ANNOTATEANYTHING: AUTOMATIC ANNOTATION OF 3D ASSETS FOR ROBOT MANIPULATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Simulation enables scalable robot data collection, but 3D assets often lack the interaction annotations needed to specify where and how robots should act. We present **AnnotateAnything**, an automatic framework that converts passive 3D assets into manipulation-ready assets with language, 3D visual, and executable action annotations. A vision-language model supplies interaction priors—functional parts, keypoints, affordance regions, and compatible skills—that are grounded in each asset's 3D geometry. A physics annotation pipeline converts these priors into action candidates through skill-specific generation, geometric optimization, and GPU-parallel validation, retaining multiple solutions in reusable candidate banks. These banks drive an asynchronous parallel simulation data-collection system. Applied to 17,005 rigid, articulated, deformable, and room-scale assets, the framework produces about 100M validated action annotations across nine skill families. On an audited simulation suite, it achieves higher readiness and full-robot execution success than ablations and framework-adapted baselines, as well as higher annotation and collection throughput than the latter. Pretrained policies fine-tuned solely on the generated simulation demonstrations achieve 70.8% mean success across six real-world tasks on three robot platforms, with 20 trials per task. Project page: `https://gilded-cupcake-c582c1.netlify.app/`.

1 INTRODUCTION



Figure 1: Overview of AnnotateAnything, a unified framework that converts raw 3D assets into manipulation-ready annotations across four asset families: rigid objects, articulated objects, deformable objects, and room-scale navigation scenes.

Training general-purpose robot policies requires large-scale and diverse data across objects, embodiments, and tasks (Intelligence et al., 2025; Bjorck et al., 2025). Simulation offers a scalable alternative to real-world collection via automatic data generation (Ye et al., 2025; Geng et al., 2025; Lu et al., 2024a;b), and recent asset collections and simulation benchmarks provide rich geometric

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.

and visual substrates for such data generation (Mittal et al., 2025; Li et al., 2023a). Yet owning a 3D model is not the same as interacting with it: assets rarely specify where to grasp, which parts can move and how, or where insertion or hanging is feasible. The bottleneck is thus not acquiring more assets but automatically converting passive 3D geometry into manipulation-ready assets with actionable interaction annotations.

Producing such annotations automatically raises four challenges. **(C1) Interaction priors:** annotations must reflect how humans use an object, such as grasping a mug by its handle rather than its rim, which geometry alone does not reveal (Tang et al., 2025; Qu et al., 2025). **(C2) Asset-specific executability:** a vision-language model (VLM) can name the part that matters but not the pose, direction, or trajectory that works on a specific asset and robot, so abstract intent must be converted into executable labels (Niu et al., 2026; Deshpande et al., 2026). **(C3) Diversity:** feasible labels should preserve multiple valid solutions rather than collapse to a single canonical one (Wang et al., 2026a; Chen et al., 2026a). **(C4) Scale:** the process must cover many assets and categories without per-asset engineering, once a skill family is supported. Manual annotation addresses C1 and C2 but scales poorly (C4) and typically yields one label per object (C3) (Chen et al., 2025b; Tian et al., 2026); RL-based generation reduces labeling but requires reward design and substantial compute, and its behaviors need not match human priors (Wang et al., 2023b; 2026c).

AnnotateAnything addresses these challenges by combining VLM interaction priors with physics-grounded annotation. A VLM proposes functional parts, keypoints, affordance regions, and the skills each region supports, grounded on the asset's 3D geometry (C1; Sec. 3.1). Skill-specific generators turn each anchor into contacts, poses, and trajectories, which are optimized under contact, collision, and kinematic constraints and filtered by GPU-parallel physics validation (C2; Sec. 3.2.2). The result is a *candidate bank* of validated solutions per anchor and skill (C3; Sec. 3.2.1), which a data-collection planner queries, falling back to another candidate when one is infeasible for the current scene or robot (Sec. 4.1). The same generators apply unchanged to new assets of a supported skill family (C4), and the labels can be repurposed as perception and reasoning supervision (Sec. 4.2).

On an audited suite of 1,160 assets and 3,698 asset–skill pairs, grounded priors match manual references for 87% of parts and 85% of affordance regions and improve on direct VLM prediction by 21 points in human-rated quality (C1); 91.9% of ready pairs execute on the complete robot versus 67.5% for the strongest ablation (VL-only; C2); a median asset retains 176 distinct grasp modes, and one fallback raises dexterous-grasp execution from 87.2% to 93.9% (C3); and the generators cover 17,005 assets from nine sources, yielding about 100M validated annotations (C4). The bank also yields higher readiness and full-robot execution than framework-adapted pipelines and modestly higher than a protocol-matched dexterous-grasp baseline, and pretrained policies fine-tuned solely on bank-driven simulation data achieve 70.8% mean success on six real-world tasks across three robot platforms; on the three tasks MoMaGen supports, they reach 71.7% versus 38.3% with MoMaGen-generated data under the same policy and protocol. Our contributions are:

- a function-conditioned, one-to-many **candidate-bank representation** linking grounded interaction priors to validated, executable action annotations, rather than a single verified grasp set per object;
- an automatic **grounding–generation–validation pipeline** for nine skill families (single-arm, dexterous, bimanual, and bimanual dexterous grasping; articulation; insertion; hanging; deformable manipulation; navigation) on four asset types and embodiments from parallel-jaw grippers and dexterous hands to dual-arm, mobile, and humanoid robots;
- an **audited evaluation** of semantic quality, full-robot executability, diversity, and scale against ablations and framework-adapted and protocol-matched baselines, with real-robot experiments.

2 Related Work

Table 1 summarizes capabilities explicitly supported or reported by each system, not a protocol-matched benchmark; InternData-A1 is primarily a pretraining dataset, and RoboTwin2.0 and GenieSim 3.0 are task or data-generation platforms without our per-asset annotation interface.

Automatic data collection for robot learning. Large and diverse robot data drive generalist policies and world models (Intelligence et al., 2026; Ye et al., 2026; Zheng et al., 2026), from whole-body humanoid control (Ben et al., 2025; Luo et al., 2025) to articulated, garment, and assembly manipulation (Chen et al., 2026c; Wu et al., 2025; Shen et al., 2025; Li et al., 2024b; 2026b; Wu et al.,

Table 1: Comparison of automatic data collection and annotation generation methods. ✔: explicitly supported or reported; ◯: partially supported; ✘: not supported or not reported.

Method	Skill / Interaction Coverage							Scenario / Embodiment			Pipeline Property		
	Grasp	DexGrasp	BiGrasp	BiDexGrasp	Articulation	Deformable	Navigation	Tabletop	Mobile	Humanoid	Physics Validation	Parallel Generation	Data Augmentation
InternData-A1 (Tian et al., 2026)	✔	✘	✔	✘	◯	◯	✘	✔	✘	✘	✘	✘	◯
RoboTwin2.0 (Chen et al., 2025b)	✔	✘	✔	✘	◯	✘	✘	✔	✘	✘	◯	✘	◯
MolmoBot (Deshpande et al., 2026)	✔	✘	✘	✘	◯	✘	◯	✔	◯	✘	✔	✘	✔
RoboGen (Wang et al., 2023b)	✔	✘	✘	✘	✔	✘	✘	✔	◯	✘	◯	✔	✘
GenManip (Gao et al., 2025)	✔	✘	✘	✘	◯	✘	✘	✔	✘	✘	✘	✘	✘
GenieSim 3.0 (Yin et al., 2026)	✔	◯	◯	◯	◯	✘	✘	✔	◯	✘	✘	✘	✘
AnnotateAnything	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔

2024; Ning et al., 2023; Li et al., 2023b). Automatic collection mainly relies on annotations (Tian et al., 2026; Chen et al., 2025b; Mu et al., 2025; Chen et al., 2026b), augmentation of human demonstrations (Mandlekar et al., 2023; Li et al., 2026a), or RL (Wang et al., 2026c; 2023b; Yang et al., 2026; Wan et al., 2023; Wang et al., 2023a; Bao et al., 2023), requiring human demonstrations, manual task design, reward engineering, or costly training; AnnotateAnything instead turns raw 3D assets into reusable, skill-consumable annotations.

Automatic annotation for manipulation. Prior automatic annotation targets task-specific labels such as grasps (Deng et al., 2025), contacts, affordances, or waypoints, limiting object diversity, embodiment coverage, or task generality (Gao et al., 2025; Chen et al., 2025a; Lin et al., 2026; Wang et al., 2025; Bao et al., 2023); the closest, ManiTwin (Wang et al., 2026a), attaches simulation-verified grasp proposals and functional annotations to 100K rigid objects, whereas we also cover articulated, deformable, and room-scale assets and nine skill families. VLM-based systems infer articulation models (Le et al., 2025), generate demonstrations (Duan et al., 2024), or predict affordance points or keypoint constraints (Yuan et al., 2024; Liu et al., 2024; Huang et al., 2024), but do not produce physically validated, per-asset banks of multiple executable solutions. BODex (Chen et al., 2025a) scales dexterous-grasp synthesis and serves as our protocol-matched DexGrasp baseline (Sec. 5.4). Additional related work is in Appendix I.

3 Method

3.1 Grounding Hierarchical Interaction Priors

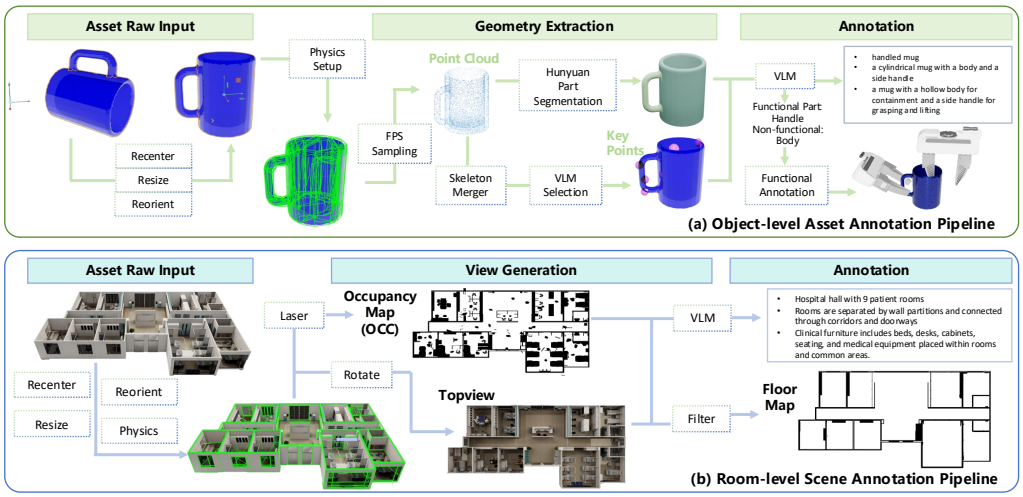


Figure 2: Hierarchical visual-language annotation pipeline. AnnotateAnything generates asset-level language and visual annotations, including descriptions, keypoints, part segmentation, and affordance regions, and composes them into room-level annotations.

Language priors. For each object, we render eight views and query Qwen3.5-27B (Qwen Team, 2026) for a phrase, a functional sentence, and a part-aware paragraph summarizing identity, manip-

ulable parts, and feasible interactions; for rooms, a labeled top-view occupancy map additionally yields layout, zone, and scene-context descriptions (App. C).

3D anchors. We fuse rendered RGB-D views into a point cloud and obtain native 3D part proposals with P3-SAM and X-Part (Ma et al., 2025; Yan et al., 2025) (Fig. 2). The VLM names each part, marks task-relevant parts, and grounds affordance regions on them, assigning each anchor h a functional affordance $\phi(h)$ (e.g., a mug handle for grasp-to-pour, a cabinet handle for pull-to-open). Keypoints are selected by the VLM among Skeleton Merger (Shi et al., 2021) candidates on the FPS-downsampled cloud, and part regions are refined for consistency with the fused point cloud (the 3D refinement ablated in Table 2). Compatible skills $\mathcal{S}(h)$ are inferred from $\phi(h)$, language priors, geometry, and task constraints, and determine the generators invoked in Sec. 3.2.2. Rooms add multi-height occupancy and floor-plan maps (App. D).

3.2 Physics-based Action Annotation

3.2.1 Candidate-Bank Schema

To preserve diversity while keeping functional consistency, annotations form a candidate bank over the grounded anchors $\mathcal{H}(\mathcal{A}) = \mathcal{P}(\mathcal{A}) \cup \mathcal{K}(\mathcal{A}) \cup \mathcal{R}^{\text{aff}}(\mathcal{A}) \cup \mathcal{R}^{\text{scene}}(\mathcal{A})$ of asset $\mathcal{A}$ rather than one canonical action per object or part, where $\mathcal{P}$, $\mathcal{K}$, $\mathcal{R}^{\text{aff}}$, and $\mathcal{R}^{\text{scene}}$ denote part regions, semantic keypoints, affordance regions, and scene-level regions; local anchors define manipulation targets and scene-level anchors define navigation or approach targets. For every anchor h and compatible skill s, we store

$$\mathcal{B}_{h,s} = \{a_{h,s}^{(i)}\}_{i=1}^{N_{h,s}}, \quad s \in \mathcal{S}(h), \qquad a_{h,s}^{(i)} = (s, o(h), h, \phi(h), x^{(i)}, \theta^{(i)}, \tau^{(i)}, v^{(i)}, d^{(i)}).$$

$N_{h,s}$ counts retained candidates, $o(h)$ is the object or scene, $x^{(i)}$ the target (e.g., contact point or base pose), $\theta^{(i)}$ skill parameters, $\tau^{(i)}$ optional waypoints, $v^{(i)}$ validation metadata, and $d^{(i)}$ a diversity descriptor (App. E.1), so each anchor has multiple validated, function-conditioned solutions.

3.2.2 Generation, Optimization, Validation, and Augmentation

Candidate targets. For skill s (Fig. 3), sampling uses only anchors whose affordance $\phi(h)$ fits s (e.g., handles for opening, free-space regions for approach). Targets are drawn by farthest-point sampling and filtered by curvature, visibility, clearance, reachability, and traversability.



Figure 3: Physics-based action annotation pipeline. Grounded visual-language priors are converted into executable action annotations through candidate target generation, trajectory generation, trajectory optimization, physics validation, and physics-aware augmentation.

Trajectory generation. Each target seeds a template from one of four generator families, conditioned on object geometry (grasping, folding, flinging), object motion (articulation), a direction (insertion, hanging, placement), or the scene (A* paths to interaction-ready base poses).

Trajectory optimization. Parameters and waypoints are refined under task, contact, collision, kinematic, and smoothness terms, $(\theta_*, \tau_*) = \arg\min_{\theta,\tau} \sum_k \lambda_k E_k$, with skill-specific active terms (e.g., hand pose, finger joints, and contact assignment for dexterous grasps; App. E.4).

Physics validation. Candidates are simulated in parallel in Isaac Sim under disturbances, including varied gravity magnitude or direction for grasps. Object-centric skills are validated with a floating gripper or hand, which filters candidates without fixing an arm pose; garment and other embodiment-dependent skills use the full manipulator. Full-robot feasibility is tested at use time (Sec. 4.1).

Physics-aware augmentation. Validated candidates are expanded by bounded perturbations that preserve the affordance $\phi(h)$ and by annotated object symmetries (e.g., rotating grasps around a bottle's axis), and each variant is kept only if it passes geometry and physics rechecks.

4 Downstream Tasks

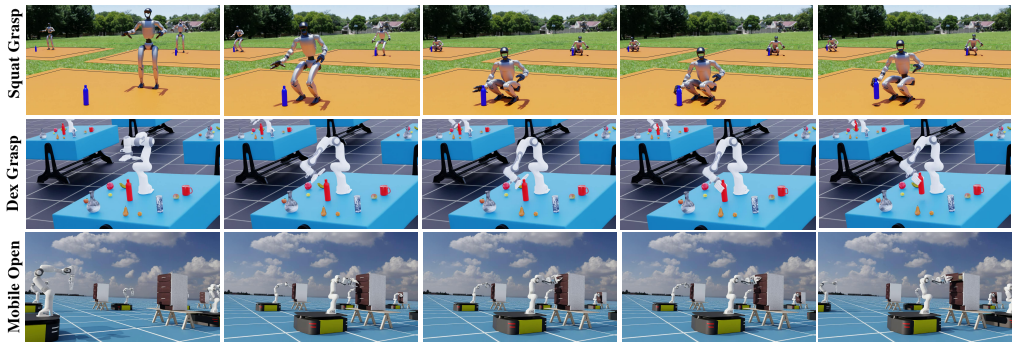


Figure 4: Representative atomic skills: covering tabletop, bimanual, whole-body, dexterous, and mobile manipulation.

4.1 Large-scale Robot Data Collection

Atomic skills and parallel rollouts. An atomic-skill library (Fig. 4) consumes the stored parameters (e.g., grasp poses, waypoints, insertion directions, navigation goals) and composes skills into long-horizon tasks. Heterogeneous parallel environments independently sample assets, tasks, object poses, and scene layouts, with asynchronous cuRobo-v2 planning (Sundaralingam et al., 2026; 2023) and domain randomization over pose, lighting, material, camera, and scene configuration.

Trajectory validation. Per randomized scene, candidates are retrieved in their generator's ranked order (baseline candidates in Sec. 5.4 keep their native ranking); for each, we solve goal-set IK for the complete robot and keep the lowest-cost feasible solution, trying subsequent candidates within a small retry budget (one retry by default, two for complex multi-step tasks); candidates that repeatedly fail validation are removed, and Exec. in Sec. 5.1 uses one fallback for all skills.

4.2 Annotation-derived Downstream Applications

Keypoint and affordance supervision. Keypoints and validated actions provide supervision projectable to images or point clouds (Chen et al., 2026c; Wu et al., 2024; Ning et al., 2023; Wu et al., 2025; Li et al., 2024b; 2023b; Hou et al., 2024; Yang et al., 2026). Beyond the direct annotation accuracy in Sec. 5.2, we keep established pipelines fixed and replace only their supervision, input, or reference annotations with ours: SAMPart3D (Yang et al., 2024) and the released LASO pipeline (Li et al., 2024a) supervised by our labels reach 93–95% of their reference scores on matched PartNeXt and LASO subsets, and substituting our keypoints for the KeypointNet references in Skeleton Merger's (Shi et al., 2021) evaluation retains 96% of the reference score (App. G.2), i.e., usable with modest gaps. As illustrative repurposings, rollout labels and scene annotations also

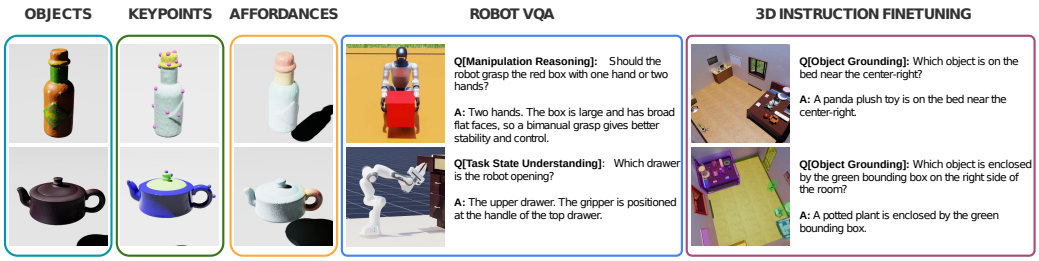


Figure 5: Downstream applications supported by AnnotateAnything. The same visual-language-action annotations can be converted into supervision for affordance and keypoint detection, robot reasoning VQA, and 3D VLM instruction finetuning. Affordance and keypoint supervision is quantitatively evaluated, while VQA and instruction tuning are presented as illustrative repurposings.

yield robot-reasoning VQA and 3D VLM instruction data (Zhang et al., 2026a;b; Pan et al., 2025) (App. G.3–G.4).

5 Experiments

5.1 Experimental Setup

We evaluate on an audited suite of 1,160 assets and 3,698 valid asset–skill pairs covering nine skill families and four asset types, sampled before any annotation outcome was computed and checked only for asset validity, simulator compatibility, and skill applicability (App. H.1); visual-language quality uses a separate balanced set of 360 manually referenced examples with part, affordance, and keypoint references derived from PartNet, PartNeXt, KeypointNet, and LASO (Mo et al., 2019; Wang et al., 2026b; You et al., 2020; Li et al., 2024a) for objects and manually annotated for garments and rooms (App. H.1). *Pass* is the fraction of generated candidates passing physics validation (averaged over pairs), *Ready* the fraction of pairs with at least one retained candidate, and *Exec.* the full-robot success of ranked retrieval with one fallback (Sec. 4.1) over ready pairs. Human scores (0–100) average five blinded non-author raters per item over three generation seeds (App. H.9).

5.2 Interaction Priors Are Meaningful and 3D-Grounded (C1)

We compare the full grounding pipeline with **Direct VLM**, which predicts the annotation bundle directly from images, **Aff.-only + heur.**, and **w/o 3D refine** (Table 2).

Table 2: Visual-language and visual annotation quality on the separate 360-example manual-reference set. The left block reports mean ± standard error human ratings for complete visual-language annotation bundles. The right block reports automatic matching scores: Part/Afford. are label-compatible bidirectional mean IoU and Keypoint is PCK@0.05 (App. H.3).

	Visual-Language Bundle Quality					Visual Annotation Quality		
Method	Semantic	3D Grounding	Coverage	Action	Overall	Part	Afford.	Keypoint
Direct VLM	85.1±0.9	62.5±1.4	69.6±1.3	65.2±1.3	70.6±1.2	72.8	68.5	70.2
Aff.-only + heur.	75.7±1.2	69.8±1.3	72.4±1.3	66.7±1.4	71.2±1.3	70.5	79.3	68.8
w/o 3D refine	87.5±0.8	73.7±1.2	78.1±1.1	75.4±1.2	78.7±1.1	82.6	78.1	80.8
Ours	**93.7±0.6**	**90.2±0.7**	**90.8±0.7**	**91.9±0.6**	**91.7±0.6**	**91.5**	**89.0**	**90.3**

The full pipeline scores best on every dimension; Direct VLM does relatively well on semantics but much worse in 3D grounding and actionability, supporting explicit 3D grounding, and the overall gain over the strongest ablation is 13.0 points (95% CI $[11.4, 14.5]$). Interface-level checks agree: 87.4% of part regions and 84.8% of affordance regions match manual references at label-compatible IoU ≥ 0.5, keypoints reach 90.3% PCK@0.05, and 83.3% of anchors $(300/360)$ receive exactly the correct skill set. On matched external subsets (re-run, not copied from the original papers), our part annotations reach 44.1 object-macro mIoU on PartNeXt (Wang et al., 2026b) (SAMPart3D (Yang et al., 2024)

46.70, P3-SAM (Ma et al., 2025) 41.01) and our affordances $41.0/54.3$ mIoU/F1 on LASO (Li et al., 2024a) (released LASO $43.6/57.2$, Direct VLM $35.8/48.9$), slightly below the strongest specialized models and above weaker baselines. VLM outputs with parsing or segmentation-linking failures ($94/3{,}240$) are rejected, and residual semantic errors remain (App. H.3).

5.3 Annotations Are Executable on the Full Robot (C2)

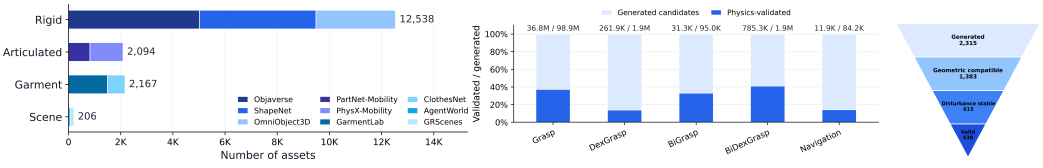
Table 3 reports annotation quality and rollout collection per skill family and for ablations. Geometry-only removes visual-language priors, VL-only instantiates skill candidates directly from grounded anchors, without optimization and physics validation, and w/o phys. validation removes only the validator (App. H.8).

Table 3: Physics-grounded action annotation quality and annotation-enabled rollout collection on the audited evaluation suite. Upper: atomic-skill results of the full pipeline; lower: aggregate baselines and ablations. Human scores are mean ± standard error on a 0–100 scale. Pass/Ready denotes candidate-level physics pass rate / asset–skill-pair readiness rate, while Exec. denotes execution on the complete task-compatible robot. †Before filtering, its candidates are those of the full pipeline, so Pass/Ready are not repeated; VL-only Pass/Ready are computed post hoc with the same validator.

Setting	Action Annotation Quality				Annotation-enabled Rollout Collection			
	Pass/Ready (%/%)	Exec. (%)	Ann./min	Ann. Human	Data Succ. (%)	Traj./h	Att./Succ.	Traj. Human
Atomic-skill results of the full pipeline								
Grasp	60.3 / 96.4	98.1	800	89.8±1.0	90.5	300	1.11	90.8±0.9
DexGrasp	43.8 / 88.7	93.9	200	86.2±1.2	86.7	170	1.15	86.5±1.2
BiGrasp	38.5 / 90.5	95.0	105	87.5±1.2	87.8	145	1.14	86.8±1.3
BiDexGrasp	27.4 / 82.4	88.5	95	84.1±1.3	82.4	115	1.21	85.0±1.1
Articulation	41.6 / 89.6	93.2	110	85.6±1.3	88.6	150	1.13	88.1±1.1
Insertion	33.2 / 84.1	89.3	70	82.9±1.3	84.2	125	1.19	81.5±1.4
Hanging	31.5 / 81.7	88.7	60	83.5±1.2	83.4	110	1.20	83.6±1.3
Deformable	30.1 / 78.9	84.1	45	80.7±1.4	80.3	90	1.25	81.9±1.3
Nav. / Approach	58.7 / 94.2	96.5	300	89.3±1.3	90.1	280	1.11	88.6±1.1
Aggregate baselines and ablations								
No annotation / random	–	–	–	–	24.0	55	4.17	52.0±0.7
Geometry-only	28.1 / 58.5	58.4	150	61.7±0.6	48.2	90	2.07	60.2±0.7
VL-only	31.7 / 65.0	67.5	130	70.8±0.6	61.8	115	1.62	69.2±0.6
w/o phys. validation†	–	63.2	350	64.5±0.6	57.6	110	1.74	65.6±0.6
Ours full	**40.6 / 87.4**	**91.9**	198	**85.5±0.4**	**86.0**	**165**	**1.16**	**85.9±0.4**

The full pipeline attains the highest Pass, Ready, Exec., and rollout success in aggregate (App. H.8); geometry-only candidates lack semantic precision, VL-only ones lack executability, and removing validation reduces execution and rollout reliability, while the validated bank raises throughput (Traj./h) over annotation-free and VL-only variants. Counting unready pairs as failures, Ready×Exec. is 80.3% for ours versus 43.9% for VL-only. Human-score gains over the strongest ablation are 14.7 ($[13.7, 15.7]$) points for action annotations and 16.6 ($[15.7, 17.6]$) for rollout trajectories.

Floating-to-full-robot audit. For DexGrasp (10 candidates for each of 520 pairs), floating-validated candidates succeed on the complete G1–Dex3-1 robot for $402/461$ ready pairs (87.2%) at the top rank and for $433/461$ (93.9%, its Exec. in Table 3) with one fallback (83.3% of all attempted pairs).



(A) Processed asset pool by asset type and source.

(B) Physics-validated candidates by skill family, with raw counts shown above each bar.

(C) Conversion funnels

Figure 6: **Annotation coverage over heterogeneous assets.** Condensed coverage statistics: (A) processed asset pool grouped by asset type and source family, (B) pooled physics-validation outcomes for five of the nine skill families, and (C) conversion before augmentation, from generated to geometrically compatible, disturbance-stable, and valid candidates (mean counts per audited pair).



Figure 7: Atomic skill taxonomy with qualitative annotation examples for the 18 atomic skills.

