UNIPHYSGEN: UNIFIED PHYSICAL GROUNDING FOR SIMULATION-READY 3D ASSETS

Anonymous authors
Paper under double-blind review

ABSTRACT

Physically grounded 3D assets are increasingly important for embodied AI and robotic simulation. However, most existing 3D assets lack unified physical semantics, including articulation semantics and intrinsic physical properties, required for realistic interaction. Current approaches either treat these semantics independently or rely on canonicalized object structures, limiting robustness across heterogeneous 3D assets. We present **UniPhys**, a scalable framework for automatically transforming raw 3D assets into simulation-ready assets with unified physical semantics. Based on UniPhys, we construct **UniPhys-40K**, a large-scale physically grounded dataset, together with **UniPhys-Bench**, a carefully verified benchmark for unified physical grounding evaluation. We further introduce **UniPhysGen**, a unified physical grounding model that jointly reasons over articulation semantics and intrinsic physical properties. UniPhysGen incorporates geometry-robust articulation grounding to mitigate geometric shortcut bias under heterogeneous part decompositions. Extensive experiments demonstrate state-of-the-art performance across articulation grounding and intrinsic physical property estimation tasks, while the resulting assets can be directly deployed in robotic simulation environments for realistic physical interaction. Code: `https://github.com/anonymous-abc123/anonymous-code`.

1 INTRODUCTION

Recent advances in generative 3D modeling (Xiang et al., 2025; 2026; Zhao et al., 2025b; Lai et al., 2025) and the emergence of large-scale 3D repositories such as Objaverse (Deitke et al., 2023b;a) have dramatically increased the availability of high-quality 3D assets. However, despite their realistic geometry and appearance, most existing assets remain fundamentally *simulation-unaware*, lacking physically grounded semantics required for realistic interaction, simulation, and embodied AI. Such semantics include both **articulation semantics**, such as articulated structures and kinematic parameters, and **intrinsic physical properties**, including material, density, friction, and mass. Consequently, a gap remains between visually realistic 3D assets and simulation-ready assets.

A central challenge therefore lies in how to automatically transform raw 3D meshes into physically consistent and simulation-ready assets with unified physical semantics. Existing physical grounding approaches remain fragmented, primarily treating articulation grounding and intrinsic physical property estimation as independent problems (Chen et al., 2024; Zhao et al., 2025a; Le et al., 2025; Zhai et al., 2024; Xu et al., 2025). However, simulation-ready assets require these semantics to be jointly consistent during physical interaction and simulation. Grounding these semantics independently often leads to physically inconsistent motions, unstable interactions, or implausible simulation behaviors. More recent physics-grounded generation frameworks (Cao et al., 2026; 2025; Yang et al., 2026) have further explored jointly modeling articulation and intrinsic physical properties. Nevertheless, these approaches typically rely on canonicalized part decompositions, clean object structures, or structural assumptions constructed under constrained synthetic settings. Such assumptions generalize poorly to the heterogeneous and inconsistent decompositions commonly observed in diverse resources, including artist-designed meshes, procedurally generated structures, and generative 3D assets. **Meanwhile,** as summarized in Table 1, existing physically grounded datasets either focus primarily on articulation semantics (Xiang et al., 2020; Iliash et al., 2026; Geng et al., 2023; Li et al., 2026) or rely heavily on manual curation, clean pre-segmented object parts, and canonical structural decompositions (Cao et al., 2025; Yang et al., 2026). More importantly, although phys-

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.

ical grounding aims to enable simulation-ready assets, existing datasets typically stop at semantic annotation, with limited use of simulation to validate or refine the grounded results.

To address these limitations, we introduce **UniPhys**, an automated physical grounding pipeline that uniformly transforms 3D meshes from diverse sources into simulation-ready assets with unified articulation and physical property semantics. As illustrated in Fig. 1, UniPhys targets three key challenges in physical grounding, which are addressed through dedicated modules in our system. **(1) Physically meaningful structural decomposition.** Physical grounding requires part decompositions that preserve motion-relevant and material-sensitive structures. Existing methods primarily optimize geometric or semantic consistency (Liu et al., 2025; Ma et al., 2025), often overlooking physically meaningful part abstractions. We therefore introduce a perceptually guided decomposition framework for physical grounding. **(2) Geometry-aware articulation grounding.** Large multimodal models provide strong semantic priors for object functionality, but often struggle to ground these priors into geometrically valid 3D articulation parameters. We therefore develop a geometry-aware articulation grounding strategy that combines multimodal semantic understanding with geometric motion reasoning to infer physically consistent articulation configurations. **(3) Simulation-driven consistency verification.** Automatically grounded physical annotations may still contain physically implausible properties or unstable articulated behaviors. To alleviate this issue, we introduce a simulation-driven verification framework that identifies and filters inconsistent annotations through articulated motion consistency and intrinsic physical plausibility checks.

Driven by the UniPhys pipeline, we further construct **UniPhys-40K**, a large-scale physically grounded dataset containing over 40K 3D assets with unified articulation and intrinsic physical properties aggregated from heterogeneous 3D asset sources, including synthetic assets, CAD models, and artist-designed collections (Mo et al., 2019; Deitke et al., 2023b; Khanna et al., 2024; Collins et al., 2022; Fu et al., 2021). To support rigorous evaluation, we additionally introduce **UniPhys-Bench**, a high-quality benchmark containing 1,927 articulated objects with 5.5K motion-relevant components, constructed through a labor-intensive human-in-the-loop annotation process. Manual verification and correction of both articulation semantics and intrinsic physical properties are performed to ensure physically consistent annotations across heterogeneous 3D assets.

Building upon UniPhys-40K, we propose **UniPhysGen**, a unified physical grounding model designed for robust reasoning under heterogeneous part decompositions. UniPhysGen jointly models articulation semantics and intrinsic physical properties within a shared representation space without assuming canonicalized object structures. To mitigate geometry shortcut learning, UniPhysGen introduces geometry-robust articulation grounding through SO(3) augmentation and spherical articulation axis parameterization. Combined with unified global positional encoding across part-level and object-level geometries, the model achieves stable articulation grounding under diverse structural variations.

Our main contributions are summarized as follows:

- We introduce **UniPhys**, a scalable pipeline that automatically grounds raw 3D assets with articulation semantics and intrinsic physical properties.
- We construct **UniPhys-40K** and **UniPhys-Bench**, enabling large-scale training and evaluation for physical grounding.
- We propose **UniPhysGen**, a unified physical grounding model that jointly reasons over articulation semantics and intrinsic physical properties under heterogeneous part decompositions through geometry-robust articulation grounding and physical semantic alignment.
- Extensive experiments demonstrate that UniPhysGen achieves state-of-the-art performance on physical grounding across most evaluation settings while remaining robust to heterogeneous part abstractions. We further validate grounded-asset deployability in robotic simulation environments.

2 Related Work

Physical Grounding for 3D Assets. Existing physical grounding methods mainly focus on either articulation semantics (Chen et al., 2024; Qiu et al., 2025; Zhao et al., 2025a; Le et al., 2025; Li et al., 2026) or intrinsic physical properties (Zhai et al., 2024; Xu et al., 2025). Prior work on articulation semantics includes image-based URDF prediction (Chen et al., 2024), program synthesis (Zhao et al., 2025a), and geometry-grounded articulation reasoning from 3D point clouds (Li et al., 2026).

Table 1: Comparison of 3D datasets for physical grounding. **S1–S3** denote no pre-segmented parts, low human effort, and raw-asset extensibility. ✓/✗: yes/no; –: not applicable or not reported.

Dataset	Primary Data Source	#Obj.	#Cat.	Articulation Semantics	Intrinsic Physical Properties	Simulation verification	Scalability		
							S1	S2	S3
ShapeNet (Chang et al., 2015)	Online	51K	55	✗	✗	✗	–	–	–
PartNet (Mo et al., 2019)	ShapeNet	26K	24	✗	✗	✗	–	–	–
3D-FUTURE (Fu et al., 2021)	Industry	10K	8	✗	✗	✗	–	–	–
ABO (Collins et al., 2022)	Amazon	8K	63	✗	Partial	✗	–	–	–
HSSD (Khanna et al., 2024)	Floorplanner	18K	466	✗	✗	✗	–	–	–
Objaverse (Deitke et al., 2023b)	Multi.	818K	21K	✗	✗	✗	–	–	–
PartNet-Mobility (Xiang et al., 2020)	ShapeNet	2.3K	46	✓	✗	✗	–	–	–
GAPartNet (Geng et al., 2023)	PartNet	1.1K	27	✓	✗	✗	–	–	–
PhysXNet (Cao et al., 2025)	PartNet	26K	24	✓	✓	✗	✗	✗	✗
PhysDB (Yang et al., 2026)	Objaverse	150K	7	Joint type only	✓	✗	✗	✗	✗
UniPhys-40K (Ours)	**Multi.**	**40K**	**84**	✓	✓	✓	✓	✓	✓

In parallel, work on intrinsic physical properties focuses on estimating attributes such as density, friction, and mass through multimodal physical reasoning (Zhai et al., 2024; Xu et al., 2025). Recent frameworks move toward unified physical grounding by jointly modeling articulation semantics and intrinsic physical properties (Cao et al., 2026; 2025; Yang et al., 2026). However, these methods typically assume clean, canonicalized part decompositions, which rarely hold for heterogeneous assets with inconsistent granularity and structural ambiguities.

Physical Grounding Datasets. Large-scale repositories such as ShapeNet (Chang et al., 2015), ABO (Collins et al., 2022), and Objaverse (Deitke et al., 2023b;a) provide abundant geometric and visual content but largely lack physical semantics required for simulation-ready interaction. Subsequent datasets introduce articulation annotations, including PartNet-Mobility (Xiang et al., 2020) and GAPartNet (Geng et al., 2023). More recent efforts further incorporate intrinsic physical properties alongside articulation semantics (Cao et al., 2025; Yang et al., 2026). However, these datasets remain largely built on clean, pre-structured assets and often require substantial manual annotation, limiting scalability to heterogeneous 3D assets.

3 UniPhys: Scalable Unified Physical Grounding Pipeline

UniPhys aims to automatically transform heterogeneous 3D assets into simulation-ready assets with grounded articulation and intrinsic physical properties. As shown in Fig. 1, the pipeline contains four stages: physically meaningful part decomposition, intrinsic physical property grounding, geometry-aware articulation grounding, and simulation-driven consistency verification. The pipeline is detailed in Sec. 3.1. Scaling this pipeline over large 3D asset repositories yields UniPhys-40K and the manually verified UniPhys-Bench benchmark, introduced in Sec. 3.2.

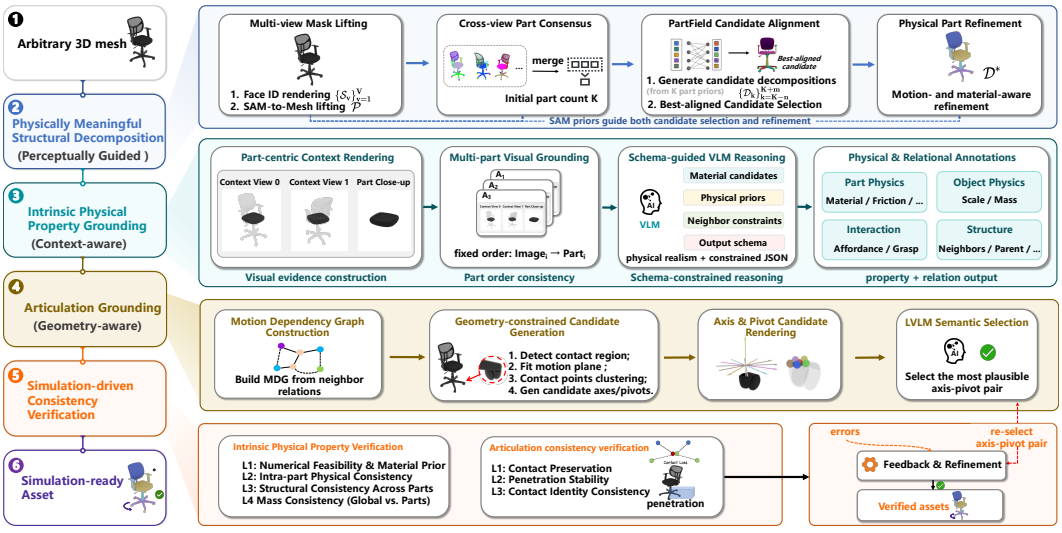

Figure 1: UniPhys: A scalable pipeline for transforming heterogeneous 3D assets into simulation-ready assets with unified articulation semantics and intrinsic physical properties.

3.1 UNIPHYS PIPELINE

Physically Meaningful Structural Decomposition. Existing decomposition methods (Ma et al., 2025; Liu et al., 2025; Yan et al., 2025) optimize geometric or semantic consistency, whereas simulation-ready grounding additionally requires articulation, material, and functional coherence. Our key insight is that perceptual grouping priors learned by segmentation foundation models often coincide with physically meaningful structures, providing effective guidance for physical decomposition. Based on this observation, UniPhys first lifts multi-view SAM (Ravi et al., 2025) segmentations into mesh space to build perceptual priors $\mathcal{P}$ and infer a part-cardinality prior K. Guided by K, PartField (Liu et al., 2025) generates candidate decompositions $\{\mathcal{D}_k\}_{k=K-n}^{K+m}$ at varying granularities, which are then aligned to $\mathcal{P}$ through Hungarian matching (Kuhn, 1955). Finally, treating components associated with each lifted SAM mask in the best-aligned decomposition as transactions, we use Apriori-based frequent itemset mining (Agrawal & Srikant, 1994) to identify stable cross-view component groupings and refine the final decomposition $\mathcal{D}^*$.

Intrinsic Physical Property Grounding. Given the physically grounded part decomposition, UniPhys infers intrinsic physical properties through part-aware multimodal reasoning. We construct context-preserving part-centric observations and organize them into a unified part atlas for schema-guided LVLM reasoning. Conditioned on object context, neighboring relations, and physical priors, the LVLM jointly predicts part- and object-level physical properties.

Geometry-aware Articulation Grounding. While LVLMs provide strong semantic priors for understanding object functionality and affordances, directly predicting articulation parameters in continuous 3D space remains geometrically unstable. UniPhys instead constructs feasible articulation hypotheses from neighboring relations and local contact geometry, reducing articulation grounding to semantic selection over geometry-constrained candidates. This combination of geometric constraints and semantic reasoning improves both robustness and physical validity.

Simulation-driven Consistency Verification. Automatically grounded annotations may still be physically inconsistent. Rather than recovering exact real-world physical quantities, UniPhys evaluates whether grounded semantics satisfy simulation-consistent interaction constraints through a simulation-driven verification stage. Invalid annotations are subsequently filtered or refined through lightweight feedback. Detailed implementations of all four stages are deferred to Appendix A.

3.2 UNIPHYS-40K AND BENCHMARK.

Dataset Overview. Driven by the proposed UniPhys pipeline, we construct UniPhys-40K, a large-scale dataset with grounded articulation semantics and intrinsic physical properties. UniPhys-40K is built from diverse 3D repositories, including Objaverse (Deitke et al., 2023b), HSSD (Khanna et al., 2024), 3D-FUTURE (Fu et al., 2021), ABO (Collins et al., 2022), and PartNet (Mo et al., 2019). To support rigorous evaluation, we further construct UniPhys-Bench, a carefully verified benchmark containing 1,927 articulated objects with 5.5K motion-relevant components spanning diverse object categories and structural complexities. As summarized in Fig. 3, UniPhys-40K covers a broad spectrum of object scales, masses, categories, materials, and articulation patterns, ranging from small household objects to large furniture and infrastructure assets. The dataset further exhibits long-tailed distributions of both physical properties and kinematic structures, covering rigid, revolute, prismatic, and contact-only interaction relationships. More dataset construction details are provided in Appendix A.5.

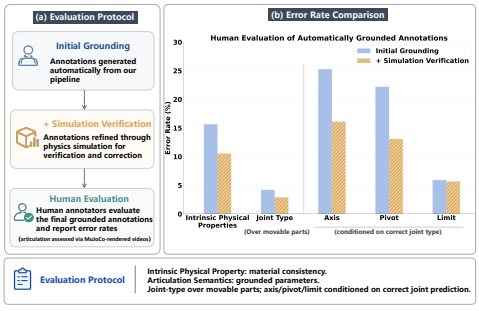

Figure 2: Human evaluation.

