FROM PARTS TO SIM: BOTTOM-UP AUTOREGRESSIVE GENERATION OF SIM-READY 3D ASSETS

Anonymous authors
Paper under double-blind review



Figure 1: Our model generates simulation-ready articulated assets with fine-grained geometric components, rigid kinematic links, and the physical attributes required for simulation. The generated assets can be exported directly to USD and URDF, where they exhibit physically plausible articulated motion. Interaction trajectories collected from these assets are further used to train manipulation policies that transfer from simulation to real-world objects.

ABSTRACT

Recent advances in 3D generation have enabled high-quality object synthesis, yet most generated assets remain static, limiting their use in physical simulation and embodied learning. A simulation-ready articulated asset must specify not only its geometry and appearance, but also how its components are organized, how they move, and which physical properties govern their interactions. Existing approaches often recover articulation after geometry generation or represent geometric parts and kinematic links at the same granularity, making it difficult to preserve fine details while maintaining a compact physical structure. We present a bottom-up autoregressive framework that models fine-grained geometric components and rigid kinematic links as distinct but connected levels, enabling the unified generation of geometry, articulation, and simulation-relevant physical attributes, including joint dynamics. To address the scarcity of fully annotated articulated data, we develop a scalable data pipeline and co-train the model using large-scale part-aware geometry and simulation-ready assets. Experiments demonstrate state-of-the-art performance in geometric fidelity, articulation accuracy, and simulation executability. Furthermore, manipulation policies trained on trajectories collected from our generated assets transfer successfully to real-world objects, demonstrating their value for sim-to-real robot learning.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

1 INTRODUCTION

Physics simulation provides scalable, reproducible, and controllable interaction data for embodied learning (Xiang et al., 2020; Makoviychuk et al., 2021), but its utility depends on the availability of simulation-ready assets. Unlike static 3D models, articulated assets must specify which components move together, how the resulting rigid bodies are connected, and which physical properties govern their interactions. Manually authoring this information requires substantial geometric and mechanical expertise, creating a major bottleneck for simulation-based robot learning.

Recent datasets have improved the diversity and physical fidelity of articulated objects (Xiang et al., 2020; Geng et al., 2023; Jin et al., 2026; Iliash et al., 2026), while reconstruction and generative methods increasingly model object mobility (Chen et al., 2024; Le et al., 2025; Li et al., 2025d; Wang et al., 2026; Zhang et al., 2026; Liu et al., 2024; Su et al., 2025; Wu et al., 2026). Existing approaches, however, typically either recover articulation from completed geometry or use the same hierarchy for geometric parts and rigid links. The former must rediscover small functional components from an assembled shape, whereas the latter couples geometric resolution to kinematic complexity. Consequently, preserving fine detail may produce unnecessarily complex physical structures, while simplifying the structure may remove functionally important components.

Our key observation is that geometric and kinematic decompositions need not have the same granularity. A drawer, for example, may contain several visible components that move together as one rigid link, while a small latch may form an independent link. We therefore represent articulated objects using a bottom-up hierarchy: fine-grained components preserve geometric detail, rigid links group components that share motion, and joints define the kinematic and physical relationships between links. This separation preserves small functional elements without unnecessarily increasing the complexity of the kinematic structure.

Building on MegaParts (Liao et al., 2026), we introduce a structured autoregressive framework that generates this hierarchy in a unified sequence. The model first generates component-level geometry, organizes the components into rigid links, and then predicts the articulation structure and simulator-facing attributes. In addition to geometry, link organization, and joint kinematics, our representation includes joint damping and stiffness, allowing dynamic behavior to be generated rather than assigned through simulator defaults. To our knowledge, this is the first learned conditional framework to jointly generate these joint dynamics with component geometry, component-to-link grouping, and articulation.

To address the scarcity of fully specified training assets, we develop Mobility-PhyX, a scalable data-construction pipeline built upon Infinite Mobility (Lian et al., 2025). Mobility-PhyX expands geometric, kinematic, and physical diversity through geometric augmentation and physical-attribute grounding. We combine its generated assets with augmented open-source datasets through heterogeneous co-training, producing a corpus of approximately 430K part-aware objects and 30K simulation-ready assets. Part-aware data provide component-geometry supervision, while simulation-ready assets supervise rigid-link organization, articulation, and physical attributes.

We evaluate generated assets in terms of geometric fidelity, link and joint correctness, physical attributes, and simulator compatibility. We also collect manipulation trajectories from generated assets to train a vision-language-action policy, which is evaluated in simulation and on real objects excluded from trajectory collection. Our contributions are threefold:

- We introduce a bottom-up representation and structured autoregressive model that distinguish fine-grained components from rigid links and jointly generate geometry, link organization, articulation, and physical attributes.
- We develop Mobility-PhyX and a heterogeneous co-training strategy combining approximately 430K part-aware objects with 30K simulation-ready assets.
- We establish an evaluation spanning geometric quality, structural correctness, physical validity, simulator compatibility, and downstream sim-to-real policy learning.

2 Related Works

2.1 Representations for Structured and Articulated 3D Objects

Neural implicit fields and VecSet representations enable high-quality 3D synthesis but encode an object holistically, causing global structure and fine details to compete for capacity (Poole et al., 2022; Lin et al., 2023; Wang et al., 2023; Zhang et al., 2023; Zhao et al., 2023; Zhang et al., 2024a; Yang et al., 2024a; Zhao et al., 2025; Li et al., 2025b;a; Wu et al., 2024; Chen et al., 2025a; Li et al., 2025c; Lai et al., 2025b). Voxel methods improve locality but allocate capacity by spatial subdivision rather than object structure (Xiang et al., 2024; 2025; Wu et al., 2025; Chen et al., 2025b; He et al., 2025; Lai et al., 2025a; Luo et al., 2026; Deng et al., 2025). Part-aware representations instead model meaningful components (Liao et al., 2026). Because geometric parts need not correspond to rigid links, we use fine-grained components for geometry and rigid links for shared motion and physical properties, decoupling geometric detail from kinematic complexity.

2.2 Simulation-Ready Asset Generation

Simulation-ready generation requires geometry, rigid-link hierarchies, joints, and physical parameters. Existing methods primarily recover articulation from observations or meshes (Chen et al., 2024; Mandi et al., 2024; Le et al., 2025; Li et al., 2025d; Qiu et al., 2025; Zhang et al., 2026; Wang et al., 2026; Wu et al., 2026). Generative methods learn articulated-object priors through diffusion, autoregressive modeling, or procedural programs (Lei et al., 2023; Liu et al., 2024; Su et al., 2025; Gao et al., 2025; Lian et al., 2025; Zhou et al., 2026). PhysX-Anything and PhysX-Omni additionally generate broader physical attributes (Cao et al., 2026a;b), but joint dynamics such as damping and stiffness remain largely unexplored.

2.3 Simulation-Ready Asset Data Sources

ShapeNet, Objaverse, PartNet, and GAPartNet provide diverse geometry and part annotations but generally lack articulation and physical properties (Chang et al., 2015; Deitke et al., 2022; Mo et al., 2019; Geng et al., 2023). PartNet-Mobility adds link and joint structures, while ArtVIP, Artiverse, PhysX-Mobility, and PhysXVerse provide broader simulator-facing annotations (Xiang et al., 2020; Jin et al., 2026; Iliash et al., 2026; Cao et al., 2026a;b). Such assets remain costly to author. Procedural systems improve scale (Lian et al., 2025; Joshi et al., 2025; Sun et al., 2025; Zhou et al., 2026), but their diversity is constrained by predefined rules and components, leaving a trade-off among scale, structural diversity, and physical completeness.

3 Preliminaries

3.1 Simulation-Ready Assets

An articulated asset is conventionally represented as a set of rigid links connected by joints (Xiang et al., 2020). Let $\mathcal{L} = \{\ell_k\}_{k=1}^{K}$ denote its K rigid links and $\mathcal{K} = (\mathcal{L}, \mathcal{E}, r)$ its rooted kinematic tree, where $r \in \mathcal{L}$ is the root link and each directed edge $e \in \mathcal{E}$ defines a joint from a parent link to a child link. Each link is associated with visual and collision geometry, together with the inertial, material, and contact properties of the corresponding rigid body. Each joint specifies its type, parent and child frames, motion axis, limits, and dynamic parameters such as damping and stiffness.

Given a joint configuration $\mathbf{q}$, forward kinematics determines the pose $\mathbf{T}_k(\mathbf{q})$ of each link. Let $\mathcal{G}_k^0$ denote the geometry of link k, expressed in the canonical object frame at the rest configuration $\mathbf{q} = \mathbf{0}$. The articulated geometry at configuration $\mathbf{q}$ is then

$$\mathcal{G}(\mathbf{q}) = \bigcup_{k=1}^{K} \mathbf{T}_k(\mathbf{q})\mathbf{T}_k(\mathbf{0})^{-1}\mathcal{G}_k^0. \tag{1}$$

We use simulation-ready to describe an asset that can be instantiated in a physics simulator without further structural or dynamic authoring. Concretely, the asset must specify its geometry shape, rigid-body organization, link-level inertial, and joint-level kinematic and dynamic parameters.

3.2 Component-Level Geometry Representation

We adopt the component-level representation and geometry generator of MegaParts (Liao et al., 2026) as the geometric foundation of our framework. MegaParts represents a complex object as a variable-size collection of fine-grained geometric components, generating their spatial layout before decoding their local geometry. We summarize the interface required by our method and refer readers to the original work for details of the geometry tokenizer, architecture, and training objectives.

Let

$$\mathcal{C} = \{c_i\}_{i=1}^{N}, \qquad c_i = (\mathbf{b}_i, \mathbf{z}_i)\,, \tag{2}$$

denote the generated collection of N geometric components, where N may vary across objects. Here, $\mathbf{b}_i$ specifies the spatial placement and extent of component i in the canonical object frame, while $\mathbf{z}_i$ is its local geometry code. Let $\mathcal{D}_{\text{geo}}$ denote the component geometry decoder and $\Gamma(\mathbf{b}_i, \cdot)$ the transformation from the normalized local component frame to the canonical object frame. The decoded rest-pose geometry of component i is

$$\mathcal{M}_i^0 = \Gamma(\mathbf{b}_i, \mathcal{D}_{\text{geo}}(\mathbf{z}_i))\,. \tag{3}$$

Importantly, $\mathcal{C}$ defines a geometric decomposition rather than a kinematic one. Multiple components may undergo exactly the same rigid transformation and should therefore belong to a single rigid link. Treating every fine-grained component as an independent link would unnecessarily couple geometric resolution to kinematic complexity. Our method instead learns how the N components are grouped into K rigid links, with $K \leq N$, and predicts the articulation and physical attributes over the resulting link structure. Please refer to Appendix A for more details.

4 Mobility-PhyX

Existing articulated asset datasets are limited in scale due to the substantial manual effort while recent procedural generation pipeline (Lian et al., 2025) still exhibits limited diversity in part geometry and articulation structures and lacks physical attributes required for simulation and interaction. To address these limitations, We introduce **Mobility-PhyX**, a procedural and physically grounded pipeline for generating simulation-ready articulated assets.

As illustrated in Fig. 2, our pipeline consists of two major components: a Dual-level Data Augmentation strategy that improves geometric and kinematic diversity, and a VLM-Assisted Coarse-to-Fine Physical Attribute Grounding pipeline that assigns physically plausible properties to generated assets. Together, these components enable the construction of diverse articulated assets with complete physical specifications. Detailed implementation is provided in Appendix E.

4.1 Dual-Level Data Augmentation

Geometric shapes and underlying kinematic structures are two important factors for the diversity of articulated assets. However, procedural generation pipelines typically rely on a limited set of manually designed meshes and predefined articulation templates, resulting in repetitive part geometries and similar kinematic configurations. To improve both geometric and structural diversity, Mobility-PhyX introduces two complementary augmentations.

First, we perform retrieval-based part geometry augmentation by retrieving semantically corresponding parts from external 3D asset collections and replacing selected components of procedural assets. The retrieved geometries are aligned with the procedural parts through category-specific support points, preserving their semantic and functional roles while introducing richer shape variations.

Second, we introduce Bidirectional Kinematic-Tree Augmentation to modify articulation structures. Given a selected joint in the URDF tree, Mobility-PhyX supports two replacement directions: child subtree replacement and parent tree replacement. The transferred structure provides link geometries, parent-child hierarchy, semantic organization, and joint motion types, while geometry-dependent placement and joint parameters are reconstructed using the procedural generation process. This design enables diverse articulation structures while maintaining valid and executable articulated assets. Detailed implementation of the dual-level augmentation strategy is provided in Appendix E.1.

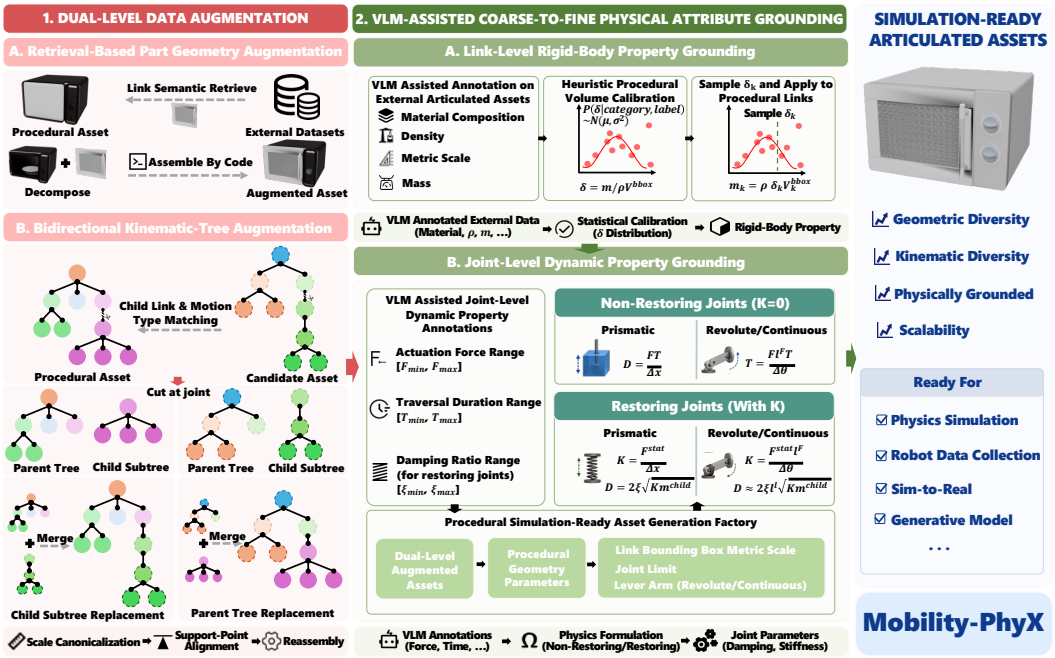

Figure 2: **Overview of our Mobility-PhyX data pipeline.** Mobility-PhyX consists of two major components: dual-level data augmentation strategy and a VLM-Assisted Coarse-to-Fine Physical Attribute Grounding pipeline. With these two carefully design, Mobility-PhyX is able to produce diverse and scalable simulation-ready assets.

4.2 VLM-Assisted Coarse-to-Fine Physical Attribute Grounding

To further construct simulation-ready assets, Mobility-PhyX assigns physical attributes at both the link level and joint level. For rigid links, we obtain material composition, density, and mass through a VLM-assisted annotation pipeline on external articulated assets. Since direct mesh volume computation is unreliable, we introduce a heuristic volume calibration procedure. The calibrated volume factor is then used to estimate the mass of procedurally generated links according to their material properties and dimensions.

For articulated joints, Mobility-PhyX further assigns dynamic attributes including stiffness and damping. The VLM provides plausible interaction force and motion duration annotations according to object category, joint type, and functional role. These annotations are converted into joint dynamic parameters through physics-based formulations. Specifically, we distinguish between non-restoring joints and restoring joints, and derive their corresponding stiffness and damping parameters using different physical formulations. Details of this process are provided in Appendix E.2.

Through the combination of dual-level data augmentation and physical attribute grounding, Mobility-PhyX generates articulated assets with diverse geometries, rich articulation structures, and physically plausible simulation parameters. These assets further facilitate our generation model training and can also be directly deployed in physics simulators for multiple downstream tasks.

5 A Complete Generation Pipeline for Sim-Ready 3D Assets

Generating a simulation-ready 3D asset requires modeling both its visual appearance, including mesh geometry and texture, and the physical specification that governs its motion and interactions. Figure 3 provides an overview of our framework, which combines heterogeneous asset supervision with a structured autoregressive representation that generates fine-grained components, groups them into rigid links, and connects the links through physically parameterized joints. In this section, we first introduce our structured representation of simulation-ready assets and then describe how heterogeneous data sources are combined to train a unified generative model. To complement the generated geometry with a coherent surface appearance, we further develop a part-aware texturing pipeline based on Xiang et al. (2025), as detailed in Section B.

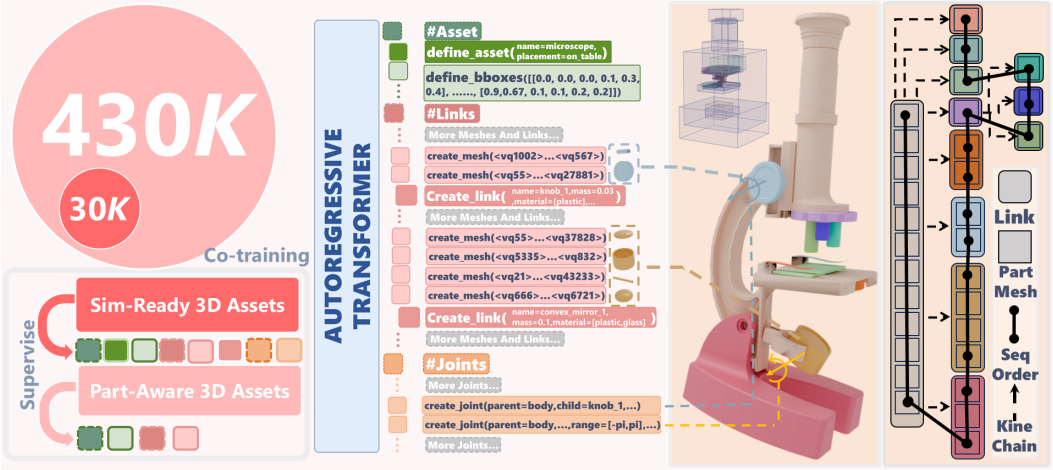

Figure 3: **Overview of our bottom-up simulation-ready asset generation framework. Left:** We co-train on 430K part-aware assets and 30K simulation-ready assets, with each source supervising its available structural levels. **Middle:** A structured autoregressive transformer generates the asset layout, fine-grained component meshes, rigid-link groupings and attributes, and joint records in a unified sequence. **Right:** Components are ordered by connection-aware spatial DFS within each link, while links follow the kinematic-tree traversal, producing an explicit hierarchy that separates geometric detail from rigid-body structure.

5.1 Structured Bottom-Up Representation for Simulation-Ready Assets

The component collection $\mathcal{C}$ introduced in Section 3 captures the detailed rest-pose geometry of an object, but does not specify which components move together or how their motion is constrained. Treating every geometric component as an independent rigid body would couple geometric resolution to kinematic complexity. We therefore introduce an intermediate rigid-link level and represent a simulation-ready asset as

$$\mathcal{A} = \left(\mathcal{C}, \mathcal{L}, \pi, \mathcal{K}, \Phi^L, \Phi^J\right), \tag{4}$$

where $\mathcal{L} = \{\ell_k\}_{k=1}^K$ is the set of rigid links, $\pi : [N] \to [K]$ assigns the N geometric components to links, $\mathcal{K} = (\mathcal{L}, \mathcal{E}, r)$ is a rooted kinematic tree, and Φ^L and Φ^J contain the link- and joint-level physical attributes, respectively. In particular, for $I_k = \pi^{-1}(k)$, the canonical geometry of link ℓ_k is $G_k^0 = \bigcup_{i \in I_k} M_i^0$. Because every component belongs to exactly one link and every link contains at least one component, $K \leq N$. This separation allows a geometrically detailed rigid body to contain many components without introducing unnecessary degrees of freedom.

Autoregressive generation additionally requires a deterministic ordering of the otherwise unordered components. We construct a connection-aware spatial ordering from mesh contacts and component sizes. Part-aware assets are serialized using a global spatial traversal, whereas simulation-ready assets are traversed first according to their kinematic tree and then spatially within each link. Disconnected components are handled using an auxiliary contraction procedure that affects only the ordering graph and leaves the original geometry unchanged. The complete construction is described in Appendix C.

Following this hierarchy, the model serializes a simulation-ready asset as

$$\begin{aligned} \mathbf{s}_{\mathcal{A}} = \big[&\langle \text{SIM-READY} \rangle, \langle \text{ASSET} \rangle, \mathbf{s}_{\text{layout}}, \langle \text{LINKS} \rangle, \\ &\left(\mathbf{s}_{\ell_k}^{\text{geo}}, \text{CreateLink}(\phi_k^L)\right)_{k=1}^K, \langle \text{JOINTS} \rangle, \\ &\left(\text{CreateJoint}(\ell_p, \ell_c, \phi_e^J)\right)_{e \in \mathcal{E}}, \langle \text{EOS} \rangle\big], \end{aligned} \tag{5}$$

where links are visited in kinematic depth-first order and $\mathbf{s}_{\ell_k}^{\text{geo}}$ contains the link's components in connection-aware spatial order. Each CreateLink record closes the preceding component block and specifies its link-level attributes; the subsequent CreateJoint records connect the generated links and specify their kinematic and dynamic properties. This produces an explicit, code-like asset description in which physical attributes can be inspected or edited.

5.2 Scalable Co-Training with Heterogeneous Asset Supervision

Complete simulation-ready assets are substantially more expensive to construct than part-aware geometric assets because they additionally require rigid-link grouping, articulation, and physical parameters. We therefore train with approximately 430K part-aware assets and 30K simulation-ready assets. The former provide broad geometric and categorical coverage, while the latter supervise the complete component-to-link hierarchy, kinematic structure, and physical attributes.