5.4 Comparison with Existing Pipelines

Because existing systems differ in native outputs and skill coverage, we convert each method's closest compatible output into our candidate interface and evaluate it with the same validator and rollout protocol, recomputing **Ours (matched)** on the same skill families (Table 4). ManiTwin grasps are stored directly as candidates, RoboGen task targets or scripted parameters and GenManip targets and waypoints are converted into candidates, MolmoBot interaction proposals anchor the closest skill templates, and MOPS (Li et al., 2025) affordance masks anchor heuristic action sampling (App. H.8). Our pipeline improves readiness, execution, human-rated quality, rollout success, and throughput on every matched subset (e.g., paired ManiTwin gains of $+4.5$ pp $[3.6, 5.4]$ in Pass, $+13.8$ $[8.1, 19.5]$ in Ready, and $+10.7$ $[8.8, 12.6]$ in Exec.); since conversion may lose information and the systems target different outputs, these are estimates rather than reproductions of those systems.

Table 4: **Framework-adapted external baseline estimates** under the common validation and rollout protocol; conversions to our candidate interface are summarized in Sec. 5.4 and detailed in App. H.8 (Table 42). Metrics are as in Table 3; Att./Succ. is attempts per successful trajectory. Each *Ours (matched)* row aggregates the listed skill families (Att./Succ. $= 100/$Data Succ.); *Ours full* is an all-family reference, not compared with subset-specific baselines.

Method	Evaluation Scope	Pass/Ready	Exec.	Ann./min	Ann. H	Data Succ.	Traj./h	Att./Succ.	Traj. H
Action-producing baselines with skill-family-matched rows									
ManiTwin-adapted	Grasp	55.8 / 82.6	87.4	95	81.6	74.0	135	1.35	79.6
Ours (matched)	Grasp	**60.3 / 96.4**	**98.1**	**800**	**89.8**	**90.5**	**300**	**1.11**	**90.8**
RoboGen-adapted	Grasp, Articulation	22.5 / 45.2	54.8	18	59.4	39.5	42	2.53	57.5
Ours (matched)	Grasp, Articulation	**51.0 / 93.0**	**95.7**	**455**	**87.7**	**89.6**	**225**	**1.12**	**89.4**
MolmoBot-adapted	Grasp, Articulation, Nav.	29.4 / 58.0	64.1	34	68.5	53.5	82	1.87	67.0
Ours (matched)	Grasp, Articulation, Nav.	**53.5 / 93.4**	**95.9**	**403**	**88.2**	**89.7**	**243**	**1.11**	**89.2**
GenManip-adapted	Grasp, Articulation, Insertion	27.8 / 55.4	61.5	28	64.0	50.2	74	1.99	62.0
Ours (matched)	Grasp, Articulation, Insertion	**45.0 / 90.0**	**93.5**	**327**	**86.1**	**87.8**	**192**	**1.14**	**86.8**
Perception-derived heuristic; no directly matched action-annotation row									
MOPS-affordance + heuristic	Visual affordance	–	–	–	71.0	42.6	60	2.35	63.0
All-family reference									
Ours full	All skill families	40.6 / 87.4	91.9	198	85.5	86.0	165	1.16	85.9

Protocol-matched DexGrasp baseline. With the same Dex3-1 hand, 520 pairs, 10 candidates per pair, validator, and G1 execution, and no output conversion, BODex (Chen et al., 2025a) reaches $40.1/84.8/90.2\%$ Pass/Ready/Exec. versus $43.8/88.7/93.9\%$ for AnnotateAnything; over all 520 pairs, 83.3% versus 76.5% execute ($+6.7$ pp, 95% CI $[1.9, 11.6]$).

5.5 Diversity and Scale (C3, C4)

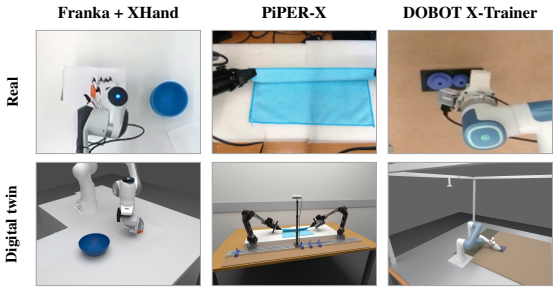
We process 17,005 assets from nine source families and retain about 100M validated action annotations across the nine skill families (Fig. 7), where one annotation is one validated candidate for an anchor–skill pair; the count includes directly generated candidates and re-validated perturbation- and symmetry-derived variants, and excludes failed raw proposals (Fig. 6; App. H.2). The count measures validated instances, not distinct strategies: after merging candidates within 2 cm/10°

(symmetry-equivalent modes not merged), the median asset retains 176 (Grasp), 153 (DexGrasp), and 21 (Articulation) distinct modes, and one fallback raises DexGrasp execution from 87.2% to 93.9%.

5.6 Real-World Experiments

Setup. We test whether our simulation data from digital twins support policy learning and zero-shot sim-to-real deployment, i.e., without task-specific real-world demonstrations, real-world fine-tuning, or test-time adaptation, on three platforms (Fig. 8): a Franka arm with an XHand dexterous hand, a PiPER-X ALOHA-style system with two leader and two follower arms (policies command only the follower arms), and a DOBOT X-Trainer dual-arm platform. The six tasks are dexterous block-to-bowl pick-and-place and toast insertion (Franka–XHand), bimanual towel folding and peg insertion (PiPER-X), and gear assembly and USB insertion (X-Trainer). In each digital twin (reconstructed real-to-sim, manually reviewed; App. H.7), our annotations drive the collection of 200 demonstrations per task (Sec. 4.1), on which a pretrained $\pi_{0.5}$ policy (Intelligence et al., 2025) is fine-tuned with domain randomization. Each task is evaluated in 20 independently reset real-robot trials.

(a) Physical platforms and digital twins



Top: real hardware. Bottom: corresponding digital-twin scenes.

(b) Zero-shot real-world evaluation

200 sim. demos / task; 20 real trials / task

Dex. Pick-and-Place		Towel Folding		Gear Assembly	
Franka + XHand		PiPER-X		X-Trainer	
Ours+$\pi_{0.5}$	**16/20**	**Ours+$\pi_{0.5}$**	**15/20**	**Ours+$\pi_{0.5}$**	**14/20**
Ours+ACT	10/20	Ours+ACT	10/20	Ours+ACT	8/20
MoMaGen+$\pi_{0.5}$	n/a	MoMaGen+$\pi_{0.5}$	n/a	MoMaGen+$\pi_{0.5}$	6/20
Toast Insertion		**Peg Insertion**		**USB Insertion**	
Franka + XHand		PiPER-X		X-Trainer	
Ours+$\pi_{0.5}$	**11/20**	**Ours+$\pi_{0.5}$**	**15/20**	**Ours+$\pi_{0.5}$**	**14/20**
Ours+ACT	9/20	Ours+ACT	10/20	Ours+ACT	9/20
MoMaGen+$\pi_{0.5}$	n/a	MoMaGen+$\pi_{0.5}$	9/20	MoMaGen+$\pi_{0.5}$	8/20

k/20 successes; n/a: task not supported by MoMaGen.

Figure 8: Zero-shot sim-to-real evaluation on three robot platforms and six tasks. **Left:** physical platforms (top) and their digital twins (bottom). **Right:** successes out of 20 real-world trials per task. Policies ($\pi_{0.5}$ or ACT) are trained on 200 simulation demonstrations per task without real-world adaptation; MoMaGen+$\pi_{0.5}$ differs from Ours+$\pi_{0.5}$ only in the data source (n/a: task unsupported).

Results. Mean success is 70.8% (85/120), ranging from 11/20 (toast insertion) to 16/20 (dexterous pick-and-place; Fig. 8). To separate the data from the policy-training pipeline, we keep the policy, digital twin, number of demonstrations, domain randomization, and protocol fixed and replace only the data source with MoMaGen (Li et al., 2026a), which requires a human source demonstration with subtask annotations per task and applies to our three rigid parallel-gripper tasks: it reaches 38.3% (23/60) versus 71.7% (43/60) with ours (Fisher's exact test $p < 0.001$), whereas our data require no human demonstrations. Keeping our data but training ACT (Zhao et al., 2023) from scratch yields 46.7% (56/120), showing that the data remain usable without large-scale policy pretraining; the gap to $\pi_{0.5}$ is consistent with the benefit of that pretraining. Replacing only our annotations with VL-only candidates that must pass the same validation during collection (VL-only+val.) gives similar success (81/120) but takes 2.7–5.5× longer to collect the 200 demonstrations, counting validation time for both, so in this setting our optimized bank mainly lowers collection cost (App. H.7, which also reports a 540-trial single-arm study).

6 Conclusion

AnnotateAnything converts passive 3D assets, through grounded VLM priors, skill-specific generation, and physics validation, into diverse, executable candidate banks for scalable robot data collection and zero-shot sim-to-real transfer. **Limitations.** Annotation quality is bounded by the VLM and part segmenter (13–15% of part and affordance regions disagree with manual references); physics validation uses simplified contact and friction models; each new skill family requires a generator and validator; and real-robot policies are trained per task or object (20 trials each). Future work will study how bank diversity, beyond enabling fallback, affects policy learning and cross-object generalization.

ETHICS STATEMENT

AnnotateAnything aims to reduce manual annotation work for robot learning. Generated annotations can inherit biases and semantic errors from upstream models and asset coverage, and simulation validation does not guarantee safe physical deployment. Appendix J discusses these limitations, broader impacts, and source-specific licensing. Raw assets are redistributed only where permitted; otherwise, derived annotations are accompanied by source mappings and acquisition instructions. Human evaluation uses 15 non-author volunteers (five per item), who gave informed consent and received no compensation, to assess generated annotations and trajectories under the blinded protocol in Appendix H.9; no personal data are collected, and ratings are reported in aggregate. These considerations follow the principles of the ICLR Code of Ethics.

REPRODUCIBILITY STATEMENT

Appendices C–F provide prompts, output schemas, geometric processing, action generation, and validation criteria. Appendix G describes annotation consumption and downstream evaluation. Appendix H documents evaluation sampling, baselines, human-rating protocols, generation seeds, uncertainty estimation, and timing boundaries; Appendix J records hardware, preprocessing, provenance, and licensing. An anonymous code snapshot and representative annotation assets are provided with the supplementary materials; videos and demo assets are also available on an anonymous project page (`https://gilded-cupcake-c582c1.netlify.app/`).

AI USE STATEMENT

We used generative AI tools in the following ways. AI coding assistants helped implement parts of the pipeline and evaluation code, and all AI-assisted code was reviewed and checked by the authors. Large language models provided feedback on the experimental design and the interpretation of results (e.g., the choice of matched real-world baselines) and helped compute statistical tests and confidence intervals, which the authors checked against the raw counts. A vision-language model (Qwen3.5-27B) is also a component of the proposed method: it produces language descriptions and grounding priors (Sec. 3.1), and VLM-assisted paraphrasing diversifies the wording of the illustrative VQA and instruction data (Appendix G). For the real-world experiments, GPT-6 Astra was used for real-to-sim reconstruction of the digital-twin scenes, all of which were manually reviewed (Appendix H.7). Apart from these uses, generative AI was not used to generate experimental data or results or to formulate research hypotheses. We also used large language models to edit and condense the manuscript, draft parts of the appendix text, search and summarize related literature, format references, revise the LaTeX layout, and build the anonymous project page. We have reviewed all AI-assisted work: reported numbers were checked against the experiment logs, references were checked against their sources, and all text was revised by the authors. The authors take full responsibility for the content of this paper, including text, claims, code, and artifacts produced with the aid of generative AI.

REFERENCES

Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Daniel Ho, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Eric Jang, Rosario Jauregui Ruano, Kyle Jeffrey, Sally Jesmonth, Nikhil J. Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Kuang-Huei Lee, Sergey Levine, Yao Lu, Linda Luu, Carolina Parada, Peter Pastor, Jornell Quiambao, Kanishka Rao, Jarek Rettinghouse, Diego Reyes, Pierre Sermanet, Nicolas Sievers, Clayton Tan, Alexander Toshev, Vincent Vanhoucke, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Mengyuan Yan, and Andy Zeng. Do as i can, not as i say: Grounding language in robotic affordances. In *Conference on Robot Learning*, 2022. URL https://openreview.net/forum?id=bdHkMjBJG_w.

Chen Bao, Helin Xu, Yuzhe Qin, and Xiaolong Wang. Dexart: Benchmarking generalizable dexterous manipulation with articulated objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 21190–21200, 2023.

Qingwei Ben, Feiyu Jia, Jia Zeng, Junting Dong, Dahua Lin, and Jiangmiao Pang. Homie: Humanoid loco-manipulation with isomorphic exoskeleton cockpit. *arXiv preprint arXiv:2502.13013*, 2025.

Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, et al. Gr00t n1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025.

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

Baijun Chen, Weijie Wan, Tianxing Chen, Xianda Guo, Congsheng Xu, Yuanyang Qi, Haojie Zhang, Longyan Wu, Tianling Xu, Zixuan Li, et al. Univtac: A unified simulation platform for visuo-tactile manipulation data generation, learning, and benchmarking. *arXiv preprint arXiv:2602.10093*, 2026a.

Jiayi Chen, Yubin Ke, and He Wang. Bodex: Scalable and efficient robotic dexterous grasp synthesis using bilevel optimization. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 01–08. IEEE, 2025a.

Tianxing Chen, Zanxin Chen, Baijun Chen, Zijian Cai, Yibin Liu, Zixuan Li, Qiwei Liang, Xianliang Lin, Yiheng Ge, Zhenyu Gu, et al. Robotwin 2.0: A scalable data generator and benchmark with strong domain randomization for robust bimanual robotic manipulation. *arXiv preprint arXiv:2506.18088*, 2025b.

Tianxing Chen, Yuran Wang, Mingleyang Li, Yanjiao Qin, Haochen Shi, Zixuan Li, Yifan Hu, Yingsheng J Zhang, Kaixuan Wang, Yue Chen, Hongchen Wang, Renjing Xu, Ruihai Wu, Yao Mu, Yaodong Yang, Hao Dong, and Ping Luo. Rmbench: Memory-dependent robotic manipulation benchmark with insights into policy design. *ArXiv*, abs/2603.01229, 2026b. URL https://api.semanticscholar.org/CorpusID:286222901.

Yue Chen, Muqing Jiang, Kaifeng Zheng, Jiaqi Liang, Chenrui Tie, Haoran Lu, Ruihai Wu, and Hao Dong. Learning part-aware dense 3d feature field for generalizable articulated object manipulation. *arXiv preprint arXiv:2602.14193*, 2026c.

Matt Deitke, Dustin Schwenk, Jordi Salvador, Luca Weihs, Oscar Michel, Eli VanderBilt, Ludwig Schmidt, Kiana Ehsani, Aniruddha Kembhavi, and Ali Farhadi. Objaverse: A universe of annotated 3d objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13142–13153, 2023. URL https://openaccess.thecvf.com/content/CVPR2023/html/Deitke_Objaverse_A_Universe_of_Annotated_3D_Objects_CVPR_2023_paper.html.

Shengheng Deng, Xun Xu, Chaozheng Wu, Ke Chen, and Kui Jia. 3d affordancenet: A benchmark for visual object affordance understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1778–1787, 2021. URL https://openaccess.thecvf.com/content/CVPR2021/html/Deng_3D_AffordanceNet_A_Benchmark_for_Visual_Object_Affordance_Understanding_CVPR_2021_paper.html.

Shengliang Deng, Mi Yan, Songlin Wei, Haixin Ma, Yuxin Yang, Jiayi Chen, Zhiqi Zhang, Taoyu Yang, Xuheng Zhang, Wenhao Zhang, et al. Graspvla: a grasping foundation model pre-trained on billion-scale synthetic action data. *arXiv preprint arXiv:2505.03233*, 2025.

Abhay Deshpande, Maya Guru, Rose Hendrix, Snehal Jauhri, Ainaz Eftekhar, Rohun Tripathi, Max Argus, Jordi Salvador, Haoquan Fang, Matthew Wallingford, et al. Molmob0t: Large-scale simulation enables zero-shot manipulation. *arXiv preprint arXiv:2603.16861*, 2026.

Jiafei Duan, Wentao Yuan, Wilbert Pumacay, Yi Ru Wang, Kiana Ehsani, Dieter Fox, and Ranjay Krishna. Manipulate-anything: Automating real-world robots using vision-language models. *arXiv preprint arXiv:2406.18915*, 2024.

Benjamin Ellenberger. Pybullet gymperium. https://github.com/benelot/pybullet-gym, 2018–2019.

Clemens Eppner, Arsalan Mousavian, and Dieter Fox. Acronym: A large-scale grasp dataset based on simulation. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021. URL https://research.nvidia.com/publication/2021-05_acronym-large-scale-grasp-dataset-based-simulation.

Ning Gao, Yilun Chen, Shuai Yang, Xinyi Chen, Yang Tian, Hao Li, Haifeng Huang, Hanqing Wang, Tai Wang, and Jiangmiao Pang. Genmanip: Llm-driven simulation for generalizable instruction-following manipulation. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 12187–12198, 2025.

Haoran Geng, Helin Xu, Chengyang Zhao, Chao Xu, Li Yi, Siyuan Huang, and He Wang. Gapartnet: Cross-category domain-generalizable object perception and manipulation via generalizable and actionable parts. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7081–7091, 2023. URL https://openaccess.thecvf.com/content/CVPR2023/html/Geng_GAPartNet_Cross-Category_Domain-Generalizable_Object_Perception_and_Manipulation_via_Generalizable_and_CVPR_2023_paper.html.

Haoran Geng, Feishi Wang, Songlin Wei, Yuyang Li, Bangjun Wang, Boshi An, Charlie Tianyue Cheng, Haozhe Lou, Peihao Li, Yen-Jen Wang, et al. Roboverse: Towards a unified platform, dataset and benchmark for scalable and generalizable robot learning. *arXiv preprint arXiv:2504.18904*, 2025.

Jiayuan Gu, Fanbo Xiang, Xuanlin Li, Zhan Ling, Xiqiang Liu, Tongzhou Mu, Yihe Tang, Stone Tao, Xinyue Wei, Yunchao Yao, Xiaodi Yuan, Pengwei Xie, Zhiao Huang, Rui Chen, and Hao Su. Maniskill2: A unified benchmark for generalizable manipulation skills. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=b_CQDy9vrD1.

Chengkai Hou, Zhengrong Xue, Bingyang Zhou, Jinghan Ke, Lin Shao, and Huazhe Xu. Keygrid: Unsupervised 3d keypoints detection using grid heatmap features, 2024. URL https://arxiv.org/abs/2410.02237.

Wenlong Huang, Chen Wang, Yunzhu Li, Ruohan Zhang, and Li Fei-Fei. Rekep: Spatio-temporal reasoning of relational keypoint constraints for robotic manipulation. *arXiv preprint arXiv:2409.01652*, 2024.

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

Physical Intelligence, Bo Ai, Ali Amin, Raichelle Aniceto, Ashwin Balakrishna, Greg Balke, Kevin Black, George Bokinsky, Shihao Cao, Thomas Charbonnier, Vedant Choudhary, Foster Collins, Ken Conley, Grace Connors, James Darpinian, Karan Dhabalia, Maitrayee Dhaka, Jared DiCarlo, Danny Driess, Michael Equi, Adnan Esmail, Yunhao Fang, Chelsea Finn, Catherine Glossop, Thomas Godden, Ivan Goryachev, Lachlan Groom, Haroun Habeeb, Hunter Hancock, Karol Hausman, Gashon Hussein, Victor Hwang, Brian Ichter, Connor Jacobsen, Szymon Jakubczak, Rowan Jen, Tim Jones, Gregg Kammerer, Ben Katz, Liyiming Ke, Mairbek Khadikov, Chandra Kuchi, Marinda Lamb, Devin LeBlanc, Brendon LeCount, Sergey Levine, Xinyu Li, Adrian Li-Bell, Vladislav Lialin, Zhonglin Liang, Wallace Lim, Yao Lu, Enyu Luo, Vishnu Mano, Nandan Marwaha, Aikys Mongush, Liam Murphy, Suraj Nair, Tyler Patterson, Karl Pertsch, Allen Z. Ren, Gavin Schelske, Charvi Sharma, Baifeng Shi, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, Will Stoeckle, Jiaming Tang, Jimmy Tanner, Shalom Tekeste, Marcel Torne, Kyle Vedder, Quan Vuong, Anna Walling, Haohuan Wang, Jason Wang, XuDong Wang, Chris Whalen, Samuel Whitmore, Blake Williams, Charles Xu, Sukwon Yoo, Lili Yu, Wuming Zhang, Zhuoyang Zhang, and Ury Zhilinsky. $\pi_{0.7}$: a steerable generalist robotic foundation model with emergent capabilities, 2026. URL https://arxiv.org/abs/2604.15483.

Stephen James, Zicong Ma, David Rovick Arrojo, and Andrew J. Davison. Rlbench: The robot learning benchmark & learning environment. *arXiv preprint arXiv:1909.12271*, 2019. URL https://arxiv.org/abs/1909.12271.

Heewoo Jun and Alex Nichol. Shap-e: Generating conditional 3d implicit functions. *arXiv preprint arXiv:2305.02463*, 2023. URL https://arxiv.org/abs/2305.02463.

Long Le, Jason Xie, William Liang, Hung-Ju Wang, Yue Yang, Yecheng Jason Ma, Kyle Vedder, Arjun Krishna, Dinesh Jayaraman, and Eric Eaton. Articulate-anything: Automatic modeling of articulated objects via a vision-language foundation model. In *International Conference on Learning Representations*, 2025.

Chengshu Li, Ruohan Zhang, Josiah Wong, Cem Gokmen, Sanjana Srivastava, Roberto Martín-Martín, Chen Wang, Gabrael Levine, Michael Lingelbach, Jiankai Sun, et al. Behavior-1k: A benchmark for embodied ai with 1,000 everyday activities and realistic simulation. In *Conference on Robot Learning*, pp. 80–93. PMLR, 2023a.

Chengshu Li, Mengdi Xu, Arpit Bahety, Hang Yin, Yunfan Jiang, Huang Huang, Josiah Wong, Sujay Garlanka, Cem Gokmen, Ruohan Zhang, et al. Momagen: Generating demonstrations under soft and hard constraints for multi-step bimanual mobile manipulation. In *International Conference on Learning Representations*, volume 2026, pp. 112425–112446, 2026a.

Maximilian Xiling Li, Paul Mattes, Nils Blank, Korbinian Franz Rudolf, Paul Werner Lödige, and Rudolf Lioutikov. Multi-objective photoreal simulation (MOPS) dataset for computer vision in robotic manipulation. In *RSS 2025 Workshop on Structured World Models for Robotic Manipulation*, 2025. URL https://openreview.net/forum?id=OHqgPaznoG.

Mingleyang Li, Yuran Wang, Yue Chen, Tianxing Chen, Jiaqi Liang, Zishun Shen, Haoran Lu, Ruihai Wu, and Hao Dong. Garmentpile++: Affordance-driven cluttered garments retrieval with vision-language reasoning. *arXiv preprint arXiv:2603.04158*, 2026b.

Xiaoqi Li, Yanzi Wang, Yan Shen, Ponomarenko Iaroslav, Haoran Lu, Qianxu Wang, Boshi An, Jiaming Liu, and Hao Dong. Imagemanip: Image-based robotic manipulation with affordance-guided next view selection. *arXiv preprint arXiv:2310.09069*, 2023b.

Yicong Li, Na Zhao, Junbin Xiao, Chun Feng, Xiang Wang, and Tat-seng Chua. Laso: Language-guided affordance segmentation on 3d object. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 14251–14260. IEEE, 2024a.

Yitong Li, Ruihai Wu, Haoran Lu, Chuanruo Ning, Yan Shen, Guanqi Zhan, and Hao Dong. Broadcasting support relations recursively from local dynamics for object retrieval in clutters. *ArXiv*, abs/2406.02283, 2024b. URL https://api.semanticscholar.org/CorpusID:270226492.

Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. Code as policies: Language model programs for embodied control. *arXiv preprint arXiv:2209.07753*, 2022. URL https://arxiv.org/abs/2209.07753.

Mu Lin, Yi-Lin Wei, Jiaxuan Chen, Yuhao Lin, Shuoyu Chen, Jiangran Lyu, Jiayi Chen, Yansong Tang, He Wang, and Wei-Shi Zheng. Bidexgrasp: Coordinated bimanual dexterous grasps across object geometries and sizes, 2026. URL https://arxiv.org/abs/2604.06589.

Fangchen Liu, Kuan Fang, Pieter Abbeel, and Sergey Levine. Moka: Open-world robotic manipulation through mark-based visual prompting. *arXiv preprint arXiv:2403.03174*, 2024.

Weiyu Liu, Jiayuan Mao, Joy Hsu, Tucker Hermans, Animesh Garg, and Jiajun Wu. Composable part-based manipulation. In *Proceedings of The 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pp. 1300–1315. PMLR, 2023. URL https://proceedings.mlr.press/v229/liu23e.html.

Haoran Lu, Yitong Li, Ruihai Wu, Chuanruo Ning, Yan Shen, and Hao Dong. Unigarment: A unified simulation and benchmark for garment manipulation. In *ICRA Workshop on Deformable Object Manipulation*, volume 2, 2024a.

Haoran Lu, Ruihai Wu, Yitong Li, Sijie Li, Ziyu Zhu, Chuanruo Ning, Yan Shen, Longzan Luo, Yuanpei Chen, and Hao Dong. Garmentlab: A unified simulation and benchmark for garment manipulation. *ArXiv*, abs/2411.01200, 2024b. URL https://api.semanticscholar.org/CorpusID:273811186.

Zhengyi Luo, Ye Yuan, Tingwu Wang, Chenran Li, Sirui Chen, Fernando Castañeda, Zi-Ang Cao, Jiefeng Li, David Minor, Qingwei Ben, Xingye Da, Runyu Ding, Cyrus Hogg, Lina Song, Edy Lim, Eugene Jeong, Tairan He, Haoru Xue, Wenli Xiao, Zi Wang, Simon Yuen, Jan Kautz, Yan Chang, Umar Iqbal, Linxi Fan, and Yuke Zhu. Sonic: Supersizing motion tracking for natural humanoid whole-body control. *arXiv preprint arXiv:2511.07820*, 2025.

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

Jeffrey Mahler, Jacky Liang, Sherdil Niyaz, Michael Laskey, Richard Doan, Xinyu Liu, Juan Aparicio Ojea, and Ken Goldberg. Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics. In *Robotics: Science and Systems*, 2017. URL https://www.roboticsproceedings.org/rss13/p58.pdf.

Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, and Gavriel State. Isaac gym: High performance gpu-based physics simulation for robot learning. In *Thirty-Fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2021. URL https://arxiv.org/abs/2108.10470.

Ajay Mandlekar, Soroush Nasiriany, Bowen Wen, Iretiayo Akinola, Yashraj Narang, Linxi Fan, Yuke Zhu, and Dieter Fox. Mimicgen: A data generation system for scalable robot learning using human demonstrations. In *Proceedings of The 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pp. 1820–1864. PMLR, 2023. URL https://proceedings.mlr.press/v229/mandlekar23a.html.

Mayank Mittal, Pascal Roth, James Tigue, Antoine Richard, Octi Zhang, Peter Du, Antonio Serrano-Munoz, Xinjie Yao, René Zurbrügg, Nikita Rudin, et al. Isaac lab: A gpu-accelerated simulation framework for multi-modal robot learning. *arXiv preprint arXiv:2511.04831*, 2025.

Kaichun Mo, Shilin Zhu, Angel X. Chang, Li Yi, Subarna Tripathi, Leonidas J. Guibas, and Hao Su. Partnet: A large-scale benchmark for fine-grained and hierarchical part-level 3d object understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 909–918, 2019. URL https://openaccess.thecvf.com/content_CVPR_2019/html/Mo_PartNet_A_Large-Scale_Benchmark_for_Fine-Grained_and_Hierarchical_Part-Level_3D_CVPR_2019_paper.html.

Kaichun Mo, Leonidas Guibas, Mustafa Mukadam, Abhinav Gupta, and Shubham Tulsiani. Where2act: From pixels to actions for articulated 3d objects. *arXiv preprint arXiv:2101.02692*, 2021a. URL https://arxiv.org/abs/2101.02692.

Kaichun Mo, Yuzhe Qin, Fanbo Xiang, Hao Su, and Leonidas Guibas. O2o-afford: Annotation-free large-scale object-object affordance learning. In *Conference on Robot Learning*, 2021b. URL https://openreview.net/forum?id=EougVeukEH9.

