MESA: An Evaluation Framework for Compositional, Semantic, and Spatial Generalization in Robotics

Albert Wilcox[12] Frank Chang[1] Aishani Chakraborty[1] Nhi Nguyen[1]
Jeremy Collins[1] Vaibhav Saxena[1] Siddharth Karamcheti[1] Animesh Garg[1]
[1]Georgia Institute of Technology [2]Georgia Tech Research Institute

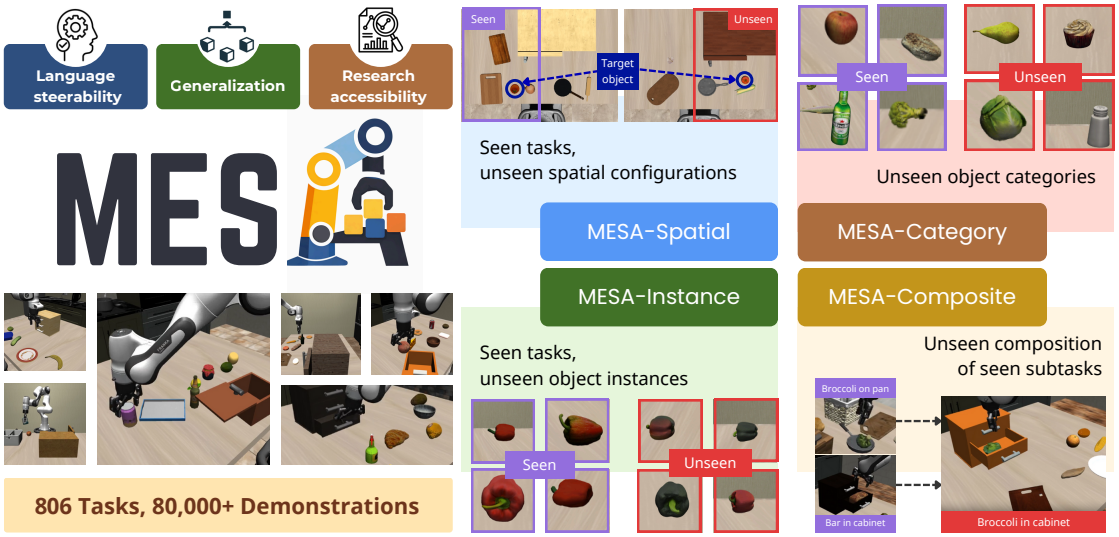


Figure 1: MESA (**Left**) is a dynamic evaluation framework designed to enable researchers to probe language steering and generalization capabilities in settings accessible to researchers with academic budgets. We introduce MESA-Bench (**Right**), which includes test sets designed to probe generalization to unseen spatial configurations, object instances, object categories and subtask compositions.

***Abstract*—Despite growing interest in foundation models for robotic manipulation, systematic evaluation of their capabilities remains elusive. Existing benchmarks are either narrow and saturated, offering little signal on SOTA methods, or broadly scoped, providing diversity at the cost of controlled experimentation. We introduce MESA, a dynamic evaluation framework designed for precise, reproducible measurement of language-conditioned policy generalization. MESA provides five evaluation suites that isolate in-distribution performance and four distinct axes of generalization: spatial configuration, object instance, object category, and subtask composition, enabling researchers to diagnose where and why policies fail, not just whether they fail. We further introduce a language-following metric that disentangles task understanding from low-level execution, revealing that state-of-the-art policies often understand commands but fail to execute them. To support rapid iteration, we develop MESA-Gen, a pipeline for scalable task and demonstration generation built by improving MimicGen. We evaluate seven baseline methods spanning diffusion policies, VLM-based approaches, and pretrained VLAs, finding that even the strongest model achieves only 52% average success, providing meaningful signal while leaving ample room for progress. The results are correlated with rankings in RoboArena evaluations ($\rho = 0.9$), validating MESA as a reproducible testbed predictive of relative real-world performance. Data, code and model weights are available at https://pairlab.github.io/MESA.**

I. Introduction

Benchmarks shape the trajectory of research. They define what the community measures, and hence what it optimizes for. In robotic manipulation, recent progress in robotic foundation models (RFMs) has produced large, language-conditioned policies capable of following open-ended natural language instructions, completing long-horizon tasks with strong robustness, and executing dexterous manipulation with impressive precision [1–12]. Despite rapid progress, the capabilities, limitations, and failure modes of these models remain poorly understood.

Current evaluation practices make systematic understanding of RFMs difficult. π_0 [3] combines a PaliGemma backbone with a flow-matching action expert, pretrains on cross-embodiment data from eight robot configurations, and evaluates on proprietary real-world setups. OpenVLA [2] fine-tunes a Llama-2 backbone on Open X-Embodiment [26], and evaluates on a proprietary Google robot as well as custom Widow-X and Franka tasks. GR00T-N1 [6] pairs an Eagle VLM with a diffusion transformer, trains on a mixture of human videos, synthetic data, and real demonstrations, and evaluates on RoboCasa and DexMimicGen in simulation and a real GR-1 humanoid. Each effort varies in the choice of architecture,

Benchmark	# Tasks	# Obj / # Instances	Multitask	Action Space	Data Gen	Metrics	OOD Evaluation
RLBench [13]	100	28 / 28	✓	Keyframe	Scripted	SR	–
COLOSSEUM [14]	20	28 / 28	✗	Keyframe	Scripted	SR	Inst.
CALVIN [15]	34	5 / 30	✓	Low-level	Human	SR, SCR	–
MetaWorld [16]	50	38 / 38	✗	Low-level	Scripted	SR, Rew.	–
LIBERO [17]	130	51 / 75	✓	Low-level	Human	SR	–
Simpler [18]	8	17 / 17	✓	Low-level	Human	SR	–
Behavior-1k [19]	1000	2211 / 9331	✓	Both	Scripted	SR	–
VLABench [20]	100	163 / 2164	✓	Keyframe	Scripted	SR	Obj., Comp.
Robomimic [21]	8	12 / 12	✗	Low-level	Human	SR	–
Robocasa [22]	100	153 / 2509	✓	Low-level	MG 1→1	SR	–
Robocasa 365 [23]	365	153 / 2509	✓	Low-level	MG 1→1	SR	Comp.
RoboTwin 2.0 [24]	50	147 / 731	✗	Low-level	Scripted	SR	–
MolmoSpaces [25]	8	2.8k / 130k	✓	Both	Scripted	SR	–
MESA	806	153 / 2509	✓	Low-level	MG few → many	SR, LF, SCR	Inst., Obj., Comp., Spat.

Table I: Comparison of VLA evaluation suites. For each benchmark, we compare scale (number of tasks, object categories, and unique asset instances), support for multitask language instructions, action space, data collection method (scripted, human-collected, none, or MimicGen (MG) variants), and evaluation signal metrics (success rate (SR), reward (Rew.), subtask completion rate (SCR), and language following (LF)). We also indicate whether the benchmark probes generalization to OOD object instances (Inst.), OOD object categories (Obj.), OOD task compositions (Comp.), and OOD spatial configurations (Spat.).

backbone, training data, and evaluation protocol. Given the prohibitive cost of training, this combinatorial explosion of training and evaluation settings makes it nearly impossible to isolate which design choices actually matter for generalization, sample efficiency, or language grounding.

Simulation offers a path toward reproducible evaluation, but existing benchmarks occupy two unsatisfying extremes. Some are narrow and saturated: LIBERO [17] enabled systematic study of transfer axes, but state-of-the-art RFMs now exceed 98% success rate [27]. Other simulation benchmarks are diverse, but imprecise: RoboCasa [22] and Behavior-1K [19] provide myriad assets, but offer no clear methodology for experimentation with controlled distribution shifts. We elaborate on this Goldilocks problem in Section § II.

This work aims to provide a simulated evaluation framework that fills these gaps, satisfying the following desiderata:

1) **Controlled Generalization**: Evaluation sets should isolate specific axes of generalization, enabling diagnosis of *where* and *why* policies fail.
2) **Language Grounding**: Evaluation should require policies to follow language instructions, not memorize task-specific behaviors or deduce tasks from visual observation alone.
3) **Extensibility**: Researchers should be able to quickly generate new tasks and evaluation suites. As model capabilities evolve, evaluation must evolve with them. A static benchmark inevitably saturates or becomes irrelevant.