The two data sources follow different output grammars. A part-aware sequence uses a global connection-aware ordering and terminates after component generation, whereas a simulation-ready sequence contains link boundaries, physical attributes, and joint records. We distinguish these tasks with a task token $\tau_d \in \{\langle \text{PART-AWARE} \rangle, \langle \text{SIM-READY} \rangle\}$ and model

$$p_\theta(\mathbf{s}_d \mid x, \tau_d) = \prod_{t=1}^{T} p_\theta\left(s_t \mid \mathbf{s}_{<t}, x, \tau_d\right), \tag{6}$$

where x is a textual object description or a reference RGB image in our experiments. The task token prevents conflicting stopping behavior: part-aware examples terminate after geometry generation, while simulation-ready examples continue with link and joint records. Both domains share the same component representation, geometry tokenizer, and local connection-aware transitions.

A standard token-averaged objective would be dominated by long component geometry spans, providing comparatively weak supervision to the sparse link, joint, and physical-attribute tokens. We therefore partition supervised positions into

$$\mathcal{F} = \{\text{layout}, \text{geometry}, \text{link}, \text{joint}\}$$

and average the autoregressive loss independently within each active token family:

$$\mathcal{L}_d(\mathbf{s}) = -\frac{1}{|\mathcal{F}_d|} \sum_{f \in \mathcal{F}_d} \frac{1}{|\Omega_f(\mathbf{s})|} \sum_{t \in \Omega_f(\mathbf{s})} \log p_\theta\left(s_t \mid \mathbf{s}_{<t}, x, \tau_d\right), \tag{7}$$

where $\mathcal{F}_{\text{part}} = \{\text{layout}, \text{geometry}\}$ and $\mathcal{F}_{\text{sim}} = \mathcal{F}$.

The complete sequence grammars, token-family definitions, and staged training schedule are provided in Appendix D.

6 Experiments

6.1 Implementation Details

Data recipes. We construct the simulation-ready corpus using the annotation pipeline described in Section 4. Its source assets are collected from existing articulated-object datasets (Xiang et al., 2020; Jin et al., 2026; Lian et al., 2025; Joshi et al., 2025; Gao et al., 2025; Cao et al., 2026b; Zhou et al., 2026; Iliash et al., 2026) and a small curated collection of structurally complex objects. The part-aware corpus is processed following MegaParts (Liao et al., 2026). Dataset sizes are summarized in Figure 3, with detailed statistics provided in Appendix F.

6.2 Quantitative and Qualitative Comparison

We compare with PhysX-Anything (Cao et al., 2026a), SINGAPO (Liu et al., 2025), Articraft (Zhou et al., 2026), SIMART (Zhang et al., 2026), ArtLLM (Wang et al., 2026), Particulate (Li et al., 2026), and URDF-Anything (Li et al., 2025d). Geometry-conditioned methods are evaluated separately because they receive a reference mesh or point cloud. Detailed protocols and metric definitions are provided in Appendix H.

Table 1 summarizes the quantitative results under both conditioning modalities. Under image conditioning, our method achieves the strongest overall performance in geometry reconstruction and asset-attribute prediction. Under text conditioning, it improves geometric quality and text alignment while maintaining competitive kinematic validity, demonstrating robust generation beyond image-based reconstruction. Structural and physical evaluations further show that the generated assets are reliably loadable and exhibit accurate joint types, axes, origins, and limits.

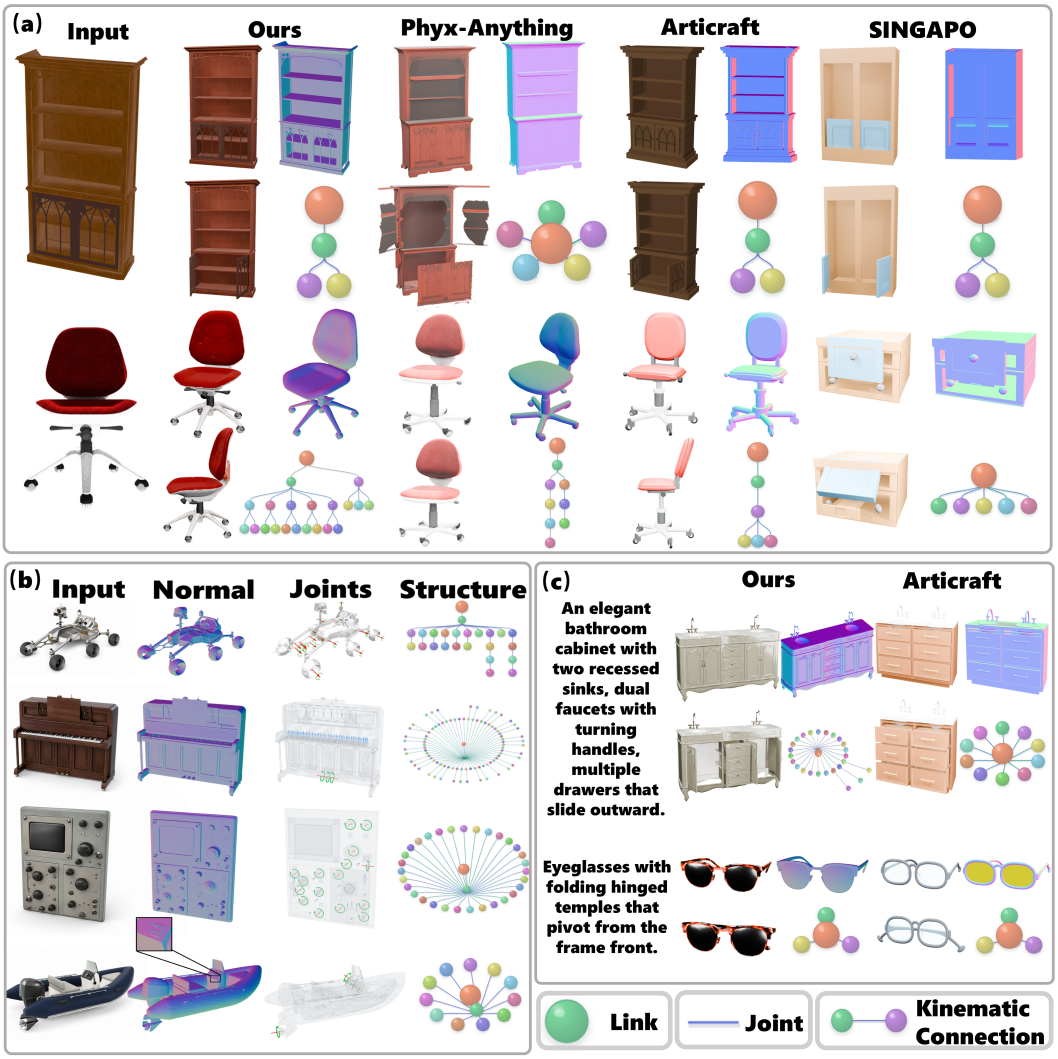

Figure 4: Qualitative results for simulation-ready asset generation. **(a)** Image-conditioned comparison with baselines. **(b)** Representative complex assets generated by our method. **(c)** Text-conditioned comparison with Articraft. Our results exhibit higher geometric fidelity and more detailed functional articulation.

Figure 4 provides qualitative comparisons and representative results, including assets with complex geometry and articulation. Additional visual results are presented in Appendix I. We further analyze the contributions of our key design choices through ablation studies in Appendix G.

6.3 Policy Learning with Generated Assets

We evaluate whether our generated assets can serve as training environments for manipulation policies that transfer to physical objects. The compared generators do not explicitly predict joint damping or stiffness, leaving these parameters to simulator defaults or manual tuning. Inappropriate values can make joints excessively resistant or overly compliant, hindering automated trajectory collection. By predicting these dynamics, our method enables generated assets to be instantiated directly as training environments.

Following InternData-A1 (Tian et al., 2026), we collect verified expert trajectories from multiple generated assets across three manipulation tasks and jointly fine-tune the released $\pi_{0.5}$ policy (Physical Intelligence et al., 2025). Training uses only simulated trajectories; no data from the physical evaluation objects are included. We evaluate the fine-tuned checkpoint in held-out simulation environments and on physical objects without object-specific adaptation. We additionally evaluate the original $\pi_{0.5}$ checkpoint on the same physical tasks and find that it fails to complete any task across all evaluation trials. Complete evaluation details are provided in Appendix O.

Table 1: Evaluation of simulation-ready asset generation on the PartNet-Mobility test set. Panels report conditional generation quality and simulator-facing structural and physical accuracy. The best result is shown in **bold** and "–" denotes an unsupported output.

(a) Image-conditioned generation

Method	Geometry			Asset attributes		
	PSNR ↑	CD ↓	F-score ↑	Scale ↓	Material ↑	Kinematic ↑
PhysX-Anything	8.899	0.06	0.45	0.68	8.27	0.77
SINGAPO	7.423	0.08	0.43	–	–	0.61
Articraft	8.565	0.08	0.33	0.55	–	0.86
Ours	**12.17**	**0.02**	**0.52**	**0.19**	**10.01**	**0.88**

(b) Text-conditioned generation

Method	PSNR ↑	Kinematic ↑	CLIP-T ↑
Articraft	8.754	**0.80**	20.8
Ours	**10.71**	**0.80**	**22.8**

(c) Image-conditioned physical parameters

Method	Mass ↓	Damp ↓	Stiff ↓
PhysX-Anything	0.21	–	–
Ours	**0.07**	**0.09**	**0.05**

(d) Structural and kinematic accuracy

Method	Input	Load ↑	Type Acc. ↑	Axis ↓	Origin ↓	Limit ↓
PhysX-Anything	Image	100.0	0.72	30.00	0.55	0.32
SINGAPO	Image	100.0	0.67	11.52	0.27	0.16
Articraft	Image	100.0	0.90	15.22	0.26	0.10
Ours	Image	**100.0**	**0.92**	**3.4**	**0.21**	**0.03**
SIMART	Mesh	100.0	0.73	8.99	0.25	0.13
ArtLLM	Mesh	100.0	0.56	22.15	0.32	0.14
Particulate	Mesh	100.0	0.90	15.31	0.26	0.07
URDF-Anything	PC + Text	96.7	0.77	44.78	0.53	0.21

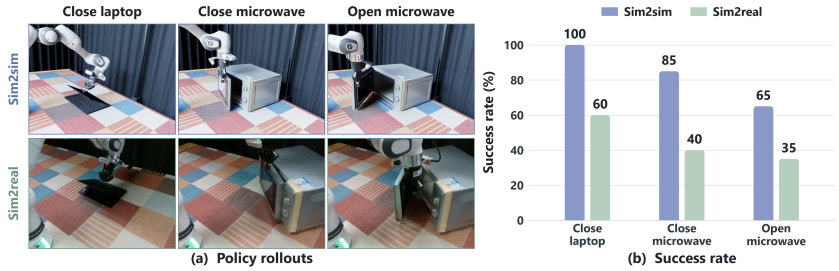

Figure 5: Policy evaluation on unseen simulated assets and real objects. (a) Representative policy rollouts for three manipulation tasks. (b) Success rates over 20 trials per task and domain after fine-tuning on simulation trajectories from our generated assets.

As shown in Figure 5, policies fine-tuned on simulation trajectories collected using our generated assets generalize to unseen simulated instances and transfer to physical objects. These results suggest that the generated geometry, articulation, and joint dynamics provide useful training experience for learning transferable manipulation policies.

7 Conclusion

We presented a bottom-up framework for generating simulation-ready 3D assets from text or images. Our hierarchical representation separates fine-grained geometric components from rigid kinematic links, enabling the model to preserve detailed geometry while generating coherent articulation and simulator-facing physical attributes. To overcome the scarcity of fully annotated assets, we introduced Mobility-PhyX and a heterogeneous co-training strategy that combines simulation-ready supervision with large-scale part-aware geometry. Experiments demonstrate strong performance in geometric fidelity, kinematic accuracy, and physical-attribute prediction. The generated structured representation is editable, exportable to standard asset formats, and supports downstream policy learning and sim-to-real transfer. While functional ambiguity and out-of-distribution inputs remain challenging, our results establish a scalable foundation for generating articulated assets that connect 3D content creation with physical simulation and embodied learning.

Reproducibility Statement

We facilitate reproduction of our results by documenting each stage of the proposed system and its evaluation. Sections 3 and 5 define the component-level geometry representation, bottom-up component-to-link hierarchy, autoregressive sequence grammar, and learning objective; Appendices A–C provide the geometry tokenizer, texturing pipeline, and complete serialization procedure. Section 4 and Appendix E describe the Mobility-PhyX construction pipeline, including geometric and kinematic augmentation, VLM-assisted physical-attribute grounding, and human-expert validation. Section 6.1 and Appendix F identify the data sources and report the filtering, sampling, and balancing procedures, while Appendix D specifies the task-conditioned grammars, token-family-balanced objective, model backbone, context lengths, parallelism settings, sampling ratio, and staged training schedule. Appendix H gives the held-out test protocol, test-derived-augmentation exclusion, comparison settings, metric definitions, and evaluation prompts; Appendix G reports controlled ablations. Finally, Appendix L specifies the deterministic USD/URDF export and Isaac Sim executability checks, and Appendix O details trajectory collection, policy training, and simulation and real-world evaluation.

AI use statement

Generative AI tools were used to draft, restructure, and edit portions of the manuscript; and format tables. Generative models are additionally used within the proposed research pipeline for VLM-assisted physical-attribute annotation, synthetic asset construction, and part-aware appearance generation. These methodological uses are described explicitly in the corresponding sections of the paper.

External AI authoring assistants were not used to fabricate, alter, or selectively report experimental measurements. Proof-related tasks, human-subject analysis, and transcription of research recordings are not applicable to this work. The authors designed and executed the experiments and inspected generated annotations and assets. All AI-assisted text was revised and approved by the authors. We take full responsibility for the final content of this work, including all text and data produced with the aid of generative AI.

Ethics statement

This work aims to reduce the cost of constructing articulated assets for physical simulation and embodied learning. The study does not involve human participants or identifiable personal data, and real-robot experiments are conducted in controlled environments without physical interaction with human subjects. Our training data are derived from existing 3D-asset collections and procedurally generated resources. Any public release will preserve source attribution and licensing information and will include only assets for which redistribution is permitted.

The generated physical attributes should be interpreted as plausible, editable priors rather than measurements or safety-certified models of particular real objects. Errors in geometry, articulation, mass, damping, or stiffness may produce unrealistic or unsafe behavior if assets are transferred directly to physical systems. Users should therefore validate generated assets, inspect or modify their structured physical parameters, and apply appropriate safety constraints before real-world deployment. The source datasets and VLM-assisted annotation process may also reflect biases toward frequently represented object categories and common human-designed mechanisms, potentially reducing reliability for unusual objects or underrepresented designs. Finally, although the proposed system may support beneficial applications in robotics and simulation, the general ability to generate executable articulated environments could be adapted for harmful applications. We encourage responsible use, transparent documentation, and human oversight.

References

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, Wenbin Ge, Zhifang Guo, Qidong Huang, Jie Huang, Fei Huang, Binyuan Hui, Shutong Jiang, Zhaohai Li, Mingsheng Li, Mei Li, Kaixin Li, Zicheng Lin, Junyang Lin, Xuejing Liu, Jiawei Liu, Chenglong Liu, Yang Liu, Dayiheng Liu, Shixuan

Liu, Dunjie Lu, Ruilin Luo, Chenxu Lv, Rui Men, Lingchen Meng, Xuancheng Ren, Xingzhang Ren, Sibo Song, Yuchong Sun, Jun Tang, Jianhong Tu, Jianqiang Wan, Peng Wang, Pengfei Wang, Qiuyue Wang, Yuxuan Wang, Tianbao Xie, Yiheng Xu, Haiyang Xu, Jin Xu, Zhibo Yang, Mingkun Yang, Jianxin Yang, An Yang, Bowen Yu, Fei Zhang, Hang Zhang, Xi Zhang, Bo Zheng, Humen Zhong, Jingren Zhou, Fan Zhou, Jing Zhou, Yuanzhi Zhu, and Ke Zhu. Qwen3-VL technical report, 2025. URL https://arxiv.org/abs/2511.21631.

Black Forest Labs. FLUX. https://github.com/black-forest-labs/flux, 2024.

Black Forest Labs, Stephen Batifol, Andreas Blattmann, Frederic Boesel, Saksham Consul, Cyril Diagne, Tim Dockhorn, Jack English, Zion English, Patrick Esser, Sumith Kulal, Kyle Lacey, Yam Levi, Cheng Li, Dominik Lorenz, Jonas Müller, Dustin Podell, Robin Rombach, Harry Saini, Axel Sauer, and Luke Smith. FLUX.1 Kontext: Flow matching for in-context image generation and editing in latent space, 2025. URL https://arxiv.org/abs/2506.15742.

Marco Callieri, Paolo Cignoni, Massimiliano Corsini, and Roberto Scopigno. Masked photo blending: Mapping dense photographic data set on high-resolution sampled 3D models. Computers & Graphics, 32(4):464–473, 2008. doi: 10.1016/j.cag.2008.05.004. URL https://doi.org/10.1016/j.cag.2008.05.004.

Tianshi Cao, Karsten Kreis, Sanja Fidler, Nicholas Sharp, and Kangxue Yin. TexFusion: Synthesizing 3D textures with text-guided image diffusion models. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 4169–4181, 2023.

Ziang Cao, Fangzhou Hong, Zhaoxi Chen, Liang Pan, and Ziwei Liu. PhysX-Anything: Simulation-ready physical 3D assets from single image. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 5839–5848, 2026a. URL https://openaccess.thecvf.com/content/CVPR2026/html/Cao_PhysX-Anything_Simulation-Ready_Physical_3D_Assets_from_Single_Image_CVPR_2026_paper.html.

Ziang Cao, Yinghao Liu, Haitian Li, Runmao Yao, Fangzhou Hong, Zhaoxi Chen, Liang Pan, and Ziwei Liu. PhysX-Omni: Unified simulation-ready physical 3D generation for rigid, deformable, and articulated objects. arXiv preprint arXiv:2605.21572, 2026b. URL https://arxiv.org/abs/2605.21572.

Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. ShapeNet: An information-rich 3D model repository, 2015. URL https://arxiv.org/abs/1512.03012.

Dave Zhenyu Chen, Yawar Siddiqui, Hsin-Ying Lee, Sergey Tulyakov, and Matthias Nießner. Text2Tex: Text-driven texture synthesis via diffusion models. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 18558–18568, 2023a.

Jingye Chen, Yupan Huang, Tengchao Lv, Lei Cui, Qifeng Chen, and Furu Wei. TextDiffuser: Diffusion models as text painters. In Advances in Neural Information Processing Systems, volume 36, 2023b. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/1df4afb0b4ebf492a41218ce16b6d8df-Abstract-Conference.html.

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 Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 16251–16261, June 2025a.

Yiwen Chen, Zhihao Li, Yikai Wang, Hu Zhang, Qin Li, Chi Zhang, and Guosheng Lin. Ultra3D: Efficient and high-fidelity 3D generation with part attention, 2025b. URL https://arxiv.org/abs/2507.17745.

Zoey 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. In Robotics: Science and Systems, 2024. URL https://arxiv.org/abs/2405.11656.

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. arXiv preprint arXiv:2212.08051, 2022. URL https://arxiv.org/abs/2212.08051.

Kangle Deng, Hsueh-Ti Derek Liu, Yiheng Zhu, Xiaoxia Sun, Chong Shang, Kiran Bhat, Deva Ramanan, Jun-Yan Zhu, Maneesh Agrawala, and Tinghui Zhou. Efficient autoregressive shape generation via octree-based adaptive tokenization. In Proceedings of the IEEE/CVF International Conference on Computer Vision, 2025. URL https://arxiv.org/abs/2504.02817.

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, pp. 618–627, 2025. URL https://openaccess.thecvf.com/content/CVPR2025/html/Gao_MeshArt_Generating_Articulated_Meshes_with_Structure-Guided_Transformers_CVPR_2025_paper.html.

Tianhong Gao, Cheng Yu, Yinghao Xu, and Mengyu Chu. JODA: Composable joint dynamics for articulated objects. arXiv preprint arXiv:2605.09954, 2026. doi: 10.48550/arXiv.2605.09954. URL https://arxiv.org/abs/2605.09954.

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 Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 7081–7091, 2023. URL https://openaccess.thecvf.com/content/CVPR2023/html/Geng_GAPartNet_Cross-Category_Domain-Generalizable_Object_Perception_and_Manipulation_via_Generalizable_and_CVPR_2023_paper.html.

Google DeepMind. Nano Banana Pro: Gemini 3 pro image. https://deepmind.google/models/gemini-image/pro/, 2026. Accessed: 2026-08-27.

Xianglong He, Zi-Xin Zou, Chia-Hao Chen, Yuan-Chen Guo, Ding Liang, Chun Yuan, Wanli Ouyang, Yan-Pei Cao, and Yangguang Li. SparseFlex: High-resolution and arbitrary-topology 3D shape modeling. arXiv preprint arXiv:2503.21732, 2025. URL https://arxiv.org/abs/2503.21732.

Jie Huang, Xiaohe Li, Jiahao Li, Fangli Mou, Chen Qian, Yuqiang Fang, Junhao Fan, Kaixin Zhang, and Zide Fan. PhysX-CoT: Structured physical reasoning from a single image to simulation-ready 3D assets. arXiv preprint arXiv:2608.08053, 2026. doi: 10.48550/arXiv.2608.08053. URL https://arxiv.org/abs/2608.08053.

Denys Iliash, Jiayi Liu, Egor Fokin, Qirui Wu, Ali Mahdavi-Amiri, Manolis Savva, and Angel X. Chang. Artiverse: A diverse and physically grounded dataset for articulated objects. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 8932–8942, 2026. URL https://openaccess.thecvf.com/content/CVPR2026/html/Iliash_Artiverse_A_Diverse_and_Physically_Grounded_Dataset_for_Articulated_Objects_CVPR_2026_paper.html.

