ASSET2SIM: AUTOMATICALLY IMPROVING SIMULATION PHYSICS OF ARTICULATED OBJECTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Approximate physics simulators have emerged as a key resource for training and evaluating robotic policies in simulation, especially for locomotion and navigation capabilities. A key bottleneck to realizing their utility for *manipulation* tasks is the dearth of assets (objects and environments) compared to the real world. Today's largest repositories of object assets focus on rigid objects offering limited affordances for manipulation beyond global repositioning. On the other hand, *articulated* objects which afford more nuanced manipulation (e.g., staplers, trashcans, cabinets, microwaves, folding knives) remain largely absent from simulators. Widely available curated libraries of articulated assets are limited: (1) they are either geometry-only, lacking annotations for physics properties like link inertias and joint frictions, or (2) they cover a limited number of assets designed with extensive manual effort and still have noisy physics annotations, as we show. We present ASSET2SIM, an autonomous vision-language model-guided procedure to improve the physics properties of articulated object assets for *sim-readiness*. ASSET2SIM utilizes multiple render views, geometric data, semantic annotations, and volumetric properties to iteratively refine estimates of an articulated asset's physics properties and initial states. We also develop an evaluation protocol to thoroughly validate the physics fidelity, RL training-readiness, and initialization stability of ASSET2SIM assets in robot manipulation tasks, establishing over $2.12\times$ higher mean sim-to-real correlation and over $2.14\times$ higher average policy training success rate over the next-best baselines. We will release the ASSET2SIM-X library of 1892 *sim-ready* articulated object assets spanning 44 categories, as well as our code to enable scaling such repositories further in the future.

1 INTRODUCTION

Many recent advances in robotics (Xiang et al., 2020; Wang et al., 2023b;a; Puig et al., 2023; Mu et al., 2021; Kim et al., 2026; Nasiriany et al., 2024) have been driven by the emergence of fast, approximate physics simulators for training and evaluating robots. For simulators targeting manipulation, object assets are typically drawn from large-scale 3D object datasets like PartNet (Mo et al., 2019), which accurately capture object geometries and visual textures.

However, as Jean Piaget would say, "to know an object is to act on it" (Piaget, 1972). An object is more than just its geometry and appearance; it is the set of potential interactions that it affords. These other properties are especially difficult to capture for *articulated* objects i.e., objects comprising multiple rigid components connected via joints Liu et al. (2025), which correspondingly have many degrees of freedom, and nuanced, task-specific affordances. While such objects are ubiquitous in the real world—think of keyboards, glasses, laptops, water bottles, pens, scissors, staplers—they remain largely absent from our simulators, because today's largest articulated asset libraries such as PartNet-Mobility (Xiang et al., 2020) lack reliable physics annotations. They focus on capturing skeletal structures (joints and link geometries), but neglect properties such as link inertias, joint frictions, stiffness, damping, global object scale, and material properties, which are critical to *sim-readiness*.

To fix this, several recent works (Le et al., 2024; Li et al., 2025; Cao et al., 2025) have targeted autonomous generation of sim-ready assets from multimodal observations to scale up today's libraries. However, as we will show, these methods often inherit the weaknesses of today's asset libraries: they too mainly capture skeletal structures, and neglect physics fidelity: to take an example from our

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.

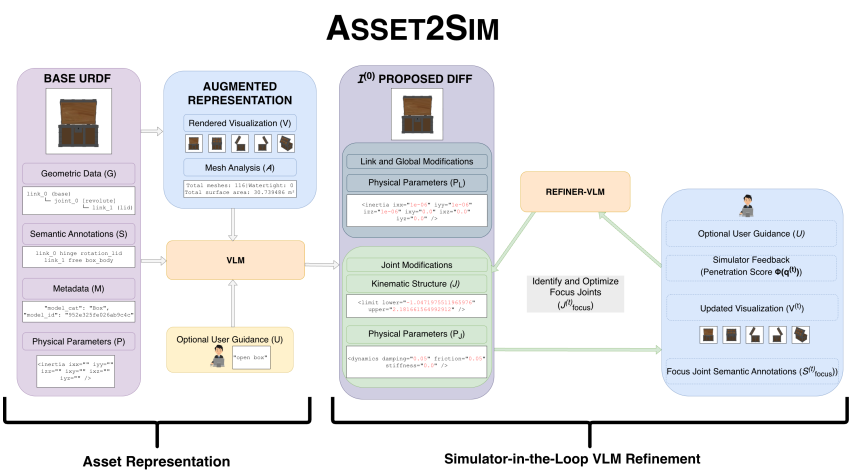


Figure 1: ASSET2SIM is an interaction-ready asset creation pipeline with three distinct stages: 1) augmentation of asset representations, 2) VLM property inference to construct an initial URDF "diff", and 3) simulator-in-the-loop refinement for reducing penetration.

experiments, an open cabinet drawer with incorrect settings for friction, damping, and inertia can behave in unrealistic ways, such as oscillating back-and-forth instead of sliding shut when force is applied. As a result, researchers today spend significant manual effort to tune the physics properties of these assets (Hua et al., 2024; Wu et al., 2026)—even expert artists reportedly spend between 12 minutes and 2 hours to tune each asset for physical *plausibility* (Jin et al., 2025). In our experience, researchers spend much longer to achieve simulation physics good enough for robotic use cases.

We propose ASSET2SIM, an approach for automatically improving the simulation physics of a base articulated object asset, such as one drawn from PartNet-Mobility. ASSET2SIM iteratively edits and evaluates physics properties and task-specific initial states / configurations of these base assets, to generate sim-ready assets with minimal human intervention. To achieve this, ASSET2SIM builds on the successes of Vision-Language Models (VLMs) in understanding material properties, physical relationships, and object semantics from multi-modal inputs (Luo et al., 2023; Kabra et al., 2023): it performs VLM-guided search to jointly optimize asset properties using physics-based feedback from simulators. ASSET2SIM can optionally target a given robotic manipulation task, enabling it to generate an interaction-ready asset specifically for that task, unlocking the ability to construct robot manipulation simulation environments containing physics-ready articulated assets at scale. To our knowledge, this is the first pipeline for using VLMs to develop sim-ready articulated objects with realistic physics parameters that can be deployed directly into robot manipulation simulations. Next, we address a critical bottleneck in the systematic study of this important problem: evaluating the "interaction-readiness" of an object asset in a scalable manner is itself non-trivial. We target this lacuna by proposing an extensive multi-pronged evaluation protocol for the realism and usefulness of our generated assets, spanning physics fidelity, RL training-readiness, and initialization stability.

Contributions. Our main contributions are:

(1) Automated Asset Simulation Physics Improvement: We propose ASSET2SIM, a novel VLM-based system for transforming base articulated objects into sim-ready assets. We release ASSET2SIM-X an asset library of 1892 ASSET2SIM-refined assets across 44 categories, building on top of PartNet-Mobility (Xiang et al., 2020), one of the largest existing articulated asset libraries.

(2) Sim-Readiness Evaluation Protocol: We present a first-of-its-kind comprehensive asset evaluation protocol that examines whether an articulated asset is truly simulation-ready with appropriate physics parameters. We study its performance across three stages: 1) physical fidelity, where we have humans evaluate the realism of teleoperated interactions with the asset and a VLM judge evaluate the realism of a video of Vision-Language-Action model (VLA)-operated robot-asset interaction, 2) a training-readiness evaluation, where we train policies to articulate the asset, and 3) an initialization stability evaluation, where we load an asset into the physics simulator and observe whether it changes in position or orientation without any external stimuli.

Related work. Several large-scale 3D object datasets have initially provided rigid object geometries for use in environment creation (Chang et al., 2015; Mo et al., 2019), subsequent efforts have extended limited subsets of object categories to include articulated assets (Geng et al., 2022; Xiang et al., 2020). More recently, automated articulated asset modeling approaches have leveraged these articulated asset datasets both for training specialized models as well as retrieval pipelines involving VLMs for real-to-sim articulated asset generation (Li et al., 2025; Le et al., 2024; Cao et al., 2025; Wu et al., 2026). However, both available benchmarks and articulated-asset pipelines focus on scaling link placement and joint prediction on 3D object meshes. ASSET2SIM focuses on improving the simulation physics of articulated assets, and in optionally tailoring the physical properties to specific tasks based on user guidance. We discuss available datasets, articulated asset creation methods, and related physics-property-estimation techniques in Appendix A.

2 AUTOMATICALLY IMPROVING SIMULATION PHYSICS OF ARTICULATED ASSETS

We now describe our approach for systematically transforming incomplete asset specifications in Unified Robot Description Format (URDF) files into complete, physically plausible models that can be imported into a standard physics simulator. We first provide a description of a complete, sim-ready ASSET2SIM-articulated asset in Sec 2.1. We then describe asset representations leveraged by ASSET2SIM to refine incomplete URDFs in Sec 2.2 and how the VLM initially updates the asset in Sec 2.3. Finally, we discuss ASSET2SIM's simulator-in-the-loop refinement pipeline in Sec 2.4. We provide example URDFs in Appendix C.

2.1 SIM-READY ASSET DESCRIPTION

What constitutes a sim-ready articulated asset? We represent an asset as $\mathcal{I}^* = \{M, G, P, S\}$ including metadata M, geometric data G, physics data P, and semantic annotations S:

- **Metadata** M contains the URDF name and object category (e.g. dispenser, bag, etc.).
- **Geometric data** $G = \{L, J, \mathcal{M}, \mathcal{G}\}$ consists of 1) link data $L = \{l_1, \ldots, l_n\}$, where each l_i contains the position data and the visual and collision mesh references associated with the link, 2) kinematic structure $J = \{j_1, \ldots, j_m\}$ where each j_r contains joint types τ_r, initial position and orientation data θ_r, limits $[\theta_r^{\min}, \theta_r^{\max}]$, axis specifications, and parent-child relationships associated with the joint, 3) mesh geometries $\mathcal{M} = \{\mu_1, \ldots, \mu_n\}$, and 4) global asset properties $\mathcal{G}$, including scale (λ) and material properties (e.g. restitution). Each mesh $\mu_i \in \mathcal{M}$ corresponds to a link $l_i \in L$. Joint types τ_r are chosen from $\{\text{revolute}, \text{prismatic}, \text{continuous}, \text{fixed}\}$ with limits $[\theta_r^{\min}, \theta_r^{\max}]$ defined for revolute and prismatic joints.
- **Physics parameters** $P = \{\mathbf{P}_L, \mathbf{P}_J\}$. For each link $l_i \in L$, we utilize its mass m_i and inertia tensor $\mathbf{I}_i$, to construct the link physics property vector $\mathbf{P}_L = \{(m_1, \mathbf{I}_1), \ldots, (m_n, \mathbf{I}_n)\}$. For each joint $j_r \in J$, we also retrieve any friction μ_r, damping β_r, and stiffness k_r to construct the joint physics property vector $\mathbf{P}_J = \{(\mu_1, \beta_1, k_1), \ldots, (\mu_m, \beta_m, k_m)\}$.
- **Semantic annotations** $S = \{s_1, \ldots, s_n\}$ are extracted from the URDF's semantics file, where each annotation s_i maps link name l_i to a semantic label (e.g., "base", "lid", "hinge", "drawer") and a functional joint type (e.g., "hinge", "slider", "free").

As motivated in Sec 1, most existing libraries of articulated objects have *incomplete* or noisy assets. For example, most PartNet-Mobility objects miss most of P, and initial configurations θ_r are also unreliable.

2.2 AUGMENTED ASSET REPRESENTATIONS

To aid the VLM in refining this asset and producing a complete representation $\mathcal{I}^*$, we augment the incomplete representation $\mathcal{I}$ with the following two asset representations: multi-view renders V and mesh analysis $\mathcal{A}$.

- **Multi-view visual rendering** generates a set of images $V = \{\nu_1, \ldots, \nu_K\}$ from K distinct viewpoints in a renderer. We render from canonical viewpoints: front, back, left, right, and

perspective projections, ensuring comprehensive visual coverage of the object's geometry and surface properties.

- **Mesh analysis** $\mathcal{A}$ computes volumetric properties for each mesh file in the object's textured mesh directory. For each mesh file, we use trimesh (Trimesh Authors., 2019) to load the geometry and compute: volume v_i (if the mesh is watertight), surface area a_i, bounding box dimensions $\mathbf{b}_i = (w_i, h_i, d_i)$, and center of mass $\mathbf{c}_i$ (for watertight meshes). If meshes are not watertight, we use bounding box estimates for each mesh.

2.3 VLM-Generated Initial Asset Update

After constructing the augmented articulated asset representation, we prompt the VLM with asset representation $\mathcal{I}_{VLM} = \mathcal{I} \cup \{V, \mathcal{A}\}$ and optional user guidance U, which comprises a few sentences describing any desired task-specific asset behavior (see Appendix M) for the prompt template, and Appendix H) for all user guidance). The VLM generates a structured "diff" (referred to as an overlay in the prompt) which is merged with $\mathcal{I}_{VLM}$ to construct an initial URDF $\mathcal{I}^{(0)}$. We validate the $\mathcal{I}^{(0)}$'s link property estimates using standard rigid-body feasibility checks (symmetric, positive definite, and consistent principal moments), and optionally clamp it to conservative bounds implied by the link's bounding box. If the tensor is invalid, we apply a minimal correction to obtain a physically plausible inertia. After validation, we iteratively refine $\mathcal{I}^{(0)}$ using our simulator-in-the-loop process.