Quality Analysis. We further perform human evaluation on sampled assets before and after simulation-driven verification. As shown in Fig. 2, the results show that verification substantially improves the reliability and physical consistency of automatically grounded simulation-ready assets. This observation supports the effectiveness of simulation-driven consistency verification as a scalable quality-control mechanism for UniPhys.

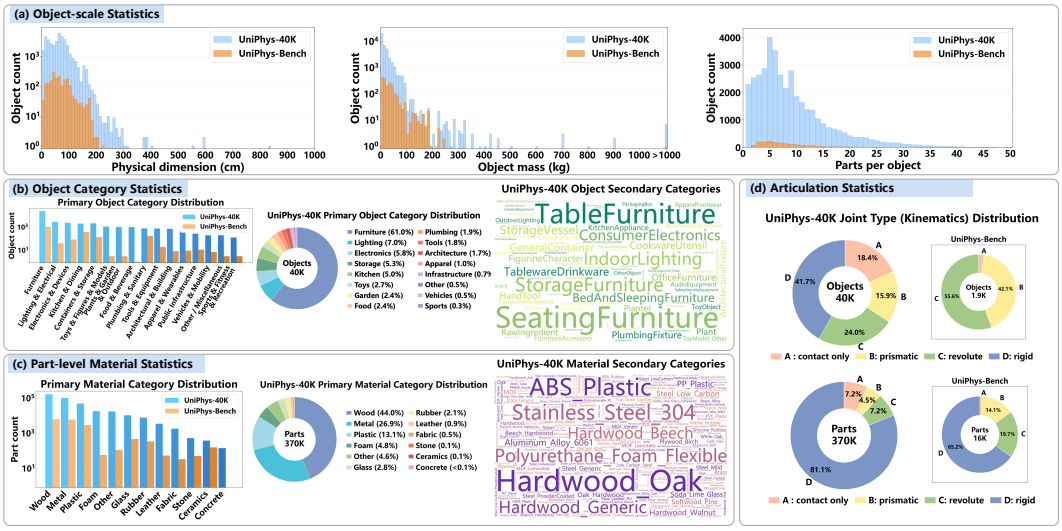

Figure 3: Statistics of UniPhys-40K and UniPhys-Bench.

4 UniPhysGen

We formalize UniPhysGen as learning a unified physical grounding framework that maps input 3D geometry to structured physical semantics. Given an object mesh and a target part mesh, we convert them into point clouds and denote the resulting input as $\mathcal{X}$. UniPhysGen predicts four categories of physical semantics:

$$\mathcal{Y} = f_\theta(\mathcal{X}), \qquad \mathcal{Y} = \{\mathcal{F}_p, \mathcal{A}, \mathcal{S}, \mathcal{F}_o\}, \tag{1}$$

where f_θ denotes a shared UniPhysGen architecture instantiated across all four tasks. Besides, $\mathcal{F}_p$, $\mathcal{A}$, $\mathcal{S}$, and $\mathcal{F}_o$ denote part-level intrinsic physical properties, articulation kinematics, articulation structure, and object-level intrinsic physical properties, respectively. Specifically, UniPhysGen adopts Qwen3 (Yang et al., 2025) as the backbone and performs physical grounding through four stages shown in Fig. 4: (1) physical semantic alignment pretraining for $\mathcal{F}_p$; (2) geometry-robust articulation grounding for $\mathcal{A}$, including joint type, axis, pivot, and motion limits, on motion-related parts obtained either from the physical semantic alignment stage or from external annotations; (3) articulation structure grounding for $\mathcal{S}$; and (4) object-level physical grounding for $\mathcal{F}_o$. Unlike prior approaches that rely on canonicalized structures or fixed decomposition templates, UniPhysGen directly operates on heterogeneous 3D asset decompositions. Additional details of the four stages are provided in Appendix B.1–B.4.

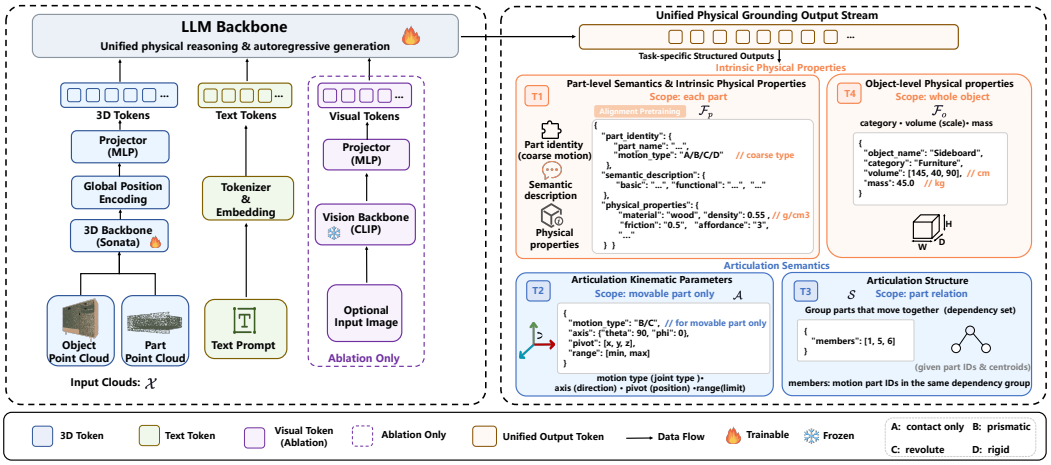

Figure 4: Overview of UniPhysGen. The model takes 3D object and part geometry as input (Left) and generates four types of structured physical semantics (Right): part-level properties, articulation kinematics, articulation structure, and object-level properties via a unified multimodal backbone.

4.1 Physical Semantic Alignment Pretraining

A key challenge of unified physical grounding is learning a shared representation space that jointly captures geometry, articulation semantics, and intrinsic physical properties. To this end, we introduce a physical semantic alignment pretraining stage that aligns local part geometry and global object context with structured physical semantic descriptions, including part identity, functional semantics, articulation behavior, and intrinsic physical properties. This representation serves as a unified grounding space shared across downstream tasks, enabling robust physical reasoning under heterogeneous part decompositions.

4.2 Geometry-Robust Articulation Grounding

Articulation grounding is highly sensitive to geometric shortcuts and rotational bias. Existing methods often assume upright object orientations (Li et al., 2026) and apply z-axis rotation augmentation, $\mathbf{p}' = \mathbf{R}_z(\theta)\mathbf{p}$, where $\mathbf{p} \in \mathbb{R}^3$ is a 3D point coordinate and $\mathbf{R}_z(\theta)$ denotes rotation around the vertical axis (Fig. 5(a)). This encourages view-dependent motion reasoning and generalizes poorly to geometrically incomplete or structurally ambiguous assets.

To mitigate this issue, we replace canonical-orientation augmentation with SO(3)-based rotation augmentation (Fig. 5(b)): $\mathbf{p}' = \mathbf{R}\mathbf{p}$, $\mathbf{R} \in SO(3)$. This significantly improves structure-aware articulation reasoning by forcing the model to rely on motion-relevant geometry semantics rather than view-dependent shortcuts.

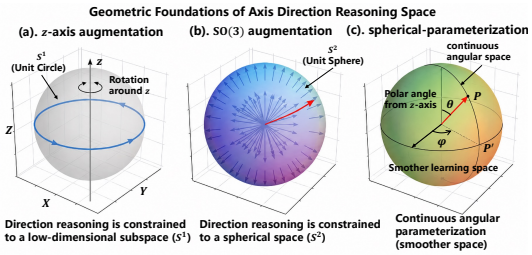

Figure 5: Axis direction reasoning space.

However, we observe that SO(3)-based augmentation substantially destabilizes articulation axis reasoning due to directional ambiguity under arbitrary rotations. We therefore represent normalized articulation axes with spherical coordinates instead of directly generating Cartesian directions (Fig. 5(c)): $\theta = \arccos(z)$, $\phi = \operatorname{atan2}(y, x)$, where θ and ϕ denote the polar and azimuth angles, respectively. This formulation reduces sensitivity to explicit coordinate-wise sign ambiguity while preserving directional information, and provides a smoother angular representation space for articulation reasoning.

To improve pivot estimation, we further introduce a shared global positional encoding for both part-level and object-level point clouds. Specifically, instead of independently normalizing local coordinates, all geometries share the same global voxel coordinate system:

$$\mathbf{g}_i = \left\lfloor \frac{\mathbf{p}_i}{s} \right\rfloor - \mathbf{g}_{\min}^{\text{obj}}, \tag{2}$$

where $\mathbf{g}_{\min}^{\text{obj}}$ is computed from the complete object geometry. The resulting voxel coordinates are subsequently encoded using Fourier positional encoding. Compared with independent local normalization, this strategy preserves global spatial correspondence and substantially improves pivot localization under heterogeneous decompositions.

4.3 Articulation Structure Grounding

Building upon the shared physical grounding representation, UniPhysGen predicts motion-dependent part groups that jointly participate in articulated motion. By reasoning over part identities, geometric locations, and global object context, the model captures articulation-aware structural dependencies without relying on canonical hierarchies or fixed decomposition templates.

4.4 Object-Level Physical Grounding

Finally, UniPhysGen performs holistic object-level physical reasoning by predicting global physical properties such as object category, scale, and mass. Sharing the pretrained physical semantic representation across part-level and object-level tasks enables physically consistent reasoning across semantic granularities while maintaining robustness under heterogeneous 3D assets.

5 EXPERIMENTS

5.1 EXPERIMENTAL SETUP

Implementation Details. UniPhysGen is built upon Qwen3-1.7B (Yang et al., 2025) with Sonata (Wu et al., 2025) as the point cloud encoder. Following SpatialLM (Mao et al., 2025), all grounding tasks are trained using full-parameter fine-tuning. For the image-modality ablation experiments, as illustrated in Fig. 4, we incorporate frozen CLIP ViT-L/14 (Radford et al., 2021) features through a lightweight projector and apply LoRA fine-tuning. See Appendix C for details.

Benchmarks, Baselines, and Metrics. We primarily evaluate unified physical grounding on UniPhys-Bench, which contains jointly annotated articulation semantics and intrinsic physical properties. Additional articulation evaluation on PartNet-Mobility (Xiang et al., 2020) is provided in Appendix F.1. For articulation grounding, we compare with Real2Code (Zhao et al., 2025a), URDFormer (Chen et al., 2024), Articulate-Anything (Le et al., 2025), and PARTICULATE (Li et al., 2026). For intrinsic physical property grounding, we compare against NeRF2Physics (Zhai et al., 2024). Kinematic parameters are evaluated only on ground-truth movable parts, decoupling kinematic estimation from movable-part identification. Following prior articulation evaluation protocols (Le et al., 2025), we report joint type accuracy, axis, pivot, and motion limit metrics, together with articulation structure grounding. For intrinsic physical properties, following NeRF2Physics (Zhai et al., 2024), we report ALDE and MnRE for mass and scale estimation. Density is evaluated using ALDE, while friction and affordance grounding measured by MAE, and material prediction by classification accuracy. Detailed evaluation protocols, fairness adaptations, and metric definitions are provided in Appendix D and Appendix E.

Table 2: Quantitative evaluation of unified physical grounding on UniPhys-Bench.

Method	Articulation Semantics						Intrinsic Physical Properties							
	Kinematic Parameters				Articulation Structure		Material Properties			Scale & Mass Properties				Affordance
	Joint Type Acc ↑	Axis Ang. Err (°) ↓	Pivot Dist. Err ↓	Limit mIoU ↑	Struct. mIoU ↑	Struct. F1 ↑	Mat. Acc ↑	ρ ALDE ↓	μ MAE ↓	Dim. ALDE ↓	Dim. MnRE ↑	Mass ALDE ↓	Mass MnRE ↑	Aff. MAE ↓
Real2Code	60.13	52.26	0.317	–	–	–	–	–	–	–	–	–	–	–
URDFormer	62.13	10.78	0.427	79.61	–	–	–	–	–	–	–	–	–	–
Articulate-Anything*	**94.09**	**5.65**	0.108	72.16	–	–	–	–	–	–	–	–	–	–
PARTICULATE	77.17	12.80	0.261	63.42	–	–	–	–	–	–	–	–	–	–
NeRF2Physics	–	–	–	–	–	–	53.09	0.807	0.142	–	–	0.788	0.524	–
UniPhysGen (Ours)	89.96	9.77	**0.099**	**84.95**	**80.46**	**83.21**	**77.92**	**0.438**	**0.062**	**0.251**	**0.819**	**0.664**	**0.663**	**1.951**

*Note: ρ and μ denote density and friction coefficient, respectively. * Articulate-Anything relies on semantic priors, canonical coordinates, and in-context articulation templates.*

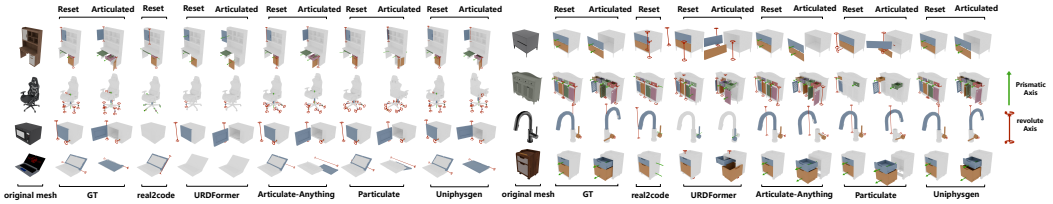

Figure 6: Qualitative comparison of articulation grounding on UniPhys-Bench. "Reset" denotes the initial object state, while "Articulated" denotes the maximally articulated state.

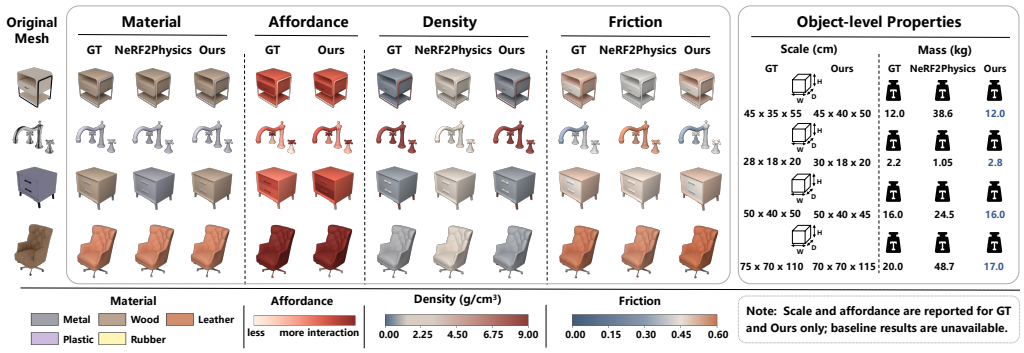

Figure 7: Qualitative comparison of intrinsic physical property grounding on UniPhys-Bench.

5.2 MAIN RESULTS

Unified Physical Grounding Results. Table 2 reports quantitative results on unified physical grounding, covering articulation semantics and intrinsic physical properties. UniPhysGen consistently outperforms prior methods across most metrics, with clear advantages in articulation structure reasoning, pivot localization, motion limit estimation, and intrinsic physical property grounding, demonstrating the effectiveness of unified physical semantic reasoning. Fig. 6 further shows that geometry-centric methods such as PARTICULATE are susceptible to geometric shortcuts on incomplete or ambiguous assets, while Articulate-Anything often produces physically inaccurate pivots and motion limits despite competitive joint-type prediction. In contrast, UniPhysGen achieves more stable articulation grounding by jointly leveraging geometric and physical semantic cues. Beyond articulation grounding, Fig. 7 further shows that UniPhysGen produces more coherent material, affordance, scale, and mass estimates. More results are provided in Appendix F

Table 3: Articulation generalization under different rotation augmentation and axis parameterization.

Training Strategy	**Test Condition**	**Motion Parameters**				
		Joint Type Acc ↑	Axis Ang. Err (°) ↓	Axis Acc ↑	Pivot Dist. Err ↓	Limit mIoU ↑
Canonical Rotation (Z-axis)	Canonical-orientation	80.63	9.46	89.45	0.107	84.51
	Rotated-Z	81.67	9.68	88.81	0.097	84.20
	Challenging-Rotated	50.09	13.19	85.27	0.075	80.96
SO(3)-based Rotation	Canonical-orientation	90.12	6.28	93.16	0.096	85.09
	Rotated-Z	86.39	19.39	69.96	0.118	82.94
	Challenging-Rotated	80.53	30.61	55.46	0.081	79.25
SO(3)-based Rotation + Spherical Augmentation	Canonical-orientation	89.96	9.77	89.33	0.099	84.95
	Rotated-Z	86.25	15.29	82.42	0.105	83.82
	Challenging-Rotated	78.11	23.40	74.05	0.085	80.42

Note: Axis Acc denotes the percentage of predicted axes within 10° *of ground truth. Canonical-Orientation uses original orientations, Rotated-Z applies random z-axis rotations, and Challenging-Rotated evaluates the challenging subset under the same rotations.*

Table 4: Robustness analysis under different part composition granularities.

