MolmoSpaces

A Large-Scale Open Ecosystem for Robot Navigation and Manipulation

Yejin Kim♥[1*] **Wilbert Pumacay**♥[1*] **Omar Rayyan**♥[3*] **Max Argus**♥[1*]

Winson Han♥[1] **Eli VanderBilt**♥[1] **Jordi Salvador**♥[1] **Abhay Deshpande**♥[1] **Rose Hendrix**♥[1] **Snehal Jauhri**♥[5] **Shuo Liu**♥[2]

Nur Muhammad Mahi Shafiullah[4] **Maya Guru**[1] **Arjun Guru**[2] **Ainaz Eftekhar**[2] **Karen Farley**[1] **Donovan Clay**[2] **Jiafei Duan**[1,2] **Piper Wolters**[1] **Alvaro Herrasti**[1] **Ying-Chun Lee**[2] **Georgia Chalvatzaki**[5] **Yuchen Cui**[3]

Ali Farhadi[1,2] **Dieter Fox**[1,2] **Ranjay Krishna**♥[1,2]

[1]Allen Institute for AI, [2]University of Washington, [3]University of California, Los Angeles, [4]University of California, Berkeley, [5]Technische Universität Darmstadt

*denotes equal contribution in no particular order. ♥ marks core contributors. See full author contributions here.

Data: `Assets and Scenes`
Code: `https://github.com/allenai/molmospaces`
Blog: `https://allenai.org/blog/molmospaces`

Abstract

Deploying robots at scale demands robustness to the long tail of everyday situations. The countless variations in scene layout, object geometry, and task specifications that characterize real environments are vast and underrepresented in existing robot benchmarks. Measuring this level of generalization requires infrastructure at a scale and diversity that physical evaluation alone cannot provide. We introduce **MolmoSpaces**, a fully open ecosystem to support large-scale benchmarking of robot policies. MolmoSpaces consists of over 230k diverse indoor environments, ranging from handcrafted household scenes to procedurally generated multiroom houses, populated with 130k richly annotated object assets, including 48k manipulable objects with 42M stable grasps. Crucially, these environments are simulator-agnostic, supporting popular options such as MuJoCo, Isaac, and ManiSkill. The ecosystem supports the full spectrum of embodied tasks: static and mobile manipulation, navigation, and multiroom long-horizon tasks requiring coordinated perception, planning, and interaction across entire indoor environments. We also design **MolmoSpaces-Bench**, a benchmark suite of 8 tasks in which robots interact with our diverse scenes and richly annotated objects. Our experiments show MolmoSpaces-Bench exhibits strong sim-to-real correlation ($R = 0.96$, $\rho = 0.98$), confirm newer and stronger zero-shot policies outperform earlier versions in our benchmarks, and identify key sensitivities to prompt phrasing, initial joint positions, and camera occlusion. Through MolmoSpaces and its open-source assets and tooling, we provide a foundation for scalable data generation, policy training, and benchmark creation for robot learning research.

1 Introduction

Recent advances in robot learning [1–4], have given rise to increasingly general, open-vocabulary policies, capable of zero-shot deployment. As we work towards generalist robots, it becomes important to consider how to evaluate and measure the performance of these policies. State-of-the-art models are already nearing saturated performance on several established tasks, providing little signal to drive further progress [5]. Moreover, most manipulation benchmarks frequently focus on short-horizon skills in a single scene, failing to probe the long-horizon, compositional challenges

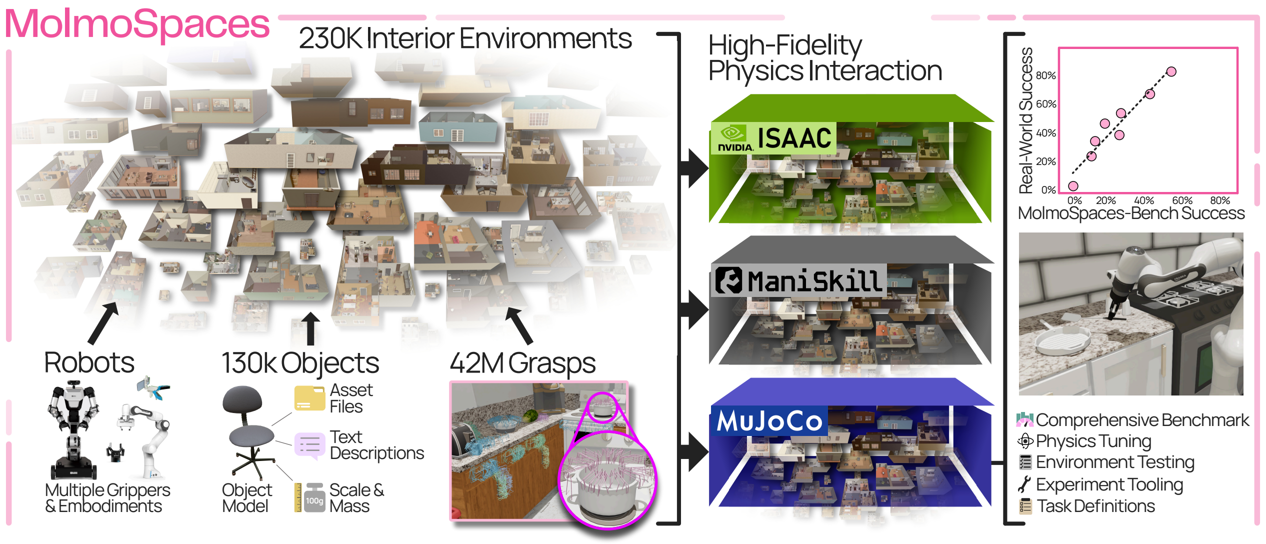


Figure 1 **MolmoSpaces** is an open ecosystem consisting of a large number of simulation environments, 3D articulated objects, and tasks for training and evaluating robot navigation and manipulation at scale. It provides object metadata, grasps, and tooling to generate training data, create benchmarks, and evaluate policies in a manner that correlates with real-world performance.

that arise in realistic environments [6–11].

The real world presents an extraordinarily long tail of situations a robot must handle. Kitchens vary in layout, lighting, and clutter. Objects come in countless shapes, sizes, and materials. Instructions can be phrased in myriad ways. A truly generalist policy must be robust to not just the common cases but to the vast combinatorial space of environments, objects, and tasks that constitute everyday life. Estimating a policy's ability to do so requires evaluating on a far broader distribution of tasks, environments, and objects than ever before.

Simulation offers a compelling path to enable this level of rigor and scale in evaluation. Unlike physical experiments, which are expensive, slow, and difficult to reproduce, simulation enables systematic assessment across thousands of controlled scenarios. Rather than testing on a handful of cherry-picked scenarios, we can characterize policy performance across the full distribution of environments a robot might encounter. However, effective simulation for mobile manipulation must simultaneously support scene-scale diversity, physical realism, articulated interactions, and long-horizon compositional tasks in realistic indoor environments. For simulation experiments to be useful, results in simulation must attain strong correlation with real-world performance [12]. However, existing simulators and benchmarks remain limited. Many provide only dozens of scenes or objects, lack realistic physics or visuals, or support a narrow range of tasks.

We introduce **MolmoSpaces**, an end-to-end large-scale ecosystem for robotics research illustrated in Figure 1. MolmoSpaces unifies diverse scenes, objects, tasks, and tools for training and evaluating generalist robot policies. It contains over 230k diverse indoor environments spanning a wide range of layouts and scene types, which enables evaluation across the long tail of real-world spatial configurations. It also includes more than 130k high-quality rigid and articulated object models with rich semantic and physical metadata, which supports assessment of generalization to novel objects. In addition, MolmoSpaces provides over 42M annotated grasps across 48k interactive rigid and articulated objects, which offers ground-truth supervision for grasp success evaluation. Our assets and scenes dataset can be loaded into multiple simulators (MuJoCo [13], IsaacSim [14], and ManiSkill [15]), all backed by high-fidelity physics.

Using this ecosystem, we construct MolmoSpaces-Bench, a new benchmark suite that evaluates robot policies on 8 base tasks–*navigate-to*, *pick*, *pick-and-place*, *pick-and-place-next-to*, *pick-and-place-color*, *open*, *close*, and *open-door* (Sec. 4.1)– in never-before-seen environments, all zero-shot (i.e. with no fine-tuning on benchmark data). Importantly, the entire MolmoSpaces platform is open-source and extensible. This means that beyond the benchmarks we report, researchers can leverage MolmoSpaces to synthesize their own datasets of scenes, objects, and tasks for training robust robotic policies at scale. We hope that by providing a community-driven ecosystem of this scope, we will accelerate

Feature	Molmo-Spaces	Robo-Casa	AI2-THOR	Habitat 2.0	iGibson 2.0	RL-Bench	Behavior 1K	robo-mimic	Mani-Skill 2	OPTIMUS	LIBERO	Mimic-Gen
Scenes	232k	120	–	1	15	1	50	3	–	4	20	1
Objects	130k	2509	3578	169	1217	28	9318	15	2144	72	–	40
Object Categories	2.8k	153	–	46	–	28	1949	–	–	–	–	–
Tasks	8	100	–	3	6	100	1000	8	20	10	130	12
Realistic Physics	✓	✓	×	×	✓	✓	✓	✓	✓	✓	✓	✓
Realistic Rendering	(✓)	✓	✓	✓	×	×	✓	×	✓	✓	×	×
Muli-Embodiment	✓	✓	✓	×	✓	×	✓	×	×	✓	×	✓
Room-Scale Scenes	✓	✓	✓	✓	✓	×	✓	×	×	×	×	×
Multi-Room Scenes	✓	×	✓	✓	✓	×	✓	×	×	×	×	×
Annotated Object Grasps	✓	×	×	×	×	×	×	×	×	×	×	×
Mobile Manipulation	✓	✓	✓	✓	✓	×	✓	×	✓	×	×	✓
Scripted Datagen	✓	✓	×	×	×	✓	×	✓	✓	✓	×	✓
AI-generated Tasks	✓	✓	×	×	×	×	×	×	×	×	×	×

Table 1 Comparison to popular simulation frameworks used in the robot learning literature. The definition of tasks varies strongly across papers; many take it to be unique verb-object or category combinations.

progress toward truly general-purpose robotic intelligence.

In zero-shot evaluations (no task-specific fine-tuning), our benchmark distinguishes performance among several state-of-the-art policies, including VLA models (π-models [1, 2, 16]) and classical modular baselines, across a wide range of unseen environments and objects. The results reveal steady progress over model generations, but also expose brittleness to distribution shifts. For instance, we find that minor changes in instruction phrasing or initial robot pose can cause significant drops in success for some policies, especially earlier-generation VLAs. This sensitivity highlights the importance of training on more diverse data, which MolmoSpaces can readily supply for future work. Encouragingly, we observe a strong sim-to-real correlation: policies that score higher in our simulation benchmarks also achieve better real-world success rates on equivalent tasks (with Pearson $R^2 \approx 0.92$ for object picking). This validates that high-fidelity simulation can be a reliable proxy for real-world performance. Moreover, by systematically perturbing scene parameters and sensor inputs in simulation, we pinpoint specific failure modes (e.g. dependence on certain camera viewpoints and lighting conditions) that are costly to uncover with physical trials. These analyses demonstrate how an open ecosystem like MolmoSpaces not only measures overall progress but also yields insights to drive algorithm improvement.

By dramatically expanding the scale of available simulated environments and making them openly accessible, MolmoSpaces enables researchers to measure generalization more rigorously than ever before and can support future work in generating diverse training data for tackling the next generation of robotics problems.

2 Related work

Robot simulation frameworks provide scalable, safe, and repeatable platforms for rapid prototyping, policy learning, and evaluation. Modern simulators such as MuJoCo, Isaac, and ManiSkill offer high-fidelity physics simulations that support contact- and force-based manipulation [13–15, 17]. Building on these engines, the research community has developed a range of simulation frameworks. Projects like AI2-THOR [18], Habitat, and Habitat-Lab [19, 20] emphasize photorealistic visual navigation, but provide only limited manipulation support, often relying on “magic grasps” that bypass realistic contact dynamics [18, 21]. RoboCasa and RoboCasa365 [22, 23] build on MuJoCo to provide single-room environments, synthetic datasets, and task definitions for multi-task manipulation and navigation, but remain limited in scene and asset diversity.

Large-scale datasets in robotics are increasingly important, as demonstrated by efforts such as Open X-Embodiment [24]. In parallel, the community has pursued data scaling through simulation. Objaverse [25] provides internet-scale 3D assets compatible with simulators, while ProcTHOR expands AI2-THOR with tens of thousands of procedurally generated multi-room houses [21]. Holodeck [26] introduces LLM-guided scene generation beyond household environments, and InternScenes [27] combines real scans, procedural layouts, and designer-created environments to provide diverse indoor scenes at scale. Despite addressing scene and asset scale, these datasets offer limited support for physics-based manipulation and are primarily evaluated on navigation tasks. Conversely, GraspGen [28] provides large-scale grasp annotations for Objaverse assets, but these remain at the asset level and require substantial effort to integrate into interactive scenes.

By contrast, **MolmoSpaces** addresses both scale and task diversity through a ready-to-use ecosystem that unifies 230K scenes from AI2-THOR [18], ProcTHOR [21], and Holodeck [26], and makes them compatible across MuJoCo, IsaacSim, and ManiSkill. The ecosystem further incorporates 130K object assets from Objaverse [25], with over 48K objects annotated with grasp data and validated to be pickable and articulable under realistic physics. Together, these components enable scalable, diverse, and physically grounded evaluation of navigation, manipulation, and mobile manipulation policies. Comparisons between MolmoSpaces and prior work are summarized in Table 1.

Benchmarks are central to progress in robotics, providing standardized tasks and evaluation protocols for fair comparison, reproducibility, and diagnosis of failure modes. However, real-world benchmarking remains difficult to scale due to hardware heterogeneity, differences in sensing and control stacks, and the time and labor required for evaluation. Recent efforts such as RobotArena [29] partially address these challenges through distributed, crowd-sourced evaluation, while others like AutoEval [30] leverage success classifiers and reset policies to facilitate near-autonomous real-world evaluations of specific tasks. However, real-world evaluation alone remains limited in the scale and diversity needed to robustly assess generalist policies.

Simulation-based benchmarks offer a scalable and reproducible alternative, enabling controlled variation and systematic stress testing that are impractical in the real world. A wide range of benchmarks have emerged for manipulation [6, 8] and navigation [18, 31, 32], becoming standard testbeds for evaluation. RoboVerse [33] unifies several of these benchmarks under a shared framework. As policies converge on vision–language–action (VLA) models, recent benchmarks such as LIBERO [6], CALVIN [7], LIBERO-Plus [9], LIBERO-Pro [10], VLABench [34], and RobotArena-Infinity [11] expand evaluation to include language grounding and generalization. Additional works like the COLOSSEUM [35] and VLATest [36] evaluate the robustness of generalist policies to changes in lighting, distractor objects, camera poses, and more. Other efforts, including BEHAVIOR-1K [37], ManiSkill-HAB [38], and EmbodiedBench [39], extend benchmarks to mobile manipulation, but remain biased toward household environments and limited in scene scale.

