AnnotateAnything: Automatic Annotation of 3D Assets for Robot Manipulation

Anonymous Author(s)
Affiliation
Address
email

Abstract

Simulation enables scalable robot data collection, but raw 3D assets provide only geometry and appearance, lacking the semantic, interactive, and physical knowledge needed to specify where and how robots should act. In this work, we present **AnnotateAnything**, a general automatic annotation framework that converts passive 3D assets into manipulation-ready assets with structured, diverse, and executable manipulation labels. AnnotateAnything is built around two complementary pipelines. First, a unified **visual-language annotation pipeline** leverages vision-language reasoning to infer object semantics, interaction constraints, and 3D-grounded cues such as keypoints, and parts, providing human-prior guidance for identifying meaningful interaction regions. Second, a fully automatic and massively parallel **physics annotation pipeline** grounds these priors in each asset's geometry, kinematics, and physical constraints through candidate generation, geometry optimization, trajectory generation and data augmentation. This pipeline produces diverse and executable action annotations, including grasp poses, dexterous contacts, articulation waypoints, insertion directions, hanging affordances, and navigation targets. Based on the generated annotations, we further build an asynchronous parallel simulation data-collection system across diverse objects, tasks, and robot embodiments. Experiments demonstrate that AnnotateAnything achieves superior annotation efficiency, data-collection efficiency, and task success rates over existing annotation and data-generation pipelines, while also supporting downstream tasks such as affordance detection, robotic VQA, and visual instruction finetuning. We will open-source our code, annotations, and benchmark to facilitate future research. Project page: XXXX.

1 Introduction

Training general-purpose robotic agents requires large amounts of diverse interaction data [1–5]. Recent progress in imitation learning and robot foundation models has further highlighted the importance of scaling robot data across objects, scenes, and tasks [6–12]. However, collecting real-world robot trajectories remains expensive, labor-intensive, and difficult to scale [13–16]. Simulation provides a promising alternative, where interactions can be generated automatically, safely, and in parallel [17–20]. Recent asset repositories and simulation benchmarks further provide rich geometric and visual substrates for simulation-based data generation [21–24]. Yet assets and environments alone are not enough: raw 3D assets usually describe only shape and appearance, but lack the semantic, interactive, and physical knowledge that makes robot data valuable, such as where to grasp, which part can move, how an object should be articulated, or where insertion and hanging are feasible. Therefore, the central bottleneck is not merely obtaining more assets, but automatically producing manipulation annotations that turn passive 3D geometry into actionable robot interaction data.

Submitted to 40th Conference on Neural Information Processing Systems (NeurIPS 2026). Do not distribute.

Figure 1: Teaser. High-level overview of AnnotateAnything and its visual-language-action annotation pipeline.

Generating manipulation annotations for raw 3D assets presents four key challenges. First, useful annotations must encode human interaction priors so that the resulting robot trajectories are human-like, safe, and likely to succeed. *This requires automatic annotation to incorporate high-level human-prior reasoning and interaction understanding* **(C1)** [25, 26]. Second, semantic annotation alone is insufficient: labels must be grounded in each asset's geometry and kinematics, since feasible actions depend on shape, collision constraints, and articulated structure. *The challenge is to transform abstract interaction intent into asset-specific executable labels* **(C2)** [27, 28]. Third, diversity is essential for task deployment, as manipulation often admits multiple valid solutions across grasps, poses, and interaction strategies. *Automatic annotation must thus preserve diverse feasible labels rather than collapse to a single canonical solution* **(C3)** [29, 30]. Finally, annotations must scale across large numbers of assets, categories, and tasks while preserving each asset's geometric precision and interaction specificity. *This calls for a fully scalable and automated process that avoids per-asset manual design or task-specific engineering* **(C4)**.

Producing annotations that meet these requirements remains difficult. Existing works often rely on human annotation to provide semantic, interactive, or physical knowledge [31, 30, 32, 33], which is costly, labor-intensive, hard to scale, and limited in diversity. RL-based alternatives reduce manual labeling but require careful reward engineering, complex training, and substantial computation, while often producing behaviors misaligned with human priors and hard to deploy in realworld [34, 35].

Therefore, we present **AnnotateAnything**, a general automatic annotation framework that transforms raw 3D assets into manipulation-ready assets with structured, actionable, and executable robot manipulation labels. AnnotateAnything provides three complementary types of annotations: language annotations for semantic and interaction-level reasoning, visual annotations for 3D-grounded understanding, and action annotations that specify executable manipulation parameters such as grasp poses, articulation waypoints, and navigation trajectory.

To inject human interaction priors, AnnotateAnything builds a vision-language annotation pipeline that leverages VLMs to bridge language reasoning and visual grounding. Language annotations provide multi-granularity reasoning from object semantics to interaction constraints, while visual annotations translate such reasoning into 3D-grounded cues such as affordances, keypoints, and part segmentation. These cues localize meaningful interaction regions and guide the generation of action annotations, incorporating VLM-derived human priors and interaction understanding into automatic annotation **(tackling C1)**. To ground annotations in each specific asset, AnnotateAnything combines geometry-based optimization with asset-specific physical reasoning. Geometry optimization refines grasps, contacts, and interaction regions under local shape and collision constraints, while physical reasoning derives waypoints and motion directions from the asset's kinematic structure and physical properties. This converts abstract priors into precise, asset-specific executable labels **(tackling C2)**[36–38]. To preserve diversity, AnnotateAnything generates annotations at three levels. Within each task, it produces many feasible candidates with different contacts, approach directions,

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 [33]	✔	✘	✔	✘	◯	◯	✘	✔	✘	✘	✘	✘	✘
InternData-N1 [44]	✘	✘	✘	✘	✘	✘	✔	✘	✔	✘	◯	✘	✔
RoboTwin2.0 [31]	✔	✘	✔	✘	◯	✘	✘	✔	✘	✘	✘	✘	✘
MolmoBot [45]	✔	✘	✘	✘	◯	✘	◯	✔	◯	✘	✔	✘	✔
RoboGen [34]	✔	✘	✘	✘	✔	✘	✘	✔	◯	✘	◯	✔	✘
GenManip [46]	✔	✘	✘	✘	◯	✘	✘	✔	✘	✘	✘	✘	✘
OmniManip [47]	✔	✘	✘	✘	◯	✘	✘	✔	✘	✘	✘	✘	✘
GenieSim 3.0 [48]	✔	◯	◯	◯	◯	✘	✘	✔	◯	✘	✘	✘	✘
AnnotateAnything	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔

orientations, and interaction configurations, rather than a single canonical label. Across tasks, it covers primitives such as grasping, dexterous manipulation, bimanual interaction, articulation, folding, hanging, and mobile manipulation. Across embodiments, it supports parallel-jaw grippers, dexterous hands, dual-arm systems, mobile manipulators, and humanoids **(tackling C3)**[39, 40]. To enable scalability, AnnotateAnything builds a complete and fully parallelized physics annotation pipeline, covering candidate generation, geometry optimization, label selection, trajectory generation, physical validation, and data augmentation without per-asset manual design. With CUDA-accelerated optimization and validation [36], the pipeline scales across assets and tasks while preserving asset-specific accuracy, interaction feasibility, and high execution success rates **(tackling C4)**..

Based on the generated annotations, we set up an asynchronous parallel data-collection system for efficient simulation-based robot data collection. Beyond data collection, our integrated visual-language-action annotations further support several downstream applications, including affordance and keypoint detection [41–43], robotics-oriented VQA and reasoning, and visual instruction finetuning for multimodal robot models. Experiments show that AnnotateAnything achieves substantially higher annotation efficiency, data-collection efficiency, and task success rates than existing annotation or data-generation pipelines, demonstrating the practical value of automatic, diverse, and asset-specific manipulation annotations.

In summary, our contributions are as follows:

- We present **AnnotateAnything**, a general **automatic annotation framework** that converts passive 3D assets into **manipulation-ready assets** through structured, actionable, and executable annotations.
- We design a unified **visual-language-action annotation pipeline** that integrates language reasoning, 3D visual grounding, and asset-specific action label generation, enabling **human-prior reasoning**, **3D-grounded understanding**, and **executable manipulation specification** across diverse objects, tasks, and robot embodiments.
- We develop a fully automatic and massively parallel **physics annotation pipeline**, covering candidate generation, geometry optimization, label selection, trajectory generation, physical validation, and data augmentation, with CUDA-accelerated modules for scalable annotation without sacrificing **diversity**, **feasibility**, or **asset-specific precision**.
- We demonstrate **superior annotation efficiency**, data-collection efficiency, and task success rates over existing methods, together with utility across several downstream tasks.

2 Related Work

Automatic Data Collection for Robot Learning. Automatic data collection mainly follows two paradigms. Annotation-based methods generate structured, often human-like trajectories from demonstrations, scripts, or manually specified task annotations [33, 31, 49], but are limited by human effort and task-specific design. RL-based methods automate data generation in simulation [35, 34, 50–53], but require costly training, reward engineering, and careful reset or success definitions, while often producing behaviors that are not human-like. In contrast, AnnotateAnything automatically generates skill-consumable annotations from raw 3D assets, enabling reusable atomic skills to collect structured and scalable robot interaction data.

Automatic Annotation for Manipulation Existing automatic annotation methods often formulate manipulation annotation as task-specific optimization, such as generating grasps, contact points, affordances, or motion waypoints for a particular skill. While effective in specific settings, these methods usually cover limited object diversity and annotation scale[29, 46], suffer from low success rates on complex assets[50, 51, 38, 54], or support only a narrow set of embodiments and task types[55, 53]. AnnotateAnything provides a more general annotation pipeline that produces diverse skill-consumable labels across grasping, dexterous manipulation, bimanual interaction, articulation, insertion, and hanging, enabling scalable data collection over broad 3D assets and manipulation skills. Detailed discussion please refer to Appendix F

3 Method

AnnotateAnything takes as input a raw 3D asset, ranging from a single object to a room-scale scene, and converts it into a manipulation-ready asset with hierarchical language, visual, and action annotations. Our framework first leverages VLMs to infer human interaction priors and ground them into multi-level asset and scene annotations, including sparse/dense descriptions, keypoints, part segmentation, occupancy maps, floor plans, and top-view layouts (Sec. 3.1). These annotations are generated through hierarchical language reasoning (Sec. 3.1.1), hierarchical visual grounding (Sec. 3.1.2), and cross-level asset-to-room composition (Sec. 3.1.3). The physics-based pipeline then converts these visual-language cues into executable action annotations across rigid objects, articulated objects, garments and deformable objects, and room-scale scenes (Sec. 3.2), using a unified action schema (Sec. 3.2.1), a general generation-and-validation pipeline (Sec. 3.2.2), and primitive-specific instantiations (Sec. D).

Table 2: 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;
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

3.1 Hierarchical Visual-Language Annotation Pipeline

3.1.1 Hierarchical Language Annotation

Asset-level language annotation. For each object-level asset, we render multi-view RGB observations and use Qwen3-VL [56] to generate **three-level language annotations**: a semantic phrase, a functional sentence, and a dense paragraph for **part-aware interaction reasoning**. The dense annotation grounds segmented parts with their functions and feasible interactions, guiding downstream visual grounding and action generation.

Room-level language annotation. Given selected room views and the labeled top-view occupancy map, we use Qwen3.5-VL [56] to produce **three-level room annotations**: a layout summary, a furniture-and-zone description, and a dense paragraph covering object relations, navigable regions, interaction areas, and possible long-horizon robot tasks. These annotations complement asset-level labels with functional scene context.

3.1.2 Hierarchical Visual Annotation

Asset-level visual annotation. For each object-level asset, we reconstruct a fused 3D point cloud from multi-view RGB-D observations, making the pipeline consistent with real-world visual inference. We annotate **semantic keypoints** by sampling candidates with FPS and using the VLM to select functionally important anchors for interaction reasoning. In parallel, we generate **part segmentation annotations** with a native 3D pipeline based on Hunyuan3D-style decomposition, P3-SAM [57],

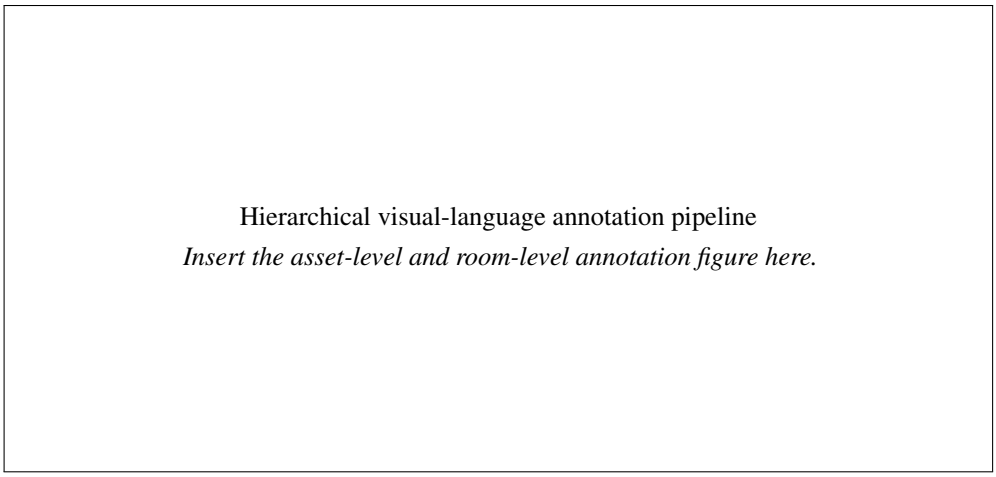

Figure 2: Hierarchical visual-language annotation pipeline. AnnotateAnything generates asset-level language and visual annotations, including sparse/dense descriptions, keypoints, part segmentation, and affordance regions, and composes them into room-level annotations such as scene descriptions, occupancy maps, floor plans, top-view maps, and object layouts.

and X-Part [58]. These visual annotations provide 3D-grounded cues for downstream physics-based action annotation.

Room-level visual annotation. For each room-scale scene, we generate visual annotations by actively scanning the environment with simulated LiDAR and ray casting, and then stitching the observations from individual rooms into a global scene representation. To capture spatial structure at different levels, we construct multi-height occupancy maps, where each height slice encodes free space, obstacles, and object geometry visible at that elevation. We further derive floor-plan and wall-structure annotations by using VLM-based filtering to remove movable objects and furniture from the raw occupancy evidence, retaining only persistent structural elements such as walls, doors, and room boundaries. These room-level visual annotations provide global geometric context for spatial exploration, navigation target generation, and scene-level action annotation.

3.1.3 Cross-level Composition and Consistency

We build room-level annotations by composing object-level annotations through scene instance identities and 6D object poses. For each room-scale scene, we extract individual object instances and process each one with the asset-level annotation pipeline, producing object-centric language descriptions, keypoints, and part-masks. These annotations are then transformed from the object coordinate frame to the global room coordinate frame and associated with room-level occupancy maps, floor plans, top-view layouts, and scene descriptions. The resulting cross-level representation links each object instance in the room to its semantic, visual, and affordance annotations, while maintaining shared identifiers across language annotations, visual maps, and physics-based action labels.

3.2 Physics-based Action Annotation Pipeline

The visual-language pipeline provides semantically meaningful 3D interaction regions, which are further grounded in robot embodiment and physical feasibility by our physics-based action annotation pipeline. This pipeline converts grounded visual-language priors into structured, executable, and validation-aware action annotations through candidate target generation, trajectory generation, trajectory optimization, physics validation, and physics-aware augmentation. The resulting manipulation-ready assets contain diverse action labels, including grasp poses, dexterous contacts, articulation waypoints, insertion directions, hanging poses, deformable-object trajectories, and navigation targets.

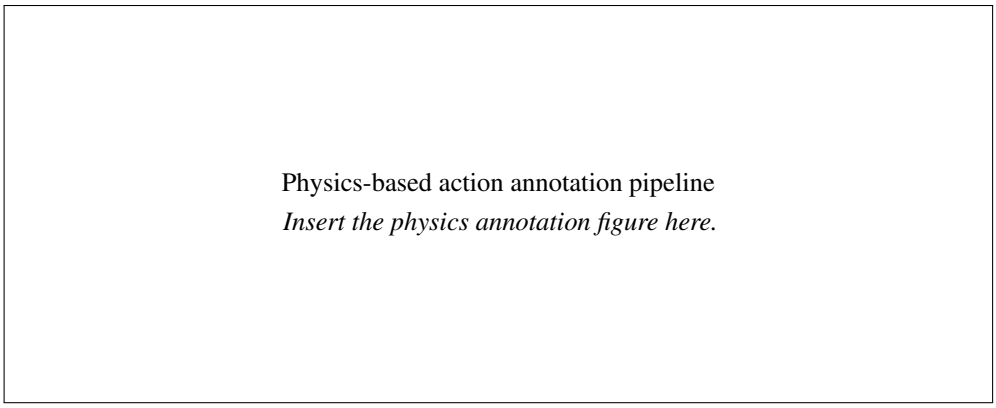

Figure 3: Physics-based action annotation pipeline. Grounded visual-language priors are converted into executable action annotations through candidate target generation, trajectory generation, trajectory optimization, physics validation, and physics-aware augmentation.

3.2.1 Unified Action Annotation Schema

A key design goal of our action annotation schema is to preserve **action diversity** while maintaining **functional consistency**. Instead of assigning a single canonical action to each object or part, we organize action annotations as a hierarchical candidate bank built on top of visual annotations. Given an asset $\mathcal{A}$, the visual annotation stage provides grounded interaction anchors

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

where $\mathcal{P}$ denotes part-level regions, $\mathcal{K}$ denotes semantic keypoints, $\mathcal{R}^{\mathrm{aff}}$ denotes affordance regions, and $\mathcal{R}^{\mathrm{scene}}$ denotes scene-level regions such as reachable free space or object-centric approach regions. Importantly, visual grounding provides not only the spatial anchor h, but also its functional affordance $\phi(h)$, such as grasp-for-lifting, grasp-for-opening, insertion, hanging, or navigation approach. Thus, parts and keypoints define local functional manipulation targets, while scene-level regions define navigation and approach targets.

For each anchor $h \in \mathcal{H}(\mathcal{A})$, we infer compatible skills $\mathcal{S}(h)$ from its functional affordance $\phi(h)$, language priors, geometry, and task constraints. This is crucial for long-horizon manipulation, where a physically stable grasp may be insufficient unless it supports the intended downstream interaction, e.g., grasping a mug handle for pouring or a cabinet handle for opening. We then maintain a **candidate bank** for every anchor-skill pair:

$$\mathcal{B}_{h,s} = \left\{ a_{h,s}^{(i)} \right\}_{i=1}^{N_{h,s}}, \qquad s \in \mathcal{S}(h),$$

where $N_{h,s}$ is the number of feasible candidates for skill s at anchor h. Each candidate action is represented as

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

Here, s is the skill type, $o(h)$ is the associated object, part, or scene instance, and $\phi(h)$ conditions action generation with functional affordance. The concrete target $x^{(i)}$ is instantiated from the anchor, such as a contact point, surface patch, support edge, opening, articulated handle, or navigation base pose. The parameters $\theta^{(i)}$ store skill-specific quantities such as a 6D grasp pose, gripper width, contact set, hand configuration, insertion direction, hanging pose, or target base pose. The optional trajectory $\tau^{(i)}$ stores waypoints or motion templates for extended actions. The metadata $v^{(i)}$ records physical and functional feasibility, including collision status, IK feasibility, contact stability, task success, trajectory statistics, and affordance preservation. The descriptor $d^{(i)}$ records diversity factors such as target location, approach direction, contact mode, trajectory family, and embodiment parameters.

This schema defines action annotation as a **one-to-many mapping** from grounded visual anchors to function-conditioned executable candidates. Diversity exists across anchors and within each anchor-skill pair, since $\mathcal{B}_{h,s}$ stores multiple validated candidates with different poses, contacts, approach directions, trajectories, and feasibility scores. As a result, AnnotateAnything provides downstream

modules with a diverse bank of **physically feasible and functionally meaningful** action labels rather than a single heuristic annotation.

3.2.2 General Physics-based Action Generation Pipeline

Candidate target generation. Given a skill type s, we first use language annotations and visual grounding to localize skill-compatible candidate regions $\mathcal{C}_s$ from part masks, semantic keypoints, affordance regions, or room-level occupancy/BEV maps. For example, functional grasping selects grasp-for-use regions such as mug handles, articulation selects handles or movable parts, and navigation selects traversable free-space regions around the target object. We then sample concrete anchor targets from $\mathcal{C}_s$ using farthest-point sampling on object surfaces or grid-based sampling on scene maps. Depending on the primitive, a sampled target can be a contact point, surface patch, support edge, opening, articulated handle point, navigation goal, or interaction-ready base pose. The sampled targets are filtered by geometric and embodiment constraints, including surface normals, curvature, visibility, collision margins, reachability, and traversability, before being used to instantiate the candidate banks.

Trajectory generation. Given a retained target $x^{(i)}$ for skill s, we generate an initial trajectory or waypoint sequence

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

where $\mathcal{G}_s$ is selected by the skill's constraint source. **Object-property-conditioned** generators use static geometry or garment keypoints to produce grasp, dexterous grasp, fling, or fold templates. **Object-trajectory-conditioned** generators follow articulated part motion, e.g., preserving the end-effector–handle relation while interpolating a door or drawer joint. **Object-vector-conditioned** generators align actions with insertion, hanging, pouring, or placement vectors. **Scene-trajectory-conditioned** generators use occupancy or BEV maps to sample interaction-ready base poses and compute coarse navigation paths with A*, connecting free-space regions to manipulation targets. These trajectories serve as template-level seeds for subsequent optimization and validation.

Trajectory optimization. The template trajectories are further refined by optimizing action parameters and waypoint sequences under contact, geometry, and embodiment constraints:

$$(\theta_*^{(i)}, \tau_*^{(i)}) = \arg\min_{\theta,\tau} E_{\text{task}} + E_{\text{contact}} + E_{\text{coll}} + E_{\text{kin}} + E_{\text{smooth}}.$$

Here, E_{task} encourages the trajectory to preserve the intended functional affordance, E_{contact} improves contact alignment and stability, E_{coll} penalizes penetration and unsafe clearance, E_{kin} enforces robot kinematics and embodiment constraints, and E_{smooth} regularizes waypoint motion. For dexterous grasping, we optimize hand pose, finger configuration, and contact assignment so that the selected contacts lie on the grounded functional region, satisfy joint limits and self-collision constraints, and can support task-relevant object wrenches. For adaptive skills, we adjust waypoints according to asset-specific geometry, e.g., scaling garment fold targets by sleeve length or hem-to-shoulder distance. For navigation, the global path from A* is locally optimized through DWB-style rollout under embodiment-specific dynamics, such as differential-drive, Ackermann, or holonomic motion constraints, producing feasible trajectories that respect clearance, turning radius, and manipulation reachability.