Part Composition	**Motion Parameters**			
	Joint Type Acc ↑	Axis Ang. Err (°) ↓	Pivot Dist. Err ↓	Limit mIoU ↑
Original Parts	89.96	**9.77**	**0.099**	84.95
Merged Parts	**90.03**	9.85	0.105	**85.04**

Table 5: Ablation study on positional encoding strategies for 3D point cloud tokenization.

Positional Encoding	**Motion Parameters**			
	Joint Type Acc ↑	Axis Ang. Err (°) ↓	Pivot Dist. Err ↓	Limit mIoU ↑
Separate PE	83.26	11.59	0.319	83.72
Unified Global PE	**89.96**	**9.77**	**0.099**	**84.95**

5.3 ROBUSTNESS AND ABLATION ANALYSIS

Robust Articulation Grounding Analysis. (1) Geometric Shortcut Mitigation. As shown in Table 3, we compare canonical rotation augmentation, SO(3) augmentation, and spherical axis parameterization under different test conditions. To stress-test structurally incomplete assets, we construct a Challenging articulation subset containing 360 assets and 1,175 articulated parts, including drawer front panels without underlying drawer bodies. Under canonical rotation augmentation, UniPhysGen performs well on Canonical-Orientation but degrades substantially on Challenging-Rotated, where isolated drawer panels are frequently misclassified as revolute joints due to geometric shortcut bias. SO(3) augmentation improves joint type accuracy on Challenging-Rotated from 50.09% to 80.53%, but makes Cartesian axis prediction less stable under Rotated-Z as the reasoning space expands from canonical planar directions to full spherical orientations (Fig. 5). By further introducing spherical axis parameterization, UniPhysGen achieves more robust axis estimation while maintaining balanced performance across test conditions. **(2) Heterogeneous Part Decomposition Robustness.** We further evaluate UniPhysGen under heterogeneous part decompositions by merging motion-dependent auxiliary components into primary articulated parts according to annotated dependency relations, resulting in coarser decompositions with over 60% merged components. As shown in Table 4, UniPhysGen maintains strong articulation grounding performance across different

decomposition granularities. Notably, joint type and motion limit estimation even improve slightly after merging, suggesting robustness to structurally diverse part abstractions.

Ablation Study. (1) Global Positional Encoding. As shown in Table 5, the proposed unified global positional encoding consistently improves articulation grounding over separate part- and object-level encodings, especially reducing pivot error from 0.319 to 0.099. This demonstrates the benefit of globally consistent spatial indexing for cross-scale geometric reasoning between parts and objects. **(2) Image Modality Ablation.** Table 6 shows that image features improve appearance-related properties such as material classification, density, and friction, but provide limited overall gains. For geometry- and functionality-dominated properties such as scale, mass, and affordance, adding image features leads to slight performance degradation, suggesting that 3D geometry and structural-functional context remain the dominant cues for these properties. **(3) Model Scaling Analysis.** Table 7 compares different backbone sizes. Scaling from 0.6B to 1.7B yields overall improvements across both articulation semantics and intrinsic physical properties, particularly for structure reasoning and material-aware grounding, indicating that larger models better support unified geometric-semantic reasoning.

5.4 Simulation-Based Embodied Interaction

We qualitatively validate UniPhysGen in physics simulation (Isaac Sim) by integrating predicted articulation semantics and intrinsic physical properties into simulation-ready assets and evaluating representative embodied interactions, including drawer pulling, faucet turning, and laptop closing. As shown in Fig. 8, the grounded assets support stable and physically consistent interactions in simulation.

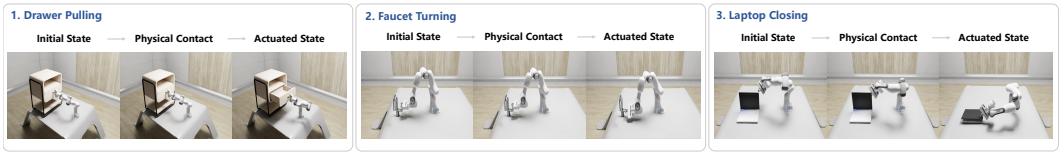

Figure 8: Qualitative simulation-based validation of UniPhysGen-grounded assets. Each sequence shows the initial state, physical contact, and the actuated state after robot interaction.

Table 6: Ablation study on physical property estimation.

Method	Material Properties			Scale & Mass Properties				Affordance
	Mat. Acc ↑	ρ ALDE ↓	μ MAE ↓	Dim. ALDE ↓	Dim. MnRE ↑	Mass ALDE ↓	Mass MnRE ↑	Aff. MAE ↓
UniPhysGen (w/o Image Modality)	77.92	0.438	0.062	**0.251**	**0.819**	**0.664**	**0.663**	**1.951**
UniPhysGen (w/ Image Modality)	**78.22**	**0.435**	**0.062**	0.260	0.813	0.685	0.657	1.981

Table 7: Scaling analysis of UniPhysGen with different model sizes.

Method	Articulation Semantics						Physical Properties							
	Kinematic Parameters				Articulation Structure		Material Properties			Scale & Mass Properties				Affordance
	Joint Type Acc ↑	Axis Ang. Err (°) ↓	Pivot Dist. Err ↓	Limit mIoU ↑	Struct. mIoU ↑	Struct. F1 ↑	Mat. Acc ↑	ρ ALDE ↓	μ MAE ↓	Dim. ALDE ↓	Dim. MnRE ↑	Mass ALDE ↓	Mass MnRE ↑	Aff. MAE ↓
UniPhysGen-0.6B	89.19	**8.66**	0.110	**85.36**	78.71	80.30	74.89	0.475	0.064	0.283	0.801	0.718	0.643	2.108
UniPhysGen-1.7B	**89.96**	9.77	**0.099**	84.95	**80.46**	**83.21**	**77.92**	**0.438**	**0.062**	**0.251**	**0.819**	**0.664**	**0.663**	**1.951**

6 Conclusion

In this paper, we introduce UniPhys, a scalable pipeline for unified physical grounding of heterogeneous 3D assets, enabling the transformation of raw meshes into simulation-ready assets. To support this direction, we construct UniPhys-40K and a manually verified benchmark UniPhys-Bench. Based on these resources, we further propose UniPhysGen, a unified physical grounding model that jointly reasons over articulation semantics and intrinsic physical properties across diverse 3D assets. Extensive experiments demonstrate strong performance and robustness on both articulation grounding and physical property estimation, and enable realistic physical interaction in simulation environments. We hope UniPhys can serve as a foundation for physically grounded 3D understanding and embodied interaction.

AI USE STATEMENT

In this work, we did not use generative AI tools for tasks requiring mandatory disclosure under the ICLR 2027 AI Policy. We used generative AI tools solely for language polishing and editing the manuscript to improve readability, which falls under the recommended disclosure category. We have reviewed all AI-assisted work. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

To facilitate reproducibility, we provide an anonymous repository containing the implementation and evaluation code of UniPhysGen: `https://github.com/anonymous-abc123/anonymous-code`. The supplementary material provides comprehensive details of the UniPhys pipeline, including dataset construction procedures, annotation and verification protocols, model architectures, training configurations, input/output formats, and evaluation protocols. We further provide complete prompt templates, benchmark specifications, and metric definitions to enable faithful reproduction of our results. The UniPhys-40K and UniPhys-Bench annotations and pretrained models will be released with the final version of the paper.

REFERENCES

Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules in large databases. In *VLDB'94, Proceedings of 20th International Conference on Very Large Data Bases, September 12-15, 1994, Santiago de Chile, Chile*, pp. 487–499, 1994.

Ziang Cao, Zhaoxi Chen, Liang Pan, and Ziwei Liu. Physx-3d: Physical-grounded 3d asset generation. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

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, 2026.

Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015.

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. *arXiv preprint arXiv:2405.11656*, 2024.

Jasmine Collins, Shubham Goel, Kenan Deng, Achleshwar Luthra, Leon Xu, Erhan Gundogdu, Xi Zhang, Tomas F Yago Vicente, Thomas Dideriksen, Himanshu Arora, et al. Abo: Dataset and benchmarks for real-world 3d object understanding. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 21094–21104. IEEE, 2022.

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

Matt Deitke, Dustin Schwenk, Jordi Salvador, Luca Weihs, Oscar Michel, Eli VanderBilt, Ludwig Schmidt, Kiana Ehsani, Aniruddha Kembhavi, and Ali Farhadi. Objaverse: A universe of annotated 3d objects. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 13142–13153, 2023b.

Huan Fu, Rongfei Jia, Lin Gao, Mingming Gong, Binqiang Zhao, Steve Maybank, and Dacheng Tao. 3d-future: 3d furniture shape with texture. *International Journal of Computer Vision*, 129 (12):3313–3337, 2021.

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 *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7081–7091. IEEE, 2023.

Denys Iliash, Hanxiao Jiang, Yiming Zhang, Manolis Savva, and Angel X Chang. S2o: Static to openable enhancement for articulated 3d objects. In *2026 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pp. 6785–6795. IEEE, 2026.

Mukul Khanna, Yongsen Mao, Hanxiao Jiang, Sanjay Haresh, Brennan Shacklett, Dhruv Batra, Alexander Clegg, Eric Undersander, Angel X Chang, and Manolis Savva. Habitat synthetic scenes dataset (hssd-200): An analysis of 3d scene scale and realism tradeoffs for objectgoal navigation. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 16384–16393. IEEE, 2024.

Harold W Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.

Zeqiang Lai, Yunfei Zhao, Haolin Liu, Zibo Zhao, Qingxiang Lin, Huiwen Shi, Xianghui Yang, Mingxin Yang, Shuhui Yang, Yifei Feng, et al. Hunyuan3d 2.5: Towards high-fidelity 3d assets generation with ultimate details. *arXiv preprint arXiv:2506.16504*, 2025.

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*, volume 2025, pp. 17578–17602, 2025.

Adrian Leyland and Allan Matthews. On the significance of the h/e ratio in wear control: a nanocomposite coating approach to optimised tribological behaviour. *Wear*, 246(1-2):1–11, 2000.

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.

Minghua Liu, Mikaela Angelina Uy, Donglai Xiang, Hao Su, Sanja Fidler, Nicholas Sharp, and Jun Gao. Partfield: Learning 3d feature fields for part segmentation and beyond. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 9704–9715. IEEE, 2025.

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

Yongsen Mao, Junhao Zhong, Chuan Fang, Jia Zheng, Rui Tang, Hao Zhu, Ping Tan, and Zihan Zhou. SpatialLM: Training large language models for structured indoor modeling. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

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 *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 909–918. IEEE, 2019.

Warren Carl Oliver and George Mathews Pharr. An improved technique for determining hardness and elastic modulus using load and displacement sensing indentation experiments. *Journal of materials research*, 7(6):1564–1583, 1992.

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.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pp. 8748–8763. PmLR, 2021.

Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, et al. Sam 2: Segment anything in images and videos. In *International Conference on Learning Representations*, volume 2025, pp. 28085–28128, 2025.

Aaditya Singh, Adam Fry, Adam Perelman, Adam Tart, Adi Ganesh, Ahmed El-Kishky, Aidan McLaughlin, Aiden Low, AJ Ostrow, Akhila Ananthram, et al. Openai gpt-5 system card. *arXiv preprint arXiv:2601.03267*, 2025.

David Tabor. *The hardness of metals*. Oxford university press, 2000.

Xiaoyang Wu, Daniel DeTone, Duncan Frost, Tianwei Shen, Chris Xie, Nan Yang, Jakob Engel, Richard Newcombe, Hengshuang Zhao, and Julian Straub. Sonata: Self-supervised learning of reliable point representations. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 22193–22204. IEEE, 2025.

Fanbo Xiang, Yuzhe Qin, Kaichun Mo, Yikuan Xia, Hao Zhu, Fangchen Liu, Minghua Liu, Hanxiao Jiang, Yifu Yuan, He Wang, et al. Sapien: A simulated part-based interactive environment. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11094–11104. IEEE, 2020.

Jianfeng Xiang, Zelong Lv, Sicheng Xu, Yu Deng, Ruicheng Wang, Bowen Zhang, Dong Chen, Xin Tong, and Jiaolong Yang. Structured 3d latents for scalable and versatile 3d generation. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 21469–21480. IEEE, 2025.

Jianfeng Xiang, Xiaoxue Chen, Sicheng Xu, Ruicheng Wang, Zelong Lv, Yu Deng, Hongyuan Zhu, Yue Dong, Hao Zhao, Nicholas Jing Yuan, et al. Native and compact structured latents for 3d generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14419–14429, 2026.

Xinli Xu, Wenhang Ge, Dicong Qiu, ZhiFei Chen, Dongyu Yan, Zhuoyun Liu, Haoyu Zhao, Hanfeng Zhao, Shunsi Zhang, Junwei Liang, et al. Gaussianproperty: Integrating physical properties to 3d gaussians with lmms. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 7231–7240. IEEE, 2025.

Xinhao Yan, Jiachen Xu, Yang Li, Changfeng Ma, Yunhan Yang, Chunshi Wang, Zibo Zhao, Zeqiang Lai, Yunfei Zhao, Zhuo Chen, et al. X-part: high fidelity and structure coherent shape decomposition. *arXiv preprint arXiv:2509.08643*, 2025.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Yunhan Yang, Chunshi Wang, Junliang Ye, Yang Li, Zanxin Chen, Zehuan Huang, Yao Mu, Zhuo Chen, Chunchao Guo, and Xihui Liu. Physforge: Generating physics-grounded 3d assets for interactive virtual world. In *Forty-third International Conference on Machine Learning*, 2026.

Albert J Zhai, Yuan Shen, Emily Y Chen, Gloria X Wang, Xinlei Wang, Sheng Wang, Kaiyu Guan, and Shenlong Wang. Physical property understanding from language-embedded feature fields. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 28296–28305. IEEE, 2024.

Mandi Zhao, Yijia Weng, Dominik Bauer, and Shuran Song. Real2code: Reconstruct articulated objects via code generation. In *International Conference on Learning Representations*, volume 2025, pp. 668–686, 2025a.

Zibo Zhao, Zeqiang Lai, Qingxiang Lin, Yunfei Zhao, Haolin Liu, Shuhui Yang, Yifei Feng, Mingxin Yang, Sheng Zhang, Xianghui Yang, et al. Hunyuan3d 2.0: Scaling diffusion models for high resolution textured 3d assets generation. *arXiv preprint arXiv:2501.12202*, 2025b.

Supplementary Material

This appendix provides additional details on the UniPhys pipeline, UniPhysGen implementation, evaluation metrics, baseline adaptation, supplementary experiments, prompt templates, and algorithms.

Contents

- **A. Details of UniPhys Pipeline** ..... A
  - Physically Meaningful Structural Decomposition A.1
  - Intrinsic Physical Property Grounding A.2
  - Geometry-aware Articulation Grounding A.3
  - Simulation-driven Consistency Verification A.4
  - Dataset Construction Details A.5
- **B. Details of UniPhysGen** ..... B
  - Physical Semantic Alignment Pretraining B.1
  - Geometry-Robust Articulation Grounding B.2
  - Articulation Structure Grounding B.3
  - Object-Level Physical Grounding B.4
- **C. Implementation Details** C
- **D. Evaluation Metrics** ..... D
- **E. Baseline Adaptation and Fairness Protocols** ..... E
- **F. More Results** ..... F
  - Results on PartNet-Mobility F.1
  - More Qualitative Comparisons F.2
- **G. Prompt Templates** ..... G
  - Intrinsic Physical Property Grounding Prompt G.1
  - Geometry-aware Articulation Grounding Prompt G.2
- **H. Algorithms** ..... H

A Details of UniPhys Pipeline

A.1 Physically Meaningful Structural Decomposition

Here we provide additional implementation details for the perceptually guided decomposition framework. Given a mesh $\mathcal{M}$, the objective is to recover a physically grounded decomposition $\mathcal{D}^*$ that preserves articulation-sensitive structures, material-aware regions, and functional connectivity for downstream physical grounding.