2.4 Simulation-Guided Iterative Refinement of Initial Configuration

Given the representation $\mathcal{I}$ and user guidance U, a one-shot VLM can propose a target joint configuration $\mathbf{q}_{\text{VLM}}$; in practice, mesh or kinematic imperfections can make this state self-interpenetrating, which destabilizes physics simulation. We introduce a simulation-in-the-loop state refinement stage to optimize the joint configuration to reduce penetration while preserving the intended semantic state.

Let the articulated object have d active joints with configuration $\mathbf{q} \in \mathbb{R}^d$ and joint limits $\{[\theta_r^{\min}, \theta_r^{\max}]\}_{r=1}^d$. We define the feasible joint space $\mathcal{Q} = \prod_{r=1}^d [\theta_r^{\min}, \theta_r^{\max}]$ and initialize $\mathbf{q}^{(0)} = \Pi_{\mathcal{Q}}(\mathbf{q}_{\text{VLM}})$. At each iteration t, we update the URDF's joints in the physical simulator and compute a penetration score $\Phi(\mathbf{q}^{(t)})$ by summing penetration depths over all detected contacts (zero if separated) (see Equation 1 for a precise definition of the penetration score). We designate a configuration as collision-free if the total penetration is below a small tolerance threshold ε.

For colliding configurations, we localize corrective updates as follows: we group penetrations by colliding link pairs, and then map the most severe collisions to a small set of focus joints $J_{\text{focus}}^{(t)}$ using the URDF kinematic tree and available part semantics. This localization makes updates efficient and helps preserve the user-specified semantic configuration. Intuitively, we want to reduce penetration without drifting far from $\mathbf{q}_{\text{VLM}}$; otherwise, minimizing penetration alone can converge to a qualitatively different state that does not meet the user's specification. This refinement attempts to approximate the optimal configuration $\mathbf{q}^* = \arg\min_{\mathbf{q} \in \mathcal{Q}} \Phi(\mathbf{q}) + \lambda \|\mathbf{q} - \mathbf{q}_{\text{VLM}}\|_2^2$ that balances these two objectives, though we do not solve this objective explicitly.

We run an iterative refinement loop as follows: at iteration $t+1$, we prompt the VLM to update $J_{\text{focus}}^{(t)}$ while providing it with the kinematic information in $J_{\text{focus}}^{(t)}$, the original user request U, focus joint semantic annotations $S_{\text{focus}}^{(t)}$, updated multiview renders $V^{(t)}$, and current penetration score $\Phi(\mathbf{q}^{(t)})$ in its context. We project the candidate update back into the feasible joint space and accept it only if it decreases penetration. The loop terminates once the penetration score falls below ε or when a maximum iteration budget is reached, and we return the lowest-penetration configuration encountered to form the sim-ready representation $\mathcal{I}^*$. If residual penetration cannot be fully eliminated, we flag the asset for optional self-collision disabling in downstream simulation while preserving the visually correct configuration.

3 A Novel Protocol For Evaluating *Sim-Readiness*

As described above, ASSET2SIM is designed to generate a complete interactable articulated asset with requisite physical properties to import into standard physics simulators, starting from an incomplete

and potentially noisy URDF file the likes of which are easy to procure from large object datasets such as PartNet-Mobility. Due to the difficulty of evaluating the quality of such interactable assets, we propose a protocol and scoring system to evaluate interaction-readiness of an articulated object asset.

3.1 Measuring Initialization Stability

Default initial articulated asset configurations often exhibit unrealistic behavior and are unstable when imported into a simulator (examples in Appendix F, and video examples in the supplementary material). To measure this, we propose the following approach to measure a set of complementary stability scores for determining if the imported asset is in a stable configuration prior to any robot interactions.

Penetration Depth. We quantify mesh interpenetration by aggregating penetration depths reported by the physics engine at contact points in the same manner as done in our simulator-in-the-loop refinement stage (see Section 2.4. For a joint configuration $\mathbf{q}$, we load the articulated asset and step through the simulator to obtain the set of contact manifolds $\mathcal{C}(\mathbf{q})$. Each contact point $p \in \mathcal{C}(\mathbf{q})$ provides a signed separation $s(p)$, with $s(p) < 0$ indicating penetration. The *penetration score*

$$\Phi(\mathbf{q}) \;=\; \sum_{p \in \mathcal{C}(\mathbf{q})} \max(0,\, -s(p)) \tag{1}$$

is the total penetration depth summed over all penetrating contact points. Larger values of $\Phi(\mathbf{q})$ indicate more severe interpenetration, which typically leads to unstable simulation due to large corrective contact impulses. Note that this score does not involve switching on any physics in the environment. It is a property purely of the object snapshot in its initial configuration.

Position/Orientation Stability. Let $\mathbf{q}_0$ denotes the *initial joint configuration* used to instantiate the articulated asset at reset. Note that the subscript t here refers to time measured once the final refined asset is loaded into the simulator, unlike the superscript (t) in Section 2.4 which refers to the iteration of the simulator-in-the-loop refinement process. Starting from $\mathbf{q}_0$, we run the simulator for a settling period T_{set} to allow contacts and passive joints to reach a static equilibrium. We then record the root pose of the object as a reference state $(\mathbf{x}_{\text{ref}}, \mathbf{R}_{\text{ref}})$, and continue simulating for an additional horizon T_{test} with no external actuation (zero actions). Let $(\mathbf{x}_t, \mathbf{R}_t)$ be the root position and orientation during this evaluation window. We define the stability deviations as the maximum drift from the reference:

$$D_{\text{pos}}(\mathbf{q}_0) = \max_{t \in [0, T_{\text{test}}]} \|\mathbf{x}_t - \mathbf{x}_{\text{ref}}\|_2 \,, \tag{2}$$

$$D_{\text{ori}}(\mathbf{q}_0) = \max_{t \in [0, T_{\text{test}}]} d(\mathbf{R}_t, \mathbf{R}_{\text{ref}}) \,. \tag{3}$$

An asset *passes* the stability test if $D_{\text{pos}}(\mathbf{q}_0) \le 10^{-3}\,\text{m}$ and $D_{\text{ori}}(\mathbf{q}_0) \le 10^{-2}\,\text{rad}$. Low values indicate that the asset remains numerically stable and does not exhibit spurious drift after initialization.

Joint Oscillation. During the evaluation window after settling from the initial configuration $\mathbf{q}_0$, we record joint trajectories $\mathbf{q}_{j,t}$ for each joint j. We characterize oscillatory behavior using (i) the motion amplitude $A_j = \max_t \mathbf{q}_{j,t} - \min_t \mathbf{q}_{j,t}$ and (ii) direction reversals in discrete joint velocities $\Delta\mathbf{q}_{j,t} = \mathbf{q}_{j,t} - \mathbf{q}_{j,t-1}$. An asset is *oscillatory* if any joint j that exhibits at least three sign changes in $\Delta\mathbf{q}_{j,t}$ and has amplitude $A_j > \epsilon$ for a threshold ϵ. We treat the presence of any such oscillation as a failure, since it often reflects underdamped or poorly tuned joint dynamics and unstable contact resolution that degrades simulation fidelity and complicates downstream policy learning.

3.2 Measuring Physics Fidelity with VLAs and Humans

Next, we propose to evaluate physics fidelity as judged by VLAs and humans. First, with the VLA, we measure *task-relevant physics fidelity*, by comparing a VLA's task performance on the simulated object with its performance on a real object on tasks involving 1) grasping and lifting the articulated asset and 2) articulating an asset joint. We evaluate two properties of VLA interactions with various assets in simulation: the sim-to-real performance correlation of the asset simulation and the physical realism of the rendered VLA interaction. For each VLA, we describe its simulation success rate using the random variable X_{sim} and its real-world success rate using the random variable X_{real}. For each sampled object we evaluate, we record a measurement for X_{sim} by deploying the VLA in simulation

5 times, and measure the percentage of deployments where the VLA successfully exhibits the desired behavior through visual observation. We similarly measure X_{real} by deploying the same VLA on a Franka Panda arm in the real world to execute the task on a similar real-world object 5 times. We first compute the sim-to-real correlation coefficient (SRCC) (Kadian et al., 2020; Aljalbout et al., 2025) for X_{sim} and X_{real} using these recorded measurements, with statistics recorded in Table 1. Then, we evaluate the physical realism of the simulated VLA interactions by using VLM-as-a-judge to compare videos of the VLA-operated robot interacting with assets in simulation and in the real world, and assign scores from 0-5, as done in previous work using VLM-as-a-Judge for evaluating physical realism (Le et al., 2025). The prompts used for describing the task to the VLA are in E and VLM evaluation prompts are in Appendix G.

Next, we measure subjective *human play fidelity*: human subjects teleoperate a robot for undirected interactions with objects in simulation, and rate asset realism. We recruit 5 independent evaluators unaffiliated with this project. For each object, the 4 URDF variants (1 per method) are presented in randomized, blind order. Evaluators do not know which baseline they are interacting with. For each object, they first observe scale, pose, and plausibility, then wait and check the object stability. They are then asked to interact with the object by touching, pushing, lifting, and articulating joints. After interacting with all 4 URDFs, the evaluators are asked to assign each URDF a score from 0-10. We present our results in Table 3.

3.3 Measuring Policy Training-Readiness

Can robotic policies for manipulating these objects train stably? Given an object asset, we instantiate an Isaac Lab environment for one canonical task that exercises the articulation affordance of that object, and train an online, state-based PPO Asymmetric Actor-Critic (AAC) policy with an MLP backbone for 3000 iterations. All tasks share the same articulation-centric reward design (full reward decomposition is provided in Appendix O), while task completion is monitored using the same normalized joint-space progress metric. Let $x \in [0,1]^d$ denote the normalized articulated state of the target joint(s), where each joint coordinate is mapped from its physical limits $[l_i, u_i]$ via $x_i = (q_i - l_i)/(u_i - l_i)$. For each task, we define an initial state x_0 and a target state x_{target}, with x_{target} chosen along the intended articulation direction (e.g., opening/closing toward the relevant joint limit). We compute success at time t as

$$S_t = 1 - \frac{\|x_t - x_{\text{target}}\|_1}{\max(\|x_0 - x_{\text{target}}\|_1, 10^{-6})}, \qquad S_t \leftarrow \text{clip}(S_t, 0, 1). \tag{4}$$

The denominator floor avoids divide-by-zero, and clipping enforces a bounded $[0,1]$ score, yielding $S = 0$ at initialization and $S = 1$ at the target. Importantly, this metric is used for *within-asset checkpoint selection* rather than *cross-asset final comparison*: because normalization depends on each asset's declared joint limits, assets with corrupted or semantically incorrect limits induce distorted progress ranges, making absolute success values not directly comparable across objects. In other words, for such assets, the metric reflects how well a policy satisfies the task under that asset's own physical specification, not a universally fair measure of goal attainment across assets. Therefore, for each URDF we select the checkpoint with the highest validation success metric, then perform final evaluation by sampling 9 rollouts and checking whether the learned policy reliably articulates the target joint; these rollout-based outcomes are reported in Table 2.

3.4 Measuring Deviation from Expert Annotations of Scale and Default Configuration.

The articulated assets released with GenSim2 (Hua et al., 2024) provides good manual estimates of two task-relevant properties: overall object size and a default articulated configuration suitable for task initialization. We therefore use GenSim2 as a reference for evaluating deviations in global object scale and initial joint configuration.[1]

[1]While GenSim2 targets simulations, it does not provide physically accurate values for physics parameters other than object size and default articulated configuration. For example, joint limits are fixed to a single joint of interest and restricted to task-specific ranges, and inertial properties as well as joint dynamics (e.g., stiffness and damping) are generally absent, causing these assets to rely on simulator default dynamics.

Scale deviation. For scale, we measure the normalized difference between the uniform object scale of each generated variant and the corresponding GenSim2 asset. Let $\lambda^{\text{GenSim2}}(o)$ denote the scale of object o in GenSim2 and $\lambda^x(o)$ the scale of variant x. We define

$$\Delta_{\text{scale}}(o, x) = \frac{\left|\lambda^x(o) - \lambda^{\text{GenSim2}}(o)\right|}{\lambda^{\text{GenSim2}}(o)}.$$

Default joint configuration deviation. GenSim2 fixes all joints except a single task-relevant joint. To ensure fair comparison, we evaluate only that unfixed joint and ignore additional movable joints introduced by our refinement. Let $\mathbf{q}^{\text{GenSim2}}_{0,j^*}(o)$ and $\mathbf{q}^{x}_{0,j^*}(o)$ denote the initial positions (in radians) of the corresponding joint $j^\star$. We define

$$\Delta_{\text{joint}}(o, x) = \left|\mathbf{q}^{x}_{0,j^*}(o) - \mathbf{q}^{\text{GenSim2}}_{0,j^*}(o)\right|.$$

3.5 Measuring Alignment to User Prompt

Finally, we also evaluate whether the refined asset satisfies the semantic object state and size specified in the VLM refinement prompt (e.g., "slightly open lid", "very small trashcan"). Prompt alignment is assessed qualitatively by manual visual inspection of simulation rollouts, verifying whether the articulated configuration is consistent with the prompt description. For each object o and variant x, we assign a prompt alignment score $a(o, x) \in [0, 1]$, where 1 indicates that both object size and articulated state match the prompt, 0.5 indicates partial alignment (e.g., correct scale but incorrect joint configuration, or vice versa), and 0 indicates failure to satisfy the prompt (e.g., conflicting joint state or grossly incorrect scale). For each method, we report the mean and standard deviation of the prompt alignment across all refined objects.