To meet these objectives, we introduce MESA, an evaluation framework for language-conditioned robotic manipulation. MESA is not a fixed benchmark, but a *live* evaluation ecosystem (Fig. 2). We envision two primary modes of use: (1) **Evaluate existing models with MESA-Bench.** We provide five evaluation suites isolating language steering and four distinct generalization axes: spatial configuration, object instance, object category, and subtask composition. As we show in § V, current RFMs exhibit meaningful differences across these axes, with substantial headroom remaining. (2) **Generate new evaluations with MESA-Gen.** Researchers can curate targeted evaluations for emerging questions and synthesize the corresponding data to power their experiments.

Moreover, we also introduce a *language-following metric* to disentangle instruction comprehension from execution. Existing benchmarks conflate these factors within the success rate. Our metric reveals that RFMs often understand instructions, yet fail to execute them, laying the groundwork for future work. Our contributions are as follows:

1. MESA-Gen, a framework for scalable task and demonstration generation. We introduce improvements to MimicGen [28] that enable higher-quality data generation under greater spatial variation and support generation for novel tasks (§ III-C).

2. MESA-Bench, a benchmark comprising a canonical training set (MESA-70) and five evaluation suites targeting: in-distribution performance, spatial generalization, instance generalization, category generalization, and compositional generalization. We introduce a language-following metric to disentangle instruction comprehension from control execution.

3. Rigorous experiments evaluating seven learning methods spanning diffusion policies, VLM-based approaches, and pre-trained VLAs. Our strongest baseline policy achieves only 52% average success, dropping to 26% on novel object categories. Results correlate with real-world rankings, validating MESA as a useful testbed with room for methodical improvements.

II. Related Work

A. Simulation-Based Evaluation of Multitask Policies

Simulation offers reproducible, scalable evaluation without the cost and variability of real-world rollouts. However, existing simulation benchmarks for RFMs occupy two unsatisfying extremes, creating a Goldilocks problem: benchmarks are either narrow enough for controlled experimentation but quickly saturate, or diverse enough to resist saturation but too uncontrolled for systematic study. A detailed taxonomy of existing simulated benchmarks is provided in Table I.

Saturated Benchmarks. MetaWorld [16] provides 50 tasks without language instructions, requiring policies to infer tasks from only visual input. RLBench [13] offers 100 tasks but uses keyframe-level actions ill-suited for behavior cloning.

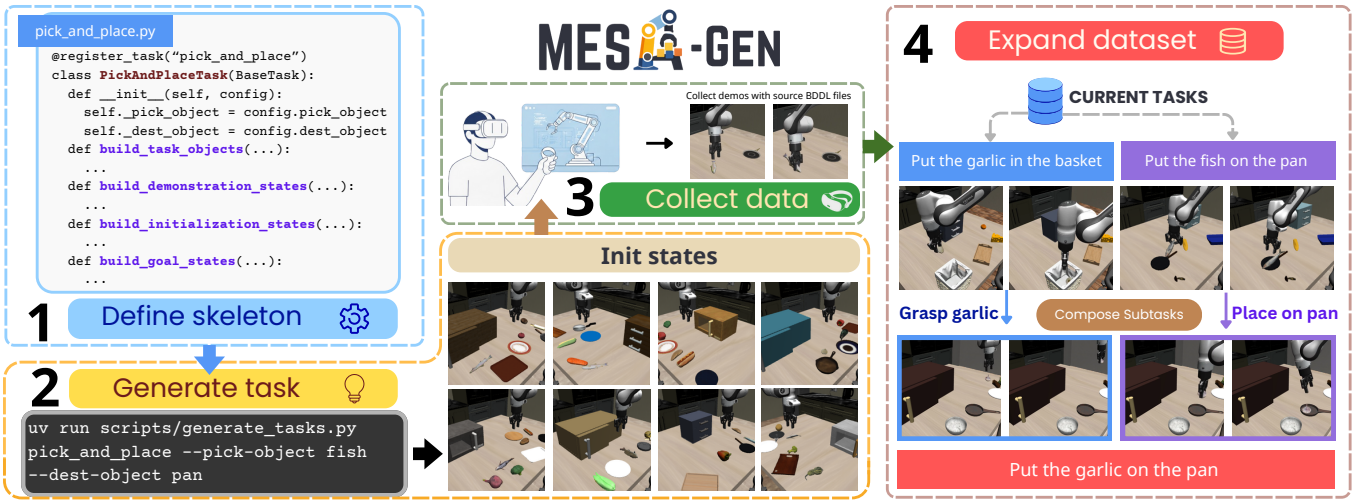


Figure 2: MESA-Gen experiment building framework. (1) If a suitable task skeleton does not already exist, define it in Python. (2) Instantiate a task from the skeleton and generate BDDL files for the training and evaluation sets. (3) Optionally, collect a small number of source demonstrations. (4) Generate a large dataset for imitation learning. Using subtask stitching enables data generation for tasks without human data.

CALVIN [15] pioneered long-horizon language-conditioned evaluation, but contains limited object diversity. LIBERO [17] enables systematic decomposition of transfer axes across 130 tasks. However, it has been saturated, with recent RFMs exceeding 98% success rate [27]. This prompted follow-ups like LIBERO-Plus [29], which targets perceptual robustness within the same task distribution, and LIBERO-PRO [30]. Since these extensions train on the original LIBERO data, geared towards overfitting, they have very low success rates for most SOTA policies, offering limited signal.

Imprecise Benchmarks. At the other extreme are benchmarks that provide diversity at the cost of tractable experimentation. RoboCasa [22] and RoboCasa-365 [23] scale to hundreds of tasks across kitchen scenes, but this diversity demands massive datasets and yields low success rates on composite or unseen tasks. Behavior-1K [19] defines 1,000 activities, but provides demonstrations for only 50 tasks. VLABench [20] prioritizes heavy domain randomization across many task categories, objects, and scene variations, but does not isolate controlled OOD axes of generalization, yielding single-digit success rates for strong baseline VLAs. MolmoSpaces [25] offers impressive scale, but still lacks controlled OOD settings. These benchmarks offer scale, but without controlled experimental design, it is difficult to draw scientific conclusions about model capabilities.

Evaluating Targeted Generalization. A third category of benchmarks attempts to probe specific generalization axes, but each faces limitations. COLOSSEUM [14] provides systematic perturbation analysis across 14 axes (color, lighting, texture, camera pose), but focuses on visual robustness rather than semantic or compositional generalization, and its keyframe action space is incompatible with modern behavior cloning architectures. The STAR-Gen taxonomy [31] formalizes generalization axes (visual, semantic, behavioral) and demonstrates that RFMs struggle with semantic generalization despite internet-scale pretraining, but requires real-world evaluation that is costly and difficult to reproduce. Real-to-Sim approaches like SIMPLER [18] and PolaRiS [32] achieve strong correlation between simulated and real performance, but cover limited task diversity due to the complexity of digitizing each real scenario. MESA addresses this gap by combining the rigor of controlled evaluation with scalable simulation, while providing infrastructure that evolves as model capabilities grow.

B. Data Generation for Robot Learning

Data generation is essential for scalable training and evaluation. MimicGen [28] scales small human demonstration sets to large datasets through object-centric trajectory transformation, generating 50K+ demonstrations from ∼200 human demos. Extensions include DexMimicGen [33] for bimanual manipulation and SkillMimicGen [34] for improved scene variation handling via skill segmentation. Procedural approaches like RoboGen [35] and GenSim [36] use LLMs to propose tasks and populate scenes, enabling increased task diversity.

A key finding from recent work on data scaling laws in robotics [37] is that generalization follows a power-law relationship with environment diversity, and diversity matters far more than quantity; once demonstrations per environment reach a threshold, additional data provides minimal benefit. This directly informs MESA's design. Rather than maximizing raw data scale, we prioritize systematic coverage of generalization axes with efficient data generation, enabling rigorous evaluation with academic compute budgets. MESA-Gen builds upon MimicGen with improvements for spatial variation and novel tasks.

C. Dynamic Benchmarking

Static benchmarks face an inherent tension: narrow enough to enable controlled comparison, they quickly saturate; or broad enough to resist saturation, they sacrifice experimental control. This problem is not unique to robotics. In language modeling, HELM [38] introduced a holistic evaluation emphasizing broad coverage, multi-metric measurement, and standardization, posi-

tioning itself as a dynamic benchmark, updating continuously as models evolve. MESA adopts this philosophy for robotic manipulation. Rather than a fixed task suite, MESA provides infrastructure (MESA-Gen) for researchers to generate new tasks and evaluation splits as model capabilities advance.

III. MESA: A Dynamic Benchmarking Framework

In this section, we describe the open-source MESA-Gen framework, which is designed to allow robotics practitioners to scalably generate large sets of new tasks and corresponding data, as seen in Fig. 2. We use this framework to instantiate MESA-Bench (§ IV), an initial dataset and set of five evaluation suites for probing generalization of learned policies.