Sim-to-real benchmarks focus on providing simulation evaluations that match real-world results. SIMPLER [40] studies distributional shifts by pairing simulated evaluations with real-robot rollouts, while PolaRiS [12] constructs digital twins from real-world videos and demonstrates strong sim-to-real correlation across realistic scenes, albeit limited to tabletop manipulation and partial environment reconstruction. Other works analyze large behavior models across simulated and real environments, but rely on proprietary evaluation pipelines and closed datasets. Moreover, many sim-to-real benchmarks require training on simulation data, limiting their ability to assess true zero-shot generalization [12, 37].

In contrast, our benchmark evaluates zero-shot generalist policies across navigation, manipulation, and mobile manipulation tasks using a validation set of over 20K diverse indoor scenes—spanning both household and non-household environments—and more than 22K interactable rigid and articulated objects. It supports large-scale distributional analysis under controlled perturbations to scenes, objects, sensors, and language prompts, enabling rigorous assessment of generalization and failure modes.

3 MolmoSpaces

MolmoSpaces contains:

1. **MolmoSpaces-Scenes:** Over 230k diverse indoor environments spanning a wide range of layouts and scene types, enabling evaluation across the long tail of real-world spatial configurations.
2. **MolmoSpaces-Objects:** More than 130k high-quality rigid and articulated object models with rich semantic and physical metadata, supporting assessment of generalization to novel objects.
3. **MolmoSpaces-Grasp:** Over 42M annotated grasps across 48k interactive rigid and articulated objects, providing ground-truth supervision for grasp success evaluation.
4. **MolmoSpaces-Bench:** A benchmark suite comprising object-centric tasks across 8 task types, zero-shot evaluations with no fine-tuning, and sim-to-real correlation analysis demonstrating that lessons learned in simulation transfer to real-world performance.
5. **Simulation Infrastructure:** Scalable tooling for task composition, benchmark creation, and reproducible evaluation.

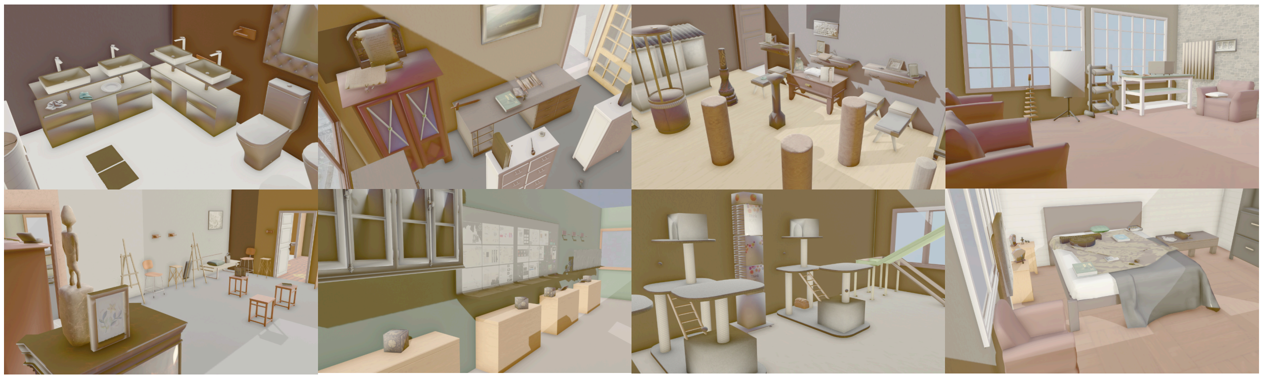


Figure 2 Examples of diverse scene environments from MolmoSpaces-Scenes-MultiType with the Filament rendering engine. Our ecosystem contains scenes from art studies, cat cafes, lounges, museums, and many other scenes, all pre-populated with objects to be manipulated.

Dataset Name	Scene Count	Object Set	Obj / Scene	Creation Method
MSTwin	1	-	8	Manual
MSCrafted	120	THOR	∼30	Hand-crafted
MSProc	12k	THOR	∼72	Heuristic
MSProcObja	110k	THOR+	∼60	Heuristic
MSMultiType	110k	THOR+	∼97	LLM Proc.

Table 2 MolmoSpaces-Scenes contains five scene datasets with varying scene types, scales, objects, and creation methods.

Dataset Name	Stab.	Lift	Inter.	Artic.
MSCrafted	100.0	97.5	98.3	99.1
MSProc	99.4	99.7	99.9	97.4
MSProcObja	98.4	98.7	99.1	93.7
MSMultiType	94.9	99.6	93.7	65.2

Table 3 Scene datasets are all quality tested in the MuJoCo simulator with high pass rates.

Dataset Name	MSCraft	MSProc	MSObja	MSMulti
Stability test (%)	92.5	93.9	95.9	95.2

Table 4 Isaac Sim simulator pass rates.

3.1 MolmoSpaces-Scenes

We provide five scene datasets, summarized in Table 2. The scenes were originally sourced from multiple datasets in AI2-THOR: MolmoSpaces-Scenes-Crafted, MolmoSpaces-Scenes-MultiType, MolmoSpaces-Scenes-Procedural, and MolmoSpaces-Scenes-Procedural-Objа. We process and tune these scenes to be physically realistic in MuJoCo, ManiSkill, and IsaacSim. MolmoSpaces-Scenes-Crafted (MSCrafted) contains 120 hand-crafted single-room scenes evenly distributed among kitchens, bedrooms, living rooms, and bathrooms, split into 48/48/24 train/validation/test scenes, with object placement carefully curated to ensure physical stability. MolmoSpaces-Scenes-Procedural (MSProc) provides 12k procedurally generated residential scenes with 10k/1k/1k train/validation/test splits, where each scene represents a house with between one and ten rooms with layouts designed for realistic household navigation and manipulation. MolmoSpaces-Scenes-Procedural-Obja (MSProcObja) extends this with 110k train/validation scenes containing both THOR and Objaverse objects. MolmoSpaces-Scenes-MultiType (MSMultiType) provides 110k diverse scene types generated via LLM-based procedural generation, also with THOR and Objaverse objects. Finally, MolmoSpaces-Scenes-DigitalTwin (MSTwin) is a high-fidelity manual reconstruction of our real-world kitchen. All datasets contain both rigid and articulated objects. Some examples of these scenes are shown in Figure 2.

To generate MSMultiType, we extend the diverse scene generation pipline presented in Holodeck [26]. We select and extend indoor scene types in the SUN database [41], based on the suitability to the available THOR and Objaverse object taxonomy, and reorganize them into a hierarchy with between five and fifteen concrete scene types for each of ten generic types. In total, 546 room types (including many scene-specific) and 101 scene types are available for scene sampling (Fig. 17 left, in the appendix). Each scene specification contains a generic or concrete scene type and between one and ten rooms of diverse types. We also sample from a subset of 52k persona descriptions from [42] – chosen by their suitability to produce visual and stylistic differences in objects, materials, or layout constraint selection – and accentuate some particular style in 90% of the scene specifications, which are finally converted to text prompts for LLM scene generation. We add a ‘grid’ constraint to the DFS-based object placement optimizer in Holodeck to simplify the

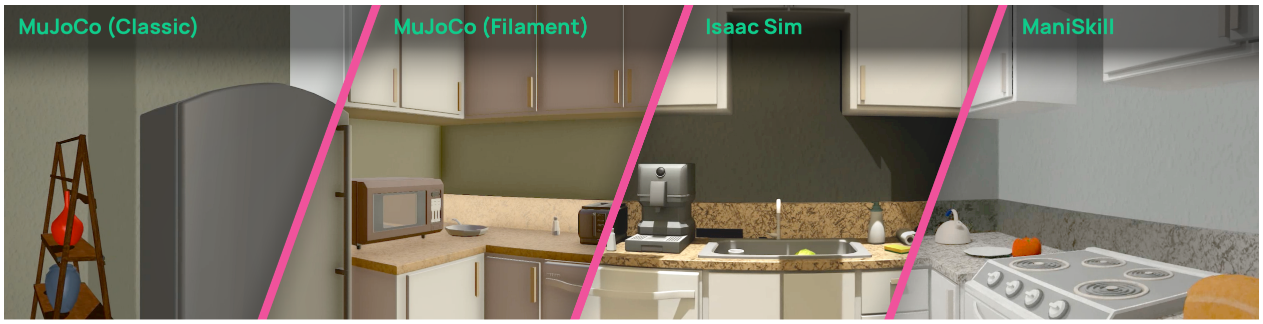


Figure 3 An example scene rendered across different simulators: MuJoCo, Issac Sim, and ManiSkill. When using MuJoCo, the scenes can be rendered using either the OpenGL renderer (Classic) or with Filament (Filament).

uniform placement of objects in available free space commonly occurring in non-residential scenes.

In order to enable the wide-spread use of our simulation assets, we also release them converted to the USD format, which can be natively loaded in IsaacSim. Additionally, we provide both an asset and scene loader for ManiSkill. Figure 3 illustrates the same scene rendered across aforementioned simulators and their respective rendering engines. Finally, we generate occupancy maps for all scenes to identify collision-free starting poses for robots, ensuring safe initialization for both manipulation and navigation experiments.

Scene quality testing: For an environment to be suitable for both navigation and manipulation tasks, objects must remain physically stable, respond appropriately to applied forces, and be accessible for interaction. Specifically, objects should not drift or randomly move, rigid objects must be pick-upable, articulated objects must allow motion across their defined ranges, and masses, friction, and other physical parameters should yield realistic dynamics under standard forces. Ensuring these properties is particularly important when integrating assets not originally designed for physics-based simulation. Additionally, objects that are originally designed for such simulations often have innately incompatible parameters by default which also warrant additional tuning for integration.

To systematically enforce these conditions, we implemented four tests addressing different aspects of physical validity: scene stability, object intersections, rigid object liftability, and articulation of movable objects. First, each scene was settled by simulating approximately 20 seconds and then saved with the updated, settled poses. Stability tests simulate the environment for multiple steps after settling and remove any objects that continue to jitter. Intersection tests detect colliding objects: if a collision occurs between a fixed and a free object, the free object is removed; if the collision is between two free objects, the smaller one is removed. Lift tests apply upward forces to all free objects and measure their displacement along the z-axis; objects that cannot be lifted by at least 5 cm and are detected inside another object's site are removed. Articulation tests apply forces at joints to open or close articulated objects; if an object cannot be actuated through at least 70% of its joint range, free objects located within its articulation site or blocking its motion are removed

Across MuJoCo scenes (Table 3), over 95% of environments pass these stability and manipulation checks. The lowest success rate is observed for articulation tests on the MSMultiType scene dataset (63%). We note that these scenes were designed to maximize navigability by biasing large object placement to prevent blocking door-to-door navigation while ensuring many small objects can be accessibly placed for manipulation tasks. Manual inspection confirms that most failures arise from scene layout rather than limitations of the simulation engine. We additionally evaluated a subset of scenes in Isaac Sim, observing consistently high pass rates as shown in Table 4, which demonstrates the robustness and cross-platform validity of our validation pipeline.

3.2 MolmoSpaces-Objects

We provide two object model datasets consisting of 1.6k THOR and 129k Objaverse objects, with samples shown in Figure 4. These assets populate the generated scenes described in Section 3.1. Table 5 reports the average number of objects per scene for each dataset. To ensure physical realism, rigid objects were validated by estimating mass and density against LLM-estimated values, and articulable objects were tuned by manipulating them via teleoperation of a simulated Franka FR3 robot. Collider meshes were generated using COACD [43], with primitive colliders *human annotated* for all THOR assets. For stability, receptacle objects primarily use primitive colliders, while manipulable objects use convex decomposition except for very small or thin items, where primitives are preferred. Meshes in

Dataset	Total	Pickupable	Non-Pick	Articulated	THOR	Objaverse
MSCrafted	$\sim$30	$\sim$14	$\sim$7	$\sim$9	$\sim$23	–
MSProc	$\sim$72	$\sim$34	$\sim$31	$\sim$7	$\sim$41	–
MSProcObja	$\sim$105	$\sim$47	$\sim$50	$\sim$7	$\sim$72	$\sim$32
MSMultiType	$\sim$150	$\sim$61	$\sim$77	$\sim$12	$\sim$73	$\sim$77

Table 5 Average number of objects per scene on each dataset

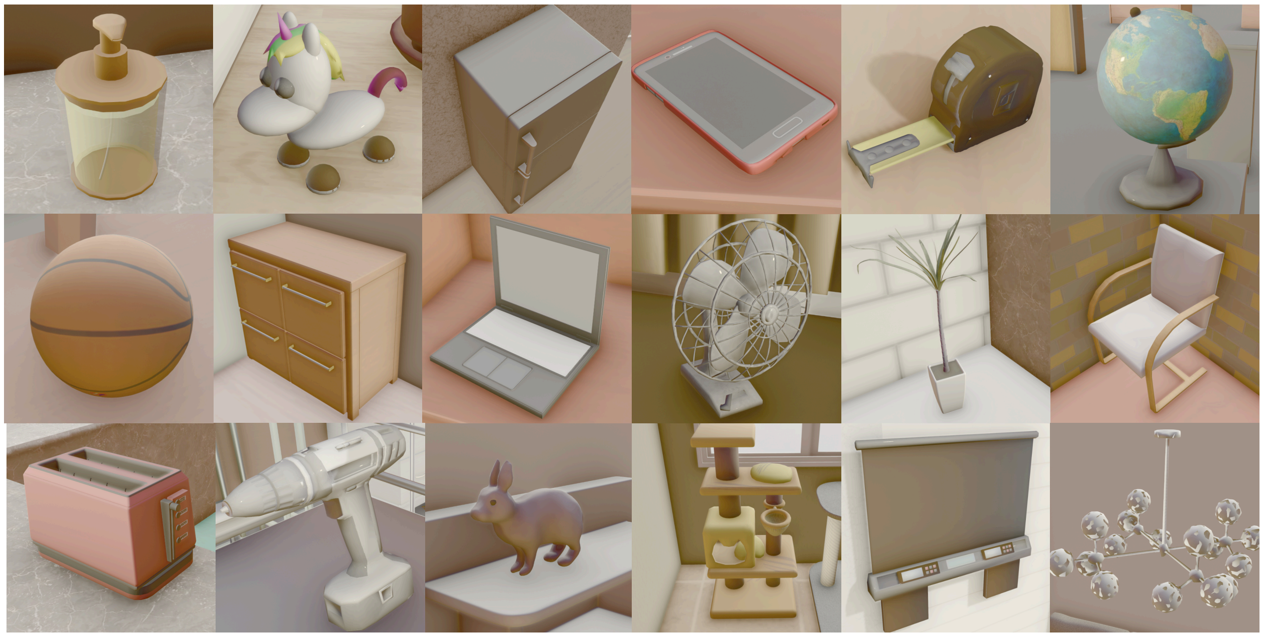


Figure 4 A random sampling of object types in our ecosystem, with different sizes, shapes, and articulations. These examples are rendered with Filament.