Physics validation. Optimized candidates are validated through batched physics simulation before being stored as action annotations. For most object-centric skills, we use floating embodiments, such as a floating parallel gripper or floating dexterous hand, to validate contact, collision, stability, and task progress independently of a specific robot arm or base pose. This decoupling avoids prematurely rejecting valid object interactions, since downstream data collection may randomize robot-object poses and embodiments. For strongly embodiment-dependent skills, such as garment pushing, washing, or retrieval from entangled configurations, we instead validate with the full manipulator to account for reachability, arm collision, and entanglement. To test robustness, we additionally apply disturbances and vary gravity magnitude or direction during validation, especially for grasp and dexterous grasp candidates. Accepted candidates must satisfy skill-specific success criteria and are stored with metadata such as success flags, collision statistics, contact stability, task progress, rollout length, disturbance setting, and validation embodiment. Because many generated candidates fail

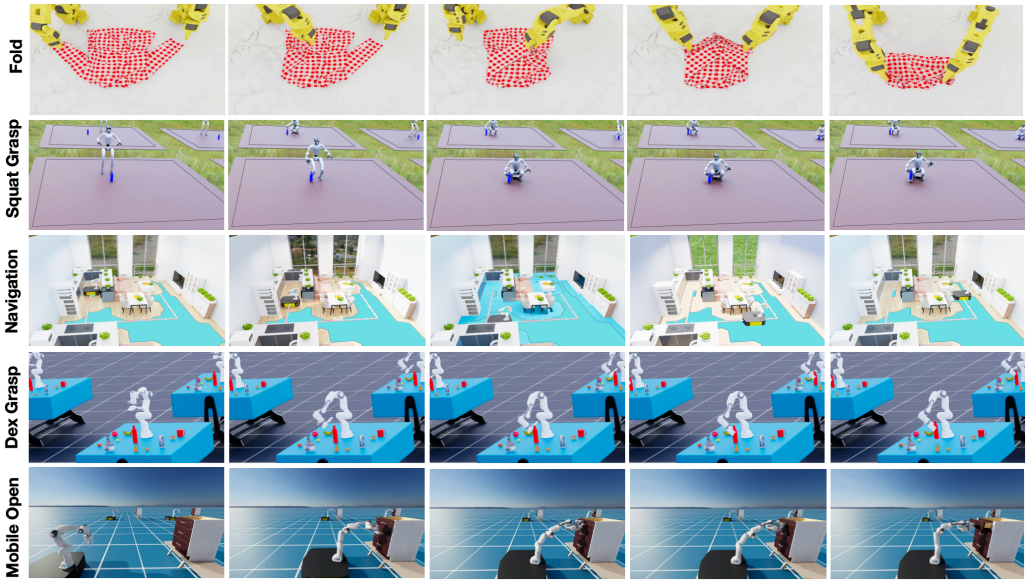

Figure 4: Atomic skill. Placeholder for the atomic-skill representation or taxonomy.

these checks, scalable parallel validation is critical for retaining high-quality labels and improving downstream success.

Physics-aware augmentation. To further increase diversity without breaking executability, we augment validated candidates in two ways. **Local perturbation** samples small variations around the validated anchor and action parameters while preserving the same functional affordance $\phi(h)$. For example, garment fold or fling points are jittered within local keypoint confidence regions, fold/place targets are perturbed within a bounded range, grasp candidates perturb contact centers, approach offsets, and in-plane rotations, and articulation candidates perturb handle contact points and pulling directions. These perturbations are bounded by part masks, affordance regions, collision margins, and embodiment constraints. **Symmetry-aware augmentation** uses symmetry priors from visual-language annotations to expand candidates on symmetric objects or parts. Let $\mathcal{G}_{\text{sym}}(o)$ denote the annotated symmetry group of object or part o. For a validated candidate $a_{h,s}^{(i)}$, we generate transformed candidates

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

by consistently transforming the target, pose parameters, and trajectory waypoints. For instance, grasp poses on a rotationally symmetric bottle can be rotated around its symmetry axis to produce additional feasible grasps. All augmented candidates are rechecked by lightweight geometric and physical constraints, and only feasible, affordance-preserving augmentations are retained in the candidate bank.

4 Downstream Tasks

AnnotateAnything turns passive 3D assets into reusable visual-language-action annotations for robot learning. As shown in Fig. 5, we study two downstream usages: using executable action annotations for large-scale simulation data collection, and converting the same annotations into supervision for affordance and keypoint detection, robot reasoning VQA, and 3D VLM instruction finetuning.

4.1 Large-scale Robot Data Collection

We use the generated action annotations as executable interfaces for large-scale simulation-based robot data collection. **This system is not the main contribution of this paper; we include it to demonstrate that AnnotateAnything annotations can be directly consumed by downstream robot execution and data-collection pipelines.**

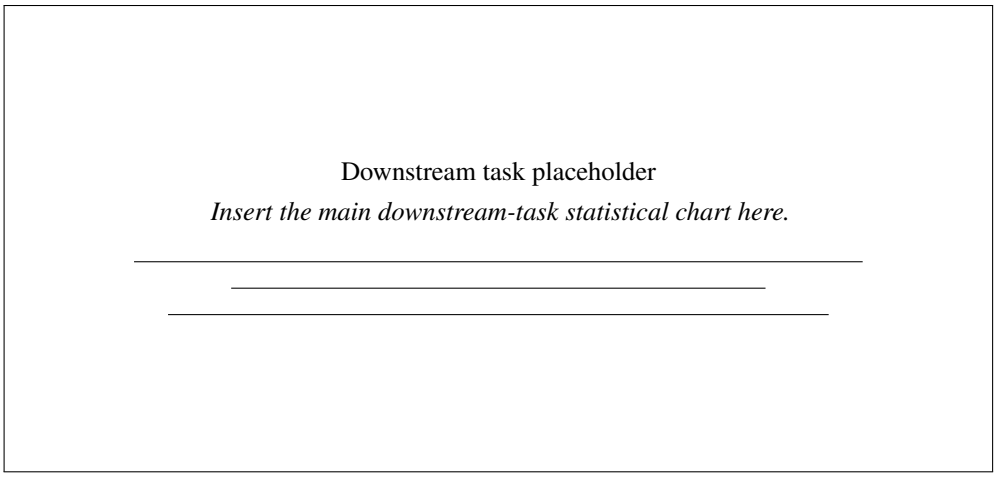

Figure 5: Downstream applications supported by AnnotateAnything. The same visual-language-action annotations can be converted into supervision for affordance and keypoint detection, robot reasoning VQA, and 3D VLM instruction finetuning.

Atomic skill interface. We build an atomic-skill library aligned with our annotation format, where skills consume action parameters such as grasp poses, target parts, waypoints, insertion directions, hanging anchors, and navigation goals. The library supports tabletop manipulation, bimanual manipulation, whole-body control, humanoids, dexterous hands, and mobile manipulation, and can compose atomic skills into long-horizon tasks.

Parallel rollout generation. We run heterogeneous parallel simulation environments, where each environment independently samples assets, tasks, robot embodiments, object poses, and scene layouts. For each rollout, asynchronous cuRobo-v2 planning is used for goal reaching and obstacle avoidance [36, 37], with domain randomization over pose, lighting, material, camera, and scene configuration.

Trajectory validation. For each randomized scene, we retrieve candidates from the annotation bank, solve goal-set IK, and select the feasible solution with the lowest planning cost. We also remove action candidates that repeatedly fail simulation validation, ensuring that the retained annotations remain executable under diverse robot-object configurations.

4.2 Annotation-derived Downstream Applications

4.2.1 Keypoint Generation and Affordance Grounding

We reuse visual annotations as keypoint supervision and physics-validated action annotations as affordance supervision. The former provides VLM-selected 3D functional anchors, while the latter provides executable interaction labels, such as grasp directions, articulation trajectories, garment pick-and-place points, and room-level navigation or interaction regions. These labels can be projected to images, point clouds, or BEV maps for robot-centric perception training.

4.3 Robot Reasoning VQA

Robot reasoning VQA is constructed as a by-product of trajectory generation. During rollout, we record execution-level ground truth, including the selected object, part anchor, skill type, IK goal-set solution, approach side, navigation target, articulation direction, and garment manipulation points. These labels depend on the runtime robot-object relative pose and the candidate selected from the annotation bank, and therefore cannot be obtained from static assets alone. We use this mainly to show that simulation rollouts generated from AnnotateAnything naturally provide grounded QA supervision for robot execution reasoning.

Table 3: Visual-language annotation quality on the audited evaluation suite. Human scores are reported on a 0–100 scale. Detailed rubrics and sampling procedures are provided in the appendix.

Method	Sem.	Ground.	Cover.	Action.	Overall
Direct VLM	XX.X	XX.X	XX.X	XX.X	XX.X
w/o 3D refine	XX.X	XX.X	XX.X	XX.X	XX.X
Ours	**XX.X**	**XX.X**	**XX.X**	**XX.X**	**XX.X**

4.4 3D VLM Instruction-tuning Data

AnnotateAnything can also provide instruction-tuning data for 3D VLMs. In simulation, assets and rollouts provide dense grounding signals such as 3D bounding boxes, projected 2D boxes, instance masks, object poses, and part identities, which can be converted into grounding-style instruction-response pairs. Meanwhile, the language annotations and cross-level composition in Sec. 3.1.3 provide object relations, room layouts, and task contexts, enabling QA pairs about 3D spatial relationships, object localization, and scene-level reasoning. We use this mainly as a lightweight downstream demonstration that our annotations can be repurposed into multimodal instruction data, rather than as a full 3D VLM benchmark.

5 Experiment

5.1 Experimental Setup

Evaluation scope. We evaluate AnnotateAnything as an automatic annotation pipeline rather than as a dataset release or a standalone data-generation system. Given heterogeneous raw 3D assets, our experiments test whether the pipeline can convert passive geometry into manipulation-ready language, visual, and action annotations that are semantically meaningful, 3D-grounded, physically valid, and useful for downstream simulation rollouts. We evaluate four aspects: annotation scale and conversion statistics, visual-language annotation quality, physics-grounded action annotation quality, and annotation-enabled data collection.

Audited evaluation suite. All quality and success-rate results are reported on an audited evaluation suite sampled from a larger heterogeneous asset pool. The pool spans multiple asset sources and asset types, including rigid objects, articulated objects, deformable or garment assets, and room-scale scenes. Since our goal is annotation conversion rather than curated asset release, we report source and scale statistics separately, while pass rates, success rates, and human scores are computed on the audited suite. The suite is stratified by valid asset–skill pairs, covering grasping, dexterous grasping, bimanual grasping, bimanual dexterous grasping, articulation, insertion, hanging, deformable manipulation, and navigation or approach-target generation.

Evaluation protocol. Each experiment defines its metrics at the beginning of the corresponding subsection. Human evaluation is reported as an overall 0–100 score in the main paper, with detailed rubrics, sampling procedures, evaluator instructions, and per-dimension scores provided in the supplementary material.

5.2 Annotation Coverage over Heterogeneous Assets

5.3 Visual-Language Annotation Quality

We evaluate whether the visual-language stage produces annotations that are semantically correct, accurately grounded in 3D, and useful for subsequent action annotation. We compare **Direct VLM**, which directly predicts language and interaction regions from multi-view renderings; **w/o 3D refinement**, which keeps hierarchical language reasoning but removes explicit 3D grounding refinement; and **Ours**, the full visual-language annotation pipeline. Human raters score sampled annotation bundles from the audited evaluation suite on a 0–100 scale. Table 3 reports semantic correctness, 3D grounding accuracy, coverage of interaction-relevant regions, actionability, and the overall score. Fig. **??** qualitatively shows how the predicted semantic priors are grounded into keypoints, parts, affordance regions, and scene-level spatial cues.

Table 4: Physics-grounded action annotation quality on the audited evaluation suite. The upper block reports results by action family, and the lower block reports macro-level ablations. Human scores are reported on a 0–100 scale. Full per-skill results, validation criteria, and human-evaluation rubrics are provided in the appendix.

Setting	Phys. Pass	Asset Ready	Valid/Ready	Exec. Succ.	Ann./Hour	Human
Action-family results						
Grasping	XX.X	XX.X	XX.X	XX.X	XX.X	XX.X
Dexterous / Bimanual	XX.X	XX.X	XX.X	XX.X	XX.X	XX.X
Articulation	XX.X	XX.X	XX.X	XX.X	XX.X	XX.X
Insertion / Hanging	XX.X	XX.X	XX.X	XX.X	XX.X	XX.X
Deformable	XX.X	XX.X	XX.X	XX.X	XX.X	XX.X
Navigation / Approach	XX.X	XX.X	XX.X	XX.X	XX.X	XX.X
Macro ablations						
Geometry-only	XX.X	XX.X	XX.X	XX.X	XX.X	XX.X
VL-only	XX.X	XX.X	XX.X	XX.X	XX.X	XX.X
w/o physics validation	–	XX.X	XX.X	XX.X	XX.X	XX.X
Ours full	**XX.X**	**XX.X**	**XX.X**	**XX.X**	**XX.X**	**XX.X**

Ours achieves the highest overall score. Direct VLM prompting often produces plausible language descriptions but is less reliable at localizing interaction regions in 3D. Removing 3D refinement reduces grounding accuracy and actionability, showing that language reasoning alone is insufficient for manipulation-ready annotation. These results indicate that the visual-language pipeline provides reliable human-prior guidance for the physics annotation stage.

5.4 Physics-grounded Action Annotation Quality

We next evaluate whether grounded visual-language priors can be converted into executable action annotations. For each valid asset–skill pair in the audited evaluation suite, the physics pipeline generates candidates, filters them by geometry and embodiment constraints, optimizes poses or trajectories, and validates them in simulation. Table 4 summarizes the resulting annotation quality. The upper block reports results by action family; the lower block reports macro-level ablations comparing **Geometry-only**, **VL-only**, **w/o physics validation**, and **Ours**. We report physics validation pass rate, asset readiness, valid annotations per ready asset–skill pair, execution success, annotation throughput, and human score. Fig. **??** illustrates the final action labels produced by the physics pipeline.

The full pipeline produces validated annotations across all action families while maintaining high asset readiness and execution success. The ablations show that geometry-only sampling lacks semantic precision, VL-only annotation lacks physical executability, and removing physics validation reduces downstream execution reliability. These results demonstrate that AnnotateAnything converts human-prior visual-language cues into diverse, asset-specific, and physically executable action annotation banks.

6 Conclusion

Do not change any aspects of the formatting parameters in the style files. In particular, do not modify the width or length of the rectangle the text should fit into, and do not change font sizes. Please note that pages should be numbered.

Most of the margin problems come from figures positioned by hand using `\special` or other commands. We suggest using the command `\includegraphics` from the `graphicx` package. Always specify the figure width as a multiple of the line width.

A number of width problems arise when LaTeX cannot properly hyphenate a line. Please give LaTeX hyphenation hints using the `\-` command when necessary.

References

[1] Physical Intelligence. Pi-0.7: A steerable generalist robotic foundation model with emergent capabilities. arXiv preprint, 2026. CorpusID: 287607456.

[2] NVIDIA. Gr00t n1: An open foundation model for generalist humanoid robots. arXiv:2503.14734, 2025.

[3] Generalist AI Team. Gen-0: Embodied foundation models that scale with physical interaction. Generalist AI Blog, 2025. November 4, 2025.

[4] Seonghyeon Ye, Yunhao Ge, Kaiyuan Zheng, and Joel Jang. World action models are zero-shot policies. arXiv:2602.15922, 2026.

[5] Ruijie Zheng, Dantong Niu, Yuqi Xie, and Linxi Jim Fan. Egoscale: Scaling dexterous manipulation with diverse egocentric human data. arXiv:2602.16710, 2026.

[6] 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 Rich Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky. π0.5: a vision-language-action model with open-world generalization. *ArXiv*, abs/2504.16054, 2025. URL `https://api.semanticscholar.org/CorpusID:277993634`.

[7] 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, 2026. URL `https://arxiv.org/abs/2602.14193`.

[8] 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)*, pages 13234–13241, 2025. doi: 10.1109/ICRA55743.2025.11128651.

[9] Yan Shen, Ruihai Wu, Yubin Ke, Xinyuan Song, Zeyi Li, Xiaoqi Li, Hongwei Fan, Haoran Lu, and Hao Dong. Biassemble: Learning collaborative affordance for bimanual geometric assembly. *ArXiv*, abs/2506.06221, 2025. URL `https://api.semanticscholar.org/CorpusID:279244917`.

[10] 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, 2024. URL `https://api.semanticscholar.org/CorpusID:270226492`.

[11] 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. Preprint, 2026. CorpusID: 286238271.

[12] Ruihai Wu, Haoran Lu, Yiyan Wang, Yubo Wang, and Hao Dong. Unigarmentmanip: A unified framework for category-level garment manipulation via dense visual correspondence. *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16340–16350, 2024. URL `https://api.semanticscholar.org/CorpusID:269757227`.

[13] Open X-Embodiment Collaboration, Abby O'Neill, Abdul Rehman, Abhinav Gupta, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, Albert Tung, Alex Bewley, Alex Herzog, Alex Irpan, Alexander Khazatsky, Anant Rai, Anchit Gupta, Andrew Wang, Andrey Kolobov, Anikait Singh, Animesh Garg, Aniruddha Kembhavi, Annie Xie, Anthony Brohan, Antonin Raffin, Archit Sharma, Arefeh Yavary, Arhan Jain, Ashwin Balakrishna, Ayzaan Wahid, Ben Burgess-Limerick, Beomjoon Kim, Bernhard Schölkopf, Blake Wulfe, Brian Ichter, Cewu Lu, Charles Xu, Charlotte Le, Chelsea Finn, Chen Wang, Chenfeng Xu, Cheng Chi, Chenguang Huang, Christine Chan, Christopher Agia, Chuer Pan, Chuyuan Fu, Coline Devin, Danfei Xu, Daniel

Morton, Danny Driess, Daphne Chen, Deepak Pathak, Dhruv Shah, Dieter Büchler, Dinesh Jayaraman, Dmitry Kalashnikov, Dorsa Sadigh, Edward Johns, Ethan Foster, Fangchen Liu, Federico Ceola, Fei Xia, Feiyu Zhao, Felipe Vieira Frujeri, Freek Stulp, Gaoyue Zhou, Gaurav S. Sukhatme, Gautam Salhotra, Ge Yan, Gilbert Feng, Giulio Schiavi, Glen Berseth, Gregory Kahn, Guangwen Yang, Guanzhi Wang, Hao Su, Hao-Shu Fang, Haochen Shi, Henghui Bao, Heni Ben Amor, Henrik I Christensen, Hiroki Furuta, Homanga Bharadhwaj, Homer Walke, Hongjie Fang, Huy Ha, Igor Mordatch, Ilija Radosavovic, Isabel Leal, Jacky Liang, Jad Abou-Chakra, Jaehyung Kim, Jaimyn Drake, Jan Peters, Jan Schneider, Jasmine Hsu, Jay Vakil, Jeannette Bohg, Jeffrey Bingham, Jeffrey Wu, Jensen Gao, Jiaheng Hu, Jiajun Wu, Jialin Wu, Jiankai Sun, Jianlan Luo, Jiayuan Gu, Jie Tan, Jihoon Oh, Jimmy Wu, Jingpei Lu, Jingyun Yang, Jitendra Malik, João Silvério, Joey Hejna, Jonathan Booher, Jonathan Tompson, Jonathan Yang, Jordi Salvador, Joseph J. Lim, Junhyek Han, Kaiyuan Wang, Kanishka Rao, Karl Pertsch, Karol Hausman, Keegan Go, Keerthana Gopalakrishnan, Ken Goldberg, Kendra Byrne, Kenneth Oslund, Kento Kawaharazuka, Kevin Black, Kevin Lin, Kevin Zhang, Kiana Ehsani, Kiran Lekkala, Kirsty Ellis, Krishan Rana, Krishnan Srinivasan, Kuan Fang, Kunal Pratap Singh, Kuo-Hao Zeng, Kyle Hatch, Kyle Hsu, Laurent Itti, Lawrence Yunliang Chen, Lerrel Pinto, Li Fei-Fei, Liam Tan, Linxi "Jim" Fan, Lionel Ott, Lisa Lee, Luca Weihs, Magnum Chen, Marion Lepert, Marius Memmel, Masayoshi Tomizuka, Masha Itkina, Mateo Guaman Castro, Max Spero, Maximilian Du, Michael Ahn, Michael C. Yip, Mingtong Zhang, Mingyu Ding, Minho Heo, Mohan Kumar Srirama, Mohit Sharma, Moo Jin Kim, Muhammad Zubair Irshad, Naoaki Kanazawa, Nicklas Hansen, Nicolas Heess, Nikhil J Joshi, Niko Suenderhauf, Ning Liu, Norman Di Palo, Nur Muhammad Mahi Shafiullah, Oier Mees, Oliver Kroemer, Osbert Bastani, Pannag R Sanketi, Patrick "Tree" Miller, Patrick Yin, Paul Wohlhart, Peng Xu, Peter David Fagan, Peter Mitrano, Pierre Sermanet, Pieter Abbeel, Priya Sundaresan, Qiuyu Chen, Quan Vuong, Rafael Rafailov, Ran Tian, Ria Doshi, Roberto Mart'in-Mart'in, Rohan Baijal, Rosario Scalise, Rose Hendrix, Roy Lin, Runjia Qian, Ruohan Zhang, Russell Mendonca, Rutav Shah, Ryan Hoque, Ryan Julian, Samuel Bustamante, Sean Kirmani, Sergey Levine, Shan Lin, Sherry Moore, Shikhar Bahl, Shivin Dass, Shubham Sonawani, Shubham Tulsiani, Shuran Song, Sichun Xu, Siddhant Haldar, Siddharth Karamcheti, Simeon Adebola, Simon Guist, Soroush Nasiriany, Stefan Schaal, Stefan Welker, Stephen Tian, Subramanian Ramamoorthy, Sudeep Dasari, Suneel Belkhale, Sungjae Park, Suraj Nair, Suvir Mirchandani, Takayuki Osa, Tanmay Gupta, Tatsuya Harada, Tatsuya Matsushima, Ted Xiao, Thomas Kollar, Tianhe Yu, Tianli Ding, Todor Davchev, Tony Z. Zhao, Travis Armstrong, Trevor Darrell, Trinity Chung, Vidhi Jain, Vikash Kumar, Vincent Vanhoucke, Vitor Guizilini, Wei Zhan, Wenxuan Zhou, Wolfram Burgard, Xi Chen, Xiangyu Chen, Xiaolong Wang, Xinghao Zhu, Xinyang Geng, Xiyuan Liu, Xu Liangwei, Xuanlin Li, Yansong Pang, Yao Lu, Yecheng Jason Ma, Yejin Kim, Yevgen Chebotar, Yifan Zhou, Yifeng Zhu, Yilin Wu, Ying Xu, Yixuan Wang, Yonatan Bisk, Yongqiang Dou, Yoonyoung Cho, Youngwoon Lee, Yuchen Cui, Yue Cao, Yueh-Hua Wu, Yujin Tang, Yuke Zhu, Yunchu Zhang, Yunfan Jiang, Yunshuang Li, Yunzhu Li, Yusuke Iwasawa, Yutaka Matsuo, Zehan Ma, Zhuo Xu, Zichen Jeff Cui, Zichen Zhang, Zipeng Fu, and Zipeng Lin. Open X-Embodiment: Robotic learning datasets and RT-X models. `https://arxiv.org/abs/2310.08864`, 2023.