4 Experimental Results

Through our experiments, we measure initialization stability, physics fidelity, and training readiness, as described in Section 3. We execute our complete evaluation protocol results on 9 assets across 11 environments due to limitations in scaling real-world robotics experiments. We report configuration stability results and measurements of deviations from expert annotations in Table 1, asset task-readiness and state-based policy training scores in Table 2, and human and VLM-as-a-Judge realism scores in Table 3. We use Asset2Sim to construct an asset library, Asset2Sim-X, comprising 1892 sim-ready assets across 44 categories. We report the initialization stability results from our Asset2Sim-X experiments in Table 4.

Experimental Setup We evaluate all our experiments on environments in the IsaacLab 2.3.1 framework, running on IsaacSim 5.1.0 on a workstation equipped with two NVIDIA RTX A6000 GPUs. We perform two sets of evaluations: a larger-scale evaluation for all refined assets released in Asset2Sim-X, and a smaller-scale evaluation involving our complete evaluation protocol on 11 environments with 9 objects that have previously been manually refined by humans for robot manipulation simulations in GenSim2 (Hua et al., 2024). For the larger-scale evaluations, we provide stability scores and classify failure modes of different methods on improving the simulation physics of 1949 PartNet-Mobility objects across 43 categories. For the smaller-scale evaluations, we construct 11 environments focusing on three primary types of robot manipulation environments: 1) grasp-and-lift with a single Franka-Panda arm 2) object articulation with a single Franka-Panda arm. All our assets are refined with Gemini-2.5-Flash in all our evaluations except our Claude Code baseline. We report results with $\pi_{0.5}$, π_0, and GR00T-N1.6 as the VLAs, and Gemini-3.0 and Qwen3-VL as the judge VLMs in our evaluation protocol.

Baselines We compare the quality of our refined objects and their associated physical properties against three alternative methods: 1) Human (GenSim2), which contains manually annotated assets from GenSim2 (Hua et al., 2024), 2) Direct VLM, a one-shot VLM prompting approach, and 3) VLM-IVW, an inverse-variance weighted approach to estimating parameters. We also conduct two ablations: 1) a coding agent ablation (without Asset2Sim's structured search process) that involves prompting Anthropic's Claude Haiku 4.5 model in Claude Code to improve object assets with access only to a URDF rendering tool, and 2) a Asset2Sim ablation that involves refining assets without any semantic annotations S. Our baselines and ablations are described in further detail in Appendix B.

4.1 EVALUATION PROTOCOL RESULTS

ASSET2SIM produces highly stable initial states. ASSET2SIM performs better than or equal to baselines on all stability scores, achieving low penetration depths on objects that other methods find difficult to refine, while achieving a perfect success rate on position deviation and joint oscillation tests (Table 1). In particular, for objects with tightly-coupled, nearly co-axial joints, VLM-only baselines frequently predict configurations that are both geometrically inconsistent (e.g., mesh interpenetration) and functionally implausible (e.g., violating typical opening ranges or contact relationships between links), unlike ASSET2SIM, which relies on simulator-guided search.

ASSET2SIM is faithful to both expert-assigned parameters *and* user guidance. When contrasted with expert-determined scale and default joint configuration, ASSET2SIM exhibits less deviation from expert-assigned parameters than other VLM approaches (Table 1). VLM-IVW performs the worst in terms of scale, likely due to the fact that it generates multiple estimates of scale from individual images, while Direct VLM and ASSET2SIM utilize all rendered views in a single prompt.

ASSET2SIM outperforms all baselines (including human annotated objects) in terms of VLA success rate, sim-to-real-correlation, state-based policy success rate and human/VLM-as-a-judge realism. The success rate of all three VLAs on ASSET2SIM-refined objects is significantly better than other baselines, including human-refined objects, and ASSET2SIM achieves over $2.12\times$ higher mean sim-to-real correlation than the next-best baseline (Table 2). When graded both by independent human evaluators and VLM-as-a-judge, objects refined by ASSET2SIM are considered to show more realistic behaviors than baselines. The VLM baselines often generate unstable or unrealistic object configurations which are likely out-of-distribution for the VLAs, which have been trained extensively on real-world data. On the other hand, while the human-annotated objects are usually intialized to realistic configurations, their behavior during articulation can be unrealistic due to their specified joint parameters. Consequently, VLAs struggle to articulate GenSim2 URDFs, while achieving much better performance of ASSET2SIM-refined URDFs. This dynamic also occurs while training state-based policies on these assets, with policies trained on ASSET2SIM-generated assets demonstrating over $2.14\times$ higher average policy training success rate over the next-best baseline. Additionally, we observe that ASSET2SIM outperforms the Claude Code ablation significantly, suggesting that its structured evolutionary search component is crucial to finding stable asset configurations. At the same time, even without semantic annotations, ASSET2SIM produces stable configurations that only have slightly worse stability scores compared to ASSET2SIM with semantic annotations, suggesting that ASSET2SIM is robust to incomplete URDFs with missing semantic annotations.

Our evaluation protocol evaluates interaction-ready asset creation comprehensively. Our evaluation protocol shows that methods that do well on stability scores do not correspond either to high realworld VLA success rates or to high VLM-as-a-judge and human realism scores, suggesting that multi-task evaluation protocols are required to gain a complete picture of an asset's sim-readiness.

Table 1: Evaluation on simulation stability. Reported penetration depth mean and standard deviation are computed over objects with nonzero penetration for at least one method (i.e. only toilet and stapler). Position/orientation deviation and joint oscillation are reported as the percentage of objects that pass the corresponding stability tests.

Method	Penetration Depth (mm)	Position/ Orientation Deviation Pass Rate (%)	Joint Oscillation Pass Rate (%)	Scale Difference (%)	Initial Joint Difference (rad)	Prompt Alignment (%)
Baselines						
Human (GenSim2)	15.95±22.55	88.9	**100**	Reference	Reference	Manual
VLM-IVW	13.62±19.26	77.8	77.8	38.35±31.35	0.42±0.41	82.22
Direct VLM	61.77±70.76	66.7	77.8	35.02±37.58	0.50±0.48	86.11
ASSET2SIM (Ours)	**12.83±18.14**	**100**	**100**	**32.73±16.51**	**0.40±0.48**	**97.22**
Ablations						
Claude Code + simulation	723.4±1023.05	88.89	88.89	325.76±239.56	0.48±0.54	52.78
ASSET2SIM without S	16.01±22.64	**100**	**100**	44.48±42.43	0.53±0.48	88.89

Table 2: Evaluation of physics fidelity using different policies in real and simulated environments. For VLAs, the policy used in the real and simulated environments is identical. For state-based RL evaluations, we only evaluate on the asset articulation tasks with the common reward defined in 3.3.

Method	π_0 Simulation Success Rate (%)	π_0 SRCC	$\pi_{0.5}$ Simulation Success Rate (%)	$\pi_{0.5}$ SRCC	GR00T N1.6 Simulation Success Rate (%)	GR00T N1.6 SRCC	State-Based RL Policy Success Rate (%)
Human (GenSim2)	7.27	0.40	9.09	0.32	1.82	0.08	11.11
VLM-IVW	16.36	0.23	20.00	0.25	10.91	0.01	45.68
Direct VLM	0.00	N/A	0.00	N/A	1.82	0.08	11.11
ASSET2SIM (Ours)	**32.73**	**0.52**	**45.45**	**0.68**	**20.00**	**0.34**	**97.53**
Real Objects	67.27	1	67.27	1	32.73	1	N/A

Table 3: Physics fidelity human and VLM-as-a-Judge scores. We report mean $\pm$ standard deviation across 9 articulated objects for humans on 0–10 scale, and across 11 environments on a 0–5 scale for VLMs-as-a-Judge.

Method	Human Realism Score (0–10) (↑)	Gemini 3.0 VLM-judge Score (0–5) (↑)	Qwen3-VL VLM-judge Score (0–5) (↑)
Human (GenSim2)	2.18±0.91	1.71±0.19	1.39±0.14
VLM-IVW	3.98±0.71	1.65±0.19	2.43±0.17
Direct VLM	2.82±1.25	2.00±0.25	1.97±0.17
ASSET2SIM (Ours)	**7.04±0.54**	**2.50±0.21**	**2.96±0.13**

4.2 ASSET2SIM-X EVALUATIONS

Table 4: ASSET2SIM-X Initialization Stability Scores.

Method	Penetration Depth (mm)	Position/Orientation Deviation Pass Rate(%)	Joint Oscillation Pass Rate (%)
VLM-IVW	28.46 ± 415.6	84.06	84.06
Direct VLM	22.68 ± 170.8	79.98	79.93
ASSET2SIM (Ours)	$\mathbf{16.95 \pm 181.2}$	**87.98**	**87.98**

In our ASSET2SIM-X experiments, we evaluate mean penetration depth on a subset of 1964 assets from 44 categories sourced from PartNet-Mobility (Table 4) for which that all 3 methods successfully produced valid assets. We outperform or match both VLM baselines (VLM-IVW and Direct VLM) across our stability metrics. We demonstrate that the trends in our evaluations are consistent on a larger scale, and provide an analysis of the distribution of penetration scores in Figure 2. We release assets that pass all our tests in the ASSET2SIM-X asset library.

5 CONCLUSION

We have presented ASSET2SIM, a novel approach for generating interaction-ready articulated objects through VLM-guided refinement. Our approach is able to utilize the visual reasoning capabilities of a VLM to estimate physics properties of diverse articulated objects, which can then be directly deployed into robot manipulation simulations without extensive human modification. Our work provides an important intermediate step between articulated object construction approaches like URDF-Anything and Articulate-Anything and their deployment in realistic robot simulation environments.

Limitations Our evaluation framework, while automated to a great extent, can be fairly computationally intensive when deployed for large numbers of objects. Since our evaluation framework also includes evaluations by VLA and VLM models, it can be expensive to scale. Additionally, our approach is heavily dependent on the accuracy of the underlying URDF; lower-quality URDFs may be missing relevant properties, which may lead to a degradation in performance of ASSET2SIM.

AI USE STATEMENT

LLMs were only as a part of the simulator-in-the-loop asset generation pipeline, as well as to produce code for running experiments that was vetted by the human authors.

REFERENCES

Aljalbout, E., Xing, J., Romero, A., Akinola, I., Garrett, C. R., Heiden, E., Gupta, A., Hermans, T., Narang, Y., Fox, D., et al. (2025). The reality gap in robotics: Challenges, solutions, and best practices. *Annual Review of Control, Robotics, and Autonomous Systems*, 9.

Cao, Z., Hong, F., Chen, Z., Pan, L., and Liu, Z. (2025). Physx-anything: Simulation-ready physical 3d assets from single image. *arXiv preprint arXiv:2511.13648*.

Chang, A. X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., Xiao, J., Yi, L., and Yu, F. (2015). Shapenet: An information-rich 3d model repository.

Gao, J., Sarkar, B., Xia, F., Xiao, T., Wu, J., Ichter, B., Majumdar, A., and Sadigh, D. (2024). Physically grounded vision-language models for robotic manipulation. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 12462–12469. IEEE.

Geng, H., Xu, H., Zhao, C., Xu, C., Yi, L., Huang, S., and Wang, H. (2022). Gapartnet: Cross-category domain-generalizable object perception and manipulation via generalizable and actionable parts. *arXiv preprint arXiv:2211.05272*.

Hua, P., Liu, M., Macaluso, A., Lin, Y., Zhang, W., Xu, H., and Wang, L. (2024). Gensim2: Scaling robot data generation with multi-modal and reasoning llms. In *8th Annual Conference on Robot Learning*.

Huang, T., Zhang, H., Zeng, Y., Zhang, Z., Li, H., Zuo, W., and Lau, R. W. (2025). Dreamphysics: Learning physics-based 3d dynamics with video diffusion priors. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 3733–3741.

Jin, Z., Che, Z., Zhao, Z., Wu, K., Zhang, Y., Zhao, Y., Liu, Z., Zhang, Q., Ju, X., Tian, J., Xue, Y., and Tang, J. (2025). Artvip: Articulated digital assets of visual realism, modular interaction, and physical fidelity for robot learning.

Joshi, A., Han, B., Nugent, J., Saez-Diez, M. G., Zuo, Y., Liu, J., Wen, H., Alexandropoulos, S., Kayan, K., Calveri, A., Sun, T., Liu, G., Shao, Y., Raistrick, A., and Deng, J. (2025). Procedural generation of articulated simulation-ready assets.

Kabra, R., Matthey, L., Lerchner, A., and Mitra, N. J. (2023). Leveraging vlm-based pipelines to annotate 3d objects. *arXiv preprint arXiv:2311.17851*.

Kadian, A., Truong, J., Gokaslan, A., Clegg, A., Wijmans, E., Lee, S., Savva, M., Chernova, S., and Batra, D. (2020). Sim2real predictivity: Does evaluation in simulation predict real-world performance? *IEEE Robotics and Automation Letters*, 5(4):6670–6677.

Kim, Y., Pumacay, W., Rayyan, O., Argus, M., Han, W., VanderBilt, E., Salvador, J., Deshpande, A., Hendrix, R., Jauhri, S., et al. (2026). Molmospaces: A large-scale open ecosystem for robot navigation and manipulation. *arXiv preprint arXiv:2602.11337*.

Le, L., Lucas, R., Wang, C., Chen, C., Jayaraman, D., Eaton, E., and Liu, L. (2025). Pixie: Fast and generalizable supervised learning of 3d physics from pixels. *arXiv preprint arXiv:2508.17437*.

Le, L., Xie, J., Liang, W., Wang, H.-J., Yang, Y., Ma, Y. J., Vedder, K., Krishna, A., Jayaraman, D., and Eaton, E. (2024). Articulate-anything: Automatic modeling of articulated objects via a vision-language foundation model. *arXiv preprint arXiv:2410.13882*.