The decomposition pipeline consists of four stages.

(1) Multi-view Perceptual Prior Construction. We first render the mesh under multiple viewpoints and establish dense face-level correspondences through face-ID rendering:

$$\phi : p \to f_i,$$

where p denotes a rendered pixel and $f_i \in \mathcal{F}$ denotes the associated mesh face. Applying SAM (Ravi et al., 2025) on rendered images produces view-wise segmentations

$$\{\mathcal{S}_v\}_{v=1}^{V},$$

which are subsequently lifted into mesh space through ϕ to construct perceptual grouping priors $\mathcal{P}$.

(2) Cross-view Part Consensus. To estimate the number of physically meaningful structures shared across observations, we aggregate lifted segmentations across views through mesh-level overlap analysis. Specifically, pairwise IoU is computed between lifted mask regions in mesh space, and regions with IoU larger than 0.5 are merged into the same perceptual group. The resulting cross-view perceptual consensus produces an initial part cardinality prior K.

(3) Candidate Decomposition Alignment. We adopt PartField (Liu et al., 2025) as the geometry-aware decomposition backbone due to its continuous feature-field formulation, which enables flexible granularity control while preserving local geometric consistency across structurally heterogeneous meshes. Compared with discrete clustering-based decomposition methods, PartField provides substantially more stable controllable abstraction under varying structural complexities.

Based on the estimated part-count prior K, we generate candidate decompositions over a local range around K:

$$\{\mathcal{D}_k\}_{k=\max(1,K-n)}^{K+m},$$

where n and m control the search extent below and above the estimated prior, respectively. We set $n = 3$ and $m = 15$ by default.

For each candidate decomposition $\mathcal{D}_k$, we evaluate its consistency with the perceptual grouping prior $\mathcal{P}$ by performing Hungarian matching (Kuhn, 1955) independently for each view:

$$\mathcal{L}_{\text{match}}(\mathcal{D}_k) = \frac{1}{V}\sum_{v=1}^{V} \text{Match}\Big(\mathcal{S}_v, \mathcal{D}_k^{(v)}\Big),$$

where $\mathcal{S}_v$ denotes the lifted SAM regions in the v-th view, $\mathcal{D}_k^{(v)}$ denotes the candidate decomposition components restricted to the mesh faces visible in that view, and $\text{Match}(\cdot)$ computes the Hungarian matching cost under the optimal one-to-one assignment between the two sets of regions.

The candidate decomposition with the lowest cross-view matching cost is selected:

$$\mathcal{D}_{\text{match}} = \arg\min_{\mathcal{D}_k} \mathcal{L}_{\text{match}}(\mathcal{D}_k).$$

(4) Structural Refinement. To further eliminate over-segmentation, we exploit the consistency of merge evidence across multiple views. For each view, candidate decomposition components that exhibit sufficient overlap with the same lifted SAM region are grouped into a transaction, yielding a collection of multi-view merge observations. Apriori-based frequent itemset mining (Agrawal & Srikant, 1994) is then applied to discover stable co-occurring merge groups shared across views. To improve robustness, redundant subset patterns are removed and highly overlapping frequent groups

are further consolidated. The resulting merge groups $\mathcal{M}_{\text{freq}}$ are finally applied to refine the selected decomposition:

$$\mathcal{D}^* = \mathcal{R}\left(\mathcal{D}_{\text{match}}, \mathcal{M}_{\text{freq}}\right).$$

The refined decomposition preserves articulation-sensitive structures while reducing view-specific over-segmentation, providing reliable structural support for downstream articulation grounding and intrinsic physical property estimation.

Algorithm 1 summarizes the overall perceptually guided decomposition pipeline. Fig. 9 further visualizes the key stages of the decomposition process, including multi-view SAM priors, Hungarian-matched candidate selection, and structural refinement, and provides qualitative examples against geometry-centric decomposition methods such as PartField (Liu et al., 2025) and P^3-SAM (Ma et al., 2025). These examples illustrate that the proposed decomposition yields physically meaningful part abstractions for downstream articulation grounding and intrinsic physical property estimation.

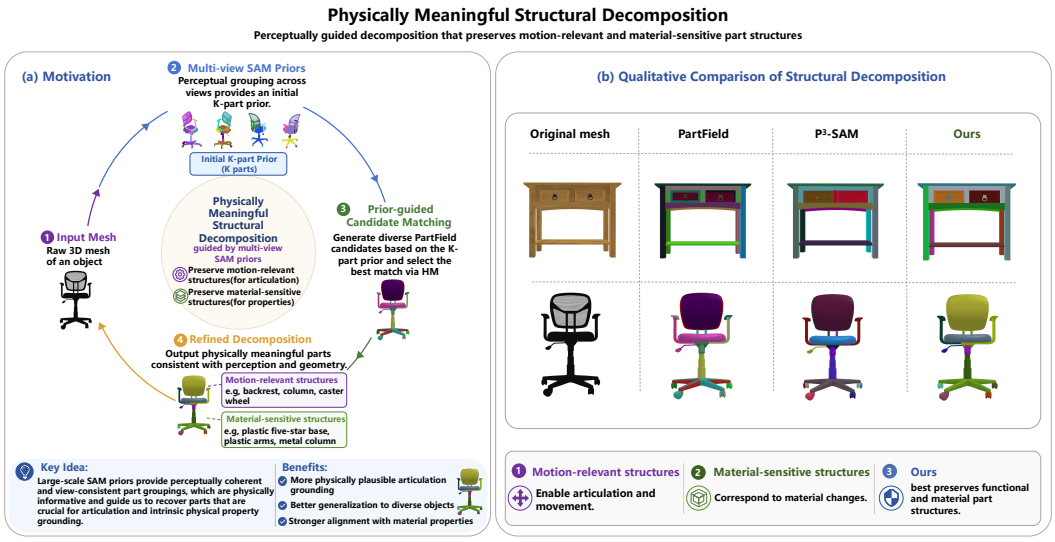

Figure 9: Details of Perceptually guided physically meaningful structural decomposition pipeline.

A.2 Intrinsic Physical Property Grounding

This section provides additional implementation details for the part-aware multimodal physical property grounding stage. Given the physically grounded decomposition $\mathcal{D}^*$, our goal is to infer intrinsic physical properties for both individual parts and the complete object while preserving global structural context.

Context-preserving Part-centric Rendering. For each decomposed component $d_i \in \mathcal{D}^*$, we construct a set of context-preserving renderings rather than isolated part crops. Specifically, the target part is rendered with its original texture and appearance cues, while the remaining object geometry is retained as a semi-transparent or visually weakened contextual reference. This design allows the LVLM to observe fine-grained material appearance of the target part while maintaining awareness of its global structural role and neighboring components. We further combine multi-view object-level observations with localized close-up renderings to capture both global object semantics and part-level material details. The rendered observations of all decomposed parts are organized into a unified part atlas for LVLM reasoning. For clarity, Fig. 10 illustrates only a representative subset of part groups from the complete atlas.

LVLM-based Physical Prior Inference. We use GPT-5 (Singh et al., 2025) as the multimodal reasoning model to infer structured physical priors from the rendered part atlas. The LVLM is prompted to jointly reason over part identity, material semantics, functional affordance, neighboring structural relations, and object-level physical context. The output is constrained into a structured



Figure 10: Visualization of context-preserving part-centric renderings for intrinsic physical property grounding. For each decomposed part, the target component is rendered with preserved appearance cues while the surrounding object structure is retained as contextual reference. Multi-view object observations and localized close-up views are organized into a unified part atlas, enabling the LVLM to jointly reason over material semantics, functional affordance, and global object context.

JSON format containing part-level properties such as material category, density, friction coefficient, elasticity-related attributes, affordance score, and object-level properties such as scale and mass. This structured output format enables automatic parsing and downstream consistency verification.

Prompt Template. To improve annotation consistency, we use a fixed prompt template for all assets. As shown in Listing 9, the prompt specifies the role of the LVLM, the organization of the visual input atlas, the definition of each physical attribute, the required structured output schema, and constraints on physical plausibility, numerical ranges, and part-label consistency.

A.3 Geometry-aware Articulation Grounding

This section provides additional details for the geometry-aware articulation grounding stage. Given the motion dependency graph (MDG) constructed from neighboring relations, UniPhys generates geometrically feasible articulation hypotheses for each candidate parent-child motion pair, and then uses a multimodal reasoning model to select the most semantically plausible hypothesis from the constrained candidate space.

Geometry-constrained Candidate Generation. For each movable component, we first identify the local contact regions between the component and its parent component. The contact geometry is used to fit a motion-consistent plane, from which candidate articulation axes are derived. Specifically, we sample six in-plane directions from the fitted motion plane and include the plane normal as an additional candidate. We further include the three global coordinate axes, i.e., the x-, y-, and z-axes, as valid reference candidates, resulting in ten candidate motion axes in total. These candidate axes are rendered from two diagonal viewpoints to improve spatial disambiguation.

For revolute joints, UniPhys additionally generates pivot candidates from the contact geometry. We cluster contact points into five groups and use their cluster centers as candidate pivot points. We further include the centroid of the entire contact point cloud as an additional candidate, resulting in six pivot candidates. To make pivot locations visually distinguishable, the target part is rendered semi-transparently while pivot candidates are shown as opaque colored markers.

LVLM-based Semantic Selection. The generated articulation hypotheses are rendered into a structured visual input set, as illustrated in Fig. 11. For each movable part, the LVLM (GPT-

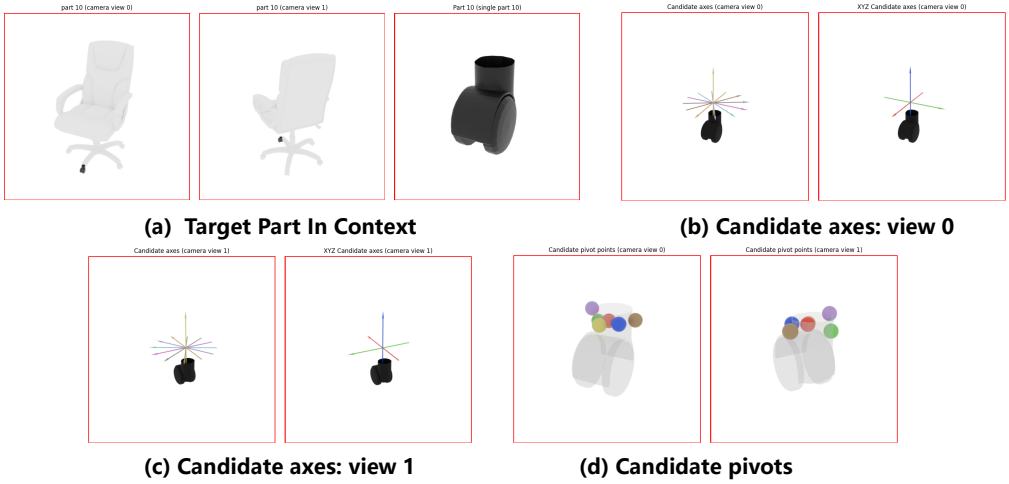

Figure 11: Visualization of the LVLM input for geometry-aware articulation grounding. For each movable part, UniPhys provides an assembly visualization, two candidate-axis views, and, for revolute joints, pivot-candidate views. The candidate axes and pivot points are generated from contact-aware geometric reasoning and rendered as visual-semantic hypotheses, allowing the LVLM to select the physically plausible articulation configuration from a constrained candidate space.

5 (Singh et al., 2025)) receives: (1) an assembly visualization showing the target part in context; (2) two candidate-axis views containing the ten rendered axes; and (3) for revolute joints, two pivot-candidate views. We additionally provide the motion type and part-level textual descriptions from the intrinsic physical property grounding stage, including basic, functional, and movement descriptions. This allows the LVLM to select the articulation axis and pivot point by combining geometric feasibility with semantic affordance reasoning.

Prompt Template. To improve annotation consistency, we use a fixed prompt template for articulation grounding. As shown in Listing 10, the prompt specifies the LVLM role, the organization of rendered inputs, the candidate-axis and pivot-color conventions, the motion type, the provided part descriptions, and the required structured JSON output schema.

A.4 Simulation-driven Consistency Verification

This section provides implementation details for the simulation-driven consistency verification framework. Given automatically grounded annotations, the goal is not to recover exact real-world physical quantities, but to determine whether the annotations satisfy internally consistent and simulation-admissible constraints for downstream use.

We instantiate the verification framework with two complementary protocols: (i) *intrinsic physical property verification*, which checks numerical feasibility, material-aware plausibility, intra-part consistency, structural constraints, and object-level mass consistency; and (ii) *articulation consistency verification*, which evaluates whether inferred kinematic configurations remain stable during articulated simulation. Annotations failing these checks are either filtered or refined through a lightweight feedback-based correction mechanism. The overall verification procedure is summarized in Algorithm 2.

A.4.1 Intrinsic Physical Property Verification.

Given a grounded object with decomposed parts $\{p_i\}_{i=1}^{N}$, UniPhys verifies the physical plausibility of part-level properties through four levels.

Level 1: Numerical Feasibility and Material Prior. We first check whether each physical attribute lies within a globally admissible numerical range and whether it is compatible with material-specific empirical priors. For a part assigned to a known material category m_i, its density, friction

coefficient, Young's modulus, hardness, and Poisson's ratio are required to fall within the corresponding material-dependent plausible intervals. Parts assigned to the "other" material category are excluded from material-specific prior checking.

Level 2: Intra-part Physical Consistency. We further check whether physical attributes within a single part are mutually compatible. Before consistency verification, Vickers hardness values are converted from the conventional HV scale to Pascals using

$$H_{\mathrm{Pa}} = 9.807 \times 10^6 H_{\mathrm{HV}},$$

while Young's modulus values are converted from GPa to Pascals using

$$E_{\mathrm{Pa}} = 10^9 E_{\mathrm{GPa}}.$$

Motivated by prior studies on the relationship between hardness and elastic properties and the use of hardness-to-modulus ratios for characterizing mechanical behavior (Tabor, 2000; Oliver & Pharr, 1992; Leyland & Matthews, 2000), we use the hardness-to-Young's-modulus ratio H/E as a plausibility indicator for engineering materials where hardness has a meaningful mechanical interpretation:

$$\frac{H_i}{E_i} \in [10^{-3}, 10^{-1}],$$

where H_i and E_i denote the converted hardness and Young's modulus of part i, respectively. The range is selected as a broad empirical threshold for large-scale annotation verification. This rule serves as an empirical plausibility heuristic rather than a strict physical law. Soft, porous, highly deformable, or anisotropic material categories such as rubber, foam, fabric, leather, and "other" are excluded from this check.

Level 3: Structural Consistency Across Parts. We impose structure-induced physical constraints derived from the functional role of parts. For example, highly interactive or graspable components are required to have sufficient surface friction:

$$\mu_i \geq 0.2,$$

where μ_i denotes the friction coefficient of part p_i. This level verifies whether local physical attributes are consistent with part-level affordance and interaction semantics.

Level 4: Object-level Mass Consistency. Finally, object-level mass is used as a global consistency constraint over part-level density and volume annotations. The mass induced by part-level annotations is estimated as:

$$\hat{M}_{\text{parts}} = \sum_{i=1}^{N} \rho_i V_i,$$

where ρ_i and V_i denote the density and estimated volume of part p_i, respectively. We define the relative mass inconsistency as:

$$\epsilon = \frac{|\hat{M}_{\text{parts}} - M_{\text{obj}}|}{\max(M_{\text{obj}}, 10^{-6})},$$

where M_{obj} denotes the object-level mass annotation. Annotations with $\epsilon > 2$ are considered globally inconsistent and are filtered or sent to feedback-based correction. This threshold is selected to balance annotation coverage and physical plausibility, rather than to enforce high-precision physical simulation.

Part Volume Estimation. Part volumes are estimated through voxelized mesh approximation. For each part mesh, we voxelize its geometry at a resolution of 128^3 relative to the axis-aligned bounding box and compute the volume as the number of occupied voxels multiplied by the voxel volume. A global scaling factor is then applied to align mesh-space volume with real-world object dimensions:

$$s_V = \frac{V_{\mathrm{AABB}}^{\mathrm{real}}}{V_{\mathrm{AABB}}^{\mathrm{mesh}}}, \qquad V_i^{\mathrm{real}} = s_V V_i^{\mathrm{mesh}}.$$

The scaled part volumes and associated physical quantities are normalized to a consistent unit system (e.g., m^3 for volume and kg for mass) before consistency verification. This approximation provides a robust and efficient estimate for large-scale consistency verification.