A. Background

MimicGen [28] is a framework for generating large robot demonstration datasets from a small set of human-collected seed trajectories. The core idea is to decompose a demonstration trajectory $\tau = \{(T_o(t), T_e(t))\}_{t=1}^{N}$ into object-centric segments where $T_o(t) \in SE(3)$ and $T_e(t) \in SE(3)$ denote the object and end-effector poses, respectively. For each segment, MimicGen computes the relative transform $T_{\text{rel}}(t) = T_o(t)^{-1}\, T_e(t)$, which decouples motion from the original global frame. Given a new target object pose $\tilde{T}_o(t)$, the corresponding adapted end-effector pose is reconstructed as $\tilde{T}_e(t) = \tilde{T}_o(t)\, T_{\text{rel}}(t)$. By stitching these transformed segments together in their original temporal order, MimicGen generates new demonstrations that are physically consistent in novel contexts.

The *BEHAVIOR Domain Definition Language (BDDL)* [39, 40] allows users to specify task initial and goal states using logical predicates, and has been used for several high-profile benchmarks [17, 19]. More recently, Saxena et al. [41] adapts it to MimicGen, specifying the object-centric subtasks using demonstration predicates and enabling MimicGen-ready environment generation purely in the BDDL language.

B. Generating Diverse Manipulation Tasks

The first step in MESA-Gen is to procedurally generate tasks and task variants for training and evaluation. This involves defining a task skeleton and using that skeleton to generate BDDL files, as shown in Fig. 2.

The task skeleton is implemented as a simple Python class and describes task-relevant objects as well as initialization, demonstration and goal predicates. For example, a pick and place task skeleton would define the target and destination object, lay out demonstration predicates consisting of grasp and placement motions, and specify a goal state with the target object being on top of the destination. MESA handles many important task generation functions such as distractor object sampling (discussed below) and sampling object instances. We include a sample pick-and-place skeleton in § C.4.

After defining the task skeleton, users simply need run our BDDL generation script with their desired object categories as inputs. For example, as illustrated in Fig. 2, to generate a task "`Put the fish in the frying pan`", users would run the script while specifying the `pick_and_place` task skeleton and `fish` and `pan` for the target and destination objects, respectively. MESA will then generate groups of BDDLs for the training and evaluation sets, each with a unique set of distractor objects, as well as a set of BDDLs without distractor objects for MimicGen source data collection. Although we did not use this capability when instantiating MESA-Bench, it is also possible to vary scene layout, textures, camera pose, and robot embodiment using this framework.

Ensuring Visual Confusion: To benchmark language steerability, we must prevent the policy from inferring the task solely from visual observations. If a scene contains only one interactable object, the policy may ignore the language instruction and simply act on that object, a form of *causal confusion* [42]. For example, if the policy only ever sees an `apple` when tasked with "`put the apple in the basket`", it can ignore the language instruction entirely and succeed by visually searching for apples. To ensure high observation-conditioned task distribution entropy, we sample distractors such that every scene contains multiple object-destination pairs present in our dataset. This forces the policy to condition on the language instruction to resolve ambiguity. We provide further distractor sampling details in § C.3.

C. Scalable Data Generation with MESA-Gen

Next, we introduce MESA-Gen's pipeline for scalable data generation with substantial variety in tasks, objects and initial configurations. We discuss our method for enabling data generation with unseen tasks in § III-C1 and our method for improving data generated under large source-target object pose difference in § III-C2.

1) Subtask Stitching with MESA-Gen

While MimicGen is built to enable scalable data generation, it still requires human data collection for each task. This requires linear human effort in the number of created tasks, which is cumbersome for researchers with limited resources. To alleviate this, we use subtask stitching which, instead of using only demonstrations from the target task, uses MESA's predicate structure to enable data collection for unseen tasks by composing data from other tasks with similar predicates. For example, for a target task "`pick up the apple and put it on the plate`", MESA-Gen might combine the `Grasp(apple)` subtask from "`put the apple in the drawer`" with the `On(banana, plate)` subtask of "`put the banana on the plate`". We provide further detail in § D.1.

2) Starting point optimization

As described in § III-A, MimicGen works by applying an affine transformation based on the target object pose in the source and current task. This is effective when source and target object poses are relatively close to one another and the target object is quasistatically stable, but can lead to unpredictable behaviors when either of these assumptions is broken. Thus, in this section, we outline changes to the MimicGen pipeline leading to behaviors that quantitatively lead to higher data generation rates and qualitatively lead to more sensible behaviors.

Removing Roll and Pitch: MimicGen can produce unstable trajectories for objects without stable resting poses. We observe

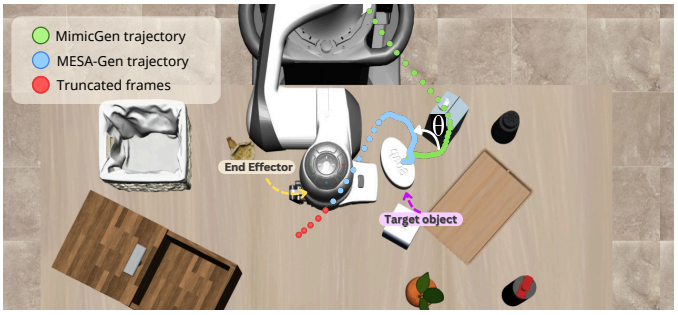


Figure 3: Due to the large change in object pose, the original MimicGen trajectory (●) starts from a position near the robot base, leading to a large interpolation and self-collision. We optimize the trajectory (●) by rotating it about the vertical axis through the object position and truncating unnecessary frames (●) to minimize the Euclidean distance from the end-effector to a point along the trajectory.

that most tabletop objects either have effectively fixed roll and pitch under our initialization distribution or admit roll- and pitch-invariant grasps, so we ignore roll and pitch when computing MimicGen transformations.

Trajectory Start Optimization: Another issue with MimicGen arises when there is a large change between the source and target object poses, sometimes leading to unnatural behaviors. We show an example of this in Fig. 3, where the source trajectory comes from a configuration where the soap was further away from the robot, leading to a trajectory beginning with a large unnecessary interpolation and self-collision. In the best case this is unnatural and suboptimal; in the worst case it leads to dangerous behaviors and kinematic singularities.

To address this, we optimize the trajectory by rotating it about the vertical axis going through the target object to minimize the L_2 distance between the current end-effector pose and the start of the trajectory, and then truncate any additional frames so that the starting frame is the closest to the end-effector. We rigorously describe this process as well as the pipeline for optimizing end-effector orientations in § D.3.

D. Evaluation Metrics

In addition to success rate, MESA-Gen uses the BDDL predicate system to provide a richer set of metrics to probe different properties of policy behavior:

- **Distractor Task Completion Rate**: As described in § III-B, we populate every scene with objects corresponding to a variety of distractor tasks. Next, we enumerate all pairs of graspable and destination distractor objects which comprise our set of *distractor tasks*. For example, for a target task "`put the orange in the basket`" with a bowl and apple for distractor objects, the distractor task predicates would be `In(apple, basket)`, `In(apple, bowl)`, `In(orange, bowl)` and `In(bowl, basket)`. After a failed rollout, if any of these predicates were satisfied, we log a distractor task completion. Intuitively, a high distractor task completion rate indicates a policy that is outputting reasonable actions without following language correctly.
- **Language Following Rate** is reported in several high-profile papers [4, 6, 8, 43], but is generally defined imprecisely. In order to compute this reliably, we define language following rate L to be, for a policy with a success rate S and distractor task completion rate D, $L = \frac{S}{S+D}$. This is the probability, given that the policy has completed some feasible task, that it completed the correct one.

IV. MESA-Bench

In this section, we describe the MESA-Bench benchmark, which is comprised of five evaluation suites described in § IV-A and a shared training set described in § IV-B. Additionally, we provide several quantitative metrics, as described in § III-D.

We use the pipeline from § III to generate hundreds of tasks for MESA-Bench. To do so, we define a handful of task skeletons such as `PickAndPlace` and `MultiStepArticulatedManipulation`. Then, we use GPT-5.1 to label each object with container compatibility. For example, it is reasonable to place an apple but not a bottle of wine in a bowl. Next, we generate BDDL files for all possible tasks according to the compatibility labels.

The next step is to build a dataset. To start, we collect data on the source datasets, ensuring to cover a wide range of subtasks. Then, we run MESA-Gen on the full set of tasks, noting subtasks where it is particularly weak and collecting new source data to fill in the gaps. After iterating on this process, we collect a total of 1125 human demonstrations across 80 tasks. For all tasks with a sufficiently high data generation rate (we choose a 10% cutoff), we run MESA-Gen, yielding 100 demonstrations each for 806 tasks which we use to construct the dataset for MESA-Bench. A potential concern is that generated data may be biased towards regions with high data generation success rates, and we study this in § B.4.