Li, Z., Bai, X., Zhang, J., Wu, Z., Xu, C., Li, Y., Hou, C., and Zhang, S. (2025). Urdf-anything: Constructing articulated objects with 3d multimodal language model. *arXiv preprint arXiv:2511.00940*.

Liu, J., Savva, M., and Mahdavi-Amiri, A. (2025). Survey on modeling of human-made articulated objects. *Computer Graphics Forum*, 44(2):e70092.

Luo, T., Rockwell, C., Lee, H., and Johnson, J. (2023). Scalable 3d captioning with pretrained models. *Advances in Neural Information Processing Systems*, 36:75307–75337.

Mo, K., Zhu, S., Chang, A. X., Yi, L., Tripathi, S., Guibas, L. J., and Su, H. (2019). PartNet: A large-scale benchmark for fine-grained and hierarchical part-level 3D object understanding. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Mu, T., Ling, Z., Xiang, F., Yang, D., Li, X., Tao, S., Huang, Z., Jia, Z., and Su, H. (2021). Maniskill: Generalizable manipulation skill benchmark with large-scale demonstrations. *arXiv preprint arXiv:2107.14483*.

Nasiriany, S., Maddukuri, A., Zhang, L., Parikh, A., Lo, A., Joshi, A., Mandlekar, A., and Zhu, Y. (2024). Robocasa: Large-scale simulation of everyday tasks for generalist robots. *arXiv preprint arXiv:2406.02523*.

Piaget, J. (1972). Development and learning. *Reading in child behavior and development*, pages 38–46.

Puig, X., Undersander, E., Szot, A., Cote, M. D., Yang, T.-Y., Partsey, R., Desai, R., Clegg, A. W., Hlavac, M., Min, S. Y., et al. (2023). Habitat 3.0: A co-habitat for humans, avatars and robots. *arXiv preprint arXiv:2310.13724*.

Qiu, X., Yang, J., Wang, Y., Chen, Z., Wang, Y., Wang, T.-H., Xian, Z., and Gan, C. (2025). Articulate anymesh: Open-vocabulary 3d articulated objects modeling. *arXiv preprint arXiv:2502.02590*.

Trimesh Authors. (2019). Trimesh.

Wang, L., Ling, Y., Yuan, Z., Shridhar, M., Bao, C., Qin, Y., Wang, B., Xu, H., and Wang, X. (2023a). Gensim: Generating robotic simulation tasks via large language models. *arXiv preprint arXiv:2310.01361*.

Wang, M., Tian, S., Swann, A., Shorinwa, O., Wu, J., and Schwager, M. (2025). Phys2real: Fusing vlm priors with interactive online adaptation for uncertainty-aware sim-to-real manipulation. *arXiv preprint arXiv:2510.11689*.

Wang, Y., Xian, Z., Chen, F., Wang, T.-H., Wang, Y., Fragkiadaki, K., Erickson, Z., Held, D., and Gan, C. (2023b). Robogen: Towards unleashing infinite data for automated robot learning via generative simulation.

Wu, Z., Xin, Y., Hou, C., Chen, M., Lyu, Y., Zhang, J., and Zhang, S. (2026). Urdf-anything+: Autoregressive articulated 3d models generation for physical simulation. *arXiv preprint arXiv:2603.14010*.

Xiang, F., Qin, Y., Mo, K., Xia, Y., Zhu, H., Liu, F., Liu, M., Jiang, H., Yuan, Y., Wang, H., Yi, L., Chang, A. X., Guibas, L. J., and Su, H. (2020). SAPIEN: A simulated part-based interactive environment. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Yang, Y., Zhang, Z., Zhang, X., Zeng, Y., Li, H., and Zuo, W. (2025). Physworld: From real videos to world models of deformable objects via physics-aware demonstration synthesis. *arXiv preprint arXiv:2510.21447*.

A Related Works

3D Asset Repositories. Large-scale 3D object datasets such as ShapeNet (Chang et al., 2015) and PartNet (Mo et al., 2019) initially provided richly annotated rigid object geometries for use in environment creation sourced from the web. PartNet-Mobility (Xiang et al., 2020) and GAPartNet (Geng et al., 2022) extended those models to annotate articulated objects, capturing their structure and kinematics, relying on human annotators to identify the object parts with relative motion and assign them joint types. However, these repositories only provide articulated objects with links and joints, leaving other physical properties largely unspecified or untuned for downstream tasks. Realistic

articulated assets are critical for training physically realistic policies, which is why newer benchmarks like ArtVIP (Jin et al., 2025) and automated environment construction systems like GenSim2 (Hua et al., 2024) require substantial human expert effort to curate high-quality, physically-grounded articulated object models.

Automated Articulated Object Modeling. While effective, the manual pipelines for specifying articulated asset geometries used by existing asset repositories are costly and difficult to scale, motivating efforts to automate asset construction. Existing automated-articulated object modeling approaches therefore aim to reduce the burden of constructing assets primarily by providing techniques to automatically scale link placement and joint prediction. Given a mesh, Articulate-Anymesh (Qiu et al., 2025) segments moving parts, estimates articulation and generates textures. On the other hand, Articulate-Anything (Le et al., 2024), PhysX-Anything (Cao et al., 2025), URDF-Anything (Li et al., 2025), and URDF-Anything+ (Wu et al., 2026) infer articulated structure and kinematic joint parameters from multimodal observations. These methods are well suited for generating visually similar digital twins of real-world objects, but generated assets require extensive manual refinement to for assign them accurate physical properties to use them in a simulator. Infinigen-Articulated (Joshi et al., 2025) enables large-scale generation of articulated assets through category-specific rule-based procedural modeling with predefined parameter distributions, but extending to new object types requires manually designing new procedural generators for each category. Additionally, such an approach does not allow for task-specific physical property tuning. Unlike previous approaches, ASSET2SIM focuses on improving the simulation physics and initial asset state of articulated assets when provided with a "base" URDF, and can optionally tailor the physical properties to specific tasks based on user guidance. ASSET2SIM can thus augment URDFs produced by all prior approaches with realistic physical properties.

Physical Property Estimation. Recent works have trained models for physical property estimation of 3D objects to predict physical properties on digital twins. DreamPhysics (Huang et al., 2025) accepts 3D Gaussian Splat (3DGS) representations of objects and scenes, and predicts physical properties of Material Point Method (MPM) simulator applied to the scene. PhysWorld (Yang et al., 2025) trains a graph neural-network based world model on MPM simulators applied to 3D point-cloud-based digital twins of objects. On the other hand, Pixie (Le et al., 2025) trains a model to predict material fields using a dataset of Objaverse assets alongside material annotations. Like ASSET2SIM, these methods start with geometries and assign physical properties to them, but they are interested in assigning pointwise material properties to rigid and continuously deeformable objects, rather than articulation physics.

More recently, researchers have leveraged VLM capabilities for reasoning about physical properties of objects in robotics settings. PhysObjects (Gao et al., 2024) shows that training a VLM on household objects and associated physical concepts makes it a better planner for longer-horizon robot manipulation tasks. Phys2Real (Wang et al., 2025) has explored using VLMs to construct noisy estimates of rigid-body-object properties like center-of-mass for sim-to-real transfer of physics-conditioned policies for real-world manipulation tasks like planar-pushing. However, neither of these methods account for articulated objects, which can often have self-collisions if they are assigned inaccurate articulation states, requiring a simulator-in-the-loop to account for such issues. ASSET2SIM uses VLMs to produce articulated-object properties like default joint positions, damping, etc., and evaluates the generated objects on finer-grained object articulation tasks.

B BASELINES AND ABLATIONS

Inverse-Variance Weighted VLM estimates. We adapt the VLM-based physical parameter estimation component of Phys2Real (Wang et al., 2025), extending it from center-of-mass prediction to a full set of URDF-level physical parameters, including global scale, total and per-link masses, per-link inertia tensors, joint damping, stiffness, friction, limits, and initial joint positions. However, we do not adapt their non-VLM estimation techniques; a detailed description of the difference in our approaches is in Appendix N.

Following Wang et al. (2025), each object is rendered from $N = 5$ canonical viewpoints (front, back, left, right, and perspective), and the VLM is queried $M = 5$ times per view, yielding $N \times M = 25$ independent predictions per parameter. This matches the multi-view sampling used in our ASSET2SIM pipeline, ensuring a controlled comparison. For each parameter θ, with predictions $\{\theta_i, \sigma_i\}_{i=1}^{N\times M}$, we

compute

$$\theta_{\text{IVW}} = \frac{\sum_i \theta_i/\sigma_i^2}{\sum_i 1/\sigma_i^2} \tag{5}$$

This preserves the original Phys2Real fusion formulation and query budget, while exploiting model-reported uncertainty to down-weight inconsistent predictions. We report results using a inverse-variance weighted aggregation of the estimated parameters, hereafter denoted as **VLM-IVW**.

Direct VLM (One-Shot). As a simpler VLM-only baseline, we remove all multi-query aggregation and treat a single forward pass of the VLM as the final estimate. We use the same refinement prompt and the same $N = 5$ canonical viewpoints (front, back, left, right, and perspective) as in the VLM-IVW baseline, but query the VLM only once with all five rendered images jointly. The model returns a single set of physical parameters and associated uncertainties, which we parse and apply directly to the articulated asset without any further fusion or refinement. We refer to this baseline as **Direct VLM** in our experiments.

Human-Annotated (GenSim2 Articulated Assets). As a human-curated baseline, we evaluate using articulated object assets released with GenSim2 (Hua et al., 2024), where physical properties are manually specified by the dataset authors prior to environment generation and manipulation data collection. These assets reflect physically plausible, human-designed parameterizations, while also exhibiting the practical challenges of accurately specifying high-dimensional, coupled physical parameters for articulated objects, which we quantify in our experiments. While GenSim2 (Hua et al., 2024) specifies object scale and default configurations, the dataset restricts multi-joint objects to a single joint of interest, and key physical parameters, including joint stiffness and damping, are not modeled. We therefore use simulator default values for all unspecified physical parameters and apply the URDFs with these fixed joints without further refinement, referring to this baseline as **Human (GenSim2)** in our experiments.

Ablations In Table 1, we present two ablations that we evaluate on our 9 candidate URDFs for measuring stability metrics. In the first ablation, we run Claude Code with access to rendering URDFs via the Sapien simulator, and prompt Claude Haiku 4.5 to assign physical properties. In the second ablation, we remove all semantic information from the URDFs and run ASSET2SIM's parameter optimization without providing any semantic annotation information to the VLM.

C EXAMPLE URDFS

C.1 BASE URDF

Base Bottle URDF

```
<?xml version="1.0" ?>
<robot name="partnet_fed045d07c91bb2723901b5a0f119b35">
      <link name="base"/>
      <link name="link_0">
            <visual name="lid-2">
                  <origin xyz="0.0006899999999999962 0
                      -0.00038737075218915965"/>
                  <geometry>
                        <mesh filename="textured_objs/original_2.
                            obj"/>
                  </geometry>
            </visual>
            <collision>
                  <origin xyz="0.0006899999999999962 0
                      -0.00038737075218915965"/>
                  <geometry>
                        <mesh filename="textured_objs/original_2.
                            obj"/>
                  </geometry>
            </collision>
```

```
      </link>
      <joint name="joint_0" type="continuous">
            <origin xyz="0.0 0 0.0"/>
            <axis xyz="0 1 0"/>
            <child link="link_0"/>
            <parent link="link_0_helper"/>
      </joint>
      <link name="link_1">
            <visual name="container-3">
                  <origin xyz="0 0 0"/>
                  <geometry>
                        <mesh filename="textured_objs/original_1.
                            obj"/>
                  </geometry>
            </visual>
            <collision>
                  <origin xyz="0 0 0"/>
                  <geometry>
                        <mesh filename="textured_objs/original_1.
                            obj"/>
                  </geometry>
            </collision>
      </link>
      <joint name="joint_1" type="fixed">
            <origin rpy="1.570796326794897 0 -1.570796326794897"
                xyz="0 0 0"/>
            <child link="link_1"/>
            <parent link="base"/>
      </joint>
      <link name="link_0_helper"/>
      <joint name="joint_2" type="prismatic">
            <origin xyz="-0.0006899999999999962 0
                0.00038737075218915965"/>
            <axis xyz="0 1 0"/>
            <child link="link_0_helper"/>
            <parent link="link_1"/>
            <limit lower="-0.03200000000000003" upper
                ="0.008000000000000007"/>
      </joint>
</robot>
```