Zhao Jin, Zhengping Che, Tao Li, Zhen Zhao, Kun Wu, Yuheng Zhang, Yinuo Zhao, Zehui Liu, Qiang Zhang, Xiaozhu Ju, Jing Tian, Yousong Xue, and Jian Tang. ArtVIP: Articulated digital assets of visual realism, modular interaction, and physical fidelity for robot learning. In International Conference on Learning Representations, 2026. URL https://openreview.net/forum?id=SqPLEZ66BO.

Abhishek Joshi, Beining Han, Jack Nugent, Yiming Zuo, Jonathan Liu, Hongyu Wen, Stamatis Alexandropoulos, Tao Sun, Alexander Raistrick, Gaowen Liu, Yi Shao, and Jia Deng. InfinigenSim: Procedural generation of articulated simulation assets, 2025. URL https://arxiv.org/abs/2505.10755.

jpcy. xatlas: Mesh parameterization / UV unwrapping library. https://github.com/jpcy/xatlas, 2026. GitHub repository, accessed 2026-08-27.

Zeqiang Lai, Yunfei Zhao, Haolin Liu, Zibo Zhao, Qingxiang Lin, Huiwen Shi, Xianghui Yang, Mingxin Yang, Shuhui Yang, Yifei Feng, Sheng Zhang, Xin Huang, Di Luo, Fan Yang, Fang Yang, Lifu Wang, Sicong Liu, Yixuan Tang, Yulin Cai, Zebin He, Tian Liu, Yuhong Liu, Jie Jiang, Linus, Jingwei Huang, and Chunchao Guo. Hunyuan3D 2.5: Towards high-fidelity 3D assets generation with ultimate details, 2025a. URL https://arxiv.org/abs/2506.16504.

Zeqiang Lai, Yunfei Zhao, Zibo Zhao, Haolin Liu, Qingxiang Lin, Jingwei Huang, Chunchao Guo, and Xiangyu Yue. LATTICE: Democratize high-fidelity 3D generation at scale, 2025b. URL https://arxiv.org/abs/2512.03052.

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 International Conference on Learning Representations, 2025. URL https://arxiv.org/abs/2410.13882.

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, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/655846cc914cb7ff977a1ada40866441-Abstract-Conference.html.

Ruining Li, Yuxin Yao, Chuanxia Zheng, Christian Rupprecht, Joan Lasenby, Shangzhe Wu, and Andrea Vedaldi. Particulate: Feed-forward 3D object articulation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 27708–27718, 2026. URL https://openaccess.thecvf.com/content/CVPR2026/html/Li_Particulate_Feed-Forward_3D_Object_Articulation_CVPR_2026_paper.html.

Weiyu Li, Jiarui Liu, Hongyu Yan, Rui Chen, Yixun Liang, Xuelin Chen, Ping Tan, and Xiaoxiao Long. CraftsMan3D: High-fidelity mesh generation with 3D native generation and interactive geometry refiner, 2025a. URL https://arxiv.org/abs/2405.14979.

Yangguang Li, Xianglong He, Zi-Xin Zou, Zexiang Liu, Wanli Ouyang, Ding Liang, and Yan-Pei Cao. ShapeGen: Towards high-quality 3D shape synthesis. In Proceedings of the SIGGRAPH Asia 2025 Conference Papers, SA Conference Papers '25, New York, NY, USA, 2025b. Association for Computing Machinery. ISBN 9798400721373. doi: 10.1145/3757377.3763812. URL https://doi.org/10.1145/3757377.3763812.

Yangguang Li, Zi-Xin Zou, Zexiang Liu, Dehu Wang, Yuan Liang, Zhipeng Yu, Xingchao Liu, Yuan-Chen Guo, Ding Liang, Wanli Ouyang, et al. TripoSG: High-fidelity 3D shape synthesis using large-scale rectified flow models. arXiv preprint arXiv:2502.06608, 2025c. URL https://arxiv.org/abs/2502.06608.

Yuchen Li, Ujjwal Upadhyay, Habib Slim, Tezuesh Varshney, Ahmed Abdelreheem, Arpit Prajapati, Suhail Pothigara, Peter Wonka, and Mohamed Elhoseiny. 3D CoMPaT: Composition of materials on parts of 3D things. In Computer Vision – ECCV 2022, pp. 110–127. Springer, 2022. doi: 10.1007/978-3-031-20074-8_7. URL https://doi.org/10.1007/978-3-031-20074-8_7.

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 Advances in Neural Information Processing Systems, volume 38, pp. 94974–95002, 2025d. doi: 10.52202/085713-3175. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/88445acb45c922bdb06952e31f8a60ec-Abstract-Conference.html.

Xinyu Lian, Zichao Yu, Ruiming Liang, Yitong Wang, Li Ray Luo, Kaixu Chen, Yuanzhen Zhou, Qihong Tang, Xudong Xu, Zhaoyang Lyu, Bo Dai, and Jiangmiao Pang. Infinite Mobility: Scalable high-fidelity synthesis of articulated objects via procedural generation. arXiv preprint arXiv:2503.13424, 2025. URL https://arxiv.org/abs/2503.13424.

Manwen Liao, Xinyu Lian, Jian Mao, Kaixu Chen, Li Luo, Jinghao Yan, Wanshui Gan, Qiao Yu, Weitian Zhang, Chunhua Shen, Guang Chen, Bo Dai, Xudong Xu, and Zhaoyang Lyu. MegaParts:

Scaling part-aware 3D object generation to 300 parts via token-efficient autoregressive modeling, 2026. URL https://arxiv.org/abs/2608.14783.

Chen-Hsuan Lin, Jun Gao, Luming Tang, Towaki Takikawa, Xiaohui Zeng, Xun Huang, Karsten Kreis, Sanja Fidler, Ming-Yu Liu, and Tsung-Yi Lin. Magic3D: High-resolution text-to-3D content creation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2023.

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, pp. 17880–17889, 2024. URL https://openaccess.thecvf.com/content/CVPR2024/html/Liu_CAGE_Controllable_Articulation_GEneration_CVPR_2024_paper.html.

Jiayi Liu, Denys Iliash, Angel X. Chang, Manolis Savva, and Ali Mahdavi-Amiri. SINGAPO: Single image controlled generation of articulated parts in objects. In International Conference on Learning Representations, 2025. URL https://openreview.net/forum?id=OdMqKszKSd.

Guan Luo, Xiu Li, Rui Chen, Xuanyu Yi, Jing Lin, Chia-Hao Chen, Jiahang Liu, Song-Hai Zhang, and Jianfeng Zhang. TopoMesh: High-fidelity mesh autoencoding via topological unification, 2026. URL https://arxiv.org/abs/2603.24278.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-Refine: Iterative refinement with self-feedback. In Advances in Neural Information Processing Systems, volume 36, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/91edff07232fb1b55a505a9e9f6c0ff3-Abstract-Conference.html.

Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, and Gavriel State. Isaac Gym: High performance GPU-based physics simulation for robot learning. In Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks, volume 1, 2021. URL https://datasets-benchmarks-proceedings.neurips.cc/paper_files/paper/2021/hash/28dd2c7955ce926456240b2ff0100bde-Abstract-round2.html.

Zhao Mandi, Yijia Weng, Dominik Bauer, and Shuran Song. Real2Code: Reconstruct articulated objects via code generation. arXiv preprint arXiv:2406.08474, 2024. URL https://arxiv.org/abs/2406.08474.

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, pp. 909–918, June 2019. URL https://openaccess.thecvf.com/content_CVPR_2019/html/Mo_PartNet_A_Large-Scale_Benchmark_for_Fine-Grained_and_Hierarchical_Part-Level_3D_CVPR_2019_paper.html.

Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. Large language diffusion models. In Advances in Neural Information Processing Systems, volume 38, 2025. doi: 10.52202/085713-1689. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/48b383b24230e0e6e649d9c98dae4d8c-Abstract-Conference.html.

NVIDIA. Tuning joint drive gains. Isaac Sim Documentation, 2026. URL https://docs.isaacsim.omniverse.nvidia.com/latest/robot_setup_tutorials/joint_tuning.html. Accessed: 2026-08-31.

OpenAI. ChatGPT Images 2.0 System Card, April 2026a. URL https://deploymentsafety.openai.com/chatgpt-images-2-0. Published April 21, 2026.

OpenAI. GPT-5.6 Preview System Card, June 2026b. URL https://deploymentsafety.openai.com/gpt-5-6-preview. Accessed: 2026-08-24.

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, James Tanner, Quan Vuong, Homer Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky. $\pi_{0.5}$: A vision-language-action model with open-world generalization. arXiv preprint arXiv:2504.16054, 2025. doi: 10.48550/arXiv.2504.16054. URL https://arxiv.org/abs/2504.16054.

Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. DreamFusion: Text-to-3D using 2D diffusion. arXiv preprint arXiv:2209.14988, 2022. URL https://arxiv.org/abs/2209.14988.

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. arXiv preprint arXiv:2502.02590, 2025. URL https://arxiv.org/abs/2502.02590.

Elad Richardson, Gal Metzer, Yuval Alaluf, Raja Giryes, and Daniel Cohen-Or. TEXTure: Text-guided texturing of 3D shapes. In ACM SIGGRAPH 2023 Conference Proceedings. Association for Computing Machinery, 2023. doi: 10.1145/3588432.3591503. URL https://doi.org/10.1145/3588432.3591503.

Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. Megatron-LM: Training multi-billion parameter language models using model parallelism. arXiv preprint arXiv:1909.08053, 2019. URL https://arxiv.org/abs/1909.08053.

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, pp. 1894–1904, 2025. URL https://openaccess.thecvf.com/content/CVPR2025/html/Su_ArtFormer_Controllable_Generation_of_Diverse_3D_Articulated_Objects_CVPR_2025_paper.html.

Jianhua Sun, Yuxuan Li, Jiude Wei, Longfei Xu, Nange Wang, Yining Zhang, and Cewu Lu. Arti-PG: A toolbox for procedurally synthesizing large-scale and diverse articulated objects with rich annotations. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 6396–6405, 2025. URL https://openaccess.thecvf.com/content/ICCV2025/html/Sun_Arti-PG_A_Toolbox_for_Procedurally_Synthesizing_Large-Scale_and_Diverse_Articulated_ICCV_2025_paper.html.

Yang Tian, Yuyin Yang, Yiman Xie, Zetao Cai, Xu Shi, Ning Gao, Hangxu Liu, Xuekun Jiang, Zherui Qiu, Feng Yuan, Yaping Li, Ping Wang, Junhao Cai, Jia Zeng, Hao Dong, and Jiangmiao Pang. InternData-A1: Pioneering high-fidelity synthetic data for pre-training generalist policy. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 976–985, 2026. URL https://openaccess.thecvf.com/content/CVPR2026/html/Tian_InternData-A1_Pioneering_High-Fidelity_Synthetic_Data_for_Pre-training_Generalist_Policy_CVPR_2026_paper.html.

Yuxiang Tuo, Wangmeng Xiang, Jun-Yan He, Yifeng Geng, and Xuansong Xie. AnyText: Multilingual visual text generation and editing. In International Conference on Learning Representations, 2024. URL https://openreview.net/forum?id=ezBH9WE9s2.

Michael Waechter, Nils Moehrle, and Michael Goesele. Let there be color! large-scale texturing of 3D reconstructions. In Computer Vision – ECCV 2014, pp. 836–850. Springer,

2014. doi: 10.1007/978-3-319-10602-1_54. URL https://doi.org/10.1007/978-3-319-10602-1_54.

Penghao Wang, Siyuan Xie, Hongyu Yan, Xianghui Yang, Jingwei Huang, Chunchao Guo, and Jiayuan Gu. ArtLLM: Generating articulated assets via 3D LLM. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 34281–34291, 2026. URL https://openaccess.thecvf.com/content/CVPR2026/html/Wang_ArtLLM_Generating_Articulated_Assets_via_3D_LLM_CVPR_2026_paper.html.

Zhengyi Wang, Cheng Lu, Yikai Wang, Fan Bao, Chongxuan Li, Hang Su, and Jun Zhu. ProlificDreamer: High-fidelity and diverse text-to-3D generation with variational score distillation. In Advances in Neural Information Processing Systems, 2023.

Shuang Wu, Youtian Lin, Feihu Zhang, Yifei Zeng, Jingxi Xu, Philip Torr, Xun Cao, and Yao Yao. Direct3D: Scalable image-to-3D generation via 3D latent diffusion transformer. arXiv preprint arXiv:2405.14832, 2024. URL https://arxiv.org/abs/2405.14832.

Shuang Wu, Youtian Lin, Feihu Zhang, Yifei Zeng, Yikang Yang, Yajie Bao, Jiachen Qian, Siyu Zhu, Philip Torr, Xun Cao, and Yao Yao. Direct3D-S2: Gigascale 3D generation made easy with spatial sparse attention. arXiv preprint arXiv:2505.17412, 2025. URL https://arxiv.org/abs/2505.17412.

Zhuangzhe Wu, Yue Xin, Chengkai Hou, Minghao Chen, Yaoxu Lyu, Jieyu Zhang, and Shanghang Zhang. URDF-Anything+: Autoregressive articulated 3D models generation for physical simulation. arXiv preprint arXiv:2603.14010, 2026. URL https://arxiv.org/abs/2603.14010.

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 Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020. URL https://openaccess.thecvf.com/content_CVPR_2020/html/Xiang_SAPIEN_A_SimulAted_Part-Based_Interactive_ENvironment_CVPR_2020_paper.html.

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. arXiv preprint arXiv:2412.01506, 2024. URL https://arxiv.org/abs/2412.01506.

Jianfeng Xiang, Xiaoxue Chen, Sicheng Xu, Ruicheng Wang, Zelong Lv, Yu Deng, Hongyuan Zhu, Yue Dong, Hao Zhao, Nicholas Jing Yuan, and Jiaolong Yang. Native and compact structured latents for 3D generation. Tech report, 2025. URL https://github.com/microsoft/TRELLIS.2.

Xianghui Yang, Huiwen Shi, Bowen Zhang, Fan Yang, Jiacheng Wang, Hongxu Zhao, Xinhai Liu, Xinzhou Wang, Qingxiang Lin, Jiaao Yu, Lifu Wang, Jing Xu, Zebin He, Zhuo Chen, Sicong Liu, Junta Wu, Yihang Lian, Shaoxiong Yang, Yuhong Liu, Yong Yang, Di Wang, Jie Jiang, and Chunchao Guo. Hunyuan3D 1.0: A unified framework for text-to-3D and image-to-3D generation, 2024a. URL https://arxiv.org/abs/2411.02293.

Zhengyuan Yang, Jianfeng Wang, Linjie Li, Kevin Lin, Chung-Ching Lin, Zicheng Liu, and Lijuan Wang. Idea2Img: Iterative self-refinement with GPT-4V for automatic image design and generation. In Computer Vision – ECCV 2024, pp. 167–184. Springer, 2024b. doi: 10.1007/978-3-031-72920-1_10. URL https://doi.org/10.1007/978-3-031-72920-1_10.

Xianfang Zeng, Xin Chen, Zhongqi Qi, Wen Liu, Zibo Zhao, Zhibin Wang, Bin Fu, Yong Liu, and Gang Yu. Paint3D: Paint anything 3D with lighting-less texture diffusion models. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 4252–4262, 2024.

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, 42(4), July 2023. ISSN 0730-0301. doi: 10.1145/3592442. URL https://doi.org/10.1145/3592442.

Chuanrui Zhang, Minghan Qin, Yuang Wang, Baifeng Xie, Hang Li, and Ziwei Wang. SIMART: Decomposing monolithic meshes into sim-ready articulated assets via MLLM. arXiv preprint arXiv:2603.23386, 2026. URL https://arxiv.org/abs/2603.23386.

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, 43(4), July 2024a. ISSN 0730-0301. doi: 10.1145/3658146. URL https://doi.org/10.1145/3658146.

Yuqing Zhang, Yuan Liu, Zhiyu Xie, Lei Yang, Zhongyuan Liu, Mengzhou Yang, Runze Zhang, Qilong Kou, Cheng Lin, Wenping Wang, and Xiaogang Jin. DreamMat: High-quality PBR material generation with geometry- and light-aware diffusion models. ACM Transactions on Graphics, 43 (4), 2024b. doi: 10.1145/3658170. URL https://doi.org/10.1145/3658170.

Zibo Zhao, Wen Liu, Xin Chen, Xianfang Zeng, Rui Wang, Pei Cheng, Bin Fu, Tao Chen, Gang Yu, and Shenghua Gao. Michelangelo: Conditional 3D shape generation based on shape-image-text aligned latent representation. In Thirty-seventh Conference on Neural Information Processing Systems, 2023. URL https://openreview.net/forum?id=xmxgMij3LY.

Zibo Zhao, Zeqiang Lai, Qingxiang Lin, Yunfei Zhao, Haolin Liu, Shuhui Yang, Yifei Feng, Mingxin Yang, Sheng Zhang, Xianghui Yang, Huiwen Shi, Sicong Liu, Junta Wu, Yihang Lian, Fan Yang, Ruining Tang, Zebin He, Xinzhou Wang, Jian Liu, Xuhui Zuo, Zhuo Chen, Biwen Lei, Haohan Weng, Jing Xu, Yiling Zhu, Xinhai Liu, Lixin Xu, Changrong Hu, Shaoxiong Yang, Song Zhang, Yang Liu, Tianyu Huang, Lifu Wang, Jihong Zhang, Meng Chen, Liang Dong, Yiwen Jia, Yulin Cai, Jiaao Yu, Yixuan Tang, Hao Zhang, Zheng Ye, Peng He, Runzhou Wu, Chao Zhang, Yonghao Tan, Jie Xiao, Yangyu Tao, Jianchen Zhu, Jinbao Xue, Kai Liu, Chongqing Zhao, Xinming Wu, Zhichao Hu, Lei Qin, Jianbing Peng, Zhan Li, Minghui Chen, Xipeng Zhang, Lin Niu, Paige Wang, Yingkai Wang, Haozhao Kuang, Zhongyi Fan, Xu Zheng, Weihao Zhuang, YingPing He, Tian Liu, Yong Yang, Di Wang, Yuhong Liu, Jie Jiang, Jingwei Huang, and Chunchao Guo. Hunyuan3D 2.0: Scaling diffusion models for high resolution textured 3D assets generation, 2025. URL https://arxiv.org/abs/2501.12202.

Chunting Zhou, Lili Yu, Arun Babu, Kushal Tirumala, Michihiro Yasunaga, Leonid Shamis, Jacob Kahn, Xuezhe Ma, Luke Zettlemoyer, and Omer Levy. Transfusion: Predict the next token and diffuse images with one multi-modal model. In International Conference on Learning Representations, 2025. URL https://openreview.net/forum?id=SI2hI0frk6.

Matt Zhou, Ruining Li, Xiaoyang Lyu, Zhaomou Song, Zhening Huang, Chuanxia Zheng, Christian Rupprecht, Andrea Vedaldi, and Shangzhe Wu. Articraft: An agentic system for scalable articulated 3D asset generation. arXiv preprint arXiv:2605.15187, 2026. URL https://arxiv.org/abs/2605.15187.

A COMPONENT-LEVEL GEOMETRY VQ-VAE

Following the practice of MegaParts (Liao et al., 2026), we employ a fully causal VQ-VAE that represents each geometric component as an adaptive-length sequence of discrete tokens.

Component normalization. Given a component mesh $\mathcal{M}$ with bounding-box center $\mathbf{m}$ and side lengths $\mathbf{e}$, we define

$$\alpha = \frac{1}{2} \max_{d \in \{1,2,3\}} e_d. \tag{8}$$

Each vertex $\mathbf{v} \in \mathcal{M}$ is transformed into the component-local coordinate system as

$$\overline{\mathbf{v}} = \frac{\mathbf{v} - \mathbf{m}}{\alpha}, \tag{9}$$

producing a normalized mesh $\overline{\mathcal{M}} \subseteq [-1, 1]^3$. The VQ-VAE therefore models local shape independently of the component's global placement and metric scale.

Causal encoding and quantization. We sample a surface point set

$$\mathcal{S} = \{(\mathbf{p}_j, \mathbf{f}_j)\}_{j=1}^{N_s}, \tag{10}$$

where $\mathbf{p}_j \in \mathbb{R}^3$ is a surface point and $\mathbf{f}_j$ optionally contains attributes such as its surface normal. These samples are embedded into input tokens and processed by an encoder with a learnable latent-query sequence of maximum length $L_{\max}$. The encoder combines cross-attention from the latent queries to the surface tokens with causal self-attention between latent positions, producing

$$\mathbf{H} = \mathrm{Enc}_{\mathrm{geo}}(\mathcal{S}) = (\mathbf{h}_1, \ldots, \mathbf{h}_{L_{\max}}). \tag{11}$$

Let $\mathcal{E} = \{\mathbf{e}_v\}_{v=1}^{V}$ denote the discrete codebook. Each encoder feature is quantized using spherical vector quantization:

$$z_t = \arg\max_{v \in [V]} \left\langle \frac{\mathbf{h}_t}{\|\mathbf{h}_t\|_2}, \frac{\mathbf{e}_v}{\|\mathbf{e}_v\|_2} \right\rangle, \qquad t \in [L_{\max}]. \tag{12}$$

The component is consequently represented by a sequence of discrete codebook indices

$$z = (z_1, \ldots, z_{L_{\max}}), \qquad z_t \in [V]. \tag{13}$$

Causal decoding. The decoder is also causal over latent positions. Given a prefix $z_{1:L}$ and a spatial query point $\mathbf{x} \in [-1, 1]^3$, it predicts a signed-distance value

$$\widehat{d}\,(\mathbf{x}; z_{1:L}) = \mathrm{Dec}_{\mathrm{geo}}\left(\mathbf{x}, \mathbf{e}_{z_1}, \ldots, \mathbf{e}_{z_L}\right). \tag{14}$$

The reconstructed normalized mesh is obtained from the predicted zero level set using marching cubes:

$$\widehat{\overline{\mathcal{M}}}(L) = \mathrm{MC}\left(\widehat{d}\,(\cdot; z_{1:L}), 0\right). \tag{15}$$

It can subsequently be mapped back to the original component coordinates through

$$\widehat{\mathbf{v}} = \mathbf{m} + \alpha\widehat{\overline{\mathbf{v}}}, \qquad \widehat{\overline{\mathbf{v}}} \in \widehat{\overline{\mathcal{M}}}(L). \tag{16}$$

Prefix-based training objective. During training, we randomly truncate the latent sequence to a prefix length

$$L \in \mathscr{L} = \{2^n\}_{n=4}^{12}. \tag{17}$$

Because both the encoder and decoder are causal, each prefix constitutes a valid representation. Earlier tokens are encouraged to encode essential coarse geometry, while later tokens progressively introduce finer details.