Arsalan Mousavian, Clemens Eppner, and Dieter Fox. 6-dof graspnet: Variational grasp generation for object manipulation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 2901–2910, 2019. URL https://openaccess.thecvf.com/content_ICCV_2019/html/Mousavian_6-DOF_GraspNet_Variational_Grasp_Generation_for_Object_Manipulation_ICCV_2019_paper.html.

Tongzhou Mu, Zhan Ling, Fanbo Xiang, Derek Yang, Xuanlin Li, Stone Tao, Zhiao Huang, Zhiwei Jia, and Hao Su. Maniskill: Generalizable manipulation skill benchmark with large-scale demonstrations. *arXiv preprint arXiv:2107.14483*, 2021. URL https://arxiv.org/abs/2107.14483.

Yao Mu, Tianxing Chen, Zanxin Chen, Shijia Peng, Zhiqian Lan, Zeyu Gao, Zhixuan Liang, Qiaojun Yu, Yude Zou, Mingkun Xu, Lunkai Lin, Zhiqiang Xie, Mingyu Ding, and Ping Luo. Robotwin: Dual-arm robot benchmark with generative digital twins. In *Proceedings of the Computer Vision and Pattern Recognition Conference (CVPR)*, pp. 27649–27660, June 2025.

Chuanruo Ning, Ruihai Wu, Haoran Lu, Kaichun Mo, and Hao Dong. Where2explore: Few-shot affordance learning for unseen novel categories of articulated objects. *ArXiv*, abs/2309.07473, 2023. URL https://api.semanticscholar.org/CorpusID:261823492.

Yida Niu, Xinhai Chang, Xin Liu, Ziyuan Jiao, and Yixin Zhu. Scalable trajectory generation for whole-body mobile manipulation. In *CVPR*, 2026.

NVIDIA. *Isaac Sim Documentation*, 2025. URL https://docs.isaacsim.omniverse.nvidia.com/4.5.0/index.html. Official documentation.

OpenAI, Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Józefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, Jonas Schneider, Szymon Sidor, Josh Tobin, Peter Welinder, Lilian Weng, and Wojciech Zaremba. Learning dexterous in-hand manipulation. *arXiv preprint arXiv:1808.00177*, 2018. URL https://arxiv.org/abs/1808.00177.

Zhenyu Pan, Yiting Zhang, Zhuo Liu, Yolo Yunlong Tang, Zeliang Zhang, Haozheng Luo, Yuwei Han, Jianshu Zhang, Dennis Wu, Hong-Yu Chen, Haoran Lu, Haoyang Fang, Manling Li, Chenliang Xu, Philip S. Yu, and Han Liu. AdvEvo-MARL: Shaping internalized safety through adversarial co-evolution in multi-agent reinforcement learning. *arXiv preprint arXiv:2510.01586*, 2025. doi: 10.48550/arXiv.2510.01586. URL https://arxiv.org/abs/2510.01586. Accepted to ICML 2026.

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

Delin Qu, Haoming Song, Qizhi Chen, Yuanqi Yao, Xinyi Ye, Yan Ding, Zhigang Wang, JiaYuan Gu, Bin Zhao, Dong Wang, et al. Spatialvla: Exploring spatial representations for visual-language-action model. *arXiv preprint arXiv:2501.15830*, 2025.

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026. URL https://qwen.ai/blog?id=qwen3.5.

SAPIEN. Partnet-mobility dataset. https://sapien.ucsd.edu/, 2020. Official dataset entry point within the SAPIEN ecosystem.

Yanming Shao and Chenxi Xiao. Bimanual grasp synthesis for dexterous robot hands. *arXiv preprint arXiv:2411.15903*, 2024. URL https://arxiv.org/abs/2411.15903.

Yan Shen, Ruihai Wu, Yubin Ke, Xinyuan Song, Zeyi Li, Xiaoqi Li, Hongwei Fan, Haoran Lu, et al. Biassemble: Learning collaborative affordance for bimanual geometric assembly. *arXiv preprint arXiv:2506.06221*, 2025.

Ruoxi Shi, Zhengrong Xue, Yang You, and Cewu Lu. Skeleton merger: an unsupervised aligned keypoint detector. *arXiv preprint arXiv:2103.10814*, 2021.

Sanjana Srivastava, Chengshu Li, Michael Lingelbach, Roberto Martín-Martín, Fei Xia, Kent Vainio, Zheng Lian, Cem Gokmen, Shyamal Buch, C. Karen Liu, Silvio Savarese, Hyowon Gweon, Jiajun Wu, and Li Fei-Fei. Behavior: Benchmark for everyday household activities in virtual, interactive, and ecological environments. *arXiv preprint arXiv:2108.03332*, 2021. URL https://arxiv.org/abs/2108.03332.

Balakumar Sundaralingam, Siva Kumar Sastry Hari, Adam Fishman, Caelan Garrett, Karl Van Wyk, Valts Blukis, Alexander Millane, Helen Oleynikova, Ankur Handa, Fabio Ramos, et al. curobo: Parallelized collision-free minimum-jerk robot motion generation. *arXiv preprint arXiv:2310.17274*, 2023.

Balakumar Sundaralingam, Adithyavairavan Murali, and Stan Birchfield. curobov2: Dynamics-aware motion generation with depth-fused distance fields for high-dof robots. *arXiv preprint arXiv:2603.05493*, 2026.

Martin Sundermeyer, Arsalan Mousavian, Rudolph Triebel, and Dieter Fox. Contact-graspnet: Efficient 6-dof grasp generation in cluttered scenes. *arXiv preprint arXiv:2103.14127*, 2021. URL https://arxiv.org/abs/2103.14127.

Richard S. Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1-2):181–211, 1999. doi: 10.1016/S0004-3702(99)00052-1.

Bingjie Tang, Michael A. Lin, Iretiayo Akinola, Ankur Handa, Gaurav S. Sukhatme, Fabio Ramos, Dieter Fox, and Yashraj Narang. Industreal: Transferring contact-rich assembly tasks from simulation to reality. In *Robotics: Science and Systems*, 2023. URL https://arxiv.org/abs/2305.17110.

Peijun Tang, Shangjin Xie, Binyan Sun, Baifu Huang, Kuncheng Luo, Haotian Yang, Weiqi Jin, and Jianan Wang. Mind to hand: Purposeful robotic control via embodied reasoning. *arXiv preprint arXiv:2512.08580*, 2025.

Yang Tian, Yuyin Yang, Yiman Xie, Zetao Cai, Xu Shi, Ning Gao, Hangxu Liu, Xuekun Jiang, Zherui Qiu, Feng Yuan, et al. Interndata-a1: Pioneering high-fidelity synthetic data for pre-training generalist policy. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 976–985, 2026.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033. IEEE, 2012. doi: 10.1109/IROS.2012.6386109.

Weikang Wan, Haoran Geng, Yun Liu, Zikang Shan, Yaodong Yang, Li Yi, and He Wang. Unidexgrasp++: Improving dexterous grasping policy learning via geometry-aware curriculum and iterative generalist-specialist learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 3891–3902, 2023.

Kaixuan Wang, Tianxing Chen, Jiawei Liu, Honghao Su, Shaolong Zhu, Minxuan Wang, Zixuan Li, Yue Chen, Huan-ang Gao, Yusen Qin, et al. Manitwin: Scaling data-generation-ready digital object dataset to 100k. *arXiv preprint arXiv:2603.16866*, 2026a.

Lirui Wang, Yiyang Ling, Zhecheng Yuan, Mohit Shridhar, Chen Bao, Yuzhe Qin, Bailin Wang, Huazhe Xu, and Xiaolong Wang. Gensim: Generating robotic simulation tasks via large language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=OI3RoHoWAN.

Penghao Wang, Yiyang He, Xin Lv, Yukai Zhou, Lan Xu, Jingyi Yu, and Jiayuan Gu. Partnext: A next-generation dataset for fine-grained and hierarchical 3d part understanding. *Advances in Neural Information Processing Systems*, 38, 2026b.

Ruicheng Wang, Jialiang Zhang, Jiayi Chen, Yinzhen Xu, Puhao Li, Tengyu Liu, and He Wang. Dexgraspnet: A large-scale robotic dexterous grasp dataset for general objects based on simulation. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 11359–11366. IEEE, 2023a. URL https://arxiv.org/abs/2210.02697.

Wenbo Wang, Fangyun Wei, Qixiu Li, Xi Chen, Yaobo Liang, Chang Xu, Jiaolong Yang, and Baining Guo. Mobilemanibench: Simplifying model verification for mobile manipulation. In *European Conference on Computer Vision*, pp. 242–259. Springer, 2026c.

Yufei Wang, Zhou Xian, Feng Chen, Tsun-Hsuan Wang, Yian Wang, Katerina Fragkiadaki, Zackory Erickson, David Held, and Chuang Gan. Robogen: Towards unleashing infinite data for automated robot learning via generative simulation. *arXiv preprint arXiv:2311.01455*, 2023b.

Yufei Wang, Ziyu Wang, Mino Nakura, Pratik Bhowal, Chia-Liang Kuo, Yi-Ting Chen, Zackory Erickson, and David Held. Articubot: Learning universal articulated object manipulation policy via large scale simulation. *ArXiv*, abs/2503.03045, 2025. URL https://api.semanticscholar.org/CorpusID:276781877.

Ruihai Wu, Yan Zhao, Kaichun Mo, Zizheng Guo, Yian Wang, Tianhao Wu, Qingnan Fan, Xuelin Chen, Leonidas Guibas, and Hao Dong. Vat-mart: Learning visual action trajectory proposals for manipulating 3d articulated objects. In *The Tenth International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=iEx3PiooLy.

Ruihai Wu, Haoran Lu, Yiyan Wang, Yubo Wang, and Hao Dong. Unigarmentmanip: A unified framework for category-level garment manipulation via dense visual correspondence. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 16340–16350. IEEE, 2024.

Ruihai Wu, Haozhe Chen, Mingtong Zhang, Haoran Lu, Yitong Li, and Yunzhu Li. Neural dynamics augmented diffusion policy. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 13234–13241. IEEE, 2025.

Fanbo Xiang, Yuzhe Qin, Kaichun Mo, Yikuan Xia, Hao Zhu, Fangchen Liu, Minghua Liu, Hanxiao Jiang, Yifu Yuan, He Wang, Li Yi, Angel X. Chang, Leonidas J. Guibas, and Hao Su. Sapien: A simulated part-based interactive environment. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11097–11107, 2020. URL https://openaccess.thecvf.com/content_CVPR_2020/html/Xiang_SAPIEN_A_SimulAted_Part-Based_Interactive_ENvironment_CVPR_2020_paper.html.

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

Sizhe Yang, Yiman Xie, Zhixuan Liang, Yang Tian, Jia Zeng, Dahua Lin, and Jiangmiao Pang. Ultradexgrasp: Learning universal dexterous grasping for bimanual robots with synthetic data, 2026. URL https://arxiv.org/abs/2603.05312.

Yunhan Yang, Yukun Huang, Yuan-Chen Guo, Liangjun Lu, Xiaoyang Wu, Edmund Y Lam, Yan-Pei Cao, and Xihui Liu. Sampart3d: Segment any part in 3d objects. *arXiv preprint arXiv:2411.07184*, 2024.

Guo Ye, Zexi Zhang, Xu Zhao, Shang Wu, Haoran Lu, Shihan Lu, and Han Liu. Learning to feel the future: Dreamtacvla for contact-rich manipulation. *arXiv preprint arXiv:2512.23864*, 2025.

Seonghyeon Ye, Yunhao Ge, Kaiyuan Zheng, Shenyuan Gao, Sihyun Yu, George Kurian, Suneel Indupuru, You Liang Tan, Chuning Zhu, Jiannan Xiang, et al. World action models are zero-shot policies. *arXiv preprint arXiv:2602.15922*, 2026.

Chenghao Yin, Da Huang, Di Yang, Jichao Wang, Nanshu Zhao, Chen Xu, Wenjun Sun, Linjie Hou, Zhijun Li, Junhui Wu, et al. Genie sim 3.0: A high-fidelity comprehensive simulation platform for humanoid robot. *arXiv preprint arXiv:2601.02078*, 2026.

Yang You, Yujing Lou, Chengkun Li, Zhoujun Cheng, Liangwei Li, Lizhuang Ma, Cewu Lu, and Weiming Wang. Keypointnet: A large-scale 3d keypoint dataset aggregated from numerous human annotations. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 13644–13653. IEEE, 2020.

Yifan You, Lin Shao, Toki Migimatsu, and Jeannette Bohg. Omnihang: Learning to hang arbitrary objects using contact point correspondences and neural collision estimation. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021. URL https://arxiv.org/abs/2103.14283.

Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Avnish Narayan, Hayden Shively, Adithya Bellathur, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. *arXiv preprint arXiv:1910.10897*, 2019. URL https://arxiv.org/abs/1910.10897.

Wentao Yuan, Jiafei Duan, Valts Blukis, Wilbert Pumacay, Ranjay Krishna, Adithyavairavan Murali, Arsalan Mousavian, and Dieter Fox. Robopoint: A vision-language model for spatial affordance prediction for robotics. *arXiv preprint arXiv:2406.10721*, 2024.

Jianshu Zhang, Yijiang Li, Huifeixin Chen, Haoran Lu, Letian Xue, Bingyang Wang, and Han Liu. SPACENUM: Revisiting spatial numerical understanding in VLMs. *arXiv preprint arXiv:2605.23898*, 2026a. doi: 10.48550/arXiv.2605.23898. URL https://arxiv.org/abs/2605.23898.

Jianshu Zhang, Chengxuan Qian, Haosen Sun, Haoran Lu, Dingcheng Wang, Letian Xue, and Han Liu. ProgressLM: Towards progress reasoning in vision-language models. *arXiv preprint arXiv:2601.15224*, 2026b. doi: 10.48550/arXiv.2601.15224. URL https://arxiv.org/abs/2601.15224. ACL 2026 Main Conference Oral; ICLR 2026 Workshop on World Models.

Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. In *Robotics: Science and Systems (RSS)*, 2023.

Ruijie Zheng, Dantong Niu, Yuqi Xie, Jing Wang, Mengda Xu, Yunfan Jiang, Fernando Castañeda, Fengyuan Hu, You Liang Tan, Letian Fu, et al. Egoscale: Scaling dexterous manipulation with diverse egocentric human data. *arXiv preprint arXiv:2602.16710*, 2026.

Bingyang Zhou, Haoyu Zhou, Tianhai Liang, Qiaojun Yu, Siheng Zhao, Yuwei Zeng, Jun Lv, Siyuan Luo, Qiancai Wang, Xinyuan Yu, Haonan Chen, Cewu Lu, and Lin Shao. Clothesnet: An information-rich 3d garment model repository with simulated clothes environment, 2023. URL https://arxiv.org/abs/2308.09987.

Yuke Zhu, Josiah Wong, Ajay Mandlekar, Roberto Martín-Martín, Abhishek Joshi, Kevin Lin, Abhiram Maddukuri, Soroush Nasiriany, and Yifeng Zhu. robosuite: A modular simulation framework and benchmark for robot learning. *arXiv preprint arXiv:2009.12293*, 2020. URL https://arxiv.org/abs/2009.12293.

A Appendix Overview

This appendix collects the detailed prompts, schemas, implementation details, supporting experiments, governance notes, and qualitative examples that complement the main paper. **Videos, code, demo assets, and annotations are provided in supplementary materials.**

- **Appendix B (Qualitative Example).** Provides an end-to-end qualitative walkthrough of the annotation pipeline on representative assets and scenes.

- **Appendix C (Language Annotation Pipeline).** Details the asset-level and room-level language prompts, output schemas, and example annotations used in the visual-language stage.

- **Appendix D (Visual Annotation Pipeline).** Describes the visual grounding procedures, including part segmentation, keypoint candidates and selection, 3D refinement, occupancy maps, and room-level geometric annotations.

- **Appendix E (Details of Physics Annotation Pipeline).** Expands the unified action schema and the general generation, optimization, validation, and augmentation pipeline for executable action annotations.

- **Appendix F (Primitive-specific Action Annotation).** Summarizes skill-specific instantiations for grasping, articulation, insertion, hanging, garment manipulation, and navigation-related actions.

- **Appendix G (Additional Details of Large-scale Robot Data Collection).** Covers downstream data generation, keypoint and affordance supervision, the downstream-utility evaluation with established pipelines, and the illustrative robot reasoning VQA and 3D VLM instruction-tuning data derived from the annotations.

- **Appendix H (Additional Experimental Details).** Reports audited-suite construction, coverage statistics, evaluation protocols and per-skill results, the real-world protocol with matched data-source comparisons, baseline adaptation protocols, human-evaluation and uncertainty reporting, and timing metrics.

- **Appendix I (Related Work).** Provides a broader discussion of adjacent literature on simulation data generation, affordance annotation, assets, and skill-centric learning.

- **Appendix J (Limitations, Broader Impact, and Asset Licensing).** Documents compute resources, limitations, broader impacts, external asset licensing, and release considerations for new derived assets.



Figure 9: Qualitative gallery of asset-level annotations across diverse object categories. Each row shows multiple instances within a category (mugs, hammers, teapots, kettles, lipsticks, bottles, knives), illustrating the geometric and appearance diversity covered by our automatic annotation pipeline at the object-asset level.

B QUALITATIVE EXAMPLE

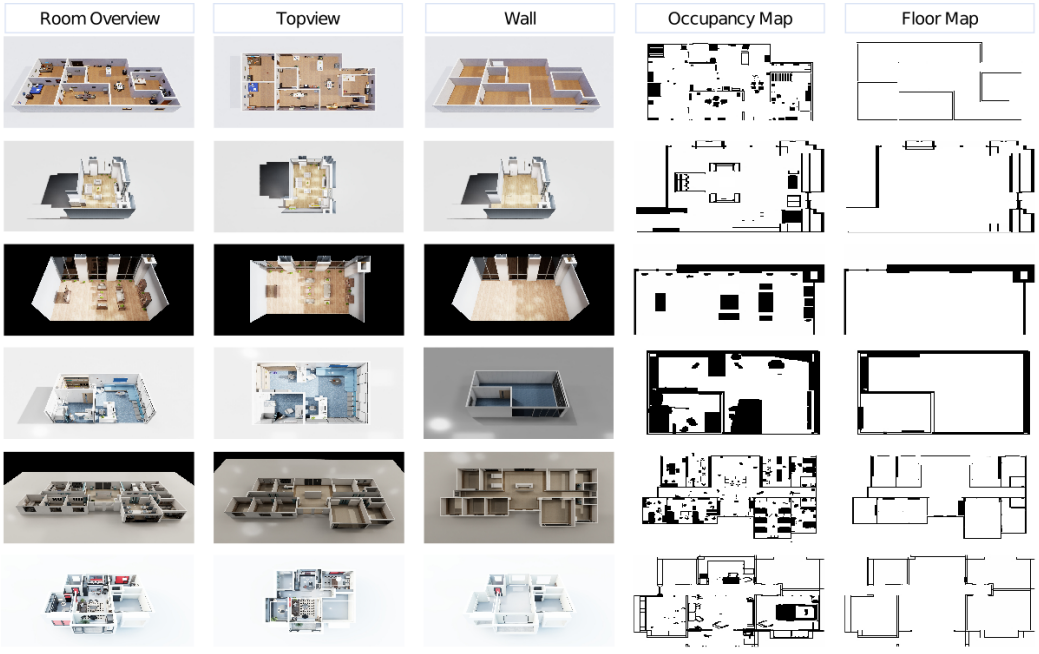


Figure 10: Qualitative examples of room-level visual annotations. From left to right: full 3D room overview, top-view rendering, wall-only structural map, multi-height occupancy map, and cleaned floor plan. Each row corresponds to a different scene, demonstrating that the pipeline produces consistent structural annotations across rooms with varying layouts and clutter.



Figure 11: Qualitative examples of action annotations across manipulation skills. Each row shows several instances of one skill family, including bimanual dexterous grasp, bimanual gripper grasp, dexterous grasp, gripper open-door, fling, fold, grasp, hang, insert, and navigation (row labels use internal names; e.g., "BiDexhand grasp" and "Bigripper grasp" correspond to Dex-Bimanual Grasp and Bimanual Grasp in Fig. 7). The examples illustrate that the unified action schema produces functionally meaningful candidate trajectories across both rigid and deformable objects.



Figure 12: Additional qualitative examples of action annotations for articulated and contact-rich skills. Each row shows several instances of one skill, including gripper/dexterous opening by handle, gripper/dexterous opening by edge, place, press, and rotate. The samples highlight how the same articulated object admits multiple feasible interaction strategies, parameterized by different anchor types and end-effector embodiments.

C Language Annotation Pipeline

C.1 Asset-level Language Annotation Details

Multi-view and hierarchical annotation. For each object-level asset, we normalize camera placement using the center $\mathbf{c}$ and diagonal length d of its axis-aligned bounding box. Eight RGB views are rendered from evenly spaced azimuths on a horizontal circle, with camera distance proportional to d. Qwen3.5-27B Qwen Team (2026) then produces a phrase-level label, a sentence-level functional description, and a paragraph-level manipulation description. For part-aware descriptions, each segmented part is highlighted in turn and queried with the remaining object as context, linking the output to its segmentation ID.

Table 5: Hierarchical annotation schema of AnnotateAnything.

Level	Annotation Type	Example Outputs
Asset	Language	Sparse tags; dense object-, part- and task-level descriptions
Asset	Visual	3D keypoints; part segmentation; affordance regions
Room	Language	Scene descriptions; object relations; task contexts
Room	Visual	Occupancy maps; floor plans; top-view/BEV maps; object layouts
Action	Rigid object	Parallel-jaw grasps; dexterous contacts; insertion and hanging poses
Action	Articulated object	Handles; motion axes; articulation waypoints; opening and closing trajectories
Action	Deformable object	Garment keypoints; bimanual grasp points; folding and hanging trajectories
Action	Room-scale scene	Navigation targets; approach poses; interaction-ready candidate base poses



Figure 13: Asset-level language annotation. Eight views from normalized camera placements support global descriptions, while highlighted part renderings produce part-aware descriptions grounded in segmentation regions.

Output and filtering. The model returns structured JSON. Prompts restrict descriptions to visible evidence and require uncertain attributes to be marked as `unknown`. We discard unparseable outputs and part descriptions that cannot be matched to a valid segmentation ID. Retained annotations provide semantic priors for visual grounding and physics-based action generation.

Global asset-level prompt.

Global asset-level language prompt

```
System: You are a careful annotation engine for robot manipulation
assets. Your task is to describe the visible 3D object for robot
reasoning and manipulation. Use only visual evidence from the
provided views. Do not invent hidden mechanisms or unsupported
functions. If an attribute is unclear, write "unknown".
User: You are given eight RGB renderings of the same 3D object
from surrounding viewpoints. Please analyze the object as a robot
manipulation asset and output a structured annotation with three
levels of detail.
Level 1: phrase-level label. Write a short phrase, no more than 6
words, that identifies the object category or functional identity.
Level 2: sentence-level description. Write one sentence, no more
than 30 words, describing the object's main function and major
manipulable parts.
```

```
Level 3: paragraph-level description. Write one dense paragraph
for robot manipulation reasoning. Describe the object's visible
parts, their functions, likely affordances, feasible interactions,
and manipulation constraints. Focus on what a robot can grasp, pull,
push, support, open, insert into, hang on, or avoid.
Return the result in JSON format with the following fields:
phrase, sentence, paragraph, major_parts, uncertain_attributes.
```

Part-aware prompt.

Part-aware language prompt

```
System: You are a robot manipulation annotation engine. You will be
given an image of a 3D object where one segmented part is highlighted.
The highlighted region is the target part. The rest of the object is
shown only as context. Describe only the highlighted part.
User: Please analyze the highlighted part for robot manipulation. Do
not describe unrelated parts unless they are needed to explain the
target part. If the highlighted part's function is uncertain, write
"unknown" instead of guessing.
Please answer the following questions: 1. What is the name of the
highlighted part? 2. What visual evidence supports this part name?
3. What is the function of this part? 4. What robot interactions
are feasible with this part? 5. What constraints should a robot
consider when interacting with this part? 6. How is this part
related to the whole object?
Return the result in JSON format with the following fields:
part_name, visual_evidence, part_function, feasible_robot_interactions,
manipulation_constraints, relation_to_whole_object, confidence.
```

Example output.

Global asset-level output: mug

Phrase-level label: Blue handled mug.
Sentence-level description: A blue mug with a hollow cup body and a side handle, designed for containing liquid and being grasped by the handle.
Paragraph-level description: The object is a mug composed of a cylindrical cup body, an open rim, a hollow interior, a side handle, and a bottom base. The cup body provides the main container volume and can be used for placement and containment reasoning. The side handle is the most important manipulation part because it supports stable grasping, lifting, and carrying. The open rim and hollow interior indicate that the mug can contain liquid or small objects, but they should be avoided as forceful grasp targets. The bottom base supports stable placement on flat surfaces.

Part-aware output: mug handle

Highlighted part: Mug handle.
The highlighted part is the mug handle. It is a curved loop attached to the side of the cup body. Its function is to provide a safe and stable grasping region for lifting or carrying the mug. Feasible robot interactions include grasping the outer handle, inserting fingers or gripper tips through the loop, and pulling or lifting the mug through the handle. The robot should avoid colliding with the cup body near the handle attachment points and should align the grasp with the handle opening.

C.2 Room-level Language Annotation Details

Room-view and layout annotation. For each room-scale scene, we sample candidate camera poses within individual rooms and use simulator segmentation as a quality signal. We retain eight views with diverse positions and directions after rejecting views with too few visible non-structural objects or views dominated by walls and floors. The VLM receives these RGB views together with a labeled top-view occupancy map containing free space, obstacles, room boundaries, and object instances. Qwen3.5-27B Qwen Team (2026) generates short-sentence, long-sentence, and paragraph-level

descriptions covering room type, furniture and functional zones, spatial relations, navigation structure, interaction regions, and possible robot tasks.

Output and filtering. The VLM returns structured JSON, using RGB views for local evidence and the occupancy map for global layout. Prompts prohibit unsupported object names or hidden room functions and require uncertain attributes to be marked as `unknown`. We discard unparseable outputs and object references that cannot be matched to labeled occupancy-map instances.

Room-level prompt.

Room-level language prompt

```
System: You are a careful annotation engine for robot navigation and
manipulation scenes. Your task is to describe the visible room or
indoor scene for robot reasoning. Use the provided RGB views for
local visual evidence and the labeled top-view occupancy map for
global layout, spatial relations, and navigable structure. Do not
invent hidden rooms, unsupported objects, or functions that are not
visible or labeled. If an attribute is unclear, write "unknown".
User: You are given multiple RGB renderings sampled inside a
room-scale scene, together with a top-view occupancy map. The
occupancy map marks free space, obstacles, room boundaries, and
object instances with labels. Please analyze the scene as a robot
interaction environment and output a structured room-level annotation
with three levels of detail.
Level 1: short-sentence description. Write one short sentence, no
more than 15 words, summarizing the room type and global layout.
Level 2: long-sentence description. Write one to two long sentences
describing the major furniture, functional zones, spatial arrangement,
and navigation structure.
Level 3: paragraph-level scene description. Write one dense
paragraph for robot navigation and manipulation reasoning.
Describe the room layout, object relations, navigable regions,
interaction-relevant areas, and possible long-horizon robot tasks.
Focus on what a robot can navigate to, avoid, approach, pick, place,
open, retrieve, or interact with.
Return the result in JSON format with the following fields:
short_sentence, long_sentence, paragraph, room_type,
functional_zones, object_inventory, spatial_relations,
navigable_regions, interaction_relevant_areas, possible_robot_tasks,
uncertain_attributes.
```

Example output.

Room-level output: dining room

Short-sentence description: A rectangular dining room with a central table and surrounding chairs.
Long-sentence description: The room contains a central dining area formed by a round table and multiple chairs, with additional support and storage furniture near the rear and side walls. Open floor space around the furniture provides navigation routes, while the table, chairs, and shelf create local obstacles and approach constraints.
Paragraph-level description: The room is a rectangular dining or meeting space with most interaction objects concentrated near the center and right wall. A round table with several chairs forms the main functional zone for placing, picking, and tabletop manipulation. A secondary table near the rear wall provides another support surface, while the side bookshelf can be used for object retrieval or placement. The open floor around the furniture provides navigable space, but the chairs and table legs create local obstacles that require careful path planning. A robot can navigate around the table, approach chairs or tables, retrieve objects from the table or shelf, and place objects on available support surfaces.