A. Evaluation Suites

The benchmark comprises five evaluation sets, designed to probe in-distribution performance as well as performance under perturbations along four axes of generalization, as described below. For all trials, we randomly sample an unseen set of distractor objects and poses. We generate a fixed set of initial states to guarantee unbiased comparisons between policies. The full list of tasks is included in § B.6.

MESA-70 tests policy performance for 70 in-distribution tasks. To succeed in this task set, policies must reliably follow language instructions despite distractor tasks while precisely manipulating the correct target objects.

MESA-Spatial is designed to evaluate policies under change in object location, and consists of 10 tasks that each interact with one of three objects. We partition the workspace into two disjoint regions $\mathcal{R}_{\text{ID}}$ and $\mathcal{R}_{\text{OOD}}$ and only initialize these objects in $\mathcal{R}_{\text{ID}}$ during training and $\mathcal{R}_{\text{OOD}}$ while evaluating. Note that for each task in MESA-Spatial, we include the variant where the target object spawns in $\mathcal{R}_{\text{ID}}$ in MESA-70, allowing direct comparison under the change in object arrangement.

MESA-Instance is designed to test generalization to unseen object instances, and comprises 20 tasks. For each task, we partition the set of object assets into disjoint in-distribution and out-of-distribution subsets, training exclusively on the in-distribution assets and evaluating on held-out instances at test time. As with MESA-Spatial, the in-distribution asset variants

Setting	DP	PG-Bin	PG-FM	GR00T-N1.6	π_0	FAST	$\pi_{0.5}$
MESA-70	1.6	0.9	30.7	25.0	43.1	46.5	**62.8**
MESA-Spatial	1.0	3.0	35.0	23.2	42.2	53.6	**68.6**
MESA-Instance	1.9	0.6	22.7	22.7	34.0	39.1	**55.4**
MESA-Composite	0.3	0.2	17.7	15.0	27.5	28.9	**46.7**
MESA-Category	0.7	0.0	7.7	14.1	11.6	8.4	**26.4**
Average	1.1	0.9	22.8	20.0	31.7	35.3	**52.0**

Table II: Success rates (%) for all baseline methods across all task suites. $\pi_{0.5}$, which is trained with the largest robotics pretraining dataset and uses knowledge insulation to preserve its VLM backbone, is consistently a strong baseline. π_0 and FAST also perform well, followed by PG-FM and GR00T-N1.6. DP and PG-Bin, which have naive conditioning and action representations, respectively, perform poorly.

are included in MESA-70, enabling direct comparison under changes in object appearance and geometry.

MESA-Composite consists of 20 unseen composite tasks comprised of seen subtasks. Twelve of these have a familiar task structure to tasks in MESA-70 and eight are longer-horizon tasks with unseen task structures.

MESA-Category consists of 20 new tasks involving objects never seen during training. To succeed in this set, policies must make use of broad vision-language understanding capabilities from internet-scale pretraining.

B. Training Set

In order to minimize iteration time, we have one training set, the MESA-70 dataset, shared between the evaluation settings described in § IV-A. It is comprised of 100 demonstrations per task for each of the 70 tasks in MESA-70, which we generate using our MimicGen pipeline discussed in § III-C. Optionally, we have a set of 736 auxiliary tasks for which we generated 100 demonstrations each to be used for studying policy performance after scaling data diversity, as we do in § V-D. For some of our experiments we co-train on a subset of this dataset containing 10 demonstrations from each task, which we refer to as MESA-Aux.

V. Experiments

We organize our experiments around the following questions:

1) How do state-of-the-art RFMs and baseline policies perform under controlled distribution shifts?
2) To what extent is poor language following responsible for policy failures?
3) How does naïvely scaling data quantity and diversity influence policy performance?
4) Are the results from MESA well-correlated with results from real-world policy evaluations?

To answer these questions, we use MESA-Bench to evaluate several tabula rasa and pretrained baseline methods:

1) **Multitask Diffusion Policy** [44] using the DiT Block Policy from Dasari et al. [45] with 80M parameters. This policy uses AdaLN [46] to condition on the average pooled outputs of a transformer-based observation encoder.
2) **PaliGemma-Binning** (PG-Bin) which finetunes a PaliGemma2-3B [47] backbone to predict actions discretized using RT-2-style [1] binning.
3) **PaliGemma-Flow Matching** (PG-FM), which finetunes a PaliGemma2-3B backbone with a flow matching head. In other words, π_0 [3] without robot pretraining.
4) **GR00T-N1.6** [6], which attaches a flow matching action head to a Cosmos-2B VLM [48].
5) $\boldsymbol{\pi_0}$ [3], which attaches a flow matching head to a PaliGemma2-3B backbone and pretrains with a large cross-embodiment robot dataset.
6) $\boldsymbol{\pi_0}$**-FAST** [4], which replaces π_0's flow matching head with discretized action chunk prediction.
7) $\boldsymbol{\pi_{0.5}}$ [8], which trains a π_0-like architecture with knowledge insulation [43].

We finetune each baseline on the MESA-70 dataset, using 50,000 gradient steps for VLM or VLA policies and 320,000 gradient steps for diffusion policy. We use relative joint angle action space, which is featured extensively in pretraining data for all of our baselines. Additional details are available in Appendix F.

A. Baseline Results

We present the performance of our suite of baseline methods on the evaluation sets in Table II. A clear hierarchy emerges: $\pi_{0.5}$ achieves the strongest average performance (52.0%), followed by π_0-FAST (35.3%), π_0 (31.7%), PG-FM (22.8%), and GR00T-N1.6 (20.0%). DP and PG-Bin achieve near-zero success rates across all settings.

Robot pretraining is a primary differentiator. The comparison between π_0 and PG-FM is particularly informative: these two methods share the same PaliGemma2-3B backbone and flow matching action head, differing only in that π_0 is pretrained on a large cross-embodiment robot dataset. This pretraining accounts for a consistent 10–12 percentage point advantage across all evaluation suites. The additional gap from π_0 to $\pi_{0.5}$ (a further 15–26 percentage points) demonstrates the importance of knowledge insulation, which preserves the backbone's VL-grounding capabilities during action fine-tuning.

Action representation matters. PG-Bin, which uses RT-2-style action binning, fails almost entirely (0.9%) despite sharing its VLM backbone with PG-FM (22.8%). This indicates that directly discretizing continuous action dimensions into language tokens is a poor fit for the fine-grained control required by MESA tasks, at least without the large-scale pretraining that made this strategy viable for RT-2. Similarly, DP's near-zero performance reflects poor multi-task scaling properties, which are analyzed further in § B.1.

MESA-Category represents an open frontier. Performance drops sharply for all models on novel object categories: $\pi_{0.5}$ falls from 62.8% to 26.4%, π_0-FAST from 46.5% to 8.4%, and π_0 from 43.1% to 11.6%. Notably, GR00T-N1.6 outperforms both π_0 and π_0-FAST in this setting despite weaker overall performance, possibly due to its frozen Cosmos backbone providing good priors for recognizing novel objects. These results suggest that open-vocabulary object generalization is not adequately addressed by current VLM pretraining and represents a key challenge for future work.

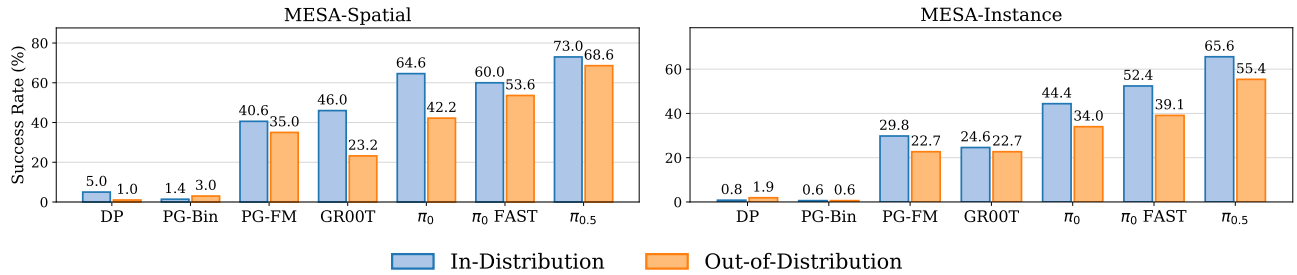


Figure 4: Comparing in- and out-of-distribution performance for the tasks in MESA-Spatial and MESA-Instance. We see that distribution shift consistently leads to a performance drop, indicating an opening for future work on generalizing more reliably to unseen spatial configurations and object instances.