The VQ-VAE is optimized using

$$\mathcal{L}_{\text{VQ-VAE}} = \lambda_{\mathrm{tsdf}}\mathcal{L}_{\mathrm{tsdf}} + \lambda_{\mathrm{eik}}\mathcal{L}_{\mathrm{eik}} + \lambda_{\mathrm{normal}}\mathcal{L}_{\mathrm{normal}} + \lambda_{\mathrm{vq}}\mathcal{L}_{\mathrm{vq}}, \tag{18}$$

where $\mathcal{L}_{\mathrm{tsdf}}$ supervises the truncated and normalized signed-distance field, $\mathcal{L}_{\mathrm{eik}}$ imposes an Eikonal constraint within a near-surface band, $\mathcal{L}_{\mathrm{normal}}$ aligns the predicted field gradient with the reference surface normal, and $\mathcal{L}_{\mathrm{vq}}$ optimizes the correspondence between encoder features and codebook embeddings.

Phased optimization. Directly optimizing the encoder and decoder jointly through a discrete bottleneck can produce rapidly changing code assignments, preventing the decoder from learning a stable correspondence between code sequences and geometry. We therefore follow the two-stage phased optimization strategy of MegaParts (Liao et al., 2026). In the first stage, the encoder, decoder, and quantization bottleneck are jointly optimized to establish an initial discrete representation. In the second stage, training alternates between encoder-only and decoder-only phases. During an encoder-only phase, the decoder is frozen while the encoder adapts its feature organization and code assignments. During a decoder-only phase, the encoder is frozen, providing stable discrete targets from which the decoder can improve geometric reconstruction.

This alternating schedule separates the evolution of the discrete codes from the learning of their geometric interpretation, reducing the instability caused by simultaneous changes on both sides of the bottleneck. Following MegaParts, the phased schedule is combined with optimal-transport-based quantization and the stochastic gradient shortcut to improve codebook utilization and stabilize gradient propagation through the quantization layer.

Adaptive representation length. To allocate different token budgets to components of different complexity, we select the final prefix length using a rate–distortion objective:

$$L^\star = \arg\min_{L \in \mathscr{L}} \left[D(L) + \lambda_{\text{rate}} \frac{L}{L_{\max}} \right], \tag{19}$$

where $D(L)$ combines Chamfer distance, surface-normal inconsistency, and a bounding-box discrepancy between the input and reconstructed meshes. The final component representation is

$$z_{1:L^\star} = (z_1, \ldots, z_{L^\star}). \tag{20}$$

Simple components can therefore be encoded using short prefixes, whereas geometrically intricate components receive longer sequences without changing the VQ-VAE architecture or codebook.

B Texturing Pipeline

B.1 Part-Based Texture Generation for Articulated Objects

Given the geometry, part decomposition, and kinematic structure of a generated articulated object, we further synthesize physically based materials for its movable parts while maintaining global appearance consistency. A straightforward solution would be to texture each part independently. However, this strategy discards the semantic and spatial relationships among parts and may therefore produce inconsistent colors, material types, and texture scales across adjacent components (Li et al., 2022; Richardson et al., 2023; Chen et al., 2023a; Cao et al., 2023). Instead, we formulate part-based texturing as a single object-level conditional generation process followed by a deterministic material remapping step. All parts share one TRELLIS.2 (Xiang et al., 2025) inference pass, while their identities are preserved through spatial partitions in a UV atlas and the corresponding inverse mappings. This design requires neither retraining nor fine-tuning TRELLIS.2 and introduces no additional part tokens or explicit part identifiers into the network.

Let an articulated object consist of N triangular part meshes,

$$\mathcal{O} = \{\mathcal{M}_i\}_{i=1}^{N}, \tag{21}$$

where $\mathcal{M}_i$ denotes the mesh of the i-th part. We first independently parameterize each part in a local UV domain using a UV unwrapping algorithm (jpcy, 2026):

$$\phi_i : \mathcal{M}_i \to [0,1]^2. \tag{22}$$

We then set $g = \lceil \sqrt{N} \rceil$ and arrange all local UV maps in a regular $g \times g$ global atlas. For a local UV coordinate (u, v), its global atlas coordinate for the i-th part is defined as

$$\Phi_i(u,v) = \left(\frac{u + c_i}{g}, \frac{v + r_i}{g} \right), \quad c_i = (i-1) \bmod g, \quad r_i = \left\lfloor \frac{i-1}{g} \right\rfloor. \tag{23}$$

While retaining the original mesh, name, and local UV coordinates of every part, we concatenate their vertices and faces to construct a combined mesh $\mathcal{M}_{\text{atlas}}$ equipped with the global atlas coordinates. This combination is used only for material inference and does not alter the original part hierarchy or articulation structure.

Before inference, we normalize $\mathcal{M}_{\text{atlas}}$ to the canonical coordinate space expected by TRELLIS.2 and use a reference image I_{ref} as the appearance condition. The reference image may be supplied directly by an upstream appearance-generation module or synthesized from normal and clay renderings of the combined mesh. TRELLIS.2 performs a single material inference pass over the complete geometry and predicts a spatial field of physically based rendering (PBR) attributes:

$$\mathcal{V}_{\text{pbr}} = F_\theta(\mathcal{M}_{\text{atlas}}, I_{\text{ref}}), \qquad \mathcal{V}_{\text{pbr}}(\mathbf{x}) = \big[\mathbf{c}(\mathbf{x}), m(\mathbf{x}), r(\mathbf{x}), a(\mathbf{x})\big], \tag{24}$$

where $\mathbf{c}$, m, r, and a denote base color, metallic, roughness, and alpha, respectively. Because all parts jointly participate in shape encoding and texture generation within the same inference pass, the model can exploit the geometry and image semantics of the complete object. This produces more coherent materials across parts than independent part-wise generation.

To convert the three-dimensional PBR attributes into two-dimensional textures, we rasterize $\mathcal{M}_{\text{atlas}}$ in the global UV domain and compute the three-dimensional surface position $\mathbf{x}_p$ associated with each valid texel. The atlas value is then obtained by trilinearly sampling $\mathcal{V}_{\text{pbr}}$:

$$\mathcal{T}_{\text{atlas}}(p) = \text{TriInterp}(\mathcal{V}_{\text{pbr}}, \mathbf{x}_p). \tag{25}$$

Invalid texels near rasterization boundaries are filled through a finite number of neighborhood-propagation steps to reduce seams during texture sampling. If the desired texture resolution of each part is $s \times s$, the global atlas is generated at a resolution of $(gs) \times (gs)$. Finally, the local PBR texture of part i is recovered by cropping the atlas cell indexed by (r_i, c_i):

$$\mathcal{T}_i = \text{Crop}_{r_i, c_i}(\mathcal{T}_{\text{atlas}}). \tag{26}$$

The resulting base-color, metallic, and roughness maps are reassigned to the original local parameterization ϕ_i and exported as an independent material for each part. Since every material remains defined in the local UV space of its associated part, subsequent joint transformations affect only the spatial pose of the part and leave its texture parameterization unchanged. The proposed stage thus preserves the part structure and mobility of the articulated object while producing a globally consistent appearance across its components, making it directly applicable as a modular texturing stage in an articulated-object generation pipeline.

B.2 Agentic Texture Generation for Text, Symbols, and Graphic Patterns

Although the part-based pipeline described in Section B.1 produces spatially coherent PBR materials, it remains challenging to synthesize appearance elements containing text, symbols, logos, or structured graphic patterns. These elements typically contain high-frequency edges and strong semantic constraints. Directly generating them in a three-dimensional texture field often results in blurred boundaries, distorted symbols, misspelled text, or inconsistent patterns across different views. Merely increasing the texture resolution may be insufficient, because the errors originate not only from spatial undersampling but also from the difficulty of preserving discrete semantic content during generative inference (Chen et al., 2023b; Tuo et al., 2024).

We address this limitation using an agentic (Yang et al., 2024b; Madaan et al., 2023) texture-generation procedure that explicitly decouples low-frequency material synthesis from high-frequency semantic-detail generation. The low-frequency component contains the underlying colors, materials, and surface variation, whereas the high-frequency component contains text, symbols, logos, and other structured graphics. Instead of requiring the three-dimensional texture model to generate both components simultaneously, an agent coordinates a sequence of geometry-conditioned image generation, semantic-detail removal, multiview detail insertion, texture baking, and quality verification. Each stage is executed only after its inputs have passed a stage-specific validation procedure.

Let $\mathcal{M}_{\text{atlas}}$ denote the combined mesh with the global UV parameterization introduced in Section B.1. Given a textual appearance description y and one or more geometry renderings I_{geo}, such as normal, depth, or clay renderings, an instruction-following multimodal image model G_ψ first synthesizes an appearance reference image

$$I^A = G_\psi(I_{\text{geo}}, y). \tag{27}$$

The geometry rendering provides the intended object shape, pose, and camera configuration, while y specifies the desired materials and semantic appearance elements. We refer to the resulting image

as image A. In our implementation, G_ψ can be instantiated by a vision–language image generation or editing model, such as Gemini 3 Pro Image (Nano Banana Pro) (Google DeepMind, 2026) or GPT-Image-2 (OpenAI, 2026a).

Before modifying the image, the agent analyzes I^A and constructs a semantic-detail manifest

$$\mathcal{E} = \{e_j\}_{j=1}^{K}, \qquad e_j = \left(\tau_j, s_j, a_j, \Omega_j^A\right), \tag{28}$$

where τ_j is the type of the j-th element, such as text, symbol, logo, or pattern; s_j describes its semantic content; a_j records its visual attributes, including color, style, orientation, and relative scale; and Ω_j^A denotes its image-space support in the reference view. For textual elements, s_j includes the exact character sequence, capitalization, and layout. This manifest provides a persistent description shared by all subsequent views and prevents the image editor from independently interpreting the semantic content in each view.

The semantic elements are then removed from image A to obtain a clean appearance reference, denoted as image B. Let

$$M^A = \bigcup_{j=1}^{K} \Omega_j^A \tag{29}$$

be the union of the element masks. An image editor E_ψ inpaints these regions using the surrounding material appearance:

$$\widetilde{I}^B = E_\psi\big(I^A; \mathrm{remove}(\mathcal{E})\big). \tag{30}$$

To prevent unintended modifications outside the edited regions, the final clean reference is explicitly composited as

$$I^B = \big(1 - M^A\big) \odot I^A + M^A \odot \widetilde{I}^B, \tag{31}$$

where $\odot$ denotes element-wise multiplication. Consequently, pixels outside the semantic-detail masks remain identical to those in image A. The agent verifies that all specified elements have been removed, that the inpainted regions are locally compatible with the surrounding material, and that the silhouette, geometry boundaries, and unedited appearance remain unchanged.

Image B is subsequently used as the conditioning image of the part-based texture pipeline:

$$\mathcal{T}_{\mathrm{base}} = F_\theta\big(\mathcal{M}_{\mathrm{atlas}}, I^B\big), \tag{32}$$

where $\mathcal{T}_{\mathrm{base}}$ denotes the generated PBR material without text, symbols, or graphic patterns. Since the high-frequency semantic elements have been removed from the conditioning image, the texture model can focus on synthesizing the underlying material and its low- and medium-frequency variations. The resulting texture is checked for residual semantic elements, material discontinuities, UV seams, and inconsistencies across articulated parts before the pipeline proceeds.

To restore the removed elements, we render the textured object from a set of calibrated viewpoints $\mathcal{V}$. For each view $v \in \mathcal{V}$, we generate image C as

$$I_v^C = R(\mathcal{M}_{\mathrm{atlas}}, \mathcal{T}_{\mathrm{base}}; \Pi_v, \rho), \tag{33}$$

where Π_v contains the camera intrinsics and extrinsics and ρ collects all rendering parameters, including image resolution, lighting, background, antialiasing, and color management. Whenever possible, we use an albedo-only or illumination-neutral rendering mode to prevent shading from being baked into the base-color texture (Zeng et al., 2024; Zhang et al., 2024b). The camera set is selected to cover the surfaces on which semantic details should appear, while avoiding excessively grazing viewing angles.

For each rendered view, the multimodal editor inserts the semantic elements described by $\mathcal{E}$ into I_v^C, using image A as the appearance reference:

$$\left(\widetilde{I}_v^D, M_v\right) = E_\psi\big(I_v^C; I^A, \mathcal{E}, \Pi_v\big), \tag{34}$$

where M_v is the image-space mask of the inserted elements. The edited view, referred to as image D, is formed by

$$I_v^D = (1 - M_v) \odot I_v^C + M_v \odot \widetilde{I}_v^D. \tag{35}$$

This construction guarantees exact pixel preservation outside the edited regions. All views are conditioned on the same semantic-detail manifest, including the same text strings, symbols, colors, styles, and relative placements. The editor is instructed to modify only surface appearance and not to alter the object geometry, silhouette, pose, camera, shading, or background.

Pixel alignment between I_v^C and I_v^D is essential because the edited images are subsequently projected onto the mesh. We therefore keep the image dimensions, camera matrices, rasterization configuration, and color-processing pipeline fixed throughout rendering, editing, and baking. An edited view is rejected if it introduces measurable displacement of the object silhouette, depth discontinuities, part boundaries, or other geometry-derived edges. Pixels close to silhouettes and occlusion boundaries can additionally be excluded from baking, since small image-space errors in these regions may correspond to large surface-space errors.

Rather than baking the complete edited images, we bake only the detail-layer residual

$$\Delta I_v = M_v \odot \left(I_v^D - I_v^C\right). \tag{36}$$

This prevents the multiview images from overwriting the base material or introducing view-dependent illumination into the texture. For a valid texel p, let $\mathbf{x}_p$ and $\mathbf{n}_p$ denote its surface position and normal, respectively. Its projection in view v is

$$\mathbf{q}_{v,p} = \pi_v(\mathbf{x}_p), \tag{37}$$

where π_v is the projection associated with Π_v. Following standard multi-view texture-mapping practice, we combine visibility, view-angle quality, and per-pixel confidence during texture fusion (Callieri et al., 2008; Waechter et al., 2014). Specifically, we define the following view weight:

$$w_{v,p} = \chi_{v,p}\, \kappa_{v,p}\, \left[\max\left(0, \mathbf{n}_p^{\mathsf{T}} \mathbf{d}_{v,p}\right)\right]^{\gamma}, \tag{38}$$

where $\chi_{v,p}$ indicates that $\mathbf{x}_p$ is visible according to the renderer's depth buffer, $\mathbf{d}_{v,p}$ is the direction from the surface toward the camera, $\kappa_{v,p}$ is the confidence assigned to the edited pixel, and γ controls the penalty applied to grazing-angle observations. The baked detail residual is computed as

$$\Delta\mathcal{T}(p) = \frac{\sum_{v\in\mathcal{V}} w_{v,p}\, \Delta I_v(\mathbf{q}_{v,p})}{\sum_{v\in\mathcal{V}} w_{v,p} + \varepsilon}. \tag{39}$$

Inconsistent observations are assigned low confidence or rejected as outliers before aggregation. The final base-color texture is

$$\mathcal{T}_{\text{final}}(p) = \text{clip}(\mathcal{T}_{\text{base}}(p) + \Delta\mathcal{T}(p), 0, 1). \tag{40}$$

Metallic and roughness maps are preserved from $\mathcal{T}_{\text{base}}$ unless the inserted element is explicitly specified as a physically distinct surface layer, such as a metallic label or a painted marking with a different roughness.

The complete procedure is controlled by an agentic validation loop. After reference generation, the agent checks geometric consistency and whether all requested appearance elements are present. After detail removal, it verifies removal completeness and preservation of unedited pixels. After base-material generation, it checks PBR plausibility, cross-part consistency, and the absence of residual text or patterns. For every edited view, it evaluates text correctness using optical character recognition, symbol and pattern similarity, cross-view semantic consistency, and pixel alignment with the corresponding rendering. Finally, the baked asset is re-rendered using the same camera set and compared with the accepted images D to detect missing details, projection errors, seams, and contradictory observations.

Formally, if S_k is the output of stage k and $Q_k(S_k) \in \{0, 1\}$ is its validation result, the agent advances only when

$$Q_k(S_k) = 1. \tag{41}$$

Otherwise, it localizes the failure and repeats the affected stage with a revised prompt, mask, sampling seed, or camera selection. This stage-wise retry mechanism avoids unnecessarily regenerating already accepted materials. By separating semantic details from the base material and introducing explicit geometric, semantic, and multiview quality constraints, the proposed procedure produces sharper and more reliable text, symbols, and graphic patterns while preserving the part-level material consistency and articulation structure of the generated object.

C Bottom-Up Representation and Serialization Details

C.1 Component-to-Link Hierarchy

Let $\mathcal{C} = \{M_i^0\}_{i=1}^N$ denote the geometric components in the canonical rest configuration and $\mathcal{L} = \{\ell_k\}_{k=1}^K$ denote the rigid links. Their relationship is defined by the surjective assignment

$$\pi : [N] \to [K], \qquad I_k = \pi^{-1}(k), \qquad G_k^0 = \bigcup_{i \in I_k} M_i^0. \tag{42}$$

Here, I_k contains the components assigned to link ℓ_k, and G_k^0 is their assembled rest-pose geometry. Every component belongs to exactly one link and every link contains at least one component. All components in I_k therefore share the same rigid transformation during articulation. This construction separates geometric resolution, measured by N, from kinematic resolution, measured by K.

The articulation is represented as a rooted directed tree

$$\mathcal{K} = (\mathcal{L}, \mathcal{E}, r), \tag{43}$$

where $r \in \mathcal{L}$ is the root link and each directed edge $e = (\ell_p, \ell_c) \in \mathcal{E}$ defines a joint from parent ℓ_p to child ℓ_c. We organize the physical attributes as

$$\Phi^L = \{\phi_k^L\}_{k=1}^K, \qquad \Phi^J = \{\phi_e^J\}_{e \in \mathcal{E}}. \tag{44}$$

A link record ϕ_k^L contains the rigid-body, inertial, material, and contact attributes associated with ℓ_k. A joint record ϕ_e^J contains the joint type, parent and child frames, motion axis, limits, damping, stiffness, and actuation parameters when applicable.

C.2 Connection-Aware Spatial Ordering

Autoregressive generation requires a deterministic ordering of geometric components. For a component set $S \subseteq [N]$, we first sort its components by decreasing bounding-box volume, using their original component indices to break ties. We then process the components in this order. A component is attached to the highest-ranked previously processed component whose mesh intersects it or lies within a fixed contact tolerance. If no such component exists, it initializes a new tree. This procedure produces a spatial forest whose edges preferentially follow actual geometric contacts and whose larger supporting components tend to precede smaller attached components.

Small gaps, disconnected decorations, or imperfect meshes may leave multiple trees. To obtain a single deterministic serialization tree, we apply an auxiliary contraction procedure. Let $\mathbf{c}_i^{(0)}$ be the original center of component i and let $\bar{\mathbf{c}}_S$ be the center of the component set. At contraction step t, the center used for connectivity testing is updated as

$$\mathbf{c}_i^{(t+1)} = \bar{\mathbf{c}}_S + \gamma \left(\mathbf{c}_i^{(t)} - \bar{\mathbf{c}}_S \right), \qquad 0 < \gamma < 1. \tag{45}$$

Only component centers are translated during this auxiliary test; component shape, scale, and orientation remain fixed. Whenever components from two different trees first come into contact, we connect the corresponding trees through that component pair. The larger component is selected as the parent, with the predefined size and index ordering resolving simultaneous contacts. Established attachments are not subsequently replaced. Contraction terminates once all trees have merged.

A depth-first traversal of the resulting tree defines the component sequence, with siblings visited in decreasing order of bounding-box volume. The contracted meshes are used only to determine this ordering: the original component geometry, bounding boxes, link assignments, and physical attributes are retained without modification.

C.3 Global and Link-Local Serialization

For a part-aware asset, which lacks rigid-link annotations, we construct one spatial tree over $S = [N]$ and serialize all components using its global depth-first traversal. If the initial construction produces a forest, contraction is performed with respect to the center of the complete object.

For a simulation-ready asset, links are first visited using a depth-first traversal of the annotated kinematic tree. Within each visited link ℓ_k, we independently construct a spatial tree over $S = I_k$ and serialize its components using the corresponding link-local traversal. If I_k initially forms a forest, contraction is restricted to that link and uses its local center. Components assigned to different links are therefore never connected by the auxiliary spatial-ordering procedure.

The resulting grammars are

$$\begin{aligned}
\mathbf{s}_{\text{part}} &= \big[\langle\text{PART-AWARE}\rangle, \langle\text{ASSET}\rangle, \mathbf{s}_{\text{layout}}, \mathbf{s}^{\text{geo}}_{\text{global}}, \langle\text{EOS}\rangle\big], \\
\mathbf{s}_{\text{sim}} &= \big[\langle\text{SIM-READY}\rangle, \langle\text{ASSET}\rangle, \mathbf{s}_{\text{layout}}, \langle\text{LINKS}\rangle, \\
&\qquad \big(\mathbf{s}^{\text{geo}}_{\ell_k}, \text{CreateLink}(\phi^L_k)\big)_{k=1}^{K}, \langle\text{JOINTS}\rangle, \\
&\qquad \big(\,\text{CreateJoint}(\ell_p, \ell_c, \phi^J_e)\big)_{e\in\mathcal{E}}, \langle\text{EOS}\rangle\big].
\end{aligned} \tag{46}$$

The explicit records are deterministically converted into URDF or USD. Because the representation exposes link- and joint-level physical attributes as structured fields, users may inspect or modify these values before export without altering the generated component geometry or kinematic topology.