D VISUAL ANNOTATION PIPELINE

D.1 ASSET-LEVEL VISUAL ANNOTATION DETAILS

Multi-view point-cloud reconstruction. For each object-level asset, we reconstruct the input point cloud from eight initial RGB-D views rather than sampling the mesh directly, keeping the final geometry derived from rendered observations. Cameras are placed at evenly spaced azimuths around the asset, oriented toward the bounding-box center, with distance scaled by object size.

Given depth D_i, intrinsics K_i, and camera-to-world extrinsics $T_{c_i}^{w} = [R_i \mid t_i]$, each valid pixel (u, v) is back-projected as

$$\mathbf{x}_{uv}^{c_i} = D_i(u,v) K_i^{-1} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}, \qquad \mathbf{x}_{uv}^{w} = R_i \mathbf{x}_{uv}^{c_i} + t_i.$$

The fused cloud is

$$\mathcal{P}_{cam} = \bigcup_{i=1}^{N} \left\{ \mathbf{x}_{uv}^{w} \mid D_i(u,v) > 0,\ M_i(u,v) = 1 \right\},$$

where $N = 8$ by default and M_i is the simulated foreground mask. Masked points from all views are transformed into a common frame and merged.

Coverage verification with mesh-based reference points. Mesh samples are used only to verify camera-view coverage. For reference points $\mathcal{P}_{mesh}$, we compute

$$\mathrm{Cov}(\mathcal{P}_{cam}, \mathcal{P}_{mesh}) = \frac{1}{|\mathcal{P}_{mesh}|} \sum_{\mathbf{p} \in \mathcal{P}_{mesh}} \mathbb{1} \left[\min_{\mathbf{q} \in \mathcal{P}_{cam}} \|\mathbf{p} - \mathbf{q}\|_2 < \epsilon \right],$$

where ϵ is the distance threshold. We also compute voxel-region overlap

$$\mathrm{IoU}_{roi} = \frac{|\mathcal{V}(\mathcal{P}_{cam}) \cap \mathcal{V}(\mathcal{P}_{mesh})|}{|\mathcal{V}(\mathcal{P}_{cam}) \cup \mathcal{V}(\mathcal{P}_{mesh})|}.$$

If either metric is below its threshold, we add views targeting poorly covered regions and repeat reconstruction, improving coverage of thin, concave, or occluded structures without replacing the camera-derived cloud with mesh samples.

Point-cloud downsampling. Before segmentation, we downsample $\mathcal{P}_{cam}$ to $M = 80{,}000$ points by farthest point sampling (FPS). Starting from an initialized set $\mathcal{S}$, FPS iteratively selects

$$\mathbf{p}^{*} = \arg \max_{\mathbf{p} \in \mathcal{P}_{cam}} \min_{\mathbf{s} \in \mathcal{S}} \|\mathbf{p} - \mathbf{s}\|_2,$$

and updates $\mathcal{S} \leftarrow \mathcal{S} \cup \{\mathbf{p}^{*}\}$ until $|\mathcal{S}| = M$. We denote the result by $\mathcal{P}_{80k}$.

Native 3D part segmentation. We segment $\mathcal{P}_{80k}$ using the 182M-parameter Hunyuan3D part-segmentation model, built on native 3D decomposition modules such as P3-SAM Ma et al. (2025) and X-Part Yan et al. (2025). It predicts

$$\mathcal{M} = \{m_k\}_{k=1}^{K}, \qquad m_k \in \{0,1\}^{|\mathcal{P}_{80k}|},$$

with one mask per part. When denser geometry is needed, labels are propagated to $\mathcal{P}_{cam}$ by nearest-neighbor assignment:

$$\ell(\mathbf{p}) = \ell \left(\arg \min_{\mathbf{q} \in \mathcal{P}_{80k}} \|\mathbf{p} - \mathbf{q}\|_2 \right), \qquad \mathbf{p} \in \mathcal{P}_{cam}.$$

Under our setup, segmentation takes approximately 30 seconds per object.

Keypoint candidates and selection. Keypoint candidates are produced by Skeleton Merger Shi et al. (2021) on the FPS-downsampled point cloud, with the number of candidates adjusted automatically to the number of candidate poses required for the asset. The VLM selects task-relevant candidates, such as handles, openings, corners, garment landmarks, or scene-level targets, using the rendered views, and the selected keypoints are retained as semantic anchors.

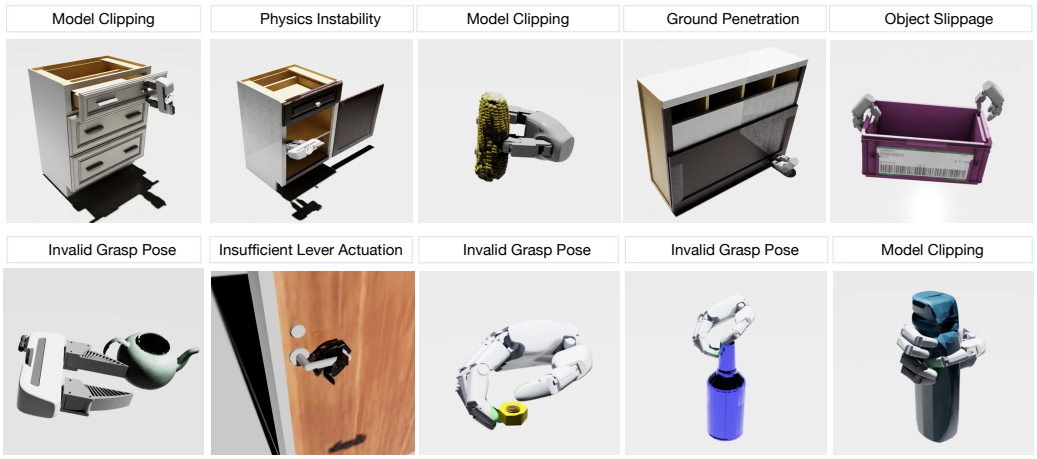


Figure 14: Common failures during physics validation of generated action candidates: model clipping, physics instability, ground penetration, object slippage, invalid grasp poses, and insufficient lever actuation. These cases motivate simulation-based filtering beyond geometric and language priors.

3D refinement. Part and affordance regions obtained from segmentation and VLM selection are refined for consistency with the fused point cloud. First, multi-view voting keeps a point in a region only if the region's masks in the majority of views that observe the point agree on it. Second, outlier removal keeps the largest connected component of each region and discards isolated points. The "w/o 3D refine" ablation in Table 2 skips both steps and uses the unrefined regions.

Segmentation limitation. A key limitation of part segmentation is semantic granularity mismatch. For example, a wine glass may be returned as one part although manipulation benefits from separating its bowl, stem, and base. We therefore treat segmentation as a visual prior; task-specific geometric reasoning and physics validation remain necessary for executable action annotations.

D.2 Room-level Visual Annotation Details



Figure 15: Room-scale scene composition. A populated 3D room (center) is assembled from the diverse object assets shown in the surrounding panels, producing cluttered scenes for scan-pose sampling and occupancy-map construction.

Wall-based room decomposition. We decompose each scene into rooms using wall boundaries and layout structure. Each room is represented by a global-frame floor-plan polygon $\mathcal{R}_j$, processed independently before its maps are stitched into the scene representation.

Rule-based scan-pose sampling. For each $\mathcal{R}_j$, we randomly sample a candidate set Π_j of camera and LiDAR poses inside the room polygon, requiring free-space placement, clearance from walls and large obstacles, and a viewing direction toward the room interior. Simulator instance segmentation is used only to score view quality. For candidate pose π, the visible object set is

$$\mathcal{V}(\pi) = \left\{ o \in \mathcal{O}_j \;\middle|\; \sum_{u,v} \mathbf{1}[S_\pi(u,v) = o] > \tau_{\text{area}} \right\},$$

where $\mathcal{O}_j$ is the room's object set, S_π is the rendered segmentation map, and τ_{area} removes barely visible instances. We greedily maximize newly covered IDs:

$$\pi_t = \arg\max_{\pi \in \Pi_j} \left| \mathcal{V}(\pi) \setminus \bigcup_{k<t} \mathcal{V}(\pi_k) \right|,$$

until

$$\text{Cov}_{obj} = \frac{|\bigcup_t \mathcal{V}(\pi_t)|}{|\mathcal{O}_j|}$$

reaches its threshold or the pose budget is exhausted. This avoids uninformative wall-facing views and improves object-instance coverage.

Multi-height occupancy from LiDAR and ray tracing. For each selected pose π_t and height $h_\ell \in \mathcal{H}$, we cast horizontal rays in uniformly sampled directions. If ρ_α is the first-hit distance along direction $\mathbf{d}_\alpha$, cells before the hit are free and the hit cell is occupied:

$$\mathcal{F}_{t,\ell,\alpha} = \{ g(\mathbf{x}_t + r\mathbf{d}_\alpha, h_\ell) \mid 0 < r < \rho_\alpha - \delta \}, \quad \mathcal{O}_{t,\ell,\alpha} = g(\mathbf{x}_t + \rho_\alpha \mathbf{d}_\alpha, h_\ell),$$

where $\mathbf{x}_t$ is the scan position, $g(\cdot)$ maps 3D points to grid cells, and δ is a safety margin. We aggregate rays with

$$L_\ell(c) \leftarrow L_\ell(c) + \begin{cases} \lambda_{\text{occ}}, & c \in \mathcal{O}_{t,\ell,\alpha}, \\ \lambda_{\text{free}}, & c \in \mathcal{F}_{t,\ell,\alpha}, \\ 0, & \text{otherwise}, \end{cases}$$

and compute $P_\ell(c) = \sigma(L_\ell(c))$. The multi-height annotation is

$$\mathcal{G}_j = \{G_j^{h_1}, G_j^{h_2}, \ldots, G_j^{h_L}\},$$

whose slices capture floor-level free space, low obstacles, furniture geometry, and wall-level structure.

Global map stitching. Room maps are aligned using the local-room-to-world transform T_j^w. For room map G_j,

$$G^w(c) = \text{Fuse}_j \left(G_j \left((T_j^w)^{-1} c \right) \right),$$

where Fuse combines overlapping evidence from adjacent rooms and shared boundaries. The result includes global occupancy, object-layout, and top-view maps.

VLM-assisted wall and floor-plan filtering. Because raw occupancy includes both structure and movable objects, we generate a labeled connected-component map and ask the VLM to retain persistent structural components while removing furniture and clutter. Given $\mathcal{C} = \{C_k\}$ and predicted structural subset $\mathcal{C}_{wall} \subset \mathcal{C}$,

$$W = \bigcup_{C_k \in \mathcal{C}_{wall}} C_k.$$

The resulting annotations comprise multi-height occupancy, object-layout, cleaned floor-plan, and wall-structure maps, which support exploration, navigation-target generation, and scene-level action annotation.

Prompt for VLM-assisted wall filtering.

Wall-structure filtering prompt

```
System: You are a careful annotation engine for indoor scene
structure. Your task is to identify persistent architectural
structures in a top-view map. Keep walls, room boundaries, door
frames, and other fixed structural elements. Remove movable objects
such as tables, chairs, cabinets, shelves, sofas, lamps, and small
objects.
User: You are given a top-view occupancy map with connected
components and object labels. Each connected component has an
ID. Please decide which components correspond to permanent wall or
room-boundary structures.
Rules: 1. Keep components that form long, continuous room
boundaries. 2. Keep components that separate rooms or define door
openings. 3. Remove components that correspond to furniture, movable
objects, or clutter. 4. If a component is uncertain, mark it as
"uncertain" instead of guessing.
Return the result in JSON format with the following
fields: wall_component_ids, removed_object_component_ids,
uncertain_component_ids, reasoning.
```

E Details of the Physics Annotation Pipeline

E.1 Unified Action Annotation Schema

Action annotations are organized as a one-to-many mapping from grounded visual anchors to compatible skills and multiple action candidates:

$$\text{visual anchor} \rightarrow \text{functional affordance} \rightarrow \text{compatible skills} \rightarrow \text{candidate bank}.$$

For an asset $\mathcal{A}$, the visual pipeline provides grounded anchors

$$\mathcal{H}(\mathcal{A}) = \mathcal{P}(\mathcal{A}) \cup \mathcal{K}(\mathcal{A}) \cup \mathcal{R}^{\text{aff}}(\mathcal{A}) \cup \mathcal{R}^{\text{scene}}(\mathcal{A}),$$

where $\mathcal{P}$, $\mathcal{K}$, $\mathcal{R}^{\text{aff}}$, and $\mathcal{R}^{\text{scene}}$ denote part regions, semantic keypoints, affordance regions, and scene-level regions, respectively. Each anchor h has a grounded functional affordance $\phi(h)$ and a compatible skill set $\mathcal{S}(h)$. For each $s \in \mathcal{S}(h)$, we store

$$\mathcal{B}_{h,s} = \{a_{h,s}^{(i)}\}_{i=1}^{N_{h,s}}, \qquad a_{h,s}^{(i)} = \left(s, o(h), h, \phi(h), x^{(i)}, \theta^{(i)}, \tau^{(i)}, v^{(i)}, d^{(i)}\right).$$

Here, $o(h)$ is the object or scene associated with anchor h. Candidate fields are populated progressively with targets, parameters, trajectories, optimization outputs, and validation metadata; only candidates that pass physics validation are retained as final action annotations. This schema separates physical feasibility from functional suitability while preserving multiple valid actions for each anchor–skill pair.

Examples. For a mug, the handle is assigned a grasp-for-use affordance and supports parallel-jaw and dexterous grasp banks with varied contacts, orientations, and approach directions. The body can still support physical or stabilizing grasps, but these candidates receive different functional metadata because they may obstruct pouring or access to the opening. For a cabinet, the same handle anchor can support grasp and articulation banks, while a scene-level free-space anchor supports interaction-ready navigation poses. Thus, each asset is represented by structured, function-conditioned candidate banks rather than a single canonical action.

E.2 General Physics-based Action Generation Pipeline

E.2.1 Candidate Target Generation

Candidate target generation converts language and visual annotations into concrete anchors:

$$\text{skill } s + \mathcal{Y}^{\text{lang}} + \mathcal{Y}^{\text{vis}} \rightarrow \mathcal{C}_s(\mathcal{A}) \rightarrow \text{sampled and filtered targets}.$$

Table 6: Symbols in the unified action annotation schema.

Symbol	Meaning	Representative contents
$\mathcal{A}$	Input asset	Object or room-scale scene.
$\mathcal{H}(\mathcal{A})$	Grounded anchors	Parts, keypoints, affordance regions, and scene regions.
$h, \phi(h)$	Anchor and functional affordance	Mug handle with grasp-for-use; cabinet handle with grasp-for-opening; free space with navigation approach.
$\mathcal{S}(h)$	Compatible skills	Grasp, dexterous grasp, articulation, insertion, hanging, deformable manipulation, or navigation.
$\mathcal{B}_{h,s}$	Candidate bank	Candidates for skill s at anchor h; final retained entries have passed validation.
$o(h)$	Associated instance	Object or scene containing anchor h.
$a_{h,s}^{(i)}$	Action candidate	One target, parameterization, trajectory, metadata record, and diversity descriptor.
$x^{(i)}$	Concrete target	Contact point, surface patch, support edge, opening frame, handle point, or base pose.
$\theta^{(i)}$	Action parameters	Pose, width, hand configuration, direction, joint range, or base orientation.
$\tau^{(i)}$	Trajectory	Approach-contact-retreat motion, articulation waypoints, or navigation path.
$v^{(i)}$	Validation metadata	Collision, IK, stability, task progress, robustness, and functional success.
$d^{(i)}$	Diversity descriptor	Target location, approach direction, contact mode, trajectory family, or embodiment profile.

The skill-compatible regions are

$$\mathcal{C}_s(\mathcal{A}) = \Gamma_s\big(\mathcal{Y}^{\mathrm{lang}}, \mathcal{Y}^{\mathrm{vis}}, \mathcal{G}(\mathcal{A})\big)\,,$$

where $\mathcal{Y}^{\mathrm{lang}}$ contains language semantics and functional priors, $\mathcal{Y}^{\mathrm{vis}}$ contains grounded parts, keypoints, affordance regions, and scene maps, $\mathcal{G}(\mathcal{A})$ denotes asset geometry, and Γ_s is the skill-specific region-construction operator. Each region $c \in \mathcal{C}_s(\mathcal{A})$ is paired with a functional affordance $\phi(c)$, preventing uniform sampling over semantically irrelevant geometry.

Table 7: Candidate-region sources, sampling methods, and principal filters.

Skill	Region source	Sampling	Main filters
Functional and physical grasp	Part mask, language, geometry	FPS on eligible surfaces	Part size, contact area, normals, gripper width, collision, reachability, and affordance consistency.
Dexterous grasp	Part/affordance region	Contact-set or FPS seeds	Joint limits, self-collision, object penetration, support, and functional-region consistency.
Garment manipulation	Semantic keypoints	Keypoint perturbations and bimanual pairs	Visibility, confidence, pair distance, bimanual reachability, and deformation feasibility.
Articulation	Handle or movable-part region	FPS or keypoint sampling	Motion-axis consistency, contact maintenance, collision, reachability, and actuation direction.
Insertion and hanging	Opening, socket, hook, or support edge	Region and frame sampling	Size, alignment, clearance, support stability, and collision-free approach.
Navigation	Occupancy/BEV free space	Grid sampling or 2D FPS	Traversability, clearance, connectivity, visibility, and manipulation reachability.

Table 8: Trajectory-generation families and representative templates.

Constraint family	Representative skills	Main inputs	Template
Object property	Grasp, dexterous grasp, fold, fling	Geometry, part masks, keypoints, grasp frames, garment scale	Pre-grasp/contact/retreat motion or keypoint-conditioned bimanual motion.
Object trajectory	Open, close, rotate, push	Joint axis/state, moving-part transform, handle pose	End-effector waypoints that follow the moving part while preserving grasp or contact.
Object vector	Insert, hang, pour, place	Insertion axes, support edge, pour direction, surface normal	Alignment, approach, execution, release, and retreat along semantic/geometric vectors.
Scene trajectory	Navigation and approach	Occupancy/BEV map, free space, target region, base model	A* seed followed by embodiment-aware local refinement.

For a surface region c, spatially diverse anchors are sampled by FPS:

$$p_1 \sim \mathcal{P}_c, \qquad p_j = \arg\max_{p \in \mathcal{P}_c} \min_{1 \le k < j} \|p - p_k\|_2,$$

where $\mathcal{P}_c$ contains the surface points inside c. Local normals, tangents, curvature, and reference frames are retained for later pose and trajectory generation. Semantic keypoints can be used directly or locally perturbed; bimanual pairs are additionally filtered by distance, symmetry, visibility, and reachability. For scene-level skills, we sample

$$q_j = (x_j, y_j, \psi_j)$$

from traversable cells, with ψ_j oriented toward the target object or affordance.

The retained targets are

$$\mathcal{T}_{c,s} = \{x_{c,s}^{(j)}\}_{j=1}^{M_{c,s}},$$

and seed the corresponding bank $\mathcal{B}_{h,s}$. At this stage, candidates contain target-level information; later stages instantiate parameters and trajectories, optimize them for the asset and embodiment, and attach physics-validation metadata. For example, mug-handle samples seed functional grasp candidates, garment keypoints seed bimanual fold or fling candidates, and cabinet handles seed both grasp and articulation candidates. Room-scale free-space samples additionally seed approach poses around the cabinet.

E.3 Trajectory Generation

Each retained target is converted into a template-level trajectory

$$\tau_0^{(i)} = \mathcal{G}_s\Big(h, \phi(h), x^{(i)}, \theta^{(i)}; \mathcal{A}\Big),$$

which is subsequently optimized and validated. We group generators by the source of their constraints.

Object-property-conditioned templates. For a parallel-jaw grasp with pose $T_{\text{grasp}} = [R_{\text{grasp}}, p_{\text{grasp}}]$ and approach direction $\mathbf{a}_{\text{grasp}}$, we use

$$T_{\text{pre}} = [R_{\text{grasp}}, p_{\text{grasp}} - \delta_{\text{pre}}\mathbf{a}_{\text{grasp}}], \qquad T_{\text{ret}} = [R_{\text{grasp}}, p_{\text{grasp}} + \delta_{\text{ret}}\mathbf{r}]$$

$$\tau_0 = (T_{\text{pre}}, T_{\text{grasp}}, T_{\text{close}}, T_{\text{ret}}).$$

Here, $\mathbf{r}$ is a retreat direction and T_{close} denotes gripper closure at the grasp pose. Dexterous templates additionally specify a palm pose, contact seeds, joint configuration, and closure schedule.

Garment templates are generated from semantic keypoints. For an upper-body garment, a sleeve endpoint is reflected across the shoulder–bottom fold line,

$$k^{\star}_{\mathrm{lslv}} = \mathrm{Ref}_{\ell(k_{\mathrm{lsh}},k_{\mathrm{lb}})}(k_{\mathrm{lslv}}),$$

and the bimanual sequence lifts, transfers, and places the sleeve near this target. Bottom-up folds move hem keypoints toward the shoulder region. Pants use a side-to-center fold followed by one or more bottom-to-waist folds. Fling templates select two keypoints and scale lift and stretch distances with their separation while enforcing bimanual reachability.

Object-trajectory-conditioned templates. For articulated parts, let $T_{\mathrm{part}}(q)$ be the moving-part pose at joint state q. After establishing the initial contact,

$$T_{\mathrm{rel}} = T_{\mathrm{part}}(q_{\mathrm{init}})^{-1} T^{\mathrm{init}}_{\mathrm{eef}}, \qquad T_{\mathrm{eef}}(q) = T_{\mathrm{part}}(q) T_{\mathrm{rel}}.$$

Waypoints are generated by interpolating q from the initial to the target state. This covers revolute and prismatic opening/closing, partial pulls, lid motion, button rotation, and short pushes; push-based actions can instead maintain surface contact without a fixed grasp.

Object-vector-conditioned templates. For insertion, both $\mathbf{u}_{\mathrm{act}}$ and $\mathbf{u}_{\mathrm{pas}}$ denote the positive physical insertion direction. When the passive-side geometry initially provides an outward surface normal, we negate it while constructing the passive insertion frame. The active and passive axes are then aligned as

$$R\mathbf{u}_{\mathrm{act}} \approx \mathbf{u}_{\mathrm{pas}}.$$

then the trajectory approaches and translates along the passive axis:

$$p_{\mathrm{pre}} = p_{\mathrm{tar}} - \delta_{\mathrm{pre}}\mathbf{u}_{\mathrm{pas}}, \qquad p(t) = p_{\mathrm{pre}} + t\delta_{\mathrm{ins}}\mathbf{u}_{\mathrm{pas}}.$$

Hanging templates approach a support edge or hook, establish support, and lower or release under gravity. Pouring rotates the grasped object toward a receiving region, while placement aligns with a support normal before release and retreat.

Scene-trajectory-conditioned templates. For a target base pose $q = (x, y, \psi)$, A* provides a coarse path on the occupancy or BEV grid. Because this seed does not encode turning radius, coupled base velocities, or other embodiment constraints, it is refined by a DWB-style local rollout optimizer before validation.

E.4 Trajectory Optimization

Optimization adapts each template to the asset and robot embodiment $\mathcal{R}$:

$$(\theta^{(i)}_{*}, \tau^{(i)}_{*}) = \arg\min_{\theta,\tau} \mathcal{E}_s\Big(\theta, \tau; h, \phi(h), x^{(i)}, \mathcal{A}, \mathcal{R}\Big),$$

with

$$\mathcal{E}_s = \lambda_{\mathrm{func}}E_{\mathrm{func}} + \lambda_{\mathrm{task}}E_{\mathrm{task}} + \lambda_{\mathrm{geom}}E_{\mathrm{geom}} + \lambda_{\mathrm{contact}}E_{\mathrm{contact}} + \lambda_{\mathrm{coll}}E_{\mathrm{coll}} + \lambda_{\mathrm{kin}}E_{\mathrm{kin}} + \lambda_{\mathrm{smooth}}E_{\mathrm{smooth}}.$$

The active terms depend on the skill; the result is still a candidate and must pass physics validation.

Dexterous and contact-aware optimization. Let q_{hand} and T_{hand} denote hand joints and root pose, and let $p_c = \mathrm{FK}_c(T_{\mathrm{hand}}, q_{\mathrm{hand}})$ be a contact-link position. Contacts are kept on the grounded functional region through

$$E_{\mathrm{region}} = \sum_{c\in\mathcal{C}} \mathrm{dist}(p_c, h)^2.$$

For a selected contact set $\mathcal{C}$, a wrench term can additionally encourage task-relevant support:

$$E_{\mathrm{wrench}} = \sum_{j} \left\| w_j - \sum_{c\in\mathcal{C}} G_c f_{c,j} \right\|_2^2, \qquad f_{c,j} \in \mathcal{K}_{\mu_c}.$$

Here, w_j is a target wrench, G_c maps a contact force to an object wrench, and $\mathcal{K}_{\mu_c}$ is the friction cone at contact c. Signed-distance, clearance, joint-limit, and self-collision penalties discourage penetration and infeasible configurations. The approach offset, direction, and intermediate waypoints are jointly refined to obtain a collision-aware approach-contact-close-retreat motion.

Table 9: Representative trajectory-optimization variables and objectives.

Skill family	Optimized variables	Main objectives	Stored outputs
Dexterous/contact	Hand pose, finger joints, contacts, approach	Region consistency, wrench support, collision, joint limits, self-collision	Final hand configuration, contact set, pre-grasp, and objective terms.
Garment/adaptive	Waypoints, fold/place targets, timing	Keypoint consistency, scale adaptation, reachability, smoothness, collision	Geometry-adapted bimanual trajectory and task parameters.
Articulation	Contact pose, waypoints, joint schedule, pull/push direction	Part tracking, contact maintenance, collision, reachability, joint progress	Refined opening, closing, rotation, or push trajectory.
Vector-conditioned	Alignment and execution poses, depth, release	Vector alignment, clearance, support stability, task progress	Refined insertion, hanging, pouring, or placement trajectory.
Scene-level	Base path, commands, orientation, approach pose	Path following, local dynamics, clearance, smoothness, manipulation reachability	Embodiment-feasible path and interaction-ready final pose.

Geometry-adaptive waypoint optimization. For template waypoints p_t^0, we optimize corrections

$$\{\Delta p_t\}_{t=1}^{T} = \arg\min_{\{\Delta p_t\}} \sum_t \|\Delta p_t\|_2^2 + \lambda_{\text{geom}} E_{\text{geom}} + \lambda_{\text{reach}} E_{\text{reach}} + \lambda_{\text{smooth}} E_{\text{smooth}},$$

$$p_t^* = p_t^0 + \Delta p_t.$$

Garment fold displacements and fling amplitudes are scaled by keypoint distances, bounded, and adjusted for bimanual reachability. This avoids over-folding short garments, under-folding long garments, and bimanual trajectories outside the reachable workspace.

Articulation refinement. For object joint state q_t, the end-effector tracks the ideal pose induced by the moving part:

$$E_{\text{art-track}} = \sum_t \left\| T_{\text{eef},t} \ominus \widehat{T}_{\text{eef}}(q_t) \right\|^2.$$

Contact-maintenance, collision, reachability, and smoothness terms refine the contact pose, waypoint spacing, pull direction, and joint schedule. Handle-based actions preserve grasp contact, while push-based closing and button actions maintain bounded surface contact along the valid motion direction.

Vector-conditioned refinement. For insertion, the alignment cost is

$$E_{\text{align}} = 1 - (R\mathbf{u}_{\text{act}})^{\top}(\mathbf{u}_{\text{pas}}),$$

and is combined with insertion-depth and collision terms. Hanging optimization adjusts support contact and release so the post-release configuration is gravity stable. Pouring refines rotation toward the receiving region subject to wrist and collision limits, and placement refines support-normal alignment, release height, and retreat direction.

Scene-level optimization. DWB-style optimization samples short-horizon controls under the robot's base model. Differential-drive, Ackermann, and holonomic embodiments use (v, ω), (v, δ), and (v_x, v_y, ω) controls, respectively. Rollouts are scored by

$$J_{\text{DWB}} = w_{\text{path}} J_{\text{path}} + w_{\text{goal}} J_{\text{goal}} + w_{\text{obs}} J_{\text{obs}} + w_{\text{clear}} J_{\text{clear}} + w_{\text{smooth}} J_{\text{smooth}} + w_{\text{reach}} J_{\text{reach}}.$$

Table 10: Validation settings used across skill families.