[14] Alexander Khazatsky, Karl Pertsch, Suraj Nair, et al. Droid: A large-scale in-the-wild robot manipulation dataset. arXiv:2403.12945, 2024.

[15] Modi Shi, Yuxiang Lu, Huijie Wang, and Shaoze Yang. Open-sourcing go-1: The bitter lessons of building vla systems at scale. `https://opendrivelab.com/OpenGO1/`, September 2025. Blog post.

[16] Modi Shi, Yuxiang Lu, Huijie Wang, Chengen Xie, and Qingwen Bu. Introducing agibot world colosseo: A large-scale manipulation platform for scalable and intelligent embodied systems. `https://opendrivelab.com/AgiBot-World/`, March 2025. Blog post.

[17] 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*, abs/2512.23864, 2025. URL `https://api.semanticscholar.org/CorpusID:284350273`.

[18] Haoran Geng, Feishi Wang, Songlin Wei, Yuyang Li, Bangjun Wang, Boshi An, Charlie Tianyue Cheng, Haozhe Lou, Peihao Li, Yen-Jen Wang, Yutong Liang, Dylan Goetting, Chaoyi Xu, Haozhe Chen, Yuxi Qian, Yiran Geng, Jiageng Mao, Weikang Wan, Mingtong Zhang, Jiangran Lyu, Siheng Zhao, Jiazhao Zhang, Jialiang Zhang, Chengyang Zhao, Haoran Lu, Yufei

Ding, Ran Gong, Yuran Wang, Yuxuan Kuang, Ruihai Wu, Baoxiong Jia, Carlo Sferrazza, Hao Dong, Siyuan Huang, Yue Wang, Jitendra Malik, and Pieter Abbeel. Roboverse: Towards a unified platform, dataset and benchmark for scalable and generalizable robot learning. *ArXiv*, abs/2504.18904, 2025. URL https://api.semanticscholar.org/CorpusID:277741767.

[19] Haoran Lu, Yitong Li, Ruihai Wu, Chuanruo Ning, Yan Shen, and Hao Dong. Unigarment: A unified simulation and benchmark for garment manipulation. Preprint, 2024. CorpusID: 275782214.

[20] 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, 2024. URL https://api.semanticscholar.org/CorpusID:273811186.

[21] Lightwheel. Lightwheel kitchen: 3d kitchen asset collection for nvidia isaac sim. GitHub repository, 2025. URL https://github.com/LightwheelAI/Lightwheel_Kitchen. Open-source 3D assets under CC BY-NC 4.0.

[22] 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.

[23] Lightwheel Team. Lw-benchhub: Lightwheel's end-to-end embodied ai simulation platform. GitHub repository, 2025. URL https://github.com/lightwheel-ai/lw_benchhub.

[24] Chengshu Li, Ruohan Zhang, Josiah Wong, Cem Gokmen, Sanjana Srivastava, Roberto Martín-Martín, Chen Wang, Gabrael Levine, Wensi Ai, Benjamin Jose Martinez, Hang Yin, Michael Lingelbach, Minjune Hwang, Ayano Hiranaka, Sujay Garlanka, Arman Aydin, Sharon Lee, Jiankai Sun, Mona Anvari, Manasi Sharma, Dhruva Bansal, Samuel Hunter, Kyu-Young Kim, Alan Lou, Caleb R. Matthews, Ivan Villa-Renteria, Jerry Huayang Tang, Claire Tang, Fei Xia, Yunzhu Li, Silvio Savarese, Hyowon Gweon, C. Karen Liu, Jiajun Wu, Fei-Fei Li, and Salesforce AI Research. Behavior-1k: A human-centered, embodied ai benchmark with 1, 000 everyday activities and realistic simulation. *ArXiv*, abs/2403.09227, 2024. URL https://api.semanticscholar.org/CorpusID:268520018.

[25] 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, 2025. URL https://arxiv.org/abs/2512.08580.

[26] Delin Qu, Haoming Song, Qizhi Chen, Yuanqi Yao, Xinyi Ye, Yani Ding, Zhigang Wang, Jiayuan Gu, Bin Zhao, Dong Wang, and Xuelong Li. Spatialvla: Exploring spatial representations for visual-language-action model. *ArXiv*, abs/2501.15830, 2025. URL https://api.semanticscholar.org/CorpusID:275921131.

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

[28] Abhay Deshpande, Maya Guru, Rose Hendrix, Snehal Jauhri, Ainaz Eftekhar, Rohun Tripathi, Max Argus, Jordi Salvador, Haoquan Fang, Matthew Wallingford, Wilbert Pumacay, Yejin Kim, Quinn Pfeifer, Ying-Chun Lee, Piper Wolters, Omar Rayyan, Mingtong Zhang, Jiafei Duan, Karen Farley, Winson Han, Eli Vanderbilt, Dieter Fox, Ali Farhadi, Georgia Chalvatzaki, Dhruv Shah, and Ranjay Krishna. Molmob0t: Large-scale simulation enables zero-shot manipulation, 2026. URL https://arxiv.org/abs/2603.16861.

[29] Kaixuan Wang, Tianxing Chen, Jiawei Liu, Honghao Su, Shaolong Zhu, Minxuan Wang, Zixuan Li, Yue Chen, Huan-ang Gao, Yusen Qin, Jiawei Wang, Qixuan Zhang, Lan Xu, Jingyi Yu, Yao Mu, and Ping Luo. Manitwin: Scaling data-generation-ready digital object dataset to 100k. Preprint, 2026. CorpusID: 286583937.

[30] 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*, 2026.

[31] Tianxing Chen, Zanxin Chen, Baijun Chen, Zijian Cai, Yibin Liu, Qiwei Liang, Zixuan Li, XianLian Lin, Yiheng Ge, Zhenyu Gu, Weiliang Deng, Yubin Guo, Tian Nian, Xuanbing Xie, Qiangyu Chen, Kailun Su, Tianling Xu, Guodong Liu, Mengkang Hu, Huan ang Gao, Kaixuan Wang, Zhixuan Liang, Yusen Qin, Xiaokang Yang, Ping Luo, and Yao Mu. Robotwin 2.0: A scalable data generator and benchmark with strong domain randomization for robust bimanual robotic manipulation. *ArXiv*, abs/2506.18088, 2025. URL https://api.semanticscholar.org/CorpusID:279999539.

[32] 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, 2026. URL https://api.semanticscholar.org/CorpusID:286222901.

[33] Yang Tian, Yuyin Yang, Yiman Xie, Zetao Cai, Xu Shi, Ning Gao, Hangxu Liu, Xuekun Jiang, Zherui Qiu, Feng Yuan, Yaping Li, Ping Wang, Junhao Cai, Jia Zeng, Hao Dong, and Jiangmiao Pang. Interndata-a1: Pioneering high-fidelity synthetic data for pre-training generalist policy. *ArXiv*, abs/2511.16651, 2025. URL https://api.semanticscholar.org/CorpusID:283110021.

[34] 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, 2023.

[35] Wenbo Wang, Fangyun Wei, QiXiu Li, Xi Chen, Yaobo Liang, Chang Xu, Jiaolong Yang, and Baining Guo. Mobilemanibench: Simplifying model verification for mobile manipulation. *arXiv preprint arXiv:2602.05233*, 2026.

[36] Balakumar Sundaralingam, Adithyavairavan Murali, and Stan Birchfield. curobov2: Dynamics-aware motion generation with depth-fused distance fields for high-dof robots, 2026.

[37] Balakumar Sundaralingam, Siva Kumar Sastry Hari, Adam Fishman, Caelan Garrett, Karl Van Wyk, Valts Blukis, Alexander Millane, Helen Oleynikova, Ankur Handa, Fabio Ramos, Nathan Ratliff, and Dieter Fox. curobo: Parallelized collision-free minimum-jerk robot motion generation, 2023. URL https://arxiv.org/abs/2310.17274.

[38] Jiayi Chen, Yubin Ke, and He Wang. Bodex: Scalable and efficient robotic dexterous grasp synthesis using bilevel optimization, 2025. URL https://arxiv.org/abs/2412.16490.

[39] 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.

[40] 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.

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

[42] Ruoxi Shi, Zhengrong Xue, Yang You, and Cewu Lu. Skeleton merger: an unsupervised aligned keypoint detector. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 43–52, 2021.

[43] 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.

[44] InternData-N1 Dataset contributors. Interndata-n1 dataset. https://huggingface.co/datasets/InternRobotics/InternData-N1, 2025.

[45] Abhay Deshpande, Maya Guru, Rose Hendrix, Snehal Jauhri, Haoquan Fang, Wilbert Pumacay, Yejin Kim, Quinn Pfeifer, Ying-Chun Lee, Piper Wolters, Omar Rayyan, Mingtong Zhang, Karen Farley, Winson Han, Eli Vanderbilt, Dieter Fox, Ali Farhadi, Georgia Chalvatzaki, Dhruv Shah, and Ranjay Krishna. Molmob0t: Large-scale simulation enables zero-shot manipulation. Preprint, 2026. CorpusID: 286498371.

[46] 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*, pages 12187–12198, 2025.

[47] Mingjie Pan, Jiyao Zhang, Tianshu Wu, Yinghao Zhao, Wenlong Gao, and Hao Dong. Omnimanip: Towards general robotic manipulation via object-centric interaction primitives as spatial constraints. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 17359–17369, June 2025.

[48] Chenghao Yin, Da Huang, Di Yang, Jichao Wang, Nanshu Zhao, Chen Xu, Wenjun Sun, Linjie Hou, Zhijun Li, Junhui Wu, Zhaobo Liu, Zhen Xiao, Sheng Zhang, Lei Bao, Rui Feng, Zhenquan Pang, Jiayu Li, Qian Wang, and Maoqing Yao. Genie sim 3.0 : A high-fidelity comprehensive simulation platform for humanoid robot, 2026. URL https://arxiv.org/abs/2601.02078.

[49] 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)*, pages 27649–27660, June 2025.

[50] 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.

[51] 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*, pages 3891–3902, 2023.

[52] 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. *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11359–11366, 2022. URL https://api.semanticscholar.org/CorpusID:252734719.

[53] 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*, pages 21190–21200, 2023.

[54] 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.

[55] 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.

[56] Qwen Team. Qwen3.5-omni technical report, 2026. URL https://arxiv.org/abs/2604.15804.

[57] 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.

[58] 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.

[59] 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.

[60] 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*, pages 1820–1864. PMLR, 2023. URL https://proceedings.mlr.press/v229/mandlekar23a.html.

[61] 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. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 51936–51983. PMLR, 2024. URL https://proceedings.mlr.press/v235/wang24cc.html.

[62] 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.

[63] 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.

[64] 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.

[65] 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.

[66] 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.

[67] 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.

[68] 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*, pages 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.

[69] 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.

[70] 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)*, pages 11359–11366. IEEE, 2023. URL https://arxiv.org/abs/2210.02697.

[71] 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.

[72] 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*, 2021. URL https://arxiv.org/abs/2101.02692.

[73] 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.

[74] 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*, pages 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.

[75] 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*, 2021. URL https://openreview.net/forum?id=EougVeukEH9.

[76] 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.

[77] 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.

[78] 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*, pages 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.

[79] 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.

[80] Kaichun Mo, Shilin Zhu, Angel X. Chang, Li Yi, Subarna Tripathi, Leonidas J. Guibas, and Hao Su. Partnet: A large-scale benchmark for fine-grained and hierarchical part-level 3d object understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 909–918, 2019. 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.

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

[82] 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*, pages 11097–11107, 2020. URL https://openaccess.thecvf.com/content_CVPR_2020/html/Xiang_SAPIEN_A_SimulAted_Part-Based_Interactive_ENvironment_CVPR_2020_paper.html.

[83] 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*, pages 13142–13153, 2023. URL https://openaccess.thecvf.com/content/CVPR2023/html/Deitke_Objaverse_A_Universe_of_Annotated_3D_Objects_CVPR_2023_paper.html.

[84] 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.

[85] 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.

[86] 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*, pages 5026–5033. IEEE, 2012. doi: 10.1109/IROS.2012.6386109.

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

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

[89] 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.

[90] 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.

[91] 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.

[92] 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.

[93] 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.

[94] 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.

[95] 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.

[96] 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*, pages 1300–1315. PMLR, 2023. URL https://proceedings.mlr.press/v229/liu23e.html.

A Language Annotation Pipeline

A.1 Asset-level Language Annotation Details

Automatic multi-view rendering. For each object-level asset, we automatically render multi-view observations before querying the VLM. We first compute the asset's axis-aligned bounding box and use its center and diagonal length to normalize camera placement. Specifically, let $\mathbf{c}$ denote the bounding-box center and d denote the diagonal length. We place eight cameras on a horizontal circle around the object with evenly spaced azimuth angles $\{0°, 45°, 90°, 135°, 180°, 225°, 270°, 315°\}$, and set the camera distance proportional to d so that the object is centered and consistently scaled in every rendered image. All cameras look at $\mathbf{c}$ and render RGB observations in simulation. We use eight views by default because they provide sufficient coverage of most object geometries while remaining close to the practical multi-image input budget for stable VLM reasoning in our implementation.

Global hierarchical description. Given the eight rendered views, we prompt Qwen3.5-VL [56] to generate asset-level language annotations at three granularities: a phrase-level label, a sentence-level description, and a paragraph-level description. The phrase-level label provides a compact semantic index of the object, the sentence-level description summarizes its main function and major manipulable components, and the paragraph-level description provides dense interaction reasoning about object parts, affordances, feasible actions, and manipulation constraints.

Part-aware dense description. To make the dense paragraph grounded in object structure, we further query the VLM with part-specific masked renderings. For each segmented part, we render an image where the target part is highlighted while the remaining object is kept as context. The VLM is then asked to describe only the highlighted part, including its part name, visual evidence, function, feasible robot interactions, and manipulation constraints. Each part-level description is associated with its segmentation ID, which allows the final dense language annotation to be traced back to concrete visual regions rather than free-form text alone.

Output format and filtering. We require the VLM to produce a structured JSON output. To reduce hallucination, the prompt explicitly asks the model to describe only visible evidence, avoid unsupported assumptions, and mark uncertain attributes as `unknown`. We discard outputs that cannot be parsed or whose part descriptions cannot be matched to a valid segmentation ID. The resulting language annotations are used as semantic priors for visual grounding and physics-based action generation.

Prompt for global asset-level language annotation. We use the following prompt to generate phrase-, sentence-, and paragraph-level asset descriptions from the eight rendered views.

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.
```

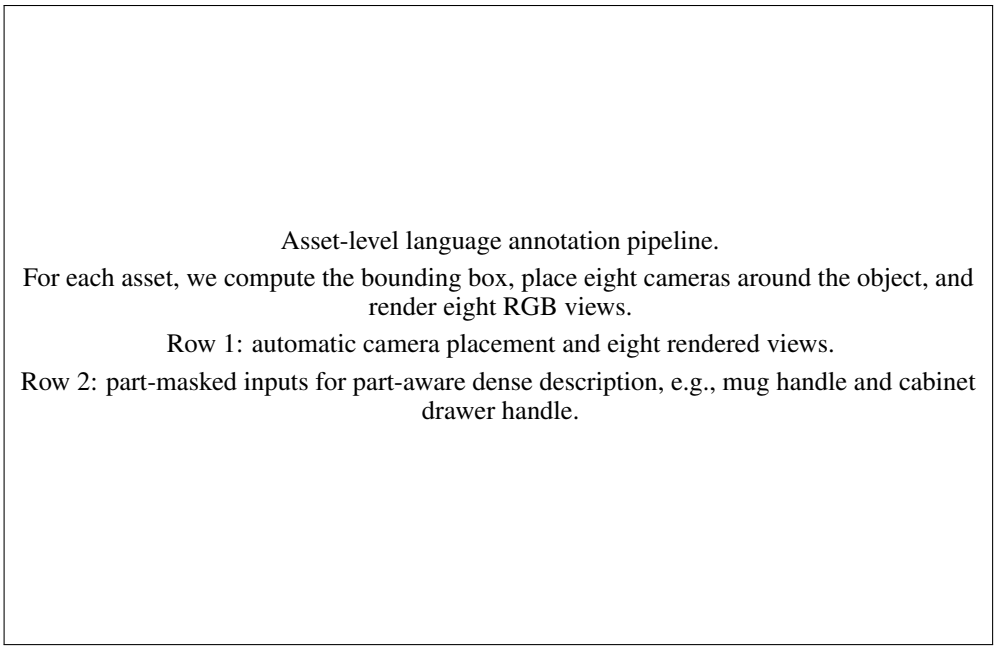

Figure 6: Supplementary illustration of asset-level language annotation. We automatically normalize camera placement using the object bounding box and render eight surrounding views for global asset-level description. For paragraph-level dense reasoning, segmented parts are individually highlighted and fed to the VLM, enabling part-aware descriptions grounded in visual evidence.

