting: given a single RGB image I, the task is to infer a plausible articulated representation $\mathcal{A}$, i.e. $I \to \mathcal{A}$. We consider rigid links connected by revolute or prismatic joints, including mechanisms with multiple movable joints. When multiple hidden mechanisms are consistent with the image, the model samples a plausible interpretation according to its learned prior. Since recovering articulated structure from visual input is inherently ambiguous, we cast it as sampling from a conditional distribution $p(\mathcal{A} \mid I)$, from which valid URDF models can be sampled for downstream real-to-sim-to-real policy learning. We instantiate this distribution through an autoregressive factorization over parts, detailed in the following subsections.

3.2 Latent 3D Shape Modeling via Diffusion

Directly modeling probability distributions over continuous 3D surfaces is intractable due to their infinite-dimensional nature. We therefore adopt a finite-dimensional latent representation and perform generative modeling in this structured latent space.

We represent 3D geometry with the pretrained encoder-decoder pair (E, D) of TripoSG [18], which instantiates the *VecSet* formulation [46]. Given a surface $\mathcal{X}$, the encoder maps an N-point cloud $\mathcal{P} = \{\boldsymbol{p}_i \in \mathbb{R}^3\}_{i=1}^{N} \subseteq \mathcal{X}$ to a compact latent representation $\boldsymbol{z} = E(\mathcal{P}) = (z_1, \ldots, z_M) \in \mathbb{R}^{M \times d_z}$, where $M \ll N$ and each z_i is a d_z-dimensional feature vector. The encoder E is implemented as a permutation-equivariant transformer that subsamples points and aggregates global context via cross-attention. The decoder D evaluates the signed distance function by mapping a query point $\boldsymbol{p} \in \mathbb{R}^3$ and the latent tokens $\boldsymbol{z}$ to $\mathrm{SDF}(\boldsymbol{p} \mid \mathcal{X}) = D(\boldsymbol{p} \mid \boldsymbol{z})$, using a Perceiver-style architecture [13]. This provides a continuous implicit representation of the 3D shape.

To generate shapes conditioned on visual evidence y (e.g., an input image), we learn a conditional distribution $p(\boldsymbol{z} \mid y)$ using latent diffusion. We define a forward noising process

$$\boldsymbol{z}_t = \sqrt{\bar{\alpha}_t}\, \boldsymbol{z}_0 + \sqrt{1 - \bar{\alpha}_t}\, \epsilon, \quad \epsilon \sim \mathcal{N}(0, I), \tag{2}$$

with a predefined noise schedule $\{\alpha_t\}_{t=0}^{T}$ and $\bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s$. Following [35], we parameterize the diffusion process via the flow velocity

$$\boldsymbol{v}(t, \boldsymbol{z}_0, \epsilon) = \sqrt{\bar{\alpha}_t}\, \epsilon - \sqrt{1 - \bar{\alpha}_t}\, \boldsymbol{z}_0. \tag{3}$$

A neural network $\hat{\boldsymbol{v}}_\theta(z_t, t \mid y)$ is trained to predict the velocity at diffusion step t, conditioned on the noised latent z_t and the input evidence y.

$$\mathcal{L}_{diff} = \mathbb{E}_{y, \boldsymbol{z}_0, t, \epsilon} \left\| \hat{\boldsymbol{v}}_\theta(t, \boldsymbol{z}_t \mid y) - \boldsymbol{v}(t, \boldsymbol{z}_0, \epsilon) \right\|^2. \tag{4}$$

After training, the model can sample latent shape representations consistent with the input evidence, which are then decoded into 3D geometry.

3.3 Autoregressive Articulated Model Generation

We build on the compositional autoregressive generation of AutoPartGen [6] by predicting each link together with its joint type, origin, axis, and limits. The link geometry and its joint origin and axis are expressed in the same object coordinate frame, which allows direct URDF assembly without fitting joints to the generated meshes. Fig. 2a provides an overview of the generation process.

Let f_I denote the DINOv3 image features and $\mathcal{X}_{\text{whole}}$ the complete object-level mesh, which may be either pre-given or produced by an off-the-shelf 3D reconstruction model, e.g., TripoSG [18]. The reconstructed mesh is encoded into a global 3D latent $\boldsymbol{z}_{\text{whole}} = E(\mathcal{X}_{\text{whole}})$. The pair $(f_I, \boldsymbol{z}_{\text{whole}})$ serves as global context for autoregressive generation. This fixed whole-object condition anchors each generation step to the observed object geometry. At each autoregressive step k, the model generates a single articulated part represented by the tuple $\left(\boldsymbol{z}_{\text{3D}}^{(k)}, \mathcal{J}^{(k)}\right)$, where $\boldsymbol{z}_{\text{3D}}^{(k)}$ is a latent code representing the geometry of the k-th part and $\mathcal{J}^{(k)}$ denotes its articulation parameters. Given the global context $(f_I, \boldsymbol{z}_{\text{whole}})$ and the context accumulated from previously generated parts $\boldsymbol{z}_{\text{pre}}^{[1:k-1]}$ (defined in Eq. 11), the joint distribution factorizes autoregressively:

$$p\Big(\{\boldsymbol{z}_{\text{3D}}^{(k)}, \mathcal{J}^{(k)}\}_{k=1}^{K} \mid f_I, \boldsymbol{z}_{\text{whole}}\Big) = \prod_{k=1}^{K} p\Big(\boldsymbol{z}_{\text{3D}}^{(k)}, \mathcal{J}^{(k)} \mid f_I, \boldsymbol{z}_{\text{whole}}, \boldsymbol{z}_{\text{pre}}^{[1:k-1]}\Big). \tag{5}$$

The number of parts K is not predetermined: alongside the geometry and articulation targets, the model is trained to predict an end-of-token (`[EoT]`) latent that signals termination of the autoregression (Appendix B.2).

Latent Diffusion with Shared Latent. Part generation is driven by a diffusion transformer (DiT) operating in latent space. At step k, the DiT takes a noisy latent $\boldsymbol{z}_t$ together with conditioning tokens derived from $(f_I, \boldsymbol{z}_{\text{whole}})$ and $\boldsymbol{z}_{\text{pre}}^{[1:k-1]}$, and predicts a denoising velocity field. Running the reverse process from Gaussian noise yields the *shared latent*

$$\boldsymbol{z}_{\text{shared}}^{(k)} \sim p_\theta\Big(\cdot \mid f_I, \boldsymbol{z}_{\text{whole}}, \boldsymbol{z}_{\text{pre}}^{[1:k-1]}\Big), \tag{6}$$

which aggregates global geometric structure and contextual information for the current part. Both the geometry head and all articulation heads decode from this single $\boldsymbol{z}_{\text{shared}}^{(k)}$, hence *shared*.

Geometry Latent and 3D Decoding. The geometry latent of the k-th part is obtained by projecting the shared latent through a dedicated 3D head: $\boldsymbol{z}_{\text{3D}}^{(k)} = h_{\text{3D}}\Big(\boldsymbol{z}_{\text{shared}}^{(k)}\Big)$. We then reconstruct the part geometry from the SDF produced by the TripoSG decoder D via marching cubes: $\mathcal{X}^{(k)} = \text{MC}\Big(D\Big(\cdot \mid \boldsymbol{z}_{\text{3D}}^{(k)}\Big)\Big)$.