Setting	Representative skills	Principal checks	Purpose
Floating gripper	Parallel-jaw grasp, insertion, hanging, placement, handle articulation	Contact, penetration, local collision, stability, task progress, release outcome	Tests reusable object-centric interactions without over-constraining them to one arm pose.
Floating dexterous hand	Dexterous grasp and articulation	Joint validity, hand-object contact, self-collision, object stability, contact maintenance	Verifies that the hand actually constrains the object.
Full manipulator	Selected garment and long-horizon deformable skills	IK, singularity, arm/self-collision, entanglement, task outcome	Captures feasibility that depends on the arm and body.
Base-aware robot	Navigation and approach-pose annotation	Traversability, local dynamics, turning radius, clearance, path connectivity, manipulation reachability	Verifies embodiment-feasible paths and useful final base poses.

The terms score path tracking, goal error, obstacle collision, clearance, control smoothness, and terminal manipulation reachability. The reachability term favors terminal poses from which the target anchor is manipulable, using reachability maps, IK queries, or workspace heuristics. We store the optimized parameters and trajectory, the objective breakdown, embodiment-feasibility flags, functional-consistency status, and diversity descriptor for subsequent validation and sampling.

E.5 Physics Validation

Physics validation is the final candidate-generation filter before entry into the annotation bank. Full-robot execution feasibility is evaluated separately during embodiment-specific candidate retrieval and execution. For an optimized candidate, we run skill-specific rollouts under validation embodiment e_{val} and perturbations Ξ:

$$v^{(i)} = \mathcal{V}_s\left(a_{h,s}^{(i)}, \mathcal{A}, e_{\text{val}}, \Xi\right).$$

The result records success, collisions, contact stability, task progress, robustness, rollout length, perturbations, and embodiment. Failed candidates are rejected.

Validation embodiments. Floating grippers or hands validate local object interaction without tying an annotation to one arm configuration. They are used for object-centric grasps, insertion, hanging, placement, pouring, and many handle-based articulation candidates. Non-floating validation is reserved for embodiment-dependent skills: full-manipulator validation checks selected long-horizon deformable interactions for IK, singularities, self-collision, and arm–object or arm–cloth interaction, while base-aware validation checks navigation for local dynamics, clearance, and manipulation reachability.

Robustness tests. For grasp and dexterous-grasp candidates, we sample $K = 8$ random gravity directions with increased magnitude,

$$\mathbf{g}_k = 1.5 g_0 \mathbf{u}_k, \qquad \mathbf{u}_k \sim \mathbb{S}^2, \qquad k = 1, \ldots, 8,$$

where g_0 is the nominal gravity magnitude. These perturbations are applied after the gripper or hand closes; during approach and closure, gravity is disabled so that the object stays static (App. F).

Table 11: Skill-specific validation criteria and representative detected failures.

Skill family	Required checks	Representative failure signal
Parallel-jaw grasp	Collision-free approach, closure contact, stability, randomized gravity, disturbances	Object drops, slips, or rotates beyond tolerance; pre-grasp penetrates geometry.
Dexterous grasp	Joint limits, self-collision, penetration, contact coverage, stability, gravity robustness	Fingers surround the object visually but fail to constrain it, or penetrate/self-collide.
Articulation	Contact maintenance, joint progress, collision, reachability, grasp/push validity	Motion stalls, the gripper jams, or the hand loses contact or becomes trapped.
Insertion	Axis alignment, entry clearance, insertion depth, post-insertion stability	Collision at the entrance or insufficient depth.
Hang/place/pour	Alignment, clearance, release stability, task progress	Object falls after release, settles unstably, or collides during execution.
Garment/deformable	Full-arm IK, singularities, arm-cloth collision, entanglement, geometric outcome	Garment wraps around the arm, trajectory becomes unreachable, or intended state is not achieved.
Navigation	Traversability, clearance, connectivity, base dynamics, final-pose reachability	A* path is dynamically infeasible or the final pose cannot support manipulation.

Depending on the skill, validation can also perturb object pose, gripper or hand pose, trajectory timing, or apply small external forces. These tests reject candidates that succeed only under a single idealized initialization or load direction.

Parallel execution. Validation is parallelized across assets, skills, anchors, candidates, and perturbation trials. Jobs with similar simulation settings are batched by skill and embodiment, and failed rollouts terminate early after penetration, contact loss, object drop, stalled task progress, or joint-limit violation. This parallel filtering is necessary because candidate banks can be large and pass rates are low for several contact-rich and deformable skills.

E.6 Physics-aware Augmentation

After validation, we expand each bank using bounded local perturbations and symmetry-aware transforms while preserving s, h, and $\phi(h)$. For a validated candidate,

$$\mathcal{U}(a_{h,s}^{(i)}) = \{\tilde{a}_{h,s}^{(i,j)}\}_{j=1}^{M_i}, \qquad \widetilde{\mathcal{B}}_{h,s} = \mathcal{B}_{h,s} \cup \bigcup_{a_{h,s}^{(i)} \in \mathcal{B}_{h,s}} \mathcal{U}(a_{h,s}^{(i)}).$$

Local perturbations. We apply bounded, skill-dependent perturbations to targets, parameters, and waypoints:

$$\tilde{x}^{(i,j)} = x^{(i)} + \Delta x^{(j)}, \qquad \tilde{\theta}^{(i,j)} = \theta^{(i)} \oplus \Delta\theta^{(j)},$$
$$\tilde{\tau}^{(i,j)} = \{T_t \oplus \Delta T_t^{(j)}\}_{t=1}^{T}.$$

Surface targets remain inside the same part or affordance region, keypoint targets remain inside local confidence regions, and navigation targets remain in traversable reachable space. Perturbed quantities can include grasp center and yaw, palm pose and finger seeds, fold or fling targets, handle contact and pull direction, insertion alignment, support contact, or base pose. The functional affordance and task intent are preserved.

Symmetry-aware augmentation. When visual-language annotations indicate a valid symmetry group $\mathcal{G}_{\mathrm{sym}}(o)$, we transform the target, parameters, trajectory, and relevant vectors consistently:

$$\tilde{a}_{h,s}^{(i,g)} = g \cdot a_{h,s}^{(i)}, \qquad g \in \mathcal{G}_{\mathrm{sym}}(o),$$

$$\tilde{T} = GT, \qquad \tilde{\tau} = \{GT_1, \ldots, GT_T\}, \qquad \tilde{\mathbf{u}} = R_G\mathbf{u}.$$

Here, G is the rigid transform induced by g and R_G is its rotational component. Examples include rotating bottle grasps around a valid symmetry axis, transferring candidates across repeated equivalent handles, or mirroring bilateral garment folds. Symmetry is applied only when it preserves the functional affordance; a mug handle grasp, for example, is not rotated onto the mug body.

Feasibility rechecks. Augmented candidates are accepted only when

$$\mathbb{I}_{\text{accept}} = \mathbb{I}[C_{\text{region}} \wedge C_{\text{func}} \wedge C_{\text{coll}} \wedge C_{\text{kin}} \wedge C_{\text{task}}] .$$

The checks enforce region membership, functional consistency, collision and clearance, embodiment constraints, and task-specific validity. Inexpensive pose-level augmentations use geometry and short-rollout physics checks, whereas high-risk full-manipulator deformable trajectories are returned to full physics validation before retention.

Table 12: Representative physics-aware augmentation strategies and required rechecks.

Skill/object	Perturbation or transform	Constraint	Recheck
Parallel-jaw grasp	Contact center, yaw, width, pre-grasp offset	Remain on the same graspable region and grasp family	Collision, width, approach, and stability.
Dexterous grasp	Palm pose, contact seeds, joint offsets, closure timing	Preserve functional region and feasible hand configuration	Joint limits, self-collision, penetration, and object constraint.
Garment fold/fling	Keypoints, fold line, place target, lift/stretch amplitude	Stay within confidence regions and preserve fold/fling intent	Bimanual reachability, cloth clearance, entanglement, and outcome.
Articulation	Handle point, grasp orientation, pull direction, waypoint spacing, joint range	Preserve valid joint direction and contact mode	Contact maintenance, joint progress, collision, and jamming.
Insertion and hanging	Entry or support pose, depth, axis, release pose	Preserve alignment or gravity-stable support	Clearance, task progress, and post-release stability.
Navigation	(x, y, ψ) around an approach pose	Remain in traversable, reachable free space and face the target	Connectivity, DWB feasibility, clearance, and manipulation reachability.
Bottle/round knob	Rotation around annotated symmetry axis	Transform lies in a functionally equivalent region	Collision, stability, rotation direction, and wrist feasibility.
Repeated handles / bilateral garments	Copy, translation, or mirror transform	Source and target regions are functionally equivalent	Local collision context, reachability, and task validity.

Physics-aware augmentation therefore increases diversity around validated functional anchors without treating geometric transformations as automatically valid labels.

F PRIMITIVE-SPECIFIC ACTION ANNOTATION

This section instantiates the unified candidate-bank schema for five primitive families: parallel-jaw and dexterous grasping (Grasp, DexGrasp, BiGrasp, BiDexGrasp), articulation, insertion and hanging, garment manipulation (Deformable), and navigation, which together cover the nine skill families. Each candidate

$$a_{h,s}^{(i)} = (s, o(h), h, \phi(h), x^{(i)}, \theta^{(i)}, \tau^{(i)}, v^{(i)}, d^{(i)})$$

stores a grounded target, primitive-specific parameters, optional waypoints, validation metadata, and diversity attributes. All annotators follow the same sequence of target generation, pose or trajectory construction, optional optimization, physics validation, and serialization.

Parallel-jaw and dexterous grasp annotation. Both grasp annotators use geometry-based proposals followed by simulation validation; dexterous grasps additionally optimize wrist pose, hand joints, and semantic contacts.

Single-arm parallel-jaw proposals (Grasp). Single-arm grasps use antipodal sampling restricted to the grounded part and affordance regions compatible with the skill. Surface points are sampled on the object mesh; for each point, a ray cast along the inward surface normal finds the opposing contact, and the pair is kept if its width is within the maximum gripper aperture and the two contact normals are approximately antipodal. For each retained contact pair, gripper orientations are sampled around the grasp axis at a fixed standoff distance, with small lateral perturbations along the axis for diversity. Candidates in collision with the object are rejected, and the remaining candidates are validated in simulation as described below and expanded by physics-aware augmentation (App. E.6).

Bimanual rim proposals and poses (BiGrasp). We convert each USD asset to a triangle mesh, sample 10^5 area-weighted surface points, and construct each local neighborhood from its $K = 64$ nearest samples. Rim-style bimanual candidates are prefiltered from the top 30% of object height and finally selected from the top 20% band. Each point i receives

$$S_i = \mathbf{1}_{\text{top}} + 0.4E_i + 1.5B_i + 0.8P_i + 0.8A_i,$$

where E_i, B_i, P_i, and A_i measure normal variation, bilateral rim support, XY convex-hull proximity, and outward accessibility, and $\mathbf{1}_{\text{top}}$ indicates membership in the desired upper band. We retain anchors satisfying

$$P_i > 0.1, \qquad A_i > 0.05, \qquad w_i \le 0.06\text{m}, \qquad \angle(n_i, \hat{z}) \le 60^\circ,$$

where w_i is the estimated local gripper width and n_i is the surface normal. We then apply 0.005m non-maximum suppression and keep at most 500 anchors.

For anchor position p_a and normal n_a, the local frame is

$$z_a = n_a, \qquad x_a = \text{PCA}_{\text{tangent}}(\mathcal{N}(p_a)), \qquad y_a = z_a \times x_a,$$

with $R_a = [x_a, y_a, z_a]$, where x_a is estimated by PCA over the local XY neighborhood. The factor $R_x(\pi)$ below converts this frame to the gripper convention. We evaluate up to 3000 opposite-side pairs and score anchor quality, along-rim consistency, across-object separation, center balance, and tangent consistency. For each retained anchor,

$$R_g = R_a R_x(\pi), \qquad p_g = p_a + (0.14\text{m})n_a, \qquad p_{\text{pre}} = p_g + (0.20\text{m})n_a.$$

Pairs whose transformed gripper AABBs overlap are rejected before simulation.

Parallel-jaw validation. The asset is converted to a rigid body and floating grippers are simulated in Isaac Sim with static and dynamic friction set to 2.0. Because objects need not rest upright on a support at real grasp time, grasp poses are not filtered by object orientation, and the support surface is not modeled during approach; gravity is therefore disabled during approach and closure so that the object stays static. After closure, stability is tested under the multi-direction gravity perturbations of App. E.5. Grippers start at finger target 0.04m, approach for 80 steps, close to 0.0m over 32 steps, hold for 50 steps, then lift the object by 1.0m over 200 steps with gravity enabled. Let z_{close} be the object height after closure and hold, and z_{lift} its final height after lifting. A pair passes when

$$z_{\text{lift}} - z_{\text{close}} \ge 0.95\text{m}.$$

Accepted single-gripper poses are stored as $[x, y, z, q_w, q_x, q_y, q_z]$ with anchor and validation metadata; bimanual candidates additionally store paired left/right poses and pairing metadata.

Single-hand dexterous grasps (DexGrasp). Single-hand dexterous grasps are generated by an optimizer that builds on the GPU-parallel bilevel formulation of BODex, restricts contact targets to the VLM-grounded functional regions (e.g., a mug handle rather than the whole object surface), and uses optimization parameters tuned for the Dex3-1 hand. In the protocol-matched comparison of Sec. 5.4, BODex and our generator share the hand, candidate budget, validator, and execution protocol, so the reported difference reflects candidate generation, including the grounded contact targets. This comparison therefore evaluates our DexGrasp generator as a whole; separating the effect of grounding from that of the optimization settings would require running the same generator without the region constraint, which we leave to future work. Candidates are validated with the floating Dex3-1 hand as in the bimanual case below.

Bimanual dexterous proposals and optimization (BiDexGrasp). Dexterous proposals are category-conditioned: category 1 targets top rims or edges, category 2 lower-edge support, category 3 bimanual side holds, and category 4 convex side or top holds. Each category uses its own height, normal, width, perimeter, accessibility, and suppression rules. The object is voxelized into an SDF $d : \mathbb{R}^3 \to \mathbb{R}$ at $5\mathrm{mm}$ resolution with 8 padding voxels, using $d(x) > 0$ outside the object and $d(x) < 0$ for penetration. The hand is represented by semantically labeled collision spheres for thumb, finger pads, palm support, and avoid regions. Category-specific surface patches provide contact targets and desired contact sides.

For each hand, we optimize

$$x = [t_x, t_y, t_z, \omega_x, \omega_y, \omega_z, q_1, \ldots, q_m], \qquad R = R_0 \operatorname{Exp}(\omega),$$

under bounded wrist and joint updates, where t, ω, and q are the wrist translation, rotation-vector update, and hand joint angles, and R_0 is the seed wrist orientation. The staged objective is

$$J_s = J_{\text{clear}} + J_{\text{target}} + J_{\text{normal}} + J_{\text{opp}} + J_{\text{coll}} + J_{\text{self}} + J_{\text{prior}} + J_{\text{cat}}.$$

For contact sphere center c_i, radius r_i, activity weight λ_i, and desired clearance ϵ_i,

$$J_{\text{clear}} = \sum_i w_c \lambda_i (d(c_i) - r_i - \epsilon_i)^2.$$

Using the SDF projection $\pi(c_i) = c_i - d(c_i)\hat{\nabla} d(c_i)$, where $\hat{\nabla} d$ is the normalized SDF gradient, target and normal alignment are

$$J_{\text{target}} = \sum_i w_p \lambda_i \left\| D_s R_c^\top (\pi(c_i) - p_i^\star) \right\|^2, \qquad J_{\text{normal}} = \sum_i w_n \lambda_i (1 - \hat{\nabla} d(c_i)^\top n_i^\star)^2.$$

Here $p_i^\star$ and $n_i^\star$ are the assigned target point and normal, R_c is the local contact frame, and D_s is a stage-dependent diagonal weighting matrix. For paired contacts,

$$J_{\text{opp}} = \sum_{(i,j)} w_o \left(1 - \frac{c_j - c_i}{\|c_j - c_i\|}^\top o_{ij}^\star \right)^2,$$

where $o_{ij}^\star$ is the desired opposition direction. Object and self-collision use hinge penalties,

$$J_{\text{coll}} = \sum_j w_{\text{coll},j} [m_j - (d(c_j) - r_j)]_+^2, \qquad J_{\text{self}} = \sum_{(i,j)} w_{\text{self}} [r_i + r_j + m_{\text{self}} - \|c_i - c_j\|]_+^2,$$

where m_j is the required object-clearance margin and m_{self} is the self-collision margin. We use $4\mathrm{mm}$ clearance for inactive spheres, $10\mathrm{mm}$ for palm spheres, and a $2\mathrm{mm}$ self-collision margin. Seed regularization relative to (t_0, R_0, q_0) is

$$J_{\text{prior}} = w_t \|t - t_0\|^2 + w_R \|\log(R_0^\top R)\|^2 + w_q \|q - q_0\|^2,$$

and J_{cat} encodes category-specific contact priors.

We solve with bounded L-BFGS-B for at most 140 iterations, $f_{\text{tol}} = g_{\text{tol}} = 10^{-8}$, finite-difference step 10^{-4}, and at most 80 line-search steps. Pregrasp, grasp, and squeeze stages progressively

increase contact, target, and collision weights. Category 3 additionally performs a fast translation and wrist-roll pose search before refinement.

Category 1 and 2 solutions are paired across opposite edge anchors, while category 3 solutions are paired across opposite side anchors with similar height and cross-section. We reject pairs with overlapping hand AABBs, wrist separation below $0.10\mathrm{m}$, inconsistent side-hold directions, or excessive combined penetration. Each pair stores

$$\theta^{(i)} = (T_L, q_L, \mathcal{C}_L, T_R, q_R, \mathcal{C}_R),$$

where T, q, and $\mathcal{C}$ denote wrist pose, hand joints, and semantic contact assignments. Floating-hand validation executes pregrasp, grasp, squeeze, settle, gravity hold, and a $0.3\mathrm{m}$ lift. A candidate passes if the object remains supported and drops by at most $2\mathrm{cm}$ during lift and hold.

Articulation waypoint annotation. We recover the USD joint model, generate a primitive-specific contact pose, analytically map joint motion to end-effector waypoints, and validate the trajectory in simulation.

Joint and contact extraction. After fixing the asset base to the world, we discover each revolute or prismatic joint and store

$$\mathcal{J} = (j, \mathrm{type}, a, c, q_{\min}, q_{\max}, b_{\mathrm{parent}}, b_{\mathrm{child}}),$$

where j identifies the joint, type is revolute or prismatic, a and c are the world-space axis and pivot, and $b_{\mathrm{parent}}, b_{\mathrm{child}}$ are the connected bodies; revolute limits are converted to radians. Let q_0 denote the initial joint state. Let $q_{\mathrm{open}} \in \{q_{\min}, q_{\max}\}$ denote the limit in the selected opening direction. We set

$$q^\star = \mathrm{clip}(q_0 + 0.7(q_{\mathrm{open}} - q_0), q_{\min}, q_{\max}), \qquad \Delta q^\star = q^\star - q_0.$$

Closing analogously moves toward the selected closed limit, optionally with a small overshoot before the final target is clamped to $[q_{\min}, q_{\max}]$.

Open-by-handle restricts grasp generation to handle geometry and filters approach, accessibility, collision, and panel-normal consistency. Open-on-edge samples outer child-link edges far from the hinge to increase moment arm. Close-by-push samples inset points on flat outer panels, again favoring large hinge distance, and aligns the pushing direction with closure. Each contact stores $T_0 = (p_0, R_0)$ plus joint, child-link, approach, and contact-mode metadata.

Analytic waypoints and validation. For a revolute joint with displacement $\Delta\theta^\star$ and uniformly sampled fractions $\alpha_k \in [0, 1]$,

$$G_k = \mathrm{Exp}(\alpha_k \Delta\theta^\star [a]_\times), \qquad p_k = c + G_k(p_0 - c), \qquad R_k = G_k R_0.$$

For a prismatic joint with displacement $\Delta d^\star$,

$$p_k = p_0 + \alpha_k \Delta d^\star a, \qquad R_k = R_0.$$

We prepend

$$p_{\mathrm{pre}} = p_0 - \delta_{\mathrm{app}} u_{\mathrm{app}},$$

where u_{app} is the approach direction and δ_{app} is a primitive- and geometry-dependent distance from 0.08 to $0.70\mathrm{m}$. Typical rollouts use 100–200 approach or motion steps and 30–64 closing steps. If q_T is the measured final joint state, a candidate passes when

$$|q_T - q_0| \geq 0.95 |\Delta q^\star|$$

and moves in the intended direction. Accepted waypoints are serialized as $[x, y, z, q_w, q_x, q_y, q_z]$ and stored with joint properties, contact and approach parameters, primitive type, and validation metadata.

Insertion and hanging annotation. These pairwise skills use active and passive anchors. For $s \in \{\mathrm{insert}, \mathrm{hang}\}$,

$$\mathcal{B}_{(h_{\mathrm{act}}, h_{\mathrm{pas}}), s} = \{a^{(i)}_{(h_{\mathrm{act}}, h_{\mathrm{pas}}), s}\}_{i=1}^{N_{(h_{\mathrm{act}}, h_{\mathrm{pas}}), s}},$$

where each candidate stores both objects, anchors, affordances, targets, parameters, trajectory, validation, and diversity metadata. Functional and geometric compatibility determines valid pairings.

Table 13: Active and passive roles for insertion and hanging. The passive side provides the receiving or supporting affordance, while the active side provides the object-side affordance that must be aligned or matched.

Skill	Passive side	Active side	Annotation goal
Insertion	Socket, hole, port, opening, container mouth, receptacle	Peg, plug, cable connector, key, tool tip, object to be inserted	Annotate where insertion is possible on the passive object and where the active object should enter, together with their insertion directions.
Hanging	Hook, peg, rod, rail, support edge, hanger slot	Mug handle, bag strap, clothes hanger loop, garment collar point, object loop	Annotate where support is possible on the passive object and which active object region should engage with it. For rigid objects, vectors are aligned; for deformable objects, keypoints are matched.

Table 14: Examples of insertion annotation. The passive object provides the receiving location and direction, while the active object provides the insertion tip and positive insertion direction.

Example	Passive annotation	Active annotation	Candidate meaning
HDMI insertion	HDMI port center, inward insertion direction, port size, port type	HDMI plug tip, plug-forward direction, plug size, plug type	Align the plug-forward vector with the port insertion direction and translate the plug into the port.
Peg-in-hole	Hole center, hole axis, hole radius, hole depth	Peg tip, peg axis, peg radius, peg length	Align the peg axis with the hole axis and insert to a feasible depth with clearance.
Key insertion	Keyhole entry point, keyhole direction, slot type	Key tip, key-forward direction, key blade geometry	Match key type and direction, align the key blade, and move along the keyhole axis.
Tool insertion	Receptacle opening, insertion axis, opening size	Tool tip or shaft endpoint, shaft direction, shaft diameter	Align the tool shaft with the receptacle and insert without collision.

Insertion. The passive frame is

$$F_{\text{pas}}^{\text{ins}} = (p_{\text{pas}}, \mathbf{u}_{\text{pas}}, r_{\text{pas}}, \ell_{\text{pas}}, \text{type}_{\text{pas}}),$$

and the active frame is

$$F_{\text{act}}^{\text{ins}} = (p_{\text{act}}, \mathbf{u}_{\text{act}}, r_{\text{act}}, \ell_{\text{act}}, \text{type}_{\text{act}}).$$

Here p denotes the entry or tip, $\mathbf{u}$ the positive physical insertion direction, r a cross-section descriptor (shown as a scalar size in the compatibility condition below), ℓ the available depth or effective length, and `type` an optional semantic connector type. Under this convention, a candidate orientation $R^{(i)}$ should satisfy

$$R^{(i)}\mathbf{u}_{\text{act}} \approx \mathbf{u}_{\text{pas}};$$

passive outward normals are sign-flipped before this check. Compatibility requires

$$\text{type}_{\text{act}} \sim \text{type}_{\text{pas}}, \qquad r_{\text{act}} + \epsilon_{\text{clear}} \le r_{\text{pas}}, \qquad \angle(R^{(i)}\mathbf{u}_{\text{act}}, \mathbf{u}_{\text{pas}}) \le \epsilon_{\text{angle}}.$$

Here ϵ_{clear} and ϵ_{angle} are clearance and angular tolerances. The approach-align-insert trajectory uses

$$p_{\text{pre}} = p_{\text{pas}} - \delta_{\text{pre}}\mathbf{u}_{\text{pas}}, \qquad p_{\text{ins}} = p_{\text{pas}} + \delta_{\text{ins}}\mathbf{u}_{\text{pas}}.$$

Here δ_{pre} is the pre-entry offset and δ_{ins} is the target insertion depth. Parameters store alignment, direction, depth, and clearance; validation records alignment error, penetration, achieved depth, and post-insertion stability.

Table 15: Examples of hanging annotation. Rigid objects use vector or frame alignment, while deformable objects can use keypoint-to-support matching.

Example	Passive annotation	Active annotation	Candidate meaning
Mug on hook	Hook point or support curve, hook entry direction, local clearance	Mug handle center, vector through handle opening, handle clearance	Align the handle-through vector with the hook direction, move the handle onto the hook, and release after gravity seating.
Bag on peg	Peg tip or support edge, support direction, clearance	Bag strap centerline or loop vector, strap opening size	Place the strap loop over the peg and lower until the bag is supported.
Clothes hanger on rod	Rod support line, approach side, gravity direction	Hanger hook center and hook opening direction	Move the hanger hook over the rod, lower onto the rod, and release.

Hanging. For rigid supports,

$$F_{\text{pas}}^{\text{hang}} = (p_{\text{sup}}, \mathbf{u}_{\text{sup}}, \mathbf{g}, c_{\text{clear}}, \text{type}_{\text{sup}}),$$

where p_{sup} and $\mathbf{u}_{\text{sup}}$ specify the support point and entry direction, $\mathbf{g}$ is the gravity direction, and c_{clear} and type_{sup} encode clearance and support type. The active loop or handle is

$$F_{\text{act}}^{\text{hang}} = (p_{\text{loop}}, \mathbf{u}_{\text{loop}}, r_{\text{loop}}, \text{type}_{\text{loop}}),$$

where p_{loop}, $\mathbf{u}_{\text{loop}}$, and r_{loop} specify the loop center, through-opening direction, and clearance. A candidate orientation $R^{(i)}$ aligns $R^{(i)}\mathbf{u}_{\text{loop}} \approx \mathbf{u}_{\text{sup}}$, engages the support, lowers under gravity, and releases. For deformable objects, the active side is instead

$$F_{\text{act,def}}^{\text{hang}} = (p_{\text{kp}}, \text{type}_{\text{kp}}, c_{\text{conf}}),$$

where p_{kp}, type_{kp}, and c_{conf} denote the keypoint position, semantic role, and confidence. A semantic keypoint such as a collar or loop is then matched to the support without assuming a rigid active vector. The trajectory is

$$\text{approach} \rightarrow \text{engage support} \rightarrow \text{lower under gravity} \rightarrow \text{release or retreat}.$$

Validation and metadata. Insertion checks semantic type, vector alignment, opening clearance, approach collision, achieved depth, and post-insertion alignment. Hanging checks support compatibility, clearance, contact formation, post-release gravity stability, and, for deformables, entanglement and over-stretching. We store

$$v^{(i)} = \{\text{align_err}, \text{clearance}, \text{collision}, \text{depth}, \text{stability}, \text{release_success}, \text{failure_reason}\},$$

while $d^{(i)}$ records angle, depth, approach offset, support point, loop orientation, keypoint choice, and release height.

Garment and deformable-object trajectory annotation. Garment actions use semantic keypoints in a canonical garment frame and type-specific bimanual templates. The detected set $\mathcal{K}_{\text{garment}} = \{k_1, \ldots, k_M\}$ depends on whether the asset is an upper-body garment, dress, pants, or rectangular cloth.