```
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.
```

Prompt for part-aware dense description. For paragraph-level annotation, we further query the VLM with masked part renderings.

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: mug.

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.

Example output: cabinet.

Global asset-level output: cabinet

Phrase-level label: Three-drawer cabinet.
Sentence-level description: A wooden cabinet with stacked front drawers, horizontal handles, side panels, and an open top cavity for storage and drawer manipulation.
Paragraph-level description: The object is a cabinet with multiple stacked drawer fronts, horizontal metal handles, side panels, an open top cavity, and a supporting base. The drawer fronts are likely movable storage components that translate outward when pulled. The handles are the primary interaction parts for grasping and pulling the drawers. The side panels and frame provide rigid structural support and should generally be treated as non-manipulable surfaces. The open top cavity can be used for placing or retrieving objects, while the base supports the cabinet on the floor.

Part-aware output: cabinet drawer handle

Highlighted part: Drawer handle.
The highlighted part is a drawer handle on the front of the cabinet. It is an elongated horizontal bar attached to a drawer panel and is designed as the main interaction target for opening the drawer. Feasible robot interactions include parallel-jaw grasping, dexterous hooking, and pulling along the drawer's translation direction. The robot should approach the handle from the front, avoid collisions with neighboring drawer panels, and keep the pulling motion aligned with the drawer direction.

A.2 Room-level Language Annotation Details

Automatic room-view rendering. For each room-scale scene, we generate room-level language annotations from both egocentric room views and a top-view spatial representation. We first decompose the scene into individual rooms and randomly sample camera poses inside each room. Since uniformly sampled cameras may face walls or empty regions, we use simulator ground-truth segmentation as an automatic view-quality signal. For each candidate camera pose, we render an RGB image and its segmentation map, count the number of visible non-structural object instances, and reject views with too few visible objects or views dominated by walls and floors. We keep a fixed number of valid views, eight in our implementation, with diverse positions and viewing directions. This produces informative room-level observations without manual camera placement.

Top-view layout input. In addition to the selected RGB views, we provide the VLM with a top-view occupancy map of the room or scene. The map marks free space, obstacles, room boundaries,

and object instances with their corresponding labels. The RGB views provide local appearance and object-level evidence, while the labeled occupancy map provides global layout, spatial relations, and navigable structure. This combination allows the VLM to reason about both what objects are present and how they are arranged in the room.

Hierarchical room-level description. Given the selected room views and the labeled top-view occupancy map, we prompt Qwen3.5-VL [56] to generate room-level language annotations at three granularities. The phrase-level annotation summarizes the room type or global layout pattern. The sentence-level annotation describes the major furniture, functional zones, and spatial arrangement. The paragraph-level annotation provides dense scene-level reasoning, including object relations, navigable regions, interaction-relevant areas, and possible long-horizon robot tasks. These descriptions complement asset-level annotations by explaining how individual objects compose into a functional scene.

Output format and filtering. We require structured JSON outputs and explicitly instruct the VLM to use the labeled occupancy map for global layout reasoning and the RGB views for local visual evidence. To reduce hallucination, the prompt asks the model to avoid unsupported object names or hidden room functions and to mark uncertain attributes as `unknown`. We discard outputs that cannot be parsed or whose object references cannot be matched to the labeled occupancy map.

Prompt for room-level language annotation. We use the following prompt to generate short-sentence, long-sentence, and paragraph-level room descriptions from selected room views and the labeled top-view occupancy map.

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: multi-room scene.

Room-level output: multi-room furnished apartment

Short-sentence description: A multi-room furnished apartment with connected rooms and open navigation corridors.

Long-sentence description: The scene contains several enclosed rooms arranged around central open spaces and corridors, with furniture mostly placed near walls. Major interaction areas include tables, cabinets, shelves, beds or sofas, and small objects on support surfaces.
Paragraph-level description: The scene contains multiple connected rooms with corridors, open passages, and several functional zones. The top-view layout shows central open areas surrounded by smaller rooms, with furniture arranged mostly along walls and corners. Visible interaction-relevant objects include tables, cabinets, shelves, beds or sofas, lamps, and small tabletop items. The main navigable regions are the open floor areas, corridors, and doorway passages, while dense furniture clusters and narrow passages should be treated as collision-sensitive regions. A robot can perform long-horizon tasks such as navigating between rooms, approaching furniture, retrieving objects from tables or shelves, placing objects on support surfaces, and interacting with storage furniture.

Example output: dining room.

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.

B Visual Annotation Pipeline

B.1 Asset-level Visual Annotation Details

Multi-view point-cloud reconstruction. For each object-level asset, we construct the input point cloud from multi-view RGB-D observations rather than directly sampling points from the mesh. This design makes the annotation pipeline closer to real-world inference, where object geometry is reconstructed from camera observations. Similar to asset-level language annotation, we first compute the object bounding box and place eight cameras around the asset with evenly spaced azimuth angles. Each camera is oriented toward the bounding-box center and its distance is normalized by the object scale.

Given the depth image D_i from camera i, camera intrinsics K_i, and camera-to-world extrinsics $T_{c_i}^{w} = [R_i \mid t_i]$, each valid depth pixel (u, v) is back-projected into 3D 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 object point cloud is obtained by aggregating valid points from all rendered views:

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

where $N = 8$ by default and M_i denotes the object foreground mask from simulation. We remove background points using the mask and merge all points into the object coordinate frame.

Coverage verification with mesh-based reference points. Although the final annotation point cloud is reconstructed from rendered observations, we use mesh-based sampling only as a coverage

check. Specifically, we sample a reference point set $\mathcal{P}_{mesh}$ from the object mesh and compare it with the camera-fused point cloud $\mathcal{P}_{cam}$ inside the object's 3D region of interest. We measure the surface coverage ratio as

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

where ϵ is a distance threshold. We also compare the reconstructed 3D region of interest with the mesh-level region using voxel occupancy:

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

If either the surface coverage or ROI overlap is below a predefined threshold, we add extra cameras targeting poorly covered regions and repeat the reconstruction. This adaptive step ensures that thin structures, concave regions, and partially occluded parts are sufficiently observed while preserving the camera-based nature of the final point cloud.

Point-cloud downsampling. The fused point cloud can contain a large number of points, especially for high-resolution renderings. Before part segmentation, we downsample $\mathcal{P}_{cam}$ to $M = 80{,}000$ points using farthest point sampling (FPS). Starting from an initial point set $\mathcal{S}_1$, 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 $|\mathcal{S}| = M$. This preserves global shape coverage and avoids over-concentrating points on large planar surfaces. The downsampled point cloud is denoted as $\mathcal{P}_{80k}$.

Native 3D part segmentation. We perform asset-level part segmentation on $\mathcal{P}_{80k}$ using the Hunyuan3D part segmentation model with 182M parameters, built on native 3D part decomposition modules such as P3-SAM [57] and X-Part [58]. The model predicts a set of part masks

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

where each m_k corresponds to one segmented part. For downstream modules that require denser geometry, we propagate part labels from $\mathcal{P}_{80k}$ back to the original fused cloud by nearest-neighbor assignment:

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

In our implementation, the segmentation stage takes approximately half a minute per object, making it practical for large-scale automatic annotation.

Failure cases. The main failure mode of asset-level part segmentation is semantic granularity mismatch. Some objects are segmented at a coarser level than required for manipulation. For example, a wine glass may be predicted as a single part, while functional manipulation would benefit from separating the bowl, stem, and base. Such over-merged parts can limit part-aware language annotation and functional grasp generation, because different subparts correspond to different interaction strategies. We therefore treat part segmentation as a visual prior rather than a final executable label; downstream physics-based validation and task-specific geometric reasoning are still required to produce executable action annotations.

B.2 Room-level Visual Annotation Details

Wall-based room decomposition. For each room-scale scene, we first decompose the scene into individual rooms using wall boundaries and room-layout structure. Each room is represented as a 2D floor-plan polygon $\mathcal{R}_j$ in the global scene coordinate frame. This decomposition allows us to scan and annotate each room independently, while later stitching the resulting maps into a global room-level representation.

Rule-based scan-pose sampling. For each room $\mathcal{R}_j$, we randomly sample candidate camera and LiDAR scan poses inside the room polygon. A sampled pose is accepted only if it satisfies simple rule-based constraints: it must lie in free space, maintain a minimum distance from walls and large obstacles, and have a valid viewing direction toward the room interior rather than directly facing a nearby wall. We use simulator ground-truth segmentation only as an automatic view-quality signal. For each candidate pose π, we render an RGB image and an instance segmentation map, and compute the set of visible object IDs:

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

where $\mathcal{O}_j$ is the set of object instances in room $\mathcal{R}_j$, S_π is the rendered segmentation map, and τ_{area} filters out tiny or barely visible objects. We then greedily select scan poses that maximize object-ID coverage:

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

until the coverage ratio

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

exceeds a predefined threshold or a maximum number of poses is reached. This prevents the room-level annotation pipeline from using uninformative views that mostly face walls or empty regions, and ensures that nearly all object instances in each room are observed at least once.

Multi-height occupancy from LiDAR and ray tracing. Given the selected scan poses, we construct room-level occupancy maps using simulated LiDAR and ray tracing. For each pose π_t and each height level $h_\ell \in \mathcal{H}$, we cast horizontal rays in uniformly sampled directions. For a ray direction $\mathbf{d}_\alpha$, let ρ_α be the first-hit distance returned by ray tracing. The cells before the hit are marked as free, while the hit cell is marked as 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 a 3D point to a 2D grid cell, and δ is a small safety margin before the surface hit. We aggregate all rays with a log-odds occupancy update:

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

and obtain the occupancy probability by $P_\ell(c) = \sigma(L_\ell(c))$. The final multi-height occupancy annotation is

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

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

Global map stitching. After scanning each room independently, we stitch all room-level maps into a global scene representation using the floor-plan alignment and the global coordinate frame. Let T_j^w denote the transform from the local room frame to the global world frame. For each room map G_j, we project its grid cells into the global map:

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

where $\text{Fuse}(\cdot)$ combines overlapping evidence from adjacent rooms and shared boundaries. This produces a globally consistent occupancy map, object-layout map, and top-view representation for the entire scene.

VLM-assisted wall and floor-plan filtering. Raw occupancy maps often contain both structural elements and movable objects such as tables, chairs, cabinets, and shelves. To obtain a clean floor plan and wall-structure annotation, we use the VLM as a structural component selector. We first generate a top-view map with connected components and object labels, then prompt the VLM to keep persistent room-structure components, such as walls, doors, and room boundaries, while filtering out movable furniture and object clutter. Formally, given connected components $\mathcal{C} = \{C_k\}$ from the raw top-view occupancy, the VLM predicts a subset $\mathcal{C}_{wall} \subset \mathcal{C}$, and the wall map is constructed as

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

The resulting room-level visual annotations include multi-height occupancy maps, object-layout maps, cleaned floor plans, and wall-structure maps. These representations provide global spatial priors for exploration, navigation target generation, and scene-level action annotation.

Prompt for VLM-assisted wall filtering. We use the following prompt to remove movable objects from the raw top-view occupancy map and retain structural room boundaries.

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.
```

C Details of physics annotation pipeline

C.1 Details of the Unified Action Annotation Schema

In the main paper, we describe action annotation as a hierarchical candidate-bank schema. Here we provide additional details and concrete examples. The key idea is that action labels are not attached to an object as a single canonical action. Instead, they are organized as a **one-to-many mapping** from grounded visual anchors to compatible skills and then to multiple feasible action candidates:

visual anchor $\rightarrow$ functional affordance $\rightarrow$ compatible skills $\rightarrow$ candidate bank $\rightarrow$ action candidates.

This design explicitly preserves both **action diversity** and **functional consistency**. The visual-language annotations do not only identify where an object part or keypoint is; they also provide functional cues about how the region should be used. For example, a mug handle is not merely a graspable geometry: it is a functional handle that supports holding, lifting, and pouring. In contrast, the mug body may also be physically graspable, but such grasps are less functionally aligned for tasks that require using the mug as intended. Similarly, a cabinet handle supports both grasping and articulated opening, while the free space in front of the cabinet supports navigation and approach-pose annotation.

Given an asset $\mathcal{A}$, the visual annotation pipeline produces a set of grounded interaction 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 $\mathcal{P}$ denotes part-level regions, $\mathcal{K}$ denotes semantic keypoints, $\mathcal{R}^{\text{aff}}$ denotes affordance regions, and $\mathcal{R}^{\text{scene}}$ denotes scene-level regions. Each anchor $h \in \mathcal{H}(\mathcal{A})$ is associated with a grounded functional affordance $\phi(h)$, such as grasp-for-holding, grasp-for-opening, articulation, hanging, or

navigation approach. We use $\phi(h)$, together with language priors, geometric cues, and task constraints, to infer a set of compatible skills $\mathcal{S}(h)$. For each anchor-skill pair, we maintain a candidate bank:

$$\mathcal{B}_{h,s} = \left\{ a_{h,s}^{(i)} \right\}_{i=1}^{N_{h,s}}, \qquad s \in \mathcal{S}(h).$$

Each candidate action is represented as

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

The notation is summarized in Table 5.

Example: mug. For a mug asset, the visual annotation pipeline may identify several anchors, including the handle region, the mug body surface, and semantic keypoints on the handle or rim. In this example, we focus on grasp-related skills. The mug handle is a **functional grasp anchor**: grasping the handle preserves the intended use of the mug and supports downstream tasks such as lifting, carrying, pouring, or placing. The mug body may also be physically graspable, but it is not necessarily a functional grasp region for these long-horizon tasks, since grasping the body can occlude the cup, interfere with pouring, or produce task-irrelevant contact. Therefore, the schema distinguishes **functional grasp candidates** from merely **physically feasible grasp candidates**.

For example, if $h = h_{\text{handle}}$ and $s = s_{\text{grasp}}$, then the candidate bank is

$$\mathcal{B}_{h_{\text{handle}},s_{\text{grasp}}} = \left\{ a_{h_{\text{handle}},s_{\text{grasp}}}^{(1)}, a_{h_{\text{handle}},s_{\text{grasp}}}^{(2)}, \ldots, a_{h_{\text{handle}},s_{\text{grasp}}}^{(N_{h_{\text{handle}},s_{\text{grasp}}})} \right\}.$$

These candidates may correspond to grasping the handle from different sides, using different gripper orientations, or approaching with different pre-grasp offsets. Because the handle carries the affordance $\phi(h_{\text{handle}}) =$ grasp-for-use, these candidates are interpreted as functional grasp candidates rather than merely stable grasps. In contrast, candidates from h_{body} may still be useful for simple pickup or stabilization, but they are marked differently because they may not support downstream functional tasks such as pouring. This allows downstream modules to select not only physically feasible grasps, but also functionally appropriate grasps for long-horizon manipulation.

Example: cabinet. For a cabinet or drawer asset, the visual annotation pipeline may identify anchors such as the handle, door panel, drawer front, and reachable free-space region in front of the cabinet. These anchors support different skills. A handle anchor may support both grasping and articulation, while a scene-level free-space anchor supports navigation and approach-pose annotation. The key point is that the handle is not simply a graspable part: it carries the functional affordance of opening or closing the articulated structure.

The cabinet example illustrates how functional grounding interacts with diversity. Diversity exists **across anchors**: the handle and free-space region support different types of interaction. Diversity also exists **within each anchor-skill pair**: the same handle-articulation pair may contain multiple opening trajectories with different pull directions, waypoint sequences, and contact configurations. At the same time, these candidates are function-conditioned by $\phi(h_{\text{handle}}) =$ grasp-for-opening, so a valid grasp or trajectory is not only collision-free and kinematically feasible, but also useful for opening the cabinet or drawer. Therefore, the cabinet is not annotated with a single "open" action; it is annotated with a structured bank of validated, function-conditioned grasp, articulation, and navigation candidates.

Discussion. This hierarchical representation provides a common interface for heterogeneous action primitives while separating physical feasibility from functional suitability. Parallel-jaw grasping, dexterous grasping, articulation, insertion, hanging, deformable manipulation, and navigation all share the same high-level structure, while differing in the interpretation of $\phi(h)$, $x^{(i)}$, $\theta^{(i)}$, and $\tau^{(i)}$. This allows downstream systems to query action annotations uniformly. For example, a manipulation policy can request all functionally valid grasp candidates for a mug handle, a mobile manipulator can request all feasible base poses for approaching a cabinet, and a data-collection module can sample diverse candidates from $\mathcal{B}_{h,s}$ according to validation score, functional affordance, or diversity descriptor.

Table 5: Symbols in the unified action annotation schema. The additional functional affordance term $\phi(h)$ distinguishes functionally meaningful actions from merely physically feasible ones.

Symbol	Meaning	Examples
$\mathcal{A}$	Input asset	A single object such as a mug, or a room-scale scene containing a cabinet, table, and floor space.
$\mathcal{H}(\mathcal{A})$	Grounded anchors	All actionable parts, keypoints, affordance regions, and scene-level regions detected on asset $\mathcal{A}$.
$\mathcal{P}(\mathcal{A})$	Part regions	Mug handle, mug body, cabinet handle, cabinet door, drawer front.
$\mathcal{K}(\mathcal{A})$	Semantic keypoints	Handle endpoint, drawer handle center, cabinet door corner, cloth corner.
$\mathcal{R}^{\mathrm{aff}}(\mathcal{A})$	Affordance regions	Functional handle region, graspable region, support edge, hook region, insertion socket.
$\mathcal{R}^{\mathrm{scene}}(\mathcal{A})$	Scene regions	Reachable free space in front of a cabinet, object-centric approach region, collision-free base-pose region.
h	Visual anchor	The handle region of a mug, the center of a cabinet handle, or a free-space region in front of a cabinet.
$\phi(h)$	Functional affordance	For a mug handle: grasp-for-holding or grasp-for-pouring. For a mug body: physical grasp or stabilizing grasp. For a cabinet handle: grasp-for-opening. For free space: navigation approach.
$\mathcal{S}(h)$	Compatible skills	For a mug handle: functional grasp or dexterous grasp. For a cabinet handle: grasp and articulation. For free space: navigation.
$\mathcal{B}_{h,s}$	Candidate bank	All feasible function-conditioned grasps on a mug handle, or all feasible opening trajectories for a cabinet handle.
$N_{h,s}$	Bank size	The number of validated grasp poses, dexterous contact sets, articulation trajectories, or base poses for skill s at anchor h.
$a_{h,s}^{(i)}$	Action candidate	One grasp pose, one dexterous contact configuration, one articulation trajectory, or one navigation base pose.
s	Skill type	Grasp, dexterous grasp, articulation, insertion, hanging, deformable manipulation, or navigation.
$o(h)$	Associated instance	The mug associated with a handle anchor, the cabinet door associated with a handle anchor, or the scene containing a navigation region.
$x^{(i)}$	Concrete target	A contact point, surface patch, support edge, opening center, articulation handle point, or navigation base pose.
$\theta^{(i)}$	Action parameters	6D grasp pose, gripper width, contact set, hand configuration, insertion direction, hanging pose, articulation axis, or target base pose.
$\tau^{(i)}$	Trajectory	Approach-contact-retreat trajectory for grasping, pull trajectory for articulation, or navigation approach path.
$v^{(i)}$	Validation metadata	Collision status, inverse-kinematics feasibility, contact stability, task success score, path feasibility, trajectory statistics, and functional success.
$d^{(i)}$	Diversity descriptor	Target location, approach direction, contact mode, trajectory family, grasp type, embodiment parameters, and perturbation seed.

C.2 Details of the General Physics-based Action Generation Pipeline

C.2.1 Details of Candidate Target Generation

In the main paper, candidate target generation is summarized as the first stage of the physics-based action annotation pipeline. Here we provide additional details. This stage converts visual-language annotations into concrete target anchors that seed the candidate banks. The overall process is:

skill s+language annotation+visual grounding $\rightarrow$ candidate region $\mathcal{C}_s$ $\rightarrow$ sampled target anchors $\rightarrow$ filtered candidate bank.

Table 6: Example instantiations of the unified action annotation schema for a mug. The handle is treated as a functional grasp anchor, while the body surface mainly provides physically feasible but less functionally aligned grasp candidates.

Symbol	Mug example	Interpretation
$\mathcal{A}$	Mug asset	The input object to be annotated, including its handle, body surface, rim, and opening geometry.
h_{handle}	Handle region	A part-level anchor from $\mathcal{P}(\mathcal{A})$ and an affordance anchor from $\mathcal{R}^{\text{aff}}(\mathcal{A})$. It provides the spatial support for functional grasp candidates.
$\phi(h_{\text{handle}})$	Grasp-for-use	The functional affordance of the handle. It indicates that grasps on this anchor are intended for holding, lifting, carrying, or pouring the mug.
s_{grasp}	Parallel-jaw grasp	A compatible skill in $\mathcal{S}(h_{\text{handle}})$. This skill instantiates function-conditioned 6D gripper poses on the handle.
$\mathcal{B}_{h_{\text{handle}},s_{\text{grasp}}}$	Functional grasp bank	A set of feasible parallel-jaw grasp candidates on the mug handle, with different contact locations, approach directions, gripper orientations, and gripper widths. These candidates are both physically feasible and functionally aligned with using the mug.
$a_{h,s}^{(i)}$	One handle grasp	One concrete functional grasp candidate. Here, $x^{(i)}$ is a contact point or patch on the handle, $\theta^{(i)}$ contains a 6D grasp pose and gripper width, $\tau^{(i)}$ stores an approach-contact-retreat trajectory, $v^{(i)}$ records IK, collision, and functional feasibility, and $d^{(i)}$ records approach direction and grasp mode.
s_{dex}	Dexterous grasp	Another compatible skill in $\mathcal{S}(h_{\text{handle}})$. The same functional handle anchor can also instantiate multi-finger contact configurations.
$\mathcal{B}_{h_{\text{handle}},s_{\text{dex}}}$	Functional dexgrasp bank	A set of feasible dexterous grasp candidates on the handle. Candidates differ in fingertip contacts, palm pose, joint configuration, and contact mode, while preserving the handle's functional role.
h_{body}	Mug body surface	Another part-level visual anchor. It may be physically graspable, but it is generally less functionally aligned with using the mug for long-horizon tasks such as pouring or drinking.
$\phi(h_{\text{body}})$	Physical grasp	The affordance of the body surface is treated as physical or stabilizing grasp rather than functional grasp-for-use. It can be useful for relocation or simple pickup, but may not preserve the intended mug functionality.
$\mathcal{B}_{h_{\text{body}},s_{\text{dex}}}$	Body grasp bank	A set of dexterous or stabilizing grasp candidates on the mug body surface, with different contact sets, hand configurations, and stability scores. These candidates may pass physical validation but receive different functional metadata from handle grasps.

The key point is that candidate targets are not sampled from the entire asset uniformly. Instead, they are sampled from skill-compatible and function-conditioned regions identified by the visual-language annotation pipeline. For example, a mug handle can be selected as a functional grasp region for holding or pouring, while the mug body can be selected as a physically graspable region for simple pickup or stabilization.

Candidate region construction. Given an asset $\mathcal{A}$ and a skill type s, we first construct a set of skill-compatible candidate regions:

$$\mathcal{C}_s(\mathcal{A}) = \Gamma_s\left(\mathcal{Y}^{\text{lang}}, \mathcal{Y}^{\text{vis}}, \mathcal{G}(\mathcal{A})\right),$$

where $\mathcal{Y}^{\text{lang}}$ contains language-level semantics and functional priors, $\mathcal{Y}^{\text{vis}}$ contains visual annotations such as part masks, keypoints, affordance regions, and scene maps, and $\mathcal{G}(\mathcal{A})$ denotes the asset geometry. Each candidate region $c \in \mathcal{C}_s(\mathcal{A})$ is associated with a functional affordance $\phi(c)$, such

as grasp-for-use, physical grasp, grasp-for-opening, support-for-hanging, or navigation approach. This functional affordance determines whether a region is merely physically usable or useful for a downstream task.

Different annotation sources provide candidate regions for different skills. Part masks are used to identify object parts such as mug handles, mug bodies, cabinet handles, drawer fronts, and hooks. Semantic keypoints are used for point-like anchors such as garment corners, handle centers, or articulation keypoints. Affordance regions are used to identify functional regions such as handles, openings, support edges, and hanging points. Room-level occupancy or BEV maps are used to identify traversable free space and interaction-ready base-pose regions for navigation.

Sampling target anchors. After constructing $\mathcal{C}_s(\mathcal{A})$, we sample concrete target anchors from each candidate region. For object-centric skills, we first extract the subset of surface points or mesh faces belonging to the candidate region. We then use farthest-point sampling (FPS) to obtain spatially diverse target anchors:

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

where $\mathcal{P}_c$ is the set of surface points inside candidate region c. FPS avoids generating many redundant anchors in the same local area and encourages coverage over the entire functional region. For each sampled point, we estimate local geometric attributes such as surface normal, tangent directions, curvature, local patch size, and local reference frame. These attributes are later used to instantiate grasp poses, contact frames, insertion frames, or articulation waypoints.

For keypoint-based skills, the annotated keypoints can directly serve as anchors. For example, garment manipulation may use cloth corners or edge keypoints as grasp anchors. We can also sample small perturbations around each keypoint to create nearby candidate anchors, then form bimanual anchor pairs for skills such as stretching, folding, or lifting. These pairs are filtered by distance, symmetry, visibility, and robot reachability.

For scene-centric skills, we sample targets on occupancy or BEV maps. Let $\mathcal{M}^{\text{bev}}$ denote the BEV map and $\mathcal{F}$ denote the set of traversable cells. We sample 2D positions from $\mathcal{F}$ using grid sampling or 2D FPS:

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

where (x_j, y_j) is a reachable location and ψ_j is a base orientation. The orientation is chosen to face the target object, affordance region, or manipulation anchor, possibly with multiple discrete approach angles. These samples become navigation goals or interaction-ready base poses.

Geometry and embodiment filtering. The sampled anchors are filtered before they are inserted into the candidate bank. Filtering removes targets that are unlikely to produce physically feasible or functionally meaningful actions. The filters depend on the skill.

For grasping, we remove regions that are too small for the gripper, lack sufficient contact area, have unsuitable surface normals, violate gripper-width limits, or cause immediate collision between the gripper and the object. For example, if a part mask corresponds to a very thin or tiny decorative detail, it may be visually detected but filtered out because the gripper cannot form a stable grasp around it. For functional grasping, we further check whether the sampled target is consistent with the intended affordance. A mug-handle grasp is retained as a functional grasp-for-use candidate, while a mug-body grasp may be retained only as a physical or stabilizing grasp.