A.4.2 ARTICULATION CONSISTENCY VERIFICATION.

For articulated annotations, UniPhys validates whether the inferred kinematic configuration remains physically feasible during motion simulation. Given an articulated part, we simulate its predicted motion in MuJoCo and evaluate three levels of motion consistency.

Level 1: Contact Preservation. Let $\mathcal{S}_m$ and $\mathcal{S}_s$ denote the surfaces of the movable component and its adjacent component, respectively. For a distance threshold τ, the contact region is defined as

$$\mathcal{P}(\tau) = \left\{ \mathbf{x} \in \mathcal{S}_s \;\middle|\; \min_{\mathbf{y} \in \mathcal{S}_m} \|\mathbf{x} - \mathbf{y}\|_2 < \tau \right\}.$$

The initial contact threshold τ_0 is obtained as the minimum threshold that yields a valid contact region. During simulation, the corresponding threshold τ_t is recomputed at every simulation step. The articulated motion is regarded as invalid if

$$\tau_t > \tau_0 + \delta,$$

where δ is a tolerance margin. This criterion indicates that maintaining the original contact requires an unrealistically enlarged contact distance, implying that the articulated parts have effectively lost physical contact.

Level 2: Penetration Stability. We detect severe interpenetration during articulated motion by comparing the thickness of the contact region before and after articulation. Given the initial contact point set $\mathcal{P}_0 = \{\mathbf{p}_i\}$, we estimate its dominant contact plane

$$\mathbf{n}^\top \mathbf{x} + D = 0,$$

using robust plane fitting, where $\mathbf{n}$ is the unit plane normal and D is the plane offset.

The contact thickness is defined as the extent of the contact points projected onto the plane normal,

$$T(\mathcal{P}) = \max_{\mathbf{p} \in \mathcal{P}} (\mathbf{n}^\top \mathbf{p} + D) - \min_{\mathbf{p} \in \mathcal{P}} (\mathbf{n}^\top \mathbf{p} + D).$$

Let

$$T_{\text{ori}} = T(\mathcal{P}_0)$$

denote the initial contact thickness. During simulation, the contact region is recomputed at every simulation step, producing a contact point set $\mathcal{P}_t$ and the corresponding thickness ratio

$$r_t = \frac{T(\mathcal{P}_t)}{T_{\text{ori}}}.$$

A motion is regarded as physically implausible if

$$\text{Percentile}_{95}(r_t) > \gamma,$$

where γ is a predefined threshold. A large thickness ratio indicates excessive expansion of the contact region along the contact normal, suggesting severe geometric interpenetration.

Level 3: Contact Identity Consistency. Although local contact locations may vary during articulated motion, the contact region should remain associated with the same physical interface. For each simulation step, we establish nearest-neighbor correspondences between the initial contact point set $\mathcal{P}_0$ and the current contact point set $\mathcal{P}_t$. The contact identity overlap is defined as

$$O_t = \frac{|\{\mathbf{p} \in \mathcal{P}_0 \mid \text{dist}(\mathbf{p}, \mathcal{P}_t) < \epsilon\}|}{|\mathcal{P}_0|},$$

where $\text{dist}(\mathbf{p}, \mathcal{P}_t)$ denotes the Euclidean distance from $\mathbf{p}$ to its nearest point in $\mathcal{P}_t$, and ϵ is a predefined matching tolerance.

The articulation is considered inconsistent if

$$\text{Percentile}_{10}(O_t) < \eta,$$

where η is the minimum acceptable overlap ratio. A persistently low overlap indicates that the original contact interface has been replaced by unrelated contact regions, suggesting an implausible articulation configuration.

A.5 Dataset Construction Details

This section provides additional details on the construction of UniPhys-40K and UniPhys-Bench. UniPhys-40K is built from diverse 3D repositories and further processed by the UniPhys pipeline to obtain unified articulation semantics and intrinsic physical property annotations.

Objaverse-Sketchfab Curation. We use the Sketchfab subset of Objaverse-XL (Deitke et al., 2023a), also known as ObjaverseV1 (Deitke et al., 2023b), as one of the primary sources of raw 3D assets. Although the assets are collected from a single community-driven platform, this subset contains a large-scale and diverse collection of Creative-Commons-licensed 3D models, covering broad object categories, visual styles, and creation processes, including artist-designed models, procedurally created assets, and real-world 3D scans. To better align the data with household, robotics, and embodied-AI scenarios, we select category-labeled assets related to furniture, containers, appliances, tools, and other interactive indoor objects. We perform category-based filtering over both LVIS-labeled and non-LVIS category annotations, followed by duplicate removal and quality filtering. This filtering process removes empty-category assets, irrelevant outdoor or decorative categories, and geometrically unsuitable meshes, producing a curated subset for downstream physical grounding.

Quality-filtered Repository Assets. In addition to Objaverse-Sketchfab, UniPhys-40K also incorporates assets from HSSD (Khanna et al., 2024), 3D-FUTURE (Fu et al., 2021), ABO (Collins et al., 2022), and PartNet (Mo et al., 2019). For Objaverse, HSSD, 3D-FUTURE, and ABO, we adopt quality-filtered assets released by Trellis (Xiang et al., 2025), which improves geometric fidelity and removes severely corrupted or incomplete meshes. These assets provide diverse object categories and appearance variations, while the UniPhys pipeline further converts them into physically grounded assets with consistent articulation and intrinsic physical property semantics.

PartNet Texture Recovery. PartNet (Mo et al., 2019) provides high-quality hierarchical part decompositions based on ShapeNet assets, but the released part meshes often lack the original texture and appearance information required for material-aware physical property grounding. To make PartNet assets suitable for intrinsic physical property annotation, we recover part-level textures from the corresponding ShapeNet models.

Specifically, we first align each PartNet object with its corresponding ShapeNet mesh using the coordinate transformation between the two representations. We then split the ShapeNet mesh into material-aware submeshes according to its OBJ/MTL material references. For each PartNet component, we identify the best-matched ShapeNet submesh based on local geometric overlap in the aligned coordinate system. When UV coordinates and texture maps are available, we transfer the texture to the PartNet component through nearest-surface projection and barycentric UV interpolation; otherwise, we fall back to transferring available material appearance attributes. Finally, the textured components are regrouped according to an appropriate PartNet hierarchy level to avoid overly coarse or excessively fragmented decompositions.

This part-aware texture recovery preserves fine-grained component-level appearance cues, which are important for material, density, friction, affordance, and other intrinsic physical property annotations.

UniPhys-Bench Construction. To support reliable evaluation, we construct UniPhys-Bench as a carefully verified benchmark containing 1,927 articulated objects with 5.5K motion-relevant components. The benchmark consists of two complementary subsets. The first subset contains 454 objects sampled from the same data distribution as UniPhys-40K but held out exclusively for evaluation and is manually refined with human verification and refinement of articulation annotations and intrinsic physical property plausibility. The second subset contains 1,473 articulated objects provided by an industry partner, where part decompositions are created by professional designers. This designer-created subset introduces additional diversity in part granularity, structural organization, and asset modeling style, making the benchmark more representative of heterogeneous real-world 3D assets. For both subsets, articulation parameters and intrinsic physical properties are initially annotated by the UniPhys pipeline and subsequently inspected and corrected by human annotators. Human verification focuses on motion-relevant components, including joint type, motion axis, pivot,

motion range, motion-dependent part groupings, and physical property plausibility. UniPhys-Bench therefore provides a rigorous testbed for evaluating unified physical grounding under diverse object categories, structural complexities, and part decomposition styles.

B Details of UniPhysGen

B.1 Physical Semantic Alignment Pretraining

This section provides additional details on the physical semantic alignment pretraining stage. The goal of this stage is to align 3D geometric tokens with structured physical semantic descriptions, so that the model can jointly reason over local part geometry, global object context, articulation behavior, and intrinsic physical properties.

Input Representation. For each training sample, the model receives both the target part point cloud and the complete object point cloud. The target part provides fine-grained local geometry and material-sensitive shape cues, while the complete object provides global structural context for functional and articulation-aware reasoning. Both point clouds, with each point represented by 3D coordinates, RGB color, and surface normal vectors, are encoded using the Sonata point cloud encoder (Wu et al., 2025), and the resulting geometric tokens are projected into the language model embedding space through a lightweight two-layer MLP projector.

Instruction Template. During physical semantic alignment pretraining, UniPhysGen follows an instruction-tuning format. Given the target part point cloud and the complete object point cloud, the model is prompted to generate a structured physical semantic description of the target part. The instruction specifies that the output should include part identity, semantic descriptions, articulation behavior, and intrinsic physical properties in a machine-readable JSON-style format. A compact instruction template is shown in Listing 1.

Listing 1: Full instruction template for physical semantic alignment pretraining.

```
[PHYSICS]
Estimate part identity, semantic descriptions, and physical properties.

# Multimodal Inputs
<object_point_cloud>
<part_point_cloud>
<image>

# Notes
- affordance: smaller value means higher affordance.
- motion_type:
  A = contact-only (no relative motion),
  B = translation,
  C = rotation,
  D = rigid (fixed, no motion).
- Units: density in g/cm^3 (if applicable), young (Young's modulus) in GPa,
  hardness in HV, poisson (Poisson's ratio) unitless, friction unitless.

# Output (JSON only)
Return a single JSON object with the following schema (no extra keys):
{
  "part_identity": {
    "part_name": string|null,
    "motion_type": "A"|"B"|"C"|"D"|null,
  },
  "semantic_description": {
    "basic": string|null,
    "functional": string|null,
    "movement": string|null,
    "grasp": string|null
  },
  "physical_properties": {
    "material": string|null,
    "density": number|null,
    "young": number|null,
    "hardness": number|null,
    "poisson": number|null,
    "friction": number|null,
    "graspable": boolean|null,
    "affordance": 1|2|3|4|5|6|7|8|9|10|null
```

```
  }
}
```

Structured Physical Semantic Supervision. Instead of supervising isolated scalar attributes independently, we organize the target output as a structured physical semantic description. Each annotation contains three complementary levels: (i) *part identity*, which specifies the part name and motion type; (ii) *semantic description*, which describes the part's appearance, function, movement behavior, and human interaction mode; and (iii) *physical properties*, which include material category, density, Young's modulus, hardness, Poisson's ratio, friction coefficient, graspability, and affordance score. This hierarchical supervision format encourages the model to learn correlations among geometry, function, articulation behavior, and material-dependent physical properties within a shared physical semantic space. Listing 2 shows an example of the structured physical semantic supervision used during pretraining. The output is represented in a JSON-style format for automatic parsing while preserving open-ended semantic descriptions.

Listing 2: Example structured physical semantic supervision used for physical semantic alignment pretraining.

```
{
  "part_identity": {
  "part_name": "Door Window Glass",
  "motion_type": "D"
    },
    "semantic_description": {
      "basic": "Circular soda-lime glass window mounted in the door; diameter ~30 cm,
          thickness ~0.6 cm.",
      "functional": "Provides visibility into the drum while maintaining a barrier to water
          and detergent.",
      "movement": "Rigidly fixed to the door frame; moves only when the door rotates.",
      "grasp": "Not grasped; occasionally touched during cleaning."
    },
    "physical_properties": {
      "material": "glass/Soda_Lime_Glass",
      "density": 2.5,
      "young": 70.0,
      "hardness": 550.0,
      "poisson": 0.23,
      "friction": 0.5,
      "graspable": false,
      "affordance": 6
    }
}
```

By learning to generate such structured descriptions from part-level and object-level geometry, the model establishes a unified physical grounding representation that is subsequently shared by downstream tasks, including kinematic parameter estimation, articulation structure grounding, and object-level physical property estimation.

B.2 Geometry-Robust Articulation Grounding

While large vision-language models possess strong semantic priors for reasoning about object functionality, articulation grounding remains particularly sensitive to geometry shortcut learning and rotational bias. Unlike intrinsic physical property grounding, articulation parameter estimation requires motion-aware reasoning over geometrically equivariant structures, making it substantially more challenging under heterogeneous and incomplete 3D assets.

B.2.1 View-Dependent Bias under Canonical Training

We first observe that commonly adopted canonical augmentation strategies based on upright indoor priors introduce strong view-dependent bias during articulation reasoning. Existing approaches typically assume the input object is aligned with the global upright direction and only apply rotation augmentation around the vertical axis:

$$\mathbf{p}' = \mathbf{R}_z(\theta)\mathbf{p}, \tag{3}$$

where $\mathbf{p} \in \mathbb{R}^3$ denotes a point coordinate and $\mathbf{R}_z(\theta)$ represents a rotation matrix around the z-axis.

Under this setting, articulation axis prediction is effectively constrained to a low-dimensional subspace on the unit circle S^1. Although such augmentation performs well under canonical indoor configurations, we observe severe degradation under structure-degenerate cases frequently appearing in 3D asset repositories. For example, incomplete drawer assets containing only front panels are often incorrectly predicted as rotational door-like structures due to shortcut correlations learned from canonical upright priors.

B.2.2 SO(3)-BASED ROTATION-ROBUST ARTICULATION LEARNING

To mitigate geometry shortcut learning, we remove canonical orientation assumptions and introduce SO(3)-based rotation augmentation:

$$\mathbf{p}' = \mathbf{R}\mathbf{p}, \quad \mathbf{R} \in SO(3), \tag{4}$$

where $\mathbf{R}$ denotes an arbitrary 3D rotation sampled from the special orthogonal group $SO(3)$.

Compared with Eq. 3, articulation reasoning is no longer restricted to a planar rotational subspace and instead becomes a spherical reasoning problem over S^2, as illustrated in Fig. 5. This significantly improves structure-aware articulation reasoning by forcing the model to rely on motion-relevant geometric semantics rather than view-dependent structural shortcuts.

However, we further observe that naïvely introducing SO(3) augmentation substantially destabilizes articulation axis estimation under arbitrary test-time rotations.

B.2.3 SPHERICAL AXIS PARAMETERIZATION

We identify that the instability under SO(3)-based augmentation partly originates from token-level sensitivity in motion direction representation. Under arbitrary 3D rotations, articulation axis prediction is lifted from a constrained planar space into a full spherical solution space, substantially increasing the difficulty of open-ended axis generation. Directly generating Cartesian directions is particularly sensitive to discrete sign tokens in LLM decoding. For example, two normalized directions

$$\mathbf{a}_1 = (0.707, 0.707, 0), \qquad \mathbf{a}_2 = (-0.707, 0.707, 0)$$

differ only by a negative sign in the first coordinate, but correspond to substantially different parsed 3D directions. Such a small token-level change can lead to a large angular error after parsing, even when the textual supervision loss changes only slightly.

To mitigate this representation instability, we represent normalized articulation axes using spherical coordinates rather than directly generating Cartesian vectors:

$$\theta = \arccos(z), \qquad \phi = \operatorname{atan2}(y, x),$$

where $\mathbf{a} = (x, y, z)$ denotes a normalized axis direction, $\theta \in [0, \pi]$ is the polar angle, and $\phi \in [0, 2\pi)$ is the azimuth angle. This angular representation reduces reliance on explicit coordinate sign tokens and provides a smoother output space for LLM-based axis generation under arbitrary object rotations. The corresponding Cartesian direction can be recovered by:

$$\mathbf{a} = \begin{bmatrix} \sin\theta\cos\phi \\ \sin\theta\sin\phi \\ \cos\theta \end{bmatrix}.$$

B.2.4 GLOBAL POSITION ENCODING FOR PIVOT LOCALIZATION

Accurate pivot estimation further requires consistent spatial reasoning between local part geometry and global object structure. Existing methods typically normalize part-level and object-level coordinates independently before positional encoding:

$$\mathbf{g}_i = \left\lfloor \frac{\mathbf{p}_i}{s} \right\rfloor - \min_j \left(\left\lfloor \frac{\mathbf{p}_j}{s} \right\rfloor \right), \tag{5}$$

where s denotes the voxel grid size.

Although effective for isolated geometry understanding, such local normalization destroys global spatial correspondence between part-level and object-level geometries, leading to unstable pivot localization under heterogeneous decompositions.