D Heterogeneous Co-Training Details

D.1 Task-Conditioned Grammars

Part-aware and simulation-ready assets provide supervision at different structural levels. Part-aware examples supervise the asset layout and component geometry, while simulation-ready examples additionally supervise component grouping, link properties, joint topology, and joint dynamics. We prepend each sequence with a task token

$$\tau_d \in \{\langle\text{PART-AWARE}\rangle, \langle\text{SIM-READY}\rangle\}, \qquad \mathbf{s}_d = (\tau_d, s_1, \ldots, s_T). \tag{47}$$

Conditioned on the textual object description x and the requested task grammar, the autoregressive distribution is

$$p_\theta(\mathbf{s}_d \mid x, \tau_d) = \prod_{t=1}^{T} p_\theta\left(s_t \mid \mathbf{s}_{<t}, x, \tau_d\right). \tag{48}$$

At inference time, the part-aware token requests a globally ordered component sequence, whereas the simulation-ready token requests the complete hierarchy of components, links, physical attributes, and joints.

Although the two domains do not share an identical global ordering, they use the same component bounding-box representation, geometry tokenizer, component-generation syntax, size-based parent preference, and local connection-aware traversal. Part-aware examples therefore improve the shared conditional distribution over component geometry. Simulation-ready examples additionally teach the model when to close a component block, how to assign link-level properties, and how to connect the resulting links. Missing kinematic or physical labels are never imputed for part-aware examples.

D.2 Token-Family-Balanced Objective

Let

$$\mathcal{F} = \{\text{layout}, \text{geometry}, \text{link}, \text{joint}\} \tag{49}$$

denote the four token families. Layout tokens describe the asset-level spatial structure; geometry tokens encode component bounding boxes and local shapes; link tokens contain link boundaries and link-level physical attributes; and joint tokens contain kinematic topology and joint-level parameters.

Let $\Omega_f(\mathbf{s})$ denote the supervised positions belonging to family f. We first average the loss independently within each active family:

$$\mathcal{L}_f(\mathbf{s}) = -\frac{1}{|\Omega_f(\mathbf{s})|} \sum_{t\in\Omega_f(\mathbf{s})} \log p_\theta\left(s_t \mid \mathbf{s}_{<t}, x, \tau_d\right). \tag{50}$$

Table 2: Staged training schedule. TP and CP denote tensor parallelism and context parallelism, respectively.

Stage	Max. length	Training data	Steps	TP	CP
1	20K	Simulation-ready only	20,000	4	1
2	50K	Simulation-ready only	5,000	4	2
3	100K	Simulation-ready + part-aware	5,000	4	4
4	150K	Simulation-ready + part-aware	5,000	4	6
5	256K	Simulation-ready + part-aware	2,000	4	8
		Total	37,000	–	–

The active families are

$$\begin{aligned} \mathcal{F}_{\text{part}} &= \{\text{layout}, \text{geometry}\}, \\ \mathcal{F}_{\text{sim}} &= \{\text{layout}, \text{geometry}, \text{link}, \text{joint}\}. \end{aligned} \tag{51}$$

Using equal weights among active families gives

$$\mathcal{L}_d(\mathbf{s}) = \frac{1}{|\mathcal{F}_d|} \sum_{f \in \mathcal{F}_d} \mathcal{L}_f(\mathbf{s}). \tag{52}$$

This normalization prevents long geometry spans from overwhelming the substantially fewer link, joint, and physical-attribute records.

During mixed-data stages, part-aware and simulation-ready examples are sampled with equal probability:

$$\mathcal{L}_{\text{mix}} = \frac{1}{2}\mathbb{E}_{\mathbf{s}\sim\mathcal{D}_{\text{part}}}[\mathcal{L}_{\text{part}}(\mathbf{s})] + \frac{1}{2}\mathbb{E}_{\mathbf{s}\sim\mathcal{D}_{\text{sim}}}[\mathcal{L}_{\text{sim}}(\mathbf{s})]. \tag{53}$$

D.3 Staged Training Schedule

Following MegaParts Liao et al. (2026), we adopt Qwen3-VL-8B (Bai et al., 2025) as the generative backbone and fine-tune it with Megatron-LM (Shoeybi et al., 2019). Because the computational cost and per-step wall-clock time increase substantially with context length, we progressively increase the maximum sequence length during training. The first two stages use only simulation-ready assets. Part-aware data are introduced from the 100K stage onward, after which the two sources are sampled with equal probability. The complete schedule is reported in Table 2. We optimize the model using fused Adam ($\beta_1 = 0.9$, $\beta_2 = 0.95$, $\epsilon = 10^{-8}$) with a peak learning rate of 5×10^{-5}, cosine decay to 10^{-7}, a linear warm-up from zero over the first $1{,}000$ updates, and an effective constant weight decay of 0.033; we use a global batch size of 256, BF16 mixed-precision training, global gradient-norm clipping at 1.0, and a random seed of 42 for model, data, and modality sampling.

The model is optimized for 37,000 steps in total, after which the training loss exhibits no appreciable further decrease.

E More Details about Mobility-PhyX

In this section, we provide additional details about Mobility-PhyX, including the implementation details of the dual-level data augmentation strategy and the VLM-assisted coarse-to-fine physical attribute grounding pipeline.

E.1 Dual-Level Data Augmentation

E.1.1 Retrieval-Based Part Geometry Augmentation

Given a procedurally generated articulated asset, we retrieve geometrically diverse parts from external 3D asset collections with the same semantic category and use them to replace selected components of the original asset. This operation preserves the semantic and functional role of each part while introducing richer geometric variations.

Following Infinite-Mobility (Lian et al., 2025), we use the bounding box of each procedural part to determine the initial scale and placement of the retrieved mesh. To further ensure compatibility between the retrieved geometry and the surrounding structure, we define category-specific critical support points that characterize the functional interface between neighboring parts. The retrieved mesh is aligned with the corresponding support points of the procedural part, ensuring geometric consistency after replacement. Compared with Infinite-Mobility (Lian et al., 2025), which focuses on a limited set of part categories, Mobility-PhyX extends this retrieval-based augmentation procedure to a broader range of semantic part categories. This extension increases the diversity of generated articulated assets while maintaining semantic and functional consistency.

E.1.2 Bidirectional Kinematic-Tree Augmentation

Part-level replacement improves the geometric diversity of individual components but does not modify the underlying articulation topology. To further enrich kinematic structures, Mobility-PhyX introduces two complementary tree augmentation operations: child subtree replacement and parent tree replacement.

Following the standard URDF convention, an articulated object is represented as a rooted tree $\mathcal{T} = (\mathcal{V}, \mathcal{E})$, where each node $v \in \mathcal{V}$ denotes a rigid link and each directed edge $e \in \mathcal{E}$ denotes an articulation joint. For an asset source $s \in \{\mathrm{proc}, \mathrm{cand}\}$, where proc represents procedurally generated assets and cand represents candidate assets retrieved from external 3D asset collections, a selected joint is defined as:

$$e^s = \left(v^s_{\mathrm{parent}}, v^s_{\mathrm{child}}\right). \tag{54}$$

Here, v^s_{parent} and v^s_{child} represent the parent link and child link of joint e^s, respectively. Removing the selected joint partitions the URDF tree into two connected structures: the parent tree $\mathcal{T}^s_{\mathrm{parent}}(e^s)$ and the child subtree $\mathcal{T}^s_{\mathrm{child}}(e^s)$.

The parent tree is defined as the connected component containing v^s_{parent} and its ancestors, while the child subtree is rooted at v^s_{child} and contains the child link, its descendants, and the corresponding internal joints. The original articulated tree can therefore be represented as:

$$\mathcal{T}^s = \mathrm{Compose}\left(\mathcal{T}^s_{\mathrm{parent}}(e^s), e^s, \mathcal{T}^s_{\mathrm{child}}(e^s)\right). \tag{55}$$

Given a procedural asset $\mathcal{T}^{\mathrm{proc}}$ and a candidate asset $\mathcal{T}^{\mathrm{cand}}$, we select corresponding joints by matching their child links. Specifically, $v^{\mathrm{proc}}_{\mathrm{child}}$ and $v^{\mathrm{cand}}_{\mathrm{child}}$ are required to share similar semantic and functional roles, while their corresponding joints must have compatible motion types, such as revolute or prismatic motion.

Before replacement, both assets are normalized into a shared category-specific canonical pose, orientation, and scale convention. The target dimensions and placement constraints are determined by the procedural generator. A scale factor is computed from the bounding boxes of the matched child links and applied to all candidate geometries involved in the replacement. The child link serves as the reference because it defines the articulation interface between the retained and replaced structures.

The scaled candidate geometries are not directly placed according to their original donor link poses. Instead, they are reassembled using the geometry-aware placement rules of the procedural generator. Category-specific support points are used to align the interface links, and the remaining candidate geometries are recursively instantiated according to the transferred parent-child hierarchy, semantic roles, scaled dimensions, and procedural placement constraints.

We denote the replacement direction as $r \in \{\mathrm{child}, \mathrm{parent}\}$. For child subtree replacement, the procedural parent tree is retained and the candidate child subtree is inserted. For parent tree replacement, the procedural child subtree is retained and the candidate parent tree is inserted.

The replacement operation is formulated as:

$$\widetilde{\mathcal{T}}(r) = \begin{cases} \mathrm{Compose}\left(\mathcal{T}^{\mathrm{proc}}_{\mathrm{parent}}(e^{\mathrm{proc}}), \widehat{e}(r), \widehat{\mathcal{T}}^{\mathrm{cand}}_{\mathrm{child}}(e^{\mathrm{cand}})\right), & r = \mathrm{child}, \\ \mathrm{Compose}\left(\widehat{\mathcal{T}}^{\mathrm{cand}}_{\mathrm{parent}}(e^{\mathrm{cand}}), \widehat{e}(r), \mathcal{T}^{\mathrm{proc}}_{\mathrm{child}}(e^{\mathrm{proc}})\right), & r = \mathrm{parent}. \end{cases} \tag{56}$$

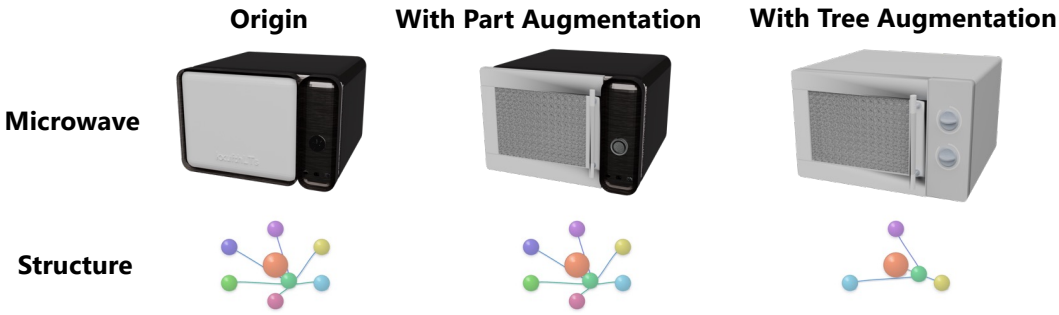

Figure 6: Effectiveness of our Dual-Level Data Augmentation.

The reconstructed boundary joint is defined as:

$$\widehat{e}(r) = \begin{cases} \left(v^{\text{proc}}_{\text{parent}}, \widehat{v}^{\text{cand}}_{\text{child}}\right), & r = \text{child}, \\ \left(\widehat{v}^{\text{cand}}_{\text{parent}}, v^{\text{proc}}_{\text{child}}\right), & r = \text{parent}. \end{cases} \tag{57}$$

The candidate asset provides transferred link geometries, parent-child hierarchy, semantic organization, and joint motion types. However, original donor link poses and geometry-dependent joint parameters are not directly copied. After geometry reassembly, joint origins, motion axes, and motion ranges are reconstructed using the same geometry-aware articulation procedure used by the procedural generator. Motion limits are further constrained according to collision conditions on the reassembled geometry.

Through the above Dual-Level Data Augmentation pipeline, we introduce variations in the number of movable links, parent-child dependencies, articulation depth, link connectivity, branch composition, and joint-type organization, the effectiveness of this data augmentation design is shown in Fig. 6.

E.2 VLM-Assisted Coarse-to-Fine Physical Attribute Grounding

E.2.1 Link-Level Rigid-Body Property Grounding

For each rigid link, Mobility-PhyX assigns material composition, density, and mass. These properties follow a progressive dependency: material composition is inferred from visual appearance and functional role, density is determined from the material composition, and mass is obtained from density and effective occupied volume.

For procedurally generated assets, material assignments are predefined. Therefore, density values can be obtained from a material-density lookup table. When a link contains multiple materials and exact volume fractions are unavailable, the effective density is approximated using the mean density of the constituent materials.

Computing exact mesh volume is challenging for articulated assets due to open surfaces, non-manifold geometry, duplicated faces, double-layered surfaces, and other mesh irregularities. Therefore, Mobility-PhyX uses bounding-box volume and introduces a statistical calibration procedure to estimate the discrepancy between bounding-box volume and effective occupied volume.

The bounding-box volume is defined as:

$$V_i^{\text{bbox}} = h_i w_i d_i. \tag{58}$$

We collect articulated assets from public datasets, including PartNet-Mobility (Xiang et al., 2020). For each link, material-aware multi-view images are rendered and annotated using a VLM-assisted pipeline to obtain material composition, density, and mass attributes based on visual appearance, semantic identity, and functional role.

For each annotated link, we compute:

$$\delta_i = \frac{m_i}{\rho_i V_i^{\text{bbox}}}. \tag{59}$$

The calibration factor captures the statistical discrepancy between bounding-box volume and effective occupied volume among semantically corresponding links. Links are grouped according to object category and semantic label, and the calibration factors are modeled using Gaussian distributions. During procedural generation, calibration factors are sampled from the corresponding distribution.

For a generated link v_k, the effective volume and mass are estimated as:

$$\widehat{V}_k = \delta_g V_k^{\mathrm{bbox}}, \qquad \widehat{m}_k = \rho_g \widehat{V}_k. \tag{60}$$

E.2.2 Joint-Level Dynamic Properties

Mobility-PhyX further assigns dynamic attributes to movable joints, including stiffness K and damping D. For each object category and joint type, the VLM provides plausible interaction force and motion duration annotations according to the visual appearance, semantic identity, and functional role of the articulated mechanism. These annotations are then converted into joint dynamic parameters using physics-based formulations.

We adopt a force-based joint drive model and estimate joint stiffness and damping according to whether a joint exhibits an inherent restoring behavior. Since directly measuring joint forces and dynamic responses is infeasible for procedurally generated assets, the proposed formulation uses VLM-provided interaction annotations together with analytical physical relationships to generate scalable and physically consistent joint parameters.

Non-Restoring Joints. For joints without inherent elastic return behavior, such as ordinary drawers and freely rotating doors, the stiffness is set to zero. Instead, their resistance during motion is represented using an equivalent damping coefficient. This formulation does not aim to recover the exact mechanical properties of each individual mechanism, but provides a physically consistent approximation that can be applied across large collections of articulated assets.

For each object category and joint type, the VLM provides plausible ranges of the actuation force required to move the joint through its motion range and the corresponding traversal duration. During asset generation, the actuation force F_j and traversal time T_j are sampled from these ranges and converted into damping parameters according to the physical relationship between force, velocity, torque, and angular velocity.

For a prismatic joint, the damping coefficient is computed from the relationship between the applied force and the average linear velocity:

$$K_j = 0, \qquad D_j = \frac{F_j}{v_j} = \frac{F_j T_j}{\Delta x_j}, \tag{61}$$

where Δx_j denotes the full travel distance and $v_j = \Delta x_j / T_j$ denotes the corresponding average linear velocity.

For a revolute or continuous joint, the applied force is converted into torque according to the lever arm determined from the procedural geometry. The damping coefficient is then obtained from the relationship between torque and angular velocity:

$$K_j = 0, \qquad D_j = \frac{\tau_j}{\omega_j} = \frac{F_j \ell_j^F T_j}{\Delta \theta_j}, \tag{62}$$

where $\Delta\theta_j$ denotes the prescribed angular displacement and ω_j denotes the corresponding average angular velocity. The lever arm ℓ_j^F is directly determined from the procedural geometry. For continuous joints without a finite motion limit, $\Delta\theta_j$ denotes a predefined reference angular displacement.

Restoring Joints. For joints with inherent elastic return behavior, such as spring-loaded buttons and self-returning mechanisms, we estimate stiffness under a quasi-static equilibrium assumption. Specifically, the VLM provides a plausible range of the static force required to hold the joint at a prescribed displacement, from which a holding force F_j^{stat} is sampled.

We model restoring behavior using a linear spring approximation around the operating displacement. This approximation provides a simple and physically consistent formulation for generating joint dynamic attributes while avoiding the need to model complex internal mechanisms.

For a prismatic joint, the stiffness is computed as:

$$K_j = \frac{F_j^{\text{stat}}}{\Delta x_j}, \qquad D_j = 2\zeta_j \sqrt{K_j m_j^{\text{child}}}, \tag{63}$$

where K_j is obtained by balancing the restoring force with the sampled static holding force at displacement Δx_j.

For a revolute or continuous joint, the sampled static force is converted into a holding torque through:

$$\tau_j^{\text{stat}} = F_j^{\text{stat}} \ell_j^F, \tag{64}$$

and the corresponding stiffness and damping are computed as:

$$K_j = \frac{F_j^{\text{stat}} \ell_j^F}{\Delta \theta_j}, \qquad D_j \approx 2\zeta_j \ell_j^I \sqrt{K_j m_j^{\text{child}}}, \tag{65}$$

where ℓ_j^F denotes the perpendicular lever arm for converting force into torque, and ℓ_j^I denotes the effective distance from the joint axis to the child link's center of mass.

For restoring joints, we adopt an empirically defined damping ratio range $\zeta_j \sim \mathcal{U}(1.0, 1.2)$, which covers the critically to mildly over-damped regime commonly used for self-returning mechanisms. This range provides rapid stabilization without visible oscillations while introducing moderate dynamic diversity.

The grounded mass m_j^{child} of the child link directly connected to the joint is used as a scalable approximation of the effective moving mass. This approximation is suitable for common object-level articulated mechanisms, such as drawers, doors, and buttons, where the movable joint typically connects a fixed parent structure with a single movable child component. For rotational joints, we further use a point-mass approximation based on the grounded child-link mass and the effective lever arm. Directly computing rotational inertia from arbitrary articulated meshes can be unreliable due to mesh irregularities, including open surfaces, non-manifold structures, and inconsistent topology. Therefore, the point-mass approximation provides a stable and scalable estimate of the rotational response for simulation-oriented joint dynamic generation.

E.2.3 Human-Expert Validation of VLM-Assisted Physical Annotations

To evaluate the reliability of the VLM-assisted annotation pipeline, we conducted a human-expert correction study on PartNet-Mobility(Xiang et al., 2020).

For each asset, domain experts were provided with multi-view renderings, component and link labels, metric dimensions, and visualizations of the articulated motion. They reviewed the VLM-generated annotations for material composition, mass, restoring behavior, actuation-force range, and traversal-duration range. Each annotation could either be accepted without modification or replaced with an expert-corrected value or interval.

Let $\hat{y}_i^a$ denote the VLM annotation for attribute a of sample i, and let $y_i^{a,*}$ denote its expert-corrected value. For categorical attributes, an annotation is considered erroneous when it differs from the expert correction. For a continuous attribute, we compute its relative deviation as

$$e_i^a = \frac{\left|\hat{y}_i^a - y_i^{a,*}\right|}{\left|y_i^{a,*}\right| + \epsilon}, \tag{66}$$

where ϵ is used for numerical stability. For interval-valued annotations, the deviations of the lower and upper endpoints are averaged. An annotation is counted as requiring correction when $e_i^a > \tau_a$, where τ_a is the predefined tolerance for attribute a. The correction ratio is

$$R_a = \frac{1}{|\Omega_a|} \sum_{i \in \Omega_a} \mathbb{1}[e_i^a > \tau_a], \tag{67}$$

where Ω_a is the set of reviewed samples containing attribute a. We additionally report the mean relative deviation to quantify the magnitude of the corrections.

As shown in Table 3, the VLM-generated annotations exhibit strong agreement with the human experts, with an average correction ratio of 4.82%. Restoring-behavior annotations require no correction, while material composition, mass, and traversal-duration annotations achieve correction

Table 3: Human-expert evaluation of the VLM-assisted physical annotations on PartNet-Mobility.

Attribute	Correction ratio (%) ↓	Relative deviation (%) ↓
Material composition	1.21	–
Mass	1.09	5.22
Restoring behavior	0.00	–
Actuation-force range	20.8	12.9
Traversal-duration range	1.00	5.09

ratios close to 1%. Across the continuous attributes, the mean relative deviation is 7.74%; even for actuation-force ranges, the relative deviation remains limited to 12.9%. These results demonstrate that the VLM-assisted pipeline produces reliable and physically plausible annotations across the evaluated attributes, providing effective physical priors for large-scale simulation-ready asset construction.

Because PartNet-Mobility does not contain measured instance-level physical properties, this experiment evaluates agreement with expert judgments of physically plausible operating ranges rather than recovery of real-world measurements. Accordingly, the corrected annotations should still be interpreted as expert-validated behavioral priors, not as instance-specific physical ground truth.

F Details of Training Dataset

During co-training of part-aware and sim-ready assets, we sample the two corpora with equal probability. During training, We limit the proportion of procedurally generated assets to prevent overfit and upsample examples containing small kinematic groups to improve joint generation for fine-grained parts. We exclude assets that cannot be faithfully reconstructed by the geometry VQ-VAE, contain incomplete kinematic chains, or exceed the maximum sequence length of 256K tokens.