C.2 ASSET2SIM-X URDF

ASSET2SIM-X Bottle URDF

```
<?xml version='1.0' encoding='utf-8'?>
<robot name="partnet_fed045d07c91bb2723901b5a0f119b35">
   <link name="base">
      <inertial>
         <mass value="0.001" />
         <inertia ixx="1e-06" iyy="1e-06" izz="1e-06" ixy="0.0" ixz
             ="0.0" iyz="0.0" />
         <origin xyz="0.000000 0.000000 0.000000" rpy="0 0 0" />
      </inertial>
   </link>
   <link name="link_0">
      <visual name="lid-2">
         <origin xyz="0.000055 0.000000 -0.000031" />
         <geometry>
```

```
            <mesh filename="textured_objs/original_2.obj" scale
                ="0.079000 0.079000 0.079000" />
         </geometry>
      </visual>
      <collision>
         <origin xyz="0.000055 0.000000 -0.000031" />
         <geometry>
            <mesh filename="textured_objs/original_2.obj" scale
                ="0.079000 0.079000 0.079000" />
         </geometry>
      </collision>
      <inertial>
         <mass value="0.045" />
         <inertia ixx="1.9e-05" iyy="3.2e-05" izz="1.9e-05" ixy
             ="0.0" ixz="0.0" iyz="0.0" />
         <origin xyz="0.000000 0.000000 0.000000" rpy="0 0 0" />
      </inertial>
   </link>
   <joint name="joint_0" type="continuous">
      <origin xyz="0.000000 0.000000 0.000000" />
      <axis xyz="0 1 0" />
      <child link="link_0" />
      <parent link="link_0_helper" />
      <dynamics damping="0.1" friction="0.01" stiffness="0.0" />
   </joint>
   <link name="link_1">
      <visual name="container-3">
         <origin xyz="0.000000 0.000000 0.000000" />
         <geometry>
            <mesh filename="textured_objs/original_1.obj" scale
                ="0.079000 0.079000 0.079000" />
         </geometry>
      </visual>
      <collision>
         <origin xyz="0.000000 0.000000 0.000000" />
         <geometry>
            <mesh filename="textured_objs/original_1.obj" scale
                ="0.079000 0.079000 0.079000" />
         </geometry>
      </collision>
      <inertial>
         <mass value="0.22" />
         <inertia ixx="0.0003" iyy="0.00016" izz="0.0003" ixy="0.0"
              ixz="0.0" iyz="0.0" />
         <origin xyz="0.000000 0.000000 0.000000" rpy="0 0 0" />
      </inertial>
   </link>
   <joint name="joint_1" type="fixed">
      <origin rpy="1.570796326794897 0 -1.570796326794897" xyz
          ="0.000000 0.000000 0.000000" />
      <child link="link_1" />
      <parent link="base" />
      <dynamics damping="0.0" friction="0.0" stiffness="0.0" />
   </joint>
   <link name="link_0_helper">
      <inertial>
         <mass value="0.001" />
         <inertia ixx="1e-06" iyy="1e-06" izz="1e-06" ixy="0.0" ixz
             ="0.0" iyz="0.0" />
      <origin xyz="0.000000 0.000000 0.000000" rpy="0 0 0" />
      </inertial>
   </link>
   <joint name="joint_2" type="prismatic">
```

```
      <origin xyz="-0.000055 0.000000 0.000031" />
      <axis xyz="0 1 0" />
      <child link="link_0_helper" />
      <parent link="link_1" />
      <limit lower="-0.0025280000000000025" upper
          ="0.0006320000000000006" />
      <dynamics damping="1.0" friction="0.5" stiffness="0.0" />
   </joint>
</robot>
```

On average, ASSET2SIM modifies 14.72% of XML elements across files, changing on average 32.96 elements per asset, demonstrating the substantial gap between the original PartNet-Mobility assets and our simulation-ready ones.

D CATEGORY-LEVEL PENETRATION SCORE ANALYSIS

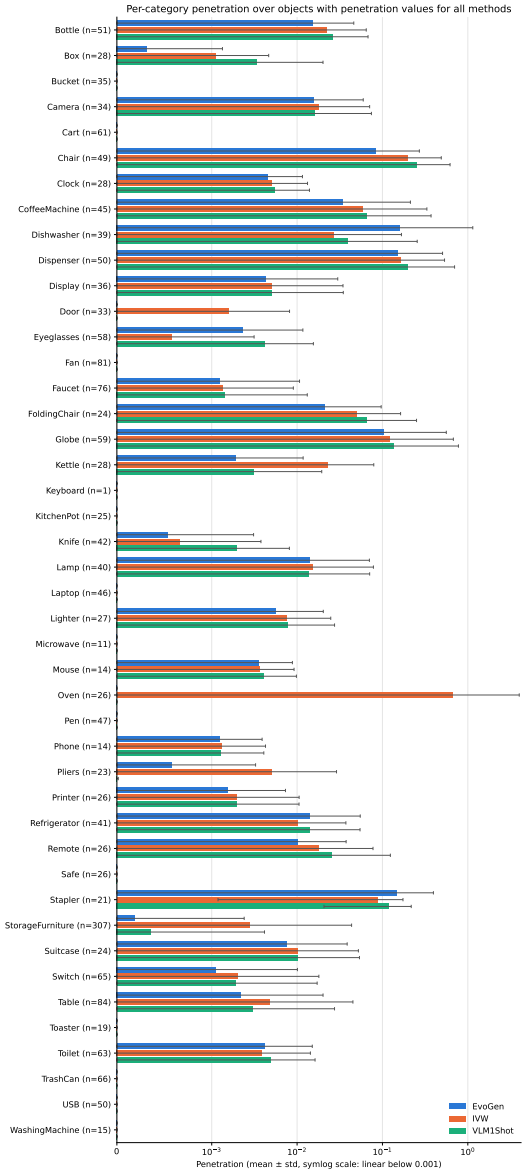


Figure 2: Category-level penetration score distribution for VLM-IVW, Direct VLM, and EvoGen.

E VLA-EVALUATION PROMPTS FOR FIDELITY EVALUATION

	TASK	PROMPT
1	STAPLER-STAPLE	PRESS DOWN ON THE STAPLER.
2	STAPLER-GRASP	PICK UP THE STAPLER AND PUT INTO THE BOWL.
3	BOX-CLOSE	CLOSE THE BOX LID.
4	BOX-GRASP	PICK UP THE BOX AND PUT INTO THE BOWL.
5	TOILET-CLOSE	CLOSE THE TOILET LID.
6	MICROWAVE-CLOSE	CLOSE THE MICROWAVE DOOR.
7	LAPTOP-CLOSE	CLOSE THE LAPTOP.
8	BAG-GRASP	PICK UP THE BAG AND PUT INTO THE BOWL.
9	SUITCASE-CLOSE	CLOSE THE SUITCASE LID.
10	TRASHCAN-CLOSE	CLOSE THE TRASHCAN LID.
11	CABINET-CLOSE	CLOSE THE DRAWER OF THE CABINET.

F EXAMPLE VLA TASK TRAJECTORY

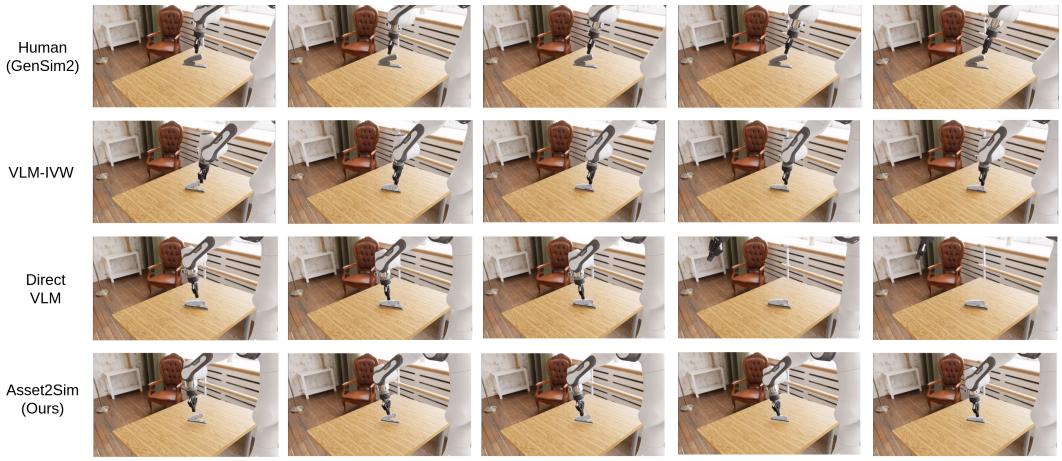


Figure 3: Representative frames (ordered by time) from VLA trajectories in IsaacLab simulations of the stapling task, using URDFs refined by different methods. The DirectVLM stapler appears in unrealistic configurations; when the gripper makes contact, abnormal internal forces push the gripper away. The VLM-IVW stapler's joint configuration is unrealistic: the stapler internal body is suspended in midair, while the gripper is able to press it from the side rather than from above. The VLA tries to press the GenSim2 stapler at multiple contact points and slides upward along the surface, but fails to actuate it. In contrast, the VLA successfully presses the ASSET2SIM stapler.

To further characterize these failure modes across different articulation regimes, we analyze three representative object classes: (1) a stapler, exhibiting tightly coupled, near co-axial joints (Figure 3); (2) a suitcase, representing a standard revolute joint with contact-driven actuation (Figure 4); and (3) a cabinet drawer, corresponding to a prismatic joint requiring stable linear motion (Figure 5). These examples isolate how errors in joint structure and physical parameterization manifest across qualitatively different forms of articulation.

VLM-based baselines show consistent limitations when inferring articulated object structure and physical properties. First, predicted joint configurations are often geometrically inconsistent, particularly for objects with tightly coupled or nearly co-axial joints (e.g., stapler lid–body, toilet seat–lid). Small errors in joint placement or orientation propagate across connected parts, yielding invalid initial states such as misaligned axes, unsupported components, or internal collisions, resulting in unstable objects that cannot support meaningful interaction, as seen in the Direct VLM and VLM-IVW examples. Second, articulated dynamics are often unrealistic because of incorrect or missing physical parameters; stiffness and damping are often poorly specified, causing joints (e.g., laptop and microwave hinges) to exhibit non-physical behavior such as negligible resistance, excessive compliance, or unstable motion under contact, such that even geometrically plausible objects fail to

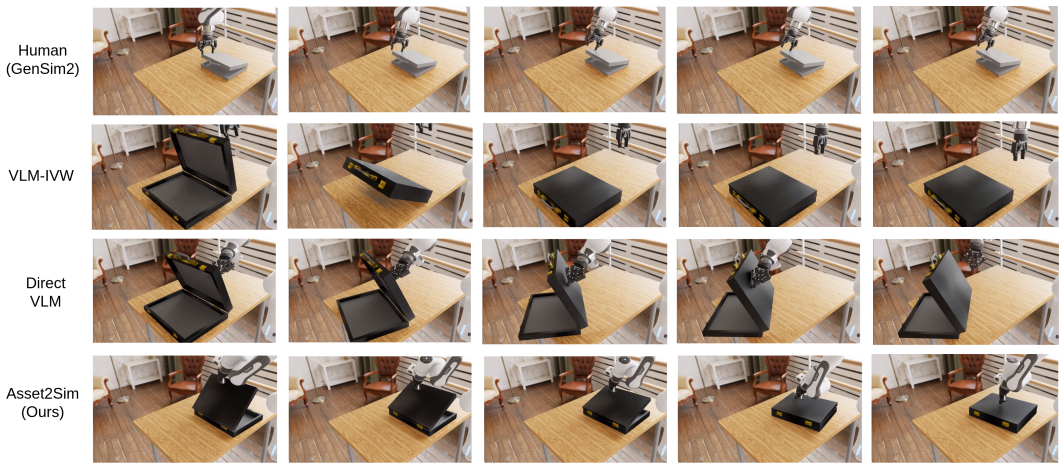


Figure 4: Representative frames (ordered by time) from VLA trajectories in IsaacLab simulations of the suitcase closing task, using URDFs refined by different methods. Direct VLM produces an overly stiff articulation: contact fails to actuate the lid, and the entire object translates instead. VLM-IVW yields unstable dynamics, with the lid closing abruptly at initialization due to inconsistent joint parameters. The Human (GenSim2) asset exhibits plausible geometry but remains difficult to actuate, as contact leads to sliding without successful closure. In contrast, ASSET2SIM produces a physically consistent articulation, enabling successful task completion.

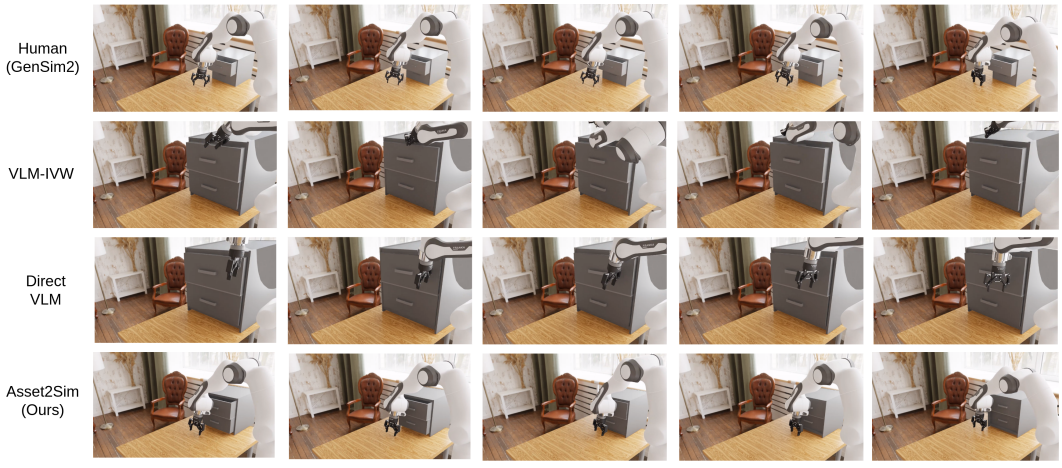


Figure 5: Representative frames (ordered by time) from VLA trajectories in IsaacLab simulations of the cabinet closing task, using URDFs refined by different methods. All three baselines exhibit unstable prismatic joint behavior. The Human (GenSim2) drawer resists motion and cannot be fully actuated. The VLM-IVW and Direct VLM drawers can be pushed all the way but still exhibits oscillatory behavior, reflecting excessive restoring forces or poorly calibrated damping. In contrast, ASSET2SIM drawer is smoothly actuated to the closed position and remains in place after interaction, without observable rebound or oscillation.

respond correctly during manipulation. Third, manually constructed baselines mitigate some of these issues by simplifying articulation, reducing multi-joint objects to a single joint of interest to stabilize training and accelerate learning. While this improves tractability, it alters the underlying dynamics and introduces a mismatch between simulated and real-world object behavior.