Folding and flinging templates. For upper-body garments, $k_{\text{lslv}}, k_{\text{rslv}}$ denote the `top_left` and `top_right` sleeve endpoints; $k_{\text{lsh}}, k_{\text{rsh}}$ and $k_{\text{lb}}, k_{\text{rb}}$ denote the shoulder points and bottom corners. Writing $\ell(a, b)$ for the line through keypoints a, b and Ref_ℓ for reflection across that line, the sleeve endpoints are reflected across shoulder–bottom fold lines,

$$\ell_{\text{L}} = \ell(k_{\text{lsh}}, k_{\text{lb}}), \qquad \ell_{\text{R}} = \ell(k_{\text{rsh}}, k_{\text{rb}}),$$
$$k^{\star}_{\text{lslv}} = \text{Ref}_{\ell_{\text{L}}}(k_{\text{lslv}}), \qquad k^{\star}_{\text{rslv}} = \text{Ref}_{\ell_{\text{R}}}(k_{\text{rslv}}).$$

A second stage nominally moves each bottom corner to its corresponding shoulder point, with the practical targets shifted or clamped as needed for stable bimanual execution. Pick points are shifted

Table 16: Common validation checks and failure cases for insertion and hanging annotations.

Skill	Check	Common failure case	Rejection criterion
Insertion	Semantic compatibility	Plug and port types do not match, or active/passive roles are reversed.	Connector type, category, or functional affordance is incompatible.
Insertion	Vector alignment	The active insertion direction is tilted relative to the passive insertion direction.	Alignment error exceeds tolerance.
Insertion	Clearance	The active object is larger than the opening or socket.	Cross-section plus clearance margin exceeds passive opening size.
Insertion	Trajectory collision	The active object collides with the rim, socket wall, or surrounding geometry.	Collision or penetration exceeds tolerance before reaching required depth.
Insertion	Insertion depth	The object enters only shallowly or slips out immediately.	Insertion depth or post-insertion stability is below threshold.
Hanging	Support compatibility	The active loop or handle cannot engage the passive support.	Loop size, hook size, or support type is incompatible.
Hanging	Vector or keypoint matching	Rigid loop direction does not align with hook direction, or deformable keypoint misses the hook.	Alignment error or keypoint-to-support distance exceeds tolerance.
Hanging	Gravity stability	Object slides off, rotates away, or falls after release.	Post-release support is not maintained under gravity.
Hanging	Clearance	Object collides with hook, wall, rail, or nearby geometry during approach.	Collision or insufficient clearance is detected.
Deformable hanging	Entanglement or overstretching	Garment twists around the hook, rail, gripper, or arm.	Cloth state violates deformation, support, or entanglement criteria.

slightly inward from boundaries, and targets are clamped to each arm's side of the garment midline to reduce hand-hand collision.

For pants, k_{top} and k_{bottom} are the `top_point` and `bottom_point` centerline keypoints. The centerline $\ell_{\text{mid}} = \ell(k_{\text{top}}, k_{\text{bottom}})$ guides a side-to-center fold, followed by a bottom-to-waist fold and optional additional compacting fold. Targets adapt to leg length, waistband width, and bimanual reachability.

Flinging selects paired sleeve, bottom-corner, or shoulder keypoints. Left/right assignment follows world-frame lateral position. The synchronized sequence reaches, closes, lifts, moves forward, drops, and releases; lift height, forward displacement, drop distance, and drop height are stored as adjustable parameters.

Bimanual representation and validation. Each waypoint stores paired end-effector poses and synchronized gripper commands,

$$T_t = (T_t^{\text{R}}, T_t^{\text{L}}), \qquad u_t^{\text{grip}} = (u_t^{\text{R}}, u_t^{\text{L}}),$$

$$\tau^{(i)} = [(T_1, u_1^{\text{grip}}), \ldots, (T_T, u_T^{\text{grip}})].$$

Trajectory distances are adapted from keypoint geometry, including sleeve length, hem-to-shoulder distance, and selected-pair separation. Full-manipulator validation is used when arm feasibility matters; trajectories are rejected for IK failure or singularity, large joint jumps, arm–cloth or table collision, entanglement, or failure to reach the intended geometric outcome. Metadata records IK, collisions, entanglement, keypoint displacement, completion, and failure reason.

Table 17: Garment keypoints used for deformable-object trajectory annotation. Keypoints are detected in a canonical garment frame and then converted back to the simulation or world frame for trajectory generation.

Garment type	Keypoints	Usage
Upper-body garment / T-shirt	`top_left`, `top_right`, `left_shoulder`, `right_shoulder`, `bottom_left`, `bottom_right`, `top_point`, `bottom_point`, `middle_point`	Sleeve folding, bottom-to-shoulder folding, fling from sleeve or bottom keypoints, bimanual lifting and stretching.
Dress	Same as upper-body garment, with longer body extent	Sleeve folding and bottom-up folding, with fold displacement adapted to garment length.
Pants	`top_left`, `top_right`, `bottom_left`, `bottom_right`, `top_point`, `bottom_point`	Side folding, leg folding, bottom-to-waist folding, compacting fold, bimanual lifting.
Plane / rectangular cloth	`corner_0`, `corner_1`, `corner_2`, `corner_3`	Corner grasping, folding, stretching, placement, and retrieval-style deformable manipulation.

Table 18: Garment and deformable-object trajectory templates. Each template is generated from semantic keypoints and then validated with full-manipulator simulation when arm reachability or cloth entanglement is important.

Skill	Keypoints / anchors	Geometric rule	Generated trajectory
Sleeve fold	Sleeve endpoints, shoulders, bottom corners	Use the shoulder–bottom line as the fold line and reflect each sleeve endpoint inward.	Pick sleeve endpoints, lift, move to reflected targets, drop, release, and retract.
Bottom-to-shoulder fold	Bottom corners and shoulder points	Move bottom corners toward the corresponding shoulder region.	Pick bottom corners, lift, fold upward toward shoulders, drop, release, and retract.
Pants side fold	Left/right leg or side keypoints and centerline	Fold side or leg regions toward the centerline.	Pick side or cuff keypoints, lift, move inward, and place to overlap the two sides.
Pants compact fold	Bottom/cuff keypoints and waistband region	Fold the lower part upward toward the waistband; optionally repeat.	Pick bottom/cuff points, lift, fold upward, place, and optionally perform another compacting fold.
Fling	A paired keypoint group, e.g., sleeves, bottom corners, or shoulders	Lift the two selected keypoints and move them forward with bimanual coordination.	Reach, close grippers, lift, fling forward, drop, and release.
Plane / cloth fold	Four corner keypoints	Use selected corners or edges to define fold lines and place targets.	Pick corners, lift, fold across a selected axis, place, and release.
Retrieval / washing / pushing	Keypoints, edges, or local cloth regions	Use keypoint or region targets together with full-arm motion constraints.	Generate task-specific push, sweep, pull, or retrieval trajectories and validate with the full manipulator.

Navigation target and approach-pose annotation. Navigation candidates are interaction-ready base poses in traversable free space,

$$x_{\text{nav}}^{(i)} = q_{\text{base}}^{(i)} = (x^{(i)}, y^{(i)}, \psi^{(i)}),$$

stored in $\mathcal{B}_{h_{\text{scene}}, s_{\text{nav}}}$ for scene-level anchor h_{scene}.

Table 19: Representative waypoint phases for garment skills. The same candidate-bank schema stores both the keypoint anchors and the generated bimanual trajectory.

Skill	Phase order	Description
Upper-body fold	Pre-reach, reach, close, lift, move, drop, open, retract for sleeve; then repeat for bottom	First folds sleeves inward using reflected targets, then folds bottom corners toward shoulder targets.
Fling	Reach, close, lift, fling forward, drop, open	Grasps a pair of keypoints, lifts the cloth, moves forward to spread it, drops, and releases.
Pants fold	Reach, close, lift, side-fold move, drop, release; then bottom-fold move and release	First aligns legs or sides, then folds the lower part toward the waistband.
Plane / cloth fold	Reach, close, lift, fold across selected axis, drop, release	Uses corner keypoints to define a fold axis and target corner positions.

Object-centric target construction. For target object center O, orientation R, and current base position B, define

$$\mathbf{u} = \frac{O - B}{\|O - B\|_2}, \qquad b_\ell = R^\top (B - O), \qquad \mathbf{u}_\ell = R^\top \mathbf{u}.$$

We intersect this local ray with the object's local AABB, obtain the entered-face center f_ℓ and outward normal n_ℓ, and transform them back:

$$f_w = O + R f_\ell, \qquad n_w = R n_\ell.$$

The goal and yaw are

$$g = f_w + \rho n_w, \qquad \psi = \mathrm{atan2}((-n_w)_y, (-n_w)_x),$$

with a face- or object-facing fallback when the horizontal normal is nearly vertical. If the local box or ray intersection is unavailable,

$$g_{\text{fallback}} = O - (r_{\text{aabb}}(\mathbf{u}) + \rho)\mathbf{u},$$

where $r_{\text{aabb}}(\mathbf{u})$ is the world-AABB extent of the object along $\mathbf{u}$. Diversity is obtained by varying target object or part, approach side, standoff ρ, and yaw offset. Each candidate stores the approach face and normal, object-relative parameters, optional coarse path, validation, and diversity metadata.

Planning and validation. Targets must lie in reachable free space with sufficient clearance and path connectivity. Invalid candidates are rejected or, when close, projected to a nearby navigable point. A* or another global planner provides a coarse path; DWB-style local refinement can enforce embodiment dynamics. Final poses must also support visibility and manipulation reachability.

During execution,

$$\mathrm{NavTo}(\text{robot_id}, q_{\text{base}})$$

succeeds when

$$\|p_{\text{base}} - p_{\text{target}}\|_2 < \epsilon_{\text{pos}}, \qquad |\Delta\psi| < \epsilon_{\text{yaw}},$$

where p_{base} is the achieved base position, $p_{\text{target}} = (x, y)$ is the positional component of q_{base}, and $\Delta\psi$ is the wrapped yaw error. Metadata records completion, timeout, planner failure, reset invalidation, and object-relative face information so targets can be recomputed after object motion.

Navigation extends the schema from object contacts to interaction-ready scene poses while retaining geometric grounding, embodiment-aware validation, and candidate diversity.

G Additional Details of Large-scale Robot Data Collection

Collection system. Our large-scale data-collection system treats AnnotateAnything annotations as executable task interfaces rather than object-specific scripted demonstrations. An annotation-aligned skill library covers tabletop and bimanual manipulation, whole-body and humanoid control,

Table 20: Navigation target and approach-pose annotation. The navigation primitive generates interaction-ready base poses around objects or parts and validates them on the navigation mesh.

Component	Annotation	Description
Target object or part	$o(h)$	The object or part that the robot should approach, such as a cabinet, drawer, table, appliance, or object of interest.
Scene anchor	h_{scene}	A free-space or object-centric approach region around the target. This region defines where valid base poses may be sampled.
Functional affordance	$\phi(h_{\text{scene}})$	The scene-level affordance, such as navigation approach, interaction-ready base pose, or approach-for-manipulation.
Entry face center	f_w	The center of the object face approached by the robot, computed from ray-AABB intersection in the object-local frame.
Outward normal	n_w	The world-frame outward normal of the entry face. The base is placed outside this face and oriented toward $-n_w$.
Base target	$q_{\text{base}} = (x, y, \psi)$	The final navigation target, including base position and yaw.
Standoff margin	ρ	The distance between the object face and the target base position. Larger margins create safer but farther approach poses.
Coarse path	τ_{nav}	An optional A*-style path or waypoint sequence from the current base pose to the target base pose.
Validation metadata	v_{nav}	Reachability flag, nav-mesh validity, path connectivity, clearance, final pose error, and planner status.

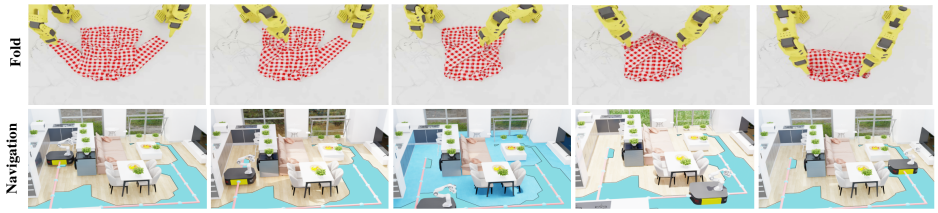


Figure 16: Qualitative closed-loop rollouts (not part of the quantitative evaluation) of policies trained from annotation-enabled simulation data: deformable-object folding on a real garment (top) and navigation in a furnished indoor scene (bottom).

dexterous-hand manipulation, and mobile manipulation. Each skill maps compact fields—including grasp poses, target parts, waypoints, insertion directions, hanging anchors, and navigation goals—to controller or planner goals. Long-horizon tasks are formed by composing these skills according to task templates and object-state transitions.

Rollouts are generated in asynchronous parallel simulation environments. Each worker independently samples assets, tasks, robot embodiments, object poses, obstacles, and scene layouts, with domain randomization over object pose, robot–object configuration, camera viewpoint, lighting, materials, textures, distractor objects, and scene layout. We use cuRobo-v2 for goal-set IK, motion planning, and obstacle-aware execution; it provides GPU-native collision-aware motion generation for standard manipulators and high-DoF embodiments (Sundaralingam et al., 2026). Because randomization changes the relative robot–object pose, a worker retrieves ranked candidates from the relevant annotation bank. For each candidate, it solves goal-set IK using the complete task-compatible robot

Table 21: Common validation checks and failure cases for navigation target annotation.

Check	Failure case	Rejection or repair rule
Nav-mesh validity	Generated target lies outside the navigable mesh or inside occupied space.	Reject the candidate or project it to the nearest valid navigable point if the projection remains close to the original target.
Clearance	The base pose is too close to the object, wall, furniture, or obstacle.	Reject poses with clearance below a safety margin.
Path connectivity	The target is valid locally but disconnected from the robot's current reachable region.	Reject if A* or the global planner cannot connect the current base pose to the target.
Yaw feasibility	The target position is valid, but the desired yaw causes poor visibility or poor manipulation reachability.	Perturb yaw or reject if the final orientation cannot face the target object or part.
Embodiment dynamics	The A* path contains turns or passages infeasible for the robot base.	Pass the path to local optimization or DWB-style rollout; reject if local rollout fails.
Manipulation reachability	The robot can navigate to the base pose but cannot reach the target part from there.	Reject if the manipulation anchor is outside the arm workspace or fails IK / reachability checks.
Stale target	The object moves or the environment resets after the target is computed.	Clear the cached target and recompute the object-centric approach pose.

embodiment and selects the feasible IK solution with the lowest planning cost. If the highest-ranked candidate is infeasible under full-robot IK or planning constraints, the worker evaluates subsequent stored candidates until a feasible solution is found or the candidate set is exhausted. Rollouts are checked for task success, collision safety, contact consistency, articulation progress, and final-state stability when applicable. Candidates that repeatedly fail under randomized conditions are removed from the bank. This system demonstrates annotation executability and scalable data generation; a complete system-level analysis is beyond the scope of this paper.

G.1 Keypoint Generation and Affordance Grounding

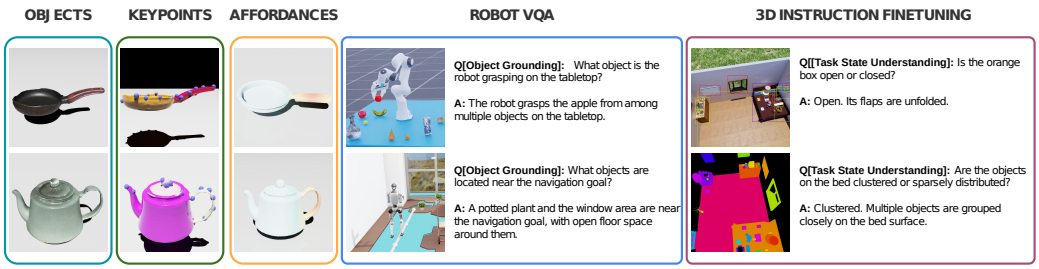


Figure 17: Downstream uses of the annotations. Object, keypoint, and affordance labels support robot-centric VQA for object grounding and navigation reasoning, and 3D instruction tuning for articulation-state and object-distribution reasoning.

Keypoints and executable affordances. AnnotateAnything provides two related but distinct perception labels. Keypoints are sparse semantic or functional anchors produced by the visual annotation stage, whereas affordances describe where and how an interaction can be executed. As described in Sec. 3.1, keypoint generation uses fused 3D observations reconstructed from rendered RGB-D views, downsamples it by farthest-point sampling, obtains candidate keypoints from Skeleton

Merger (Shi et al., 2021), and uses a VLM to select meaningful locations such as handles, openings, corners, garment landmarks, or scene-level targets. A keypoint identifies a useful region but need not itself define an executable action.

Affordance supervision is derived from the validated candidate banks produced by the physics-based action pipeline in Sec. 3.2. Each positive label can include an anchor, skill, action direction or frame, contact configuration, trajectory, and validation metadata. Thus, it represents an interaction verified under the corresponding simulation, embodiment, and validation settings rather than only a visually plausible region. Multiple successful candidates are retained to preserve a multimodal label distribution.

Representations across asset types. For rigid objects, validated parallel-jaw or dexterous grasps provide grasp centers, approach directions, orientations, widths, palm poses, finger contacts, and contact modes. For articulated objects, labels pair a contact anchor with a task-conditioned motion trajectory, distinguishing actions such as pulling, pushing, rotating, and holding. Garment and deformable-object affordances are relational: pick points are stored with paired place points, second-hand grasp points, fold lines, or manipulation trajectories. Room-scale labels use occupancy or BEV maps to represent navigable regions, object-centric approach poses, interaction-ready base poses, and feasible navigation targets, connecting navigation to local manipulation.

Perception supervision. For image-based models, 3D anchors, contact regions, and trajectories can be projected into rendered RGB-D views to form keypoint targets, affordance masks, heatmaps, or directional fields. For point-cloud models, the labels can be attached to 3D points or local patches; for navigation models, room-level labels can be rasterized into BEV supervision for traversability, target poses, and interaction readiness. Successful candidates provide positive labels. Failed candidates can serve as hard negatives when failure reflects physical infeasibility, such as collision, unstable contact, unreachable geometry, or invalid motion, but we use them conservatively because some failures depend on the selected embodiment, planner, or randomized scene rather than the object's intrinsic affordance. This construction provides reusable keypoint and affordance supervision. We quantitatively evaluate the generated annotations within established part, affordance, and keypoint pipelines below.

G.2 Utility as Supervision for Established Pipelines

We evaluate whether generated annotations can replace the supervision, inputs, or reference annotations of established downstream pipelines: SAMPart3D (Yang et al., 2024) on a matched PartNeXt subset (Wang et al., 2026b), LASO (Li et al., 2024a) on its matched evaluation subset, and Skeleton Merger (Shi et al., 2021) on the Airplane and Chair categories of KeypointNet (You et al., 2020). For each experiment, the downstream pipeline and evaluation protocol are kept fixed; only the original supervision, input, or reference annotations are replaced by AnnotateAnything outputs. Specifically, for SAMPart3D, our part regions are projected into the same multi-view renderings and replace the SAM-generated 2D masks that supervise its scale-conditioned grouping features, with the rest of the pipeline unchanged; for LASO, our affordance regions replace the human-annotated training labels, and evaluation uses the original LASO test annotations; and for Skeleton Merger, which is trained without keypoint labels, our keypoints replace the KeypointNet human annotations used as the reference in its DAS evaluation. For this experiment, keypoint candidates are generated by a Skeleton Merger model trained on categories other than Airplane and Chair, whereas the evaluated Skeleton Merger follows the original setting (training on ShapeNetCore V2); the keypoint result therefore measures agreement between our keypoints and the human reference keypoints under the DAS protocol, rather than utility as training supervision. This evaluation is distinct from the direct annotation-accuracy evaluation in Sec. 5.2: here, the reported metric is the downstream pipeline's final performance. In Table 22, the SAMPart3D and LASO reference values come from matched re-runs with original supervision. The Skeleton Merger reference values are the published results for the corresponding setting: training on ShapeNetCore V2 and evaluation on the Airplane and Chair categories of KeypointNet.

Across part segmentation, affordance grounding, and keypoint transfer, AnnotateAnything annotations yield downstream performance with modest gaps to the corresponding reference supervision. These

Table 22: Using AnnotateAnything annotations as supervision or input for established downstream pipelines. "Reference" denotes our matched re-run with the pipeline's original supervision on the same subset and metric for SAMPart3D and LASO, and the published result for Skeleton Merger.

Annotation	Pipeline	Setting	Metric	Reference	Ours
Part	SAMPart3D	247 PartNeXt objects	mIoU	46.70	43.9
Affordance	LASO	200 LASO objects	mIoU/F1	43.6/57.2	40.7/54.1
Keypoint	Skeleton Merger	KeypointNet Airplane/Chair	DAS	77.7/76.8	74.5/73.6

experiments demonstrate downstream usability with modest gaps to reference supervision, rather than parity.

G.3 Robot Reasoning VQA

As an illustrative repurposing, simulation rollouts can be converted into robot-reasoning VQA using execution-level metadata, including the selected object, target part, anchor point, skill type, goal-set IK solution, approach pose, navigation target, articulation direction, garment manipulation landmark, and task-success state. These labels are determined at runtime because domain randomization can change which candidate is feasible even for the same asset.

We convert these records into VQA pairs using VLM-assisted language generation. The answers are derived from simulator records and selected candidates; the VLM only diversifies and naturalizes question wording. Questions cover object and part selection, approach direction, navigation progress, articulation motion, garment landmarks, and execution success. Because they concern executed candidates and task progress, these questions require trajectory-level traces rather than static asset annotations alone. This lightweight task demonstrates trajectory-grounded reasoning supervision rather than a full VQA benchmark.

G.4 3D VLM Instruction-tuning Data

As another illustrative repurposing, AnnotateAnything supports instruction-tuning data from two complementary sources. Simulation provides 3D and projected 2D bounding boxes, instance masks, object poses, part identities, and object–instance associations, which yield grounding instructions for localization, instance identification, part reference, and 2D–3D association. Language annotations and the cross-level composition in Sec. 3.1 provide spatial-reasoning supervision. Because object-level annotations are linked to room layouts through instance identities and 6D poses, they support questions about object relations, room structure and layout, containment, support, proximity, and object-to-room associations.

Structured templates ground answers in simulator or annotation metadata, while VLM-assisted paraphrasing increases linguistic diversity without supplying ground truth. The resulting data illustrates how the same annotated assets can support 3D visual grounding and spatial reasoning. Full-scale 3D VLM training and evaluation remain outside the scope of this paper.

H Additional Experimental Details

This section supplements Secs. 5.1–5.6 with evaluation scopes, metrics, disaggregated results, validation criteria, rollout protocols, baselines, human evaluation, and timing definitions. Pipeline details appear in Secs. C, D, E, and F; the rollout system is described in Sec. G.

H.1 Evaluation Suites and Manual References

We use three evaluation scopes. The **processed asset pool** supports source-coverage and scale statistics. The **audited suite** supports action-quality, rollout, human-evaluation, and reported per-skill throughput metrics. The suite was sampled before any annotation outcomes were computed and was manually checked only for asset validity, simulator compatibility, and independently defined skill

applicability; no success- or failure-based filtering was applied. Metrics are computed over asset–skill pairs whose applicability is defined independently of annotation outcomes. The **manually annotated subset** is a separate balanced set containing reference language, parts, affordances, keypoints, and scene cues for visual-language and type-specific visual evaluation. Real-world trials are reported separately and are excluded from audited-suite averages. Because AnnotateAnything is not trained on the processed asset pool or audited suite, a conventional train/test split is not applicable to annotation generation; downstream learning follows its own training and evaluation protocol.

Table 23: Audited evaluation suite. Simulation-based action-quality, rollout, and action/trajectory human-evaluation metrics in the main paper are computed over valid asset–skill pairs from this suite unless otherwise specified.

Asset Type	#Assets	#Valid Asset–Skill Pairs	Evaluated Skills
Rigid objects	480	1,700	Grasp, DexGrasp, BiGrasp, Insertion, Hanging
Articulated objects	320	740	Grasp, Articulation, Navigation / Approach
Deformable / garments	280	760	BiGrasp, BiDexGrasp, Deformable, Hanging
Room-scale scenes	80	498	Navigation / Approach, Mobile Manipulation
Total	1,160	3,698	–

Table 23 defines the scale of the action-quality and rollout evaluation. An asset–skill pair is an asset together with one skill family and interaction target; an asset contributes several pairs when it has several targets, for example one pair per joint of an articulated object or per target object in a room, so pair counts can exceed asset counts. Mobile manipulation composes Navigation/Approach with manipulation skills such as grasping, and its pairs are counted within the nine reported skill families. The visual-language study uses a separate, balanced subset so that each asset type contributes equally to the manual-reference analysis.

Table 24: Manually annotated subset used for visual-language and visual annotation evaluation. Reference annotations include language descriptions, functional parts, affordance regions, keypoints, and scene-level spatial cues when applicable.

Asset Type	#Examples	Reference Annotation Types
Rigid objects	90	Language, parts, keypoints, affordances
Articulated objects	90	Language, parts, keypoints, affordances, articulation cues
Deformable / garments	90	Language, garment keypoints, parts, affordances
Room-scale scenes	90	Scene language, object layout, spatial cues, approach regions
Total	360	–

The manually annotated subset contains 90 examples from each of the four asset families and is used only for visual-language and visual-annotation evaluation. Object references are derived from PartNet, PartNeXt, KeypointNet, and LASO; garment and room references, which these datasets do not cover, were manually annotated by the authors and by professional annotators (an industry partner and a commercial annotation vendor). It is not used to select, filter, or tune the audited action-quality suite.

Table 25 summarizes the population and denominator of each quantity reported in the main paper.

H.2 Annotation Coverage and Conversion Statistics

The statistics underlying Fig. 6 characterize annotation conversion rather than a curated asset release. The processed pool contains 17,005 assets from 9 source families. The reported 100M total includes original candidates and validated perturbation- and symmetry-derived variants. Failed raw proposals are excluded, and every augmented variant is counted only after passing the corresponding geometry and physics validation checks. The total therefore measures validated annotation instances rather than 100M distinct manipulation strategies.

Table 27 summarizes how raw candidates are filtered through geometry, trajectory, and physics checks before entering the annotation bank; augmentation is applied afterwards.

On the audited suite, each attempted pair produces on average 2,315 raw candidates; 1,383 pass geometry checks, 842 pass IK or trajectory checks, 615 remain stable under the physics disturbance

Table 25: Populations and denominators of the reported quantities.

Quantity	Population	Definition and denominator	Reported in
Pass	Audited suite (DexGrasp: 10 candidates per pair)	Fraction of a pair's generated candidates that pass physics validation, averaged over pairs and then over skill families	Table 3
Ready	Audited suite	Fraction of attempted pairs with at least one retained annotation	Table 3
Exec.	Ready pairs of the audited suite	Full-robot success of ranked retrieval with one fallback; the denominator is the ready pairs	Table 3
Ready$\times$Exec.	Audited suite	Product of the aggregate Ready and Exec.; unready pairs count as failures	Sec. 5.3
Ann./min	Audited suite	Retained annotations per wall-clock minute on one RTX 4090, from generation to serialization	Table 3
Data Succ., Att./Succ.	Rollout evaluation	Successful rollouts divided by attempts; Att./Succ. $= 100/$Data Succ.	Tables 3–4
Traj./h	Rollout evaluation	Successful trajectories per wall-clock hour, from candidate retrieval to trajectory validation; excludes bank construction and candidate validation	Tables 3–4
Human scores	Manual subset (360 examples); 120 pairs per skill family and method	Mean over items of the five-rater average; $\pm$ is the standard error over items	Tables 2–4
Pooled validation ratio	Processed pool	Validated divided by generated candidates, pooled over all candidates of a skill family	Fig. 6B
Conversion funnel	Audited suite	Mean counts per attempted pair before augmentation; ratios pool candidates across pairs	Fig. 6C, Table 27
Validated annotations (100M)	Processed pool	Validated instances, including re-validated perturbation- and symmetry-derived variants	Sec. 5.5
Real-world success	Physical trials	Successes out of 20 independently reset trials per task; Wilson 95% intervals	Fig. 8, Table 38

Table 26: Processed asset pool used for annotation generation. Source families are grouped coarsely because the goal is annotation conversion rather than curated asset-dataset release. #Ann./Obj./(Skill, EE) averages validated annotations over the skill–end-effector combinations run for each asset. Because different end-effectors yield different numbers of annotations and not every combination is run on every asset, the columns do not multiply to the total; the total of about 100M is reported in Sec. 5.5.