Objaverse models were further processed and decimated for simulation efficiency [44].

THOR objects span 134 categories, with 22 articulable categories (e.g., doors, refrigerators) annotated with joint types, axes, positions, and ranges. Objaverse objects, spanning almost 2.8k WordNet [45, 46] synsets, are curated from 625k models annotated with descriptions, mass estimates, canonical poses, pickable and receptacle properties, synsets, and scale estimates generated by GPT-4o [47, 48]. A complementary GPT-4.1 [49] annotation identified object counts, extraneous or missing geometry, texture quality, and receptacle presence in models with parseable annotation. Filtering provided 129k single-object models that met scale consistency, sufficient texture quality, cross-renderer fidelity, compact file size, collider quality, and synset coverage as objects to be placed in MSMultiType scenes. Additional filtering was done for MSProcObja to keep only objects with synsets heuristically mapped to placement-compatible THOR categories, resulting in 92k objects across 2k synsets. Further details in filtering and curating objects from Objaverse are described in Appendix A.

Object models are accompanied by extensive physical and semantic metadata, convex colliders, and canonical coordinate definitions, besides grasps, which are obtained as described below. To support easy integration into robotics simulation workflows, all object models are provided in formats compatible with MuJoCo, IsaacSim, and ManiSkill.

3.3 MolmoSpaces-Grasp

We introduce a comprehensive grasp dataset, MolmoSpaces-Grasp, consisting of over 42 million grasps that cover objects in MolmoSpaces scenes. Our dataset provides 6-DoF grasp poses for two types of manipulable objects: rigid objects, which can be picked and moved as single bodies, and articulable objects, whose constituent parts can move relative to one another via a revolute or prismatic joint. Our grasp generation process builds on prior work such as

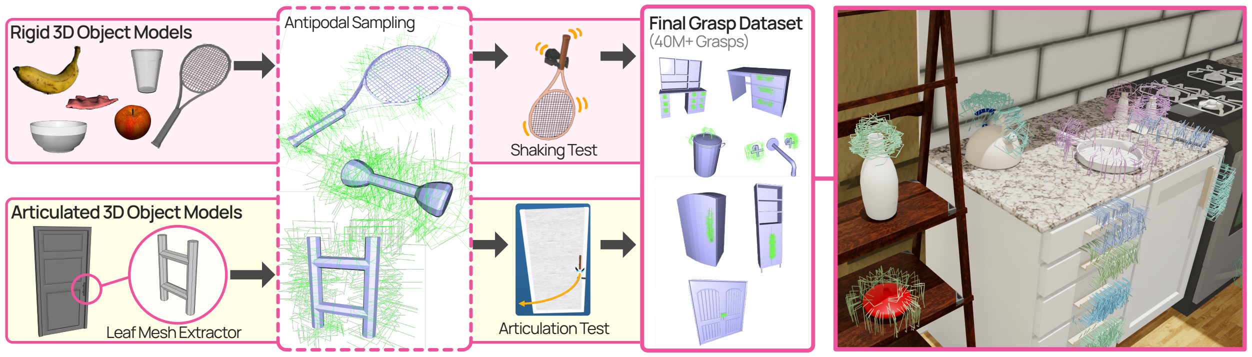


Figure 5 Our grasp generation pipeline consists of separate streams for rigid and articulated assets. We generate 42M+ verified grasps that can be utilized to create scripted interaction policies. Grasps can be used in different simulation environments, with an Issac example shown on the right.

GraspGen [28], 6-DOF GraspNet [50] and ACRONYM [51], with extensions to support articulable objects through a new functionality-based evaluation step. We apply this pipeline to 48,111 objects drawn from a curated subset of Objaverse and custom-designed THOR assets, using separate pipelines for rigid and articulated objects to reflect their distinct manipulation requirements. Table 8 compares our dataset with recent large-scale grasp datasets.

Grasp generation: Our pipeline (Figure 5) begins with 3D object models, from which we extract mesh colliders for grasp sampling. Antipodal contact pairs are sampled based on the geometry of the Robotiq 2F-85 gripper and the object. For rigid objects, sampling is performed across the full mesh surface, whereas for articulated objects, it is restricted to leaf components corresponding to handles or other functional interaction points. Grasps that result in collisions with non-leaf geometry are immediately discarded.

From the initial samples, we select up to 1,000 *diverse* and *robust* grasps per object. To ensure diversity, sampled grasps are clustered in the full 6-DoF pose space and tested uniformly across clusters. Grasp robustness is evaluated differently depending on the object type. For rigid objects, we apply controlled linear and rotational perturbations to the gripper upon grasping and discard any grasp that fails to maintain contact, leading to object slippage. For articulated objects, we define a grasp robustness by its actuation feasibility, meaning it can actuate the relevant joint through at least 70% of its valid range in both directions, while maintaining stable contact.

To evaluate the practical utility of annotated grasps in MolmoSpaces scenes, we perform in-situ tests using a floating Robotiq 2F-85 gripper. Candidate grasps that collide with scene geometry are discarded, with collision checks performed at the pre-grasp pose (4 cm offset along the gripper's negative local z-axis), grasp pose, and along the execution trajectory. The gripper then executes the pre-grasp and grasp motions, followed by lifting the object or articulating its moving part along the object's joint path. Success rates (Fig. 24 in Appendix) are reported only for grasps that pass all collision checks and complete the intended motion.

Initial evaluations showed that objects placed on surfaces often had few viable, non-colliding grasps. This is because grasp generation is performed on isolated objects without surrounding geometry and ignores gravity, resulting in high failure rates due to slippage during in-situ testing. To mitigate this, we updated the pipeline to generate more robust and stable grasps by biasing contacts toward the center of the fingertips as illustrated in Figure 6 . For small or thin objects, such as forks and pens, fingertip-edge contacts are preferred.

Remaining failures are largely due to scene- and object-level constraints: for lifting, common issues include objects too large for the lift pose, confined spaces limiting vertical clearance, collisions along the execution trajectory, and object slippage through the grasp; for articulated objects, failures occur when the object lifts instead of actuating, collisions exists along the articulation path, obstacles block articulation, or the gripper misaligns with the handle. These results underscore the importance of in-situ evaluation for producing grasps that are functional and practically useful given the way objects contextually exist within the scene.

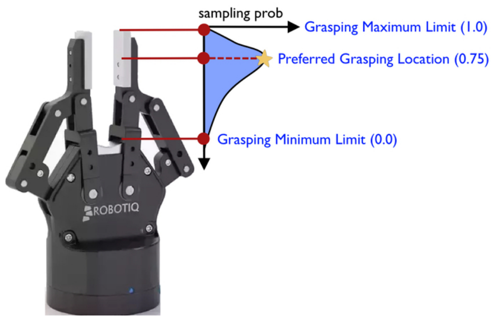

sampling prob
Grasping Maximum Limit (1.0)
Preferred Grasping Location (0.75)
Grasping Minimum Limit (0.0)
ROBOTIQ

Figure 6 RobotIQ 2F-85 gripper and preferred grasping locations for sampling grasps

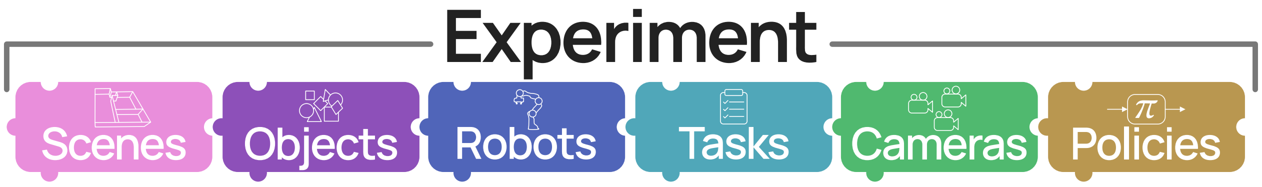


Figure 7 Code Structure with modular experiment composition.

3.4 Robots

We provide robots with varying amounts of mobility and complexity, covering the spectrum of widely-used manipulation platforms. We categorize these platforms as static or mobile, and single-arm or bimanual. To handle all of these cases, we provide a Franka FR3 arm with three gripper configurations (Franka Hand, RobotIQ 2F-85, and CAP) as a static manipulator, Rainbow RB-Y1 as a bimanual and holonomic mobile manipulator, and floating CAP [52] and RobotIQ grippers, unconnected to arms, so without kinematic limits. The Franka FR3 with the RobotIQ 2F-85 gripper is specifically set up as a DROID [53] system, with corresponding cameras with the correct intrinsics and extrinsics. The Franka FR3, RobotIQ Gripper, and Rainbow RB-Y1 robot models were sourced from [54].

Robot Control: For manipulators, our framework provides for both absolute and relative joint position commands, which are tracked internally with a gravity-compensated joint-space stiffness controller. Mobile platforms, such as the holonomic RB-Y1 base or 6DoF floating CAP, can be controlled via absolute or relative poses.

Kinematics Computation: We provide built-in forward and inverse kinematics solvers for each robot, and the modular nature of our framework makes it easy to further extend for new robots. Our parallelized inverse-kinematics solver is written in JAX, and leverages Levenberg-Marquadt optimization with null-space control for posture regularization. This solver can natively be GPU-accelerated, but even on a CPU can solve batches as large as 256 samples with high precision in $\sim$200ms. For the RB-Y1 robot, which is configured for use with the cuRobo[55] motion generator, we provide forward and inverse kinematics as a wrapper around cuRobo's functionality.

3.5 Modular experiment composition

MolmoSpaces supports modular experiment composition by flexibly combining scenes, tasks, robots, and camera configurations, as illustrated in the Figure 7. We provide camera setups for commonly used RealSense, ZED, and GoPro cameras. Beyond vision inputs—including calibrated multi-view RGB and optional depth cameras with object image points—the framework exposes rich proprioceptive signals (e.g., joint states, end-effector and base poses), task-state information (object and articulation states), as well as task annotations, planner signals, and action histories. Together, these sensors capture the full interaction context.

3.6 Data collection

Two different modes of data collection are possible. Manual data collection is possible using TeleDex [56] iPhone app, which uses iOS ARKit to stream the phone's pose. In addition to this, leveraging the pre-computed grasps, it would be possible for scripted policies to control the robot to solve the defined tasks.

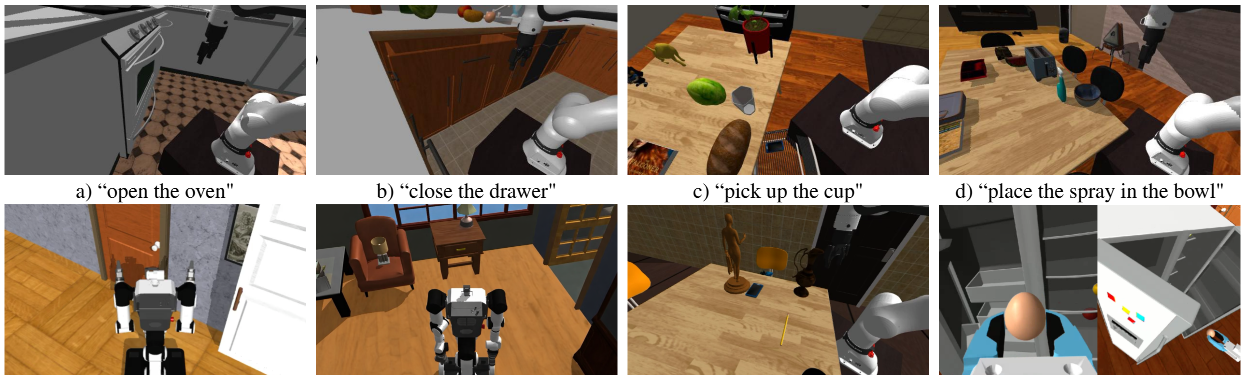


Figure 8 Example images of our range of tasks, spanning from manipulation of articulated and non-articulated assets to navigation and long-horizon tasks, shown together with their associated text instructions. These examples are from the MuJoCo simulator.

4 Benchmark

To enable rigorous and reproducible evaluation of robot policies, we introduce MolmoSpaces-Bench, which spans eight base tasks across navigation, manipulation, and mobile manipulation. These benchmarks are designed with explicit diversity requirements across scenes, object categories, and robot configurations, with each trial verified for solvability.

4.1 Tasks

Leveraging the diverse scenes, assets, and robots provided by MolmoSpaces, we introduce a suite of tasks and corresponding benchmarks designed for comprehensive policy evaluation. We define eight base tasks: *navigate-to*, *pick*, *pick-and-place*, *pick-and-place-next-to*, *pick-and-place-color*, *open*, *close*, and *open-door*. For each of these tasks, we also provide a well-defined success condition and a dense reward function.

1. **Navigate-to:** A policy must search for and navigate to a specified target object. The robot is initialized between 4 and 20 meters away from the target object, possibly in an entirely different room. The success conditions require the object to be visible from the robot’s head camera and closer than 1.5 meters away. We sample the same object candidate set as [57].
2. **Pick:** Grasp and lift a specified object from its initial location by at least 1cm.
3. **Pick-and-place:** Move a target object to be into or onto a target receptacle. To be counted as successful, at least 50% of the object’s weight must be vertically supported by the receptacle. Additionally, the target receptacle cannot have been displaced by more than 10 cm or 45 ° from its initial pose.
4. **Pick-and-place-color:** Similar to *pick-and-place*, with the same initialization and success conditions. However, the task is complicated by multiple duplicates of the target receptacle differentiated only by color. This color is included in the task prompt, requiring policies to attend to and follow specific task instructions.
5. **Pick-and-place-next-to:** Move a target object to be next to a target receptacle. The target receptacle is initialized 30-50 cm away from the target object, and to be successful, the policy must move the object to be closer than 5 cm surface-to-surface from the target receptacle. The object must additionally be on the same support surface (e.g., table) as the receptacle, which cannot have been displaced by more than 5 cm or 45° from its initial pose.
6. **Open:** Open an articulated household fixture such as a drawer, cabinet, microwave, or fridge. Starting fully closed, the robot must open the fixture by at least 15%.
7. **Close:** Close an articulated household fixture, which is initialized halfway open. To succeed, the policy must close the fixture to at most 15% open or 85% closed.
8. **Open-door:** Open a hinged door by manipulating its handle or surface. The door starts fully closed, and the policy is tasked with opening it by at least 67%.

Navigation The *navigate-to* task is evaluated with the Rainbow RB-Y1, which is instructed to locate and navigate to a given object. Following [57], the robot is initialized 4-20m away from the target object, and success is defined as being closer than 1.5m to the target object with it clearly visible in the navigation camera. For this task, policies must explicitly signal task completion, and incorrectly doing so is counted as a failure.