The original training dataset exhibits substantial imbalance in sample length, category frequency, and joint-count distribution. In particular, sample length, approximated by the number of `create_mesh` calls, varies considerably across samples, while a small number of object categories and joint counts dominate the dataset. To mitigate these biases and prevent the model from overfitting to frequently occurring structures, we perform dataset balancing across these three dimensions. The resulting joint-count distribution, sample-length distribution, and category distribution are shown in Figs. 7, 8, and 9, respectively.

To improve image-conditioned 3D generation, we use FLUX-Depth (Black Forest Labs, 2024; Black Forest Labs et al., 2025), guided by our material annotations, to synthesize diverse training images.

G Ablation Studies

We evaluate three principal design choices: part-aware co-training, connection-aware DFS ordering, and the bottom-up component-to-link representation. All variants follow the same evaluation protocol, and the model architecture, optimization settings, and training budget are kept fixed wherever applicable. In the absence of part-aware co-training, the model is trained exclusively on simulation-ready assets. Without connection-aware DFS ordering, components and links retain their original serialization order. We evaluate the bottom-up representation against two flat alternatives representing different trade-offs. **Fixed-joint assembly** treats every fine-grained mesh as an individual link and connects meshes belonging to the same rigid body using fixed joints. **Link-mesh generation** instead merges all components of each rigid link into a single mesh sequence. Because these representations inherently alter sequence length, mesh complexity, and joint-label frequency, their results should be interpreted as end-to-end comparisons between alternative representations rather than isolated changes to a single token type.

Table 4 shows that the two flat representations exhibit complementary strengths and limitations. Fixed-joint assembly retains fine-grained component geometry and consequently approaches the full model in PSNR, Chamfer distance, and F-score. However, representing every component as a link introduces redundant kinematic nodes and fixed joints between meshes that should move as

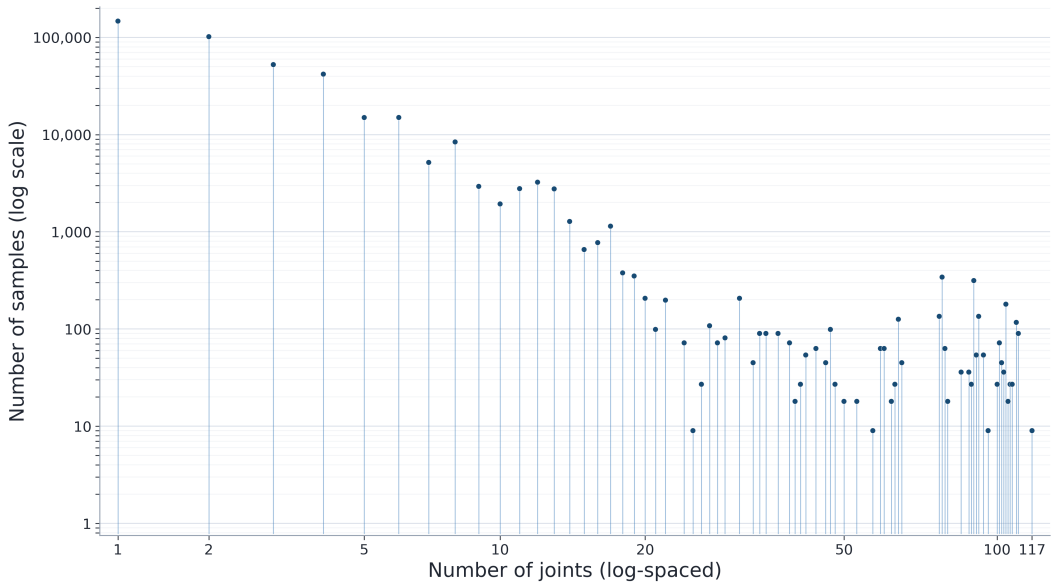

Figure 7: Joint number (create_joint number) distribution of our balanced training dataset.

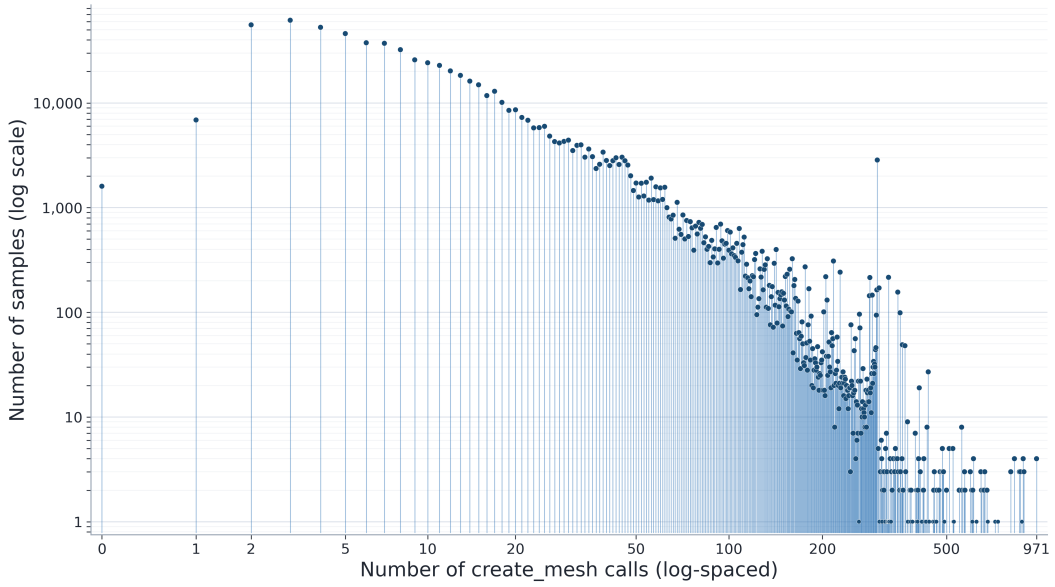

Figure 8: Part number (create_mesh number) distribution of our balanced training dataset.

one rigid body. Fixed joints therefore dominate the joint-label distribution, leaving substantially less relative supervision for revolute and prismatic joints. This imbalance is consistent with the marked reduction in joint-type accuracy and the larger axis and limit errors.

Link-mesh generation establishes a simpler one-to-one correspondence between meshes and rigid links. It also shortens the output sequence and increases the relative contribution of kinematic tokens during training. These properties simplify joint-type prediction, for which this variant achieves the highest accuracy. However, merging all components within a link often creates geometrically complex or disconnected meshes that are difficult for the geometry VQ-VAE to reconstruct. Consequently, this variant suffers a substantial loss in geometric quality, and its axis error remains higher than that of the full model. Its strong joint-type accuracy therefore comes at the cost of severely degraded geometry.

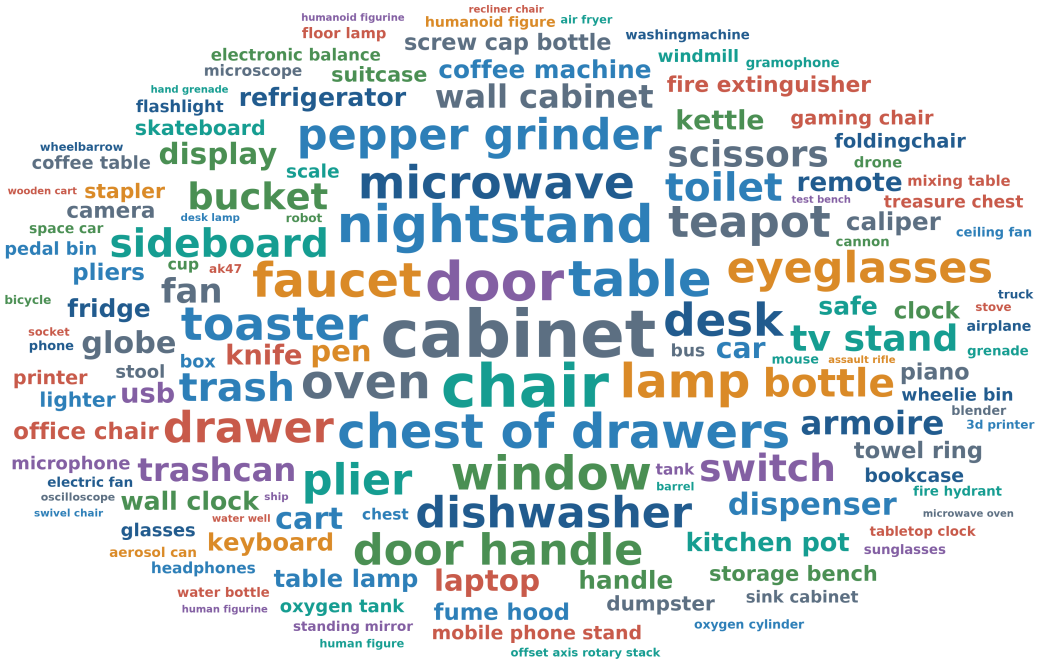

Figure 9: Category coverage of our training dataset.

Table 4: Ablation of the bottom-up representation, connection-aware DFS ordering, and part-aware co-training. The best and second-best results are shown in **bold** and underlined, respectively.

Variant	Geometry Quality			Articulation Quality		
	PSNR ↑	CD ↓	F-score ↑	Type Acc. ↑	Axis Err. ↓	Limit Err. ↓
Fixed-joint assembly	11.73	0.022	0.51	0.22	44.3	0.55
Link-mesh generation	3.21	0.23	0.23	**0.94**	7.7	0.04
w/o connection-aware DFS	9.51	0.10	0.34	0.88	4.4	0.07
w/o part-aware co-training	5.56	0.13	0.22	0.90	5.5	0.11
Full model	**12.17**	**0.020**	**0.52**	0.92	**3.4**	**0.03**

Removing the connection-aware DFS ordering reduces performance across both metric groups. The original serialization no longer provides a spatially coherent, coarse-to-fine construction order, making the relationships between nearby components less explicit. Qualitative inspection indicates that small components are particularly sensitive to this change because their prediction depends more strongly on the previously generated surrounding structure. Nevertheless, this variant retains competitive articulation accuracy, suggesting that ordering primarily improves structural consistency rather than determining articulation alone.

Removing part-aware co-training causes a pronounced decline in geometry quality, demonstrating the importance of large-scale component-level supervision for geometric generalization. Articulation performance also decreases, although less severely. This suggests that a weaker understanding of geometry tokens propagates to subsequent link and joint predictions, since accurate articulation depends on recognizing the shapes and spatial relationships of the generated components.

Overall, the full model achieves the best geometry quality, joint-axis accuracy, and joint-limit accuracy, while attaining the second-best joint-type accuracy. Although link-mesh generation performs slightly better in joint-type classification, its geometric degradation is substantial. The results therefore support the bottom-up representation as the most balanced formulation and show that structured ordering and part-aware co-training provide complementary improvements.

H Evaluation Protocol and Metric Definitions

Evaluation setting. We conduct all evaluations on the held-out PartNet-Mobility test split following the protocol of SINGAPO (Liu et al., 2025). All assets in the held-out PartNet-Mobility test split, together with their VLM-generated and expert-corrected annotations, were excluded from augmentation, retrieval, calibration-statistics estimation, and all model-training procedures. Image-conditioned methods receive a reference rendering, whereas text-conditioned methods receive the corresponding object description. Both settings use the same target assets and evaluation procedures.

We compare image-conditioned generation against PhysX-Anything (Cao et al., 2026a), SINGAPO (Liu et al., 2025), and Articraft-I (Zhou et al., 2026), and compare text-conditioned generation against Articraft-T. SIMART (Zhang et al., 2026), ArtLLM (Wang et al., 2026), Particulate (Li et al., 2026), and URDF-Anything (Li et al., 2025d) receive a reference mesh or point cloud and are evaluated separately as structural-recovery methods. A metric is reported only when the corresponding output is produced by a method.

The reference physical attributes used for evaluation are obtained through the human-expert correction procedure described in Section E.2.3. Raw VLM-generated physical annotations are not used directly as evaluation targets.

Geometry metrics. Predicted and reference assets are rendered as depth maps from 30 viewpoints uniformly distributed on a unit sphere. We report the average peak signal-to-noise ratio (PSNR) between corresponding depth renderings. This measures geometric similarity while avoiding the influence of texture, material, color, and lighting.

We additionally sample point sets $\widehat{\mathcal{P}}$ and $\mathcal{P}$ from the predicted and reference surfaces, respectively, and compute their bidirectional Chamfer distance:

$$\mathrm{CD}\left(\widehat{\mathcal{P}}, \mathcal{P}\right) = \frac{1}{|\widehat{\mathcal{P}}|} \sum_{\hat{\mathbf{p}} \in \widehat{\mathcal{P}}} \min_{\mathbf{p} \in \mathcal{P}} \|\hat{\mathbf{p}} - \mathbf{p}\|_2 + \frac{1}{|\mathcal{P}|} \sum_{\mathbf{p} \in \mathcal{P}} \min_{\hat{\mathbf{p}} \in \widehat{\mathcal{P}}} \|\mathbf{p} - \hat{\mathbf{p}}\|_2 . \tag{68}$$

The F-score at threshold $\tau = 0.05$ is defined as

$$F_\tau = \frac{2 P_\tau R_\tau}{P_\tau + R_\tau}, \tag{69}$$

where P_τ and R_τ denote the fractions of predicted and reference points, respectively, whose nearest-neighbor distances are below τ. Higher PSNR and F-score indicate better performance, whereas a lower Chamfer distance indicates greater geometric fidelity.

We evaluate semantic consistency between the generated geometry and its conditioning description using a geometry-aware CLIP score. We avoid textured renderings because Articraft (Zhou et al., 2026) does not provide PBR texture generation; evaluating rendered appearance would therefore favor methods that generate textures and materials. Instead, each asset is rendered as camera-space surface-normal maps from $V = 8$ approximately uniform viewpoints using depth-tested triangle rasterization.

Let $N_{i,v}$ denote the normal rendering of asset i from viewpoint v, and let t_i denote its text description. Using the frozen CLIP ViT-L/14 image and text encoders, f_I and f_T, we compute

$$S_{\mathrm{CLIP}} = \frac{100}{NV} \sum_{i=1}^{N} \sum_{v=1}^{V} \frac{f_I(N_{i,v})^\top f_T(t_i)}{\|f_I(N_{i,v})\|_2 \, \|f_T(t_i)\|_2}. \tag{70}$$

A higher S_{CLIP} indicates stronger semantic consistency between the generated geometry and the input description.

Scale, material, and description. Absolute-scale error is defined as

$$E_{\mathrm{scale}} = \|\widehat{\mathbf{s}} - \mathbf{s}\|_2 , \tag{71}$$

where $\widehat{\mathbf{s}}, \mathbf{s} \in \mathbb{R}^3$ denote the predicted and reference bounding-box dimensions in physical units.

Following PhysX-Anything (Cao et al., 2026a), material and part-description predictions are evaluated using PSNR between their corresponding predicted and reference surface maps. The reference material maps are constructed using the expert-corrected material annotations described in Section E.2.3. The material score therefore measures agreement with expert-validated material assignments rather than the numerical accuracy of individual physical parameters.

Individual kinematic score. PhysX-Anything evaluates generated articulations using a relative-ranking protocol (Cao et al., 2026a). Each predicted joint is actuated over its valid range, and the resulting articulation is rendered under a fixed camera and motion schedule. A frozen vision-language model (VLM) ranks multiple anonymized candidate videos according to their consistency with a reference articulation. Given M candidates and the rank $r \in \{1, \dots, M\}$ assigned to a prediction, the original normalized score is

$$S_{\text{rank}} = \frac{M - r}{M - 1}. \tag{72}$$

This rank-based score depends on the size and composition of the candidate set. The score assigned to a prediction can change when another method is added or removed, even if the prediction itself remains unchanged. For example, a prediction ranked second receives a score of $2/3$ among four candidates but $3/4$ among five candidates. Small or ambiguous differences between candidates can also change their ordering and produce discontinuous score changes. The protocol therefore measures relative performance within a particular candidate pool rather than the absolute kinematic quality of an individual prediction, limiting reproducibility across experiments.

We instead use a reference-conditioned individual scoring protocol. Each predicted articulation is evaluated independently against its corresponding reference video using a frozen VLM and a fixed rubric. Only one prediction and its reference are presented in each evaluation, ensuring that the resulting score is independent of other evaluated methods. The VLM is instructed to ignore texture, material, lighting, camera orientation, and geometric differences unrelated to identifying the articulated component.

For each matched joint j, the VLM assigns an integer score $q_{j,c} \in \{0, \dots, 4\}$ for each of five criteria: moving-part correspondence, motion type, joint axis and pivot, motion range and direction, and physical stability. The normalized per-joint kinematic score is

$$s_j = \frac{1}{4}\left(0.20q_{j,\text{part}} + 0.25q_{j,\text{type}} + 0.25q_{j,\text{axis}} + 0.20q_{j,\text{range}} + 0.10q_{j,\text{stability}}\right). \tag{73}$$

The criterion weights sum to one, and division by four maps the maximum weighted score to one.

For an articulated asset, the final kinematic score is

$$S_{\text{kin}} = \frac{\sum_{j \in \mathcal{M}} s_j}{\max\left(N_{\text{ref}}, N_{\text{pred}}\right)}, \tag{74}$$

where $\mathcal{M}$ is the set of matched predicted and reference joints, and N_{ref} and N_{pred} are the numbers of reference and predicted joints, respectively. Unmatched joints contribute zero, while the denominator penalizes both missing and superfluous joints. The final score lies in $[0, 1]$. A score of one indicates that every reference joint is correctly recovered without additional joints and that all evaluated motion criteria receive the maximum rating. Because each prediction is scored independently under the same rubric, the metric remains unchanged when other methods are added to or removed from the benchmark. We use GPT-5.6 Sol (OpenAI, 2026b) as the frozen VLM for this evaluation.

Load success. The load success rate is the percentage of generated assets that can be exported to URDF or USD, loaded into the target simulator, and simulated for a fixed duration without parsing failures, invalid transformations, or numerical instability.

Joint parameters. For joints connecting matched links, we report joint-type accuracy and errors in the predicted axis, origin, and motion limits. The sign-invariant angular error between a predicted axis $\widehat{\mathbf{a}}$ and its reference $\mathbf{a}$ is

$$E_{\text{axis}} = \frac{180}{\pi} \arccos\left(\left|\frac{\widehat{\mathbf{a}}^{\top}\mathbf{a}}{\|\widehat{\mathbf{a}}\|_2 \|\mathbf{a}\|_2}\right|\right), \tag{75}$$

measured in degrees.

Joint-origin error is the Euclidean distance between the predicted and reference origins, normalized by the object bounding-box diagonal. Joint-limit error is the mean absolute error of the predicted lower and upper limits. Revolute-joint errors are normalized by 2π, whereas prismatic-joint errors are normalized by the object bounding-box diagonal.

Physical parameters. We evaluate mass, joint damping, and joint stiffness against the expert-corrected physical references described in Section E.2.3. For mass, the reference value is the link-level mass reviewed and, where necessary, corrected by domain experts. For damping and stiffness, we first replace the upstream VLM annotations—including restoring behavior, interaction-force ranges, and traversal-duration ranges—with their expert-corrected values. The corresponding joint parameters are then computed using the physical grounding procedure described in Section E.2.2. Consequently, the reported errors are not calculated against the uncorrected outputs of the VLM-assisted annotation pipeline.

For a physical quantity $q \in \{\mathrm{mass}, \mathrm{damping}, \mathrm{stiffness}\}$, we report the normalized mean absolute error

$$E_q = \frac{1}{|\Omega_q|} \sum_{i \in \Omega_q} \frac{|\widehat{q}_i - q_i^*|}{|q_i^*| + \epsilon}, \tag{76}$$

where $\widehat{q}_i$ is the predicted value, q_i^* is its expert-corrected reference value, Ω_q contains the matched links or joints for which the corresponding quantity is defined, and ϵ is a small constant used for numerical stability.

Because PartNet-Mobility does not provide measured instance-level physical properties, these metrics quantify agreement with expert-validated behavioral priors rather than recovery of exact real-world mechanical parameters.

H.1 VLM Prompt for Individual Kinematic Evaluation

For reproducibility, we provide the complete prompt used for the reference-conditioned kinematic evaluation. Each reference and predicted joint is rendered separately and assigned an anonymous identifier. The VLM returns only joint correspondences and integer rubric scores; all weighted scores in Equations equation 73 and equation 74 are computed deterministically outside the VLM.

> **Role.** You are evaluating the kinematic consistency of one generated articulated asset against one reference asset. Evaluate this prediction independently. Do not compare it with outputs from any other method.
>
> **Input.** You are given reference joint videos labeled $R_1, \ldots, R_{N_{\mathrm{ref}}}$ and predicted joint videos labeled $P_1, \ldots, P_{N_{\mathrm{pred}}}$. Each video actuates exactly one joint over its valid range while keeping all other joints fixed. The reference and predicted videos use the same fixed camera and motion schedule whenever possible.
>
> **Task 1: Joint matching.** Establish a one-to-one correspondence between reference and predicted joints according to the spatial and functional identities of their moving parts.
>
> - Each reference joint may match at most one predicted joint, and each predicted joint may match at most one reference joint.
> - Match joints by the functional and spatial roles of their moving components, rather than by texture or exact geometric appearance.
> - Do not match two joints merely because they have the same motion type.
> - Leave a reference joint unmatched if no predicted joint controls the corresponding part.
> - Leave a predicted joint unmatched if it represents an additional or unrelated articulation.
> - Do not combine several predicted joints into one match or assign one predicted joint to multiple reference joints.
>
> **Task 2: Per-joint scoring.** For every matched joint pair, assign an integer score from 0 to 4 for each of the following criteria.
>
> 1. **Moving-part correspondence.**

- 4: The same functional rigid part or rigid group moves in both videos.
- 3: The correct part moves, with a minor difference in the included geometry.
- 2: The moving regions partially correspond, but the prediction is noticeably over-segmented, under-segmented, or incomplete.
- 1: Only a small portion corresponds, or a substantially different part moves.
- 0: The moving parts are unrelated, or the predicted part does not move.