Asset Type	#Assets	#Ann./Obj./(Skill, EE)	Main Applicable Skills	End-effectors
Rigid objects	12,538	869	Grasp, DexGrasp, BiGrasp, Insertion, Hanging	Franka, FrankaUMI, X-Trainer, and PiPER-X grippers; Sharpa, Dex3-1, and XHand hands
Articulated objects	2,094	304	Articulation, Grasp, Navigation / Approach	FrankaUMI gripper and Dex3-1 hand for articulation; grasp end-effectors as for rigid objects
Deformable / garments	2,167	83	BiGrasp, BiDexGrasp, Deformable, Hanging	Mainly FrankaUMI grippers
Room-scale scenes	206	527	Navigation / Approach, Mobile Manipulation	Mobile base
Total	17,005	–	–	–

tests, and 538 are valid and enter the bank. Validity additionally requires the skill's execution check to succeed, and a disturbance-stable candidate may still fail it, for example when the grasp shifts on the object during execution. Perturbation- and symmetry-derived variants are generated from valid candidates after this stage and count toward the 100M total only after passing the rechecks of App. E.6 (geometry and short-rollout physics checks for pose-level variants, and full physics validation for full-manipulator deformable trajectories). The corresponding ratios are reported below.

Table 27: Stage-wise candidate-to-annotation conversion on the audited suite, before augmentation. Counts are averaged per attempted asset–skill pair, and ratios pool candidates across all pairs. Pooled ratios weight pairs by their candidate counts and therefore differ from the per-pair averaged Pass in Table 28.

Statistic	Generated	Geometry Feasible	IK / Traj. Feasible	Disturbance Stable	Valid (Bank)
Avg. count / attempted pair	2,315	1,383	842	615	538
Retained ratio	100.0%	59.7%	36.4%	26.6%	23.2%

Fig. 6B reports pooled ratios over all candidates generated for the processed pool, whereas Pass in Table 3 averages per-pair pass rates over the audited suite (for DexGrasp, under the 10-candidate audit). Pooled ratios are dominated by assets and pairs with many candidates, so the two statistics can order skill families differently (e.g., BiDexGrasp and Navigation).

Table 28 reports pair counts, physics pass rate (Pass), asset–skill readiness (Ready), and retained annotations per wall-clock minute (Ann./min). Macro averages weight skill families equally.

Table 28: Atomic-skill annotation statistics on the audited suite. DexGrasp Pass/Ready (and its Exec. in Table 3) are those of the protocol-matched audit in App. H.4, which uses 10 candidates per pair.

Skill Family	#Asset–Skill Pairs	Pass	Ready	Ann./min
Grasp	760	60.3	96.4	800
DexGrasp	520	43.8	88.7	200
BiGrasp	430	38.5	90.5	105
BiDexGrasp	320	27.4	82.4	95
Articulation	390	41.6	89.6	110
Insertion	290	33.2	84.1	70
Hanging	310	31.5	81.7	60
Deformable	260	30.1	78.9	45
Navigation / Approach	418	58.7	94.2	300
Macro Avg.	–	40.6	87.4	198

Candidate-bank diversity. Raw candidate counts include dense local variations and therefore do not directly measure distinct manipulation modes. We cluster candidates within each asset–skill pair using a 2 cm translational and 10° rotational threshold for wrist poses. DexGrasp additionally uses a 0.15 rad finger-joint threshold. Articulation clustering additionally distinguishes the interacted part, motion direction, trajectory family, and waypoint geometry. The evaluated asset sets were sampled independently from the processed asset pool and are separate from the audited action-quality suite; their selection did not use annotation outcomes.

Table 29: Candidate-bank diversity after collapsing local variants. Values report median [IQR] over independently sampled assets.

Skill	Assets	Candidates / asset	Clusters / asset	Cluster ratio
Grasp	4,185	851 [560, 1,113]	176 [118, 239]	0.230 [0.142, 0.352]
DexGrasp	400	210 [158, 257]	153 [96, 193]	0.714 [0.501, 0.862]
Articulation	300	162 [63, 528]	21 [10, 42]	0.119 [0.084, 0.171]

The median inter-cluster wrist-pose separations are 20.9 cm/125.6° for Grasp, 21.4 cm/132.9° for DexGrasp, and 13.8 cm/78.6° for Articulation. The median normalized inter-cluster trajectory distance for Articulation is 0.276. These statistics quantify pose and trajectory separation beyond local perturbations. Pose-space clustering does not merge symmetry-equivalent candidates, so the cluster counts should not be interpreted as symmetry-reduced strategy counts.

H.3 Visual-Language and Visual Annotation Evaluation

Evaluation uses the balanced manual subset in Table 24. Complete bundles contain language, keypoints, part or affordance regions, and scene cues when applicable. We compare **Direct VLM**, which predicts the bundle directly from multi-view images; **Aff.-only + heur.**, which uses visual affordance anchors and geometric heuristics to construct regions and anchors; **w/o 3D refinement**, which removes point-cloud consistency and part-region refinement; and the complete method. All variants are converted to the same bundle and type-specific output schemas before scoring. These comparisons vary the grounding pipeline and do not establish robustness across VLM backbones. Five raters per example score semantic correctness, 3D grounding, coverage, and actionability from 0 to 100. The overall bundle score averages these four dimensions, and reported uncertainty is the standard error over example-level scores after averaging across raters.

Table 30: Human evaluation rubric for complete visual-language annotation bundles. All dimensions are scored from 0 to 100.

Dimension	Description
Semantic correctness	Correctness of object, part, function, affordance, and interaction-constraint descriptions.
3D grounding accuracy	Whether keypoints, part regions, and affordance anchors lie on the intended 3D regions.
Coverage	Whether major interaction-relevant parts and regions are included.
Actionability	Whether the annotation provides useful anchors for physics-grounded action generation.

For type-specific evaluation, let $\mathcal{R}$ and $\mathcal{P}$ be reference and predicted regions and let $\ell(\cdot)$ denote a compatible part or affordance label. We use bidirectional label-compatible IoU,

$$S_{\mathrm{reg}}(\mathcal{P},\mathcal{R}) = 50\left(\frac{1}{|\mathcal{R}|}\sum_{r\in\mathcal{R}}\max_{p\in\mathcal{P}:\ell(p)\sim\ell(r)}\mathrm{IoU}(p,r) + \frac{1}{|\mathcal{P}|}\sum_{p\in\mathcal{P}}\max_{r\in\mathcal{R}:\ell(p)\sim\ell(r)}\mathrm{IoU}(p,r)\right),$$

and normalized keypoint correctness,

$$S_{\mathrm{kp}} = 100\frac{1}{|\mathcal{K}^\star|}\sum_{k^\star\in\mathcal{K}^\star}\mathbf{1}\left[\min_{k:\ell(k)\sim\ell(k^\star)}\frac{\|k-k^\star\|_2}{d_{\mathcal{A}}} < 0.05\right].$$

Here $d_{\mathcal{A}}$ is the object bounding-box diagonal, or the local room scale for scene anchors. The region metric is applied separately with functional-part and affordance label spaces. These aggregate matching scores, unlike bundle ratings, are not subjective human scores.

Table 31: Type-specific visual annotation metrics used for the right block of Table 2. All scores are normalized to a 0–100 scale.

Annotation Type	Metric Definition
Part	Label-compatible bidirectional IoU between predicted and reference functional part regions.
Affordance	Label-compatible bidirectional IoU between predicted and reference affordance regions.
Keypoint	Normalized keypoint correctness under a label-compatible distance threshold.

Using these definitions, we next report complete bundle quality by asset type before separating the automatic visual-annotation scores.

Table 32: Bundle-level visual-language annotation quality by asset type. Scores are human ratings on a 0–100 scale.

Asset Type	Method	Semantic	3D Ground.	Coverage	Action.	Overall
Rigid	Direct VLM	90.0	68.8	76.5	70.8	76.5
Rigid	Aff.-only + heur.	83.0	77.8	80.9	75.1	79.2
Rigid	w/o 3D refine	92.4	81.7	84.6	82.9	85.4
Rigid	Ours	97.3	95.0	95.3	96.1	95.9
Articulated	Direct VLM	87.3	64.6	71.5	67.4	72.7
Articulated	Aff.-only + heur.	75.7	69.3	71.3	66.2	70.6
Articulated	w/o 3D refine	87.9	73.6	78.4	75.2	78.7
Articulated	Ours	94.2	90.0	90.6	92.2	91.7
Deformable / garments	Direct VLM	84.4	63.1	69.7	65.6	70.7
Deformable / garments	Aff.-only + heur.	75.5	69.9	73.2	66.5	71.3
Deformable / garments	w/o 3D refine	88.6	74.8	79.2	76.8	79.8
Deformable / garments	Ours	93.2	89.7	91.0	91.5	91.3
Room-scale scenes	Direct VLM	78.6	53.4	60.8	57.1	62.5
Room-scale scenes	Aff.-only + heur.	68.7	62.1	64.3	59.1	63.6
Room-scale scenes	w/o 3D refine	81.1	64.6	70.1	66.8	70.6
Room-scale scenes	Ours	90.2	86.0	86.3	87.8	87.6

The asset-type breakdown tests whether complete annotation bundles remain reliable across rigid, articulated, deformable, and room-scale data. We separately report the automatic part, affordance, and keypoint scores to distinguish overall bundle quality from type-specific geometric matching.

Grounding-interface reliability. We additionally evaluate each generated interface directly against manual references. Of 1,080 evaluated part regions, 944 (87.4%) satisfy label compatibility and

Table 33: Type-specific visual annotation scores by method. These scores correspond to the right block of Table 2.

Method	Part	Affordance	Keypoint
Direct VLM	72.8	68.5	70.2
Aff.-only + heur.	70.5	79.3	68.8
w/o 3D refine	82.6	78.1	80.8
Ours	**91.5**	**89.0**	**90.3**

IoU ≥ 0.5. Of 810 affordance regions, 687 (84.8%) satisfy the corresponding criterion. Generated keypoints obtain 90.3% PCK@0.05, where the threshold is normalized by the object bounding-box diagonal. For compatible-skill prediction on 40 manually labeled anchors per reported skill family, 300/360 anchors (83.3%) have exactly correct skill sets, with macro precision/recall/F1 of 88.4/84.6/86.5. Table 34 breaks these counts down by asset type.

Table 34: Grounding-interface reliability by asset type on the manual-reference subset (90 examples per type). A region matches when it is label-compatible with a reference region at IoU ≥ 0.5; a keypoint is correct when its distance to the reference is below 0.05 of the bounding-box diagonal (local room scale for scenes). Room-scale scenes have no part or affordance references.

Asset type	Part regions matched	Affordance regions matched	Keypoints correct
Rigid	454/528 (86.0%)	267/322 (82.9%)	1,091/1,113 (98.0%)
Articulated	392/429 (91.4%)	291/326 (89.3%)	344/381 (90.3%)
Deformable / garments	98/123 (79.7%)	129/162 (79.6%)	199/263 (75.7%)
Room-scale	–	–	571/685 (83.4%)
Total	944/1,080 (87.4%)	687/810 (84.8%)	2,205/2,442 (90.3%)

Matched external annotation evaluation. On a matched 247-object PartNeXt subset Wang et al. (2026b), our part annotations obtain 44.1 object-macro mIoU, compared with 46.70 for SAMPart3D Yang et al. (2024) and 41.01 for P3-SAM Ma et al. (2025). On a matched 200-object LASO subset Li et al. (2024a), our affordance annotations obtain 41.0/54.3 mIoU/F1, compared with 43.6/57.2 for the released LASO baseline and 35.8/48.9 for Direct VLM. All methods are evaluated on identical subsets and metrics in these matched re-runs; these values should not be interpreted as direct comparisons with numbers reported under different original-paper protocols. They are also lower than the manual-reference scores above because the protocols differ: the manual references are matched with label-compatible, bidirectional IoU on task-relevant functional parts, which is more lenient, whereas object-macro mIoU scores all annotated PartNeXt parts.

Detectable failures and semantic filtering. Across 3,240 asset- and part-level VLM queries, 38 fail structured parsing and 56 cannot be linked to a valid segmentation identifier. All 94/3,240 detectable failures (2.9%) are rejected before action generation. Uncertain attributes may be represented as `unknown`, and anchors with unresolved interaction-defining attributes are not used for skill instantiation. These checks reject malformed or ungrounded outputs but do not identify every possible semantic error. In a separate 18-object audit, skill-specific geometric and physics validation increases candidate-level intended-part selectivity from 50.0% to 70.0%, with improvements on 16/18 objects.

H.4 Physics-grounded Action Annotation Evaluation

For valid pair $(\mathcal{A}, s)$, let $\mathcal{C}_{\mathcal{A},s}$ be generated candidates, $\mathcal{P}_{\mathcal{A},s}$ physics-valid candidates, and $\mathcal{V}_{\mathcal{A},s}$ retained annotations. The physics pass rate averages per-pair pass rates within a skill family, readiness is the fraction of pairs with at least one retained annotation, and annotation density is the number of retained annotations per ready pair:

$$R_{\text{pass}}(s) = \frac{1}{N_s} \sum_{\mathcal{A}} \frac{|\mathcal{P}_{\mathcal{A},s}|}{|\mathcal{C}_{\mathcal{A},s}|}, \qquad R_{\text{ready}}(s) = \frac{\sum_{\mathcal{A}} \mathbf{1}[|\mathcal{V}_{\mathcal{A},s}| > 0]}{N_s}.$$

$$D_{\text{accept}}(s) = \frac{\sum_{\mathcal{A}} |\mathcal{V}_{\mathcal{A},s}|}{\sum_{\mathcal{A}} \mathbf{1}[|\mathcal{V}_{\mathcal{A},s}| > 0]},$$

where N_s is the number of attempted asset–skill pairs for s. Execution success is measured by retrieving ranked stored annotations, solving goal-set inverse kinematics and collision-aware planning with the complete task-compatible robot embodiment, and applying the skill-specific success criterion in Sec. H.5. Ann./min includes candidate generation, filtering, optimization, validation, and serialization. Macro averages weight reported skill families equally.

For object-centric skills, physics validation uses a floating end effector as a candidate filter. The reported *Exec.* metric is distinct from floating validation and evaluates goal-set inverse kinematics, collision-aware motion planning, and execution with the complete task-compatible robot embodiment. Stored candidates are evaluated in rank order, with one fallback to the next candidate when the highest-ranked candidate is infeasible under full-robot constraints; Exec. uses this one-fallback protocol for all skill families, whereas data collection may use a larger retry budget for complex multi-step tasks (Sec. 4.1). Grasp, Articulation, Insertion, and Hanging use floating FrankaUMI validation followed by full Franka–FrankaUMI execution; DexGrasp uses a floating Dex3-1 hand followed by G1–Dex3-1 execution; BiGrasp is executed on dual Franka arms with parallel-jaw grippers, and BiDexGrasp on the G1 with two Dex3-1 hands; Navigation/Approach uses base-aware validation followed by execution with LocoManip, our loco-manipulation embodiment (a G1 humanoid in this paper). Garment/deformable skills use full-manipulator validation.

Table 35: Full physics-grounded action annotation results on the audited suite. This table expands the compact metrics reported in Table 3. DexGrasp Pass/Ready/Exec. follow the 10-candidate audit, as in Table 28.

Skill Family	#Pairs	Pass	Ready	Exec.	Ann./min	Human
Grasp	760	60.3	96.4	98.1	800	89.8
DexGrasp	520	43.8	88.7	93.9	200	86.2
BiGrasp	430	38.5	90.5	95.0	105	87.5
BiDexGrasp	320	27.4	82.4	88.5	95	84.1
Articulation	390	41.6	89.6	93.2	110	85.6
Insertion	290	33.2	84.1	89.3	70	82.9
Hanging	310	31.5	81.7	88.7	60	83.5
Deformable	260	30.1	78.9	84.1	45	80.7
Navigation / Approach	418	58.7	94.2	96.5	300	89.3
Macro Avg.	–	40.6	87.4	91.9	198	85.5

Floating FrankaUMI candidates are rechecked on the full Franka–FrankaUMI system; floating Dex3-1 candidates are rechecked on G1–Dex3-1; and navigation or approach candidates are evaluated with the base-aware LocoManip embodiment (a G1 humanoid in this paper). Deformable skills already use full-manipulator validation.

Floating-to-full-robot audit. For DexGrasp, we evaluate 520 audited asset–skill pairs with 10 candidates per pair. Of $5{,}200$ candidates, $2{,}278$ (43.8%) pass floating-hand validation, leaving $461/520$ ready pairs (88.7%). On the complete G1–Dex3-1 embodiment, the first-ranked floating-valid annotation succeeds for $402/461$ pairs (87.2%). Allowing one fallback to another stored candidate increases success to $433/461$ pairs (93.9%). The latter is the annotation-bank execution protocol and corresponds to the DexGrasp Exec. value reported in the main paper; its denominator is the ready-pair set, not all attempted pairs. Accordingly, DexGrasp Pass, Ready, and Exec. in Table 3 follow this 10-candidate protocol, which is shared with the BODex comparison and uses a smaller candidate budget than the other skill families.

H.5 Skill-specific Validation Criteria

Table 36 summarizes the criteria used for the compact results in Table 3. Primitive-specific thresholds are defined in Secs. E and F and are not repeated here.

For object-centric skills, the Physics Validation column describes floating end-effector candidate filtering, while the Execution Success column describes evaluation with the complete task-compatible robot embodiment after inverse kinematics and collision-aware planning. Deformable and other strongly embodiment-dependent skills already use the full manipulator during physics validation.

Table 36: Skill-specific validation and execution-success criteria used for evaluation.

Skill Family	Candidate Representation	Physics Validation	Execution Success
Grasp	6D gripper pose, width, approach direction	Collision-free approach and closing; stable contact; bounded penetration; robustness under disturbance	Object remains supported through the primitive-specific lift-and-stability test
DexGrasp	Hand pose, joint configuration, contact set	Joint limits, self-collision, hand-object penetration, contact stability, and object support	Object remains stably constrained by the hand after closing and perturbation
BiGrasp	Two end-effector poses and coordinated contact targets	Both contacts feasible; no inter-arm collision; object remains supported by the pair	Object is controlled by both end-effectors without dropping or excessive drift
BiDexGrasp	Dual-hand poses, finger configurations, contact sets	Dual-hand joint limits, contact stability, hand-object and inter-hand collision checks	Dual-hand contact remains stable and task-relevant during execution
Articulation	Handle/contact point, motion direction, waypoint trajectory	Contact maintained along trajectory; joint progresses in intended direction; no jamming or invalid collision	Target joint reaches the primitive-specific progress threshold defined in Sec. F
Insertion	Object pose, insertion direction, approach waypoint	Semantic and geometric compatibility; alignment; collision-free approach; no jamming before insertion	Object reaches the primitive-specific insertion depth and alignment tolerance
Hanging	Hanging pose, support/contact point, release trajectory	Support clearance; contact formation; gravity stability; no immediate slip after release	Object remains supported after release under the primitive-specific stability test
Deformable	Garment keypoints, bimanual grasp points, target waypoints	Keypoint contacts feasible; full-arm reachability; no severe collision, overstretching, or entanglement	Target keypoints or shape state satisfy the task-specific deformable-object criterion
Navigation / Approach	Target base pose, approach direction, path waypoints	Traversable region; collision-free path; clearance; manipulation reachability at final pose	Robot reaches an interaction-ready target pose within the primitive-specific pose tolerance

H.6 Annotation-enabled Rollout Evaluation

The rollout system in Sec. G consumes annotations as executable task interfaces. **No annotation / random** samples targets or parameters without stored labels; **VL-only annotation** uses visual-language anchors without physics-validated actions; and **Ours annotation bank** uses the complete bank. Data Succ. is successful rollouts divided by attempts; Traj./h is successful trajectories per wall-clock hour; Att./Succ. is attempts per successful trajectory; and Traj. Human is the 0–100 human trajectory score. In the aggregate rows, Data Succ. and Traj./h are macro-averaged across skill families, Att./Succ. is computed as $100/\text{Data Succ.}$ because Data Succ. is expressed as a percentage, and Traj. Human is averaged over the sampled trajectory-level ratings for each method. For each randomized scene, stored candidates are evaluated in annotation-bank rank order. For each candidate, the feasible inverse-kinematics solution with the lowest planning cost is selected; if no full-robot solution is feasible, the system proceeds to the next stored candidate.

H.7 Real-World Transfer Protocol

This section details the real-world evaluation in Sec. 5.6 and its matched comparisons, followed by a separate controlled single-arm study.

Platforms and tasks. The evaluation uses three platforms: a Franka arm with an XHand dexterous hand, a PiPER-X ALOHA-style system with two leader and two follower arms (policies command only the follower arms), and a DOBOT X-Trainer dual-arm platform. The six tasks are dexterous block-to-bowl pick-and-place and toast insertion into a toaster with Franka–XHand, bimanual towel folding and peg insertion with PiPER-X, and gear assembly and USB insertion with X-Trainer (Table 38).

Data and policy. Each digital twin combines carefully built simulation models of the robots (e.g., their URDFs) and task objects with the surrounding scene (tabletop, lighting, and workspace setup), which is reconstructed from real-world captures with GPT-6 Astra; every reconstructed scene is manually reviewed before data collection. In each digital twin, AnnotateAnything's asset-specific annotations drive the asynchronous data-collection system (Sec. 4.1), which keeps only rollouts that succeed in simulation, until 200 demonstrations are collected per task. A pretrained $\pi_{0.5}$ policy Intelligence et al. (2025) is then fine-tuned separately for each task on these demonstrations with domain randomization. Policies are trained on multiple RTX A6000 GPUs and deployed directly

Table 37: Per-skill annotation-enabled rollout collection results.

Skill Family	Method	Data Succ.	Traj./h	Att./Succ.	Traj. Human
Grasp	No annotation / random	29.0	70	3.45	55.0
Grasp	VL-only annotation	67.5	150	1.48	72.7
Grasp	Ours annotation bank	90.5	300	1.11	90.8
DexGrasp	No annotation / random	21.5	48	4.65	50.8
DexGrasp	VL-only annotation	59.4	110	1.68	68.1
DexGrasp	Ours annotation bank	86.7	170	1.15	86.5
BiGrasp	No annotation / random	24.8	55	4.03	54.3
BiGrasp	VL-only annotation	61.2	105	1.63	71.7
BiGrasp	Ours annotation bank	87.8	145	1.14	86.8
BiDexGrasp	No annotation / random	18.6	38	5.38	52.6
BiDexGrasp	VL-only annotation	55.0	88	1.82	68.7
BiDexGrasp	Ours annotation bank	82.4	115	1.21	85.0
Articulation	No annotation / random	27.4	62	3.65	53.2
Articulation	VL-only annotation	64.8	122	1.54	73.0
Articulation	Ours annotation bank	88.6	150	1.13	88.1
Insertion	No annotation / random	19.6	42	5.10	49.4
Insertion	VL-only annotation	58.1	96	1.72	64.2
Insertion	Ours annotation bank	84.2	125	1.19	81.5
Hanging	No annotation / random	20.2	40	4.95	47.8
Hanging	VL-only annotation	57.9	90	1.73	66.5
Hanging	Ours annotation bank	83.4	110	1.20	83.6
Deformable	No annotation / random	15.0	30	6.67	49.4
Deformable	VL-only annotation	52.6	78	1.90	64.3
Deformable	Ours annotation bank	80.3	90	1.25	81.9
Navigation / Approach	No annotation / random	40.0	110	2.50	55.3
Navigation / Approach	VL-only annotation	79.4	195	1.26	73.8
Navigation / Approach	Ours annotation bank	90.1	280	1.11	88.6
Aggregate	No annotation / random	24.0	55	4.17	52.0
Aggregate	VL-only annotation	61.8	115	1.62	69.2
Aggregate	Ours annotation bank	86.0	165	1.16	85.9

on the physical robots (inference on an RTX 4090), without task-specific real-world demonstrations, real-world fine-tuning, or test-time adaptation, and each task is evaluated in 20 independently reset trials. A trial succeeds only if the task-specific terminal condition in Table 38 is reached without manual intervention.

Table 38: Real-world evaluation of Sec. 5.6. Successes are out of 20 trials per task; brackets give Wilson 95% confidence intervals. All $\pi_{0.5}$ settings use the same digital twin, 200 demonstrations per task, domain randomization, and evaluation protocol, and differ only in the data source; Ours+ACT uses our data with ACT trained from scratch. MoMaGen applies only to the rigid parallel-gripper tasks (n/a otherwise). VL-only+val. uses VL-only candidates that pass the same simulation validation during collection.

Task	Platform	Success criterion	Ours+$\pi_{0.5}$	MoMaGen+$\pi_{0.5}$	VL-only+val.	Ours+ACT
Dex. pick-and-place	Franka–XHand	Block rests inside the bowl after release	16 [58.4, 91.9]	n/a	15 [53.1, 88.8]	10 [29.9, 70.1]
Toast insertion	Franka–XHand	Toast is released fully inside the toaster slot	11 [34.2, 74.2]	n/a	9 [25.8, 65.8]	9 [25.8, 65.8]
Towel folding	PiPER-X	Towel is folded into the target folded state	15 [53.1, 88.8]	n/a	15 [53.1, 88.8]	10 [29.9, 70.1]
Peg insertion	PiPER-X	Peg is inserted into the hole and remains seated	15 [53.1, 88.8]	9 [25.8, 65.8]	15 [53.1, 88.8]	10 [29.9, 70.1]
Gear assembly	X-Trainer	Gear is seated on its shaft and meshes with the adjacent gear	14 [48.1, 85.5]	6 [14.5, 51.9]	14 [48.1, 85.5]	8 [21.9, 61.3]
USB insertion	X-Trainer	USB connector is fully inserted into the port	14 [48.1, 85.5]	8 [21.9, 61.3]	13 [43.3, 81.9]	9 [25.8, 65.8]
Total			85/120 [62.2, 78.2]	23/60 [27.1, 51.0]	81/120 [58.7, 75.2]	56/120 [38.0, 55.6]

MoMaGen comparison. To separate the data from the policy-training pipeline, we keep the policy, digital twin, number of demonstrations, domain randomization, and protocol fixed and replace only the data source with MoMaGen Li et al. (2026a). Following its single-source-demonstration setting, it generates demonstrations in the same digital twin from one human source demonstration per

task with object-centric subtask annotations; its mobile-base optimization is not used because the applicable tasks are fixed-base. It applies to the three rigid parallel-gripper tasks (peg insertion, gear assembly, and USB insertion); the two Franka–XHand tasks use a dexterous hand and towel folding involves a deformable object, both outside its supported setting. On the three applicable tasks, it reaches $23/60$ (38.3%) versus $43/60$ (71.7%) with ours (two-sided Fisher's exact test, $p \approx 4 \times 10^{-4}$; a task-stratified Cochran–Mantel–Haenszel test gives $p \approx 6 \times 10^{-4}$). Per task, the difference is individually significant only for gear assembly ($p = 0.026$; peg 0.105, USB 0.111), as expected with 20 trials per task. Our data require no human demonstrations.

VL-only annotation ablation. Under the same protocol, we replace only our annotations with VL-only candidates (App. H.8) that must pass the same simulation validation during collection (VL-only+val.; e.g., floating-hand validation for grasps), so that both data sources contain only validated interactions. Its policies reach $81/120$ (67.5%) versus $85/120$ (70.8%) with ours, a difference within the confidence intervals, but collecting the 200 successful demonstrations per task takes 2.7–$5.5\times$ longer (Table 39), counting validation time for both data sources (for ours, the time to build the bank), because more of its unoptimized candidates fail validation or rollouts. Traj./h in Table 3 excludes validation time and therefore does not reflect this cost. In this setting, our optimized bank therefore mainly reduces data-collection cost rather than changing real-world success.