Articulation Parameter Decoding. Articulation parameters are predicted directly (without diffusion) from the shared latent. We employ multiple lightweight MLP heads, each responsible for a specific joint attribute:

$$\mathbf{o}^{(k)} = h_{\text{orig}}\Big(\boldsymbol{z}_{\text{shared}}^{(k)}\Big), \tag{7}$$

$$\mathbf{a}^{(k)} = h_{\text{axis}}\Big(\boldsymbol{z}_{\text{shared}}^{(k)}\Big), \tag{8}$$

$$\ell_{\text{low}}^{(k)}, \ell_{\text{up}}^{(k)} = h_{\text{low, up}}\Big(\boldsymbol{z}_{\text{shared}}^{(k)}\Big), \tag{9}$$

$$\tau^{(k)} = h_{\text{type}}\Big(\boldsymbol{z}_{\text{shared}}^{(k)}\Big), \tag{10}$$

We denote the complete articulation specification as $\mathcal{J}^{(k)} = \big(\mathbf{o}^{(k)}, \mathbf{a}^{(k)}, \tau^{(k)}, \ell_{\text{low}}^{(k)}, \ell_{\text{up}}^{(k)}\big)$. Conditioning articulation prediction on the shared latent allows joint parameters to leverage global context captured during diffusion, leading to geometrically consistent articulation. Fig. 2b shows the DiT and parameter decoder's structure.

Re-encoding and Autoregressive Context Update. After decoding the k-th part, we merge it with previously generated geometry: $\mathcal{X}^{[1:k]} = \bigcup_{j=1}^{k} \mathcal{X}^{(j)}$. Following AutoPartGen [6], we uniformly

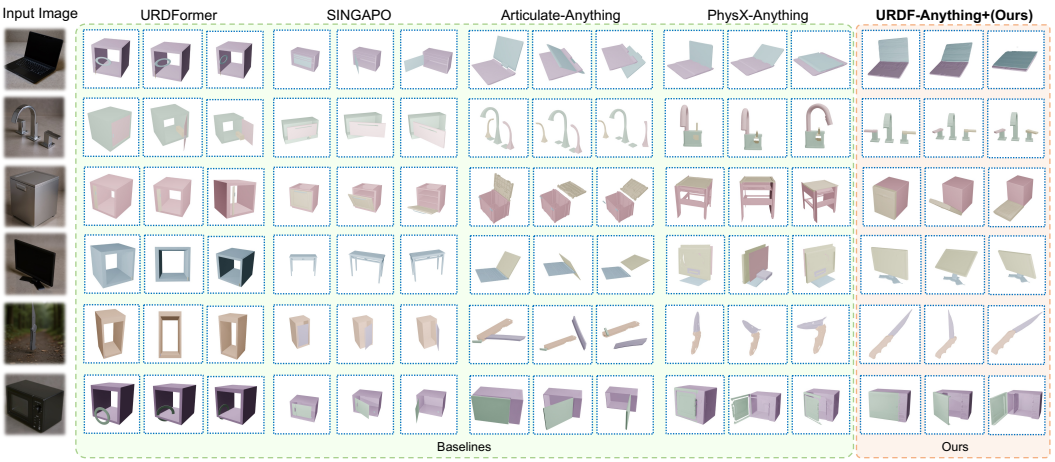


Figure 3: **Qualitative Results on the Test Set of Our Dataset.** Compared with other methods, URDF-Anything+ generates high-quality 3D assets with more accurate geometry and articulation.

sample an N-point cloud $\mathcal{P}^{[1:k]} \subseteq \mathcal{X}^{[1:k]}$ from the merged surface and re-encode it with E (Sec. 3.2) to update the context:

$$\boldsymbol{z}_{\text{pre}}^{[1:k]} = E\Big(\mathcal{P}^{[1:k]}\Big). \tag{11}$$

Overall Perspective. By iterating over shared-latent diffusion, geometry decoding, articulation prediction, mesh merging, and re-encoding, the proposed framework incrementally constructs a complete articulated object. This design preserves the compositional strengths of AutoPartGen while enabling coherent, end-to-end generation of URDF-ready articulated models.

4 Experiments

We evaluate URDF-Anything+ along three axes: geometric quality of generated part meshes (Sec. 4.2), joint parameter prediction accuracy (Sec. 4.3), and end-to-end physical executability via real-robot deployment (Sec. 4.5). We further ablate key design choices in Sec. 4.4.

4.1 Implementation Details.

Training Details. We adopt a two-stage training strategy to stabilize learning and progressively introduce articulation-aware supervision. In **Stage 1**, the model is trained for 500 epochs on 16 NVIDIA A800 GPUs using only image and part-geometry supervision, excluding the end-of-token (`[EoT]`) signal and URDF-based articulation annotations. This stage focuses on learning robust latent geometry representations and autoregressive part composition. In **Stage 2**, training continues for an additional 100 epochs with the `[EoT]` signal and URDF-based articulation annotations enabled, thereby introducing full articulation supervision. This stage allows the model to jointly learn part termination, articulation parameter prediction, and geometry generation in a unified framework. More details are reported at Appendix B.1 .

Training Data. Our training data is constructed from a curated collection of articulated 3D datasets, including PartNet [31], PartNet-Mobility, and PhysX-3D [3]. We split 2,626 objects from 10 categories at the instance level into 2,419 training objects and 207 held-out test objects. The test set contains new instances from categories represented in training. We apply extensive data cleaning and filtering to ensure geometric validity and consistent articulation structure. Further implementation details are provided in supplementary material Section A .

4.2 Geometric Quality Evaluation

We compare URDF-Anything+ with the most relevant state-of-the-art methods on articulated 3D geometry generation, including URDFormer [7], Articulate-Anything [16], SINGAPO [20], and

Table 1: **Main Results.** Geometry reconstruction quality (Parts and Whole Object), joint parameter prediction accuracy, and average inference time per generated part. We report IoU and F-Score (higher is better, ↑), CD (lower is better, ↓), joint Axis (rad), Origin (m), Limit (rad) errors (lower is better, ↓), and per-part inference time (s, lower is better, ↓). For this comparison, our model is conditioned on image features and the ground-truth whole-object mesh. All reported times include geometry decoding and URDF assembly and are measured on the same hardware.

Method	**Parts**			**Whole Object**			**Joint Accuracy**			Time / Part (s) ↓
	IoU ↑	F-Score ↑	CD ↓	IoU ↑	F-Score ↑	CD ↓	Axis Err. ↓	Origin Err. ↓	Limit Err. ↓	
URDFormer [7]	0.202	0.389	0.354	0.243	0.413	0.286	$1.071_{\pm 0.497}$	$0.504_{\pm 0.331}$	$0.882_{\pm 0.679}$	41.88
Articulate-Anything [16]	0.563	0.452	0.064	0.627	0.510	0.053	$0.230_{\pm 0.505}$	$0.267_{\pm 0.357}$	$1.108_{\pm 0.676}$	380.57
SINGAPO [20]	0.382	0.434	0.165	0.462	0.481	0.132	$0.670_{\pm 0.472}$	$0.404_{\pm 0.297}$	$0.817_{\pm 0.525}$	52.30
PhysX-Anything [4]	0.784	0.697	0.076	0.813	0.715	0.068	$0.191_{\pm 0.337}$	$0.103_{\pm 0.192}$	$0.532_{\pm 0.423}$	578.38
URDF-Anything+ (Ours)	**0.879**	**0.721**	**0.033**	**0.930**	**0.742**	**0.009**	$\mathbf{0.129}_{\pm 0.404}$	$\mathbf{0.062}_{\pm 0.096}$	$\mathbf{0.225}_{\pm 0.440}$	**34.70**

PhysX-Anything [4]. As shown in Table 1, URDF-Anything+ consistently achieves superior performance across all geometric metrics, including IoU, F-score, and Chamfer Distance (CD). In particular, URDF-Anything+ attains the highest IoU and F-score while achieving the lowest CD, indicating more accurate surface reconstruction and better geometric completeness. Compared to retrieval-based or single-pass generation methods, our approach better captures fine-grained part geometry and preserves global structural consistency across articulated components. We observe that PhysX-Anything and URDF-Anything+ achieve higher performance in geometric generation, as measured by IoU and F-score, which quantify the overlap between the generated geometry and the ground-truth shapes, compared to retrieval-based methods such as URDFormer and Articulate-Anything, which is consistent with our expectations.