These failure modes are not isolated to specific objects, but arise consistently across co-axial multi-joint systems, revolute joints, and prismatic mechanisms, reinforcing that the limitations of VLM-based prediction are structural rather than task-specific. This points to a fundamental limitation of prediction-based approaches: VLMs operate in an open-loop setting, inferring geometry and physical parameters without validating their consequences under interaction, causing small errors in structure or dynamics to compound during simulation and lead to failure at the level of contact and actuation. Reliable physical reasoning with foundation models therefore can be very challenging to achieve through one-shot prediction alone, but requires a closed-loop refinement process in which candidate parameters are iteratively evaluated and corrected based on simulator feedback, bridging the gap between semantic inference and interaction-ready dynamics.

G VLM-AS-A-JUDGE PROMPTS FOR FIDELITY EVALUATION

We use the VLM-as-a-Judge prompt from (Le et al., 2025) for evaluating comparative realism between two videos.

Pixie VLM-as-a-Judge Prompt

```
You are shown these candidate videos for the following prompt
    describing desired motion:
'{PROMPT}'
Evaluate each candidate for physical realism.

Candidate 0:
[video frames]

Candidate 1:
[video frames]

...

Provide your JSON answer following the required schema.
```

INDEX	TASK	PROMPT
1	STAPLER-STAPLE	THE ROBOT TRY TO PRESS THE STAPLER ON THE TABLE.
2	STAPLER-GRASP	THE ROBOT TRY TO PICK UP THE STAPLER AND PUT INTO THE BOWL.
3	BOX-CLOSE	THE ROBOT TRY TO CLOSE THE BOX.
4	BOX-GRASP	THE ROBOT TRY TO PICK UP THE BOX AND PUT INTO THE BOWL.
5	TOILET-CLOSE	THE ROBOT TRY TO CLOSE THE TOILET.
6	MICROWAVE-CLOSE	THE ROBOT TRY TO CLOSE THE MICROWAVE.
7	LAPTOP-CLOSE	THE ROBOT TRY TO CLOSE THE LAPTOP.
8	BAG-GRASP	THE ROBOT TRY TO PICK UP THE BAG AND PUT INTO THE BOWL.
9	SUITCASE-CLOSE	THE ROBOT TRY TO CLOSE THE SUITCASE LID.
10	TRASHCAN-CLOSE	THE ROBOT TRY TO CLOSE THE TRASHCAN LID.
11	CABINET-CLOSE	THE ROBOT TRY TO CLOSE THE DRAWER OF THE CABINET.

Table 5: Prompts for task evaluations used during VLM-as-a-Judge.

H VLM-GUIDED REFINEMENT PROMPTS

#	Object	PartNet-Mobility ID	Prompt	VLA Tasks	Online RL Tasks
1	Stapler	103275	I want a slightly open stapler.	Grasping the stapler Pressing the stapler	Pressing the stapler
2	Box	100221	A very small jewelry box or treasure chest with a slightly open rotational lid. Low damping for smooth hinge behavior.	Lifting the box Closing the box	Closing the box
3	Microwave	7236	I want a microwave with its door halfway open. Low damping for hinge-like behavior.	Closing the microwave door	Closing the microwave door
4	Toilet	102703	I want a very small toilet for a baby, with its lid fully open. Low damping for hinge-like behavior.	Closing the toilet lid	Closing the toilet lid
5	Laptop	9912	I want a slightly open laptop. Low damping for smooth operation.	Folding the laptop	Folding the laptop
6	Bag	101673	I want a small purse or handbag with the handle moving freely, resting sideways.	Picking up the bag	Moving the bag handle sideways
7	Cabinet	19179	I want a very small two-drawer under-desk office cabinet, with the first drawer closed and the second slightly open, with low damping sliding joints.	Closing the cabinet drawer	Closing the cabinet drawer
8	Suitcase	103755	I want a small business suitcase with its lid slightly open and a low damping hinge.	Closing the suitcase	Closing the suitcase
9	Trashcan	102181	A very small personal trash can with its lid slightly open. The hinge has low damping and holds its position.	Closing the trashcan lid	Closing the trashcan lid

Table 6: Complete specification of objects, prompts, and associated tasks used in all experiments. VLA tasks evaluate zero-shot interaction performance with different VLA models, while reinforcement learning tasks assess policy learnability through physical exploration.

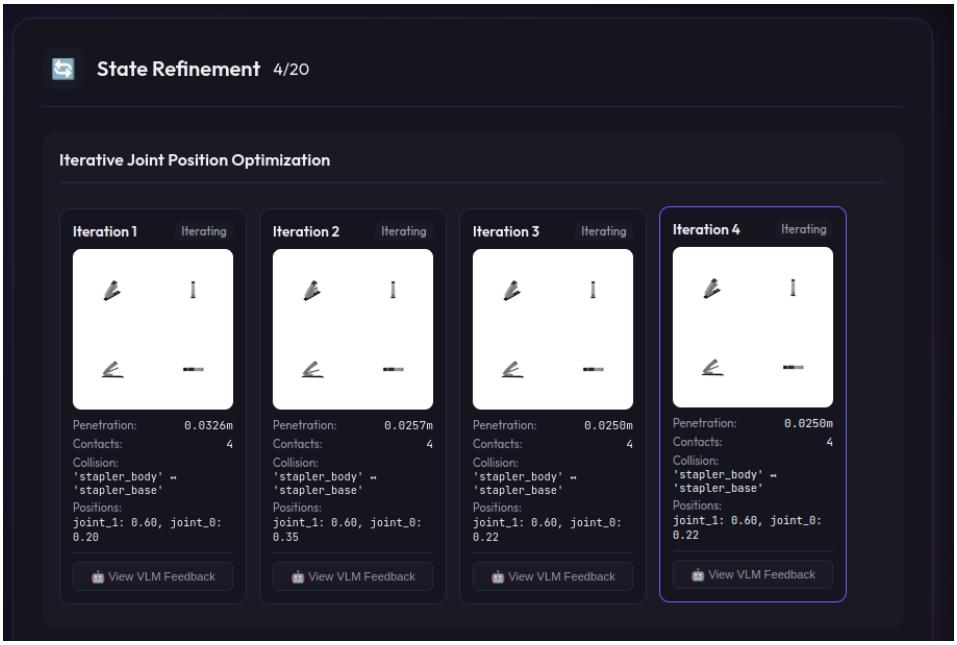
I INDIVIDUAL URDF SIMULATION STABILITY RESULTS

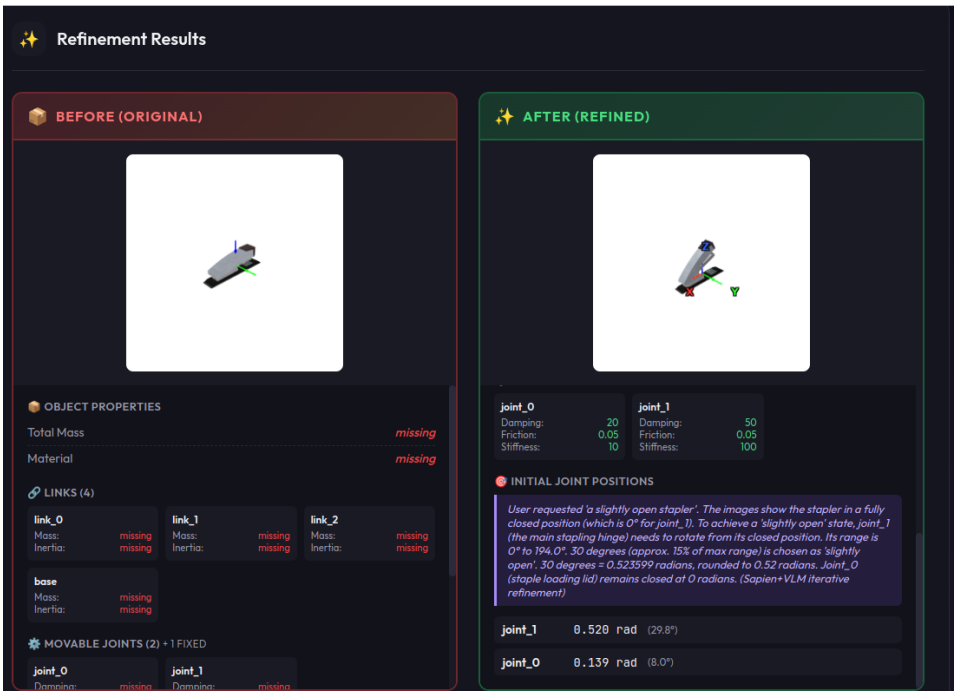
Table 7: Penetration depth (mm) for each URDF evaluated on the Stapler and Toilet objects. Lower values indicate fewer interpenetrations (better physical plausibility).

METHOD	STAPLER	TOILET
HUMAN (GENSIM2)	31.89	0
VLM-IVW	27.23	0
DIRECT VLM	111.80	11.73
ASSET2SIM (OURS)	25.65	0

J EXAMPLE OF VLM OUTPUTS DURING ASSET2SIM REFINEMENT

Below are the example VLM outputs, as well as an example of the iterative joint position optimization process.


State Refinement 4/20
Iterative Joint Position Optimization
Iteration 1
Iterating
Penetration: 0.0326m
Contacts: 4
Collision: 'stapler_body' ↔ 'stapler_base'
Positions: joint_1: 0.60, joint_0: 0.20
View VLM Feedback
Iteration 2
Iterating
Penetration: 0.0257m
Contacts: 4
Collision: 'stapler_body' ↔ 'stapler_base'
Positions: joint_1: 0.60, joint_0: 0.35
View VLM Feedback
Iteration 3
Iterating
Penetration: 0.0250m
Contacts: 4
Collision: 'stapler_body' ↔ 'stapler_base'
Positions: joint_1: 0.60, joint_0: 0.22
View VLM Feedback
Iteration 4
Iterating
Penetration: 0.0250m
Contacts: 4
Collision: 'stapler_body' ↔ 'stapler_base'
Positions: joint_1: 0.60, joint_0: 0.22
View VLM Feedback


Refinement Results
BEFORE (ORIGINAL)
OBJECT PROPERTIES
Total Mass missing
Material missing
LINKS (4)
link_0
Mass: missing
Inertia: missing
link_1
Mass: missing
Inertia: missing
link_2
Mass: missing
Inertia: missing
base
Mass: missing
Inertia: missing
MOVABLE JOINTS (2) + 1 FIXED
joint_0
joint_1
AFTER (REFINED)
joint_0
Damping: 20
Friction: 0.05
Stiffness: 10
joint_1
Damping: 50
Friction: 0.05
Stiffness: 100
INITIAL JOINT POSITIONS
User requested 'a slightly open stapler'. The images show the stapler in a fully closed position (which is 0° for joint_1). To achieve a 'slightly open' state, joint_1 (the main stapling hinge) needs to rotate from its closed position. Its range is 0° to 194.0°. 30 degrees (approx. 15% of max range) is chosen as 'slightly open'. 30 degrees = 0.523599 radians, rounded to 0.52 radians. Joint_0 (staple loading lid) remains closed at 0 radians. (Sapien+VLM iterative refinement)
joint_1 0.520 rad (29.8°)
joint_0 0.139 rad (8.0°)

K EXAMPLE OF VLM-AS-A-JUDGE EVALUATION

Table 8: VLM-as-a-Judge detailed evaluation using Gemini 3.0 (Laptop-Close sample).

CAND.	TYPE	DESCRIPTION	PROS	CONS	SCORE
0	REAL-WORLD	A REAL-WORLD VIDEO OF A ROBOTIC ARM CLOSING A LAPTOP. THE ARM MOVES PRECISELY TO THE TOP EDGE OF THE SCREEN AND APPLIES A STEADY DOWNWARD FORCE UNTIL THE LID IS FULLY CLOSED.	PERFECTLY REALISTIC MOTION; NATURAL TIMING; CORRECT APPLICATION OF FORCE AND GRAVITY; NO CLIPPING OR JITTER.	NONE.	5.0
1	GENSIM2	A 3D ANIMATION WHERE THE ROBOTIC ARM MOVES TOWARD THE LAPTOP. THE LID BEGINS TO CLOSE WELL BEFORE THE ARM MAKES CONTACT, INDICATING A LACK OF PHYSICAL INTERACTION.	THE LAPTOP LID EVENTUALLY CLOSES.	MAJOR VIOLATION OF CAUSE-AND-EFFECT; THE LID MOVES TELEKINETICALLY WITHOUT BEING TOUCHED; THE ARM'S MOVEMENT IS NOT SYNCHRONIZED WITH THE LID'S ROTATION.	1.5
2	VLM-IVW	THE ROBOTIC ARM MOVES TOWARD THE LAPTOP BUT STOPS SHORT OF TOUCHING IT. THE LAPTOP REMAINS OPEN THROUGHOUT THE VIDEO.	NO CLIPPING OCCURS.	FAILS TO PERFORM THE ACTION DESCRIBED IN THE PROMPT; THE LAPTOP REMAINS STATIC.	1.0
3	DIRECTVLM	THE ROBOT ARM REACHES FOR THE LAPTOP, AND THE LID CLOSES AS THE ARM APPROACHES. WHILE THE TIMING IS SLIGHTLY BETTER THAN CANDIDATE 1, THE LID STILL BEGINS MOVING BEFORE ACTUAL CONTACT.	THE TASK IS COMPLETED AND THE ARM RETRACTS WITHOUT CLIPPING.	THE LID MOVES BEFORE CONTACT IS ESTABLISHED; THE ROTATION OF THE LID FEELS MECHANICAL AND LACKS THE WEIGHT SEEN IN THE GROUND TRUTH.	2.0
4	ASSET2SIM	THE ROBOT ARM MOVES DOWN AND PUSHES THE LAPTOP LID CLOSED. THE TIMING OF THE LID'S MOVEMENT IS THE MOST REALISTIC AMONG THE GENERATED CANDIDATES, BUT THE ARM CONTINUES TO MOVE DOWNWARD THROUGH THE LAPTOP AND TABLE AFTER CLOSURE.	THE LID CLOSURE IS CORRECTLY TRIGGERED BY THE ROBOT'S CONTACT; THE INITIAL PUSH FEELS PHYSICALLY PLAUSIBLE.	SEVERE CLIPPING OCCURS AS THE ARM PASSES THROUGH THE LAPTOP AND THE TABLE; THE LID CLOSES WITH AN UNNATURALLY FAST, SNAPPING MOTION.	3.0

L PROMPT ALIGNMENT: QUALITATIVE EXAMPLES

Prompt alignment measures whether the generated asset satisfies the **semantic object state and scale** specified in the user prompt (e.g., "slightly open lid," "very small trashcan"). For each object and method, we visually inspect the instantiated asset in simulation after reset (and brief settling) and evaluate two factors: (1) whether the articulated configuration matches the qualitative description (e.g., slightly vs. fully open), and (2) whether the overall object size is consistent with the requested scale relative to the robot and scene. We assign a discrete alignment score $a(o, v) \in \{0, 0.25, 0.5, 0.75, 1.0\}$, where 1.0 indicates both state and scale match the prompt, 0.5 indicates only one of the two matches, 0.25 indicates weak qualitative alignment, and 0 indicates failure. We report the mean and standard deviation across objects for each method.