Setting	DP	PG-Bin	PG-FM	GR00T-N1.6	π_0	FAST	$\pi_{0.5}$
MESA-70	18.9	28.2	80.0	71.4	79.7	83.7	**87.6**
MESA-Spatial	11.4	46.9	77.4	53.7	73.0	78.6	**84.5**
MESA-Instance	23.5	18.2	66.6	60.5	66.5	69.0	**76.9**
MESA-Composite	6.4	11.1	65.1	56.2	65.5	66.3	**81.5**
MESA-Category	12.1	0.0	32.6	42.5	33.1	28.0	**44.1**
Average	14.4	20.9	64.3	56.9	63.6	65.1	**74.9**

Table III: We present language following rate as defined in § III-D. We find that models with VLM backbones exhibit strong language following, while models trained from scratch exhibit much worse language following. Lower performance by all models in MESA-Category reflects that when presented with unfamiliar objects, our baselines are bottlenecked by their ability to ground language to novel visual referents rather than low-level execution.

MESA-Bench provides strong signal without saturation. The strongest model achieves 52.0% average success, indicating substantial headroom for improvement. At the same time, most baselines make meaningful progress across all settings. Together, these make MESA-Bench a high-signal benchmark for future methods.

Controlled Distribution Shift Analysis: Since MESA-Spatial and MESA-Instance each contain tasks with in-distribution counterparts in MESA-70, we can directly compare success rates with and without distribution shift (Fig. 4).

For spatial distribution shifts, performance degradation varies dramatically across our baselines. GR00T-N1.6 suffers the largest relative drop (46.0% $\rightarrow$ 23.2%), suggesting that it overfits to the spatial configurations seen during training. In contrast, $\pi_{0.5}$ is remarkably robust (73.0% $\rightarrow$ 68.6%), and π_0-FAST also degrades modestly (60.0% $\rightarrow$ 53.6%). The gap in degradation between π_0 and $\pi_{0.5}$ suggests that knowledge insulation may improve spatial generalization by preserving the base VLM's spatial reasoning capabilities.

Instance-level shifts produce more uniform degradation: π_0-FAST drops 25%, π_0 drops 23%, and PG-FM drops 24%, while $\pi_{0.5}$ again shows the greatest robustness with a 16% relative decrease. This pattern is consistent across all models with meaningful baseline performance, indicating that instance generalization is a broadly challenging axis rather than one where specific architectures have an inherent advantage.

Across both spatial and instance-level perturbations, results consistently show that distribution shift leads to performance degradation, validating that MESA successfully tests generalization rather than rote memorization.

B. Probing Language Following

Reliably following language instructions is a core objective for RFMs, but existing benchmarks offer no way to study language following beyond using success rate as a proxy. MESA reports a language following rate (§ III-D), defined as the probability that, given the policy completed *some* feasible task, it completed the one specified by the language instruction. A low language following rate indicates a policy that executes competent actions but frequently completes the wrong task. We present these results in Table III.

On in-distribution tasks (MESA-70), models with VLM backbones exhibit strong language following: PG-FM, π_0, and π_0-FAST all exceed 80%, and $\pi_{0.5}$ reaches 87.6%. When these models fail, they typically fail to complete any task rather than completing the wrong one. In contrast, DP achieves only 18.9% language following, meaning that when it does complete a task, it is the wrong one 81% of the time. This suggests that CLIP embeddings may provide weaker language grounding than a full VLM backbone.

The more revealing finding emerges under distribution shift. On MESA-Category, language following drops significantly for *all* models: π_0 falls from 79.7% to 33.1%, π_0-FAST from 83.7% to 28.0%, and $\pi_{0.5}$ from 87.6% to 44.1%. When confronted with novel object categories, models that were rarely confused in-distribution now complete the wrong task more often than the right one. This reveals that MESA-Category failures are not purely due to low-level execution when faced with unfamiliar objects; they reflect a breakdown in the model's ability to ground language to novel visual referents.

Our language following metric provides a diagnostic that success rate alone cannot; it identifies whether the bottleneck for a given model lies in language understanding, low-level execution, or both, and how this breakdown shifts across evaluation settings.

C. Scaling Data Quantity

We study the effect of increasing demonstration quantity for π_0 and $\pi_{0.5}$ (Fig. 5). For π_0, scaling data produces near-monotonic improvements across all settings, consistent with models benefiting from additional in-domain data to bridge the pretraining-deployment gap.

For $\pi_{0.5}$, gains are modest and, on MESA-Category, performance actually *decreases* with more data. We hypothesize

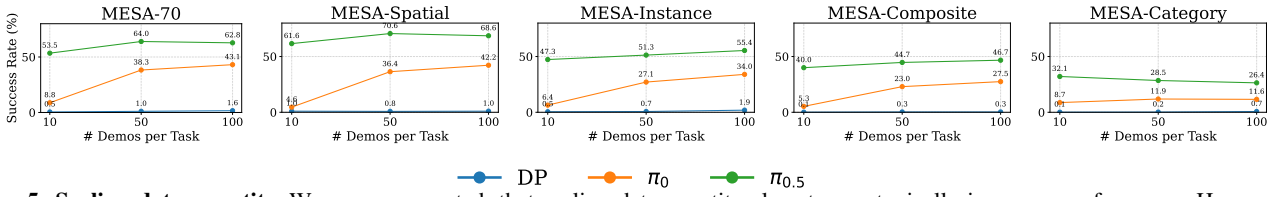


Figure 5: Scaling data quantity. We see, as expected, that scaling data quantity almost monotonically improves performance. However, we also see that pretraining consistently dominates, as $\pi_{0.5}$ with 10 demonstrations consistently outperforms π_0 with 100.

	70	Spatial	Instance	Composite	Category
DP	1.6	1.0	1.9	0.3	0.7
DP+Aux	2.3	2.4	2.2	0.6	0.3
π_0	43.1	42.2	34.0	27.5	11.6
π_0+Aux	43.4	54.4	33.2	28.8	8.7
$\pi_{0.5}$	**62.8**	68.6	**55.4**	**46.7**	**26.4**
$\pi_{0.5}$+Aux	59.0	**69.2**	53.6	45.6	23.0

Table IV: Scaling task diversity via auxiliary co-training yields negligible gains. We double the training set size by adding auxiliary demonstrations from MESA-Aux and train for the same number of steps. We find that the co-trained policies show little if any improvement, and hypothesize that naively scaling co-training task diversity does not automatically improve generalization.

that knowledge insulation means $\pi_{0.5}$ already encodes strong manipulation priors from pretraining; additional fine-tuning data may erode strong representations learned by the action expert from large-scale action data training that were valuable for out-of-distribution settings.

Another striking result was that $\pi_{0.5}$ *trained with 10 demonstrations per task consistently outperforms* π_0 *with 100.* Pretraining quality dominates data quantity, suggesting that investing in better pretraining or representation preservation yields greater returns than collecting larger fine-tuning datasets.

D. Scaling Data Diversity

We investigate whether task diversity through co-training on auxiliary tasks improves generalization. We double the training set by adding 10 demonstrations from 736 new tasks unseen in MESA-70, using the same embodiment and tabletop setup, and train for the same number of gradient steps.

We find that co-training with diverse auxiliary data yields negligible improvements in success rate, subtask completion, and language following across all evaluation suites. π_0 shows marginal gains on some settings but no consistent pattern, $\pi_{0.5}$ shows essentially no change, and Diffusion Policy shows modest gains but continues to have poor absolute performance.

This finding has important implications for benchmark and dataset design. Simply adding more diverse tasks does not automatically improve generalization along the axes MESA evaluates. We hypothesize that the auxiliary tasks, while diverse, do not specifically target the generalization axes being tested. This result supports MESA's design philosophy: controlled, targeted evaluation is more informative than broad diversity.

It also suggests that improving generalization in robotics foundation models will require algorithmic innovations, such as better representations, base models, and inductive biases, rather than naive data scaling.

	RoboArena	MESA
$\pi_{0.5}$	**1983**	**52.0**
π_0-FAST	1932	35.3
π_0	984	31.7
PG-FM	1797	22.8
PG-Bin	835	0.9

Table V: Comparing RoboArena [49] Elo and MESA success rate. With the exception of π_0 and PG-FM being swapped, we observe a consistent stack ranking of methods across both evaluations.

E. Predicting Real-World Performance

A simulation benchmark is only useful if its results predict real-world performance. Inspired by Jain et al. [32], in Table V we compare average success rates on MESA-Bench to Elo scores on RoboArena [49], a crowdsourced policy evaluation platform where higher Elo scores reflect aggregate pairwise blind preferences from human evaluators.