Qualitative Comparison. Fig. 3 and Fig. 4 compare reconstructions across object categories. URDFormer [7] and SINGAPO [20] rely on a relatively small retrieval library and recover acceptable geometry and joint estimates on simple, box-like categories such as storage furniture and cabinets, but largely fail on shapes that fall outside the library, including faucets, refrigerators, and other objects with curved or thin parts. Articulate-Anything [16] draws from a larger asset base and therefore returns more matching meshes, yet its predicted joint relations are frequently inconsistent—e.g., misaligned axes or origins placed away from the actual hinge—because segmentation and articulation are reasoned about separately. PhysX-Anything [4] is the strongest baseline overall and yields visually plausible geometry on most categories, but still mispredicts joint axes or part assemblies on objects with thin or repeated structures. In contrast, URDF-Anything+ generates parts and joints jointly from a shared latent, producing cleaner part boundaries and more consistent kinematic structure across all shown categories.

4.3 Joint Parameter Prediction Accuracy.

Following the evaluation protocol of Articulate-Anything [16], we evaluate the accuracy of joint parameter prediction by comparing the predicted URDF parameters with ground-truth annotations. We focus on articulated objects with *revolute* joints and report errors on three key joint attributes: (1) **Joint Axis Error**, computed as the angle in radians between the directed unit axes, $\arccos(\hat{\mathbf{a}}^\top \mathbf{a})$, in $[0, \pi]$; (2) **Joint Origin Error**, measured as the Euclidean distance between the predicted and ground-truth joint origins; and (3) **Joint Limit Error**, computed as the mean absolute error of the lower and upper limits.

Table 1 presents the quantitative results. We compare URDF-Anything+ (Ours) with several prior methods on the same evaluation set. As shown in the table, URDF-Anything+ consistently achieves lower errors across all reported joint parameters. This indicates that our method can more accurately recover the kinematic configuration of articulated objects, particularly in terms of joint geometry and motion constraints. These results demonstrate the effectiveness of our approach in predicting fine-grained joint parameters. We attribute the improved performance to our end-to-end framework, which couples autoregressive geometric generation with shared-latent articulation decoding. Beyond the quantitative comparison, we further present qualitative results in Fig. 3 and Fig. 4.

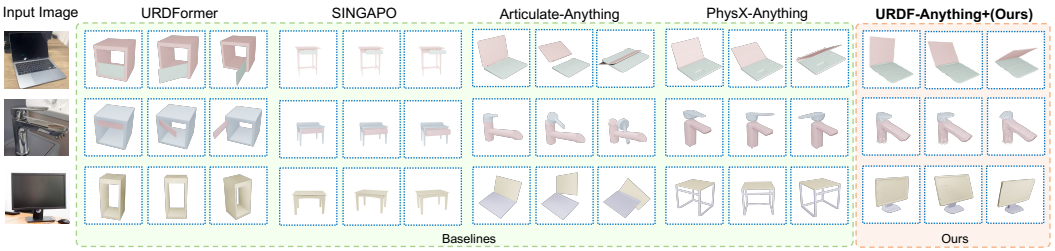


Figure 4: **Qualitative Results on In-the-wild Images**. Compared with other methods, URDF-Anything+ generates high-quality 3D assets with more accurate geometry and articulation.

Table 2: **Ablation Studies.** We evaluate the impact of input modalities (top) and link ordering strategies (bottom) on geometry reconstruction and joint prediction accuracy.

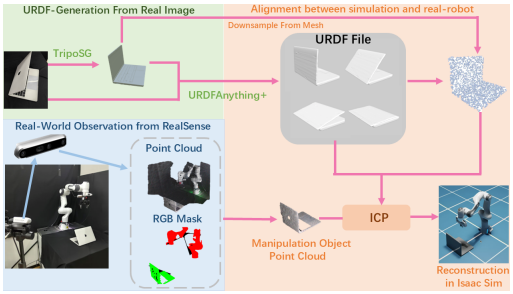
Objectives	Variation	Geometry Quality						Joint Accuracy		
		Parts			Whole Object					
		IoU ↑	F-Score ↑	CD ↓	IoU ↑	F-Score ↑	CD ↓	Axis Err. ↓	Origin Err. ↓	Limit ↓
Input Modality	Image Only	0.627	0.560	0.106	0.682	0.634	0.098	0.349	0.197	0.540
	Image + 3D Guid.(Ours)	**0.823**	**0.740**	**0.028**	**0.841**	**0.766**	**0.009**	**0.182**	**0.091**	**0.281**
Link Ordering	Random Order	0.630	0.686	0.145	0.710	0.704	0.072	0.203	0.168	0.442
	Order by Joint XYZ (Ours)	**0.805**	**0.742**	**0.096**	**0.816**	**0.753**	**0.034**	**0.120**	**0.102**	**0.323**

Inference Efficiency. Beyond reconstruction accuracy, we further evaluate the runtime cost of articulated object generation. The rightmost column of Table 1 reports the average inference time across all methods. We measure the wall-clock time elapsed from the moment model loading completes (i.e., all weights and pretrained components are resident in GPU memory) until the autoregressive generation terminates and the final URDF model is produced. We measure runtime on a single NVIDIA A800 GPU. As shown in the table, URDF-Anything+ achieves comparable or lower inference latency than previous multi-stage pipelines. Existing methods suffer higher overhead from repeated image API calls and reliance on large vision-language models (VLM), while our autoregressive generator predicts geometry and articulation together, avoiding external asset retrieval and symbolic reasoning.

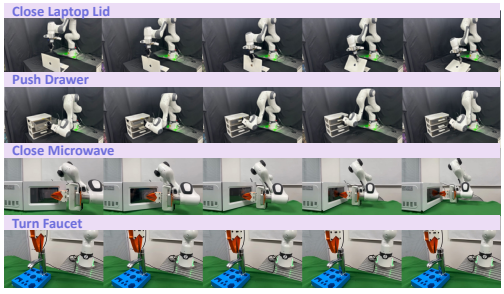
4.4 Ablation

Effect of Input Modalities. To validate our design choices, we study the effect of different input modalities on URDF-Anything+ using the ***faucet*** category, whose geometry is among the most complex across all object classes. This choice is motivated by the fact that faucets have complicated structures, making it particularly challenging to generate accurate geometry without explicit 3D whole-object guidance. All models are trained following the same two-stage protocol as the main setting: pretraining for 500 epochs without `[EoT]` or URDF supervision, followed by 100 epochs of fine-tuning with `[EoT]` and URDF parameter prediction enabled. The architecture and training schedule are kept identical across all variants, with differences only in the available input modalities. As shown in Table 2, using image input alone allows the model to infer articulation from visual cues but often results in ambiguities in fine-grained geometry and joint placement. Combining image and 3D inputs consistently achieves better performance across all evaluated metrics, including geometric quality and joint parameter accuracy.

Effect of Link Ordering. We study the impact of link ordering in the training data on articulated model generation using the refrigerator category, as a large portion of refrigerators consists of multiple articulated links. Specifically, we compare using a random permutation of links with a deterministic ordering based on the joint origin coordinates, where links are sorted by the (x, y, z) position of their associated joints in ascending order. The details are shown in Appendix Section A.4. Since our model follows an autoregressive generation paradigm, the ordering of links implicitly defines the factorization of the joint distribution. As shown in Table 2, the proposed spatially consistent ordering leads to more accurate geometry reconstruction and joint parameter prediction, indicating that a stable and physically meaningful ordering facilitates learning dependencies among articulated parts.