To address this issue, we introduce a shared global grid coordinate system jointly applied to both part-level and object-level point clouds:

$$\mathbf{g}_i = \left\lfloor \frac{\mathbf{p}_i}{s} \right\rfloor - \mathbf{g}_{\min}^{\text{obj}}, \tag{6}$$

where $\mathbf{g}_{\min}^{\text{obj}}$ is computed globally from the complete object geometry and shared across all part-level representations.

The resulting global grid coordinates are first normalized as

$$\tilde{\mathbf{g}} = \frac{\mathbf{g}}{G-1}, \tag{7}$$

where G denotes the reduced grid resolution after hierarchical voxel pooling.

The normalized coordinates are subsequently encoded using Fourier positional encoding:

$$\gamma(\tilde{\mathbf{g}}) = \Big[\tilde{\mathbf{g}}, \sin(\omega_1\tilde{\mathbf{g}}), \cos(\omega_1\tilde{\mathbf{g}}), \ldots, \sin(\omega_L\tilde{\mathbf{g}}), \cos(\omega_L\tilde{\mathbf{g}})\Big], \tag{8}$$

where

$$\omega_l = \pi\lambda_l, \qquad \lambda_l = 1 + \frac{l-1}{L-1}\left(\frac{R}{2} - 1\right), \quad l = 1, \ldots, L,$$

with L denoting the number of frequency bands and R the sampling rate.

Unlike independent local coordinate normalization, the proposed global positional encoding preserves spatial consistency between local part geometry and the complete object structure, substantially improving pivot localization accuracy across heterogeneous 3D assets.

B.2.5 Instruction and Supervision Format

UniPhysGen performs kinematic parameter grounding through open-ended text generation rather than a fixed regression head. Given the target part point cloud and the complete object point cloud, the model is prompted to infer the complete kinematic parameters—including the joint type (referred to as motion type in our annotation), articulation axis, pivot location, and joint limit (referred to as motion range in our annotation)—in a structured JSON format. For kinematic parameter grounding, motion types follow our annotation convention: B denotes prismatic translation, while C denotes revolute rotation. The axis is represented using spherical angles, while the pivot and motion range are represented as continuous numeric values for subsequent parsing and evaluation.

Listing 3: Full instruction template for kinematic parameter grounding.

```
[MOTION]
Estimate the kinematic motion of the target part.

# Multimodal Inputs
<object_point_cloud>
<part_point_cloud>

# Notes
- The input point clouds are AABB-normalized and shifted to [0, 2].
- motion_type uses letters: B = translation (prismatic), C = rotation (revolute).
- axis is represented by spherical coordinates (degrees):
  - theta: polar angle from +Z, integer in [0, 180].
  - phi: azimuth angle around Z, integer in [0, 360).
- pivot is a 3-number point [x, y, z] in the same coordinate frame as the point cloud.
- range is a 2-number array [min, max].
- For C (revolute), range is normalized by 2π.
- For B (prismatic), range is in normalized length units.

# Output (JSON only)
Return a single JSON object with the following schema (no extra keys):
{
  "motion_type": "B"|"C"|null,
  "axis": {"theta": integer, "phi": integer}|null,
  "pivot": [number, number, number]|null,
  "range": [number, number]|null
}
```

An example supervision target is shown in Listing 4.

Listing 4: Example structured supervision target for kinematic parameter grounding.

```
{
  "motion_type": "C",
  "axis": {
    "theta": 90,
    "phi": 90
  },
  "pivot": [
    0.565,
    0.1032,
    1.1591
  ],
  "range": [
    0.0,
    1.0
  ]
}
```

B.3 Articulation Structure Grounding

This section provides additional details on the articulation structure grounding task. Different from kinematic parameter grounding, which predicts joint type, axis, pivot, and limit for a target part, articulation structure grounding predicts the set of parts that move together with the target part during articulation. This task captures motion-coupled structural dependencies under heterogeneous part decompositions.

Input and Output Format. For each target part, the model receives the complete object point cloud, the target part point cloud, optional visual observations, and a list of all part identities with their centroid locations. The part list provides an explicit global structural reference, allowing the model to reason over which components are rigidly attached or motion-coupled with the target part. The output is a subset of input part IDs, where the predicted members must include the target part itself. This formulation avoids assuming a fixed articulation hierarchy and allows motion-dependent structural grouping under diverse decomposition granularities.

Instruction Template. Listing 5 shows the instruction template used for articulation structure grounding.

Listing 5: Full instruction template for articulation structure grounding.

```
[GROUP]
Predict the motion-coupled part group(s): which parts will move together when the target
     part moves.

# Inputs (placeholders)
- parts(id,position): {{PART_LIST}}

# Multimodal Inputs
<object_point_cloud>
<part_point_cloud>
<image>

# Constraints
- "members" must be selected ONLY from the part ids provided in the input parts list.
- Do NOT generate new ids.
- Each id should appear at most once.
- The output must be a subset of the input part ids.
- "members" must include the target part itself.

# Output (JSON only)
Return a single JSON object with the following schema (no extra keys):
{
  "members": [number]
}
```

Supervision Example. An example supervision target is shown in Listing 6. In this example, parts 1, 5, and 6 form a motion-coupled group and are expected to move together when the target part is articulated.

Listing 6: Example supervision target for articulation structure grounding.

```
{
  "members": [
    1,
    5,
    6
  ]
}
```

B.4 Object-Level Physical Grounding

This section provides additional details on the object-level physical grounding task. While previous tasks focus on target-part reasoning, this task estimates holistic physical properties from the complete object geometry and visual appearance. It encourages UniPhysGen to share the pretrained physical semantic representation across both part-level and object-level grounding.

Input and Output Format. Given the complete object point cloud and optional visual observations, the model predicts the object identity, category, global dimensions, and mass. The dimensions are represented as a three-number array in centimeters, corresponding to the object-level length, width, and height. The mass is represented in kilograms. This object-level task complements part-level physical grounding by providing global scale and mass supervision, which is important for simulation initialization and physical consistency checking.

Instruction Template. Listing 7 shows the instruction template used for object-level physical grounding.

Listing 7: Full instruction template for object-level physical grounding.

```
[OBJECT_LEVEL]
Estimate object identity and object-level physical properties.

# Multimodal Inputs
<object_point_cloud>
<image>

# Notes
- volume is a 3-number array [L, W, H] in centimeters (cm).
- mass is in kilograms (kg).

# Output (JSON only)
Return a single JSON object with the following schema (no extra keys):
{
  "object_name": string|null,
  "category": string|null,
  "volume": [number, number, number]|null,
  "mass": number|null
}
```

Supervision Example. An example supervision target is shown in Listing 8.

Listing 8: Example supervision target for object-level physical grounding.

```
{
  "object_name": "Chair",
  "category": "Furniture/SeatingFurniture",
  "volume": [
    60.0,
    60.0,
    95.0
  ],
  "mass": 10.5
}
```

C Implementation Details

This section provides additional training and implementation details for UniPhysGen. All experiments are conducted on NVIDIA H20 GPUs with approximately 96GB memory per GPU. UniPhys-

Gen is built upon Qwen3-1.7B (Yang et al., 2025) and uses Sonata (Wu et al., 2025) as the point cloud encoder. Following SpatialLM (Mao et al., 2025), we train UniPhysGen with full-parameter fine-tuning for all main grounding tasks. The training configurations for different stages are summarized in Table 8.

Common Training Configuration. Unless otherwise specified, we use a learning rate of 1×10^{-5} with a cosine learning-rate scheduler and a warmup ratio of 0.03. Training is performed with bfloat16 precision and TF32 enabled. The per-device batch size is set to 2 with gradient accumulation of 4. Thus, the effective batch size is 64 for 8-GPU training and 32 for 4-GPU training.

Training Stages. The physical semantic alignment pretraining stage is trained on approximately 370K part-level samples from UniPhys-40K for 6 epochs using 8 H20 GPUs. This stage aligns part-level and object-level geometry with structured physical semantic descriptions. For downstream task-specific fine-tuning, we use the pretrained checkpoint as initialization. Motion parameter grounding is trained for 20 epochs on 4 GPUs, since this task requires joint reasoning over joint type, axis, pivot, and joint limit and empirically converges more slowly. Articulation structure grounding and object-level physical grounding are trained for 10 and 6 epochs, respectively, on 4 GPUs.

Image Modality Ablation. For the image-modality ablation experiments, we incorporate CLIP ViT-L/14 (Radford et al., 2021) image features through a lightweight projector, as shown in Fig. 4. The CLIP encoder is frozen, and the language model is adapted with LoRA fine-tuning. We train the image-enhanced variants for 3 epochs using the same optimization settings unless otherwise specified.

Table 8: Training configurations of UniPhysGen.

Stage	GPUs	Epochs	LR	Batch / GPU	Grad. Acc.	Global Batch Size	Tuning
Physical Semantic Alignment Pretraining	$8 \times$ H20	6	1×10^{-5}	2	4	64	Full
Kinematic Parameter Grounding	$4 \times$ H20	20	1×10^{-5}	2	4	32	Full
Articulation Structure Grounding	$4 \times$ H20	10	1×10^{-5}	2	4	32	Full
Object-Level Physical Grounding	$4 \times$ H20	6	1×10^{-5}	2	4	32	Full
Image-Modality Ablation	$4 \times$ H20	3	1×10^{-5}	2	4	32	LoRA

D Evaluation Metrics

This section provides additional details on the evaluation metrics. Since UniPhysGen generates open-ended text outputs that are parsed into structured annotations, we evaluate each task using task-specific structured metrics.

Kinematic Parameter Metrics. We adopt a decoupled evaluation protocol for kinematic parameters. Following prior articulation evaluation protocols (Le et al., 2025), we report joint type accuracy and axis angular error. For pivot localization and limit (motion range) estimation, we use decoupled metrics that separately evaluate pivot-to-axis localization and interval coverage, avoiding the coupling between axis-direction errors, pivot localization errors, and motion-limit errors.

For motion type prediction, we report joint type accuracy.

For joint type reasoning, let $\hat{t}_i$ and t_i denote the predicted and ground-truth joint type of the i-th evaluated component, respectively. Joint type accuracy is defined as:

$$\mathrm{Acc}_{\mathrm{joint}} = \frac{1}{N} \sum_{i=1}^{N} \mathbb{1}\left[\hat{t}_i = t_i\right],$$

where N denotes the number of evaluated motion-relevant components and $\mathbb{1}[\cdot]$ is the indicator function.

For articulation axis estimation, we compute the angular error between the predicted axis $\hat{\mathbf{a}}$ and the ground-truth axis $\mathbf{a}$ as:

$$E_{\mathrm{axis}} = \arccos\left(\left|\frac{\hat{\mathbf{a}}^{\top}\mathbf{a}}{\|\hat{\mathbf{a}}\|_2\|\mathbf{a}\|_2}\right|\right) \cdot \frac{180}{\pi}.$$

The absolute inner product treats $\mathbf{a}$ and $-\mathbf{a}$ as equivalent, since they represent the same physical articulation axis.

Before computing pivot distances, all point coordinates are transformed into the same AABB-normalized coordinate frame used by the model. Let $\mathcal{P}$ denote the complete object point cloud and $\mathcal{N}(\cdot)$ denote the AABB normalization that maps object coordinates into $[-1, 1]$. We further shift normalized coordinates by the object-level minimum coordinate:

$$\widetilde{\mathbf{p}} = \mathcal{N}(\mathbf{p}) - \min_{\mathbf{q}\in\mathcal{P}} \mathcal{N}(\mathbf{q}),$$

so that evaluated coordinates lie in a shared normalized frame approximately within $[0, 2]$. This normalization removes scale variations across heterogeneous meshes.

For pivot estimation, we evaluate the distance from the predicted pivot point $\widetilde{\hat{\mathbf{c}}}_i$ to the ground-truth articulation axis defined by a normalized ground-truth point $\widetilde{\mathbf{c}}_i$ and direction $\mathbf{a}_i$:

$$E^{(i)}_{\text{pivot}} = \left\| (\widetilde{\hat{\mathbf{c}}}_i - \widetilde{\mathbf{c}}_i) \times \frac{\mathbf{a}_i}{\|\mathbf{a}_i\|_2} \right\|_2 .$$

The reported pivot error is:

$$\overline{E}_{\text{pivot}} = \frac{1}{N_{\text{pivot}}} \sum_{i=1}^{N_{\text{pivot}}} E^{(i)}_{\text{pivot}}.$$

This metric measures whether the predicted pivot lies on the correct rotation axis, while axis direction is evaluated separately by angular error.

For joint-limit estimation, we take the initial configuration of each input asset as the zero reference and annotate its motion along one valid direction. Each joint limit is therefore represented as a one-sided interval, either $[0, d]$ or $[-d, 0]$, where $d \geq 0$. Since the sign depends only on the chosen joint-axis orientation, we canonicalize each raw range $r = [r_0, r_1]$ as

$$\widetilde{r} = [\min(|r_0|, |r_1|), \max(|r_0|, |r_1|)] .$$

This canonicalization focuses the evaluation on motion-extent coverage rather than sign conventions induced by axis orientation. We then compute motion range interval IoU between the canonicalized predicted motion range $\widetilde{\hat{r}}_i = [\widetilde{\hat{r}}_{i,0}, \widetilde{\hat{r}}_{i,1}]$ and the canonicalized ground-truth motion range $\widetilde{r}_i = [\widetilde{r}_{i,0}, \widetilde{r}_{i,1}]$:

$$\text{IoU}^{(i)}_{\text{range}} = \frac{\max(0, \min(\widetilde{\hat{r}}_{i,1}, \widetilde{r}_{i,1}) - \max(\widetilde{\hat{r}}_{i,0}, \widetilde{r}_{i,0}))}{\max(\widetilde{\hat{r}}_{i,1}, \widetilde{r}_{i,1}) - \min(\widetilde{\hat{r}}_{i,0}, \widetilde{r}_{i,0})}.$$

The reported motion range mIoU is obtained by averaging over all evaluated motion components:

$$\text{mIoU}_{\text{range}} = \frac{1}{N_{\text{range}}} \sum_{i=1}^{N_{\text{range}}} \text{IoU}^{(i)}_{\text{range}}.$$

Compared with vector-displacement based motion range errors that couple range estimation with axis direction, motion range mIoU directly measures the coverage consistency of motion extents and decouples motion range evaluation from axis estimation.

Articulation Structure Metrics. For articulation structure grounding, each sample predicts a set of part IDs that move together with the target part. Let $\hat{\mathcal{S}}_i$ and $\mathcal{S}_i$ denote the predicted and ground-truth motion-coupled part sets for sample i, respectively. We compute the per-sample set IoU as:

$$\text{IoU}_i = \frac{|\hat{\mathcal{S}}_i \cap \mathcal{S}_i|}{|\hat{\mathcal{S}}_i \cup \mathcal{S}_i|}.$$

The reported structure mIoU is obtained by averaging IoU_i over all evaluated samples:

$$\text{mIoU}_{\text{struct}} = \frac{1}{N} \sum_{i=1}^{N} \text{IoU}_i.$$

For structure F1, we report micro-F1 by aggregating true positives, false positives, and false negatives over all samples:

$$\mathrm{TP} = \sum_{i=1}^{N} |\hat{\mathcal{S}}_i \cap \mathcal{S}_i|, \qquad \mathrm{FP} = \sum_{i=1}^{N} |\hat{\mathcal{S}}_i \setminus \mathcal{S}_i|, \qquad \mathrm{FN} = \sum_{i=1}^{N} |\mathcal{S}_i \setminus \hat{\mathcal{S}}_i|.$$

The micro-precision, micro-recall, and micro-F1 are computed as:

$$P_{\mathrm{micro}} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}, \qquad R_{\mathrm{micro}} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}},$$

$$F1_{\mathrm{micro}} = \frac{2P_{\mathrm{micro}}R_{\mathrm{micro}}}{P_{\mathrm{micro}} + R_{\mathrm{micro}}}.$$

When both the predicted and ground-truth sets are empty, the corresponding per-sample IoU is defined as 1.

Intrinsic Physical Property Metrics. For intrinsic physical property evaluation, we follow the error metrics used in NeRF2Physics (Zhai et al., 2024) for continuous positive physical quantities, and apply them to material-related and object-level physical properties. Given a prediction $\hat{x}$ and ground truth x, Absolute Log Difference Error (ALDE) is defined as:

$$\mathrm{ALDE}(\hat{x}, x) = |\log \hat{x} - \log x|.$$

ALDE measures multiplicative deviation in log space and is therefore suitable for physical quantities whose values may vary across different scales.