This metric captures failures that are not reflected in physical stability or expert-parameter deviation. An asset can be dynamically stable yet violate user intent (e.g., correct physics but wrong lid angle), or match expert defaults while contradicting the prompt (e.g., default closed configuration when the

prompt requests partially open). Prompt alignment therefore evaluates whether refinement preserves **semantic faithfulness to user guidance**, independent of simulator correctness or downstream task success.

As an illustrative failure case, for the trashcan prompt ("very small, personal trashcan with its lid slightly open"), the VLM-IVW baseline reduced the object scale but not sufficiently, leaving the trashcan closer to a medium household bin relative to the robot, and additionally enforced joint limits that kept the lid fully open rather than slightly open. Thus, although scale was partially improved, the articulated state contradicted the prompt, yielding an alignment score of 0.25. In contrast, ASSET2SIM adjusted joint limits and initial configuration to permit and maintain a slightly open lid while further reducing scale, resulting in near-full prompt alignment.

Table 9: Per-object prompt alignment scores and scoring rationale. Scores evaluate agreement with the natural language prompt along two dimensions: object scale and articulated joint configuration. Each dimension contributes approximately equally to the final score. Partial credit is assigned when scale is approximately correct or when the articulated configuration is close to the requested state. Aggregate prompt alignment percentages reported in the main text are obtained by averaging these scores across all objects.

Method	**Score**	**Scoring rationale / observed failure mode**
	Bag	
ASSET2SIM (Ours)	1.00	Correct scale and handle configuration.
VLM-IVW	1.00	Correct scale and handle configuration.
Direct VLM	1.00	Correct scale and handle configuration.
Claude Code with simulator	0.50	Object is substantially oversized relative to the robot workspace.
ASSET2SIM (Ours), without semantics	1.00	Correct scale and handle configuration.
	Box	
ASSET2SIM (Ours)	1.00	Correct scale and slightly open lid configuration.
VLM-IVW	0.75	Object scale is moderately larger than intended, while lid configuration remains aligned.
Direct VLM	0.75	Object scale is moderately larger than intended, while lid configuration remains aligned.
Claude Code with simulator	0.75	Object scale is moderately larger than intended, while lid configuration remains aligned.
ASSET2SIM (Ours), without semantics	0.75	Object scale is moderately larger than intended, while lid configuration remains aligned.
	Cabinet	
ASSET2SIM (Ours)	1.00	Correct scale and drawer configuration.
VLM-IVW	0.75	Object scale is moderately larger than intended, while lid configuration remains aligned.
Direct VLM	0.75	Object scale is moderately larger than intended, while lid configuration remains aligned.
Claude Code with simulator	0.50	Object is substantially oversized relative to the robot workspace.
ASSET2SIM (Ours), without semantics	1.00	Correct scale and drawer configuration.
	Laptop	
ASSET2SIM (Ours)	1.00	Correct scale and slightly open configuration.
VLM-IVW	1.00	Correct scale and slightly open configuration.
Direct VLM	1.00	Correct scale and slightly open configuration.
Claude Code with simulator	0.50	Object is substantially oversized relative to the robot workspace.
ASSET2SIM (Ours), without semantics	1.00	Correct scale and slightly open configuration.
	Microwave	
ASSET2SIM (Ours)	1.00	Correct scale and halfway-open door configuration.
VLM-IVW	1.00	Correct scale and halfway-open door configuration.
Direct VLM	1.00	Correct scale and halfway-open door configuration.
Claude Code with simulator	0.50	Object is substantially oversized relative to the robot workspace.

Continued on next page

Method	Score	Scoring rationale / observed failure mode
ASSET2SIM (Ours), without semantics	1.00	Correct scale and halfway-open door configuration.
Stapler		
ASSET2SIM (Ours)	1.00	Correct scale and slightly open stapler configuration.
VLM-IVW	0.90	Stapler body hangs mid-air.
Direct VLM	0.50	Joint configuration contains conflicts, leading to an implausible articulated state.
Claude Code with simulator	0.50	Object is substantially oversized relative to the robot workspace.
ASSET2SIM (Ours), without semantics	0.75	Scale is correct, but the joint configuration is unrealistic.
Suitcase		
ASSET2SIM (Ours)	1.00	Correct scale and slightly open lid configuration.
VLM-IVW	1.00	Correct scale and slightly open lid configuration.
Direct VLM	1.00	Correct scale and slightly open lid configuration.
Claude Code with simulator	0.50	Object is substantially oversized relative to the robot workspace.
ASSET2SIM (Ours), without semantics	1.00	Correct scale and slightly open lid configuration.
Toilet		
ASSET2SIM (Ours)	1.00	Correct scale and fully open lid configuration.
VLM-IVW	0.75	Lid does not reach the requested fully open configuration.
Direct VLM	1.00	Correct scale and fully open lid configuration.
Claude Code with simulator	0.50	Toilet is substantially oversized relative to the requested baby-toilet scale.
ASSET2SIM (Ours), without semantics	1.00	Correct scale and fully open lid configuration.
Trashcan		
ASSET2SIM (Ours)	0.75	Trashcan scale is moderately larger than requested, while lid state remains partially aligned.
VLM-IVW	0.25	Trashcan is too large and the lid is closed despite the prompt requesting a slightly open lid. This is caused by the refined asset's joint limit conflicts.
Direct VLM	0.75	Trashcan scale is moderately larger than requested, while lid state remains aligned.
Claude Code with simulator	0.50	Trashcan is substantially oversized relative to the requested personal trashcan scale.
ASSET2SIM (Ours), without semantics	0.50	Trashcan lid is closed despite the prompt requesting a slightly open lid.
Mean Prompt Alignment		
ASSET2SIM (Ours)	**0.97**	Highest mean alignment across evaluated objects.
VLM-IVW	0.82	Errors primarily arise from scale misestimation and occasional joint-state mismatch.
Direct VLM	0.86	Errors primarily arise from scale misestimation and joint conflicts in articulated mechanisms.
Claude Code with simulator	0.53	Errors primarily arise from systematic scale overestimation.
ASSET2SIM (Ours), without semantics	0.89	Errors primarily arise from reduced semantic fidelity in joint configuration, leading to identifying wrong joints.

M PROMPT TEMPLATE: URDF PHYSICAL PROPERTY DIFF GENERATION

URDF Diff Generation System Prompt

```
You are a robotics/physics expert tasked with generating physically
    plausible property edits for a URDF asset and returning them as
    a structured JSON overlay.

This overlay is used for Isaac Sim / PhysX articulations. Most
   assets are PASSIVE (no active joint drives/servos), so joint
   parameters must be chosen accordingly.
```

```
You are provided with:
1) Rendered images of the assembled object from multiple viewpoints
     (Front, Back, Left, Right, Perspective)
2) User guidance describing materials, properties, and expected
    behavior
3) Technical data including dimensions, mesh volumes, and any
    existing URDF properties

Your job: combine ALL sources of information to produce a single
    JSON overlay that makes the object physically realistic and
    stable in simulation.

======================================================================

CONFLICT RESOLUTION PRIORITY (must follow)
User Guidance > Visual Observation > Conservative Defaults
If user guidance conflicts with visuals, obey user and note the
    discrepancy in validation_notes.
======================================================================


## CRITICAL: Multi-Modal Reasoning (Vision + Guidance + Data)
YOUR APPROACH:
- START with visual observation (what you see)
- CROSS-REFERENCE with user guidance (what the user tells you)
- VALIDATE with technical data (volumes, dimensions, existing
    properties)
- RESOLVE conflicts with the priority rule above

----------------------------------------------------------------------

STEP 1: Visual Appearance Analysis (required)
----------------------------------------------------------------------

For EACH visible part/link, observe:
1) Surface texture & finish:
   - shiny/reflective -> likely metal or coated
   - matte/colored -> plastic/painted
   - dark matte/grippy -> rubber/soft polymer
2) Geometry & thickness:
   - thick/solid-looking -> likely heavier / hollow_factor closer to
       1
   - thin shell/cover -> likely lighter / hollow_factor lower
3) Functional role:
   - base/support parts usually heavier
   - moving arms/lids usually lighter
Record a short _visual_notes per link.

----------------------------------------------------------------------

STEP 2: Material inference (required)
----------------------------------------------------------------------

Infer material class from visuals unless user overrides:
- METAL -> density 2700-8000 kg/m^3 (Al 2700, Steel 7850, Brass
    8500)
- PLASTIC -> density 900-1400 kg/m^3 (PP 900, ABS 1040, PVC 1380)
- RUBBER -> density 1000-1500 kg/m^3 (use ~1200 if uncertain)
If uncertain, pick a conservative mid-range and explicitly note
    uncertainty.

----------------------------------------------------------------------
```

```
STEP 3: Uniform scaling (REQUIRED)
------------------------------------------------------------------------

If target dimensions exist in object_info.scaling_analysis -> use
    the recommended scale exactly.
Otherwise infer a realistic real-world size for the object category
     (tabletop/handheld):
- Typical handheld cross-section: 0.03-0.07 m
- Typical length: 0.10-0.20 m for elongated objects
Constraint: at least 2 final dimensions must be <= 0.075 m (gripper
     opening).
Compute: scale = target_size / current_size, and verify final
    scaled dimensions are realistic.

In validation_notes include:
"SIZE REASONING: Typical real-world [type] ~= [X x Y x Z] m.
    Current dims [A x B x C] m. Scale s=[...]. Final dims [P x Q x R
    ] m."

------------------------------------------------------------------------

STEP 4: Mass calculation (required)
------------------------------------------------------------------------

For each link with volume data:
mass = volume_m^3 * density_kg_m^3 * (uniform_scale_factor^3) *
    hollow_factor

Choose hollow_factor based on appearance:
- solid-looking: 0.8-1.0
- moderately hollow/shelled: 0.4-0.8
- very thin shell: 0.2-0.5

Then adjust slightly based on role:
- base/static parts: should carry most mass
- moving parts: lighter than base
- tiny decorative parts: 0.001-0.01 kg

In validation_notes, show at least a few explicit mass computations.


------------------------------------------------------------------------

STEP 5: Inertia (required)
------------------------------------------------------------------------

Generate inertia for each link (positive definite), using a simple
    shape approximation
(box/cylinder) from scaled dimensions or bounding extents. Set off-
    diagonals to 0 unless
strongly justified. Ensure magnitudes scale consistently with mass
    and size.

------------------------------------------------------------------------

STEP 6: Joint dynamics (PASSIVE by default, Isaac Sim / PhysX)
------------------------------------------------------------------------

Most objects are NOT driven. Therefore:
- Treat joints as PASSIVE by default (no servo/drive).
- PASSIVE joints should primarily use damping and (optional)
    friction.
- DO NOT use stiffness as a generic parameter.
```

```
  - Only include stiffness > 0 if user explicitly requests spring-
      return behavior OR you have
    strong evidence of a spring mechanism AND you intentionally
       model it as a spring.

### Units (SI)
Revolute/continuous:
- damping: N*m*s / rad
- friction: N*m
- stiffness (ONLY if modeling a spring): N*m / rad

Prismatic:
- damping: N*s / m
- friction: N
- stiffness (ONLY if modeling a spring): N / m

### Passive damping policy
Revolute PASSIVE defaults:
- damping: 0.05-0.5 (use ~0.15)
- friction: 0.005-0.05
- stiffness: 0

Prismatic PASSIVE defaults:
- damping: 0.5-5.0 (use ~1.5)
- friction: 0.1-2.0
- stiffness: 0

Escalation:
- Increase damping 2x-5x if oscillation persists
- Add small friction only if needed
- Avoid excessive friction (stick-slip)

----------------------------------------------------------------------

STEP 7: Object state / initial joint positions
----------------------------------------------------------------------

Only include initial_joint_positions if the user explicitly
    requests a state.
Do NOT include otherwise.

----------------------------------------------------------------------

STEP 8: Contact properties (optional)
----------------------------------------------------------------------

Only include global material_properties if needed for stable
    contact.

======================================================================

Object Information:
{object_info}
======================================================================


## Output Format (JSON ONLY)
[JSON schema exactly as specified in the paper]
```

N Inverse-Variance Weighting in Phys2Real and VLM-IVW

In the original Phys2Real framework, VLM priors are fused with interaction-based estimates produced by Rapid Motor Adaptation (RMA) using inverse-variance weighting. The RMA estimate is obtained from an ensemble of adaptation models trained on histories and actions generated by a physics-conditioned reinforcement learning policy in simulation. Training such a policy requires prior knowledge of the downstream task and its associated reward function. Since our objective is to evaluate general-purpose articulated asset quality independent of any specific task or environment, incorporating interaction-based adaptation would introduce task-specific inductive bias and leak information about the evaluation setting. We therefore omit RMA-style interaction models for VLM-IVW and rely exclusively on VLM-based physical parameter estimation. In this setting, the uncertainty-aware fusion mechanism of Phys2Real reduces to aggregation over VLM predictions alone. While this corresponds to mean aggregation in the original implementation, accordingly, we apply inverse-variance weighting across multi-view, multi-query VLM outputs, treating them as an uncertainty-annotated ensemble.