Rigid-body manipulation Non-articulated manipulation includes the *pick*, *pick-and-place*, *pick-and-place-color*, and *pick-and-place-next-to* tasks, which are evaluated with a Franka FR3 robot in the DROID configuration. For *pick*, the robot must grasp and lift an object by at least 1cm. For *pick-and-place*, the robot must move an object into or on a target receptacle, while *pick-and-place-next-to* requires the robot to place the object next to the receptacle. The *pick-and-place-color* task is a variant of *pick-and-place* with multiple similar but differently-colored receptacles, requiring policies to attend and adhere to specific task instructions.

Articulated manipulation Our three articulated manipulation tasks cover both static and mobile manipulation.The *open* and *close* tasks are static manipulation tasks evaluated with the FR3 in the DROID setup, where the robot must open or close a variety of articulated household fixtures, including cabinets, drawers, refrigerators, and microwaves, by at least 15%.The *open-door* task is a mobile articulated manipulation task, where an RB-Y1 robot must push or pull a door to be at least 67% open, requiring coordinated mobility and manipulation across many degrees of freedom.

Specifically, we evaluate following models:

1. **PI models (manipulation).** We evaluate generalist models from the PI family, namely π_0, π_0-`FAST`, and $\pi_{0.5}$. As these models are trained primarily on real-world data, this setting constitutes a real-to-simulation evaluation. We follow the DROID hardware setup for this set of evaluations.
2. **CAP models (manipulation).** We evaluate CAP models that are task-specific and available for the following tasks: pick, open, and close. The models are conditioned on 3D "contact" points, which are provided by using Gemini-Robotics-ER-1.5 [58] at the initial step of the episode. We use the custom gripper design that CAP policies are trained with to replace the Robotiq Gripper from the DROID setup.
3. **RING model (navigation).** `RING` is an embodiment-agnostic indoor navigation policy trained entirely in simulation using large-scale randomization over robot body geometry and sensor configurations.
4. **DualVLN (navigation).** DualVLN is a dual-system VLN foundation model that separates high-level reasoning from low-level control: a VLM-based global planner predicts mid-term waypoint goals, with a lightweight diffusion-based policy that executes smooth, real-time trajectories conditioned on these goals.

4.2 Benchmark creation

For every task, we provide one or more benchmarks designed for comprehensive policy evaluation. These benchmarks draw from multiple scene datasets, described in Sec. 3.1, and provide varying levels of difficulty and complexity. Concretely, each benchmark is defined by an initial scene, robot, and camera configuration, as well as a descriptive task instruction. To ensure benchmark quality, we perform balancing to maximize diversity of object categories and instances, as well as scenes. Additionally, each of our provided benchmarks are guaranteed to be solvable with the provided robot and initial conditions, ensuring task feasibility.

We generate a full set of benchmarks with all combinations of environments and tasks, listed in Table 6. For easier comparisons, we also include preferred evaluation configurations. For the open and close tasks, we choose the MSCrafted scenes, as these contain handcrafted kitchens with many cabinets and drawers. For most manipulation tasks, we choose the MSProcObj variants, as the presence of objaverse assets gives object diversity. For the door-opening task, we use the MSProc environment, as object diversity is irrelevant. All tasks have 2k samples. We present select benchmark results in Sec. 5.1.

4.3 Extensions

Controlled variants of benchmarks enable easier testing of hypotheses. For example, to answer the question "Does a policy perform as well on open vocabulary navigation as on closed vocabulary navigation?", one can evaluate on MSProc, which uses a closed set of object categories, versus MSMultiType, which supports open vocabulary. In addition to this, MolmoSpaces also allows for the easy creation of controlled adversarial benchmarks to test specific functionality,

Task	MSCrafted	MSProc	MSProcObj	MSMultiType
Open (Franka)	✔+e	✓	✓	✓
Close (Franka)	✔+e	✓	✓	✓
Pick (Franka)	✓	✓+e	✔	✓
Pick & Place (Franka)	✓	✓+e	✔	✓
Pick & Place Next To (Franka)	✓	✓	✔	✓
Door Open (RBY1)	✓	✔	✓	✓
Pick (RBY1)	✓	✓	✔	✓
Pick & Place (RBY1)	✓	✓	✔	✓
Navigation (RBY1)			✓	✔

Table 6 Availability of manipulation and navigation benchmarks across environments. ✓ indicates the benchmark is available, +e indicates there is an easy variant of the benchmark, ✔indicates the preferred evaluation environemnt.

such as the ability of a policy to handle varying initial robot configurations (shown in Fig. 12) or the ability to do manipulation in progressively more cluttered environments.

Additionally, while our provided benchmarks cover a variety of short-horizon tasks, long-horizon evaluation is also a critically important aspect of benchmarking. To that end, we provide an LLM-based task generation system that can generate **long-horizon tasks** with more abstract task descriptions. In this process, shown in Fig. 9, we take an existing scene and prompt an LLM to generate feasible long-horizon tasks, defined by a high-level task description and a sequence of base tasks as defined in Sec. 4.1. Unlike the base tasks, the tasks obtained by this sampling procedure can result in longer horizons, e.g., needing to open a fridge before taking objects out and placing them elsewhere.

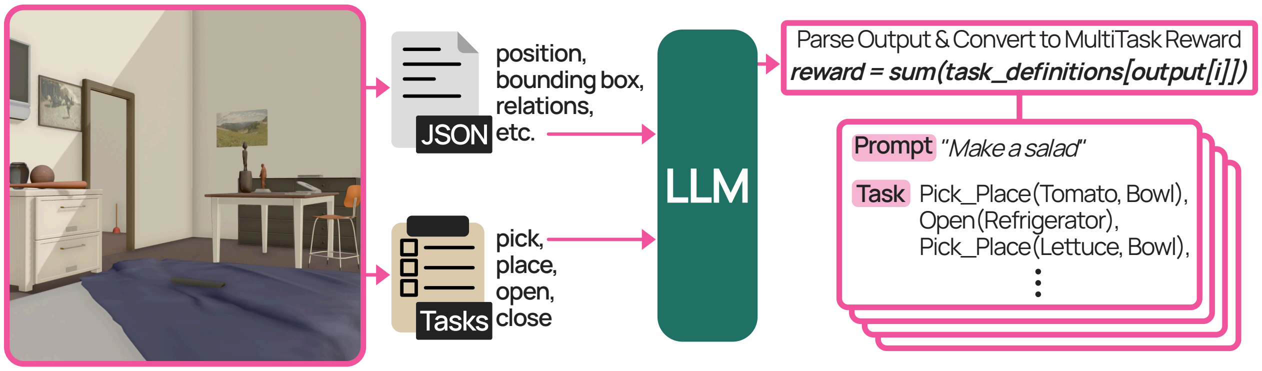

JSON
position, bounding box, relations, etc.
Tasks
pick, place, open, close
LLM
Parse Output & Convert to MultiTask Reward
reward = sum(task_definitions[output[i]])
Prompt "Make a salad"
Task Pick_Place(Tomato, Bowl), Open(Refrigerator), Pick_Place(Lettuce, Bowl),

Figure 9 The LLM-based long-horizon task generation system makes use of text-form scene descriptions to generate new task descriptions and success condition checks based on predefined atomic checks.

5 Experiments

5.1 Evaluations

We use MolmoSpaces-Bench to evaluate open-source manipulation and navigation policies across the diverse scenes of MolmoSpaces zero-shot, without task-specific fine-tuning as done in many prior evaluations. We believe this setting is more practical, as it evaluates models as they are released and intended to be used, rather than their ability to efficiently fit additional task-specific data. We report the results under the standard settings in in Fig. 10.

5.1.1 Manipulation tasks

We evaluate vision–language–action (VLA) models from the π family [1, 2, 16] (π_0, π_0-FAST, and $\pi_{0.5}$, specifically the joint position variants that have been fine-tuned on the DROID dataset [12]) as well as utility contact-action models from CAP [52]. These models make use of different setups as described in Sec. 3.4, and therefore operate under different sensing and embodiment constraints. In particular, the DROID setup provides the π models with two camera views,

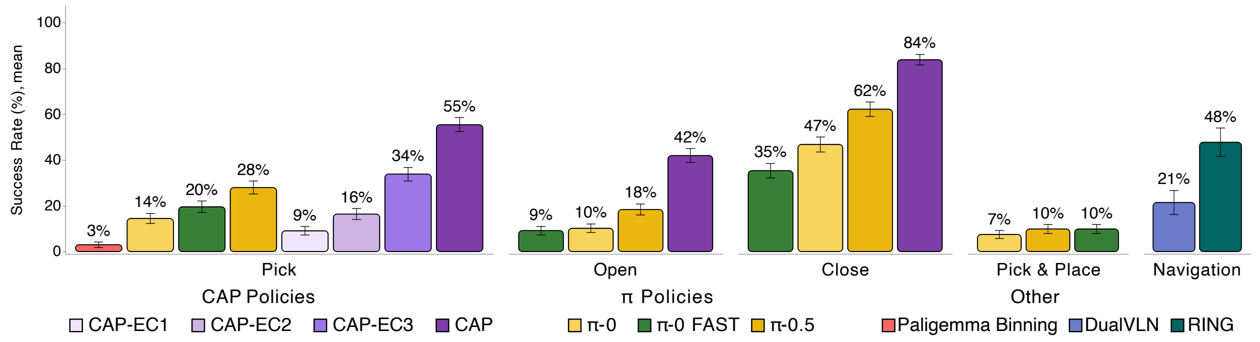


Figure 10 Zero-shot success rates of different baseline policies across five MolmoSpaces benchmark tasks. Showing expected performance improvement in improved policies. Error bars show 95% Bayesian credible intervals.

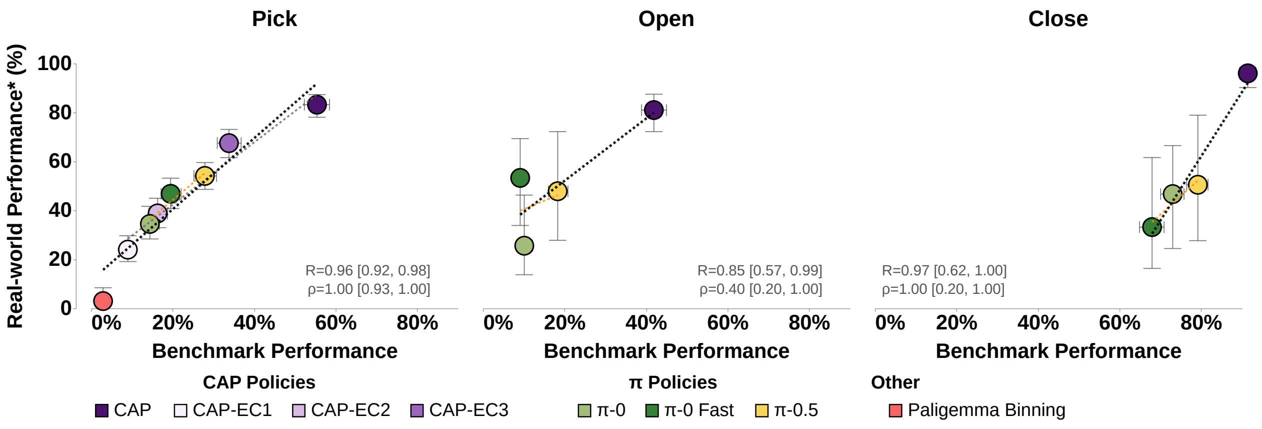


Figure 11 Sim-to-real correlation results for *pick*, *open*, and *close* task. Coefficient of determination (R) and the Spearman rank correlation coefficient (ρ) are shown.

whereas the floating CAP setup uses a single wrist-mounted camera and has no kinematic constraints. During benchmark construction, we ensure that all manipulation tasks are feasible under the DROID setup.

5.1.2 Navigation Tasks

We select two state-of-the-art prior methods for visual object-goal navigation. RING [59] is an embodiment-agnostic, transformer-based navigation policy trained entirely in simulation that demonstrates robust generalization across a wide range of real-world robot platforms. It is trained on semantic navigation tasks where instructions consist of a simple verb (e.g., “go to", “locate", “find", “search for", “navigate to") paired with an object category. DualVLN [60] is a dual-system vision–language navigation (VLN) foundation model that integrates high-level reasoning with low-level action execution. Unlike RING, DualVLN is trained on detailed instructions that specify intermediate steps the robot must follow, rather than simple semantic goals (see Fig. 4 of [60] for examples). Both policies are evaluated on the *navigate-to* benchmark, which comprises 2,000 trajectories across 679 houses and goal objects drawn from 568 synset categories. This difference in task formulation explains the performance gap between policies (see Fig. 10), as our benchmark's semantic navigation format poses a distributional mismatch for DualVLN while aligning with RING's training. We adopt semantic navigation because it is more directly compatible with downstream mobile manipulation tasks.

5.2 Sim-to-Real Correlation

Correlation between performance in real and simulated evaluations shows how predictive simulation results are of real world behavior. We therefore compare the results achieved by policies in our benchmarks to their real-world performance, which we take from RoboArena [29] and CAP [52]. We evaluate the correlation for the *pick*, *open*, and *close* tasks individually. Results for are shown in Fig. 11. For the *pick* task, we observe a strong linear correlation

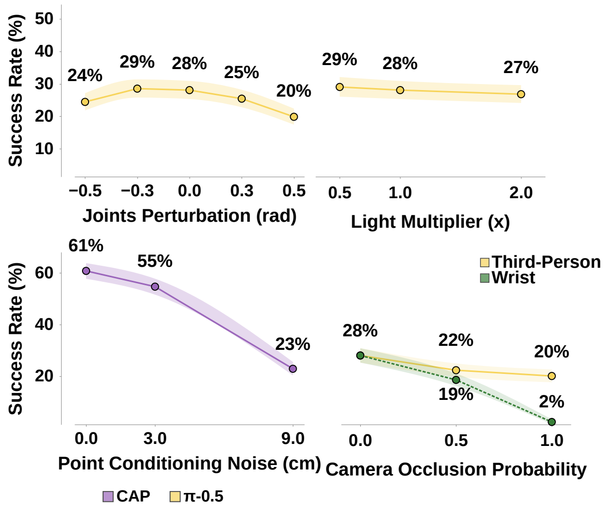


Figure 12 Robustness analysis under environmental perturbations. **Top:** Joint noise and lighting. **Bottom:** Point noise and camera occlusion.

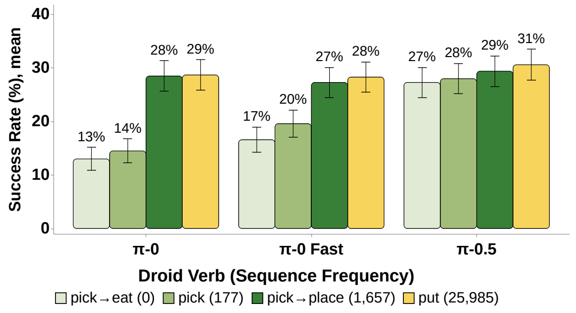


Figure 13 Prompt sensitivity: π models performance on the Pick-MSProc-1k benchmark with varying prompts. Frequent DROID prompts perform better.

between our MolmoSpaces-Bench results and the results from 752 RoboArena *pick* tasks, with Pearson and Spearman rank correlation coefficients of 0.96 and 0.98, respectively. This underlines MolmoSpaces-Bench's utility and predictive power. As in the Pick task, we compute correlations between benchmark success rates and real-world performance measured on RoboArena and CAP for the *open* and *close* tasks. While fewer real-world episodes are available for these tasks, we observe consistent positive correlations but with bigger error bars.