For dexterous grasping, we filter contact sets that violate hand joint limits, produce self-collision, lack sufficient fingertip support, or place contacts on semantically inappropriate regions. For articulation, sampled handle points are filtered by whether they can support the required motion direction, maintain contact during pulling or rotating, and avoid collision with nearby geometry. For insertion, sampled openings are filtered by entry size, alignment feasibility, collision-free approach, and whether the insertion axis is geometrically well defined. For hanging, support edges or hooks are filtered by edge orientation, contact stability under gravity, and clearance around the hanging object.

For navigation, BEV samples are filtered by traversability, clearance, path connectivity, and manipulation reachability. A base pose must be collision-free, reachable from the scene layout, and close enough for the robot to interact with the target object or part. We also remove base poses with poor visibility or orientations that make the downstream manipulation target unreachable.

Forming candidate banks. After sampling and filtering, each remaining target anchor becomes a seed for action candidate generation. For a skill s and a candidate region c, the retained anchors are denoted as

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

where $x_{c,s}^{(j)}$ may be a contact point, local surface patch, support edge, opening frame, handle interaction point, navigation goal, or interaction-ready base pose. These anchors are then used to instantiate the candidate bank:

$$\mathcal{B}_{h,s} = \left\{ a_{h,s}^{(i)} \right\}_{i=1}^{N_{h,s}}.$$

At this stage, the candidates mainly contain target-level information. Later stages fill in skill-specific action parameters, trajectories, optimization results, and validation metadata. For example, a sampled mug-handle anchor becomes several 6D grasp candidates with different approach directions and gripper widths; a sampled cabinet-handle anchor becomes several grasp and articulation candidates; and a sampled BEV base pose becomes a navigation candidate with an approach direction and optional path.

Example: mug. For a mug, language annotations identify the handle as a functional region for holding, lifting, and pouring. The visual annotation pipeline grounds this cue into a handle part mask. Candidate target generation then samples target anchors on the handle using FPS. These anchors seed a functional grasp bank:

$$\mathcal{B}_{h_{\text{handle}},s_{\text{grasp}}}.$$

In parallel, the mug body may be identified as a physically graspable surface. FPS on the body mask can produce additional grasp anchors, but these are marked with a different functional affordance, such as physical grasp or stabilizing grasp. Thus, both handle and body grasps can be physically feasible, but only the handle grasps are labeled as functional grasp-for-use candidates for tasks such as pouring.

Example: garment. For a garment, semantic keypoints such as corners, sleeves, hems, or collar points directly define candidate anchors. Instead of sampling densely over the entire cloth surface, the pipeline samples around these keypoints and forms keypoint pairs for bimanual actions. For example, two sleeve endpoints may seed a stretching trajectory, while two bottom corners may seed a folding or lifting trajectory. Candidate pairs are filtered by visibility, distance, symmetry, and whether the robot can reach both anchors simultaneously.

Example: cabinet. For a cabinet, language annotations identify the handle as a functional region for opening, and visual grounding provides the handle mask or handle keypoint. FPS samples interaction anchors on the handle. The same handle region can seed both a grasp bank and an articulation bank:

$$\mathcal{B}_{h_{\text{handle}},s_{\text{grasp}}} \quad \text{and} \quad \mathcal{B}_{h_{\text{handle}},s_{\text{art}}}.$$

Grasp candidates are filtered by gripper feasibility and collision, while articulation candidates are filtered by whether the sampled handle point can support the required pull or rotation direction. For a room-scale cabinet scene, the BEV map additionally provides free-space regions in front of the cabinet. Grid sampling or 2D FPS generates candidate base poses, which are filtered by traversability, clearance, and whether the robot can reach the handle from that pose.

C.3 Details of Trajectory Generation

After candidate target generation, each retained target anchor is converted into an initial trajectory or waypoint sequence. This stage does not yet produce the final executable trajectory; instead, it provides a template-level motion seed that is later refined by trajectory optimization and checked by physics validation. Given a candidate action with anchor h, functional affordance $\phi(h)$, target $x^{(i)}$, and skill-specific parameters $\theta^{(i)}$, we generate an initial trajectory

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

where $\mathcal{G}_s$ is a primitive-specific trajectory generator. Although each skill has its own motion template, we organize these templates by the source of their constraints, as summarized in Table 10.

Object-property-conditioned trajectories. This family uses static object or garment properties to generate motion templates. For parallel-jaw grasping, a candidate target provides a grasp pose $T_{\text{grasp}} = [R_{\text{grasp}}, p_{\text{grasp}}]$ and an approach direction $\mathbf{a}_{\text{grasp}}$. We generate a short approach-contact-retreat trajectory:

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

where δ_{pre} is the pre-grasp offset, δ_{ret} is the retreat distance, and $\mathbf{r}$ is a retreat direction such as the reverse approach direction or an upward lifting direction. The resulting template is

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

For dexterous grasping, the same idea is extended to palm poses, fingertip contact sets, and hand joint configurations. The pre-grasp pose is chosen to approach the object without early collision, and the closure phase follows the candidate contact set.

For garment manipulation, trajectories are generated from semantic keypoints. For an upper-body garment, we use keypoints such as left/right sleeve endpoints, left/right bottom corners, and left/right shoulder points:

$$\mathcal{K}_{\text{top}} = \{k_{\text{lslv}}, k_{\text{rslv}}, k_{\text{lb}}, k_{\text{rb}}, k_{\text{lsh}}, k_{\text{rsh}}\}\,,$$

where "slv" denotes sleeve, "b" denotes bottom, and "sh" denotes shoulder. A sleeve-fold trajectory is generated by defining a fold line from the shoulder to the bottom corner on each side. For the left sleeve, the fold line is

$$\ell_{\text{L}} = \ell\,(k_{\text{lsh}}, k_{\text{lb}})\,,$$

and the sleeve target is obtained by reflecting the sleeve endpoint across this line:

$$k^{\star}_{\text{lslv}} = \text{Ref}_{\ell_{\text{L}}}\,(k_{\text{lslv}})\,.$$

Similarly, the right sleeve is folded using the line

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

The generated bimanual waypoint sequence lifts the sleeve endpoint, moves it toward the mirrored target, and places it inside the garment body. After sleeve folding, we generate a bottom-up fold by moving the bottom keypoints toward the shoulder line:

$$\ell_{\text{sh}} = \ell\,(k_{\text{lsh}}, k_{\text{rsh}})\,, \qquad k^{\star}_{\text{lb}} = \Pi_{\ell_{\text{sh}}}\,(k_{\text{lb}})\,, \qquad k^{\star}_{\text{rb}} = \Pi_{\ell_{\text{sh}}}\,(k_{\text{rb}})\,,$$

where Π_{ℓ} denotes projection onto line ℓ. This produces a folding sequence in which the bottom hem is lifted and folded toward the shoulder region.

For pants, we use a symmetric folding rule. We first estimate the central axis of the pants from waistband and leg keypoints. The two side or leg regions are folded toward the central axis so that the legs overlap. Then the bottom or cuff keypoints are folded upward toward the waistband region. A second bottom-up fold can be applied to compact the garment further. Thus, pants folding follows a "side-fold then fold-again" template:

$$\text{left/right side fold} \rightarrow \text{bottom-to-waist fold} \rightarrow \text{second compacting fold.}$$

For fling-like garment motions, we choose two keypoints, such as sleeve endpoints or bottom corners, and generate a bimanual lift-and-stretch trajectory. The lift height, lateral stretch, and optional shaking amplitude are scaled by the garment length

$$L_{\text{garment}} = \|k_1 - k_2\|_2\,,$$

so that larger garments receive longer trajectories while remaining within bimanual reachability constraints.

Table 11 summarizes the garment templates.

Object-trajectory-conditioned trajectories. This family is used for articulated objects whose moving parts follow a joint trajectory. Examples include opening a cabinet door, pulling a drawer, closing a lid, rotating a button, or pushing a button. The key constraint is that the end-effector should follow the moving object part while maintaining the intended contact or grasp relation.

Let $T_{\text{part}}(q)$ denote the pose of the moving part at joint state q, and let $T_{\text{eef}}^{\text{init}}$ denote the end-effector pose after grasping or contacting the handle at the initial joint state q_{init}. For handle-based articulation, we preserve the relative transform

$$T_{\text{rel}} = T_{\text{part}}(q_{\text{init}})^{-1} T_{\text{eef}}^{\text{init}}.$$

For any later joint state q, the idealized end-effector pose is

$$T_{\text{eef}}(q) = T_{\text{part}}(q) T_{\text{rel}}.$$

We generate a waypoint sequence by interpolating the joint state

$$q_j = q_{\text{init}} + j\Delta q, \qquad j = 0, \ldots, J,$$

until a target joint state q_{goal} is reached. This formulation applies to both revolute joints, such as doors and lids, and prismatic joints, such as drawers. The generated end-effector waypoints are later checked by inverse kinematics, collision checking, and contact-maintenance validation.

Different articulation actions use different joint schedules and contact assumptions, as shown in Table 12.

Object-vector-conditioned trajectories. This family is used when the action is defined by a semantic or geometric direction rather than by an articulated part trajectory. Examples include insertion, hanging, pouring, and placement. Each action has an active vector, a passive vector, or both. The active vector belongs to the manipulated object, such as the axis of a peg or the pouring direction of a container. The passive vector belongs to the target region, such as the axis of a socket, the normal of a placement surface, or the support direction of a hook.

For insertion, we align the active insertion axis $\mathbf{u}_{\text{act}}$ with the passive insertion axis $\mathbf{u}_{\text{pas}}$:

$$R\mathbf{u}_{\text{act}} \approx -\mathbf{u}_{\text{pas}},$$

where R is the end-effector or object orientation. Given an insertion target position p_{tar} and insertion direction $\mathbf{u}_{\text{pas}}$, we generate an approach and execution trajectory:

$$p_{\text{pre}} = p_{\text{tar}} - \delta_{\text{pre}}\mathbf{u}_{\text{pas}}, \qquad p(t) = p_{\text{pre}} + t\delta_{\text{ins}}\mathbf{u}_{\text{pas}}, \quad t \in [0, 1].$$

For hanging, the end-effector approaches a support edge or hook along a clearance direction, places the object into a stable contact configuration, and then lowers or releases the object along gravity. For pouring, the trajectory rotates the grasped object so that its active pouring vector aligns with the desired passive receiving direction or target container. For placement, the trajectory aligns the object with the support normal, approaches the placement pose, releases, and retreats.

Table 13 summarizes representative object-vector-conditioned templates.

Scene-trajectory-conditioned trajectories. For scene-level skills, trajectory generation operates on room-level occupancy or BEV maps. We first extract traversable free-space cells and sample interaction-ready base poses around the target object, part, or affordance region. Each base pose is represented as

$$q = (x, y, \psi),$$

where (x, y) is a reachable location and ψ is the base orientation facing the manipulation target. Given the current robot pose and a sampled target base pose, we compute a coarse global navigation path using A* on the occupancy or BEV grid. This produces a sequence of grid-level waypoints connecting free space to the interaction region.

The A* path is used only as a geometric trajectory seed. It does not fully account for embodiment-specific motion constraints, such as the turning radius of Ackermann platforms, differential-drive dynamics, or holonomic constraints for quadrupeds and humanoids. These constraints are handled in the subsequent trajectory optimization stage, where the A* path is locally refined by rollout-based control such as DWB. The resulting scene-level trajectory provides a collision-free and interaction-ready approach path to the manipulation target.

C.4 Details of Trajectory Optimization

After trajectory generation, each candidate action has an initial template-level trajectory $\tau_0^{(i)}$ and skill-specific parameters $\theta_0^{(i)}$. These templates are intentionally simple: they are generated from geometric rules, motion templates, A* paths, or keypoint-based heuristics. The trajectory optimization stage refines them into asset-specific and embodiment-aware candidates before physics validation. Given a candidate action

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

we optimize the action parameters and trajectory as

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

where $\mathcal{R}$ denotes the robot embodiment, including its kinematics, collision model, gripper or hand model, and, for navigation, base dynamics. We use a skill-dependent objective:

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

The terms are activated depending on the skill. E_{func} encourages consistency with the grounded functional affordance $\phi(h)$, E_{task} tracks task-specific goals such as opening, insertion depth, or navigation target, E_{geom} adapts waypoints to asset geometry, E_{contact} improves contact alignment or contact maintenance, E_{coll} penalizes penetration and unsafe clearance, E_{kin} enforces embodiment-specific kinematics or dynamics, and E_{smooth} regularizes motion. The optimized output is not yet treated as a final annotation; it is passed to physics validation, where infeasible candidates are rejected.

Dexterous and contact-aware optimization. For dexterous grasping, the initial candidate usually specifies a target region, a set of possible contact points, and an initial hand or palm pose. However, this template may not be physically plausible: fingertips may not align with the object surface, the hand may penetrate the object, the joint configuration may violate limits, or the contact set may not support the intended task. We therefore optimize the hand pose, finger joints, and contact assignment jointly.

Let q_{hand} denote hand joint angles, T_{hand} denote the hand root pose, and $\mathcal{C}$ denote selected contact links or fingertips. The contact point generated by forward kinematics for contact $c \in \mathcal{C}$ is denoted by

$$p_c = \text{FK}_c(T_{\text{hand}}, q_{\text{hand}}).$$

Given a grounded functional region h with affordance $\phi(h)$, we use a region-consistency term

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

where $\text{dist}(p_c, h)$ measures the distance from the contact point to the grounded part mask or affordance region. This term prevents the optimizer from drifting to geometrically convenient but functionally incorrect contacts. For example, if $\phi(h)$ is grasp-for-use on a mug handle, contacts are encouraged to stay on the handle rather than moving to the mug body.

To improve physical grasp quality, we add contact and wrench-compatibility terms. Let n_c be the local surface normal at the matched object point and f_c be a feasible contact force under a friction cone. A generic wrench-compatibility term can be written as

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

where w_j are target wrench directions, G_c maps contact forces to object wrenches, and $\mathcal{K}_{\mu_c}$ is the friction cone at contact c. This term encourages the selected contacts to support task-relevant external wrenches. For a functional mug-handle grasp, the target wrench set can emphasize lifting or pouring stability; for a cabinet-handle grasp, the target wrench set can emphasize pulling along the opening direction.

We also use distance, collision, and kinematic penalties:

$$E_{\text{contact}} = \sum_{c\in\mathcal{C}} \rho\left(d_{\mathcal{A}}(p_c)\right), \qquad E_{\text{coll}} = \sum_b \max\left(0,\, \epsilon_{\text{safe}} - d_{\text{obs}}(b)\right)^2,$$

where $d_{\mathcal{A}}$ measures signed distance to the target object surface, $d_{\text{obs}}(b)$ measures clearance for hand or arm body element b, and $\rho(\cdot)$ encourages near-surface contact without excessive penetration. Joint limits and self-collision are included in E_{kin}. The optimized dexterous candidate stores the final hand pose, joint configuration, contact set, pre-grasp pose, and approach trajectory in $\theta_*^{(i)}$ and $\tau_*^{(i)}$.

Pre-grasp and approach refinement. For grasp-related skills, optimization also refines the pre-grasp and approach motion. Given an optimized grasp pose T_{grasp} and approach direction $\mathbf{a}$, the initial pre-grasp pose is typically placed as

$$T_{\text{pre}} = T_{\text{grasp}} \ominus \delta_{\text{pre}}\mathbf{a},$$

where δ_{pre} is an offset along the reverse approach direction. This pre-grasp can fail when nearby geometry blocks the approach path. We therefore optimize the approach offset, approach direction, and intermediate waypoints to reduce collision and improve reachability:

$$E_{\text{approach}} = \sum_t E_{\text{coll}}(T_t) + \lambda_{\text{dir}} \sum_t \|\Delta p_t - \alpha_t \mathbf{a}\|_2^2 + \lambda_{\text{smooth}} \sum_t \|T_{t+1} \ominus T_t\|^2 .$$

This produces a collision-aware approach-contact-close-retreat trajectory. For dexterous hands, the finger closure phase is also adjusted so that fingers contact the object progressively without early penetration.

Adaptive geometry-aware waypoint optimization. Some primitives require adapting template waypoints to asset-specific geometry. This is especially important for deformable or articulated objects, where a fixed motion template can fail across different shapes or scales. We represent the generated template waypoints as

$$\tau_0 = \{p_1^0, \ldots, p_T^0\},$$

and optimize corrected waypoints

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

under task and geometry constraints:

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

This keeps the optimized trajectory close to the template while adapting it to the current asset.

For garment folding, we adapt fold targets according to keypoint distances. For an upper-body garment, sleeve folding uses sleeve endpoints, shoulder keypoints, and bottom corners. The template places the sleeve endpoint at a mirrored target inside the garment body. However, the exact place point should depend on sleeve length and body width. We therefore scale the inward displacement by garment geometry:

$$\delta_{\text{sleeve}} = \text{clip}\left(\alpha_{\text{sleeve}} L_{\text{sleeve}}, \delta_{\min}, \delta_{\max}\right),$$

where L_{sleeve} is the distance between the shoulder keypoint and sleeve endpoint. The sleeve place point is shifted inward by δ_{sleeve} along the fold direction. Similarly, for bottom-up folding, the target position is adapted using the hem-to-shoulder distance:

$$\delta_{\text{bottom}} = \text{clip}\left(\alpha_{\text{bottom}} L_{\text{hem}\to\text{shoulder}}, \delta_{\min}, \delta_{\max}\right).$$

This prevents short garments from being over-folded and long garments from being under-folded.

For pants, the generated template first folds the two sides or legs toward the centerline and then folds the bottom region upward toward the waistband. Optimization adjusts the side-fold and bottom-fold targets according to leg length, waistband width, and bimanual reachability. If the two grasp points are too far apart for the embodiment, the optimizer moves them to nearby keypoint perturbations or selects a shorter folding displacement while preserving the same folding intention.

For fling-like garment motions, the template chooses two keypoints and generates a bimanual lift-and-stretch trajectory. The lift height and lateral stretch are scaled by the distance between the selected keypoints:

$$H_{\text{lift}} = \text{clip}\left(\alpha_H \|k_1 - k_2\|_2, H_{\min}, H_{\max}\right), \qquad D_{\text{stretch}} = \text{clip}\left(\alpha_D \|k_1 - k_2\|_2, D_{\min}, D_{\max}\right).$$

The optimizer then adjusts these values to satisfy bimanual reachability and avoid self-collision between the two arms.

Articulation trajectory refinement. For articulated objects, trajectory generation produces waypoints by following the motion of a movable part, such as a cabinet door, drawer, lid, or button. The initial trajectory assumes that the robot maintains an intended contact or grasp relation to the moving part. However, a direct template may fail because of robot reachability limits, collision with the object frame, or loss of contact. We therefore optimize the end-effector waypoints, joint schedule, and contact pose.

Let q_t denote the object joint state at waypoint t, and let $T_{\mathrm{eef},t}$ be the corresponding end-effector pose. We use a tracking term that encourages the end-effector to follow the moving part:

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

where $\widehat{T}_{\mathrm{eef}}(q_t)$ is the idealized end-effector pose computed from the articulated part motion and the initial grasp or contact relation. We also use a contact-maintenance term:

$$E_{\mathrm{maintain}} = \sum_t \mathrm{dist}\left(p_{\mathrm{contact},t}, h\right)^2 + \lambda_n \sum_t \left(1 - n_{\mathrm{contact},t}^\top n_{\mathrm{part},t}\right) ,$$

which keeps the contact point on the handle or movable part and encourages consistent contact normals.

For handle-based opening and closing, the optimizer adjusts the waypoint spacing and pull direction so that the robot can maintain contact while the door or drawer moves. For push-based closing, the trajectory is optimized to maintain safe pushing contact without requiring a fixed grasp. For buttons, the optimization bounds the push displacement by the button travel range and aligns the push direction with the button normal. For rotating buttons or lids, angular waypoints are refined to satisfy wrist reachability and avoid collisions.

Object-vector-conditioned trajectory refinement. Vector-conditioned skills are defined by semantic or geometric directions, such as insertion axes, hanging directions, pouring directions, and placement normals. The template trajectory aligns the object or end-effector with the vector and moves along it. Optimization improves alignment, clearance, and task-specific feasibility.

For insertion, let $\mathbf{u}_{\mathrm{act}}$ be the active insertion axis of the manipulated object and $\mathbf{u}_{\mathrm{pas}}$ be the passive insertion axis of the target opening or socket. The alignment cost is

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

where R is the object or end-effector orientation. The insertion trajectory also includes an approach-clearance term and an insertion-depth term:

$$E_{\mathrm{insert}} = \lambda_{\mathrm{align}} E_{\mathrm{align}} + \lambda_{\mathrm{depth}} \left(z_{\mathrm{target}} - z_{\mathrm{inserted}}\right)^2 + \lambda_{\mathrm{coll}} E_{\mathrm{coll}}.$$

This refines the alignment pose, entry pose, and insertion depth.

For hanging, optimization adjusts the approach pose so that the object clears the hook or support edge, establishes contact, and lowers into a gravity-stable configuration. The stability term encourages the object center of mass to lie below or behind the support contact after release:

$$E_{\mathrm{hang}} = E_{\mathrm{align}} + \lambda_{\mathrm{com}} \max\left(0,\, \Delta_{\mathrm{unstable}}\right)^2 + \lambda_{\mathrm{clear}} E_{\mathrm{clearance}}.$$

For pouring, the optimizer refines the rotation trajectory around the grasp frame so that the pouring vector points toward the receiving region while avoiding excessive wrist rotation or collision. For placement, the optimizer aligns the object with the support normal and adjusts the release height and retreat direction.

Scene-level trajectory optimization with DWB. For navigation and approach-pose annotation, trajectory generation first produces a coarse global path using A* on an occupancy or BEV map. This A* path is geometrically valid on the grid, but it may not respect the robot's embodiment dynamics. For example, an Ackermann platform has a minimum turning radius, a differential-drive base has coupled forward and angular velocities, while a quadruped or humanoid base can often be approximated as holonomic in the navigation layer. We therefore refine the A* path using a DWB-style local rollout optimizer.