(a) Real-to-Sim digital twin construction: RGB-D capture, URDF-Anything+-generated URDF, ICP alignment, and Isaac Sim instantiation.



(b) Real-world manipulation tasks used for evaluation: closing a laptop lid and closing a drawer.

Figure 5: **Real-Follow-Sim deployment.** (a) Pipeline for constructing the simulation digital twin from real RGB-D observations. (b) The real-world tasks used for evaluation.

Table 3: **Real-World performance evaluation.** We compare our method across different deployment paradigms, URDF generation baselines, and training data sources. Success rates (%) are reported.

Comparison of deployment paradigms			*Comparison of URDF generation methods (under Real-Follow-Sim)*			*Comparison of training data sources*		
Method	**Close Laptop**	**Close Drawer**	**Method**	**Close Laptop**	**Close Drawer**	**Method**	**Close Laptop**	**Close Drawer**
Sim-to-Real	80	75	Articulate-Anything	75	55	Real-Only	85	75
Real-Follow-Sim (Ours)	**100**	**90**	**URDF-Anything+ (Ours)**	**100**	**90**	**Simulation-Only (Ours)**	**100**	**90**

4.5 Real-World Experiment

Real-Follow-Sim setup. For each trial, we capture RGB-D observations of the current scene, generate an articulated URDF with URDF-Anything+, register it to the observed point cloud using ICP [2], scale it to physical size, and instantiate it in Isaac Sim (Fig. 5a). We roll out the simulation-trained policy in this registered scene and execute the resulting trajectory on the real robot without real-world fine-tuning.

We validate the effectiveness of our framework in real-world manipulation tasks using a Franka robotic arm with a Robotiq Gripper [34]. A Realsense camera [12] is mounted on the side of the robotic arm to provide an ego-centric view of the workspace. For the real-robot tasks, we evaluate two core manipulation tasks: closing a laptop lid and closing a drawer (see Figure 5b). In the experiments, we consistently use Diffusion Policy [8] for robot policy learning. For a fair comparison, both real-world and simulated datasets contain 50 demonstration episodes per task. Simulated demonstrations are generated by a scripted expert policy for each object category, with randomized object poses and scene configurations. We report the success rate over 20 real-robot trials per method and task, using the same distributions of initial object poses and joint openings across methods. A trial succeeds if the laptop lid ends within 10° of fully closed or the drawer's remaining opening is less than 3 cm, depending on the task. A trial counts as a failure if the object is pushed out of the workspace or a safety stop is triggered.

Real-Follow-Sim VS. Sim-to-Real. Both methods use the same URDF, simulator, and training demonstrations. At deployment, Sim-to-Real runs the policy directly on the real robot, while Real-Follow-Sim rolls out the policy in the registered simulator and executes the resulting trajectory on the real robot. Table 3 reports the results.

Comparison with Other Baselines Performance. We compare the real-world performance of different URDF generation methods within our proposed Real-Follow-Sim framework. Table 3 shows that URDF-Anything+ outperforms Articulate-Anything in real-world robot performance.

Comparison with Real Data Training. In this section, we primarily compare the real-world performance of two policy training paradigms: (1) policies trained on a dataset collected directly on the real-world robot using a fixed amount of real-world data. (2) policies trained entirely in

simulation using an equally sized dataset generated efficiently under our Real-Follow-Sim framework. Following the Table 3, we find that our method outperforms URDF generation baselines, improves over the standard Sim-to-Real deployment paradigm, and enables simulation-only training to achieve strong real-world performance. These results show the effectiveness of Real-Follow-Sim as a unified framework for real-world manipulation.

5 Conclusion

We present **URDF-Anything+**, an end-to-end framework generating physically articulated URDFs from visual observations, outperforming prior methods in geometry, joint accuracy, and physical executability while running considerably faster. In the evaluated manipulation tasks, **Real-Follow-Sim** uses the generated URDFs to roll out simulation-trained policies in registered scenes and executes the resulting trajectories on the real robot.

Acknowledgments and Disclosure of Funding

This work was supported by the National Natural Science Foundation of China (62476011). This work was supported by the Beijing Natural Science Foundation (L252060).

References

[1] Ben Abbatematteo, Stefanie Tellex, and George Konidaris. Learning to generalize kinematic models to novel objects. In Leslie Pack Kaelbling, Danica Kragic, and Komei Sugiura, editors, *Proceedings of the Conference on Robot Learning*, volume 100 of *Proceedings of Machine Learning Research*, pages 1289–1299. PMLR, 2020.

[2] Paul J Besl and Neil D McKay. Method for registration of 3-d shapes. In *Sensor fusion IV: control paradigms and data structures*, volume 1611, pages 586–606. Spie, 1992.

[3] Ziang Cao, Zhaoxi Chen, Liang Pan, and Ziwei Liu. Physx-3d: Physical-grounded 3d asset generation. *arXiv preprint arXiv:2507.12465*, 2025.

[4] Ziang Cao, Fangzhou Hong, Zhaoxi Chen, Liang Pan, and Ziwei Liu. Physx-anything: Simulation-ready physical 3d assets from single image. *arXiv preprint arXiv:2511.13648*, 2025.

[5] 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 *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 5881–5892, 2025.

[6] Minghao Chen, Jianyuan Wang, Roman Shapovalov, Tom Monnier, Hyunyoung Jung, Dilin Wang, Rakesh Ranjan, Iro Laina, and Andrea Vedaldi. Autopartgen: Autoregressive 3D part generation and discovery. In *Proceedings of the 39th Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2025.

[7] Zoey Qiuyu Chen, Aaron Walsman, Marius Memmel, Kaichun Mo, Alex Fang, Karthikeya Vemuri, Alan Wu, Dieter Fox, and Abhishek Gupta. Urdformer: A pipeline for constructing articulated simulation environments from real-world images. *CoRR*, abs/2405.11656, 2024.

[8] Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 44(10-11):1684–1704, 2025.

[9] Daoyi Gao, Yawar Siddiqui, Lei Li, and Angela Dai. MeshArt: Generating articulated meshes with structure-guided transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 618–627, 2025.

[10] Jun Gao, Tianchang Shen, Zian Wang, Wenzheng Chen, Kangxue Yin, Daiqing Li, Or Litany, Zan Gojcic, and Sanja Fidler. Get3d: A generative model of high quality 3d textured shapes learned from images. *Advances in neural information processing systems*, 35:31841–31854, 2022.

[11] Yicong Hong, Kai Zhang, Jiuxiang Gu, Sai Bi, Yang Zhou, Difan Liu, Feng Liu, Kalyan Sunkavalli, Trung Bui, and Hao Tan. Lrm: Large reconstruction model for single image to 3d. In *International Conference on Learning Representations (ICLR)*, 2024.

[12] Intel Corporation. Intel®realsense™lidar camera l515. `https://www.intelrealsense.com/lidar-camera-l515/`, 2025. Accessed: 2025-02-01.

[13] Andrew Jaegle, Sebastian Borgeaud, Jean-Baptiste Alayrac, Carl Doersch, Catalin Ionescu, David Ding, Skanda Koppula, Daniel Zoran, Andrew Brock, Evan Shelhamer, et al. Perceiver io: A general architecture for structured inputs & outputs. *arXiv preprint arXiv:2107.14795*, 2021.

[14] Hanxiao Jiang, Yongsen Mao, Manolis Savva, and Angel X Chang. Opd: Single-view 3d openable part detection. In *European Conference on Computer Vision*, pages 410–426. Springer, 2022.

[15] Heewoo Jun and Alex Nichol. Shap-e: Generating conditional 3d implicit functions. *arXiv preprint arXiv:2305.02463*, 2023.

[16] 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 *The Thirteenth International Conference on Learning Representations*, 2025.