5.3 Distributional Evaluations

Machine learning models typically perform best under small distribution shifts between training and evaluation. Given that the π series models are trained on datasets that include DROID [53], we evaluate how performance degrades as we move away from this distribution. This evaluation is enabled by the scale of MolmoSpaces, which allows us to systematically vary environmental and policy factors beyond aggregate success metrics.

To probe language-induced distribution shift, we vary the *pick* task prompt using verb sequences of increasing frequency in DROID instructions, as shown in Figure 13. When using phrasing that are more frequent in the DROID dataset, π_0 achieves a success rate within 1% of $\pi_{0.5}$, compared to a 14% gap otherwise. With the assumption that the underlying DROID data distributions used to fine-tune π models are similar, this sensitivity to prompt phrasing suggests that generalization limitations of earlier π models arise from the language-conditioning component rather than the action heads. We observe this effect for the *pick* task in our benchmark and leave validation of its generality in other tasks to future work.

A similar degradation is observed when perturbing the initial joint positions of the $\pi_{0.5}$ policy from the default configuration used in DROID. As shown in Figure 12, varying the starting joint positions away from this default leads to a decline in performance. In contrast, varying the lighting intensity has little effect on performance, likely because such visual distribution shifts are mitigated through image-based data augmentation.

Figure 12 also illustrates the effect of other environmental perturbations on the performance of policies. In particular, occluding the wrist camera reduces $\pi_{0.5}$ success rate to 2%, while occluding the third-person camera only lowers the performance to 20%, indicating a strong reliance on wrist-mounted visual input. Similarly, CAP's performance indicates a strong reliance on a good starting conditioning point. Figure 15 shows that $\pi_{0.5}$ and CAP prefer different grasping approaches, with $\pi_{0.5}$ favoring top-down grasps and CAP favoring side grasps. This preference helps explain why $\pi_{0.5}$ performs better on objects with top openings, such as mugs and bowls, while CAP performs better than $\pi_{0.5}$ on objects where side grasps are feasible, such as bottles and apples, as shown in Figure 14.

[*]The real-world performance is obtained from RoboArena by filtering for pick tasks and using RoboArena's partial success criteria.

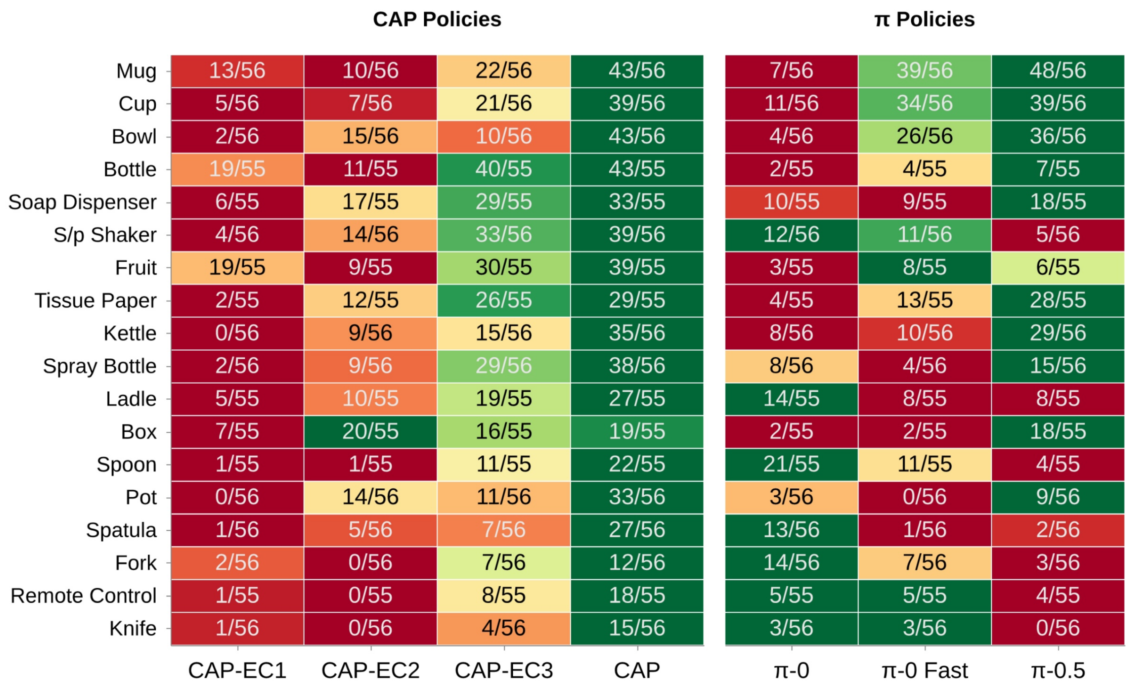


Figure 14 Per-object category success rates for CAP and π policy families on the pick task. Colors are normalized per row within each table.

5.4 Controlled Policy Comparison

To ensure a fair comparison between policies, we use a Franka FR3 arm setup as a unified embodiment for all manipulation benchmarks, alongside three RGB-D cameras: one wrist-mounted and two third-person. We allow a fixed offset in the robot base frame relative to the initial task configuration to accommodate grippers with differing geometries. This offset must be constant across episodes and should not use any privileged or task-specific information. During task generation, we filter out benchmark tasks that are not physically achievable under the standard DROID setup. We acknowledge that this filtering procedure may advantage grippers such as the RobotIQ 2F-85. We also acknowledge that the choice of end-effector can affect task difficulty, but consider this a part of system design to be evaluated.

Another variable that affects our recorded policies performance is the task horizon, which we set as 300 for the π models, and 50 for CAP. Figure 16 compares oracle termination with fixed-horizon termination of the π models, showing that policies sometimes reach a successful state but subsequently undo it before the episode ends. This behavior is also reflected in Table 7, where on average $\pi_{0.5}$ goes through 2.65 grasp-ungrasp transitions before exceeding the reward threshold in successful episodes, while π_0 makes 4.63 such transitions. These results suggest that π models benefit from sufficient time to retry actions, whereas CAP relies on VLM-based supervision for retries that is not used in our baseline benchmark experiments.

Table 7 Average number of grasp-ungrasp transitions before task success for π policies on the pick task.

Policy	Mean $\pm$ Std	Median
π-0	4.63 ± 5.26	3.0
π-0 Fast	2.02 ± 2.62	1.0
π-0.5	2.65 ± 2.74	1.0

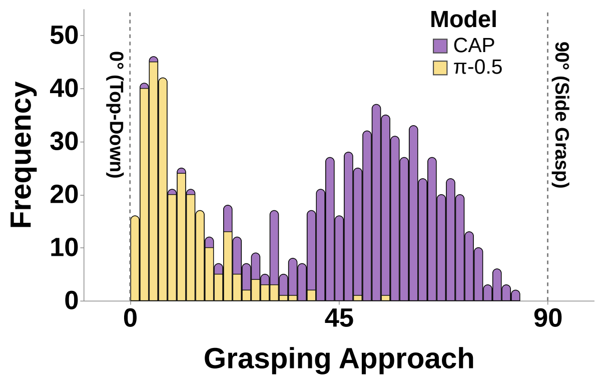


Figure 15 Grasp direction histogram of successful grasp of the $\pi_{0.5}$ and CAP policies.

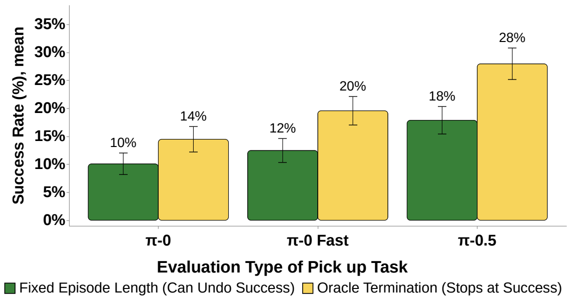


Figure 16 Comparison of oracle termination and fixed-horizon termination.

6 Conclusion

We present MolmoSpaces, a comprehensive open ecosystem comprising large-scale simulation environments, diverse object assets, and extensive grasp datasets. We further introduce MolmoSpaces-Bench, a benchmark suite spanning base skills and LLM-assisted long-horizon tasks, with controlled difficulty and perturbations that enable detailed analysis and principled comparison of generalist robot policies. We validate MolmoSpaces-Bench's utility with thorough experimentation, demonstrating strong sim-to-real correlation and benchmarking multiple SOTA policies, including distributional evaluations that reveal subtle policy behavior characteristics. Although simulation remains inherently imperfect, well-designed simulation benchmarks provide a critical foundation for evaluating robotic policies and guiding progress toward robust real-world performance. In the future, we plan to support data generation and reinforcement learning for robot policies at scale, enabling us to further study scaling behaviors for robot foundation models.

Author Contributions

This project was made possible through the equal contributions of all four co-first authors, in no particular order. Their individual contributions are as follows:

- **Yejin Kim**: Led the project and initially converted all assets and scenes from AI2-THOR to MuJoCo MJCF format; contributed to asset quality testings, grasp generation pipeline and evaluations, benchmark creation, and baseline evaluations.
- **Wilbert Pumacay**: Led the asset conversion and quality testings across MuJoCo, Isaac and ManiSkill simulators.
- **Omar Rayyan**: Led distributional benchmark evaluations, the addition of π, CAPs baselines and teleoperation data-collection; led the grasp generation pipeline and evaluations; contributed to benchmark creation.
- **Maximilian Argus**: Led the benchmark creation and baseline evaluations.

All other contributors are also deeply appreciated for their effort, which is critical to the success of the MolmoSpaces project. As not all of these can be captured, we indicate their primary contributing role in MolmoSpaces:

- For assets conversion and physical parameter tunings: Yejin Kim, Wilbert Pumacay, Winson Han, Eli VanderBilt, Jordi Salvador, and Shuo Liu.
- For grasp generation and evalautions: Omar Rayyan, and Yejin Kim.
- For benchmark creation and baseline evaluation: Maximilian Argus, Omar Rayyan, Yejin Kim, Arjun Guru, Maya Guru, Abhay Despande, Rose Hendrix, Snehal Jauhri, Shuo Liu, Ainaz Eftekhar, Ying-Chun Lee, and Piper Wolters. Nur Muhammad Mahi Shafiullah advised the zero-shot and distributional evaluations.
- For multi-simulator support: Wilbert Pumacay, Alvaro Herrasti, and Donovan Clay
- For paper writing and figures: Yejin Kim, Maximilian Argus, Wilbert Pumacay, Omar Rayyan, Winson Han, Eli VanderBilt, Abhay Deshpande, Rose Hendrix, Maya Guru, Arjun Guru, Jordi Salvador, Jiafei Duan, Nur Muhammad Mahi Shafiullah, and Ranjay Krishna.
- For project management: Karen Farley.
- For research advisory: Ranjay Krishna, Dieter Fox, and Ali Farhadi.
- Project PI: Ranjay Krishna

Acknowledgment

This work would not be possible without the support of our colleagues at Ai2:

- We thank Rachel Ratner and Karen Goodfellow for the creation and the support of robot demo webpage for MolmoSpaces.
- We thank David Albright, Crystal Nam, Kristin Cha, Sophie Lebrecht, Taira Anderson, Kyle Wiggers, Kelsey MacMillan, Katie Morigi, and Megan Bartot for project management, support to robot room and publicity of MolmoSpaces
- We thank Yoganand Chandrasekhar, Johann Dahm, Fangzhou Hu, and Caroline Wu for their work on the Ai2 cluster.

References

[1] Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A vision-language-action flow model for general robot control, 2026. URL https://arxiv.org/abs/2410.24164.

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

[3] Gemini Robotics Team, Saminda Abeyruwan, Joshua Ainslie, Jean-Baptiste Alayrac, Montserrat Gonzalez Arenas, Travis Armstrong, Ashwin Balakrishna, Robert Baruch, Maria Bauza, Michiel Blokzijl, et al. Gemini robotics: Bringing ai into the physical world. *arXiv preprint arXiv:2503.20020*, 2025.

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

[5] Martin Sedlacek, Pavlo Yefanov, Georgy Ponimatkin, Jai Bardhan, Simon Pilc, Mederic Fourmy, Evangelos Kazakos, Cees GM Snoek, Josef Sivic, and Vladimir Petrik. Realm: A real-to-sim validated benchmark for generalization in robotic manipulation. *arXiv preprint arXiv:2512.19562*, 2025.

[6] Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. *Advances in Neural Information Processing Systems*, 36:44776–44791, 2023.

[7] Oier Mees, Lukas Hermann, Erick Rosete-Beas, and Wolfram Burgard. Calvin: A benchmark for language-conditioned policy learning for long-horizon robot manipulation tasks, 2022. URL https://arxiv.org/abs/2112.03227.

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

[9] Senyu Fei, Siyin Wang, Junhao Shi, Zihao Dai, Jikun Cai, Pengfang Qian, Li Ji, Xinzhe He, Shiduo Zhang, Zhaoye Fei, Jinlan Fu, Jingjing Gong, and Xipeng Qiu. Libero-plus: In-depth robustness analysis of vision-language-action models. *CoRR*, abs/2510.13626, 2025. doi: 10.48550/ARXIV.2510.13626. URL https://doi.org/10.48550/arXiv.2510.13626.

[10] Xueyang Zhou, Yangming Xu, Guiyao Tie, Yongchao Chen, Guowen Zhang, Duanfeng Chu, Pan Zhou, and Lichao Sun. Libero-pro: Towards robust and fair evaluation of vision-language-action models beyond memorization, 2025. URL https://arxiv.org/abs/2510.03827.

[11] Anonymous. Robotarena infinity: Unlimited robot benchmarking via real-to-sim translation. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=OutljIofvS.

[12] Arhan Jain, Mingtong Zhang, Kanav Arora, William Chen, Marcel Torne, Muhammad Zubair Irshad, Sergey Zakharov, Yue Wang, Sergey Levine, Chelsea Finn, Wei Ma, Dhruv Shah, Abhishek Gupta, and Karl Pertsch. Polaris: Scalable real-to-sim evaluations for generalist robot policies, 2025.

[13] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033, 2012.

[14] NVIDIA. Isaac Sim, 2026. URL https://github.com/isaac-sim/IsaacSim.

[15] Tongzhou Mu, Z. 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. In *NeurIPS Datasets and Benchmarks*, 2021.

[16] Karl Pertsch, Kyle Stachowicz, Brian Ichter, Danny Driess, Suraj Nair, Quan Vuong, Oier Mees, Chelsea Finn, and Sergey Levine. Fast: Efficient action tokenization for vision-language-action models. *arXiv preprint arXiv:2501.09747*, 2025.