Let the robot base state be

$$q_t = (x_t, y_t, \psi_t),$$

and let u_t be a velocity command. The admissible control set depends on the embodiment:

$$u_t \in \mathcal{U}_{\mathcal{R}}.$$

For a differential-drive base,

$$u_t = (v_t, \omega_t),$$

and the rollout model is

$$x_{t+1} = x_t + \Delta t\, v_t \cos\psi_t, \qquad y_{t+1} = y_t + \Delta t\, v_t \sin\psi_t, \qquad \psi_{t+1} = \psi_t + \Delta t\, \omega_t.$$

For an Ackermann platform,

$$u_t = (v_t, \delta_t), \qquad \dot{\psi}_t = \frac{v_t}{L}\tan\delta_t, \qquad |\delta_t| \le \delta_{\max},$$

where L is the wheelbase and δ_t is the steering angle. This naturally enforces a turning-radius constraint. For holonomic bases, such as quadruped or humanoid navigation abstractions, we use

$$u_t = (v_{x,t}, v_{y,t}, \omega_t),$$

which allows lateral motion in addition to rotation.

DWB samples short-horizon velocity commands and rolls them out under the corresponding dynamics. Each rollout is scored by

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

Here, J_{path} measures deviation from the A* path, J_{goal} measures distance to the target base pose, J_{obs} penalizes collision with obstacles, J_{clear} encourages clearance, J_{smooth} penalizes abrupt velocity changes, and J_{reach} evaluates whether the final base pose can support manipulation of the target object or part. The best rollout becomes the optimized local trajectory segment. Repeating this process along the path converts the A* seed into an embodiment-feasible approach trajectory.

Manipulation reachability during navigation optimization. For scene-level trajectories, reaching a base pose is not sufficient; the pose must also support the downstream manipulation target. We therefore add a reachability score to the DWB objective. Given a target manipulation anchor h and a candidate base pose q, we estimate whether the robot arm or body can reach the anchor:

$$J_{\mathrm{reach}}(q, h) = \min_{\xi \in \Xi(q)} \left\| \mathrm{FK}(\xi) - x_h \right\|^2 + \lambda_{\mathrm{ik}} \mathbf{1}_{\text{IK fail}},$$

where $\Xi(q)$ is the set of feasible arm configurations at base pose q, and x_h is the target position or pose induced by anchor h. In practice, this can be approximated using reachability maps, IK queries, or distance-to-workspace heuristics. This term favors base poses from which the robot can both see and manipulate the target part.

Candidate scoring and metadata. After optimization, each candidate receives an optimization score and metadata before physics validation. We store both the optimized action and the individual objective terms:

$$v_{\mathrm{opt}}^{(i)} = \left\{ E_{\mathrm{func}}, E_{\mathrm{task}}, E_{\mathrm{geom}}, E_{\mathrm{contact}}, E_{\mathrm{coll}}, E_{\mathrm{kin}}, E_{\mathrm{smooth}}, E_{\mathrm{total}} \right\}.$$

These values are later combined with simulation-based validation results. The diversity descriptor $d^{(i)}$ is also updated after optimization to record the final target location, approach direction, contact mode, trajectory family, base-pose class, or embodiment-specific command profile.

Discussion. The optimization stage is deliberately separated from physics validation. Optimization improves candidate quality under differentiable or efficiently computable objectives, while validation performs stricter checks such as simulation rollout, collision verification, IK success, traversability, contact stability, and task-specific success. This separation allows AnnotateAnything to scale to many candidates per anchor-skill pair while still retaining only physically feasible and functionally meaningful action annotations.

C.5 Details of Physics Validation

Physics validation is the final filtering stage before an optimized candidate is stored as an action annotation. Although candidate generation and trajectory optimization produce geometrically plausible actions, many candidates still fail when executed in simulation due to unstable contact, insufficient force closure, object penetration, articulation blockage, unreachable arm motion, or deformable-object entanglement. Therefore, validation is a critical step for improving downstream rollout success.

Given an optimized candidate

$$a_{h,s}^{(i)} = \left(s,\, o(h),\, h,\, \phi(h),\, x^{(i)},\, \theta_*^{(i)},\, \tau_*^{(i)},\, v^{(i)},\, d^{(i)}\right),$$

we run physics rollouts under a validation embodiment e_{val} and a perturbation setting Ξ:

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

The validation result $v^{(i)}$ records success flags, collision statistics, contact stability, task progress, robustness scores, rollout length, disturbance settings, and the embodiment used for validation. Candidates that fail the corresponding skill-specific checks are rejected.

Floating embodiment validation. For most object-centric skills, we use a **floating** embodiment, such as a floating parallel gripper or floating dexterous hand. The floating embodiment directly executes the optimized object-centric trajectory without attaching the gripper or hand to a specific robot arm or mobile base. This setting is used for skills where the main question is whether the local object interaction is physically meaningful, such as parallel-jaw grasping, dexterous grasping, insertion, hanging, placement, pouring, and many handle-based articulation candidates.

This design decouples object-level physical feasibility from a particular downstream robot configuration. In downstream data collection, robot-object relative poses may be randomized, and the same asset-level annotation can be consumed by different robot embodiments. Enforcing full-arm inverse kinematics during annotation would prematurely reject many locally valid interactions simply because one particular arm pose cannot reach them. Floating validation instead focuses on contact, penetration, local collision, object stability, task progress, and whether the action preserves the intended functional affordance $\phi(h)$.

For a parallel-jaw grasp, the floating gripper follows the pre-grasp trajectory, closes at the grasp pose, and then runs a stability test. For a dexterous grasp, the floating hand moves to the pre-grasp pose, closes according to the optimized joint configuration or contact schedule, and checks whether the object is actually constrained by the hand. For articulation, the floating gripper or hand grasps or contacts the handle and follows the generated opening, closing, rotating, or pushing trajectory. For vector-conditioned skills such as insertion or hanging, the floating end-effector follows the aligned approach and execution trajectory and verifies whether the object reaches a stable or task-complete state.

Non-floating or full-robot validation. Some skills are strongly embodiment-dependent and cannot be reliably validated with a floating gripper or hand. For these skills, we use a **non-floating** validation setting, including the full manipulator, arm, gripper or hand, and, when needed, the mobile base. This setting is used when IK feasibility, arm reachability, self-collision, singularities, or robot-body interaction directly determine task success.

We use full-manipulator validation for tasks such as garment pushing, washing, retrieval from entangled configurations, and other long-horizon deformable-object interactions. These tasks can fail even when the local hand trajectory looks reasonable, because the arm may pass through an IK singularity, become unreachable, collide with the environment, or entangle with the garment. In such cases, floating validation would overestimate feasibility. Full-robot validation therefore checks arm IK, joint limits, self-collision, arm-object collision, singular configurations, and task-specific deformable outcomes.

For navigation and approach-pose annotations, we use a base-aware non-floating setting. The A* path generated in the trajectory generation stage is checked and locally refined under the robot's base model, for example through DWB-style rollout. This allows the validation to account for embodiment-specific constraints such as differential-drive velocity limits, Ackermann turning radius, or holonomic base motion.

Disturbance and gravity robustness. For grasp and dexterous grasp candidates, nominal execution is often insufficient. A grasp can appear successful under one gravity direction but fail as soon as the object is perturbed or the gravity direction changes. To reject such fragile candidates, we apply randomized disturbances and gravity robustness tests during validation.

For each grasp or dexterous grasp candidate, we sample $K = 8$ random gravity directions:

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

where g_0 is the nominal gravity magnitude and $\mathbf{u}_k$ is a random unit direction. The gravity magnitude is scaled by 1.5 to create a stricter robustness test. We also apply small randomized disturbances, such as object pose jitter, hand pose jitter, or external perturbation forces, depending on the skill and simulator setting. A candidate is accepted only if it remains stable under the required robustness tests or exceeds the skill-specific pass threshold.

This gravity randomization is especially useful for detecting grasps with insufficient contact. For example, a parallel-jaw grasp may lift the object under the default gravity direction but drop it when gravity is rotated, indicating that the contact patch is too small or the object is only supported by incidental contact. Similarly, a dexterous grasp may visually appear to wrap around the object, but the object may not actually be constrained by the fingers; randomized gravity reveals such failures.

Skill-specific validation criteria. Each skill family uses task-specific success criteria in addition to generic collision and stability checks.

For parallel-jaw grasping, we check whether the gripper can approach without collision, close on the object, maintain sufficient contact, and keep the object stable under disturbance and randomized gravity. The object should not slip, fall, or rotate beyond the allowed tolerance during the stability rollout.

For dexterous grasping, we check hand joint validity, self-collision, hand-object penetration, fingertip or palm contact, and object stability. We also check whether the optimized grasp actually constrains the object rather than merely placing fingers near the object. This distinction is important because dexterous grasp candidates can look visually plausible while providing little or no effective contact.

For articulation, we check whether the end-effector maintains contact with the handle or moving part, whether the object joint progresses toward the intended open, close, rotate, or push state, and whether the gripper or hand becomes stuck in the geometry. For dexterous articulation, we additionally check hand joint limits and self-collision, since fingers can become trapped in narrow handles or deformed by contact with the moving part.

For insertion, hanging, placement, and pouring, we check alignment, clearance, stability after release, and task progress. For example, insertion candidates should reach sufficient insertion depth without collision; hanging candidates should remain stable under gravity after release; placement candidates should settle on the support surface; and pouring candidates should maintain the intended orientation and avoid collision.

For garment and deformable-object skills, we check full-arm reachability, IK stability, collision with the cloth or environment, and whether the resulting cloth state satisfies the intended geometric outcome. We reject trajectories that pass through singularities, cause the arm to move erratically, wrap the garment around the arm, or lead to severe entanglement.

For navigation, we check that the path is traversable, collision-free, connected to the target base pose, and compatible with the robot's base model. We also check manipulation reachability at the final base pose, since a navigation target is useful only if the robot can interact with the annotated object or part from that pose.

Parallel validation. Physics validation is computationally expensive because each asset can contain many anchors, each anchor can support multiple skills, and each anchor-skill pair can store many candidates. Moreover, the pass rate can be low, especially for dexterous grasping, articulation, and deformable-object tasks. We therefore run validation in parallel across assets, skills, anchors, candidates, and perturbation trials.

The validation jobs are grouped by skill and embodiment so that candidates with similar simulation settings can be batched together. For example, floating-gripper grasp candidates can be evaluated in one group, floating-dexterous-hand candidates in another group, and full-manipulator garment

trajectories in a separate group. Within each group, multiple candidates and multiple robustness trials can be executed concurrently. We also use early termination: if a candidate penetrates the object, loses contact, drops the object, fails to progress, or violates a joint limit, the rollout is stopped and marked as failed. Parallel validation is essential for scalability; without it, the number of simulation rollouts would become prohibitive.

Common failure cases. The validation stage rejects many candidates that appear plausible from geometry alone. Table 22 summarizes common failure modes observed across different skills.

Discussion. The separation between floating and non-floating validation is important for scalability and generality. Floating validation keeps the annotation object-centric and reusable across downstream embodiments, while full-robot validation is applied only when embodiment feasibility is intrinsic to the skill. Robustness tests, especially randomized gravity directions and disturbance rollouts, prevent the dataset from retaining fragile candidates that only succeed under a single idealized condition. Finally, parallel validation makes it practical to evaluate large candidate banks and retain high-quality, physically feasible, and functionally meaningful action labels.

C.6 Details of Physics-aware Augmentation

After physics validation, each accepted candidate is physically feasible under its original target, pose, trajectory, and validation setting. However, storing only the validated seed candidates would still limit action diversity. We therefore apply **physics-aware augmentation** to expand each candidate bank while preserving physical feasibility and functional consistency. The goal is to produce multiple valid variations of an action around the same functional anchor, rather than changing the semantic meaning of the action.

Given a validated candidate

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

we generate an augmented set

$$\mathcal{U}\left(a_{h,s}^{(i)}\right) = \left\{\tilde{a}_{h,s}^{(i,j)}\right\}_{j=1}^{M_i},$$

where each augmented candidate $\tilde{a}_{h,s}^{(i,j)}$ preserves the same skill s, visual anchor h, and functional affordance $\phi(h)$, but perturbs the concrete target, action parameters, or trajectory. The augmented candidate bank becomes

$$\widetilde{\mathcal{B}}_{h,s} = \mathcal{B}_{h,s} \cup \bigcup_{a_{h,s}^{(i)} \in \mathcal{B}_{h,s}} \mathcal{U}\left(a_{h,s}^{(i)}\right).$$

We use two complementary augmentation mechanisms: **local perturbation augmentation** and **symmetry-aware augmentation**.

Local perturbation augmentation. Local perturbation samples small variations around a validated anchor target or trajectory while keeping the action inside the same grounded region. This is useful because many skills admit a local family of feasible actions. For example, a mug handle can be grasped at slightly different contact centers and approach rotations; a garment sleeve fold can place the sleeve endpoint within a small region around the nominal mirrored target; and a cabinet handle can be pulled from slightly different handle points or approach directions.

For a target $x^{(i)}$, we sample a perturbed target

$$\tilde{x}^{(i,j)} = x^{(i)} + \Delta x^{(j)}, \qquad \Delta x^{(j)} \sim \mathcal{D}_x(h, s),$$

where $\mathcal{D}_x(h, s)$ is a bounded perturbation distribution defined by the anchor type and skill. For surface-based anchors, the perturbation is restricted to the part mask or affordance region. For keypoint-based anchors, the perturbation is restricted to a keypoint confidence region. For scene-level anchors, the perturbation is restricted to traversable free space or a reachable base-pose region.

For pose-based actions, we also perturb orientation and skill parameters:

$$\tilde{\theta}^{(i,j)} = \theta^{(i)} \oplus \Delta\theta^{(j)},$$

where $\Delta\theta^{(j)}$ may include in-plane grasp rotation, approach offset, gripper width adjustment, dexterous hand contact reassignment, insertion-axis offset, pulling-direction perturbation, or base-orientation perturbation. For trajectory-based actions, we perturb waypoint locations or timing:

$$\tilde{\tau}^{(i,j)} = \left\{ T_t \oplus \Delta T_t^{(j)} \right\}_{t=1}^{T}.$$

These perturbations are bounded to preserve the original task intent. For example, a cabinet-handle articulation candidate may perturb the handle contact point and pulling direction, but it must still move along the valid opening direction. Similarly, a functional mug-handle grasp may perturb the contact center and gripper yaw, but it should remain on the handle and preserve the grasp-for-use affordance.

Garment-specific local augmentation. Garment actions benefit from local perturbation because keypoint annotations and deformable-object outcomes naturally contain uncertainty. For fling, we jitter the selected grasp keypoints within local confidence regions around sleeve endpoints, bottom corners, or hem keypoints. The lift height, lateral stretch, and release location can also be randomly perturbed within a bounded range scaled by garment size. For folding, we perturb the fold line and place target. For example, in upper-body garment folding, the sleeve endpoint is first mirrored inward using the shoulder–bottom fold line; augmentation then samples nearby place points around this mirrored target. For bottom-up folding, the bottom-corner place points are jittered around the shoulder or upper-body target region. For pants, the side-fold target and bottom-up fold target are perturbed around the centerline and waistband region. These variations increase trajectory diversity while preserving the intended fold structure.

Grasp and dexterous grasp augmentation. For parallel-jaw grasping, we augment a validated grasp by perturbing the contact center, approach offset, gripper width, and in-plane rotation around the grasp axis:

$$\tilde{T}_{\text{grasp}} = T_{\text{grasp}} \oplus (\Delta p, \Delta R),$$

where Δp is constrained to remain within the graspable part or affordance region, and ΔR is bounded to avoid changing the grasp family. For dexterous grasping, we perturb palm pose, fingertip contact seeds, finger closure timing, and small joint offsets. These perturbations must preserve contact coverage, joint-limit feasibility, self-collision constraints, and functional-region consistency. For instance, dexterous mug-handle grasps remain on the handle, while body grasps remain labeled as physical or stabilizing grasps rather than functional grasp-for-use.

Articulation augmentation. For articulation, local perturbation is applied to the handle contact point, grasp orientation, pulling direction, waypoint spacing, and target joint range. This is important because a single handle can often be pulled from multiple nearby points and directions. However, the perturbation must preserve the articulation affordance: opening trajectories should still follow the joint motion, closing trajectories should move toward the closed state, and push-button trajectories should remain aligned with the button normal. Perturbed articulation candidates are checked for collision, contact maintenance, joint progress, and whether the gripper or dexterous hand becomes stuck in the handle geometry.

Navigation and approach-pose augmentation. For scene-level annotations, we perturb interaction-ready base poses and approach directions within reachable free-space regions. Given a base pose

$$q = (x, y, \psi),$$

we sample

$$\tilde{q} = (x + \Delta x, y + \Delta y, \psi + \Delta\psi),$$

where $(x + \Delta x, y + \Delta y)$ must remain in traversable free space and $\psi + \Delta\psi$ should keep the robot oriented toward the target object or affordance region. The augmented base pose is rechecked for clearance, path connectivity, DWB feasibility, and manipulation reachability. This produces multiple approach poses around the same object, enabling downstream policies to sample different viewpoints and approach directions.

Symmetry-aware augmentation. Local perturbation only explores nearby variations. To further expand diversity, we use symmetry-aware augmentation when visual-language annotations indicate that an object or part has a valid symmetry. Let $\mathcal{G}_{\mathrm{sym}}(o)$ denote the annotated symmetry group of object or part o. For a validated candidate, we generate transformed candidates

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

where the symmetry transform g is applied consistently to the target $x^{(i)}$, pose parameters $\theta^{(i)}$, and trajectory waypoints $\tau^{(i)}$. For pose-based actions,

$$\tilde{T} = GT,$$

where G is the rigid transformation induced by the symmetry. For trajectory-based actions,

$$\tilde{\tau} = \{GT_1, GT_2, \ldots, GT_T\}.$$

For vector-conditioned actions, the relevant vectors are also transformed:

$$\tilde{\mathbf{u}} = R_G \mathbf{u},$$

where R_G is the rotational component of G.

For example, a bottle or can may have approximate rotational symmetry around its vertical axis. If one grasp pose is validated, rotating the grasp around the symmetry axis can produce additional valid grasps. A round knob may support rotationally symmetric grasp or turning candidates. A drawer with repeated identical handles may support translated or mirrored handle grasps if the visual-language annotation marks the handles as functionally equivalent. Garments may also contain bilateral structure, such as left and right sleeves, allowing sleeve-fold templates to be mirrored across the garment centerline. However, symmetry augmentation is applied only when it preserves the functional affordance. A mug with a handle is not treated as fully rotationally symmetric for functional grasping, because rotating a handle grasp around the mug body would move it away from the handle and destroy the grasp-for-use affordance.

Feasibility checks for augmented candidates. Augmented candidates are not automatically accepted. Each candidate is rechecked using lightweight geometry and physics constraints, and high-risk skills can be sent back to full physics validation. The checks include:

$$\mathbb{I}_{\mathrm{accept}} = \mathbb{I}\left[C_{\mathrm{region}} \wedge C_{\mathrm{func}} \wedge C_{\mathrm{coll}} \wedge C_{\mathrm{kin}} \wedge C_{\mathrm{task}}\right],$$

where C_{region} checks that the perturbed target remains inside the correct part mask, affordance region, keypoint confidence region, or traversable scene region; C_{func} checks that the functional affordance $\phi(h)$ is preserved; C_{coll} checks collision and clearance; C_{kin} checks embodiment constraints such as gripper width, hand joint limits, or base dynamics; and C_{task} checks task-specific validity, such as grasp stability, articulation direction, fold geometry, insertion alignment, hanging support, or navigation reachability.

For inexpensive skills, such as grasp or navigation pose augmentation, we can generate many augmented candidates and filter them by geometry, collision, and short rollout checks. For expensive skills, such as full-manipulator garment trajectories, we use fewer augmentations and run stricter validation because small perturbations can produce IK singularities, arm-cloth entanglement, or large deviations in deformable-object state.

Examples. For a bottle, the visual-language annotation may mark the object as approximately rotationally symmetric around the vertical axis. A validated side grasp can then be rotated by multiple angles around this axis to produce additional grasp candidates:

$$G_\alpha = \mathrm{Rot}_z(\alpha), \qquad \alpha \in \left\{0, \frac{\pi}{4}, \frac{\pi}{2}, \ldots\right\}.$$

The transformed grasp poses are retained only if they remain collision-free and pass lightweight grasp-stability checks.

For a mug, symmetry-aware augmentation is more restrictive. Although the mug body may be approximately cylindrical, the handle breaks rotational symmetry for functional grasping. Therefore, a handle grasp is augmented by local perturbations on the handle, but it is not rotated around the mug

body unless the transformed pose still lies on a functionally equivalent handle region. This prevents augmentation from turning a functional grasp-for-use candidate into a merely physical body grasp.

For garment folding, local perturbation is more useful than global symmetry in many cases. Sleeve folding candidates are augmented by jittering the sleeve endpoint and mirrored place target within local confidence regions. Pants folding candidates are augmented by perturbing the centerline, side-fold target, and bottom-up fold target within bounded ranges. These perturbations create multiple folding trajectories with slightly different pickup and place points, which helps downstream policies learn robust behavior under keypoint noise and garment shape variation.

For articulation, a cabinet handle candidate can be augmented by jittering the handle contact point and pull direction. If the cabinet has repeated equivalent handles, symmetry-aware or copy-based augmentation can transfer a validated handle grasp or articulation trajectory to another handle. However, each transferred candidate must be checked against the local collision context, since an adjacent wall, shelf, or drawer frame may make the copied trajectory infeasible.

Discussion. Physics-aware augmentation increases annotation diversity while avoiding uncontrolled label noise. Local perturbation captures natural variation around a functional anchor, such as different grasp centers, fold points, pull directions, and base poses. Symmetry-aware augmentation exploits object-level or scene-level regularities, such as rotationally symmetric bottles or repeated cabinet handles. Both forms of augmentation are constrained by visual grounding, functional affordance, and physical feasibility checks. As a result, the expanded candidate banks contain diverse candidates that remain executable and functionally meaningful.

D Primitive-specific Action Annotation

Parallel-jaw and dexterous grasp annotation.

Articulation waypoint annotation.

Insertion and hanging annotation.

Garment and deformable-object trajectory annotation.

Navigation target and approach-pose annotation.

E Additional detail of Large-scale Robot Data Collection

Overview. Our large-scale data collection system consumes AnnotateAnything annotations as executable task interfaces, rather than relying on manually scripted object-specific demonstrations.

Atomic-skill library. We implement an annotation-aligned atomic-skill library covering table-top manipulation, bimanual manipulation, whole-body control, humanoid control, dexterous-hand manipulation, and mobile manipulation.