2. **Motion type.**
   - 4: The motion type is fully consistent, such as revolute-to-revolute or prismatic-to-prismatic.
   - 3: The dominant motion type is correct, with a small secondary deviation.
   - 2: The motion is partially consistent but contains a substantial unintended component.
   - 1: The motion weakly resembles the reference but follows a fundamentally different mechanism.
   - 0: The motion type is incorrect or no valid motion is observed.
3. **Joint axis and pivot.**
   - 4: The translation axis, or the rotation axis and pivot, closely matches the reference.
   - 3: A small axis or pivot displacement is visible, but the intended articulation remains clear.
   - 2: The axis or pivot has a noticeable error that changes the motion trajectory.
   - 1: The axis or pivot is severely misplaced, although limited related motion remains.
   - 0: The axis or pivot is incompatible with the reference motion.

   Evaluate the observed trajectory rather than the numerical sign convention of the joint axis. Opposite axis signs should not be penalized when they describe the same physical motion.
4. **Motion range and direction.**
   - 4: The swept range and physical trajectory closely match the reference.
   - 3: The range is slightly smaller or larger but remains functionally consistent.
   - 2: The range differs substantially while retaining the intended motion.
   - 1: Only a small portion of the intended range is recovered, or the trajectory is largely incorrect.
   - 0: The motion range is invalid, unrelated, or not observable.

   Do not penalize a reversed parameterization when the joint traverses the same set of physical configurations.
5. **Physical stability.**
   - 4: The moving part remains rigid, connected, and stable throughout the motion.
   - 3: Minor jitter, penetration, or contact artifacts are visible but do not affect the articulation.
   - 2: Noticeable instability or penetration occurs, although the intended motion remains recognizable.
   - 1: Severe instability, detachment, or unintended coupled motion substantially disrupts the articulation.
   - 0: The asset fails, diverges, explodes, or does not produce an interpretable articulation.

Evaluation restrictions.

- Evaluate only the articulated motion shown in the videos.
- Ignore texture, material, color, lighting, background, rendering quality, and camera orientation.
- Ignore static geometric differences that do not affect identification of the moving part or its trajectory.
- Do not reward a prediction for containing more joints.
- Do not infer or mention the identity of the generating method.
- Do not infer numerical joint parameters that are not visually observable.
- Do not directly evaluate mass, damping, or stiffness. Under physical stability, consider only visible failures during the prescribed actuation.
- Use integer criterion scores only. Do not calculate a weighted or aggregate score.

Output. Return valid JSON only, using the following schema:

```
{
  "reference_joint_count": 0,
  "predicted_joint_count": 0,
  "matches": [
    {
      "reference_joint": "R1",
      "predicted_joint": "P1",
      "moving_part": 0,
      "motion_type": 0,
      "axis_and_pivot": 0,
      "range_and_direction": 0,
      "physical_stability": 0,
      "reason": "Brief evidence-based justification."
    }
  ],
  "unmatched_reference_joints": [],
  "unmatched_predicted_joints": []
}
```

Before returning the result, verify that every joint identifier appears exactly once, either in a matched pair or an unmatched list; all matches are one-to-one; and every criterion score is an integer in $\{0, 1, 2, 3, 4\}$. Do not include a weighted or aggregate score.

H.2 More Experiments About Our Representation

Direct hierarchy evaluation. We directly evaluate each level of the generated component-link-joint hierarchy. Generated and reference components are aligned using Hungarian matching based on geometric similarity, with matches exceeding a fixed distance threshold rejected. Let N_{pred}, N_{ref}, and M denote the numbers of predicted components, reference components, and accepted component matches. Component precision, recall, and F1 are defined as

$$P_{\text{comp}} = \frac{M}{N_{\text{pred}}}, \qquad R_{\text{comp}} = \frac{M}{N_{\text{ref}}}, \qquad F1_{\text{comp}} = \frac{2P_{\text{comp}}R_{\text{comp}}}{P_{\text{comp}} + R_{\text{comp}}}. \tag{77}$$

Grouping ARI evaluates the component-to-link assignments over the accepted component matches. Let n_{ij} be the number of matched components assigned to reference link i and predicted link j, with row sums a_i, column sums b_j, and total count n. We compute

$$\text{ARI} = \frac{\displaystyle\sum_{ij}\binom{n_{ij}}{2} - \frac{\displaystyle\sum_i \binom{a_i}{2}\sum_j \binom{b_j}{2}}{\binom{n}{2}}}{\displaystyle\frac{1}{2}\left[\sum_i \binom{a_i}{2} + \sum_j \binom{b_j}{2}\right] - \frac{\displaystyle\sum_i \binom{a_i}{2}\sum_j \binom{b_j}{2}}{\binom{n}{2}}}. \tag{78}$$

After component matching, generated and reference links are aligned by Hungarian matching using the intersection-over-union of their associated matched-component sets:

$$S_{\text{link}}(i, j) = \frac{\left|\mathcal{C}_i^{\text{ref}} \cap \mathcal{C}_j^{\text{pred}}\right|}{\left|\mathcal{C}_i^{\text{ref}} \cup \mathcal{C}_j^{\text{pred}}\right|}. \tag{79}$$

A tree-edge true positive is a predicted directed parent-child connection whose two endpoint links correspond to a reference edge. A joint true positive additionally requires the predicted joint type to match the reference type. For either tree edges or joints, precision, recall, and F1 are computed as

$$P_x = \frac{TP_x}{TP_x + FP_x}, \qquad R_x = \frac{TP_x}{TP_x + FN_x}, \qquad F1_x = \frac{2P_xR_x}{P_x + R_x}, \qquad x \in \{\text{edge}, \text{joint}\}. \tag{80}$$

Table 5: Direct evaluation of the generated component-link-joint hierarchy on the PartNet-Mobility test set. The best and second-best results are shown in **bold** and underlined, respectively.

Method	Component F1 ↑	Grouping ARI ↑	Joint F1 ↑	Tree-Edge F1 ↑
Fixed-joint assembly	0.76	0.00	0.18	0.37
Spatial post-hoc grouping	0.75	0.35	0.39	0.53
Ours w/o link-local DFS	0.62	0.53	0.64	0.73
Ours w/o family-balanced loss	0.74	0.43	0.52	0.63
Ours w/o part-aware co-training	0.46	0.29	0.56	0.66
Ours	**0.80**	**0.75**	**0.84**	**0.90**

```
#asset
create_asset(name=pepper_grinder,placement=[OnTable,OnObject])

#bboxs
define_bbox(bbox=[["-6.00e-05","-
0.0806","0.00","0.0548","0.0467","0.0548"],["0.00","0.0153","0.00","0.0593","0.145","0.0594"],["
0.00","0.0959","0.00","0.0529","0.0161","0.0529"]])

#links
create_mesh(code=<vq48781><vq42246><vq46029><vq03116><vq26423><vq51131><vq39626><vq
06512><vq58195><vq52503><vq11625><vq26869><vq63303><vq26250><vq61775><vq26753>)

create_mesh(code=<vq29735><vq31427><vq54822><vq29716><vq02497><vq16502><vq26508><v
q04128><vq06241><vq30390><vq15410><vq41867><vq47917><vq16617><vq31801><vq43147>)
create_link(link=link0_1,material=[wood,metal],mass=0.220)

create_mesh(code=<vq49249><vq42246><vq08344><vq38193><vq21080><vq48897><vq29942><v
q30978><vq32136><vq12981><vq02821><vq12001><vq04817><vq59225><vq50372><vq20143>)
create_link(link=lid_1,material=[metal,plastic],mass=0.0230)

#joints
create_joint(link_0=link0_1,link_1=lid_1,type=revolute,axis=[0.00,1.00,2.22e-16],origin_shift=[0.00,-
8.06e-03,2.52e-17],lower=-3.14,upper=3.14,force=1.38,time=2.84)
```

Figure 10: One sample of our training data.

All metrics are computed independently for each asset and macro-averaged over the PartNet-Mobility test set.

I More Generated Results

Here, we provide more image-conditioned simulation-ready 3D asset generation results in Figure 13 14 15.

J Representation Sample

We provide one sample of our training data in Figure 10.

K Discussion About Generation Speed

Following MegaParts (Liao et al., 2026), our current implementation uses Hugging Face Transformers for autoregressive decoding and extracts each component sequentially at a marching-cubes resolution of 1024^3.

The generated sequence contains component geometry together with link, joint, and physical-attribute records. Although simulation-ready generation introduces additional structural tokens, these records are short relative to the component geometry codes. Consequently, inference time

is governed primarily by the total sequence length and geometric complexity rather than by the number of links or joints alone. Part count is also an imperfect indicator of latency: an object may contain many geometrically simple components while requiring fewer tokens than an object with fewer but more detailed components.

A shorter autoregressive sequence does not necessarily imply faster end-to-end generation. Several competing pipelines first predict a compact structural description and then invoke a separate TRELLIS Stage-2 diffusion model (Xiang et al., 2025) to synthesize geometry. The iterative diffusion process introduces substantial additional latency, particularly when geometry must be generated for multiple components. Our model instead generates component geometry codes directly within the structured sequence and therefore does not require an additional diffusion stage for geometry synthesis. On matched complex assets, we observe lower end-to-end generation time than TRELLIS Stage-2-based pipelines despite generating longer autoregressive sequences.

Autoregressive decoding remains the primary bottleneck of our current implementation. Because the generator follows a standard language-model architecture, it is directly compatible with optimized serving systems such as vLLM, as well as model quantization, speculative decoding, multi-token prediction, and distributed inference. Geometry extraction can be accelerated independently by batching component decoding, using adaptive marching-cubes resolutions, and overlapping mesh extraction with autoregressive generation.

Peak memory is determined primarily by sequence length through the autoregressive key–value cache. It therefore does not increase monotonically with either component or link count. These engineering optimizations are orthogonal to our representation and provide clear opportunities for substantially reducing inference cost.

K.1 Scope and Editability of Physical Supervision

The physical attributes produced by Mobility-PhyX should be interpreted as conditional behavioral priors rather than measurements of a particular real-world instance. Everyday articulated objects are engineered for specific functions and human interaction: their mass, operating force, resistance, and motion speed are constrained by geometry, material, ergonomics, and safety requirements rather than varying independently at random. Consequently, object semantics and dimensions provide informative priors over physically plausible operating regimes.

Mobility-PhyX uses these priors to estimate coarse ranges of mass, interaction force, and traversal time, which are converted into effective joint parameters through simplified physical models. The predicted damping and stiffness are intended to yield plausible resistance and motion under typical operating conditions; they should not be interpreted as recovering the underlying mechanism itself. For example, the effective resistance of a drawer may arise from friction, detents, soft-close hardware, gravity, or configuration-dependent effects that cannot be represented exactly by a linear spring and viscous damper. Nevertheless, the resulting parameters provide a structured, object-dependent initialization that is more informative than applying identical simulator defaults or independently sampling unrelated values.

Importantly, our model generates an explicit code-like asset description rather than an opaque physical representation. Link and joint attributes—including mass, damping, stiffness, limits, and actuation parameters—can be inspected and modified before export to URDF or USD. Users can therefore override individual predictions when measurements are available or when an application requires particular interaction behavior. Our goal is not to replace system identification for a specific physical instance, but to provide reasonable and editable physical hypotheses that cover common operating conditions and enable simulation at scale.

For applications requiring instance-level physical fidelity, measured mechanisms or professionally calibrated assets remain necessary. A more complete treatment of nonlinear friction, detents, soft-close mechanisms, and configuration-dependent resistance is an important direction for future work.

L Simulation-Ready Asset Specification and Export

We target NVIDIA Isaac Sim as the execution environment for generated assets. Accordingly, our primary export is an articulated USD asset containing PhysX-compatible rigid-body, collision, joint,

and drive attributes. We additionally support URDF as an interchange format through our conversion script, but all simulation validation, trajectory collection, and downstream experiments are conducted using the corresponding USD assets in Isaac Sim.

Our autoregressive model generates a structured asset program containing component geometry, rigid-link records, kinematic structure, and physical attributes. The physical representation distinguishes between semantically interpretable behavioral descriptors and the coefficients required by Isaac Sim. In particular, the model generates whether a joint exhibits restoring behavior, the force required to actuate or hold it, its traversal time, and, for restoring joints, its damping ratio. A deterministic physical-resolution stage converts these descriptors into the joint stiffness and damping values used by the Isaac Sim physics engine according to the formulations in Section E.2.2.

This factorization is well suited to an autoregressive model initialized from a pretrained language model. Quantities such as "the force required to open a drawer," "the time required to traverse its range," and "whether a button returns after release" are closely related to functional knowledge expressed in natural language. They are more interpretable and better aligned with language-based priors than raw simulator coefficients, whose appropriate values depend jointly on geometry, motion range, lever arms, and moving mass. Because the generated records are code-like, users can also inspect or edit these behavioral descriptors before they are converted into Isaac Sim parameters.

The conversion consists of two deterministic stages. First, a physical resolver maps the generated behavioral descriptors and geometry-dependent quantities to simulator-facing parameters. Second, an Isaac Sim exporter performs coordinate and unit conversion and writes the resolved asset as an articulated USD stage. Neither stage invokes another learned model or re-infers the geometry, link organization, or kinematic topology. All physical quantities are converted to SI units before export.

Link specification. Each rigid link contains visual and collision geometry at metric scale, together with its material, mass, and contact attributes. The generated visual and collision meshes are represented as USD mesh prims, while rigid-body properties are written using the corresponding USD Physics and PhysX schemas.

The autoregressive model generates the material and mass associated with each link. During construction of the Mobility-PhyX training data, material, density, and effective occupied volume are used to obtain physically plausible mass supervision, as described in Section E.2.1. At inference time, however, the generated mass is written explicitly to the Isaac Sim asset and is not recomputed from density during simulation.

The center of mass and inertia tensor are deterministically computed from the collision geometry and generated mass. These quantities are therefore geometry-conditioned completions of the link record rather than independently generated attributes. The root link is assigned the articulation-root specification required for the complete link–joint hierarchy to be instantiated as an Isaac Sim articulation.

Generated joint specification. For each joint, the autoregressive model directly generates its parent and child links, joint type, parent and child frames, motion axis, and motion limits. It additionally generates the behavioral descriptors

$$\boldsymbol{\eta}_j = \left(r_j, F_j, F_j^{\mathrm{stat}}, T_j, \zeta_j\right), \tag{81}$$

where r_j indicates whether the joint has restoring behavior, F_j is the interaction force for a non-restoring joint, F_j^{stat} is the static holding force for a restoring joint, T_j is the traversal time of a non-restoring joint, and ζ_j is the damping ratio of a restoring joint. Only the subset relevant to the predicted behavior is instantiated: traversal time is not required for restoring joints, whereas damping ratio is not required for non-restoring joints.

These descriptors are represented as explicit structured fields instead of being embedded implicitly in an undifferentiated latent code. A user can therefore modify quantities such as the desired interaction force, traversal time, or restoring behavior before the final Isaac Sim drive parameters are resolved.

Deterministic physical resolution. The stiffness $\widehat{K}_j$ and damping $\widehat{D}_j$ used by Isaac Sim are deterministically resolved from the generated descriptors, motion range, link geometry, and generated

Table 6: Isaac Sim asset specification. Generated quantities are emitted by the autoregressive model, whereas derived quantities are deterministically resolved before USD serialization.

Level	Attribute	Source	SI unit	Isaac Sim serialization and role
Link	Visual and collision geometry	Generated	m	USD mesh prims and metric transforms
Link	Material and contact properties	Generated or material-resolved	Dimensionless	USD/PhysX physics-material attributes
Link	Mass	Generated	kg	Explicit rigid-body mass
Link	Center of mass and inertia	Derived	m, $\mathrm{kg\,m^2}$	USD mass and inertia attributes
Joint	Topology, type, frames, axis, and limits	Generated	m or rad	USD fixed, revolute, or prismatic joint
Joint	Restoring behavior	Generated intermediate	Binary	Selects the physical-resolution formulation
Joint	Interaction or holding force	Generated intermediate	N	Determines resistance or restoring strength
Joint	Traversal time	Generated intermediate	s	Determines non-restoring damping
Joint	Damping ratio	Generated intermediate	Dimensionless	Determines the restoring-joint damping regime
Joint	Damping	Derived	$\mathrm{N\,s/m}$ or $\mathrm{N\,m\,s/rad}$	Linear or angular drive damping
Joint	Stiffness	Derived	$\mathrm{N/m}$ or $\mathrm{N\,m/rad}$	Linear or angular drive stiffness
Joint	Actuation effort	Separate generated field	N or $\mathrm{N\,m}$	Maximum drive force or torque

child-link mass:

$$\left(\widehat{K}_j, \widehat{D}_j\right) = \mathcal{R}\left(\boldsymbol{\eta}_j; \Delta x_j, \Delta\theta_j, \ell_j^F, \ell_j^I, \widehat{m}_j^{\mathrm{child}}\right), \tag{82}$$

where $\mathcal{R}$ is the deterministic physical resolver. The displacement Δx_j or rotation $\Delta\theta_j$ is obtained from the generated joint limits, the force and inertia lever arms ℓ_j^F and ℓ_j^I are computed from the generated geometry, and $\widehat{m}_j^{\mathrm{child}}$ is the generated mass of the child link.

For a non-restoring joint, stiffness is set to zero and damping is derived from the generated interaction force and traversal time. Prismatic and revolute joints follow Equations equation 61 and equation 62, respectively. For a restoring joint, stiffness is derived from the generated static holding force, while damping is determined from the generated damping ratio, resolved stiffness, and child-link mass according to Equations equation 63 and equation 65. The model thus predicts the intended physical behavior, while the resolver ensures that the resulting Isaac Sim parameters are dimensionally and geometrically consistent.

Isaac Sim serialization. The resolved asset is written directly as an articulated USD asset. Link geometry, rigid-body mass, center of mass, inertia, collision properties, and physics materials are assigned through the corresponding USD Physics and PhysX schemas. Generated revolute, prismatic, and fixed joints are instantiated using their corresponding USD joint types, with explicit parent and child bodies, local frames, axes, and limits.

The resolved damping and stiffness values are written to the linear or angular joint drive through the USD Physics Drive API. For non-restoring joints, zero stiffness and the resolved damping represent resistance to motion. For restoring joints, the resolved stiffness and damping define the response around the generated rest configuration. Optional actuator effort limits are written separately as maximum drive forces or torques. The interaction force F_j used by the physical resolver is therefore not treated as an actuator effort limit.

For URDF export, properties supported by the standard specification, including link inertial properties, joint topology, axes, limits, and damping, are written directly. Because base URDF does not fully represent the Isaac Sim drive specification, our companion conversion script restores stiffness and other drive attributes when converting the URDF representation into the executable USD asset. The intermediate behavioral descriptors may also be retained as custom USD metadata, allowing the physical coefficients to be recomputed after user modification.

Table 6 summarizes the generated, derived, and serialized quantities.

Executability validation. Every exported asset is loaded as a USD articulation in Isaac Sim. We verify that all mesh references are valid, the link–joint graph forms a rooted acyclic structure, the articulation root is correctly assigned, and all transformations and physical parameters are finite. Each movable joint is then actuated across its prescribed range, and the asset is simulated with PhysX for a fixed duration to detect invalid configurations, numerical instability, unintended divergence, or collisions that prevent the intended motion. An asset is considered simulation-ready only if it passes these checks in Isaac Sim without manual correction.

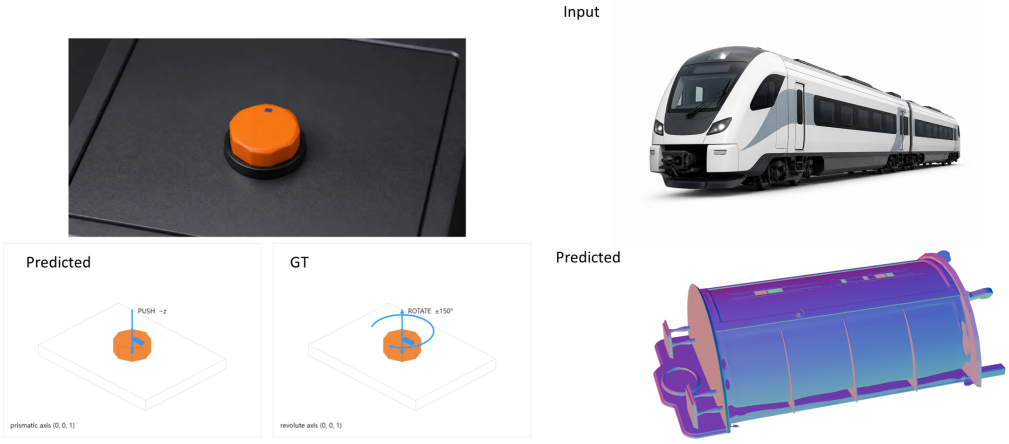

Figure 11: Representative failure cases. **Left**: A very confusing situation when revolute joint and prismatic joint are both correct. **Right**: When cases which are far from our training dataset domain are given, our model fails to generate both correct geometry and correct kinematic structure.

Scope. In this work, simulation readiness denotes structural completeness and direct executability in Isaac Sim. It does not imply that the generated force, traversal time, damping ratio, mass, damping, or stiffness corresponds to measurements of a particular physical instance. These values should instead be interpreted as physically plausible and editable priors covering common interaction settings. When instance-level measurements or task-specific requirements are available, users can modify the intermediate descriptors or directly replace the resolved Isaac Sim coefficients before export.

M Failure Cases

Our failure cases primarily fall into two categories. First, visually similar components can exhibit fundamentally different motion types. A button-like structure, for example, may correspond to either a prismatic push button or a revolute rotary control. When this functionality is not evident from the input image or text description, the model may generate a geometrically plausible component but assign an incorrect joint type. Such cases reflect an inherent functional ambiguity in the conditioning signal rather than geometry alone.