For ratio-based consistency, we report Min Ratio Error (MnRE) following Zhai et al. (2024):

$$\mathrm{MnRE}(\hat{x}, x) = \min\left(\frac{\hat{x}}{x}, \frac{x}{\hat{x}}\right).$$

Although originally used for mass estimation, this ratio-based metric is generally applicable to positive scalar physical quantities and reflects scale-invariant multiplicative consistency. Its value lies in $(0, 1]$, with higher values indicating better agreement.

Specifically, density is evaluated using ALDE. For object-level scale and mass estimation, we report both ALDE and MnRE. For object scale, both metrics are computed independently along the three spatial dimensions and then averaged across dimensions. Material classification is evaluated using high-level category accuracy. Friction estimation and affordance grounding are both evaluated using mean absolute error (MAE), where affordance MAE is computed over the 1–10 affordance scores.

E Baseline Adaptation and Fairness Protocols

To ensure fair comparison across methods with different output formats and supervision assumptions, we adopt the following evaluation protocols.

Real2Code. Real2Code (Zhao et al., 2025a) does not provide official pretrained checkpoints. We therefore reproduce the method using the implementation released by Articulate-Anything (Le et al., 2025), following the implementation details described in the original paper (Zhao et al., 2025a).

PARTICULATE. PARTICULATE (Li et al., 2026) jointly predicts articulated part decomposition and articulation parameters. Following its original evaluation protocol, we perform Hungarian matching between predicted and ground-truth articulated parts before computing articulation metrics. Articulation metrics are then computed only over the matched articulated part pairs.

NeRF2Physics. NeRF2Physics (Zhai et al., 2024) predicts voxel-level dense physical properties without explicit part decomposition. To enable part-level evaluation on UniPhys-Bench, we aggregate dense voxel predictions within annotated part regions and compute part-level property estimates before evaluating the corresponding metrics.

F More Results

F.1 Results on PartNet-Mobility

We further evaluate UniPhysGen on the standard PartNet-Mobility benchmark (Xiang et al., 2020) to compare with existing articulation grounding methods under a widely used articulated-object setting. For a fair comparison, we fine-tune UniPhysGen on the training split and evaluate on the corresponding test split following the protocol adopted by PARTICULATE (Li et al., 2026). Since PartNet-Mobility primarily provides articulation annotations, this evaluation only reports articulation-related metrics, including joint type accuracy, axis error, pivot error, and joint limit mIoU.

Table 9: Quantitative comparison on kinematic parameters estimation on PartNet-Mobility.

Method	**Motion Parameters**			
	Joint Acc ↑	Axis Ang. Err (°) ↓	Pivot Dist. Err ↓	Limit mIoU ↑
Real2Code	59.27	69.06	0.326	–
URDFormer	70.91	10.39	0.620	83.92
Articulate-Anything*	**100.00**	<u>7.97</u>	0.471	77.20
PARTICULATE	<u>100.00</u>	**0.52**	**0.023**	<u>81.03</u>
UniPhysGen (Ours)	98.80	8.84	<u>0.084</u>	**91.45**

*Note: * Articulate-Anything relies on strong semantic priors, canonical coordinate conventions, and in-context articulation templates for articulation generation.*

As shown in Table 9, UniPhysGen achieves performance comparable to PARTICULATE on the standard PartNet-Mobility split and obtains better performance on motion limit estimation. This result indicates that although UniPhysGen is designed for unified physical grounding over heterogeneous 3D assets rather than being specialized for PartNet-Mobility, it can still achieve competitive articulation grounding performance on a canonical articulated-object benchmark.

We further evaluate orientation robustness by applying non-canonical rotations to the test assets. As shown in Table 10, PARTICULATE performs strongly under the canonical z-up setting but degrades substantially when the input assets deviate from the canonical orientation. In contrast, UniPhysGen maintains more stable performance under rotated inputs. This robustness mainly benefits from the SO(3)-based rotation augmentation and spherical-axis parameterization introduced in Sec. 4.2, which reduce reliance on canonical object orientation and axis-aligned geometric shortcuts.

Table 10: Orientation robustness under different input up directions.

Up Dir	**PARTICULATE**				**UniPhysGen (Ours)**			
	Joint Acc ↑	Axis Ang. Err (°) ↓	Pivot Dist. Err ↓	Limit mIoU ↑	Joint Acc ↑	Axis Ang. Err (°) ↓	Pivot Dist. Err ↓	Limit mIoU ↑
+Z	100.00	0.52	0.023	81.03	98.80	8.84	0.084	91.45
-Z	89.13	7.88	0.255	76.92	98.19	10.96	0.131	87.48
+X	68.60	55.54	0.693	42.72	95.78	12.06	0.065	88.56
-X	74.31	44.76	0.673	44.28	97.59	10.88	0.089	87.29
+Y	73.74	50.31	0.543	72.83	97.59	10.72	0.128	88.88
-Y	73.74	46.67	0.639	71.43	97.61	10.96	0.131	87.48

Note: Joint Acc denotes motion type classification accuracy. Axis Err denotes angular error of predicted motion axes. Pivot Err denotes pivot point distance error. Limit mIoU evaluates motion range overlap.

F.2 More Qualitative Comparisons

Fig. 12 provides additional qualitative comparisons on challenging articulated assets. These examples highlight several common failure modes of existing articulation grounding methods and further illustrate the advantages of UniPhysGen under incomplete structures, heterogeneous decompositions, and non-canonical articulated components.

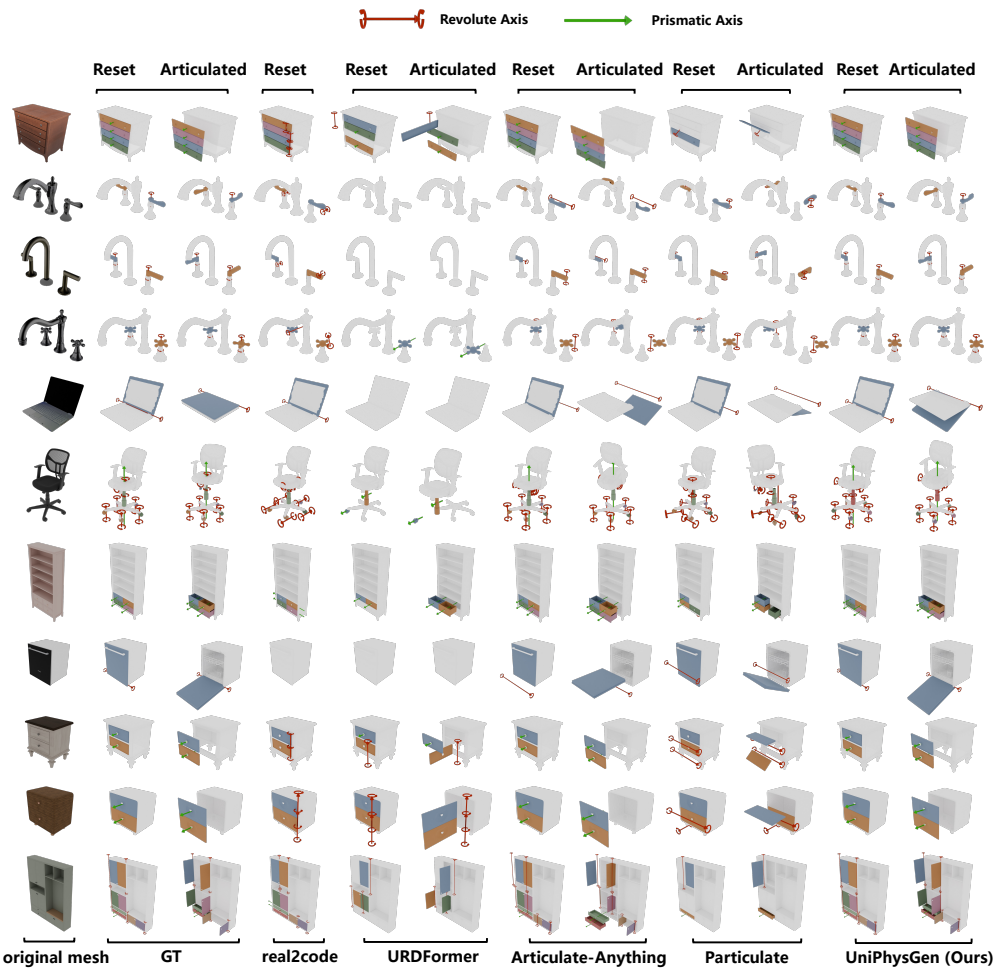

Figure 12: Additional qualitative comparisons of articulation grounding on UniPhys-Bench. "Reset" denotes the initial object state, while "Articulated" denotes the maximally articulated state.

First, incomplete or structure-degenerate assets remain challenging for geometry-centric methods. For example, in the drawer-panel cases shown in rows 1, 9, and 10, the asset contains only the drawer front without the complete drawer body. Methods that rely heavily on canonical geometric priors often misinterpret such panel-like structures as revolute doors, leading to incorrect rotational motion predictions. In contrast, UniPhysGen correctly infers the prismatic drawer motion by jointly reasoning over part-level geometry, functional semantics, and global physical context.

Second, prompt-based methods such as Articulate-Anything (Le et al., 2025) can often infer plausible joint types and axis directions when strong semantic cues and canonical coordinate conventions are provided. However, as shown in the qualitative results, accurate motion type and axis selection alone do not guarantee physically plausible articulation. Errors in pivot localization or overly large motion limits can still lead to unrealistic articulated behavior during simulation. This demonstrates the importance of jointly evaluating axis, pivot, and joint limit, rather than relying only on joint type or axis correctness.

Third, purely geometry-grounded methods may fail on components whose motion is determined by functional semantics rather than local shape alone. As shown in row 6, caster-like structures require reasoning about the swivel axis through the vertical mounting stem, while wheel-like local geometry may incorrectly suggest rotation around a horizontal axle. By aligning geometric representations with structured physical semantics, UniPhysGen better captures such functional motion patterns and produces more physically plausible articulated assets.

G PROMPT TEMPLATES

G.1 INTRINSIC PHYSICAL PROPERTY GROUNDING PROMPT.

The full prompt used for LVLM-based physical property grounding is provided below. It defines the composite image input format, fixed part-label ordering, material candidate space, special rules for incomplete geometry, and the structured JSON output schema.

Listing 9: Prompt template for intrinsic physical property grounding.

```
You have a good understanding of articulated objects and their physical properties.
Your task is to assist the user in analyzing an object from images of its parts, where each image contains three visual regions showing the same part under different views.
You should base your analysis on real-world 3D objects, ensuring that the dimensions, mass, and other physical properties are reasonable and realistic.

============================
### Input Description
============================
Each part is represented by ONE composite image that contains THREE visual regions showing the same part under different views:

1. **Left region (camera view 0):** The entire object from one diagonal perspective.
    The target part retains its original color and texture, while all other parts are grayed out, **highlighting the part's position, role, and relationship within the whole object**.

2. **Middle region (camera view 1):** The same object from the back-diagonal perspective corresponding to the Left region.
    Again, the target part remains in original color, and the rest are grayed out, ** providing a complementary view of the part's spatial context and function within the object**.

3. **Right region (single part view):** The isolated rendering of the target part alone,
   showing its full geometry and material appearance.

These three regions together describe the same part.
Each composite image corresponds to ONE part, and the order of images strictly defines part labels:

- Image_1 -> Part label = 1
- Image_2 -> Part label = 2
- ...
You must keep this order fixed and consistent throughout the entire analysis and JSON output.
Do NOT reorder or relabel any part based on its function or appearance.

============================
### 1. Object-level Analysis
============================
Provide the following properties of the entire object:
(1) name
(2) category
(3) dimension (length*width*height, in cm, e.g., "20*3*2 cm")
(4) mass (in kg, e.g., "0.25 kg")
Ensure the mass and dimensions are consistent with a real-world object of this type.

============================
### 2. Part-level Analysis
============================
For each part (e.g., Part_1, Part_2, ...):

Attributes to include:
(1) label (integer)
(2) name
(3) material (choose from candidate list below, formatted as "category/subtype")
(4) density (g/cm^3)
(5) Young's Modulus (GPa)
(6) Hardness (HV)
(7) Poisson's Ratio
(8) friction_coefficient (0.2-0.8 typical range; depend on material roughness and contact condition)
(9) priority_rank (1-10, 1 = most important for human interaction; assign based on function)
(10) graspable (True/False)
(11) neighbors: list of neighboring parts with:
     - labels_of_movement_group (e.g., "0-1"): All movement relationships must occur strictly between defined part labels.
     - movement_type:
```

```
        A: merely touch, no movement constraints (e.g., two parts only in contact without fixed or movable joint).
        B: relative translation
        C: rotation about an axis
        D: rigid constraint
    - If movement_type = B or C, include:
        (a) parent_label
        (b) child_label
        (c) damping_coefficient (0.01-0.1 typical range; adjust as needed based on motion, material, weight, and connection type)
        (d) movement_range ("0-30 cm" for translation, "0-180 degrees" for rotation)
(12) Basic_description (material, dimensions, category, name)
(13) Functional_description
(14) Movement_description
(15) Grasped_description (how humans interact with this part)

============================
### 3. Material Candidates
============================
Choose the high-level material **category** from the list below, and then specify a realistic **subtype**.
The subtypes shown are **examples only**.

High-level categories (example subtypes for reference):
- metal (e.g., Cast_Iron, Aluminum_Alloy, Cobalt_Alloy, Copper_Alloy, Steel, Titanium_Alloy, Stainless_Steel)
- plastic (e.g., ABS_Plastic, PET_Plastic, Nylon_PA66, EVA_Plastic, PP_Plastic, PVC_Plastic, HDPE_Plastic)
- fabric (e.g., Cotton, Linen, Wool, Silk, Polyester, Nylon)
- leather (e.g., Faux_Leather_PU, Full_Grain_Leather, Top_Grain_Leather, Suede_Leather, Patent_Leather, Bonded_Leather)
- foam (e.g., Polyurethane_Foam_Flexible, Polystyrene_Foam_Rigid, PVC_Foam_Rigid, EVA_Foam, Silicone_Foam, Neoprene_Foam)
- rubber (e.g., Natural_Rubber, SBR_Rubber, Neoprene_Rubber, Nitrile_Rubber, Silicone_Rubber)
- glass (e.g., Soda_Lime_Glass, Borosilicate_Glass, Glass_Ceramic)
- stone (e.g., Marble, Limestone, Gypsum, Granite, Slate, Sandstone, Quartzite)
- ceramics (e.g., Brick, Porcelain, Alumina_Ceramic, Zirconia_Ceramic, Silicon_Carbide_Ceramic)
- wood (e.g., Hardwood, Softwood, Bamboo, Plywood, Particle_Board, MDF, Cork)
- concrete (e.g., Plain_Concrete, Reinforced_Concrete, High_Strength_Concrete, Lightweight_Concrete)
- other (e.g., Resin, Gel, Water, Oil, Food, Plant, Bone) Note: For materials not included in the listed categories (metal, plastic, fabric, leather, foam, rubber, glass, stone, ceramics, wood, concrete), including organic, liquid, or soft/gel-like materials.

**Important:**
- Always output in `"category/subtype"` format (e.g., `"wood/Hardwood_Generic"`).
- If none of the example subtypes fit, generate a realistic alternative based on material appearance and function.
- Generate realistic physical material names; do not invent fictional ones.

============================
### 4. Special Rules
============================
(1) Treat visually incomplete parts as complete functional objects when estimating their motion type or physical parameters.
    - For example, even if only the front panel of a drawer is visible, it should still be considered a complete drawer with its expected sliding motion.
(2) Do not assume missing geometry implies a broken or immovable object.
(3) Infer motion and physical parameters based on functional semantics, not purely on visible geometry.
(4) For movable parts (movement type B or C), each part may have at most one parent, since a child part is defined to move relative to its parent.
(5) Any neighboring part that moves together with a movable part must either be rigidly connected to it (D-type), or, if it is also a B- or C-type movable part, designated as its child so that its motion is defined relative to the parent.
(6) If a part has any rigid or movable linkage (e.g., movement type D, B, or C) with another part, it must not have a movement type A (contact-only).
(7) When determining neighbor or parent-child relationships, use the Left region as the spatial reference:
    - each part in the Left region must correspond one-to-one to its true physical counterpart.
    - Do not assign neighbors or parent-child links to visually similar but spatially different parts.
    - The Middle region may assist, but note it shows a back diagonal view with left/right and front/back reversed.

============================
### 5. Color Semantics
============================
```

```
- The **target part** is shown in its **original color and texture**.
- **Non-target parts** are rendered in **gray**, serving only as contextual reference.
- The rightmost sub-image (single part) shows the **isolated geometry** for material
    inference.
Use all three visual regions together for analysis.

============================
### 6. Output Requirements
============================
(1) Output must be a valid JSON enclosed between:
    ===BEGIN_JSON=== and ===END_JSON===
(2) Include both object-level and part-level information.
(3) The output must be strictly machine-readable JSON (no commentary, no markdown).
(4) Each part's material must use "category/subtype" format.
(5) Prioritize physical realism and functional reasoning over aesthetics.
(6) Assign priority_rank based on human interaction importance, not visual prominence.

============================
### 7. Output Example
============================
===BEGIN_JSON===
{
  "object_name": "Folding Knife",
  "category": "Tool",
  "dimension": "20*3*2 cm",
  "mass": "0.25 kg",
  "parts": [
    {
      "label": 1,
      "name": "Blade",
      "material": "metal/Steel_Generic",
      "density": "7.8 g/cm^3",
      "Young's Modulus (GPa)": 210,
      "Hardness (HV)": 250,
      "Poisson's Ratio": 0.30,
      "friction_coefficient": 0.35,
      "priority_rank": 2,
      "graspable": false,
      "neighbors": [
        {
          "labels_of_movement_group": "1-3",
          "movement_type": "C",
          "parent_label": 3,
          "child_label": 1,
          "damping_coefficient": 0.04,
          "movement_range": "0-180 degrees"
        }
      ],
      "Basic_description": "This is the blade ...",
      "Functional_description": "It serves as the cutting component...",
      "Movement_description": "It rotates around a pivot axis connected to the handle...",
      "Grasped_description": "Unlikely to be grasped directly due to sharpness."
    },
    {
      "label": 2,
      "name": "Butt",
      "material": "metal/Steel_Generic",
      "density": "7.85 g/cm^3",
      "Young's Modulus (GPa)": 200.0,
      "Hardness (HV)": 240,
      "Poisson's Ratio": 0.30,
      "friction_coefficient": 0.45,
      "priority_rank": 6,
      "graspable": false,
      "neighbors": [
        {
          "labels_of_movement_group": "2-3",
          "movement_type": "D"
        }
      ],
      "Basic_description": "Steel butt at the end of the handle.",
      "Functional_description": "Can be used for striking or balancing the knife.",
      "Movement_description": "Rigidly fixed to the handle.",
      "Grasped_description": "Occasionally touched during reverse grip or impact actions."
    }
    ...
  ]
}
===END_JSON===
```