Skill composition. Each atomic skill maps compact annotation fields, such as grasp poses, target parts, waypoints, insertion directions, hanging anchors, and navigation goals, into controller or planner goals.

Long-horizon tasks. Long-horizon tasks are constructed by composing multiple atomic skills according to task templates and object-state transitions.

Heterogeneous asynchronous environments. Rollouts are generated in asynchronous parallel simulation environments, where each worker independently samples assets, tasks, robot embodiments, object poses, obstacles, and scene layouts.

Motion planning backend. For each rollout, we use cuRobo-v2 for goal-set IK, motion planning, and obstacle-aware execution, since it supports GPU-native collision-aware motion generation for both standard manipulators and high-DoF embodiments [36].

Domain randomization. We apply domain randomization over object pose, robot-object configuration, camera viewpoint, lighting, material, texture, distractor objects, and scene layout.

Candidate selection. Because randomization changes the relative pose between robot and object, each worker retrieves multiple candidates from the annotation bank, solves goal-set IK, and executes the feasible candidate with the lowest planning cost.

Trajectory validation. After execution, we validate task success, collision safety, contact consistency, articulation progress, and final-state stability when applicable.

Candidate filtering. Annotation candidates that repeatedly fail under randomized simulation are removed from the candidate bank to improve future rollout quality.

Scope. This data-collection system is used here to demonstrate that AnnotateAnything annotations are executable and scalable, while a full system-level analysis is left to our companion work.

E.1 Keypoint Generation and Affordance Grounding

Overview. AnnotateAnything naturally supports two related but different perception labels: keypoints and affordances. Keypoints are produced by the visual annotation stage, while affordances are derived from physics-validated interaction annotations. The distinction is important: keypoints describe sparse semantic or functional anchors on an object or scene, whereas affordances describe where and how a robot can execute an interaction. Thus, in our framework, keypoint generation is mainly a visual grounding problem, while affordance grounding is an executable-interaction grounding problem.

Keypoint generation. As described in Sec. 3.1.2, we generate keypoint annotations from fused 3D observations of the asset. Given a rendered RGB-D sequence, we first reconstruct a point cloud and sample diverse candidate points, e.g., using farthest point sampling. A VLM then selects points that are semantically or functionally important, such as handles, openings, corners, garment landmarks, or scene-level target locations. These selected keypoints provide sparse 3D anchors for downstream perception and action generation. They are not required to be directly executable by themselves; instead, they identify meaningful regions that later physics-based action annotation can refine and validate.

Affordance grounding from validated interactions. Affordance labels are generated from the physics-based action annotation pipeline in Sec. 3.2. Given visual anchors, the action pipeline generates candidate interactions and validates them through geometry checks, motion planning, collision checking, contact reasoning, and physics simulation. We therefore use the validated candidate bank as affordance supervision. Compared with manually annotated affordance ground truth, which often marks regions that are visually or semantically plausible, our labels indicate regions and directions that are executable under physical constraints. For robot learning, this can be a stronger supervision signal: the model is trained to predict not only where an interaction appears possible, but where it has been verified to succeed.

Why physics-validated affordances are useful. Manual affordance labels are usually subjective, sparse, and task-agnostic. For example, a human annotator may mark the handle of a mug or drawer as graspable, but the label may not specify which grasp pose, approach direction, contact mode, or motion trajectory is feasible for a robot. In contrast, our validated interaction annotations encode the full action context, including the target anchor, skill type, action direction, contact configuration, trajectory, and success metadata. They also preserve multiple valid solutions instead of collapsing the affordance to a single canonical point. This is especially important for manipulation, where many grasps, pulls, placements, or navigation poses can be valid for the same object.

Rigid objects. For rigid objects, affordance labels are represented by an interaction anchor together with action parameters from validated grasping or dexterous-grasping candidates. For parallel-jaw grasping, the label may include the grasp center, approach direction, gripper orientation, and width. For dexterous hands, the label may additionally include palm pose, finger contact locations, and

contact mode. These labels can be projected onto object point clouds or RGB-D images as affordance heatmaps, while the approach direction or grasp frame can be used as directional supervision.

Articulated objects. For articulated objects, affordance grounding includes both the contact anchor and the motion trajectory. For example, opening a drawer requires identifying the handle anchor and predicting a pulling trajectory, while rotating a door or knob requires an anchor together with a rotation-aware motion path. Thus, the affordance label is not only "where to touch", but also "how the interaction should move". This allows the same object part to support different task-conditioned affordances, such as pulling, pushing, rotating, or holding.

Garments and deformable objects. For garments and deformable objects, affordances are represented by pick-and-place keypoints and their associated manipulation trajectories. Examples include bimanual pick points for lifting or spreading, pick-place pairs for folding, and hanging points for placing garments on a support. Unlike rigid-object affordances, garment affordances are often relational: a pick point is meaningful only together with its paired place point, second-hand grasp point, or target folding line. We therefore represent garment affordance labels as structured point pairs or point sets rather than isolated single-point heatmaps.

Room-scale scenes and navigation affordances. For room-scale scenes, affordances are grounded on occupancy maps, BEV maps, and scene layouts. The labels include navigable regions, object-centric approach poses, interaction-ready base poses, and feasible navigation targets. These room-level affordances are useful for mobile manipulation because the robot must first reach a suitable base pose before executing object-level manipulation. Thus, scene affordance grounding connects global navigation with local interaction.

Projection to perception supervision. The generated labels can be converted into different supervision formats depending on the downstream model. For image-based models, 3D anchors, contact regions, and trajectories are projected into rendered RGB-D views to form keypoint targets, affordance masks, heatmaps, or directional fields. For point-cloud models, the same labels are attached directly to 3D points or local surface patches. For navigation models, room-level affordances are rasterized into BEV maps with traversability, target-pose, and interaction-readiness labels. When multiple validated candidates exist for the same anchor or skill, we keep the multi-modal label distribution rather than forcing a single ground-truth target.

Positive and negative supervision. Validated candidates provide positive affordance labels. Failed candidates can also be used as hard negatives when their failure is caused by physical infeasibility, such as collision, unstable contact, unreachable geometry, or invalid articulation motion. This provides richer supervision than manual affordance masks, since the model can learn the boundary between visually plausible but physically invalid regions and truly executable interaction regions. In practice, we treat negative labels conservatively, since some failures may depend on a particular embodiment, planner, or randomized scene configuration rather than the intrinsic affordance of the object.

Scope. We use this construction mainly to demonstrate that AnnotateAnything annotations can be repurposed as robot-centric perception supervision. A full benchmark of affordance or keypoint detectors is outside the main scope of this paper. The key point is that our annotation format already contains the necessary supervision: visual annotations provide semantic 3D keypoints, and physics-validated action annotations provide executable affordance labels across rigid objects, articulated objects, garments, and room-scale scenes.

E.2 Robot Reasoning VQA

Overview. Our trajectory generation pipeline records execution-level metadata during simulation rollout, which can be reused to construct robot reasoning VQA pairs. Unlike static asset-level QA, these questions focus on what the robot actually executes under a sampled scene, robot pose, and object configuration.

Runtime-grounded supervision. For each rollout, we record structured information such as the selected object, target part, anchor point, skill type, IK goal-set solution, approach pose, navigation

target, articulation direction, garment manipulation point, and task success state. These labels are determined at runtime because domain randomization changes the relative pose between the robot and the object. Thus, even for the same asset, the executed candidate may vary across rollouts.

QA construction. We convert the recorded trajectory metadata into question-answer pairs with VLM-assisted language generation. The VLM is used only to diversify and naturalize the question wording, while the answers are derived from simulator records and selected action candidates. This avoids treating the VLM prediction itself as ground truth.

Question categories. The resulting QA pairs cover several robot-centric reasoning types, including object and part selection, approach direction, navigation progress, articulation motion, garment manipulation landmarks, and execution success. For example, questions may ask which object or part the robot is interacting with, from which side the robot approaches a target, where the robot is navigating, which direction an articulated part is moved, or which garment landmark is selected for manipulation. These questions require trajectory-level execution traces and cannot be reliably answered from static asset annotations alone.

Scope. We use Robot Reasoning VQA as a lightweight downstream task to show that AnnotateAnything-generated rollouts naturally provide grounded multimodal reasoning supervision. A full VQA benchmark is outside the main scope of this paper.

E.3 3D VLM Instruction-tuning Data

Overview. AnnotateAnything can be used to construct lightweight instruction-tuning data for 3D VLMs. This data comes from two sources: simulation-grounded visual labels and language-based scene annotations. The goal is not to introduce a new 3D VLM benchmark, but to show that the generated annotations can be naturally repurposed into multimodal instruction data.

Simulation-grounded visual supervision. Because our assets and trajectories are generated in simulation, we can obtain dense ground-truth visual labels without manual annotation. These include 3D bounding boxes, projected 2D bounding boxes, instance segmentation masks, object poses, part identities, and object-instance associations. Such labels can be converted into grounding-style instruction-response pairs, where the model is asked to localize objects, identify instances, refer to parts, or associate 2D observations with 3D scene elements. The answers are derived from simulator metadata rather than from VLM predictions.

Language and spatial-reasoning supervision. The language annotations and cross-level composition in Sec. 3.1.3 provide another source of instruction data. Since object-level annotations are linked to room-level layouts through instance identities and 6D poses, we can generate QA pairs about object relations, room structure, scene layout, and task context. Typical questions involve 3D spatial relationships, such as relative position, containment, support, proximity, and object-to-room associations. This makes the data suitable for training models to reason over both object-level semantics and scene-level spatial structure.

Instruction construction. We convert structured annotations into instruction-response pairs using templates and VLM-assisted paraphrasing. Templates ensure that each answer is grounded in simulator or annotation metadata, while paraphrasing improves linguistic diversity. For example, grounding instructions may ask the model to locate an object or part, while spatial-reasoning instructions may ask about the relation between two objects or the functional area of a room. We avoid using the VLM as the source of ground truth; it is only used to diversify the language form.

Scope. This downstream task mainly demonstrates that AnnotateAnything provides reusable supervision for 3D multimodal learning. The same annotated assets can produce visual grounding labels from simulation and spatial-reasoning QA pairs from language and cross-level scene composition. A full-scale 3D VLM training and evaluation study is left outside the main scope of this paper.

F Related Work

F.1 Simulation-Based Automatic Data Collection

Prior work on simulation-based data generation can be divided into two broad families. The first family is annotation- or demonstration-based. RLBench generates large-scale expert demonstrations through waypoint-guided motion planning across many tasks [59]. MimicGen amplifies a small number of human demonstrations by recomposing subtask segments across new scenes, object instances, and robot embodiments [60]. More recent systems such as RoboGen and GenSim use foundation models to propose tasks, synthesize scenes, and generate supervision or task code automatically [61, 62]. These methods are attractive because they preserve a relatively interpretable supervision signal and often produce demonstrations that are closer to human decomposition of manipulation tasks.

The second family is RL-based automatic data collection. Here, the simulator itself is the generator: environments such as Meta-World and Isaac Gym scale data through massive parallel interaction, while dexterous manipulation systems trained entirely in simulation demonstrate that difficult manipulation can be learned without requiring human demonstrations [63–65]. The advantage is scale through optimization, but the cost is high engineering burden in reward design, curriculum design, and training stability, and the resulting supervision is often controller-centric rather than human-readable or reusable as annotation.

A persistent gap across both families is that annotation-based systems usually depend on either simple tasks, a limited library of scripted decompositions, or scarce seed demonstrations, while RL-based systems can scale interaction but rarely produce portable semantic annotations that transfer across embodiments and task families. In other words, prior systems either generate *data for a fixed training setup* or *policies for a fixed task formulation*, rather than a reusable annotation layer that can support many downstream learners.

Connection to AnnotateAnything. AnnotateAnything is motivated by exactly this gap: instead of choosing between narrow annotation reuse and reward-heavy RL optimization, it aims to generate reusable manipulation annotations that can bootstrap many downstream data-generation and policy-learning pipelines.

F.2 Automatic Manipulation Annotation and Affordance Generation

Automatic manipulation annotation has been studied extensively in grasping. Dex-Net 2.0 showed that robust grasp supervision can be synthesized at scale with analytic metrics in simulation [66]. ACRONYM further expanded large-scale simulation-based grasp labels [67], while 6-DOF GraspNet and Contact-GraspNet demonstrated learned grasp proposal generation directly from point clouds and cluttered observations [68, 69]. For dexterous manipulation, DexGraspNet substantially increased the availability of robot-hand grasp labels over diverse objects [70]. Very recent work also begins to address bimanual dexterous grasp synthesis, but this remains significantly less mature than single-handed and parallel-gripper grasp annotation [71].

Beyond grasping, affordance generation has been explored for articulated interaction, object-object relations, and task-specific manipulation. Where2Act and VAT-Mart learn where and how to interact with articulated objects from simulated interaction data [72, 73]. GAPartNet argues that generalizable and actionable parts are a better transfer unit than object categories alone, providing part semantics and part poses that bridge perception and manipulation [74]. O2O-Afford studies object-object affordance without manual labels, covering behaviors such as placement and fitting [75]. OmniHang addresses hanging as a dedicated contact- and correspondence-centric problem [76], while assembly systems such as IndustReal focus on insertion and contact-rich assembly through task-specific learning and simulation design [77]. 3D AffordanceNet provides a benchmark for visual affordance understanding in 3D, but still within a fixed affordance ontology [78].

Taken together, these works reveal a recurring tradeoff. Optimization-based annotation pipelines often provide physically grounded labels but are usually tied to a specific manipulation family, gripper model, contact formulation, or embodiment. Learned affordance models expand the label space and can capture richer semantics, but most are trained for a narrow set of task families, a narrow embodiment range, or a fixed affordance vocabulary. As a result, the field contains many strong

single-skill annotation engines, but very few systems that can annotate many manipulation skills across many embodiments in a unified way.

Connection to AnnotateAnything. AnnotateAnything is best positioned as a general annotation substrate above task-specialized affordance systems: it seeks to unify grasp-, articulation-, placement-, insertion-, and hanging-style supervision under one scalable pipeline rather than introducing one more narrow annotation engine.

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

Large-scale 3D asset repositories underpin almost all recent simulation-based manipulation work. ShapeNet established the modern large-scale CAD repository paradigm [79], PartNet added fine-grained and hierarchical part annotations [80], and PartNet-Mobility extended the ecosystem with articulated joints and motion metadata through the SAPIEN dataset stack [81, 82]. Objaverse dramatically increased asset scale and diversity by aggregating hundreds of thousands of annotated 3D objects from the web [83]. For manipulation-specific repositories, ACRONYM and DexGraspNet provide grasp annotations, while GAPartNet adds action-relevant part semantics and poses [67, 70, 74].

A parallel thread focuses on generative asset creation. DreamFusion and Shap-E demonstrate that text-conditioned 3D generation can produce novel 3D assets without manual modeling [84, 85]. These methods are promising for simulation scale-up, but they usually do not provide the robot-ready metadata that manipulation systems need: articulated joints, functional parts, interaction affordances, contact semantics, or embodiment-aware trajectories.

The central gap is therefore not simply the amount of 3D content. The field already has abundant static geometry and increasingly capable generative models. What remains scarce is a pipeline that can transform raw or generated assets into *interaction-ready annotations* suitable for manipulation learning, benchmark construction, and skill composition.

Connection to AnnotateAnything. AnnotateAnything can be framed as the missing conversion layer from assets to actionable supervision: it complements repositories and text-to-3D generators by adding the manipulation semantics that raw geometry alone does not provide.

F.4 Simulation Environments and Manipulation Benchmarks

Robot learning has benefited from a rich simulator ecosystem. MuJoCo remains a dominant engine for fast rigid-body control and benchmarked continuous-control research [86]. PyBullet offers an accessible open-source simulation stack with broad adoption in robotics and reinforcement learning practice [87]. Isaac Gym introduced end-to-end GPU-resident simulation for massive parallel robot learning [64], and Isaac Sim extends the Isaac stack toward richer industrial and photorealistic simulation workflows [88]. SAPIEN focuses on interactive articulated-object simulation and has become especially influential for part-aware and household manipulation research [82]. RoboSuite provides a modular MuJoCo-based research framework for manipulation [89].

On top of these platforms, benchmark suites define the evaluation targets. Meta-World emphasizes multitask and meta-RL over tabletop manipulation skills [63]. RLBench provides a large task library with automatically generated demonstrations [59]. ManiSkill and ManiSkill2 move toward more diverse objects, richer demonstrations, and a broader range of manipulation modes and embodiments [90, 91]. BEHAVIOR broadens the focus to everyday household activities and long-horizon embodied tasks [92].

These environments and benchmarks are foundational, but they expose a systems-level fragmentation. Some emphasize control fidelity, others articulated assets, others broad task coverage, and others long-horizon embodied realism. What they usually do *not* provide is a unified mechanism for generating transferable task annotations across assets, embodiments, and manipulation families. In practice, the benchmark and simulator literature still relies heavily on manual task authoring, limited affordance ontologies, or benchmark-specific annotation choices.

Connection to AnnotateAnything. AnnotateAnything can be presented as complementary infrastructure for this ecosystem: rather than replacing simulators or benchmarks, it aims to make them easier to populate with interaction-ready annotations at larger scale and with greater task diversity.

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

A separate but highly relevant line of work studies how robots can compose reusable skills. The options framework is the classic formal foundation for temporally extended actions and hierarchical decision making [93]. In robotics, recent systems operationalize this idea with explicit or implicit skill libraries: SayCan selects among a predefined set of robotic skills by combining language-model usefulness with affordance-grounded feasibility [94], while Code as Policies composes control APIs and primitive calls through language-model-generated programs [95]. Composable Part-Based Manipulation further pushes toward reusable, object-part-centered manipulation abstractions [96].

What all of these systems have in common is that they assume some skill inventory, primitive API, or part-level grounding already exists. In long-horizon robot learning, the bottleneck often shifts from policy optimization to *grounding*: how to obtain the contact points, manipulable regions, trajectories, constraints, or object-part correspondences that instantiate a skill on a new object. Skill composition is therefore only as scalable as the annotation layer beneath it.

Connection to AnnotateAnything. AnnotateAnything directly supports skill-centric learning by making the atomic grounding of skills more scalable: if reusable annotations can be generated automatically, then skill libraries and hierarchical planners can be expanded far beyond hand-authored primitive sets.

NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and follow the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes], [No], or [N/A].
- [N/A] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for [N/A]).

The checklist answers are an integral part of your paper submission. They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will also be asked to include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While [Yes] is generally preferable to [No], it is perfectly acceptable to answer [No] provided a proper justification is given (e.g., error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering [No] or [N/A] is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

IMPORTANT, please:

- **Delete this instruction block, but keep the section heading "NeurIPS Paper Checklist"**,
- **Keep the checklist subsection headings, questions/answers and guidelines below.**
- **Do not modify the questions and only use the provided macros for your answers**.

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: **[TODO]**

   Justification: **[TODO]**

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: **[TODO]**

   Justification: **[TODO]**

Guidelines:

- The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: **[TODO]**

   Justification: **[TODO]**

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: **[TODO]**

   Justification: **[TODO]**

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.

- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: **[TODO]**

   Justification: **[TODO]**

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).

- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: **[TODO]**

   Justification: **[TODO]**

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: **[TODO]**

   Justification: **[TODO]**

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
   - If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: **[TODO]**

   Justification: **[TODO]**

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.

- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: **[TODO]**

   Justification: **[TODO]**

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: **[TODO]**

    Justification: **[TODO]**

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: **[TODO]**

    Justification: **[TODO]**

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.

- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: **[TODO]**

    Justification: **[TODO]**

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: **[TODO]**

    Justification: **[TODO]**

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: **[TODO]**

    Justification: **[TODO]**

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: **[TODO]**

    Justification: **[TODO]**

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: **[TODO]**

    Justification: **[TODO]**

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.

Table 7: Example instantiations of the unified action annotation schema for a cabinet or drawer. The handle anchor carries the functional affordance of opening, while scene-level anchors support navigation and approach-pose annotation.

Symbol	Cabinet example	Interpretation
$\mathcal{A}$	Cabinet scene	The input asset or scene, including cabinet doors, drawers, handles, panels, and surrounding free space.
h_{handle}	Cabinet handle	A part-level anchor from $\mathcal{P}(\mathcal{A})$ and a functional affordance anchor. It can support both grasping and articulated manipulation.
$\phi(h_{\text{handle}})$	Grasp-for-opening	The functional affordance of the handle. It indicates that grasps on this anchor should support opening or closing the door or drawer.
s_{grasp}	Parallel-jaw grasp	A compatible skill in $\mathcal{S}(h_{\text{handle}})$. This skill instantiates grasp poses for holding the cabinet or drawer handle.
$\mathcal{B}_{h_{\text{handle}},s_{\text{grasp}}}$	Handle grasp bank	A set of feasible grasp candidates on the handle, including different contact points, gripper poses, approach directions, and gripper widths. These candidates are conditioned on the handle's role as an opening affordance.
$a_{h,s}^{(i)}$	One handle grasp	One concrete grasp candidate. Here, $x^{(i)}$ is a contact point or patch on the handle, $\theta^{(i)}$ stores the 6D grasp pose and gripper width, $\tau^{(i)}$ stores the approach trajectory, $v^{(i)}$ records IK, collision, and functional feasibility, and $d^{(i)}$ records approach direction and grasp family.
s_{art}	Articulation	Another compatible skill in $\mathcal{S}(h_{\text{handle}})$. The same handle anchor can instantiate opening or closing actions for a door or drawer.
$\mathcal{B}_{h_{\text{handle}},s_{\text{art}}}$	Articulation bank	A set of feasible articulation candidates, including different pull directions, waypoint sequences, contact-maintenance strategies, and motion ranges.
$a_{h,s}^{(i)}$	One opening trajectory	One concrete articulation candidate. Here, $x^{(i)}$ is the handle interaction point, $\theta^{(i)}$ stores the prismatic or revolute motion direction, $\tau^{(i)}$ contains pull or rotate waypoints, $v^{(i)}$ records collision status, contact maintenance, actuation success, and functional success, and $d^{(i)}$ records pull direction, waypoint spacing, and trajectory family.
h_{free}	Free-space region	A scene-level anchor from $\mathcal{R}^{\text{scene}}(\mathcal{A})$, corresponding to reachable space in front of the cabinet.
$\phi(h_{\text{free}})$	Navigation approach	The functional affordance of the free-space region. It indicates that this region supports approaching the cabinet for subsequent manipulation.
s_{nav}	Navigation	A compatible skill in $\mathcal{S}(h_{\text{free}})$. This skill generates interaction-ready base poses and approach directions.
$\mathcal{B}_{h_{\text{free}},s_{\text{nav}}}$	Navigation bank	A set of feasible base-pose candidates from which the robot can approach and manipulate the cabinet, with different distances, orientations, viewpoints, and approach sides.
$a_{h,s}^{(i)}$	One base pose	One concrete navigation candidate. Here, $x^{(i)}$ is a target base pose, $\theta^{(i)}$ stores base orientation and approach direction, $\tau^{(i)}$ stores an optional collision-free path, $v^{(i)}$ records path feasibility, manipulation reachability, and functional approach validity, and $d^{(i)}$ records viewpoint, clearance, and distance-to-object.