Second, although part-aware co-training substantially improves geometric generalization, inputs far outside the training distribution can still lead to catastrophic failures. For image-conditioned generation, this may occur when the observed object differs significantly from the shapes and structures represented in the training data. Similarly, text prompts describing rare categories or unusual structural compositions may produce malformed geometry or inconsistent articulation. This limitation affects both image- and text-conditioned generation, indicating that co-training broadens the learned distribution but does not fully resolve extreme out-of-distribution generalization. Representative examples of both failure categories are shown in Figure 11.

N Discussion: Autoregressive vs. Diffusion-Based Generation of Simulation-Ready 3D Assets

Autoregressive and diffusion-based approaches offer complementary advantages for 3D generation. In common latent-diffusion formulations, geometry is represented using continuous autoencoder latents, whereas discrete autoregressive models typically rely on a VQ-VAE. Continuous latents can retain comparable geometric information using fewer latent elements because each element carries continuous-valued information, while vector quantization introduces an additional compression

bottleneck. Autoregressive models, however, factorize the joint distribution into a sequence of conditional predictions,

$$p(\mathbf{z} \mid \mathbf{c}) = \prod_{t=1}^{T} p(z_t \mid z_{<t}, \mathbf{c}), \tag{83}$$

where the sequence length T can vary across objects. This provides an adaptive representational budget: simple objects can be described using short sequences, whereas geometrically complex objects can allocate additional tokens to their components. Most diffusion pipelines instead determine the size of the latent state before denoising. Although variable-size diffusion formulations exist, dynamically introducing additional latent elements during generation remains uncommon. Consequently, autoregressive factorization provides a practical advantage when scaling generation along the token-count axis, since it does not require every object to occupy the same large latent space.

This flexibility comes with important costs. Autoregressive decoding is sequential, and its computational cost grows with sequence length. Moreover, geometry tokens are commonly optimized using cross-entropy rather than a distance defined directly in the continuous latent or geometric space. The categorical objective does not distinguish between predictions corresponding to nearby and distant VQ-VAE embeddings, potentially making fine geometric distributions less sample-efficient to learn. In our experiments, this limitation is reflected by the substantial degradation in geometry quality when part-aware co-training is removed, as shown in Table 4. Large-scale geometric supervision is therefore particularly important for learning a reliable distribution over discrete shape tokens.

Simulation-ready generation nevertheless strongly favors structured autoregressive modeling. In addition to geometry, a simulation-ready asset contains discrete and relational records, including component-to-link assignments, parent–child relationships, joint types, motion axes, limits, and physical parameters. These attributes are naturally expressed as structured code rather than compressed into an undifferentiated continuous latent. Code-like generation preserves explicit references and hierarchy, supports syntactic validation, and allows users to inspect or modify individual attributes before export. Articraft (Zhou et al., 2026), for example, demonstrates that even a training-free code-generation system can construct complex articulated structures, although generating detailed and diverse geometry remains more challenging. Our model combines this strength of structured generation with learned component-level geometry.

Serialization order is both an advantage and a limitation of autoregressive generation. A connection-aware DFS provides an explicit construction trace in which coarse structures are established before their finer components. This ordering supplies useful context for subsequent predictions and makes small functional parts easier to model. However, an articulated object may admit multiple valid serializations, and an unstable or poorly chosen order can produce inconsistent conditioning contexts. Since early errors propagate to later predictions, ordering mistakes may result in missing components or incorrect kinematic structures. The degradation observed without DFS ordering in Table 4, particularly for small parts, illustrates this sensitivity. Diffusion models are less dependent on a single causal order and can iteratively revise multiple elements, but enforcing exact hierarchical references and executable structure within such formulations is less straightforward.

Autoregressive models also provide a convenient interface for co-training with language, code, and heterogeneous 3D supervision. Their shared token-based formulation makes it possible to transfer semantic and structural knowledge from pretrained language models while training only the available portions of each asset representation. Considering this compatibility together with variable-length generation and explicit structural outputs, we find autoregressive modeling to be a particularly suitable foundation for simulation-ready asset generation at present, although it is not universally superior to diffusion.

Hybrid architectures provide a promising direction for combining their strengths. Transfusion-style models (Zhou et al., 2025) could retain continuous diffusion objectives for component geometry while generating topology and physical attributes autoregressively, thereby reducing the information loss introduced by vector quantization. Conversely, diffusion language models (Nie et al., 2025) could enable bidirectional refinement of discrete asset programs and reduce dependence on a single left-to-right serialization. Exploring these hybrid formulations may alleviate both the VQ-VAE bottleneck and ordering sensitivity while retaining the editability and structural precision required by simulation-ready assets.

O SIM-TO-REAL POLICY-LEARNING DETAILS

O.1 SIMULATION TRAJECTORY COLLECTION

We export generated assets using the conversion pipeline described in Section 5.1. Their meshes, link organization, joints, and physical attributes, including joint damping and stiffness, are instantiated directly in the simulator without manual reconstruction or asset-specific dynamic tuning. For each manipulation task, a task template defines the target object, manipulated link or joint, initial-state distribution, language instruction, time horizon, and success condition.

Importance of joint dynamics for scalable data collection. Appropriate joint dynamics are essential for automated trajectory collection. Isaac Sim's URDF importer initializes joint-drive stiffness and damping to zero by default (NVIDIA, 2026), leaving an unauthored joint without active resistance or restoring behavior. Consequently, articulated components may respond unrealistically to gravity or minor contact. A laptop screen may fall shut under its own weight, while a door or drawer may swing or slide freely after a light touch and immediately reach its motion limit. These behaviors differ from real mechanisms, whose motion is commonly moderated by hinge friction, sliding resistance, detents, or soft-closing mechanisms, and can cause the collected trajectory to reflect passive simulator dynamics rather than deliberate robot manipulation. Assigning uniformly large values creates the opposite problem: excessive damping or stiffness can keep a component nearly fixed even after firm contact, preventing the robot from completing the intended action. Both extremes substantially reduce the success of procedural trajectory generation and require asset-specific manual tuning. Our model instead generates interpretable behavioral descriptors that are converted into asset-specific damping and stiffness as described in Section E.2.2. For representative tasks that fail consistently under default joint dynamics, these generated parameters increase the trajectory-collection success rate—the fraction of attempted rollouts that satisfy the task criterion—from 0% to nearly 100%, enabling scalable data collection across diverse articulated assets.

Following the compositional data-collection paradigm of InternData-A1 (Tian et al., 2026), we separate asset instantiation, scene construction, task specification, and expert trajectory generation. A privileged simulation policy uses object and robot states to generate collision-aware manipulation trajectories. Each candidate trajectory is replayed in the simulator and retained only when it satisfies the task-specific success condition. For every successful episode, we record the language instruction, multi-view RGB observations, robot proprioceptive state, and executed actions.

Each task is instantiated using multiple assets sampled from our generative model. We vary the initial object configuration, robot pose, camera parameters, illumination, and scene layout, exposing the policy to variations in geometry, articulation, appearance, and physical behavior while preserving a consistent task definition. We denote the resulting trajectory collection by $\mathcal{D}_{\mathrm{gen}}$.

O.2 POLICY TRAINING

We fine-tune the released $\pi_{0.5}$ base policy (Physical Intelligence et al., 2025) on $\mathcal{D}_{\mathrm{gen}}$. At each time step, the policy receives a language instruction, multi-view RGB observations, and robot proprioception, and predicts a continuous action chunk through its flow-matching action head. The three tasks are trained jointly using task-specific language instructions.

Training uses exclusively simulated trajectories collected with generated assets. No trajectories from the physical evaluation objects are included, so real-world performance measures direct sim-to-real transfer rather than adaptation to the test objects. We keep the policy architecture and optimization settings fixed across asset and data ablations so that performance differences reflect the utility of the generated training environments.

O.3 SIMULATION EVALUATION

We first evaluate the policy in held-out simulation environments whose asset instances, initial configurations, and scene layouts are excluded from policy training. We additionally evaluate on established simulation-ready benchmark assets to measure generalization beyond assets sampled from our generator.

A rollout is considered successful if its task-specific final-state predicate is satisfied at the end of the prescribed horizon without violating the safety constraints. For an evaluation set containing R rollouts, the success rate is

$$\mathrm{SR} = \frac{1}{R}\sum_{r=1}^{R} \mathbb{1}\left[g\left(\mathbf{s}_T^{(r)}\right) = 1\right], \tag{84}$$

where g denotes the task-specific success predicate and $\mathbf{s}_T^{(r)}$ is the final simulator state of rollout r. We report the per-task success rate and its unweighted macro-average across tasks.

O.4 Real-World Evaluation

We deploy the same policy checkpoint on physical objects corresponding to the simulated task categories. The robot embodiment, observation format, action space, and control frequency are kept consistent between simulation and reality, while the physical objects differ from the generated assets in geometry, texture, articulation, and physical properties. No object-specific policy adaptation is performed.

For each task, we conduct the same number of trials under predefined initial configurations and report the per-task real-world success rate. Figure 5 reports all results as percentages.

O.5 Hardware Setup

Our real-world platform consists of a Franka robotic arm equipped with a Robotiq 85 gripper. The corresponding robot and gripper models are used for simulation trajectory collection and evaluation. The policy receives multi-view RGB observations and robot proprioceptive states. We maintain consistent observation formats, action spaces, and control frequencies across simulated and physical execution.

We also further provide image of our simulation environment and our real experiment environment in Figure 12.

P Discussion of Recent and Concurrent Works

Several recent and concurrent preprints explore closely related directions in articulated and simulation-ready 3D asset generation. We discuss their relationship to our work for completeness and to clarify the distinct scope of our formulation.

We do not directly compare with these methods because they are either not open-sourced or have not yet been accepted at a peer-reviewed venue.

End-to-end articulated asset generation. URDF-Anything+ (Wu et al., 2026) introduces an autoregressive diffusion framework that generates executable URDF models from an RGB image and object-level 3D geometry. It sequentially predicts link geometry together with the associated joint parameters, thereby unifying geometry reconstruction and articulation estimation in an end-to-end model. Our work shares the goal of directly generating executable articulated assets, but differs in its geometric and kinematic organization. Rather than using the generated geometric parts directly as rigid links, we explicitly separate fine-grained geometric components from kinematic links. Multiple detailed components may therefore be grouped into a single rigid body, preserving small geometric structures without introducing redundant links or fixed joints. Our representation additionally generates material, mass, and simulator-facing joint-drive attributes and supports both image- and text-conditioned generation.

Unified physical 3D generation. PhysX-Omni (Cao et al., 2026b) develops a unified framework for generating rigid, deformable, and articulated physical assets. Its primary objective is breadth across asset types, supported by the PhysXVerse dataset and PhysX-Bench evaluation framework. In contrast, our work focuses specifically on the internal complexity of articulated objects. We study how highly detailed component geometry can be reconciled with a substantially more compact rigid-link structure and how this hierarchy can be modeled within a long-context autoregressive sequence. The two approaches therefore emphasize complementary scaling dimensions: PhysX-Omni scales

Table 7: Conceptual positioning relative to recent and concurrent work on simulation-ready asset generation. The comparison concerns modeling objectives and representations rather than empirical performance.

Method	Main formulation	Relation to our formulation
URDF-Anything+	Generates link geometry and associated joint parameters from visual and geometric inputs.	Directly models links and joints, whereas our representation introduces fine geometric components that are subsequently grouped into rigid links.
PhysX-Omni	Provides unified generation of rigid, deformable, and articulated assets with physical and semantic attributes.	Emphasizes breadth across asset types; our work focuses on structural complexity within articulated assets and decouples geometric components from kinematic links.
PhysX-CoT	Uses explicit physical-reasoning states and factorized geometric representations for verifiable image-to-asset generation.	Emphasizes supervised reasoning and verification; our method autoregressively generates the final component-link-joint program under heterogeneous supervision.
JODA	Predicts joint-dynamics fields, including conservative forces, dry friction, and damping, for an existing articulated asset.	Addresses downstream dynamics refinement of a given asset; our method jointly generates the articulated structure and simulator-facing scalar drive parameters.
Ours	Generates fine components, groups them into rigid links, and connects the links through joints with mass and scalar drive parameters.	Targets detailed articulated generation with separate geometric and kinematic granularities.

across physical asset types and categories, whereas our method scales the geometric and kinematic complexity within articulated assets.

Structured physical reasoning. PhysX-CoT (Huang et al., 2026) formulates single-image simulation-ready generation as an explicit sequence of supervised physical reasoning states, including object decomposition, 2D and 3D grounding, spatial relationships, local geometry, and surface attributes. It further optimizes these intermediate states through process-level rewards. Both PhysX-CoT and our method factorize spatial placement from local geometry. However, their structured sequences serve different purposes. PhysX-CoT exposes and supervises the intermediate image-to-asset reasoning process, whereas our sequence directly represents the final generative hierarchy of geometric components, rigid links, joints, and physical attributes. Our principal focus is the component-to-link abstraction and heterogeneous co-training with part-aware and simulation-ready assets, rather than explicit chain-of-thought supervision.

Joint-dynamics modeling. JODA (Gao et al., 2026) studies the complementary problem of assigning expressive dynamics to an existing articulated asset. It represents joint behavior using position-dependent conservative forces, dry friction, and damping fields and supports subsequent editing and optimization. Our physical representation is intentionally designed for a different operating regime. We jointly generate geometry, component grouping, kinematics, and the scalar joint-drive parameters required by Isaac Sim and PhysX. Interpretable behavioral descriptors, such as operating force, traversal time, restoring behavior, and damping ratio, are converted deterministically into simulator-facing damping and stiffness coefficients. We do not aim to recover detailed internal mechanisms or general position-dependent force fields. When more expressive joint behavior is required, a method such as JODA could be applied as a downstream refinement stage to assets generated by our framework.

Summary. These concurrent works demonstrate growing interest in generating assets that are not only visually plausible but also directly usable in physical simulation. Our contribution is orthogonal to general physical-asset unification, explicit reasoning supervision, and post-hoc expressive dynamics modeling. We introduce a bottom-up autoregressive representation in which fine-grained geometric components are generated first, organized into shared rigid links, and connected through physically parameterized joints. This separation allows geometric complexity and kinematic com-



(a) Real environment (b) Simulation environment

Figure 12: Experimental environments: (a) the real-world workspace and (b) its simulation counterpart.

plexity to scale independently while retaining an explicit, editable, and directly executable asset description.

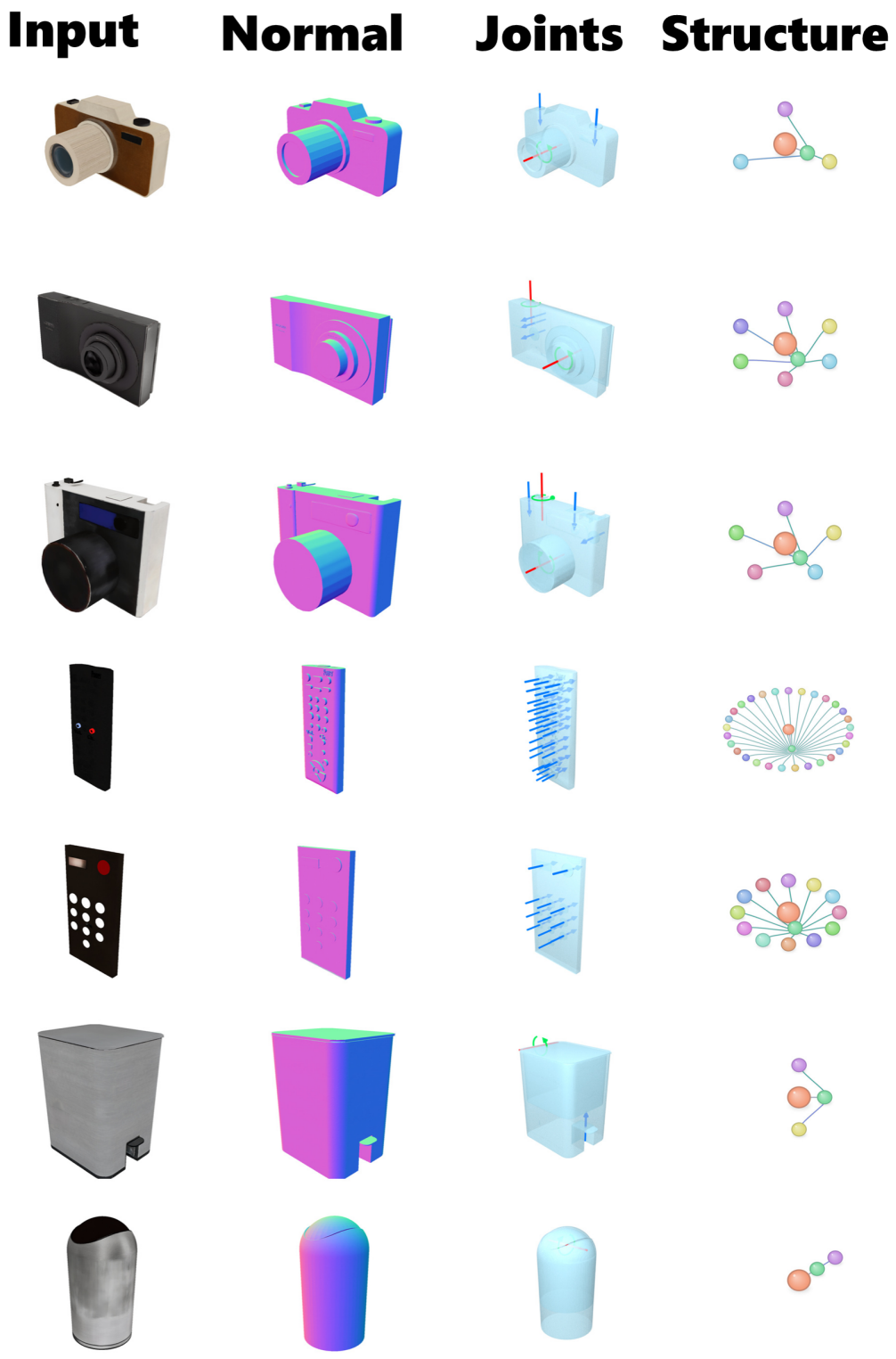

Figure 13: Generation results conditioned on daily object images.



Figure 14: Generation results of object with more complex geometry.

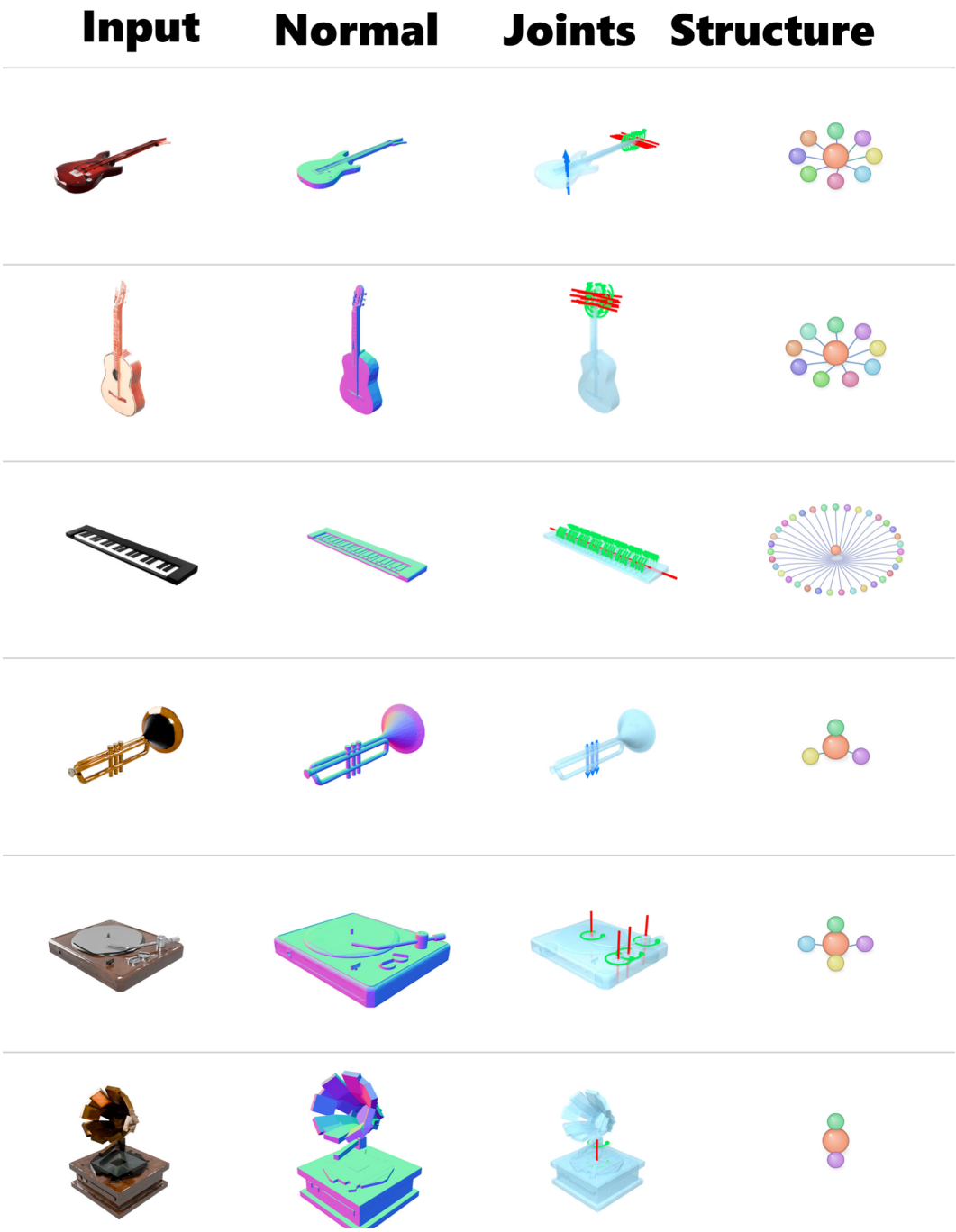

Figure 15: Generation results of rare instruments categories.