The ranking is largely consistent: $\pi_{0.5}$ and π_0-FAST occupy the top two positions in both evaluations, and PG-Bin ranks last in both. π_0 outperforms PG-FM on MESA (31.7% vs. 22.8%) but ranks lower on RoboArena (Elo 984 vs. 1797). This may be because DROID is much larger than MESA-70, so the PG-FM RoboArena model has seen much more real-robot data than the PG-FM MESA model, leading to stronger performance. Overall, results yield a Spearman rank correlation of $\rho = 0.9$ and Pearson $r = 0.71$, indicating that MESA rankings are predictive of relative real-world performance.

VI. Conclusion

We introduce MESA, a framework for evaluating language-conditioned robotic manipulation. MESA provides a living evaluation ecosystem that enables researchers to generate new tasks, demonstrations, and evaluations as model capabilities evolve. We develop MESA-Bench, comprising a canonical training set and five evaluation suites that isolate spatial, instance, category, and compositional generalization as well as in-distribution performance. Our experiments across seven methods reveal a clear hierarchy, with substantial headroom remaining. Our language following metric reveals that on novel object categories, even strong VLM-based policies complete distractor tasks more often than goal tasks, revealing vision-language grounding as a critical bottleneck. Results correlate with RoboArena rankings ($\rho = 0.9$), supporting MESA as a predictive testbed for real-world performance.

We also note the limitations of MESA. Task coverage is restricted to single-arm tabletop manipulation and does not reflect the full diversity of real-world interaction. MESA-Gen's automated initialization can occasionally yield physically implausible scenes, and its MimicGen data is ill-suited to deal with dense clutter. While our language following metric provides useful signal to decouple failure modes, it does not

capture all modes of failure, for example when the robot attempts the correct task unsuccessfully.

In the future, MESA can be extended to include additional embodiments, incorporate bimanual manipulation, or upgraded to a GPU-accelerated simulation backend to improve photo-realism or study online reinforcement learning. Despite these constraints, we view MESA as a principled foundation for rigorous, evolving evaluation. By releasing tools alongside tasks, we aim to seed a continually expanding benchmark that supports controlled, reproducible experimentation on language-conditioned robot policies.

ACKNOWLEDGMENTS

The authors would like to acknowledge the State of Georgia and the Agricultural Technology Research Program at Georgia Tech for supporting the work described in this paper.

REFERENCES

[1] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023.

[2] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

[3] 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, 2024. URL https://arxiv.org/abs/2410.24164.

[4] 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, 2025. URL https://arxiv.org/abs/2501.09747.

[5] Gemini Robotics Team, Saminda Abeyruwan, Joshua Ainslie, Jean-Baptiste Alayrac, Montserrat Gonzalez Arenas, Travis Armstrong, Ashwin Balakrishna, Robert Baruch, Maria Bauza, Michiel Blokzijl, Steven Bohez, Konstantinos Bousmalis, Anthony Brohan, Thomas Buschmann, Arunkumar Byravan, Serkan Cabi, Ken Caluwaerts, Federico Casarini, Oscar Chang, Jose Enrique Chen, Xi Chen, Hao-Tien Lewis Chiang, Krzysztof Choromanski, David D'Ambrosio, Sudeep Dasari, Todor Davchev, Coline Devin, Norman Di Palo, Tianli Ding, Adil Dostmohamed, Danny Driess, Yilun Du, Debidatta Dwibedi, Michael Elabd, Claudio Fantacci, Cody Fong, Erik Frey, Chuyuan Fu, Marissa Giustina, Keerthana Gopalakrishnan, Laura Graesser, Leonard Hasenclever, Nicolas Heess, Brandon Hernaez, Alexander Herzog, R. Alex Hofer, Jan Humplik, Atil Iscen, Mithun George Jacob, Deepali Jain, Ryan Julian, Dmitry Kalashnikov, M. Emre Karagozler, Stefani Karp, Chase Kew, Jerad Kirkland, Sean Kirmani, Yuheng Kuang, Thomas Lampe, Antoine Laurens, Isabel Leal, Alex X. Lee, Tsang-Wei Edward Lee, Jacky Liang, Yixin Lin, Sharath Maddineni, Anirudha Majumdar, Assaf Hurwitz Michaely, Robert Moreno, Michael Neunert, Francesco Nori, Carolina Parada, Emilio Parisotto, Peter Pastor, Acorn Pooley, Kanishka Rao, Krista Reymann, Dorsa Sadigh, Stefano Saliceti, Pannag Sanketi, Pierre Sermanet, Dhruv Shah, Mohit Sharma, Kathryn Shea, Charles Shu, Vikas Sindhwani, Sumeet Singh, Radu Soricut, Jost Tobias Springenberg, Rachel Sterneck, Razvan Surdulescu, Jie Tan, Jonathan Tompson, Vincent Vanhoucke, Jake Varley, Grace Vesom, Giulia Vezzani, Oriol Vinyals, Ayzaan Wahid, Stefan Welker, Paul Wohlhart, Fei Xia, Ted Xiao, Annie Xie, Jinyu Xie, Peng Xu, Sichun Xu, Ying Xu, Zhuo Xu, Yuxiang Yang, Rui Yao, Sergey Yaroshenko, Wenhao Yu, Wentao Yuan, Jingwei Zhang, Tingnan Zhang, Allan Zhou, and Yuxiang Zhou. Gemini robotics: Bringing ai into the physical world, 2025. URL https://arxiv.org/abs/2503.20020.

[6] NVIDIA, Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi "Jim" Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, Joel Jang, Zhenyu Jiang, Jan Kautz, Kaushil Kundalia, Lawrence Lao, Zhiqi Li, Zongyu Lin, Kevin Lin, Guilin Liu, Edith Llontop, Loic Magne, Ajay Mandlekar, Avnish Narayan, Soroush Nasiriany, Scott Reed, You Liang Tan, Guanzhi Wang, Zu Wang, Jing Wang, Qi Wang, Jiannan Xiang, Yuqi Xie, Yinzhen Xu, Zhenjia Xu, Seonghyeon Ye, Zhiding Yu, Ao Zhang, Hao Zhang, Yizhou Zhao, Ruijie Zheng, and Yuke Zhu. Gr00t n1: An open foundation model for generalist humanoid robots, 2025. URL https://arxiv.org/abs/2503.14734.

[7] Jason Lee, Jiafei Duan, Haoquan Fang, Yuquan Deng, Shuo Liu, Boyang Li, Bohan Fang, Jieyu Zhang, Yi Ru Wang, Sangho Lee, Winson Han, Wilbert Pumacay, Angelica Wu, Rose Hendrix, Karen Farley, Eli VanderBilt, Ali Farhadi, Dieter Fox, and Ranjay Krishna. Molmoact: Action reasoning models that can reason in space, 2025. URL https://arxiv.org/abs/2508.07917.

[8] 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.