[17] Chenghao Yin, Da Huang, Di Yang, Jichao Wang, Nanshu Zhao, Chen Xu, Wenjun Sun, Linjie Hou, Zhijun Li, Junhui Wu, Zhaobo Liu, Zhenfei Xiao, Shenglan 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.

[18] Eric Kolve, Roozbeh Mottaghi, Daniel Gordon, Yuke Zhu, Abhinav Gupta, and Ali Farhadi. AI2-THOR: an interactive 3d environment for visual AI. *CoRR*, abs/1712.05474, 2017. URL http://arxiv.org/abs/1712.05474.

[19] Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, et al. Habitat: A platform for embodied ai research. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9339–9347, 2019.

[20] Andrew Szot, Alex Clegg, Eric Undersander, Erik Wijmans, Yili Zhao, John Turner, Noah Maestre, Mustafa Mukadam, Devendra Chaplot, Oleksandr Maksymets, Aaron Gokaslan, Vladimir Vondrus, Sameer Dharur, Franziska Meier, Wojciech Galuba, Angel Chang, Zsolt Kira, Vladlen Koltun, Jitendra Malik, Manolis Savva, and Dhruv Batra. Habitat 2.0: Training home assistants to rearrange their habitat, 2022. URL https://arxiv.org/abs/2106.14405.

[21] Matt Deitke, Eli VanderBilt, Alvaro Herrasti, Luca Weihs, Kiana Ehsani, Jordi Salvador, Winson Han, Eric Kolve, Aniruddha Kembhavi, and Roozbeh Mottaghi. Procthor: Large-scale embodied AI using procedural generation. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022. URL http://papers.nips.cc/paper_files/paper/2022/hash/27c546ab1e4f1d7d638e6a8dfbad9a07-Abstract-Conference.html.

[22] Soroush Nasiriany, Abhiram Maddukuri, Lance Zhang, Adeet Parikh, Aaron Lo, Abhishek Joshi, Ajay Mandlekar, and Yuke Zhu. Robocasa: Large-scale simulation of everyday tasks for generalist robots. In *Robotics: Science and Systems*, 2024.

[23] Anonymous. Robocasa365: A large-scale simulation framework for training and benchmarking generalist robots. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=tQJYKwc3n4.

[24] Abby O'Neill, Abdul Rehman, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, et al. Open x-embodiment: Robotic learning datasets and rt-x models: Open x-embodiment collaboration 0. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6892–6903. IEEE, 2024.

[25] 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 *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2023, Vancouver, BC, Canada, June 17-24, 2023*, pages 13142–13153. IEEE, 2023. doi: 10.1109/CVPR52729.2023.01263. URL https://doi.org/10.1109/CVPR52729.2023.01263.

[26] Yue Yang, Fan-Yun Sun, Luca Weihs, Eli VanderBilt, Alvaro Herrasti, Winson Han, Jiajun Wu, Nick Haber, Ranjay Krishna, Lingjie Liu, Chris Callison-Burch, Mark Yatskar, Aniruddha Kembhavi, and Christopher Clark. Holodeck: Language guided generation of 3d embodied AI environments. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2024, Seattle, WA, USA, June 16-22, 2024*, pages 16277–16287. IEEE, 2024. doi: 10.1109/CVPR52733.2024.01536. URL https://doi.org/10.1109/CVPR52733.2024.01536.

[27] Weipeng Zhong, Peizhou Cao, Yichen Jin, Li Luo, Wenzhe Cai, Jingli Lin, Hanqing Wang, Zhaoyang Lyu, Tai Wang, Bo Dai, Xudong Xu, and Jiangmiao Pang. Internscenes: A large-scale simulatable indoor scene dataset with realistic layouts, 2025. URL https://arxiv.org/abs/2509.10813.

[28] Adithyavairavan Murali, Balakumar Sundaralingam, Yu-Wei Chao, Wentao Yuan, Jun Yamada, Mark T. Carlson, Fabio Ramos, Stanley T. Birchfield, Dieter Fox, and Clemens Eppner. Graspgen: A diffusion-based framework for 6-dof grasping with on-generator training. *ArXiv*, abs/2507.13097, 2025.

[29] Pranav Atreya, Karl Pertsch, Tony Lee, Moo Jin Kim, Arhan Jain, Artur Kuramshin, Clemens Eppner, Cyrus Neary, Edward Hu, Fabio Ramos, Jonathan Tremblay, Kanav Arora, Kirsty Ellis, Luca Macesanu, Matthew Leonard, Meedeum Cho, Ozgur Aslan, Shivin Dass, Jie Wang, Xingfang Yuan, Xuning Yang, Abhishek Gupta, Dinesh Jayaraman, Glen Berseth, Kostas Daniilidis, Roberto Martin Martin, Youngwoon Lee, Percy Liang, Chelsea Finn, and Sergey Levine. Roboarena: Distributed real-world evaluation of generalist robot policies. *ArXiv*, abs/2506.18123, 2025.

[30] Zhiyuan Zhou, Pranav Atreya, You Liang Tan, Karl Pertsch, and Sergey Levine. Autoeval: Autonomous evaluation of generalist robot manipulation policies in the real world. *arXiv preprint arXiv:2503.24278*, 2025.

[31] Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer, and Dieter Fox. Alfred: A benchmark for interpreting grounded instructions for everyday tasks, 2020. URL https://arxiv.org/abs/1912.01734.

[32] Karmesh Yadav, Jacob Krantz, Ram Ramrakhya, Santhosh Kumar Ramakrishnan, Jimmy Yang, Austin Wang, John Turner, Aaron Gokaslan, Vincent-Pierre Berges, Roozbeh Mootaghi, Oleksandr Maksymets, Angel X Chang, Manolis Savva, Alexander Clegg, Devendra Singh Chaplot, and Dhruv Batra. Habitat challenge 2023. https://aihabitat.org/challenge/2023/, 2023.

[33] 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.

[34] Shiduo Zhang, Zhe Xu, Peiju Liu, Xiaopeng Yu, Yuan Li, Qinghui Gao, Zhaoye Fei, Zhangyue Yin, Zuxuan Wu, Yu-Gang Jiang, and Xipeng Qiu. Vlabench: A large-scale benchmark for language-conditioned robotics manipulation with long-horizon reasoning tasks. *CoRR*, abs/2412.18194, 2024. doi: 10.48550/ARXIV.2412.18194. URL https://doi.org/10.48550/arXiv.2412.18194.

[35] Wilbert Pumacay, Ishika Singh, Jiafei Duan, Ranjay Krishna, Jesse Thomason, and Dieter Fox. The colosseum: A benchmark for evaluating generalization for robotic manipulation. *arXiv preprint arXiv:2402.08191*, 2024.

[36] Zhijie Wang, Zhehua Zhou, Jiayang Song, Yuheng Huang, Zhan Shu, and Lei Ma. Vlatest: Testing and evaluating vision-language-action models for robotic manipulation. *Proc. ACM Softw. Eng.*, 2(FSE), jul 2025. doi: 10.1145/3729343. URL https://doi.org/10.1145/3729343.

[37] Chengshu Li, Ruohan Zhang, Josiah Wong, Cem Gokmen, Sanjana Srivastava, Roberto Martín-Martín, Chen Wang, Gabrael Levine, Michael Lingelbach, Jiankai Sun, Mona Anvari, Minjune Hwang, Manasi Sharma, Arman Aydin, Dhruva Bansal, Samuel Hunter, Kyu-Young Kim, Alan Lou, Caleb R Matthews, Ivan Villa-Renteria, Jerry Huayang Tang, Claire Tang, Fei Xia, Silvio Savarese, Hyowon Gweon, Karen Liu, Jiajun Wu, and Li Fei-Fei. BEHAVIOR-1k: A benchmark for embodied AI with 1,000 everyday activities and realistic simulation. In *6th Annual Conference on Robot Learning*, 2022. URL https://openreview.net/forum?id=_8DoIe8G3t.

[38] Arth Shukla, Stone Tao, and Hao Su. Maniskill-hab: A benchmark for low-level manipulation in home rearrangement tasks. *ArXiv*, abs/2412.13211, 2024.

[39] Rui Yang, Hanyang Chen, Junyu Zhang, Mark Zhao, Cheng Qian, Kangrui Wang, Qineng Wang, Teja Venkat Koripella, Marziyeh Movahedi, Manling Li, Heng Ji, Huan Zhang, and Tong Zhang. Embodiedbench: Comprehensive benchmarking multi-modal large language models for vision-driven embodied agents, 2025. URL https://arxiv.org/abs/2502.09560.

[40] Xuanlin Li, Kyle Hsu, Jiayuan Gu, Karl Pertsch, Oier Mees, Homer Rich Walke, Chuyuan Fu, Ishikaa Lunawat, Isabel Sieh, Sean Kirmani, Sergey Levine, Jiajun Wu, Chelsea Finn, Hao Su, Quan Vuong, and Ted Xiao. Evaluating real-world robot manipulation policies in simulation. *arXiv preprint arXiv:2405.05941*, 2024.

[41] Jianxiong Xiao, James Hays, Krista A. Ehinger, Aude Oliva, and Antonio Torralba. SUN database: Large-scale scene recognition from abbey to zoo. In *The Twenty-Third IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2010, San Francisco, CA, USA, 13-18 June 2010*, pages 3485–3492. IEEE Computer Society, 2010. doi: 10.1109/CVPR.2010.5539970. URL https://doi.org/10.1109/CVPR.2010.5539970.

[42] Tao Ge, Xin Chan, Xiaoyang Wang, Dian Yu, Haitao Mi, and Dong Yu. Scaling synthetic data creation with 1,000,000,000 personas. *arXiv preprint arXiv:2406.20094*, 2024.

[43] Xinyue Wei, Minghua Liu, Zhan Ling, and Hao Su. Approximate convex decomposition for 3d meshes with collision-aware concavity and tree search. *ACM Transactions on Graphics (TOG)*, 41(4):1–18, 2022.

[44] The Allen Institue for AI. objathor, 2024. URL https://github.com/allenai/objathor. GitHub repository.

[45] John P. McCrae, Alexandre Rademaker, Francis Bond, Ewa Rudnicka, and Christiane Fellbaum. English wordnet 2019 - an open-source wordnet for english. In Piek Vossen and Christiane Fellbaum, editors, *Proceedings of the 10th Global Wordnet Conference, GWC 2019, Wroclaw, Poland, July 23-27, 2019*, pages 245–252. Global Wordnet Association, 2019. URL https://aclanthology.org/2019.gwc-1.31/.

[46] John P. McCrae and Alexandre Rademaker and Ewa Rudnicka and Bernard Bou and Daiki Nomura and David Cillessen and Ciara O'Loughlin and Cathal McGovern and Francis Bond and Eric Kafe and Michael Wayne Goodman and Merrick Choo Yeu Herng and Enejda Nasaj. Open English Wordnet, 2022. URL https://github.com/globalwordnet/english-wordnet/releases/tag/2022-edition. GitHub repository.

[47] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[48] Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. GPT-4o system card. *arXiv preprint arXiv:2410.21276*, 2024.

[49] OpenAI. GPT-4.1, 2024. URL https://platform.openai.com/. Proprietary large language model accessed via the OpenAI API; no public system card available.

[50] Arsalan Mousavian, Clemens Eppner, and Dieter Fox. 6-dof graspnet: Variational grasp generation for object manipulation. In *International Conference on Computer Vision (ICCV)*, 2019.

[51] Clemens Eppner, Arsalan Mousavian, and Dieter Fox. Acronym: A large-scale grasp dataset based on simulation. *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6222–6227, 2020.

[52] Zichen Jeff Cui, Omar Rayyan, Haritheja Etukuru, Bowen Tan, Zavier Andrianarivo, Zicheng Teng, Yihang Zhou, Krish Mehta, Nicholas Wojno, Kevin Yuanbo Wu, Manan H Anjaria, Ziyuan Wu, Manrong Mao, Guangxun Zhang, Binit Shah, Yejin Kim, Soumith Chintala, Lerrel Pinto, and Nur Muhammad Mahi Shafiullah. Contact-anchored policies: Contact conditioning creates strong robot utility models. *arXiv preprint arXiv:2602.09017*, 2026.

[53] Alexander Khazatsky, Karl Pertsch, Suraj Nair, Ashwin Balakrishna, Sudeep Dasari, Siddharth Karamcheti, Soroush Nasiriany, Mohan Kumar Srirama, Lawrence Yunliang Chen, Kirsty Ellis, Peter David Fagan, Joey Hejna, Masha Itkina, Marion Lepert, Yecheng Jason Ma, Patrick Tree Miller, Jimmy Wu, Suneel Belkhale, Shivin Dass, Huy Ha, Arhan Jain, Abraham Lee, Youngwoon Lee, Marius Memmel, Sungjae Park, Ilija Radosavovic, Kaiyuan Wang, Albert Zhan, Kevin Black, Cheng Chi, Kyle Beltran Hatch, Shan Lin, Jingpei Lu, Jean Mercat, Abdul Rehman, Pannag R Sanketi, Archit Sharma, Cody Simpson, Quan Vuong, Homer Rich Walke, Blake Wulfe, Ted Xiao, Jonathan Heewon Yang, Arefeh Yavary, Tony Z. Zhao, Christopher Agia, Rohan Baijal, Mateo Guaman Castro, Daphne Chen, Qiuyu Chen, Trinity Chung, Jaimyn Drake, Ethan Paul Foster, Jensen Gao, Vitor Guizilini, David Antonio Herrera, Minho Heo, Kyle Hsu, Jiaheng Hu, Muhammad Zubair Irshad, Donovon Jackson, Charlotte Le, Yunshuang Li, Kevin Lin, Roy Lin, Zehan Ma, Abhiram Maddukuri, Suvir Mirchandani, Daniel Morton, Tony Nguyen, Abigail O'Neill, Rosario Scalise, Derick Seale, Victor Son, Stephen Tian, Emi Tran, Andrew E. Wang, Yilin Wu, Annie Xie, Jingyun Yang, Patrick Yin, Yunchu Zhang, Osbert Bastani, Glen Berseth, Jeannette Bohg, Ken Goldberg, Abhinav Gupta, Abhishek Gupta, Dinesh Jayaraman, Joseph J Lim, Jitendra Malik, Roberto Martín-Martín, Subramanian Ramamoorthy, Dorsa Sadigh, Shuran Song, Jiajun Wu, Michael C. Yip, Yuke Zhu, Thomas Kollar, Sergey Levine, and Chelsea Finn. Droid: A large-scale in-the-wild robot manipulation dataset, 2024.

[54] Kevin Zakka, Yuval Tassa, and MuJoCo Menagerie Contributors. MuJoCo Menagerie: A collection of high-quality simulation models for MuJoCo, 2022. URL `http://github.com/google-deepmind/mujoco_menagerie`.