Table 39: Simulation data-collection cost for the real-world tasks: wall-clock hours on one RTX 4090 to obtain 200 successful demonstrations per task. Times are end-to-end, including program start-up, validation, and full rollouts, and are therefore not directly comparable with the per-skill Traj./h in Table 3. Ours includes bank construction (generation, optimization, and validation) and collection. Candidates tried counts every candidate attempted until the 200th success: Ours draws them from its pre-validated bank, whereas VL-only+val. validates them during collection, so its count includes candidates rejected by validation. Ratio is the VL-only+val. total divided by the Ours total.

		Ours				VL-only+val.		
Task	Platform	Bank (h)	Collection (h)	Total (h)	Candidates tried	Total (h)	Candidates tried	Ratio
Dex. pick-and-place	Franka–XHand	0.5	2.2	2.7	243	7.4	710	2.7×
Toast insertion	Franka–XHand	0.6	3.2	3.8	260	12.9	830	3.4×
Towel folding	PiPER-X	1.6	6.4	8.0	272	30.0	703	3.8×
Peg insertion	PiPER-X	0.3	1.8	2.1	233	8.2	744	3.9×
Gear assembly	X-Trainer	0.5	2.1	2.6	248	14.2	1,103	5.5×
USB insertion	X-Trainer	0.6	1.7	2.3	241	6.5	589	2.8×

Policy swap (ACT). To test whether the data depend on the pretrained policy, we keep our 200 simulation demonstrations per task, the digital twins, domain randomization, and the evaluation protocol fixed and train ACT (Zhao et al., 2023) from scratch in place of $\pi_{0.5}$. It reaches $56/120$ (46.7%), with 8–10 successes per task. Our data thus support zero-shot sim-to-real transfer without large-scale policy pretraining, and the gap to $\pi_{0.5}$ ($85/120$) is consistent with the benefit of that pretraining. On the three rigid tasks, Ours+ACT ($27/60$) is comparable to MoMaGen+$\pi_{0.5}$ ($23/60$; Fisher's exact test $p = 0.58$), although these two settings differ in both data and policy.

Controlled single-arm study. We evaluate pretrained policies adapted using annotation-enabled simulation rollouts with domain randomization. A Franka Panda with its stock two-finger gripper performs Grasp, Insertion, Open Drawer, Close Lid, and long-horizon Pick-and-Place, while the same arm with an XHand attached through a custom mount performs DexGrasp. For each evaluated object or configuration, we fine-tune the pretrained $\pi_{0.5}$ policy Intelligence et al. (2025) on $1{,}000$ automatically collected simulation demonstrations generated with the corresponding simulated embodiment. This adaptation uses no additional real-world demonstrations, real-world fine-tuning, or test-time adaptation.

Grasp and DexGrasp use ten objects each; Insertion uses two peg configurations; Open Drawer uses two cabinets; Close Lid uses three lidded objects; and the long-horizon evaluation uses five five-object sets. The controlled evaluation contains 540 single-task trials and 100 long-horizon episodes. Each evaluated object, configuration, or long-horizon object set receives 20 independently reset physical runs. Success requires the specified terminal state without manual intervention; failures include drops, jams, missed targets, incomplete articulation, or timeout.

Table 40: Controlled single-arm real-world study. Pretrained $\pi_{0.5}$ policies are fine-tuned only on annotation-enabled simulation rollouts, without real-world demonstrations or real-world fine-tuning. Confidence intervals are reported in Appendix H.9.

Task	Trials	Success Criterion	Successes
Grasp	200	Object is lifted and held stably	186
DexGrasp	200	Object remains stably held by the dexterous hand	154
Insertion	40	Object reaches and remains in the target insertion region	28
Open Drawer	40	Drawer reaches the target open state	33
Close Lid	60	Lid reaches the target closed state	53
Long-horizon Pick-and-Place	100	All five objects are sequentially manipulated and placed successfully	61

The long-horizon task requires five objects to be manipulated and placed sequentially within each episode, and an episode is counted as successful only when the complete sequence succeeds. This controlled study demonstrates instance-specific sim-to-real feasibility; it includes no matched baseline (matched comparisons are reported above for the multi-platform evaluation) and does not establish generalization to unseen objects or configurations. Additional qualitative hardware demonstrations are excluded from the controlled totals.

H.8 Baselines and Ablations

Controlled ablations isolate semantic priors, 3D grounding, and physics validation. **Geometry-only** uses surface normals, curvature, free space, collision margins, reachability, and traversability without visual-language priors. **VL-only** instantiates skill candidates directly from grounded anchors, without optimization and physics validation. **w/o physics validation** generates and optimizes candidates but stores them without validator filtering; since its candidates before filtering are those of the full pipeline, Pass/Ready are not repeated, and Exec. and rollout metrics use its unfiltered bank. **Ours full** combines visual-language priors, 3D grounding, embodiment constraints, optimization, physics validation, and physics-aware augmentation. The random variant is used only for rollout collection. Table 41 lists the components and evaluation denominators of each setting. Because Exec. is computed over each setting's ready pairs, it is conditional on readiness; counting unready pairs as failures orders the settings as Ours full, w/o physics validation, VL-only, and Geometry-only.

Table 41: Components and evaluation denominators of the ablations in Table 3. End-to-end counts unready pairs as failures (Ready $\times$ Exec.).

Setting	VL priors	3D grounding	Optimi-zation	Validation filtering	Augmen-tation	Exec. denominator	Executed candidates	End-to-end (%)
Geometry-only	No	No	Yes	Yes	No	own ready pairs (58.5%)	validated, rank order	34.2
VL-only	Yes	Yes	No	No[†]	No	post hoc ready pairs (65.0%)	all stored, rank order	43.9
w/o phys. validation	Yes	Yes	Yes	No	No	Ours' ready pairs (87.4%)	all stored, rank order	55.2
Ours full	Yes	Yes	Yes	Yes	Yes	own ready pairs (87.4%)	validated, rank order	80.3

[†]Pass and Ready are computed post hoc with the same validator.

Framework-adapted external baselines. Results are reported in Sec. 5.4 and Table 4. Table 42 lists how each method's native output is converted into our common candidate interface; all converted candidates are evaluated with the same validator and rollout protocol as ours. Because conversion may lose information, these are framework-adapted estimates rather than official or end-to-end reproduced metrics of the original systems.

Table 43 lists the source of each baseline and the native components it retains.

Protocol-matched DexGrasp baseline. We additionally compare with BODex Chen et al. (2025a) using the same Dex3-1 hand, 520 audited asset–skill pairs, 10 candidates per pair, the same floating-hand validator, and the same G1 full-arm execution protocol. No output conversion or hand retargeting is applied. BODex obtains $40.1/84.8/90.2\%$ Pass/Ready/Exec., compared with $43.8/88.7/93.9\%$ for AnnotateAnything.

Table 42: Adaptation protocol for the framework-adapted external baselines in Table 4.

Method	Native Output	Adaptation in Our Framework	Evaluation Scope
ManiTwin-adapted	Verified grasp poses and functional / grasp points	Store the grasp annotations as candidates, then apply our grasp validator and rollout protocol.	Grasp
MOPS-affordance + heuristic	Pixel-level affordance labels	Treat affordance masks as visual anchors, sample heuristic action candidates, then apply our validator and rollout protocol.	Visual affordance
RoboGen-adapted	Generated tasks, scenes, and task programs	Convert generated task targets or scripted parameters into candidates, then apply our validator and rollout protocol.	Grasp, Articulation
MolmoBot-adapted	VLM-guided interaction proposals or actions	Use proposed interaction regions as anchors, instantiate the closest skill templates, then apply our validator and rollout protocol.	Grasp, Articulation, Navigation
GenManip-adapted	Generated manipulation programs or trajectories	Convert generated targets and waypoints into candidate annotations, then apply our validator and rollout protocol.	Grasp, Articulation, Insertion

Table 43: Provenance of the external baselines. Stages not listed (validation, inverse kinematics, motion planning, execution, and retries) follow our common protocol.

Method	Source	Native components retained
ManiTwin-adapted	Released annotations	Verified grasp poses and functional points
RoboGen-adapted	Official code	Task proposal, scene construction, and LLM-generated primitive parameters
MolmoBot-adapted	Official code	VLM interaction proposals, precomputed grasp sets, and candidate ranking
GenManip-adapted	Official code	LLM task programs with generated targets and waypoints
MOPS-affordance + heuristic	Released dataset	Pixel-level affordance masks
BODex	Official code	Bilevel grasp optimization with default parameters

H.9 Human Evaluation and Uncertainty Reporting

Human evaluation covers complete visual-language bundles, retained action annotations, and rollout trajectories. Each item is independently evaluated by five raters drawn from a pool of 15 non-author volunteers with experience in robotics, perception, planning, or simulation; all raters gave informed consent and received no compensation. The raters did not participate in method development, output generation, or evaluation-sample selection. Before evaluation, all raters complete calibration on nine held-out examples that are excluded from the reported results.

Method identities and validation outcomes are hidden. Outputs are randomized and interleaved across methods, and raters see the input asset or scene, the generated annotations, and the corresponding action or rollout visualization when applicable. Ratings use a 0–100 scale in discrete 10-point increments. We evaluate 360 visual-language bundles (90 per asset type) and, for action annotations and rollout trajectories, 120 asset–skill pairs per skill family and method. Generation-based evaluations use three independent seeds that produce separately generated sample sets, rather than repeated evaluations of fixed outputs.

For each example, we first average the five raters' scores and then report the mean and standard error over example-level scores. Type-specific Part, Affordance, and Keypoint scores are automatic reference-matching metrics rather than human ratings.

Table 44: Human evaluation dimensions. All dimensions are scored from 0 to 100.

Target	Dimensions	Overall Score
Visual-language bundle	Semantic correctness; 3D grounding; coverage; actionability	Average over dimensions
Action annotation	Task relevance; physical plausibility; executability; diversity	Average over dimensions
Rollout trajectory	Task alignment; motion quality; safety plausibility; demonstration usefulness	Average over dimensions

The rubric above defines the dimensions summarized by each overall human score. The disaggregated values below show the corresponding dimension-level results for visual-language bundles, physics action annotations, and collected trajectories.

Inter-rater agreement. Using interval-scale Krippendorff's α, agreement is 0.84 for visual-language bundles, 0.77 for action annotations, and 0.79 for rollout trajectories.

Table 45: Per-dimension human evaluation scores.

Target	Dim. 1	Dim. 2	Dim. 3	Dim. 4	Overall
Visual-language bundle	93.7	90.2	90.8	91.9	91.7
Physics action annotation	87.8	86.4	83.7	84.2	85.5
Collected trajectory	88.3	86.8	83.9	84.5	85.9

Confidence intervals. Paired confidence intervals use $10{,}000$ cluster-bootstrap resamples. For action and rollout rates, the asset–skill pair is the resampling cluster, so all candidates and repeated executions from the same pair remain together. Human-score comparisons use a paired bootstrap over matched items, stratified by asset type for visual-language bundles and by skill family for action annotations and trajectories. Real-world success-rate intervals use Wilson binomial intervals computed from the exact physical-trial totals.

Table 46: Summary of paired 95% confidence intervals for the main comparisons. *BODex rows use unpaired Newcombe intervals computed from the reported counts (520 pairs; Exec. over ready pairs), which ignore the pairing.

Comparison	Improvement	95% CI
Visual-language overall score	$+13.0$	$[11.4, 14.5]$
Action-annotation human score	$+14.7$	$[13.7, 15.7]$
Rollout-trajectory human score	$+16.6$	$[15.7, 17.6]$
ManiTwin Wang et al. (2026a) matched Pass	$+4.5$ pp	$[3.6, 5.4]$
ManiTwin matched Ready	$+13.8$ pp	$[8.1, 19.5]$
ManiTwin matched Exec.	$+10.7$ pp	$[8.8, 12.6]$
BODex Chen et al. (2025a) matched Ready*	$+3.8$ pp	$[-0.3, 8.0]$
BODex matched Exec.*	$+3.7$ pp	$[0.1, 7.3]$
BODex matched end-to-end execution*	$+6.7$ pp	$[1.9, 11.6]$

Real-world uncertainty. For the multi-platform evaluation, per-task Wilson 95% confidence intervals are given in Table 38. For the controlled single-arm study, Wilson 95% confidence intervals are $[88.6, 95.8]\%$ for Grasp, $[70.7, 82.3]\%$ for DexGrasp, $[54.6, 81.9]\%$ for Insertion, $[68.1, 91.3]\%$ for Open Drawer, $[77.8, 94.2]\%$ for Close Lid, and $[51.2, 70.0]\%$ for long-horizon Pick-and-Place.

H.10 Timing Metrics

Annotation throughput is measured as retained annotations per wall-clock minute and includes candidate generation, geometry filtering, trajectory or pose optimization, physics validation, and serialization. The per-skill values are reported in Table 28. For variants without validator filtering, throughput is measured on the stored candidate bank, and VL-only Pass/Ready are computed post hoc. Rollout throughput is successful trajectories per wall-clock hour and includes candidate retrieval, goal-set inverse kinematics, collision-aware motion planning, full-robot execution, and trajectory validation. Compute resources are listed in Appendix J.2.

I Related Work

I.1 Simulation-Based Automatic Data Collection

Simulation-based data generation often relies on demonstration reuse or optimization in simulation. RLBench provides expert demonstrations generated through waypoint-guided motion planning, while MimicGen transforms a small set of human demonstrations to new scene configurations, object instances, and robot arms James et al. (2019); Mandlekar et al. (2023); MoMaGen extends this demonstration-transformation paradigm to multi-step bimanual mobile manipulation under reachability and visibility constraints Li et al. (2026a). Such methods require human source demonstrations and subtask annotations for each task, whereas AnnotateAnything derives interaction targets directly from the assets; Sec. 5.6 compares the two data sources under a matched real-world protocol. RoboGen uses foundation and generative models to propose tasks, construct scenes, and produce training supervision, whereas GenSim uses large language models to generate simulation task programs and expert demonstrations Wang et al. (2023b; 2024). Benchmark suites such as Meta-World and

GPU simulators such as Isaac Gym support large-scale reinforcement-learning experimentation, and simulation-trained dexterous systems show that difficult skills can be learned without human demonstrations Yu et al. (2019); Makoviychuk et al. (2021); OpenAI et al. (2018). Demonstration-based systems often depend on task programs or seed demonstrations; optimization-based systems require substantial reward, curriculum, and training design, and commonly produce task- or controller-specific outputs.

AnnotateAnything targets a complementary layer: reusable semantic and executable annotations that can initialize multiple data-collection and policy-learning pipelines, rather than demonstrations or policies specialized to one training setup.

I.2 Automatic Manipulation Annotation and Affordance Generation

Grasping has received especially extensive attention in automatic manipulation annotation. Dex-Net 2.0 and ACRONYM synthesize large grasp datasets using analytic metrics and physics simulation, respectively Mahler et al. (2017); Eppner et al. (2021); 6-DOF GraspNet and Contact-GraspNet learn grasp proposals from point clouds and cluttered observations Mousavian et al. (2019); Sundermeyer et al. (2021). DexGraspNet extends large-scale annotation to dexterous hands, BODex scales dexterous-grasp synthesis with GPU-parallel bilevel optimization, and recent work also addresses bimanual dexterous grasp synthesis Wang et al. (2023a); Chen et al. (2025a); Shao & Xiao (2024).

Beyond grasping, Where2Act and VAT-Mart learn where and how to interact with articulated objects Mo et al. (2021a); Wu et al. (2022), while GAPartNet provides generalizable actionable-part semantics and poses Geng et al. (2023). O2O-Afford learns object–object affordances, OmniHang estimates stable hanging configurations and collision-free paths, and IndustReal develops simulation-to-real methods for contact-rich assembly Mo et al. (2021b); You et al. (2021); Tang et al. (2023). 3D AffordanceNet benchmarks 3D affordance understanding over a predefined affordance vocabulary Deng et al. (2021). These systems provide strong supervision but are generally specialized by skill family, embodiment, contact model, or affordance vocabulary. AnnotateAnything instead uses one candidate-bank interface to organize grasping, articulation, placement, insertion, hanging, deformable manipulation, and navigation annotations while retaining skill-specific generation and validation.

Vision-language models have also been used to produce manipulation-relevant labels: Articulate-Anything infers articulation models of objects, Manipulate-Anything generates robot demonstrations with VLM guidance, and RoboPoint predicts spatial affordance points Le et al. (2025); Duan et al. (2024); Yuan et al. (2024). These systems output one kind of label or demonstration per query, whereas AnnotateAnything grounds VLM priors on each asset and stores multiple physically validated, executable candidates per anchor and skill.

I.3 Large-Scale 3D Assets and Generative Asset Creation

ShapeNet established large-scale CAD repositories, and PartNet introduced hierarchical part annotations Chang et al. (2015); Mo et al. (2019). PartNet-Mobility provides articulated objects with motion annotations and is integrated with the SAPIEN ecosystem SAPIEN (2020); Xiang et al. (2020). Objaverse further expanded the scale and diversity of web-sourced 3D objects Deitke et al. (2023). Manipulation-oriented resources add narrower forms of supervision: ACRONYM and DexGraspNet provide grasp annotations, and GAPartNet supplies actionable-part semantics and poses Eppner et al. (2021); Wang et al. (2023a); Geng et al. (2023).

Text-to-3D systems such as DreamFusion and Shap-E generate novel 3D assets from language Poole et al. (2022); Jun & Nichol (2023). However, general-purpose raw or generated assets typically do not include the complete manipulation metadata required by robot-learning systems, such as joint and part structure when applicable, functional parts, contact semantics, affordances, and executable trajectories. AnnotateAnything serves as a conversion layer from heterogeneous assets to interaction-ready supervision rather than as another asset repository or generative model.

I.4 Simulation Environments and Manipulation Benchmarks

The simulator ecosystem spans general-purpose rigid-body dynamics, articulated interaction, and large-scale GPU execution. MuJoCo and PyBullet are widely used physics engines Todorov et al.

(2012); Ellenberger (2018–2019); Isaac Gym enables GPU-resident parallel learning, and Isaac Sim supports richer industrial and photorealistic workflows Makoviychuk et al. (2021); NVIDIA (2025). SAPIEN emphasizes articulated-object interaction, while RoboSuite provides a modular MuJoCo-based manipulation framework Xiang et al. (2020); Zhu et al. (2020).

Benchmark suites emphasize different capabilities: Meta-World targets multitask and meta-reinforcement learning, RLBench provides a large task library with generated demonstrations, ManiSkill and ManiSkill2 broaden object and manipulation coverage, and BEHAVIOR focuses on household and longer-horizon embodied activities Yu et al. (2019); James et al. (2019); Mu et al. (2021); Gu et al. (2023); Srivastava et al. (2021). These simulators and benchmarks provide essential execution and evaluation infrastructure, but task authoring and annotation are commonly benchmark-specific. AnnotateAnything is complementary infrastructure intended to populate such environments with transferable, interaction-ready annotations across assets, embodiments, and skill families.

I.5 Skill-Centric Robot Learning and Atomic Manipulation Skills

The options framework formalizes temporally extended actions and hierarchical decision making Sutton et al. (1999). Recent robot systems instantiate related ideas through explicit skills or callable APIs: SayCan selects predefined skills by combining language-model scores with affordance-based feasibility, Code as Policies generates programs that call perception and control APIs, and Composable Part-Based Manipulation composes manipulation constraints using object-part decomposition and part correspondences Ahn et al. (2022); Liang et al. (2022); Liu et al. (2023).

These systems still rely on a skill or API inventory and on mechanisms that ground those skills to objects through contacts, parts, trajectories, constraints, or correspondences. AnnotateAnything addresses this grounding layer by automatically generating candidates that instantiate atomic skills, allowing skill libraries and hierarchical planners to extend beyond manually authored primitive sets.

J Limitations, Broader Impact, and Asset Licensing

J.1 Limitations

The system remains limited by its upstream models, observations, and simulation fidelity. Visual-language annotations can be incomplete or inaccurate when task-relevant parts are occluded, segmentation is coarse, materials are reflective, or geometry is ambiguous. Grounding is generally stronger for rigid and moderately articulated objects than for highly deformable assets, small contact-rich components, or severely cluttered scenes. Action validation depends on collision geometry, articulation metadata, contact modeling, and embodiment-specific planning; simulated success therefore does not guarantee real-world success. The current benchmark also does not cover every manipulation family, embodiment, or deployment constraint, and scaling to very large asset banks requires substantial compute and simulator infrastructure. Structured parsing and segmentation-identifier checks reject detectable malformed or ungrounded outputs, but they do not identify or correct every retained semantic error. Performance therefore remains dependent on the selected VLM and segmentation pipeline, especially for unusual objects, poor segmentations, and cluttered scenes. Physics and geometric validation can remove a measurable subset of inappropriate candidates but cannot guarantee semantic correctness. In addition, framework-adapted comparisons may partly reflect output-conversion fidelity, and the matched real-world comparison covers only MoMaGen on three rigid parallel-gripper tasks with 20 trials per task and does not evaluate cross-object generalization. Articulation annotation reads revolute and prismatic joints from the asset's USD joint model, so assets without articulation metadata are not covered by that skill family. Floating parallel-jaw validation uses static and dynamic friction of 2.0 and disables gravity during approach and closure without modeling the support surface; it therefore acts as a permissive filter, and execution success is reported separately on the complete robot.

J.2 Compute Resources

Visual-language annotation runs on NVIDIA RTX A6000 GPUs with 48 GB of GPU memory and includes VLM inference, multi-view processing, part, keypoint, and affordance grounding, and

evaluation-sample generation. Physics validation and rollout collection run in headless Isaac Sim on NVIDIA RTX 4090 GPUs with 24 GB of GPU memory; CPU workers handle asset loading, scene orchestration, logging, and asynchronous scheduling. Jobs are dispatched to available simulator workers according to asset complexity, embodiment, and skill type, so a scheduled job is not necessarily an independent Isaac Sim process. Visual-language annotation used two shared clusters of 9 and 5 RTX A6000 GPUs, and physics validation and rollout collection used 4 RTX 4090 GPUs. Because the A6000 clusters were shared with other jobs, we do not report total GPU-hours. Instead, the throughput in Table 3 is measured on a single RTX 4090, so generating 10^4 validated annotations takes about 13 min for Grasp, 50 min for DexGrasp, 95 min for BiGrasp, 105 min for BiDexGrasp, 91 min for Articulation, 143 min for Insertion, 167 min for Hanging, 222 min for Deformable, and 33 min for Navigation per GPU.

Ann./min includes candidate generation, geometry filtering, optimization, physics validation, and annotation-bank serialization; Traj./h includes candidate retrieval, goal-set IK, collision-aware motion planning, full-robot execution, and trajectory validation. Floating-gripper grasp validation is generally cheaper than dexterous, bimanual, articulated, deformable, or navigation-related evaluation. The reported action-annotation and rollout throughput values are measured on the audited action-quality suite and the rollout evaluation in Sec. H. The manually annotated visual-language subset is evaluated separately and is not included in the action or rollout throughput denominators. Random seeds control asset, skill, embodiment, initial pose, scene layout, camera, material, lighting, and disturbance sampling, but GPU-parallel asynchronous Isaac Sim execution is not assumed to be bitwise deterministic. The reported evaluation totals exclude exploratory development runs, failed preliminary experiments, and hyperparameter-search compute; throughput values in Tables 3 and 37 should not be interpreted as hardware-independent constants.

J.3 Broader Impact

The main potential benefit is reducing the cost of converting raw 3D assets into reusable robot-learning supervision, enabling broader benchmarks, more diverse asset coverage, and less repeated manual simulation authoring. The same annotations can support quantitatively evaluated part, affordance, and keypoint pipelines, together with illustrative robot reasoning VQA and 3D VLM instruction-tuning repurposings.

The pipeline can also propagate errors from upstream models, asset metadata, or simulators into downstream policies. In addition, scalable simulation can make it easier to train large manipulation systems whose deployment behavior is not fully understood. Generated annotations should therefore be treated as useful but fallible supervision, accompanied by human auditing, task-specific validation metrics, and additional checks before real-world transfer.

J.4 External Asset Sources and License Handling

The processed annotation pool contains 17,005 assets retained after normalization and successful simulator import. These counts describe processed assets rather than the raw sizes of the upstream repositories. For every retained asset, we retain the upstream source family, asset type, upstream identifier when available, and any available license or redistribution metadata.

Filtering and exclusions. We apply common source-independent preprocessing before annotation. Assets are excluded when they contain corrupt, empty, or unreadable geometry; unresolved dependencies that prevent rendering or simulator import; unrecoverable unit, scale, orientation, or coordinate-frame errors; unusable collision geometry; invalid articulation axes, links, limits, or moving-part metadata; garment geometry that cannot be initialized stably; or scene packages with invalid references, unusable layouts, or no valid navigable and interaction-ready region. Skill-specific incompatibility does not necessarily remove an asset from the pool: for example, an object unsuitable for insertion may remain valid for grasping. Evaluation is therefore defined over valid asset–skill pairs.

Deduplication and rejection accounting. Repeated imports sharing the same upstream identifier or ingestion record are collapsed when this information is available. We do not perform exhaustive cross-source geometric nearest-neighbor deduplication and therefore do not claim that the processed

Table 47: External asset sources, exact retained counts, and release handling based on applicable upstream terms and available per-asset metadata. Raw assets are redistributed only when permitted; otherwise, we release derived annotations, source identifiers, and acquisition instructions.

Asset family	Source	Retained assets	Release handling
Rigid	Objaverse Deitke et al. (2023)	5,000	Follow collection-level and per-object license metadata; preserve source identifiers and required attribution.
Rigid	ShapeNet Chang et al. (2015)	4,500	Release derived annotations and source mappings rather than restricted raw meshes.
Rigid	OmniObject3D	3,038	Preserve upstream identifiers and attribution under the applicable upstream terms.
Articulated	PartNet-Mobility Xiang et al. (2020); SAPIEN (2020)	800	Do not redistribute restricted raw geometry or articulation files; release derived annotations and source mappings.
Articulated	PhysX-Mobility	1,294	Preserve applicable attribution and non-commercial restrictions.
Deformable / garment	GarmentLab Lu et al. (2024b)	1,500	Release derived annotations and source mappings; do not assume blanket permission to redistribute third-party assets.
Deformable / garment	ClothesNet Zhou et al. (2023)	667	Release derived annotations and source mappings; users obtain restricted raw assets from the upstream provider.
Room-scale	AgentWorld	114	Release derived annotations and source identifiers; do not assume blanket redistribution permission for underlying scene assets.
Room-scale	GRScenes	92	Preserve applicable attribution, non-commercial, and share-alike restrictions for redistributable material.
Total	**9 source families**	**17,005**	–

pool is free of all near-duplicates; visually or geometrically related assets may still differ in scale, topology, collision geometry, articulation, or applicable manipulation annotations. Because failed downloads and imports were not logged uniformly across all upstream importers, we report exact retained processed counts and explicit exclusion criteria rather than estimating incomplete source-level rejection totals.

Release policy. For every retained asset, the planned public package records the source family, upstream identifier when available, asset type, applicable skills, conversion status, and available license or redistribution metadata; unresolved cases are marked explicitly rather than assigned an inferred license. Raw third-party assets are redistributed only when the applicable upstream terms permit it. Otherwise, the release contains derived annotations, source mappings, and instructions for obtaining the upstream assets directly from their providers. The construction of the audited suite and manual-reference subset is described separately in Appendix H.1.

J.5 Artifacts and Documentation

The anonymous supplementary materials provide supporting artifacts for the reported results. These include the paper and detailed method appendix, qualitative videos, redistributable demo assets, a code snapshot covering the principal annotation, simulation data-generation, and full-robot execution paths, and representative language, visual, and action annotation packages with asset metadata.

Public-release packaging and documentation remain to be completed. The planned public release will provide a cleaned and documented codebase, the packaged full-scale derived annotation collection, fixed manifests for the retained pool, audited asset–skill suite, and manual-reference subset, rollout metadata, evaluation and uncertainty scripts, prompts and schemas, and complete provenance and license mappings. Restricted raw third-party meshes are neither included in the anonymous supplement nor planned for redistribution; users will instead receive source identifiers and instructions for obtaining and processing the upstream assets.

Language prompts and schemas are documented in Sec. C.1 and the remaining language-annotation subsections, visual grounding in Sec. D, and action generation and validation in Sec. E. The corresponding public manifests will link these derived annotations and rollout metadata to their provenance and redistribution fields.