[9] Gemini Robotics Team, Abbas Abdolmaleki, Saminda Abeyruwan, Joshua Ainslie, Jean-Baptiste Alayrac, Montserrat Gonzalez Arenas, Ashwin Balakrishna, Nathan Batchelor, Alex Bewley, Jeff Bingham, Michael Bloesch, Konstantinos Bousmalis, Philemon Brakel, Anthony Brohan, Thomas Buschmann, Arunkumar Byravan, Serkan Cabi, Ken Caluwaerts, Federico Casarini, Christine Chan, Oscar Chang, London Chappellet-Volpini, Jose Enrique Chen, Xi Chen, Hao-Tien Lewis Chiang, Krzysztof Choromanski, Adrian Collister, David B. D'Ambrosio, Sudeep Dasari, Todor Davchev, Meet Kirankumar Dave, Coline Devin, Norman Di Palo, Tianli Ding, Carl Doersch, Adil Dostmohamed, Yilun Du, Debidatta Dwibedi, Sathish Thoppay Egambaram, Michael Elabd, Tom Erez, Xiaolin Fang, Claudio Fantacci, Cody Fong, Erik Frey, Chuyuan Fu, Ruiqi Gao, Marissa Giustina, Keerthana Gopalakrishnan, Laura Graesser, Oliver Groth, Agrim Gupta, Roland Hafner, Steven Hansen, Leonard Hasenclever, Sam Haves, Nicolas Heess, Brandon Hernaez, Alex Hofer, Jasmine Hsu, Lu Huang, Sandy H. Huang, Atil Iscen, Mithun George Jacob, Deepali Jain, Sally Jesmonth, Abhishek Jindal, Ryan Julian, Dmitry Kalashnikov, M. Emre Karagozler, Stefani Karp, Matija Kecman, J. Chase Kew, Donnie Kim, Frank Kim, Junkyung Kim, Thomas Kipf, Sean Kirmani, Ksenia Konyushkova, Li Yang Ku, Yuheng Kuang, Thomas Lampe, Antoine Laurens, Tuan Anh Le, Isabel Leal, Alex X. Lee, Tsang-Wei Edward Lee, Guy Lever, Jacky Liang, Li-Heng Lin, Fangchen Liu, Shangbang Long, Caden Lu, Sharath Maddineni, Anirudha Majumdar, Kevis-Kokitsi Maninis, Andrew Marmon, Sergio Martinez, Assaf Hurwitz Michaely, Niko Milonopoulos, Joss Moore, Robert Moreno, Michael Neunert, Francesco Nori, Joy Ortiz, Kenneth Oslund, Carolina Parada, Emilio Parisotto, Amaris Paryag, Acorn Pooley, Thomas Power, Alessio Quaglino, Haroon Qureshi, Rajkumar Vasudeva Raju, Helen Ran, Dushyant Rao, Kanishka Rao, Isaac Reid, David Rendleman, Krista Reymann, Miguel Rivas, Francesco Romano, Yulia Rubanova, Peter Pastor Sampedro, Pannag R Sanketi, Dhruv Shah, Mohit Sharma, Kathryn Shea, Mohit Shridhar, Charles Shu, Vikas Sindhwani, Sumeet Singh, Radu Soricut, Rachel Sterneck, Ian Storz, Razvan Surdulescu, Jie Tan, Jonathan Tompson, Saran Tunyasuvunakool, Jake Varley, Grace Vesom, Giulia Vezzani, Maria Bauza Villalonga, Oriol Vinyals, René Wagner, Ayzaan Wahid, Stefan Welker, Paul Wohlhart, Chengda Wu, Markus Wulfmeier, Fei Xia, Ted Xiao, Annie Xie, Jinyu Xie, Peng Xu, Sichun Xu, Ying Xu, Zhuo Xu, Jimmy Yan, Sherry Yang, Skye Yang, Yuxiang Yang, Hiu Hong Yu, Wenhao Yu, Wentao Yuan, Yuan Yuan, Jingwei Zhang, Tingnan Zhang, Zhiyuan Zhang, Allan Zhou, Guangyao Zhou, and Yuxiang Zhou. Gemini robotics 1.5: Pushing the frontier of generalist robots with advanced embodied reasoning, thinking, and motion transfer, 2025. URL https://arxiv.org/abs/2510.03342.

[10] Delin Qu, Haoming Song, Qizhi Chen, Zhaoqing Chen, Xianqiang Gao, Xinyi Ye, Qi Lv, Modi Shi, Guanghui Ren, Cheng Ruan, Maoqing Yao, Haoran Yang, Jiacheng Bao, Bin Zhao, and Dong Wang. Eo-1: Interleaved vision-text-action pretraining for general robot control. *arXiv preprint*, 2025. URL https://arxiv.org/abs/2508.21112.

[11] Chilam Cheang, Sijin Chen, Zhongren Cui, Yingdong Hu, Liqun Huang, Tao Kong, Hang Li, Yifeng Li, Yuxiao Liu, Xiao Ma, Hao Niu, Wenxuan Ou, Wanli Peng, Zeyu Ren, Haixin Shi, Jiawen Tian, Hongtao Wu, Xin Xiao, Yuyang Xiao, Jiafeng Xu, and Yichu Yang. Gr-3 technical report, 2025. URL https://arxiv.org/abs/2507.15493.

[12] Mustafa Shukor, Dana Aubakirova, Francesco Capuano, Pepijn Kooijmans, Steven Palma, Adil Zouitine, Michel Aractingi, Caroline Pascal, Martino Russi, Andres Marafioti, Simon Alibert, Matthieu Cord, Thomas Wolf, and Remi Cadene. Smolvla: A vision-language-action model for affordable and efficient robotics, 2025. URL https://arxiv.org/abs/2506.01844.

[13] Stephen James, Zicong Ma, David Rovick Arrojo, and Andrew J. Davison. Rlbench: The robot learning benchmark & learning environment. *IEEE Robotics and Automation Letters*, 2020.

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

[15] Oier Mees, Lukas Hermann, Erick Rosete-Beas, and Wolfram Burgard. Calvin: A benchmark for language-conditioned policy learning for long-horizon robot manipulation tasks. *IEEE Robotics and Automation Letters (RA-L)*, 7(3):7327–7334, 2022.

[16] Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on robot learning*, pages 1094–1100. PMLR, 2020.

[17] 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, 2024.

[18] 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.

[19] Chengshu Li, Ruohan Zhang, Josiah Wong, Cem Gokmen, Sanjana Srivastava, Roberto Martín-Martín, Chen Wang, Gabrael Levine, Wensi Ai, Benjamin Martinez, Hang Yin, Michael Lingelbach, Minjune Hwang, Ayano Hiranaka, Sujay Garlanka, Arman Aydin, Sharon Lee, Jiankai Sun, Mona Anvari, Manasi Sharma, Dhruva Bansal, Samuel Hunter, Kyu-Young Kim, Alan Lou, Caleb R Matthews, Ivan Villa-Renteria, Jerry Huayang Tang, Claire Tang, Fei Xia, Yunzhu Li, Silvio Savarese, Hyowon Gweon, C. Karen Liu, Jiajun Wu, and Li Fei-Fei. Behavior-1k:

A human-centered, embodied ai benchmark with 1,000 everyday activities and realistic simulation, 2024. URL https://arxiv.org/abs/2403.09227.

[20] 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, 2024. URL https://arxiv.org/abs/2412.18194.

[21] Ajay Mandlekar, Danfei Xu, Josiah Wong, Soroush Nasiriany, Chen Wang, Rohun Kulkarni, Li Fei-Fei, Silvio Savarese, Yuke Zhu, and Roberto Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. In *arXiv preprint arXiv:2108.03298*, 2021.

[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. *arXiv preprint arXiv:2406.02523*, 2024.

[23] Anonymous. Robocasa365: A large-scale simulation framework for training and benchmarking generalist robots. In *Submitted to The Fourteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=tQJYKwc3n4. under review.

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

[25] Yejin Kim, Wilbert Pumacay, Omar Rayyan, Max Argus, Winson Han, Eli VanderBilt, Jordi Salvador, Abhay Deshpande, Rose Hendrix, Snehal Jauhri, Shuo Liu, Nur Muhammad Mahi Shafiullah, Maya Guru, Arjun Guru, Ainaz Eftekhar, Karen Farley, Donovan Clay, Jiafei Duan, Piper Wolters, Alvaro Herrasti, Ying-Chun Lee, Georgia Chalvatzaki, Yuchen Cui, Ali Farhadi, Dieter Fox, and Ranjay Krishna. Molmospaces: A large-scale open ecosystem for robot navigation and manipulation, 2026.

[26] Quan Vuong, Sergey Levine, Homer Rich Walke, Karl Pertsch, Anikait Singh, Ria Doshi, Charles Xu, Jianlan Luo, Liam Tan, Dhruv Shah, et al. Open x-embodiment: Robotic learning datasets and rt-x models. In *Towards Generalist Robots: Learning Paradigms for Scalable Skill Acquisition@ CoRL2023*, 2023.

[27] Moo Jin Kim, Yihuai Gao, Tsung-Yi Lin, Yen-Chen Lin, Yunhao Ge, Grace Lam, Percy Liang, Shuran Song, Ming-Yu Liu, Chelsea Finn, and Jinwei Gu. Cosmos policy: Fine-tuning video models for visuomotor control and planning, 2026. URL https://arxiv.org/abs/2601.16163.

[28] Ajay Mandlekar, Soroush Nasiriany, Bowen Wen, Iretiayo Akinola, Yashraj Narang, Linxi Fan, Yuke Zhu, and Dieter Fox. Mimicgen: A data generation system for scalable robot learning using human demonstrations. In *7th Annual Conference on Robot Learning*, 2023.

[29] 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, 2025. URL https://arxiv.org/abs/2510.13626.

[30] 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. *[arXiv preprint arXiv:2510.03827]*, 2025.

[31] Jensen Gao, Suneel Belkhale, Sudeep Dasari, Ashwin Balakrishna, Dhruv Shah, and Dorsa Sadigh. A taxonomy for evaluating generalist robot manipulation policies, 2026. URL https://arxiv.org/abs/2503.01238.