[55] Balakumar Sundaralingam, Siva Kumar Sastry Hari, Adam Fishman, Caelan Reed Garrett, Karl Van Wyk, Valts Blukis, Alexander Millane, Helen Oleynikova, Ankur Handa, Fabio Tozeto Ramos, Nathan D. Ratliff, and Dieter Fox. Curobo: Parallelized collision-free robot motion generation. *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 8112–8119, 2023.

[56] Omar Rayyan, Maximilian Gilles, and Yuchen Cui. Teledex: Accessible dexterous teleoperation, 2026. URL `https://github.com/omarrayyann/teledex`.

[57] Kiana Ehsani, Tanmay Gupta, Rose Hendrix, Jordi Salvador, Luca Weihs, Kuo-Hao Zeng, Kunal Pratap Singh, Yejin Kim, Winson Han, Alvaro Herrasti, Ranjay Krishna, Dustin Schwenk, Eli VanderBilt, and Aniruddha Kembhavi. Spoc: Imitating shortest paths in simulation enables effective navigation and manipulation in the real world. *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16238–16250, 2023. URL `https://api.semanticscholar.org/CorpusID:271769346`.

[58] Gemini Team, Petko Georgiev, Ving Ian Lei, Ryan Burnell, Libin Bai, Anmol Gulati, Garrett Tanzer, Damien Vincent, Zhufeng Pan, Shibo Wang, et al. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*, 2024.

[59] Ainaz Eftekhar, Luca Weihs, Rose Hendrix, Ege Caglar, Jordi Salvador, Alvaro Herrasti, Winson Han, Eli VanderBil, Aniruddha Kembhavi, Ali Farhadi, Ranjay Krishna, Kiana Ehsani, and Kuo-Hao Zeng. The one ring: a robotic indoor navigation generalist. *ArXiv*, abs/2412.14401, 2024.

[60] Meng Wei, Chenyang Wan, Jiaqi Peng, Xiqian Yu, Yuqiang Yang, Delin Feng, Wenzhe Cai, Chenming Zhu, Tai Wang, Jiangmiao Pang, and Xihui Liu. Ground slow, move fast: A dual-system foundation model for generalizable vision-and-language navigation. *ArXiv*, abs/2512.08186, 2025. URL `https://api.semanticscholar.org/CorpusID:283711459`.

[61] Chendi Lin, Heshan Liu, Qunshu Lin, Zachary Bright, Shitao Tang, Yihui He, Minghao Liu, Ling Zhu, and Cindy Le. Objaverse++: Curated 3D Object Dataset with Quality Annotations. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*, pages 6813–6822, October 2025.

[62] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *ICML*, 2021.

[63] Gabriel Ilharco, Mitchell Wortsman, Ross Wightman, Cade Gordon, Nicholas Carlini, Rohan Taori, Achal Dave, Vaishaal

Shankar, Hongseok Namkoong, John Miller, Hannaneh Hajishirzi, Ali Farhadi, and Ludwig Schmidt. Openclip, July 2021. URL https://doi.org/10.5281/zenodo.5143773.

[64] Mehdi Cherti, Romain Beaumont, Ross Wightman, Mitchell Wortsman, Gabriel Ilharco, Cade Gordon, Christoph Schuhmann, Ludwig Schmidt, and Jenia Jitsev. Reproducible scaling laws for contrastive language-image learning. In *CVPR*, 2023.

Appendix

A Object Model Dataset Details

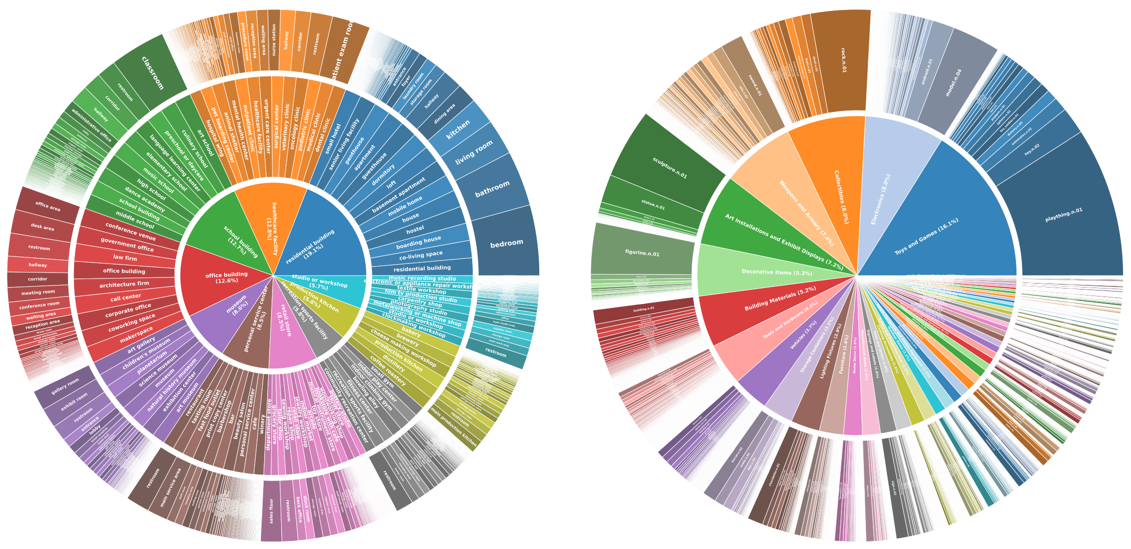

~110k MolmoSpaces-Scenes-MultiType scene specifications ~129k Objaverse objects

Figure 17 Scene specification distribution (left) –with generic and concrete scene types and room types– used to generate MolmoSpaces-Scenes-MultiType scenes. Between one and ten rooms of mostly scene-specific room types (here shown aggregated per generic scene type) are chosen to prompt LLMs. Right, distribution of WordNet synsets grouped by higher-level object categories in our curated subset of Objaverse.

We provide models from both AI2-THOR and Objaverse. We extracted and converted objects from AI2-THOR and migrated them into a file format compatible with MuJoCo as well as other simulators such as ManiSkill and IsaacSim. In total, we converted 1.6k rigid object instances across 134 categories into MuJoCo. In addition, there are 22 categories—including doors, refrigerators, and dressers—that were made articulable by annotating joint information, including joint type (slide or hinge), joint axis, joint position, and joint range.

To ensure physical realism for robot manipulation tasks, we performed several validation steps. For rigid objects, we verified that mass and density values were realistic by comparing simulated values with estimates annotated via large language models (LLMs), adjusting density as needed. For articulable objects, we created a teleoperation suite to manually tune the physical properties of joints and the density of movable parts. This was performed using a simulated Franka FR3 robot, which itself was tuned through system identification: real robot trajectories of picking and pushing cubes of known weights were collected, and simulation parameters were optimized to match the observed motions. These physics parameters were then applied to all objects during scene generation.

Collider meshes for all objects were generated using COACD [43] , and we also annotated primitive colliders for all THOR objects. For physics stability, rigid objects with receptacles (e.g., tables, dressers) primarily use primitive colliders to avoid mesh-mesh contact issues. Manipulable objects require higher fidelity, so convex decomposition was used; however, for very small and thin objects, primitive colliders were employed to maintain simulation stability. Meshes were further processed and decimated to improve simulation performance.

In addition to AI2-THOR objects, we converted a curated selection of Objaverse objects into the MJCF format for MuJoCo. This required a careful curation methodology to ensure quality and compatibility, which follows multiple filtering stages applied on top of an initial subset of 625k Objaverse objects pre-filtered with their original metadata, which are (1) converted to a format compatible with AI2-THOR [18, 44] and (2) annotated with VLM-generated [48] descriptions, estimated mass, canonical poses, pickable/receptacle properties, matching Wordnet synsets [45, 46], and scale estimates obtained via prompting with renderings from different viewpoints.

The automatic process results in some malformed or unreliable annotations, which prompts us to perform a complementary GPT-4.1 annotation (LLM prompt listed in Fig. 20) to determine (1) the type and number of object instances in each model file, (2) the presence of props or other excessive geometry not part of the main object in the model file, (3) missing geometry, which typically occurs with scans of real-world objects, (4) a texture quality score on a 0-9 scale, and (5) whether the model contains receptacle parts. While we seek extraction of very specific information to help filter objects towards usability for embodied agent training, we note other efforts in re-annotating Objaverse like [61].

To provide a subset of objects suitable for LLM-backed scene generation, we sequentially apply filtering stages to ensure (1) metadata completeness and synset validity, (2) presence of a single object in the model file, (3) statistical scale inliers using a restrictive Tukey rule on the IQR of the log-scale, (4) sufficient texture quality with annotated score 4 or higher, (5) cross–renderer fidelity with CLIP [62–64] similarity score 0.6 or higher, (6) processed model file size with compressed geometry and colliders less than 1.5 MB, (7) agreement on the receptacle property in both annotations, (8) watertight colliders with at least 80% of the horizontal surface for receptacle objects, and (9) final removal of singleton representatives of synsets. The resulting distribution of curated objects, corresponding to almost 2.8k different WordNet synsets, is illustrated by Fig. 17 (right). The count of nearly 129k curated objects, which more than doubles the available amount in prior work on LLM-backed scene generation [26], is finally divided into train (80%), validation (10%), and test (10%) splits.

We further extend the filtering stages to provide compatibility with procedural house generation via ProcTHOR [21] relying on annotated placement options (prompt used with GPT-4o listed in Fig. 22). Rather than generating new placement options for all synsets, we rely on the already diverse set of THOR object categories and map each synset to the most relevant THOR category, if any, using scale and semantic similarity-based heuristics. In more detail, for each candidate pairing, we compute a weighted dissimilarity score based on room-type compatibility, feasible placement locations, object scale ranges, boolean affordances, and calibrated WordNet semantic similarity. Candidate matches are filtered using hard constraints on room and scale compatibility, followed by a lightweight reranking stage that emphasizes semantic alignment. Each synset is assigned to the lowest-cost THOR category below a fixed threshold, yielding a robust many-to-one alignment suitable for downstream scene generation. We then let the procedural engine sample objects from any synset associated with the currently selected THOR category. Using these heuristic placement reference assignments, the additional filtering stages ensure that objects (1) have synsets with valid placement references, (2) have annotated pickable/receptacle properties compatible with those of the placement reference, (3) have bounding box scales suitable for placement in houses, (4) are sufficiently shallow if wall-placeable, (5) have synsets that are not hyponyms of weapons, and (6) have synsets present in the train split. The result of applying these additional filters is a curated subset of 92.5k objects, corresponding to $\sim$2k synsets.

For all objects, we provide an extensive metadata collection that includes the physical parameters of scale and mass as well as semantic information: name, category, and synsets. To enable the easy use of these models in robotics simulations, we also provide convex meshes and grasps. In addition to this, we make sure that these objects are defined in a canonical coordinate system to make them easily loadable into scenes.

B MolmoSpaces-Scenes-MultiType Generation Details

Grid layout placement option As mentioned in Sec. 3.1, we extend the original DFS-based floor object solver in Holodeck [26] with a 'grid' global constraint that enables structured batch placement of multiple objects suitable for several non-residential MSMultiType scene types. Objects to be placed with a grid constraint are handled jointly: the solver enumerates feasible grid shapes (rows $\times$ columns) that can accommodate an iteratively decreasing object set –starting with the original object count requested by the LLM–, prioritizing compact layouts. For each candidate grid shape, the solver attempts to place the entire grid footprint within the current room and greedily assigns objects to grid cells by selecting positions and rotations that maximize satisfaction of remaining (non-grid) constraints regarding already placed objects. Grid configurations are scored by aggregating per-object constraint satisfaction, including an additional soft bias discouraging obstruction of door-to-door circulation. The highest-scoring grid placement is retained before resuming DFS over the remaining objects. We optionally apply a small positional and rotational jitter to grid-constrained objects, retaining only collision-free perturbations. Fig. 18 illustrates how the new grid layout modality can simplify the task of placing uniformly spaced objects.

Style accents with persona descriptions Persona descriptions conveying stylistic information were filtered from the full set in PersonaHub [42] via GPT-4.1-mini prompted as shown in Fig. 23. The effects of adding samples from this filtered set to a scene specification via the simple binding structure used to generate the $\sim$110k scenes in MSMultiType are illustrated by Fig. 19.

Original Holodeck [26] prompt	No edge placement emphasis	New prompt with grid layout
`... edge: at the edge of the room, close to the wall, most of the objects are placed here... I prefer objects to be placed at the edge (the most important constraint) of the room if possible, which makes the room look more spacious...`	`... edge: at the edge of the room, close to the wall - , most of the objects are placed here... I prefer objects to be placed at the edge (the most important constraint) of the room if possible, which makes the room look more spacious...`	`... + Using a grid constraint helps me create more realistic layouts when placing multiple objects in a repetitive pattern across a floor area that are meant to be away from room edges. Since grids are likely to occupy a considerable amount of space, it is convenient to place them as early as possible after the anchor object...`

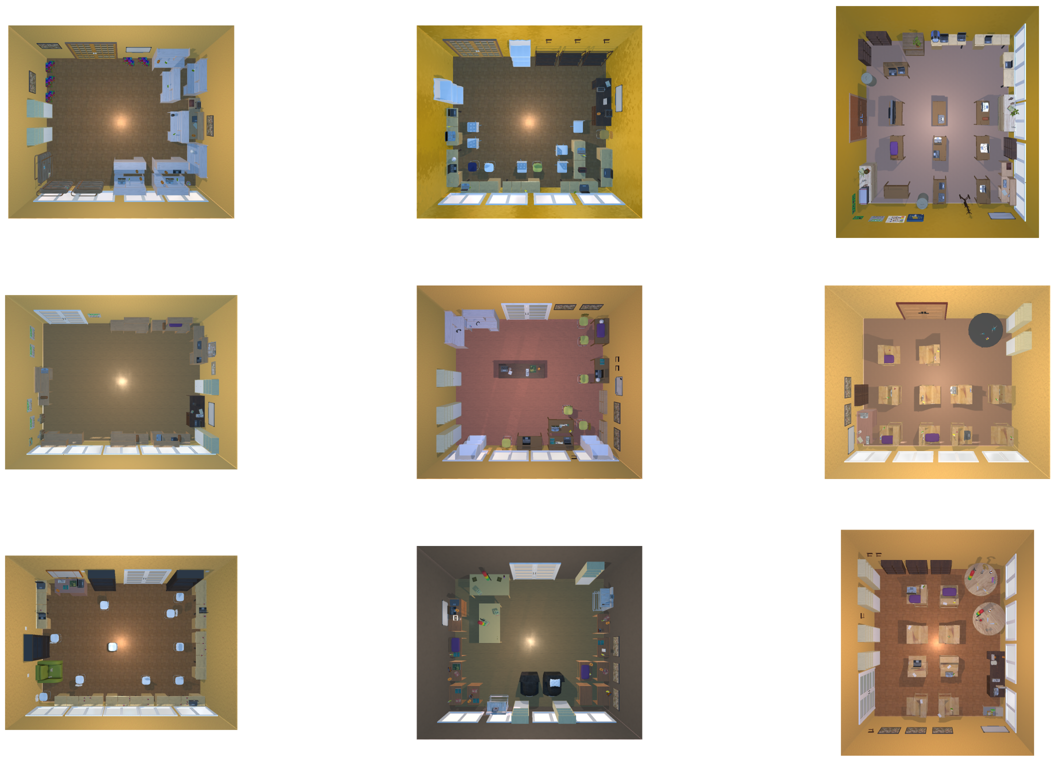


Figure 18 Object placement versus inclusion of the ‘grid’ layout constraint. In all samples, the textual scene specification in the LLM prompt is ‘`a primary school classroom`’. In the bottom row we additionally disable a bias term meant to improve door-to-door navigability in multi-room scenes. Text in red (green) indicates removal (addition).