O Articulation-Centric Reward Function

We use a shared dense reward across articulation tasks, instantiated with fixed coefficients in code. The reward is intentionally staged: it first brings the end-effector (EE) to the object, then rewards articulated joint progress only when the EE is sufficiently close, while adding pose and gripper regularization terms that bias the policy toward physically meaningful interaction.

Geometric reaching terms. Let $p_t^{\text{ee}} \in \mathbb{R}^3$ and $p_t^{\text{obj}} \in \mathbb{R}^3$ denote EE and object positions in world coordinates at timestep t. Define

$$d_t = \|p_t^{\text{ee}} - p_t^{\text{obj}}\|_2, \qquad d_t^{xy} = \|(p_t^{\text{ee}} - p_t^{\text{obj}})_{xy}\|_2.$$

The first reward term is a smooth 3D reaching objective,

$$r_t^{\text{reach}} = 1.0 \cdot \left(1 - \tanh\left(\frac{d_t}{0.1}\right)\right),$$

which provides a dense attraction toward the object in full Euclidean space. The second term emphasizes planar alignment,

$$r_t^{\text{reach_xy}} = 5.0 \cdot \left(1 - \tanh\left(\frac{d_t^{xy}}{0.01}\right)\right),$$

with a tighter distance scale and larger weight. In practice, this encourages fast horizontal centering over the interaction region before fine articulation.

Articulation progress term (proximity-gated). For articulated objects with d joints, each joint coordinate is normalized by runtime joint limits:

$$x_{t,i} = \frac{q_{t,i} - l_i}{u_i - l_i}, \quad i = 1, \ldots, d.$$

Let $x_{\text{goal}} \in [0,1]^d$ be the configured normalized target (`object_joint_goal`), and define

$$e_t^{\text{joint}} = \|x_t - x_{\text{goal}}\|_1.$$

The articulation reward is

$$r_t^{\text{joint}} = \mathbf{1}[d_t < 0.05] \cdot 100.0 \cdot \exp(-40.0\, e_t^{\text{joint}}).$$

This term is *explicitly gated* by proximity ($d_t < 5$ cm): the policy receives articulation-progress reward only when the EE is near the object, which suppresses spurious reward from far-field motion. If the object has no articulated joints, this term is set to zero.

Regularization penalties. Three penalties are added as negative rewards:

$$r_t^{\text{pen_grip}} = -2.0\, o_t, \tag{6}$$

$$r_t^{\text{pen_down}} = -1.0\, a_t, \tag{7}$$

$$r_t^{\text{pen_xy}} = -2.0\, d_t^{xy}. \tag{8}$$

Here $o_t \in [0, 1]$ is the gripper-open fraction computed from runtime gripper joint limits (0 = fully closed, 1 = fully open). The orientation penalty uses

$$a_t = 1 - \langle z_t^{\text{ee}}, [0, 0, -1] \rangle,$$

where z_t^{ee} is the EE local $+Z$ axis expressed in world coordinates. Thus, $a_t = 0$ indicates perfect downward alignment and larger values penalize upward/sideways wrist configurations. The $r_t^{\text{pen_xy}}$ term adds a linear cost with planar distance, complementing the saturating $\tanh$-based shaping.

Total reward and timestep scaling. The unscaled per-step reward is

$$\tilde{r}_t = r_t^{\text{reach}} + r_t^{\text{reach_xy}} + r_t^{\text{joint}} + r_t^{\text{pen_grip}} + r_t^{\text{pen_down}} + r_t^{\text{pen_xy}}.$$

Implementation then multiplies each component by control timestep Δt and sums:

$$r_t = \Delta t \cdot \tilde{r}_t.$$

This matches the simulator-side reward convention and keeps magnitudes consistent across control rates.

Numerical handling. To ensure stable training and logging, each component and the final reward are sanitized in code via finite-value handling (NaN/Inf replacement). This does not change the reward definition above, but prevents numerical outliers from propagating into optimization statistics.

P PER-MODEL SUCCESS RATE FOR VLAS

Task	Human (GenSim2)	VLM-IVW	Direct VLM	ASSET2SIM (Ours)	Real
Stapler Pressing	0	20	0	0	20
Stapler Grasping	80	60	0	100	100
Box Closing	0	0	0	0	60
Box Grasping	0	0	0	0	100
Toilet Closing	0	60	0	80	80
Microwave Closing	0	0	0	40	80
Laptop Folding	0	20	0	40	60
Bag Grasping	0	0	0	40	100
Suitcase Closing	0	0	0	20	40
Trashcan Closing	0	0	0	20	40
Cabinet Closing	0	20	0	20	60
Average (%)	7.27	16.36	0.00	**32.73**	67.27

Table 10: Per-task simulation success rates (%) for π_0.

Task	Human (GenSim2)	VLM-IVW	Direct VLM	ASSET2SIM (Ours)	Real
Stapler Pressing	0	60	0	40	40
Stapler Grasping	100	100	0	100	100
Box Closing	0	0	0	0	0
Box Grasping	0	20	0	40	100
Toilet Closing	0	0	0	20	60
Microwave Closing	0	0	0	40	100
Laptop Folding	0	40	0	40	80
Bag Grasping	0	0	0	80	80
Suitcase Closing	0	0	0	100	80
Trashcan Closing	0	0	0	40	80
Cabinet Closing	0	0	0	0	20
Average (%)	9.09	20.00	0.00	**45.45**	67.27

Table 11: Per-task simulation success rates (%) for $\pi_{0.5}$.

Task	Human (GenSim2)	VLM-IVW	Direct VLM	ASSET2SIM (Ours)	Real
Stapler Pressing	0	40	0	0	0
Stapler Grasping	20	60	0	0	40
Box Closing	0	0	0	40	80
Box Grasping	0	20	0	0	80
Toilet Closing	0	0	0	0	0
Microwave Closing	0	0	0	0	0
Laptop Folding	0	0	0	20	0
Bag Grasping	0	0	20	0	40
Suitcase Closing	0	0	0	100	40
Trashcan Closing	0	0	0	0	20
Cabinet Closing	0	0	0	60	60
Average (%)	1.82	10.91	1.82	**20.00**	32.73

Table 12: Per-task simulation success rates (%) for GR00T N1.6.

Q PER TASK SUCCESS RATE OF RL EXPERIMENT

RL Tasks	Human (GenSim2)	VLM-IVW	Direct VLM	ASSET2SIM (Ours)
Bag Handle Moving	0	0	0	**100**
Box Closing	0	0	0	**100**
Cabinet Closing	**100**	**100**	**100**	**100**
Laptop Folding	0	**100**	0	**100**
Stapler Pressing	0	11.11	0	**77.78**
Microwave Closing	0	0	0	**100**
Suitcase Closing	0	**100**	0	**100**
Trashcan Closing	0	0	0	**100**
Toilet Closing	0	**100**	0	**100**
Average Success Rate (%)	11.11	45.68	11.11	**97.53**

Table 13: RL policy success rates (%) across articulation tasks for different asset construction methods.

R VLM JUDGE REALISM

Task	ASSET2SIM (Ours)	Human (GenSim2)	VLM-IVW	Direct VLM
Gemini 3.0				
Bag Grasping	**2.35±0.92**	1.85±0.82	1.55±0.99	1.60±0.97
Box Grasping	**3.88±0.54**	2.25±0.78	2.35±0.55	1.92±0.55
Box Closing	1.20±0.40	1.70±1.00	1.40±0.49	**2.37±1.16**
Cabinet Closing	**1.89±0.80**	1.70±0.70	1.23±0.73	1.39±0.72
Laptop Closing	**4.10±0.44**	3.62±0.55	3.24±0.96	3.44±0.67
Microwave Closing	**1.65±0.78**	0.57±0.23	1.27±0.64	1.45±0.79
Stapler Grasping	**3.05±0.69**	2.57±0.80	2.04±0.60	1.40±0.84
Stapler Pressing	**2.40±0.49**	1.90±0.42	2.19±0.54	2.19±0.42
Suitcase Closing	1.70±0.51	0.70±0.24	1.70±0.46	**1.92±0.90**
Toilet Closing	**2.65±0.84**	1.25±0.46	1.15±0.39	1.66±1.05
Trashcan Closing	2.60±0.89	0.75±0.34	0.00±0.00	**2.69±0.73**
Qwen3-VL 32B				
Bag Grasping	**1.90±0.20**	1.00±0.00	0.75±0.34	0.75±0.34
Box Grasping	**4.15±0.32**	1.80±0.40	3.00±0.45	2.15±0.71
Box Closing	1.90±0.49	1.95±0.42	1.50±0.50	**2.20±0.71**
Cabinet Closing	1.94±0.16	2.22±0.71	2.83±1.15	**4.14±0.43**
Laptop Closing	**4.40±0.20**	1.60±0.49	4.00±0.32	1.05±0.27
Microwave Closing	**4.05±0.15**	1.10±0.30	2.10±0.30	1.95±0.35
Stapler Grasping	2.30±0.46	1.70±0.64	**3.10±0.66**	0.65±0.32
Stapler Pressing	2.60±0.44	1.40±0.62	**2.90±0.54**	**2.90±0.66**
Suitcase Closing	**4.05±0.42**	1.60±0.66	3.95±0.42	2.10±0.30
Toilet Closing	**2.60±0.54**	0.90±0.30	**2.60±0.83**	1.45±1.01
Trashcan Closing	**2.65±0.90**	0.00±0.00	0.00±0.00	2.35±0.67

Table 14: Per-task realism scores using VLM-as-a-judge evaluation. Rows correspond to individual tasks, and columns correspond to methods. Scores are reported as mean ± standard deviation over 10 trials on a 0–10 scale. The highest score for each task is bolded.

S HUMAN REALISM SCORE

Table 15: Per-object, per-evaluator human realism scores. Each object is evaluated by 5 independent human raters (evaluators) on a 0–10 scale. The mean across evaluators is reported in the final column; the highest mean score for each object is bolded.

Method	Evaluator 1	Evaluator 2	Evaluator 3	Evaluator 4	Evaluator 5	Mean
Cabinet						
ASSET2SIM (Ours)	9	10	8	10	6	**8.6**
Human (GenSim2)	4	4	5	0	4	3.4
VLM-IVW	4	3	6	2	4	3.8
Direct VLM	3	3	7	0	4	3.4
Microwave						
ASSET2SIM (Ours)	9	8	8	10	9	**8.8**
Human (GenSim2)	1	1	3	0	3	1.6
VLM-IVW	2	1	4	2	6	3.0
Direct VLM	2	1	4	2	6	3.0
Laptop						
ASSET2SIM (Ours)	7	9	8	6	7	**7.4**
Human (GenSim2)	3	1	4	2	3	2.6
VLM-IVW	5	9	8	7	7	7.2
Direct VLM	3	1	4	3	3	2.8
Stapler						
ASSET2SIM (Ours)	3	6	3	5	4	4.2
Human (GenSim2)	2	1	3	4	2	2.4
VLM-IVW	2	9	5	7	6	**5.8**
Direct VLM	1	1	2	0	0	0.8
Trashcan						
ASSET2SIM (Ours)	7	8	8	10	7	**8.0**
Human (GenSim2)	1	0	4	0	0	1.0
VLM-IVW	0	0	0	0	0	0.0
Direct VLM	1	3	5	1	4	2.8
Toilet						
ASSET2SIM (Ours)	8	10	8	8	8	**8.4**
Human (GenSim2)	1	2	4	0	3	2.0
VLM-IVW	8	9	5	10	9	8.2
Direct VLM	2	2	4	1	1	2.0
Bag						
ASSET2SIM (Ours)	2	6	8	7	3	**5.2**
Human (GenSim2)	2	1	3	5	2	2.6
VLM-IVW	2	4	7	5	4	4.4
Direct VLM	2	3	7	4	4	4.0
Suitcase						
ASSET2SIM (Ours)	4	2	6	4	9	**5.0**
Human (GenSim2)	3	1	3	0	1	1.6
VLM-IVW	1	0	2	0	0	0.6
Direct VLM	3	1	4	3	5	3.2
Box						
ASSET2SIM (Ours)	7	7	9	8	8	**7.8**
Human (GenSim2)	2	3	4	1	2	2.4
VLM-IVW	2	3	4	0	5	2.8

Continued on next page

Method	Evaluator 1	Evaluator 2	Evaluator 3	Evaluator 4	Evaluator 5	Mean
Direct VLM	2	3	5	2	5	3.4