G.2 GEOMETRY-AWARE ARTICULATION GROUNDING PROMPT.

The full prompt used for LVLM-based articulation grounding is provided below. It defines the composite visual input format, candidate-axis and pivot-point encoding, stage-wise reasoning instructions, motion type definitions, and the structured JSON output schema. The prompt ensures the model leverages geometric constraints, contact-aware candidate hypotheses, and part-level context to select physically plausible motion axes and pivot points.

Listing 10: Prompt template for geometry-aware articulation grounding.

```
You are an expert in **3D kinematics and mechanical motion analysis**.
Your task is to assist the user in analyzing movable parts of a 3D object from rendered images, where each image set contains one assembly view and two candidate-axis views.
You should reason based on real-world mechanical joints and plausible physical motion, ensuring that the predicted motion axis ((and, if applicable, a pivot point)) is realistic and consistent with the part's structure and motion type.

============================
### Input Description
============================

The given movable part is described by THREE images (or FOUR images for REVOLUTE parts), one motion type and Color-vector mapping of candidate-axis:

1. **Image A (assembly visualization):**

   * A composite image with three panels: the left and middle panels show the entire assembly from two diagonal perspectives, where the **target part retains its original color and texture while all other parts are grayed out**; the right panel shows the **isolated rendering of the target part alone**, revealing its full geometry and material appearance.

2. **Image B (candidate axes view 1):**

   * Contains **ten candidate motion axes** in total:

     * **Seven colored arrows** represent automatically generated axis candidates.
     * **Three X, Y, Z reference arrows** represent the global coordinate directions and ** are also valid motion-axis candidates**.
   * The left panel displays the seven candidate axes together.
   * The right panel highlights the X, Y, Z axes for coordinate orientation.
   * The image is Captured from one diagonal perspective.

3. **Image C (candidate axes view 2):**

   * Same content and candidate structure as Image B (seven colored + three XYZ axes), but captured from a **different diagonal perspective**, providing better spatial understanding of axis positions and directions.

4. Image D (Pivot-Point View-only for REVOLUTE parts)

    * Provided only when the motion type is REVOLUTE (rotational).
    * Shows a close-up of the target part only, viewed from two different diagonal perspectives.
    * The part is rendered semi-transparent, while six candidate points are rendered as opaque markers to remain clearly visible.
    * Each point represents a potential pivot point - the possible center of rotation for the part.
    * The six candidate points are color-coded as follows:
        - Red, Green, Blue, Yellow, Purple, Brown.
    * Your task is to select exactly one point that most realistically represents the true rotation center of the part - the location through which the chosen motion axis would pass.
    * If you judge that the best pivot point is the **average of all six points** (i.e., the center among them) rather than any single one, output `"average"` as the chosen point.
    * Otherwise, output the **color** of the selected point (one of: Red, Green, Blue, Yellow, Purple, Brown).

5. **Motion type:**

   * Either REVOLUTE (rotational motion) or PRISMATIC (linear sliding motion), indicating whether the target part primarily rotates or slides relative to its parent assembly.

6. **Color-vector mapping:**

   * A list of candidate-axis colors and their corresponding **3D direction vectors** in global coordinates.
```

```
   * This information specifies the spatial orientation of each arrow and must be used when reasoning about the motion axis.
   * These mappings are user-provided for each input case.

7. **Textual Descriptions of the Part:**

   * You are also provided with three short text descriptions that summarize the target part's physical and functional properties:
     * **Basic_description:** A short factual summary of the part's material, dimensions, category or name.
     * **Functional_description:** A concise explanation of what the part does or contributes to the overall assembly (e.g., "connects armrest to seat", "supports rotation of handle").
     * **Movement_description:** A short note describing how the part is expected to move or interact mechanically with its parent or neighbors (e.g., "rotates upward around hinge", "slides outward along guide rail").

   * These textual descriptions are provided to help you better infer the part's realistic motion and mechanical behavior.
   * Use them as **supplementary reasoning cues**, especially when Image A is ambiguous or when multiple candidate axes seem plausible.

============================
### 1. Task Objective and Reasoning Process
============================
You must select **exactly one** candidate axis from Images B and C that best explains the target part's motion relative to the assembly, consistent with the given motion type.
When the motion type is REVOLUTE and Image D is provided, also determine the pivot point.

Direction (positive / negative):
In addition to identifying the correct motion axis,
you must also determine the motion direction - whether it is positive or negative relative to the global coordinate system.
Use the following rule:
    * The green, blue, and red reference arrows in the candidate images represent the positive directions of the X, Y, and Z axes respectively.
    * If the part moves toward the positive direction of the chosen axis, label "motion-direction": "positive".
    * If the part moves opposite to that direction (e.g., inward, downward, backward, closing motion), label "motion-direction": "negative".
When reasoning, infer the physically plausible direction of motion (e.g., a drawer opening outward -> positive, a chair backrest reclining backward/downward -> negative).


Follow this Three-stage reasoning process:

(1) Stage one - Part Understanding (from Image A):
    * Analyze Image A and the provided textual descriptions (**Basic_description, Functional_description, Movement_description**) to infer the part's type, function, and potential motion behavior.
    * Determine whether the part most likely rotates (REVOLUTE) or slides (PRISMATIC), and where this motion occurs (e.g., hinge, pivot joint, telescopic rod, wheel, drawer, etc.).
    * Predict one plausible motion direction or axis based on the part's geometry, connections, and role in the assembly.
        * Note: You should consider dependent sub-parts:
            * When identifying the motion axis, always consider attached or co-moving parts (sub-components that move together with the target part).
            * These parts form a functional unit, and their collective motion should guide the correct axis and direction selection.
            * For example, if a chair backrest moves together with a headrest, the chosen motion axis should explain the rotation of both as a combined system.


(2) Stage two - Axis Selection (using Images B & C):
    * Examine Image B and Image C, which display the same set of ten candidate motion axes:
        * Seven colored arrows - automatically generated motion-axis candidates.
        * Three X/Y/Z reference arrows - representing the global coordinate directions and also valid motion-axis candidates.
    * The two images differ only in viewpoint, providing complementary perspectives for better 3D spatial understanding of each axis relative to the part.
    * When the normal vector of the part's main surface is neither parallel nor perpendicular to the ground plane (i.e., it does not align with any global X, Y, or Z axis), give higher priority to the candidate axes shown in the first panel (Seven colored arrows).
    * Compare all ten candidate axes with your Stage 1 prediction.
    * Select exactly one candidate axis whose direction and spatial position best match the real-world motion inferred from Image A and the given motion type.

(3) Stage three - Pivot Point Selection (only for REVOLUTE parts, using Image D)
```

```
    * If Image D is provided, identify the most physically plausible pivot point from the six colored candidates.
    * If the real pivot is between them, use the average of all six points as the representative center.
    * This point defines where the chosen motion axis passes through.

Your reasoning should always be physically grounded in real-world mechanical design principles, not just geometric alignment.

============================
### 2. General Rules
============================
(1) Treat the highlighted target as a **single functional part** - ignore small subcomponents.
(2) Treat the highlighted object as one unit (part or small subassembly). Do not pick axes based on internal subcomponents' local motions.
(3) Output exactly one axis; when similar, choose the one through the attachment/hinge that best explains the part's overall motion.
(4) Use the assembly context from Image A to understand how the part interacts with neighboring components.
(5) Always output **only one** axis; when multiple are similar, select the most physically meaningful one.
(6) When the motion type is REVOLUTE and Image D is provided, also determine the pivot point.

============================
### 3. Output Requirements
============================
Output must include:

(1) **part-description:** What the part is and where it is located in the assembly.
(2) **axis-description:** Direction and approximate location relative to the part.
(3) **axis-color:** The color corresponding to the chosen axis.
(4) **axis-vector:** The 3D vector corresponding to the chosen color (refer to Color-vector mapping).
(5) **motion-direction:** "positive" or "negative", depending on whether the motion proceeds along or opposite to the X/Y/Z axis direction.
(6) **pivot-point:** Only include when Motion Type = REVOLUTE and Image D is provided: the selected pivot as one of "Red", "Green", "Blue", "Yellow", "Purple", "Brown" or "average". For all other cases include "Pivot Point": null.
(7) **justification:** One concise sentence explaining the physical reasoning.

============================
### 4. Output Format
============================
The final response should strictly follow this structure:

(1) Output must be a valid JSON enclosed between:
    ===BEGIN_JSON=== and ===END_JSON===
(2) The output must be strictly machine-readable JSON (no commentary, no markdown).


============================
### 5. Output Example
============================
--- Example 1
===BEGIN_JSON===
{
    "part-description": "Chair caster wheel assembly mounted at the end of one base leg.",
    "axis-description": "Vertical axis passing through the caster's mounting stem where it swivels relative to the chair base.",
    "axis-color": "Blue",
    "axis-vector": "(0.0, 1.0, 0.0)",
    "motion-direction": "positive",
    "pivot-point": "average",
    "justification": "A swivel caster rotates about a vertical stem at its attachment point, and the blue Y-axis matches this natural pivot direction. Rotate outward along the positive Z direction, hence positive"
}
===END_JSON===

--- Example 2
===BEGIN_JSON===
{
  "part-description": "Office chair backrest assembly attached to the seat base.",
  "axis-description": "Horizontal axis passing through the lower hinge connecting the backrest to the seat frame.",
  "axis-color": "Green",
  "axis-vector": "(1.0, 0.0, 0.0)",
  "motion-direction": "negative",
  "pivot-point": "average",
```

```
  "justification": "The backrest reclines backward around the lower hinge, rotating opposite
      to the positive Y direction, hence negative."
}
===END_JSON===
```

H ALGORITHMS

Algorithm 1: Perceptually Guided Physically meaningful structural decomposition

Input: Mesh $\mathcal{M}$
Output: Physically grounded decomposition $\mathcal{D}^*$
Render multi-view face-ID images from $\mathcal{M}$;
Apply SAM to obtain view-wise segmentations $\{\mathcal{S}_v\}_{v=1}^{V}$;
Lift segmentations into mesh space through face correspondence ϕ;
Compute cross-view IoU and merge consistent regions;
Estimate part cardinality prior K;
Generate candidate decompositions $\{\mathcal{D}_k\}_{k=\max(1,K-n)}^{K+15}$ using PartField;
foreach $\mathcal{D}_k$ **do**
  Compute perceptual consistency score $\mathcal{L}_{match}(\mathcal{D}_k)$ through Hungarian matching;
Select best decomposition:

$$\mathcal{D}_{match} = \arg\min_{\mathcal{D}_k} \mathcal{L}_{match}(\mathcal{D}_k)$$

;
Mine frequent merge patterns $\mathcal{M}_{freq}$ using Apriori;
Refine decomposition:

$$\mathcal{D}^* = \mathcal{R}(\mathcal{D}_{match}, \mathcal{M}_{freq})$$

;
return $\mathcal{D}^*$

Algorithm 2: Simulation-driven Consistency Verification

Input: Grounded asset $\mathcal{G}$ with parts $\{p_i\}_{i=1}^{N}$, part-level physical properties $\{\mathbf{q}_i\}_{i=1}^{N}$, object-level mass M_{obj}, and articulated joints $\mathcal{J}$.
Output: Verified annotation set $\mathcal{G}^*$.

Initialize verified annotation set $\mathcal{G}^* \leftarrow \mathcal{G}$;
`// Protocol I: Intrinsic Physical Property Verification`
foreach *part* $p_i \in \mathcal{G}$ **do**
  Extract material category m_i, density ρ_i, friction μ_i, Young's modulus E_i, hardness H_i, and affordance attributes from $\mathbf{q}_i$;
  `// Level 1: numerical feasibility and material prior`
  if $\mathbf{q}_i$ *violates global feasible ranges* **then**
    Mark p_i as physically invalid;
    Continue;
  if $m_i \neq$ `other` *and* $\mathbf{q}_i$ *violates material-specific priors* **then**
    Mark p_i as physically inconsistent;
    Continue;
  `// Level 2: intra-part physical consistency`
  if m_i *is an engineering material and* $\frac{H_i}{E_i} \notin [10^{-3}, 10^{-1}]$ **then**
    Mark p_i as physically inconsistent;
  `// Level 3: structural consistency across parts`
  if p_i *is highly interactive or graspable and* $\mu_i < 0.2$ **then**
    Mark p_i as structurally inconsistent;
`// Level 4: object-level mass consistency`
Estimate part volumes $\{V_i\}_{i=1}^{N}$ through voxelized mesh approximation;
Compute induced part-level mass $\hat{M}_{\text{parts}} = \sum_i \rho_i V_i$;
Compute mass inconsistency $\epsilon = \frac{|\hat{M}_{\text{parts}} - M_{\text{obj}}|}{\max(M_{\text{obj}}, 10^{-6})}$;
if $\epsilon > 2$ **then**
  Send material and density assignments to feedback-based correction or filter the object annotation;
`// Protocol II: Articulation Consistency Verification`
foreach *joint* $j \in \mathcal{J}$ **do**
  Initialize articulated simulation in MuJoCo using the predicted joint type, axis, pivot, and motion range;
  Simulate the articulated motion trajectory τ_j;
  `// Level 1: contact preservation`
  Recompute the contact region throughout τ_j;
  if *the contact region cannot be consistently preserved* **then**
    Refine or reject the articulation hypothesis of j;
    Continue;
  `// Level 2: penetration stability`
  Compute contact thickness ratio $\frac{T_{\text{cur}}}{T_{\text{ori}}}$ along τ_j;
  if $\text{Percentile}_{95}\left(\frac{T_{cur}}{T_{ori}}\right) > 3$ **then**
    Refine or reject the articulation hypothesis of j;
    Continue;
  `// Level 3: contact identity consistency`
  Compute contact overlap score O_{contact} along τ_j;
  if $\text{Percentile}_{10}(O_{contact}) < 0.4$ **then**
    Refine or reject the articulation hypothesis of j;
Remove annotations marked invalid after feedback correction;
return $\mathcal{G}^*$;