`a waiting room`	`the favorite waiting room of a person who identifies as ‘A vintage vinyl record collector who is challenged to keep their growing collection in check‘`	`the favorite waiting room of a person that identifies as ‘A curator specializing in health and science exhibits, constantly seeking the epidemiologist's input to ensure accuracy and educational value‘`	`the favorite waiting room of a person who identifies as ‘A former PBA basketball player who still holds a grudge against the alumni for leaving the team‘`

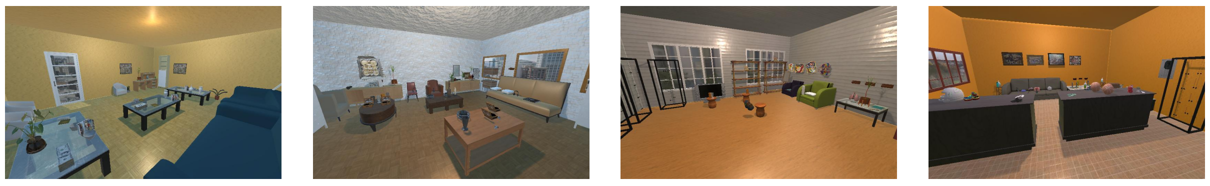


Figure 19 Complementing scene specifications with persona descriptions produces notable stylistic changes and inclusion of a wider variety of object types in LLM-generated scenes.

```
SYSTEM PROMPT:

You are an expert in 3D model analysis.  Your main task is to identify 3D models that contain more than
a single object (e.g., kitchens with cabinets and sinks, dining tables with chairs, or assortments of objects of
some or multiple categories, to name a few examples). You also decide whether the 3D model seems to have missing
geometry, e.g. resulting from an incomplete scan, or excessive geometry, e.g., resulting from a scan that also
reconstructs part of the supporting surface or background, or due to additional props added as decoration for
rendering.  Among others, a model without excessive geometry should not make the main object appear as being
mounted or placed on any type of surface or structure.  Finally, you also decide whether the model's object
contains surfaces that can be used as receptacles of smaller object types (like seats of a chair or sofa, or top
of a table, among others), and whether the model seems realistically textured for an object of the given category.

The provided data is a collage of renders of the 3D model from different perspectives, showing the
front, left, back, and right sides. The expected background should be a flat white color, with everything else
being part of the model.

Your decisions should be structured as a JSON dict with the following entries:
    - ‘object types’: List[str], with each entry corresponding to an object type present in the model, e.g.,
      [’shoe’] for a model of a pair of shoes;
    - ‘object instances’: Dict[str, int], with an approximate count of instances of each type, e.g., ’book’: 3
      for a model of a pile of 3 books);
    - ‘main object type’: str, type of either largest object, or the most meaningful one, e.g., ’plate’ for a
      model of a plate with a fork on top;
    - ‘supporting structures’: List[str], typically empty, with names of visible structures supporting the main
      object, e.g. fragments (or panels) of tables, walls, floors, rugs, etc.;
    - ‘excessive geometry’: str, typically ‘No excessive geometry’, briefly describing the presence of supporting
      or background surfaces possibly added to the model as decoration for rendering or due to poor segmentation
      of a reconstructed or scanned model;
    - ‘has excessive geometry’: bool, providing a binary response from the analysis under ‘excessive geometry’;
    - ‘detachable part types’: List[str], typically empty, including easily detachable parts of the main object
      that should not be considered additional objects as they appear in place, e.g., [‘light bulb’, ‘lamp
      shade’] for a model of a table lamp with those parts visible and mounted at their proper places on the
      main object;
    - ‘is single object’: bool, considering all previous responses, whether the model only contains one object
      (no additional objects, other than possibly detachable parts that appear properly installed on the object)
      and does not include excessive geometry of any kind;
...
```

Figure 20 System prompt used with GPT-4.1 to generate alternative annotation for Objaverse object assets filtering in batch mode (continues in Fig.21).

Table 8 Comparison of large-scale grasp datasets. MolmoSpaces provides the highest number of object instances.

Dataset	**Year**	**# Objects**	**# Grasps**
HO-3D	2020	10	78,000
EGAD	2020	2,331	233,000
DDG	2020	500	50,000
DexYCB	2021	20	582,000
Acronym	2021	8,872	17,000,000
UniGrasp	2020	1,000	2,000,000
DexGraspNet	2023	5,355	1,300,000
Fast-Grasp’D	2023	2,350	1,000,000
GenDexGrasp	2023	58	436,000
MultiGripperGrasp	2024	345	30,400,000
GraspGen	2025	8,515	**53,100,000**
MolmoSpaces-Grasps (Ours)	**2026**	**48,675**	42,000,000

```
...
    - ‘missing geometry’: str, typically ‘No missing geometry’, briefly describing where the model appears to
      have missing geometry, e.g., no geometry around the back side, resulting in empty render or some distorted
      view of the front one for that perspective. Do not take into consideration whether the model appears to
      be lacking textures;
    - ‘has missing geometry’: bool, providing a binary response from the analysis under ‘missing geometry’;
    - ‘receptacles’: List[str], generally empty for small objects, including names of the main object’s surfaces
      that can be used as receptacles for smaller objects, as their normals are oriented upward and have
      sufficient area with no/low curvature, e.g. [‘top of mattress’] for a model of a bed with an installed
      mattress.
    - ‘texture quality’: int, range 0 (no texture, making the object in the model hard to identify) to 9 (detailed
      and realistic texture for the object type, making the model appear close to real).
An example response to a query with renders of a model depicting a bookshelf with about 10 books in red, blue,
and white placed on a green tiled floor where the render from the back shows the same spines seen from the front
could be:

<EXAMPLE JSON OMITTED>

Feel free to briefly reason before providing you response as a JSON parseable dict, which must include
clear and concise values for all the required entries without requesting additional input.
```

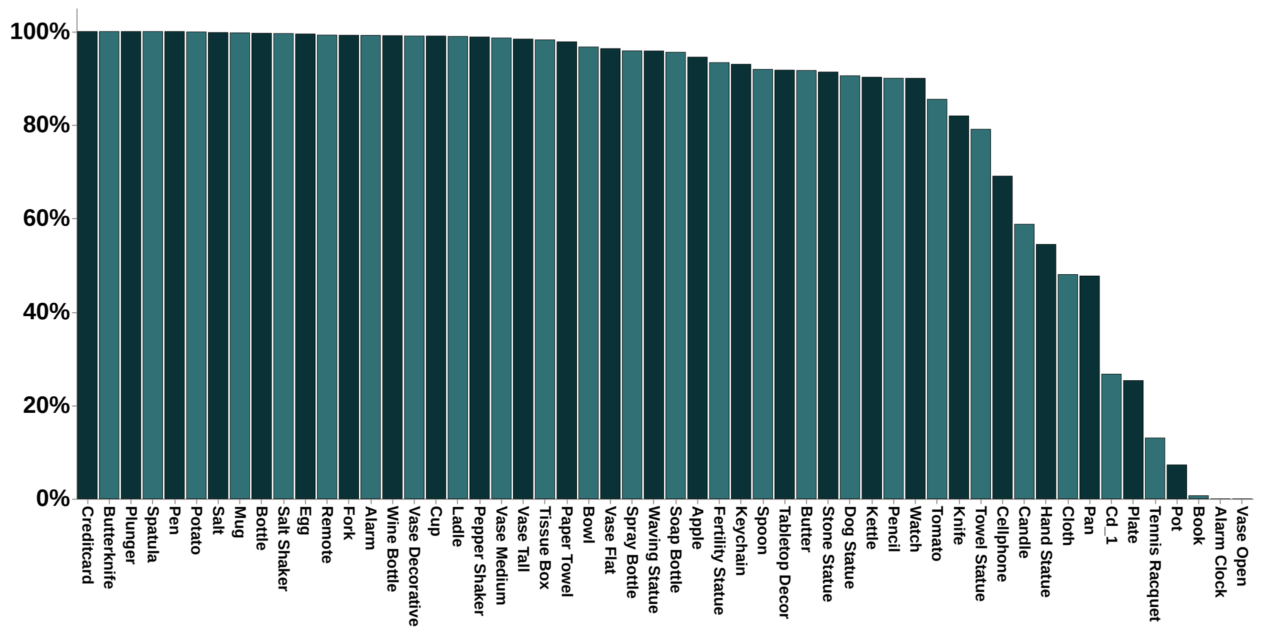
Figure 21 System prompt used with GPT-4.1 to generate alternative annotation for Objaverse object assets filtering in batch mode (continues from Fig.20).

```
We need to generate placement options for synsets used to label objects. For each of these synsets, we have
associated specific types, text descriptions of some objects, and scale ranges.

Write the placement options in JSON format, and do not add any additional comments. The structure is a
mapping from each synset to a nested mapping with Boolean entries, indicating whether each property seems likely
for objects of the given synset:
{
      ‘hasMultipleObjects’: refers to a composition or set of several objects,
      ‘isScene’: refers to a scene or a set of objects (like an assortment) rather than a single object,
      ‘roomTypes’: {
           ‘inKitchens’: appears in a kitchen,
           ‘inLivingRooms’: appears in a living room, media room, or dining room,
           ‘inBedrooms’: appears in a bedroom, office, or playroom,
           ‘inBathrooms’: appears in a bathroom, restroom, or laundry room,
           ‘inOthers’: appears in other room types like garages, balconies, etc.
      },
      ‘feasibleLocations’: {
      ‘onFloorInCorner’: placed directly on the floor, in a corner,
      ‘onFloorInMiddle’: placed directly on the floor, anywhere away from walls,
      ‘onFloorOnEdge’: placed directly on the floor and in contact with a wall,
      ‘onWall’: placed on a wall,
      ‘fromCeiling’: hangs from the ceiling
      },
      ‘isPickupable’: allows being picked up with a single hand,
      ‘isKinematic’: has an effective fixed pose, as in Unity’s kinematic bodies,
      ‘multiplePerRoom’: appears multiple times in the same room
}

Please set at least one of the ‘roomTypes’ (try to use ‘ìnOthers’ sparingly) unless the synset seems to
refer to a scene rather than a single object as signaled in ‘ìsScene’. Also note that enabling any of the
‘feasibleLocations’ options (floor, wall, or ceiling) prevents the objects os the synset from being placed on
top of other structures or furniture.

The synsets (along with associated specific types, sample descriptions, and scale ranges in cm) to
annotate are:

<OMITTED BATCH OF INPUT DATA>
```

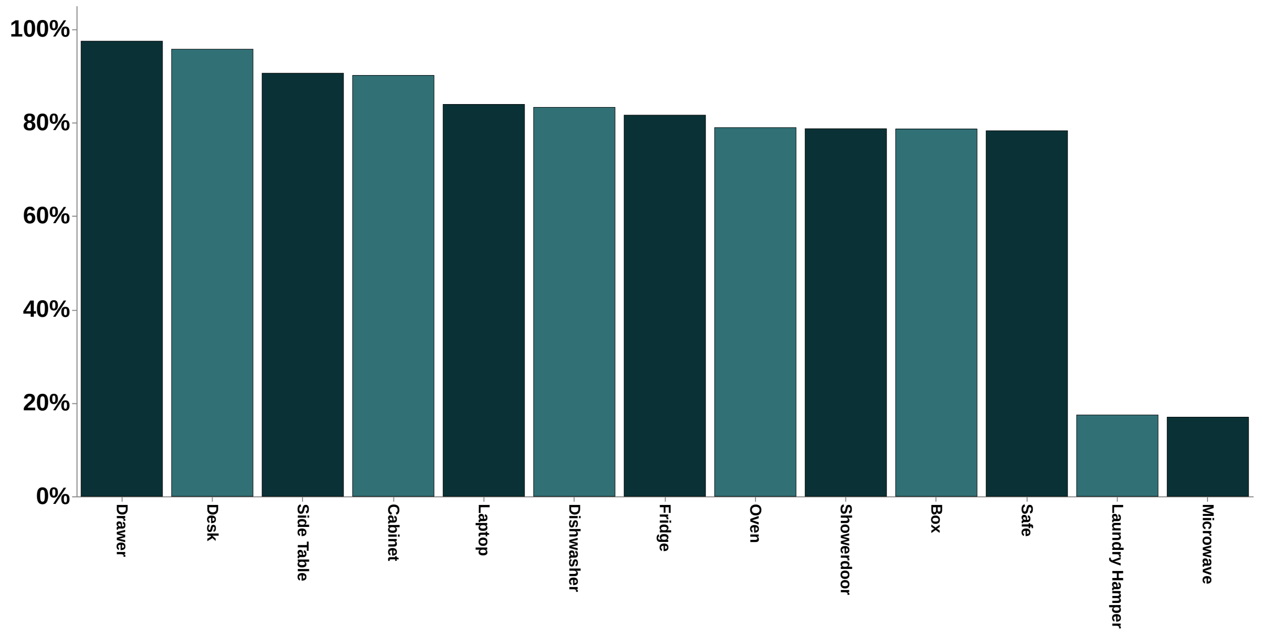
Figure 22 Prompt used with GPT-4o to determine placement options for a given batch of synsets.

```
SYSTEM PROMPT:

You are an expert in sociology, psychology, and interior design counseling.  Given a list of one-line
personal identity statements, your task is to identify which individuals are likely to find certain indoor
scenes visually memorable due to the presence of large, interest-relevant objects. These objects should stand
out even in visually cluttered environments.

Typically, only about 25% of the identities will be sufficiently specific or interest-oriented to warrant
this kind of visual sensitivity.  Avoid selecting identities that are too abstract, ethnicity-related,
enterprise-centric, or related solely to virtual environments, as these are unlikely to correspond to specific,
visually memorable physical elements.

Return your output as a JSON-parseable dict mapping indices corresponding to identity statements that
are useful for informing visually impactful interior design styles to a string describing visual objects,
decoration items, pieces of furniture, etc., that would instantly draw the attention of each chosen personality
in some indoor scene.
```

Figure 23 System prompt used with GPT-4.1-mini to select valid persona descriptions in batch mode.



(a) Average grasp success rates by object category in a random sample of MolmoSpaces houses.



(b) Average grasp success rates by category for articulated objects in a random sample of MolmoSpaces houses.

Figure 24 Average grasp success rates by category for all objects tested in a random sample of scenes from MSCrafted and MSProcObja MolmoSpaces-Scene dataset.