[17] Jiahui Lei, Congyue Deng, William B Shen, Leonidas Guibas, and Kostas Daniilidis. NAP: Neural 3d articulated object prior. In *Advances in Neural Information Processing Systems*, volume 36, pages 31878–31894, 2023.

[18] Yangguang Li, Zi-Xin Zou, Zexiang Liu, Dehu Wang, Yuan Liang, Zhipeng Yu, Xingchao Liu, Yuan-Chen Guo, Ding Liang, Wanli Ouyang, and Yan-Pei Cao. Triposg: High-fidelity 3d shape synthesis using large-scale rectified flow models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2025.

[19] 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 *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[20] Jiayi Liu, Denys Iliash, Angel X Chang, Manolis Savva, and Ali Mahdavi Amiri. SINGAPO: Single image controlled generation of articulated parts in objects. In *The Thirteenth International Conference on Learning Representations*, 2025.

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

[22] Jiayi Liu, Manolis Savva, and Ali Mahdavi-Amiri. Survey on modeling of human-made articulated objects. In *Computer Graphics Forum*, page e70092. Wiley Online Library, 2025.

[23] Jiayi Liu, Hou In Ivan Tam, Ali Mahdavi-Amiri, and Manolis Savva. Cage: Controllable articulation generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17880–17889, 2024.

[24] Minghua Liu, Ruoxi Shi, Linghao Chen, Zhuoyang Zhang, Chao Xu, Xinyue Wei, Hansheng Chen, Chong Zeng, Jiayuan Gu, and Hao Su. One-2-3-45++: Fast single image to 3d objects with consistent multi-view generation and 3d diffusion. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10072–10083, 2024.

[25] Minghua Liu, Chao Xu, Haian Jin, Linghao Chen, Mukund Varma T, Zexiang Xu, and Hao Su. One-2-3-45: Any single image to 3d mesh in 45 seconds without per-shape optimization. *Advances in Neural Information Processing Systems*, 36:22226–22246, 2023.

[26] Ruoshi Liu, Rundi Wu, Basile Van Hoorick, Pavel Tokmakov, Sergey Zakharov, and Carl Vondrick. Zero-1-to-3: Zero-shot one image to 3d object. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9298–9309, 2023.

[27] Xueyi Liu, Ji Zhang, Ruizhen Hu, Haibin Huang, He Wang, and Li Yi. Self-supervised category-level articulated object pose estimation with part-level SE(3) equivariance. In *The Eleventh International Conference on Learning Representations*, 2023.

[28] Yuan Liu, Cheng Lin, Zijiao Zeng, Xiaoxiao Long, Lingjie Liu, Taku Komura, and Wenping Wang. Syncdreamer: Generating multiview-consistent images from a single-view image. In *The Twelfth International Conference on Learning Representations*, 2024.

[29] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations (ICLR)*, 2019.

[30] Zhao Mandi, Yijia Weng, Dominik Bauer, and Shuran Song. Real2code: Reconstruct articulated objects via code generation. In *The Thirteenth International Conference on Learning Representations*, 2025.

[31] 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 *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 909–918, 2019.

[32] 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 *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 13001–13011, 2021.

[33] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 4195–4205, 2023.

[34] Robotiq Inc. Adaptive grippers. `https://robotiq.com/products/adaptive-grippers`, 2025. Accessed: 2025-02-01.

[35] Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. In *International Conference on Learning Representations*, 2022.

[36] Oriane Siméoni, Huy V. Vo, Maximilian Seitzer, Federico Baldassarre, Maxime Oquab, Cijo Jose, Vasil Khalidov, Marc Szafraniec, Seungeun Yi, Michaël Ramamonjisoa, Francisco Massa, Daniel Haziza, Luca Wehrstedt, Jianyuan Wang, Timothée Darcet, Théo Moutakanni, Leonel Sentana, Claire Roberts, Andrea Vedaldi, Jamie Tolan, John Brandt, Camille Couprie, Julien Mairal, Hervé Jégou, Patrick Labatut, and Piotr Bojanowski. Dinov3. *arXiv preprint arXiv:2508.10104*, 2025.

[37] Jiayi Su, Youhe Feng, Zheng Li, Jinhua Song, Yangfan He, Botao Ren, and Botian Xu. ArtFormer: Controllable generation of diverse 3d articulated objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1894–1904, 2025.

[38] Jiaxiang Tang, Jiawei Ren, Hang Zhou, Ziwei Liu, and Gang Zeng. Dreamgaussian: Generative gaussian splatting for efficient 3d content creation. In *International Conference on Learning Representations (ICLR)*, 2024.

[39] 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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3141–3150, 2024.

[40] Chenfei Wu, Jiahao Li, Jingren Zhou, Junyang Lin, Kaiyuan Gao, Kun Yan, Sheng-ming Yin, Shuai Bai, Xiao Xu, Yilei Chen, Yuxiang Chen, Zecheng Tang, Zekai Zhang, Zhengyi Wang, An Yang, Bowen Yu, Chen Cheng, Dayiheng Liu, Deqing Li, Hang Zhang, Hao Meng, Hu Wei, Jingyuan Ni, Kai Consultant Chen, Kuan Cao, Liang Peng, Lin Qu, Minggang Wu, Peng Wang, Shuting Yu, Tingkun Wen, Wensen Feng, Xiaoxiao Xu, Yi Wang, Yichang Zhang, Yongqiang Zhu, Yujia Wu, Yuxuan Cai, and Zenan Liu. Qwen-image technical report. *arXiv preprint arXiv:2508.02324*, 2025.

[41] Fanbo Xiang, Yuzhe Qin, Kaichun Mo, Yikuan Xia, Hao Zhu, Fangchen Liu, Minghua Liu, Hanxiao Jiang, Yifu Yuan, He Wang, et al. Sapien: A simulated part-based interactive environment. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11097–11107, 2020.

[42] Jianfeng Xiang, Xiaoxue Chen, Sicheng Xu, Ruicheng Wang, Zelong Lv, Yu Deng, Hongyuan Zhu, Yue Dong, Hao Zhao, Nicholas Jing Yuan Yuan, and Jiaolong Yang. Native and compact structured latents for 3d generation. *arXiv preprint arXiv:2512.14692*, 2025.

[43] 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 *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 21469–21480, 2025.

[44] Jiale Xu, Weihao Cheng, Yiming Gao, Xintao Wang, Shenghua Gao, and Ying Shan. Instantmesh: Efficient 3d mesh generation from a single image with sparse-view large reconstruction models. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2024.

[45] Taoran Yi, Jiemin Fang, Junjie Wang, Guanjun Wu, Lingxi Xie, Xiaopeng Zhang, Wenyu Liu, Qi Tian, and Xinggang Wang. Gaussiandreamer: Fast generation from text to 3d gaussians by bridging 2d and 3d diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6796–6807, 2024.

[46] Biao Zhang, Jiapeng Tang, Matthias Niessner, and Peter Wonka. 3dshape2vecset: A 3d shape representation for neural fields and generative diffusion models. *ACM Transactions on Graphics (TOG)*, 42(4):1–16, 2023.

[47] Bowen Zhang, Tianyu Yang, Yu Li, Lei Zhang, and Xi Zhao. Compress3d: a compressed latent space for 3d generation from a single image. In *European Conference on Computer Vision*, pages 276–292. Springer, 2024.

[48] Chuanrui Zhang, Minghan Qin, Yuang Wang, Baifeng Xie, Hang Li, and Ziwei Wang. Simart: Decomposing monolithic meshes into sim-ready articulated assets via mllm, 2026.