Table 8: Examples of candidate region construction for different skills. Candidate regions are obtained from language annotations, visual grounding, and asset geometry before concrete target anchors are sampled.

Skill	Annotation source	Candidate region	Example interpretation
Functional grasp	Part mask + language	Mug handle, tool handle	The language annotation indicates that the handle is used for holding, pouring, or operating the object. Targets sampled from this region are functional grasp candidates.
Physical grasp	Part mask + geometry	Mug body, box side, object surface	The region is physically graspable but not necessarily functionally preferred. For example, grasping a mug body may be stable for pickup but less suitable for pouring.
Dexterous grasp	Part mask + affordance region	Handle, body surface, functional contact area	The region supports multi-finger contacts. Candidates differ in fingertip locations, palm poses, and hand configurations.
Garment manipulation	Semantic keypoints	Cloth corners, edges, sleeve endpoints	Keypoints directly seed grasp anchors or bimanual keypoint pairs for lifting, folding, stretching, or placing.
Articulation	Part mask + affordance region	Cabinet handle, drawer handle, door edge	The handle or movable part is selected as the region from which pulling, pushing, or rotating anchors are sampled.
Insertion	Affordance region + geometry	Opening, socket, hole, container mouth	The opening region provides entry points, alignment frames, and insertion directions.
Hanging	Affordance region + edge geometry	Hook, support edge, handle loop	Support regions are sampled to generate hanging contacts and gravity-stable poses.
Navigation	Occupancy / BEV map	Reachable free-space region	Traversable cells around a target object are sampled to generate interaction-ready base poses and approach directions.

Table 9: Examples of sampling and filtering rules for candidate target generation. These filters convert grounded candidate regions into feasible target anchors for candidate banks.

Skill	Sampling method	Filtering criteria	Example rejected targets
Functional grasp	FPS on functional part mask	Part size, gripper width, surface normal, collision margin, reachability, functional affordance	A tiny handle decoration that is too small to grasp; a mug-body grasp when the task requires pouring from the handle.
Physical grasp	FPS on graspable surfaces	Contact area, antipodal geometry, local curvature, gripper clearance, IK feasibility	A highly curved or sharp surface patch that cannot support stable contact.
Dexterous grasp	FPS or contact-set sampling on part masks	Hand joint limits, fingertip support, self-collision, object collision, contact stability	A contact set that requires impossible finger extension or causes palm-object penetration.
Garment manipulation	Keypoint sampling and perturbation	Visibility, bimanual reachability, corner confidence, pair distance, deformation feasibility	Two cloth keypoints that are too far apart for bimanual grasping or are occluded.
Articulation	FPS on handle or movable-part mask	Motion-axis consistency, contact maintenance, collision, reachability, actuation direction	A handle point that can be grasped but cannot be pulled along the door or drawer motion direction.
Insertion	Sampling on opening region	Opening size, entry clearance, alignment direction, collision-free approach, insertion depth	An opening that is detected visually but too narrow for the inserted object.
Hanging	Sampling on hooks or support edges	Edge orientation, gravity stability, clearance, collision, contact support	A support edge that faces downward or lacks clearance for the hanging object.
Navigation	Grid sampling or 2D FPS on BEV free space	Traversability, clearance, path connectivity, visibility, manipulation reachability	A base pose that is collision-free but too far to reach the cabinet handle.

Table 10: Trajectory generation families grouped by the source of constraints. Each generator converts a sampled target anchor into an initial trajectory or waypoint sequence.

Constraint family	Representative skills	Main inputs	Generated trajectory
Object-property-conditioned	Grasp, dexterous grasp, garment fling, garment fold	Object geometry, part masks, keypoints, garment length, grasp frame	Pre-grasp, contact, closure, retreat, bimanual garment motion, or keypoint-driven folding trajectory.
Object-trajectory-conditioned	Open, close, rotate, push articulated parts	Articulation axis, joint state, moving-part transform, handle pose	End-effector waypoints that follow the moving part while preserving grasp or contact constraints.
Object-vector-conditioned	Place, hang, pour, insert	Surface normal, insertion axis, hanging direction, pouring direction, active/passive vectors	Approach and execution waypoints aligned with a semantic or geometric vector.
Scene-trajectory-conditioned	Navigation, approach-pose generation, retrieval-style motion	Occupancy map, BEV map, traversable free space, target object region	Base-pose sequence, collision-free approach path, and interaction-ready base trajectory.

Table 11: Garment trajectory templates generated from semantic keypoints. These trajectories are template-level seeds and are later optimized and validated.

Garment skill	Keypoints	Geometric rule	Generated trajectory
Sleeve fold	Sleeve endpoint, shoulder, bottom corner	Use shoulder–bottom line as fold line; mirror sleeve endpoint inward.	Pick sleeve endpoint, lift, move to mirrored target, place inside garment body.
Bottom-up fold	Bottom corners, shoulder points	Move bottom keypoints toward the shoulder line.	Bimanual pick of bottom corners, lift, fold upward toward shoulder region, place.
Pants side fold	Left/right leg or side keypoints, central axis	Fold the two side or leg regions toward the centerline.	Pick side or leg keypoints, lift, move inward, place to overlap legs.
Pants compact fold	Cuff/bottom keypoints, waistband region	Fold bottom or cuff region upward toward waistband; optionally repeat.	Pick bottom/cuff keypoints, fold upward, then perform a second compacting fold.
Fling	Two garment keypoints, garment length	Scale lift height and stretch distance by keypoint distance or garment length.	Bimanual lift, stretch, optional shake, and release or place trajectory.

Table 12: Trajectory templates for articulated-object actions. These templates differ mainly in the joint schedule, contact mode, and whether the end-effector maintains a grasp or pushes on the part.

Action type	Subcase	Trajectory rule
Open with handle	Handle open	Grasp the handle, preserve T_{rel}, and increase the revolute or prismatic joint state from closed to open.
Partially open pull	Partial pull	Grasp the handle and follow the same relative-transform rule, but stop at a partial target joint state q_{partial}.
Close with handle	Handle close	Grasp the handle and reverse the joint schedule, moving from the current joint state toward the closed state.
Push close	Collision close / push close	Contact the door, drawer, or lid surface without necessarily forming a handle grasp; push along the closing direction while maintaining collision-free contact.
Open lid	Lid open	Follow the lid's revolute joint trajectory while preserving the end-effector–lid relation or maintaining a pushing contact.
Close lid	Lid close	Reverse the lid joint trajectory and push or guide the lid toward the closed state.
Rotate button	Button rotate	Generate rotational waypoints around the button axis, with the end-effector orientation following the required rotation.
Rotate lid	Lid rotate	Rotate the lid around its axis, using angular waypoints and contact-maintenance constraints.
Push button	Button push	Generate a short linear trajectory along the button normal or push direction, with displacement bounded by the button travel range.

Table 13: Object-vector-conditioned trajectory templates. These actions are generated by aligning the end-effector or object with a semantic or geometric vector.

Skill	Vector source	Alignment rule	Generated trajectory
Place	Surface normal and pose range	Align object placement pose with support normal.	Approach placement pose, lower along normal, release, and retreat.
Hang	Support edge, hook direction, gravity	Align hanging contact with support edge and gravity-stable direction.	Approach hook or edge, establish contact, lower to stable hanging pose, release.
Pour	Active pour vector and receiving direction	Rotate object so active pour vector points toward target container or receiving region.	Lift, rotate around grasp frame, pour for a specified angular range, return or retreat.
Insert	Active insertion axis and passive socket axis	Align active axis with passive axis and translate along insertion direction.	Approach socket, align, move along insertion axis, optionally retreat or release.

Table 14: Trajectory optimization objectives for different skill families. Optimization refines the template trajectory produced by the previous stage into an asset-specific and embodiment-aware candidate.

Skill family	Optimized variables	Main objectives	Examples
Dexterous/contact skills	Hand pose, finger joints, contact assignment, approach pose	Contact alignment, functional-region consistency, wrench support, joint limits, self-collision, hand-object collision	Dexterous grasp on a mug handle, whole-hand grasp, bimanual grasp.
Adaptive object skills	Waypoint positions, placement targets, timing, approach offsets	Geometry adaptation, keypoint consistency, length/scale adaptation, bimanual reachability, smoothness	Garment sleeve fold, bottom-up fold, pants fold, fling.
Articulated-object skills	End-effector waypoints, contact pose, joint schedule, pull/push direction	Joint-motion tracking, contact maintenance, collision avoidance, reachability, smoothness	Open/close cabinet, pull drawer, rotate button, push button.
Vector-conditioned skills	Approach pose, alignment pose, execution direction, release pose	Vector alignment, insertion depth, support stability, placement normal consistency, collision-free approach	Insert, hang, pour, place.
Scene-level skills	Base path, velocity commands, base orientation, approach pose	Path following, clearance, traversability, local dynamics, turning radius, manipulation reachability	Navigation to cabinet, approach-pose generation, room-scale interaction.

Table 15: Examples of articulation trajectory refinement. The optimizer adapts template waypoints to contact, collision, reachability, and joint-motion constraints.

Action	Optimization focus	Typical adjustment
Open with handle	Maintain grasp while following revolute or prismatic motion	Adjust pull direction, waypoint spacing, and wrist orientation to preserve contact and avoid collision.
Partially open pull	Stop at a partial target state	Shorten the joint schedule while preserving a feasible pull trajectory.
Close with handle	Reverse opening trajectory	Adapt closing waypoints to avoid collision with the cabinet frame and maintain grasp.
Push close	Maintain pushing contact without grasp	Align push direction with closing motion and regulate contact force and clearance.
Open/close lid	Follow lid rotation	Adjust end-effector orientation and approach side to avoid collision with the lid and surrounding object.
Rotate button/lid	Follow angular motion	Refine angular waypoints and wrist orientation around the rotation axis.
Push button	Linear push along button normal	Bound displacement by travel range and avoid side contact.

Table 16: Examples of vector-conditioned trajectory optimization. These skills are refined by improving vector alignment, clearance, depth, and stability.

Skill	Optimized variables	Objective	Example adjustment
Insert	Alignment pose, insertion depth, approach offset	Axis alignment, entry clearance, collision-free insertion	Rotate the object to align with the socket axis and reduce insertion depth if collision occurs.
Hang	Approach pose, support contact, release pose	Support-edge alignment, gravity stability, clearance	Shift the placement pose so the object lowers onto the hook without colliding.
Pour	Wrist rotation, pouring angle, target direction	Pour-vector alignment, wrist feasibility, collision avoidance	Adjust pouring angle so the liquid direction points toward the receiving container.
Place	Placement pose, release height, retreat direction	Surface-normal alignment, stable placement, collision-free retreat	Lower along the support normal and retreat without sweeping nearby objects.

Table 17: Embodiment-aware navigation optimization. The A* path from trajectory generation is refined by DWB-style local rollout using the robot's motion model.

Embodiment	Control model	Main constraints	Optimization effect
Differential drive	(v, ω)	Velocity bounds, angular velocity bounds, clearance, path tracking	Smoothly follows the A* path while respecting coupled forward and rotational motion.
Ackermann drive	(v, δ)	Steering limit, minimum turning radius, acceleration bound, clearance	Removes sharp A* turns that are infeasible for car-like platforms.
Holonomic base	(v_x, v_y, ω)	Velocity bounds, lateral motion limits, clearance, manipulation reachability	Uses lateral motion to approach interaction targets from better viewpoints or reachability zones.
Quadruped / humanoid abstraction	Holonomic or quasi-holonomic command	Traversability, body clearance, foothold or stance feasibility if available	Produces approach poses that remain compatible with whole-body navigation and manipulation.

Table 18: Metadata stored after trajectory optimization. These fields support candidate ranking, debugging, validation, and downstream sampling.

Metadata field	Meaning	Example
Optimized parameters	Final skill-specific parameters $\theta_*^{(i)}$	Dexterous hand pose and joints, insertion alignment pose, hanging pose, base orientation.
Optimized trajectory	Final waypoint sequence $\tau_*^{(i)}$	Approach-contact-retreat path, articulated opening waypoints, DWB-refined navigation path.
Objective breakdown	Individual optimization terms	Collision cost, contact cost, reachability cost, smoothness cost, functional consistency cost.
Embodiment feasibility	Whether the optimized candidate satisfies robot-specific constraints	Joint-limit status, turning-radius feasibility, pre-grasp reachability, local base dynamics feasibility.
Functional consistency	Whether the optimized candidate preserves $\phi(h)$	Mug-handle grasp remains grasp-for-use; cabinet-handle grasp remains grasp-for-opening.
Diversity descriptor	How this candidate differs from others	Different approach direction, contact set, fold target, base-pose side, navigation route, or trajectory family.

Table 19: Validation embodiments used for different skill families. Floating validation checks object-centric physical feasibility, while non-floating validation checks embodiment-dependent feasibility.

Validation setting	Representative skills	What is checked	Motivation
Floating gripper	Parallel-jaw grasp, insertion, hanging, placement, handle articulation	Contact, collision, penetration, stable grasp, task progress, release stability	Validates local object interaction without over-constraining the annotation to a particular arm pose.
Floating dexterous hand	Dexterous grasp, dexterous handle grasp, dexterous articulation	Finger-object contact, hand-object collision, hand joint validity, object stability, contact maintenance	Tests whether a dexterous contact configuration actually constrains the object before downstream execution.
Full manipulator	Garment push, wash, retrieval, entangled deformable-object skills	IK, joint limits, singularities, arm collision, self-collision, garment-arm entanglement, deformable outcome	Needed when feasibility depends on the arm and body, not only local hand-object contact.
Base-aware validation	Navigation, approach-pose annotation, mobile manipulation setup	Traversability, local dynamics, clearance, turning radius, path connectivity, manipulation reachability	Ensures that the sampled base pose and path are feasible for the target robot base.

Table 20: Robustness tests used during physics validation. Gravity randomization is mainly applied to parallel-jaw and dexterous grasp candidates.

Robustness test	Applied skills	Purpose
Random gravity directions	Parallel-jaw grasp, dexterous grasp	Reject grasps that only succeed under one gravity direction but fail when the load direction changes.
Gravity magnitude scaling	Parallel-jaw grasp, dexterous grasp	Use $1.5g_0$ to create a stricter stability test for weak or marginal contacts.
Object pose jitter	Grasp, dexterous grasp, insertion, hanging	Reject candidates that require extremely precise initialization or have no robustness margin.
Hand or gripper pose jitter	Grasp, dexterous grasp, articulation	Test whether small execution errors cause collision, contact loss, or task failure.
External perturbation force	Grasp, dexterous grasp, hanging, placement	Check whether the object remains stable under small disturbances after contact or release.
Trajectory perturbation	Garment, articulation, vector-conditioned skills	Detect candidates that are overly sensitive to waypoint timing, contact pose, or approach direction.

Table 21: Skill-specific validation criteria. A candidate must satisfy both generic physics checks and task-specific success conditions.

Skill family	Validation criteria	Typical success signal
Parallel-jaw grasp	Approach collision, closure contact, object stability, gravity robustness, disturbance robustness	Object remains held under randomized gravity and perturbation.
Dexterous grasp	Joint validity, self-collision, hand-object penetration, contact coverage, object stability, gravity robustness	Object is actually constrained by the hand and does not slip or fall.
Articulation	Contact maintenance, joint progress, collision, penetration, handle blockage, grasp or push validity	Door, drawer, lid, or button reaches the intended joint state without contact loss or jamming.
Insertion	Axis alignment, entry clearance, collision-free approach, insertion depth, post-insertion stability	Object reaches the target opening or socket without collision and remains aligned.
Hanging	Support contact, gravity stability, release outcome, clearance around hook or edge	Object remains hanging after release and does not slide or fall.
Garment / deformable	Full-arm IK, singularity avoidance, arm-cloth collision, entanglement, geometric outcome	Cloth reaches the intended fold, push, wash, or retrieval state without wrapping around the arm.
Navigation	Traversability, clearance, local dynamics, path connectivity, final-pose reachability	Robot reaches an interaction-ready base pose from which manipulation is feasible.

Table 22: Common failure cases detected by physics validation. These failures motivate the use of simulation-based filtering instead of relying only on geometric candidate generation.

Skill	Failure case	Cause	How validation detects it
Parallel-jaw grasp	Object drops under different gravity directions	The gripper has too little contact area, contacts only an edge, or does not form a stable antipodal grasp.	The grasp may pass nominal lifting but fails under one of the eight randomized gravity directions or disturbance rollouts.
Parallel-jaw grasp	Immediate gripper-object collision	The pre-grasp or approach pose intersects the object or nearby geometry.	Collision and penetration statistics exceed the tolerance before closure.
Dexterous grasp	Looks like a grasp but does not actually hold the object	The hand pose surrounds the object visually, but fingertips do not establish effective contacts or contact forces.	The object slips or falls during gravity robustness tests; contact coverage or stability score is too low.
Dexterous grasp	Finger penetration or self-collision	The optimized joint configuration places fingers inside the object or causes hand self-intersection.	Hand-object penetration, self-collision, or joint-limit violation is detected during rollout.
Dexterous articulation	Hand gets stuck in the handle or part geometry	Fingers enter a narrow handle, cavity, or gap and cannot move with the articulated part.	Contact maintenance fails, joint motion stalls, or penetration grows during opening or closing.
Dexterous articulation	Dexterous hand joint deformation or damage in simulation	The moving object part pushes against fingers or joints in an infeasible configuration.	Joint limits, self-collision, abnormal contact forces, or unstable hand motion are detected.
Articulation with floating gripper	Gripper jams into the object or cabinet frame	The handle trajectory is geometrically plausible but the gripper body collides with surrounding geometry.	Articulation progress stops, collision increases, or the gripper cannot maintain contact.
Insertion	Detected opening is too narrow or misaligned	The visual affordance is correct, but the insertion axis or clearance is insufficient.	Trajectory collides near the entrance or fails to reach insertion depth.
Hanging	Object falls after release	The support edge or hook does not create a gravity-stable contact.	After release, the object slides, rotates away, or drops under gravity.
Garment push / wash	Trajectory passes through IK singularity and moves erratically	The local end-effector path is valid, but the full arm has unstable or near-singular IK.	Full-manipulator validation shows large joint jumps, unstable end-effector motion, or IK failure.
Garment / deformable	Garment wraps around the arm or gripper	The trajectory sweeps through the cloth or pulls it across the robot body.	Arm-cloth collision, cloth entanglement, or task-outcome failure is detected.
Navigation	A* path is geometrically valid but dynamically infeasible	The grid path has sharp turns or narrow passages incompatible with the robot base.	DWB or base-aware rollout fails due to turning radius, velocity constraints, or clearance.
Navigation	Final base pose cannot support manipulation	The robot can navigate to the pose but cannot reach or see the manipulation target.	Manipulation reachability, visibility, or IK query at the final pose fails.

Table 23: Common physics-aware augmentation strategies. Local perturbation explores nearby variations around validated anchors, while symmetry-aware augmentation expands candidates using symmetry priors from visual-language annotations.

Skill / object type	Augmented quantity	Augmentation rule	Required recheck
Parallel-jaw grasp	Contact center	Jitter the grasp center within the graspable part mask or affordance region.	Target remains in region; gripper width valid; no collision; grasp remains stable.
Parallel-jaw grasp	Grasp rotation	Perturb in-plane rotation, approach yaw, or pre-grasp offset within a bounded range.	Approach path remains collision-free; contact is preserved; gravity robustness remains valid.
Dexterous grasp	Fingertip contacts	Perturb contact seeds on the same functional region and readjust hand configuration.	Contacts remain on region; hand joint limits, self-collision, and object stability are valid.
Dexterous grasp	Palm pose / joint offsets	Apply small palm-pose or finger-joint perturbations around a validated grasp.	No hand-object penetration; no self-collision; object remains constrained.
Garment fling	Grasp keypoints	Jitter sleeve endpoints, hem points, or bottom corners within keypoint confidence regions.	Bimanual reachability, visibility, cloth clearance, and fling outcome remain valid.
Garment fold	Fold/place target	Randomize sleeve place point, bottom-up fold point, or pants centerline target within a bounded region.	Fold target remains consistent with garment keypoints; no IK singularity or cloth-arm entanglement.
Articulation	Handle contact point	Jitter the contact point on the handle or movable-part mask.	Contact remains on handle; grasp or push remains feasible; no jamming.
Articulation	Pulling direction / waypoint spacing	Perturb pull direction, waypoint interval, and target joint range.	Joint progress remains valid; contact is maintained; gripper or hand does not collide with the frame.
Insertion	Alignment pose	Perturb entry pose, insertion depth, or axis orientation within tolerance.	Axis alignment, clearance, collision-free approach, and insertion depth remain valid.
Hanging	Support contact	Jitter the hook or support-edge contact point and release pose.	Gravity-stable support, clearance, and post-release hanging stability remain valid.
Navigation	Base pose	Perturb (x, y, ψ) around an interaction-ready base pose.	Traversability, clearance, path connectivity, DWB feasibility, and manipulation reachability remain valid.
Bottle / can	Rotationally symmetric grasp	Rotate validated grasp poses around the annotated symmetry axis.	Symmetry transform preserves object geometry; transformed grasp remains collision-free and stable.
Round knob	Rotationally symmetric contact	Rotate grasp or turning candidates around the knob axis.	Contact remains on knob; rotation direction and wrist feasibility remain valid.
Repeated cabinet handles	Translated or mirrored handle action	Copy grasp or articulation candidate to another functionally equivalent handle.	Target handle is equivalent; opening direction and collision context remain valid.
Garment bilateral structure	Mirrored sleeve fold	Mirror left-sleeve fold to right-sleeve fold across the garment centerline.	Mirrored keypoints are detected; fold direction and bimanual reachability remain valid.