[49] Longwen Zhang, Ziyu Wang, Qixuan Zhang, Qiwei Qiu, Anqi Pang, Haoran Jiang, Wei Yang, Lan Xu, and Jingyi Yu. Clay: A controllable large-scale generative model for creating high-quality 3d assets. *ACM Transactions on Graphics (TOG)*, 43(4):1–20, 2024.

Appendix

This supplementary material provides additional details and results to complement the main paper. It includes the following sections:

A Data Preparation Details

A.1 Dataset Expansion Pipeline

To enhance data diversity, we expanded the **PartNet-Mobility** dataset by converting static models from **PartNet**[31] into articulated URDF formats. The pipeline consists of:

- **Automated Synthesis**: We parsed hierarchical JSON metadata and mesh geometries to infer kinematic chains. Contact analysis was performed to determine joint types, transformation matrices, and joint limits.
- **Human-in-the-loop Verification**: All synthesized models were manually inspected to ensure physical plausibility. We specifically corrected joint axes, motion ranges, and collision mesh alignments, while discarding models with topological defects.

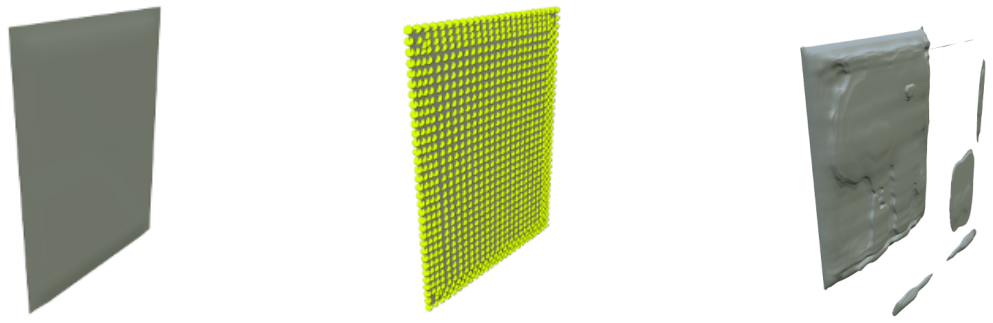
Meanwhile, the PhysX-3D[3] dataset also provides dynamic URDF annotations generated from PartNet. While the dataset contains a substantial amount of high-quality data, some joint annotations are erroneous. We therefore select a subset of the dataset and manually correct the joint parameters.

A.2 Normalization

To ensure a consistent input distribution, all objects underwent the following preprocessing: We apply standard mesh preprocessing commonly used in 3D geometry pipelines. Each mesh is uniformly scaled from its original resolution while preserving aspect ratios, then translated so that the center of its axis-aligned bounding box is aligned to $(0, 0, 0)$. After normalization, the mesh is guaranteed to lie within the unit cube $[-1, 1]^3$.

A.3 Mesh Thickening

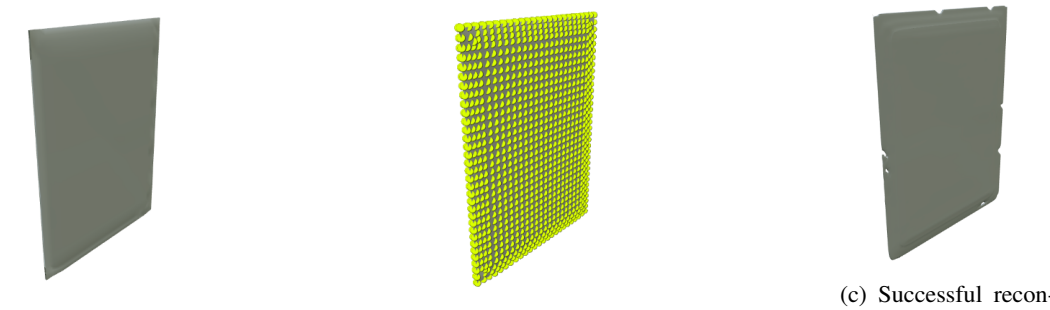
Our 3D encoding relies on the encoder from TripoSG[18], which operates on surface geometry and explicitly uses surface normals as input. However, a significant portion of meshes in PartNet and PhysX-3D are single-layer or zero-thickness surfaces. Such meshes often exhibit ill-defined or inconsistent normals, which in practice leads to unstable encoding and frequent reconstruction failures after decoding, which is shown in figure 6.



(a) Input mesh (b) Sampled points (c) Failed reconstruction

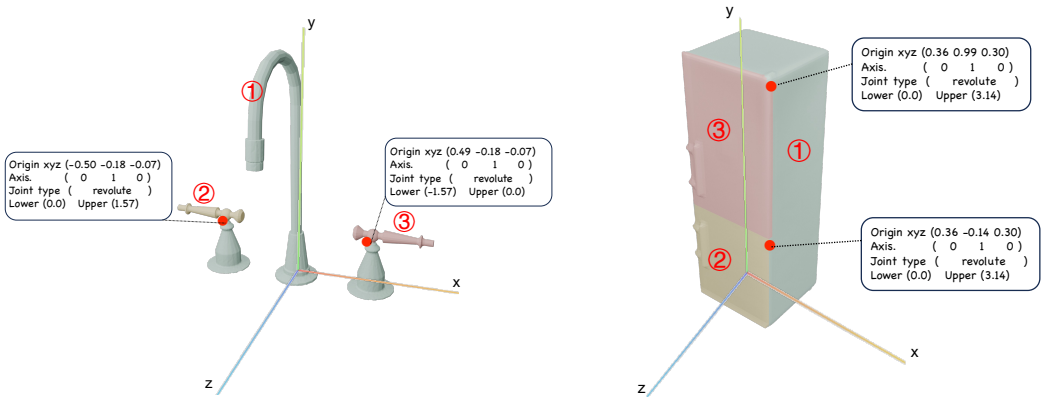
Figure 6: Failure case caused by single-layer meshes. From left to right: the input mesh, sampled surface points with unreliable normals, and the failed reconstruction after encoding and decoding.

To address this issue, we apply a mesh thickening preprocessing step to affected assets. Specifically, we convert single-layer surfaces into thin volumetric shells by offsetting the mesh along its normal



(a) Thickened mesh (b) Sampled points (c) Successful reconstruction

Figure 7: Effect of mesh thickening. From left to right: the thickened mesh, sampled surface points with well-defined normals, and the decoded reconstruction after encoding and decoding. Mesh thickening stabilizes normal estimation and enables reliable latent encoding.



(a) Example 1: Faucet (b) Example 2: Refrigerator

Figure 8: The visualization of link order setting

direction, thereby creating a small but non-zero thickness. This process yields well-defined, consistently oriented normals and ensures that the encoder–decoder pipeline can reliably reconstruct the geometry, which is shown in figure 7.

As a result, the encoded 3D latents correspond to valid and stable shape representations, which is critical for both diffusion-based generation and subsequent articulated model synthesis.

A.4 Ordering of parts

Our model takes a full image and a complete 3D shape as input and does not impose any explicit constraint on the generation order. To ensure that the model outputs parts in a consistent and semantically correct sequence during autoregressive generation, the training data must follow a predefined ordering scheme.

Specifically, for articulated objects, we designate immobile components such as the base or outer frame as the first part and require the model to generate them first. For the remaining movable parts, we follow the ordering strategy of AutoPartGen[6].

In the *automatic* setting, a fixed canonical order is used during training. All training assets are defined in a normalized canonical space, and parts are sorted lexicographically based on their axis-aligned bounding boxes: from bottom to top (Z), then left to right (X), and finally front to back (Y), following Blender's ZXY axis convention. Concretely, parts are first compared by their minimum Z values; if these are similar, minimum X values are compared, and if still similar, minimum Y values are used. The visualization is shown at figure 8.

The effect of this scheme is discussed in Section 4.4. Here is an example of our data format:

```
    {
    "id": "258",
    "whole_image": "base.png",
    "urdf": "test.urdf",
    "links": [
        {
            "name": "link_1",
            "obj": "link_1.obj"
        },
        {
            "name": "link_3",
            "obj": "link_3.obj",
            "origin_xyz": "-0.49765583872795105 -0.17762422561645508
           -0.06716569513082504",
            "axis_xyz": "0.0 1.0 0.0",
            "motion_type": "revolute",
            "lower": 0.0,
            "upper": 1.5707963267948966
        },
        {
            "name": "link_2",
            "obj": "link_2.obj",
            "origin_xyz": "0.48748037219047546 -0.1764167696237564
           -0.06509745121002197",
            "axis_xyz": "0.0 1.0 0.0",
            "motion_type": "revolute",
            "lower": -1.5707963267948966,
            "upper": 0.0
        }
    ]
}
```


A.5 Rendered Images

To improve the generalization of our model to real-world visual inputs, we augment the training data with rendered images. Specifically, we leverage the open-source image editing model *Qwen-Image-Edit* [40] to transform object images from the PartNet dataset into images that more closely resemble objects observed in real-world scenes. The examples are shown in figure 9 By enhancing visual



(a) Image from PartNet (b) Rendered Image (c) Image from PartNet (d) Rendered Image

Figure 9: Examples of rendered images generated by Qwen-Image-Edit

realism in terms of lighting, background, and appearance, this rendering process reduces the domain gap between synthetic training data and natural images, thereby improving the robustness of our model when conditioned on input images at inference time.

B Implementation Details

B.1 Training

3D Variational Autoencoder. We directly adopt the pretrained 3D VAE from TripoSG [18] without further training. The VAE encodes geometry using a signed distance function (SDF) representation and operates on dense surface samples. During encoding and decoding, we uniformly sample 204,800 points per shape and represent each shape using a fixed token length of 512. This pretrained VAE provides stable and high-quality 3D latent representations, which are used throughout diffusion training and inference.

Pretraining and fine-tuning. For diffusion training, we follow a two-stage strategy. We first train an image-to-3D diffusion model without URDF parameters or End-of-Token (`[EoT]`) generation. The diffusion backbone follows DiT [33], consisting of 24 transformer layers with a hidden dimension of 2048. Image conditions are encoded using the DINO-v3 ViT-H/16+ model pretrained on LVD-1689M (`dinov3-vith16plus-pretrain-lvd1689m`), which has approximately 0.8B parameters. The geometry condition is provided by the pretrained TripoSG 3D VAE. To reduce computational cost during diffusion training, we precompute the 3D latent representations for all object parts and whole objects (denoted as $\boldsymbol{z}_{\text{pre}}^{[1:k]}$ and $\boldsymbol{z}_{\text{whole}}$) by running the 3D VAE offline, and cache the resulting latents for efficient reuse. This stage is trained for 500 epochs on 16 NVIDIA A800 GPUs.

We then fine-tune the model for an additional 100 epochs by incorporating URDF parameters and `[EoT]` generation into the training objective. Unless otherwise specified, all other training settings, optimization strategies, and diffusion configurations remain identical to those used in the initial training stage.

Training loss configuration and hyperparameter settings During the pretraining stage, the diffusion model is optimized solely using the latent diffusion objective $\mathcal{L}_{\text{diff}}$ defined in Eq. 4, which supervises the denoising velocity prediction in latent space. In the fine-tuning stage, we extend the training objective to jointly supervise articulated structure generation by incorporating additional losses. Specifically, the total loss is given by

$$\mathcal{L} = \mathcal{L}_{\text{diff}} + 0.1 * \mathcal{L}_{\text{EOT}} + 0.01 * \mathcal{L}_{\text{joint}},$$

where $\mathcal{L}_{\text{EOT}}$ denotes the End-of-Token (`[EoT]`) prediction loss, and $\mathcal{L}_{\text{joint}}$ aggregates losses for joint parameter prediction. For continuous joint parameters (e.g. joint origin, axis direction, and limits), we use ℓ_1 loss, while joint type prediction is supervised using a categorical cross-entropy loss.

Notably, for a large portion of articulated objects, the joint rotation axes are aligned with canonical directions (e.g., $(0, 0, 1)$, $(0, 1, 0)$, or $(1, 0, 0)$). In such cases, the coordinate of the joint origin along the axis direction does not affect the kinematic behavior of the joint. To avoid introducing unnecessary noise into training, we mask out the origin component along the predicted joint axis when computing the origin ℓ_1 loss. This axis-aware masking focuses supervision on the physically meaningful degrees of freedom and leads to more stable and accurate joint parameter learning.

All models are optimized using AdamW [29] with a learning rate of 1×10^{-5} during fine-tuning, while other optimization settings follow those used in pretraining.

B.2 Inference

At inference time, the articulated object is generated in an autoregressive manner until a termination condition is met. We explicitly introduce an End-of-Token (`[EoT]`) signal in the latent space to indicate the end of part generation. Specifically, the `[EoT]` target is defined as a zero vector with the same dimensionality as the shape latent, i.e., a zero latent that matches the geometry latent representation.

During inference, after each autoregressive step, we measure the distance between the predicted latent and the `[EoT]` latent. If this distance falls below a predefined threshold, the generation process is terminated, indicating that no further valid parts remain to be generated.

In addition, we incorporate a geometric redundancy check to prevent duplicate or degenerate part generation. After decoding the current part geometry, we evaluate its overlap and similarity with the

previously generated part. If the overlap exceeds a predefined similarity threshold, indicating that the newly generated link is largely redundant with an existing one, the generation is also terminated.

B.3 Evaluation Metrics

We use three standard metrics to evaluate the reconstruction quality: Intersection over Union (IoU), F-score, and Chamfer Distance (CD).

Intersection over Union (IoU). IoU measures the volumetric overlap between the predicted shape P and the ground-truth shape G:

$$\text{IoU} = \frac{|P \cap G|}{|P \cup G|} \tag{12}$$

where $|\cdot|$ denotes the volume (number of occupied voxels or mesh volume) of the shape.

F-score. F-score measures the geometric similarity between two point sets P (predicted) and G (ground truth) based on a distance threshold τ. In our experiments, we set $\tau = 0.02$:

$$\text{Precision} = \frac{|\{p \in P \mid d(p, G) < \tau\}|}{|P|}, \tag{13}$$

$$\text{Recall} = \frac{|\{g \in G \mid d(g, P) < \tau\}|}{|G|}, \tag{14}$$

$$\text{F-score} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}, \tag{15}$$

where $d(p, G) = \min_{g \in G} \|p - g\|_2$ is the distance from a predicted point to the nearest ground-truth point.

Chamfer Distance (CD). Chamfer Distance measures the average closest-point distance between two point sets:

$$\text{CD}(P, G) = \frac{1}{|P|} \sum_{p \in P} \min_{g \in G} \|p - g\|_2^2 + \frac{1}{|G|} \sum_{g \in G} \min_{p \in P} \|g - p\|_2^2 \tag{16}$$

This metric captures the overall similarity between two point clouds.

C Visualization of Generated Shapes Across Object Categories

To qualitatively evaluate the performance of our method, we visualize the generated shapes for different object categories.

D Limitations

While URDF-Anything+ achieves strong performance in generating articulated object models and enabling the Real-Follow-Sim pipeline, several limitations remain. First, the diversity of articulated objects in the training data is still limited. Our model primarily relies on publicly available datasets, where certain object categories contain relatively few instances. This data imbalance may restrict the model's ability to generalize to rare or unseen articulated structures. Expanding the dataset with more diverse articulated objects and richer annotations would likely further improve performance and robustness.

Second, although our method predicts geometric structure and kinematic parameters, it does not explicitly model more complex physical properties of objects, such as mass distribution, friction, or material-dependent dynamics. In simulation, we use each generated link mesh for both visual

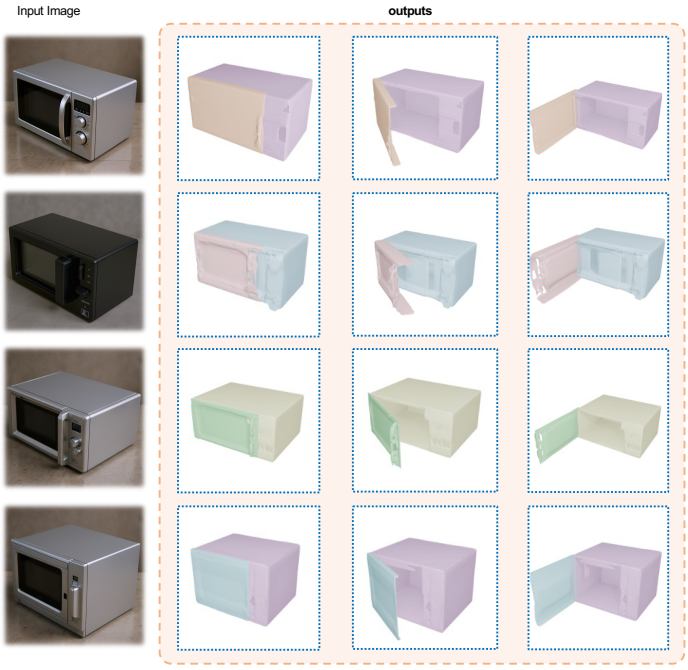


Figure 10: Generation Results of Microwaves.

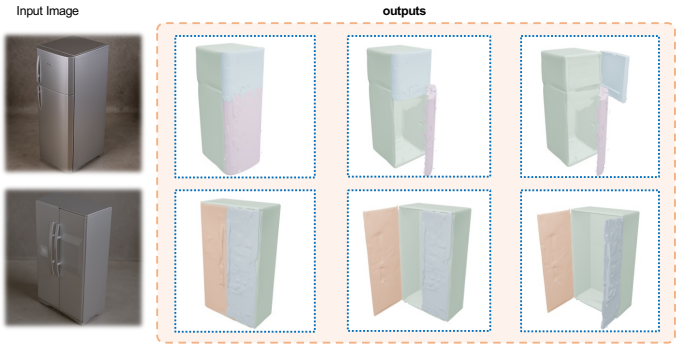


Figure 11: Generation Results of Refrigerators.

and collision geometry, compute mass and inertia from the mesh assuming uniform density, and use the simulator's default friction coefficient. As a result, some physical behaviors in simulation may not perfectly match those in the real world. Incorporating richer physical parameter estimation or physics-aware learning could further improve the fidelity of the generated digital twins.

E Broader Impact

Our work focuses on automatically generating simulation-ready articulated object models from images, which lowers the barrier for building physics-based simulation environments and digital twins for robot manipulation research. By reducing manual asset construction effort, this technology can broaden access to large-scale embodied learning, particularly for groups without dedicated 3D modeling resources, and may accelerate research in sim-to-real transfer, household robotics, and interactive virtual environments.

Potential negative impacts are limited but worth acknowledging. Generated digital twins may not faithfully capture safety-critical physical properties such as mass distribution, friction, or material strength; deploying policies trained on such twins to real robots in safety-critical settings without

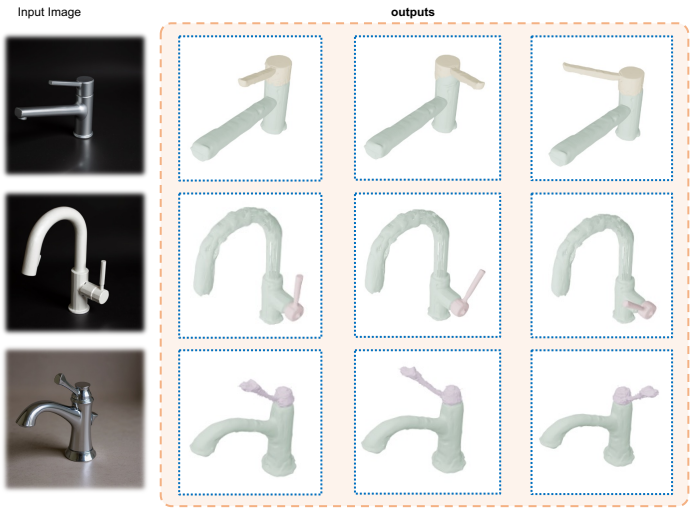


Figure 12: Generation Results of Faucets.

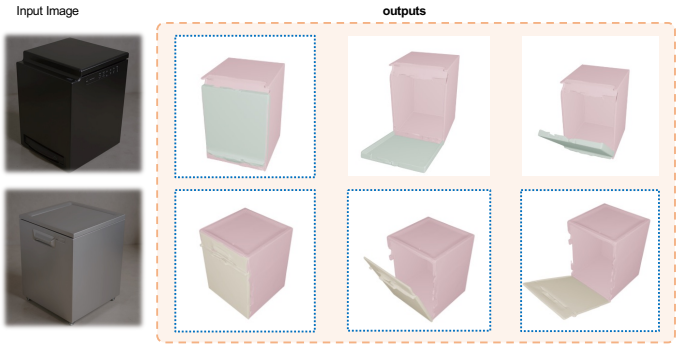


Figure 13: Generation Results of Dishwashers.

further verification could lead to unintended behaviors. We therefore recommend that downstream users validate generated models—and the policies trained on them—before applying them to high-stakes manipulation tasks.

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 three contributions stated in the abstract and Section 1—the URDF-Anything+ end-to-end articulated URDF generator, the Real-Follow-Sim deployment scheme, and the empirical gains in geometry, joint accuracy, and physical executability—are directly substantiated by the experiments in Section 4.

   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: We provide discussion of limitations in Appendix D.

   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: The model architecture, two-stage training schedule, optimizer, and supervision targets are described in Section 4; dataset construction, network hyperparameters, and the inference protocol are provided in Appendix A, Appendix B.1, and Appendix B.2.

   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: We will publicly release the code, model weights, and dataset preparation scripts upon acceptance.

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: Training data, training stages, and the evaluation protocol are described in Section 4; full hyperparameters, optimizer settings, and loss configuration are reported in Appendix B.1.

   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: [Yes]

   Justification: Joint accuracy results in Table 1 are reported as mean$\pm$std across the test set; geometric metrics (IoU, F-Score, Chamfer Distance) are aggregated over the same evaluation set.

   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: Training is performed on 16 NVIDIA A800 GPUs as described in Section 4 and Appendix B.1; per-method inference time is measured on a single A800 GPU and reported in Section 4.3 (Inference Efficiency) and the rightmost column of Table 1.

   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 research conducted in this paper conforms to 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: We discuss broader impacts of our work in Appendix E.

    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 does not present any 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: All third-party assets used in this work, including PartNet [31], PartNet-Mobility [41], PhysX-3D [3], TripoSG [18], DINOv3 [36], and Qwen-Image-Edit [40], are cited where they appear, and their licenses and terms of use are respected.

    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: [Yes]

    Justification: The paper introduces a new generative model (URDF-Anything+) and an expanded articulated dataset built on PartNet, PartNet-Mobility, and PhysX-3D. Their construction, supervision, and training/inference procedures are documented in Section 4, Appendix A, and Appendix B.1; the assets will be released through our project page upon acceptance.

    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 experiments or 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 study participants or human-subject research.

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: [N/A]

    Justification: LLMs are not used as an important, original, or non-standard component of the core methodology.

    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.