[32] Arhan Jain, Mingtong Zhang, Kanav Arora, William Chen, Marcel Torne, Muhammad Zubair Irshad, Sergey Zakharov, Yue Wang, Sergey Levine, Chelsea Finn, Wei-Chiu Ma, Dhruv Shah, Abhishek Gupta, and Karl Pertsch. Polaris: Scalable real-to-sim evaluations for generalist robot policies, 2025. URL https://arxiv.org/abs/2512.16881.

[33] Zhenyu Jiang, Yuqi Xie, Kevin Lin, Zhenjia Xu, Weikang Wan, Ajay Mandlekar, Linxi Fan, and Yuke Zhu. Dexmimicgen: Automated data generation for bimanual dexterous manipulation via imitation learning. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, 2025.

[34] Caelan Garrett, Ajay Mandlekar, Bowen Wen, and Dieter Fox. Skillmimicgen: Automated demonstration generation for efficient skill learning and deployment, 2024. URL https://arxiv.org/abs/2410.18907.

[35] Yufei Wang, Zhou Xian, Feng Chen, Tsun-Hsuan Wang, Yian Wang, Katerina Fragkiadaki, Zackory Erickson, David Held, and Chuang Gan. Robogen: Towards unleashing infinite data for automated robot learning via generative simulation, 2024. URL https://arxiv.org/abs/2311.01455.

[36] Lirui Wang, Yiyang Ling, Zhecheng Yuan, Mohit Shridhar, Chen Bao, Yuzhe Qin, Bailin Wang, Huazhe Xu, and Xiaolong Wang. Gensim: Generating robotic simulation tasks via large language models, 2024. URL https://arxiv.org/abs/2310.01361.

[37] Yingdong Hu, Fanqi Lin, Pingyue Sheng, Chuan Wen, Jiacheng You, and Yang Gao. Data scaling laws in imitation learning for robotic manipulation, 2025. URL https://arxiv.org/abs/2410.18647.

[38] Percy Liang, Rishi Bommasani, Tony Lee, Dimitris Tsipras, Dilara Soylu, Michihiro Yasunaga, Yian Zhang, Deepak Narayanan, Yuhuai Wu, Ananya Kumar, et al.

Holistic evaluation of language models. *arXiv preprint arXiv:2211.09110*, 2022.

[39] Drew McDermott, Malik Ghallab, Adele E. Howe, Craig A. Knoblock, Ashwin Ram, Manuela M. Veloso, Daniel S. Weld, and David E. Wilkins. Pddl-the planning domain definition language. 1998. URL https://api.semanticscholar.org/CorpusID:59656859.

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

[41] Vaibhav Saxena, Matthew Bronars, Nadun Ranawaka Arachchige, Kuancheng Wang, Woo Chul Shin, Soroush Nasiriany, Ajay Mandlekar, and Danfei Xu. What matters in learning from large-scale datasets for robot manipulation. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://arxiv.org/pdf/2506.13536.

[42] Pim de Haan, Dinesh Jayaraman, and Sergey Levine. Causal confusion in imitation learning, 2019. URL https://arxiv.org/abs/1905.11979.

[43] Danny Driess, Jost Tobias Springenberg, Brian Ichter, Lili Yu, Adrian Li-Bell, Karl Pertsch, Allen Z. Ren, Homer Walke, Quan Vuong, Lucy Xiaoyang Shi, and Sergey Levine. Knowledge insulating vision-language-action models: Train fast, run fast, generalize better, 2025. URL https://arxiv.org/abs/2505.23705.

[44] Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *arXiv preprint arXiv:2303.04137*, 2023.

[45] Sudeep Dasari, Oier Mees, Sebastian Zhao, Mohan Kumar Srirama, and Sergey Levine. The ingredients for robotic diffusion transformers. *arXiv preprint arXiv:2410.10088*, 2024.

[46] William Peebles and Saining Xie. Scalable diffusion models with transformers, 2023. URL https://arxiv.org/abs/2212.09748.

[47] Andreas Steiner, André Susano Pinto, Michael Tschannen, Daniel Keysers, Xiao Wang, Yonatan Bitton, Alexey Gritsenko, Matthias Minderer, Anthony Sherbondy, Shangbang Long, Siyang Qin, Reeve Ingle, Emanuele Bugliarello, Sahar Kazemzadeh, Thomas Mesnard, Ibrahim Alabdulmohsin, Lucas Beyer, and Xiaohua Zhai. Paligemma 2: A family of versatile vlms for transfer, 2024. URL https://arxiv.org/abs/2412.03555.

[48] NVIDIA, Alisson Azzolini, Junjie Bai, Hannah Brandon, Jiaxin Cao, Prithvijit Chattopadhyay, Huayu Chen, Jinju Chu, Yin Cui, Jenna Diamond, Yifan Ding, Liang Feng, Francesco Ferroni, Rama Govindaraju, Jinwei Gu, Siddharth Gururani, Imad El Hanafi, Zekun Hao, Jacob Huffman, Jingyi Jin, Brendan Johnson, Rizwan Khan, George Kurian, Elena Lantz, Nayeon Lee, Zhaoshuo Li, Xuan Li, Maosheng Liao, Tsung-Yi Lin, Yen-Chen Lin, Ming-Yu Liu, Xiangyu Lu, Alice Luo, Andrew Mathau, Yun Ni, Lindsey Pavao, Wei Ping, David W. Romero, Misha Smelyanskiy, Shuran Song, Lyne Tchapmi, Andrew Z. Wang, Boxin Wang, Haoxiang Wang, Fangyin Wei, Jiashu Xu, Yao Xu, Dinghao Yang, Xiaodong Yang, Zhuolin Yang, Jingxu Zhang, Xiaohui Zeng, and Zhe Zhang. Cosmos-reason1: From physical common sense to embodied reasoning, 2025. URL https://arxiv.org/abs/2503.15558.

[49] Pranav Atreya, Karl Pertsch, Tony Lee, Moo Jin Kim, Arhan Jain, Artur Kuramshin, Clemens Eppner, Cyrus Neary, Edward Hu, Fabio Ramos, et al. Roboarena: Distributed real-world evaluation of generalist robot policies. In *Proceedings of the Conference on Robot Learning (CoRL 2025)*, 2025.

[50] Atharva Mete, Haotian Xue, Albert Wilcox, Yongxin Chen, and Animesh Garg. Quest: Self-supervised skill abstractions for learning continuous control, 2024. URL https://arxiv.org/abs/2407.15840.

[51] Albert Wilcox, Mohamed Ghanem, Masoud Moghani, Pierre Barroso, Benjamin Joffe, and Animesh Garg. Adapt3r: Adaptive 3d scene representation for domain transfer in imitation learning, 2025. URL https://arxiv.org/abs/2503.04877.

[52] Yuke Zhu, Josiah Wong, Ajay Mandlekar, Roberto Martín-Martín, Abhishek Joshi, Soroush Nasiriany, Yifeng Zhu, and Kevin Lin. robosuite: A modular simulation framework and benchmark for robot learning. In *arXiv preprint arXiv:2009.12293*, 2020.

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

[54] Franka Emika GmbH. *Panda User Manual*, 2023. Available at https://www.franka.de.

[55] 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.

[56] Frederik Ebert Jedrzej Orbik. Oculus reader: Robotic teleoperation interface, 2021. URL https://github.com/rail-berkeley/oculus_reader. Accessed: YYYY-MM-DD.

[57] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[58] Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. Film: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[59] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019. URL https://arxiv.org/abs/1711.05101.

APPENDIX A
SUPPLEMENT OVERVIEW AND FAQS

- In § A.1, we answer some commonly asked questions.
- In Appendix B, we provide several new experimental results.
- In Appendix C we provide detailed information about the MESA-Gen framework.
- In Appendix D we provide additional information about our data generation pipeline.
- In Appendix E, we provide additional details about MESA-Bench
- In Appendix F, we provide detailed policy training information.

A.1 Frequently Asked Questions (FAQs)

Why should I use MESA? MESA enables quick iteration while giving strong signal about language steering and performance along our axes of generalization. Furthermore, it is an easily extensible platform for designing new experimental settings.

What explains diffusion policy's unexpectedly poor performance? We study this question in depth in § B.1. To summarize, we find that our implementation works well in LIBERO and that DP simply struggles to fit the diverse setting presented by the benchmark.

What is this benchmark testing? We designed this benchmark to test language steering and policy performance under changes in spatial arrangement, object instance, object category and with unseen subtask compositions.

What does this benchmark not test for? This benchmark is not geared towards bimanual manipulation, dexterous or precise tasks, or open-ended language steerability.

What motivates the lack of scene and camera pose diversity? We purposely limit diversity along axes we don't test in order to minimize the amount of training required. That being said, this is implemented and easy to add back if desired.

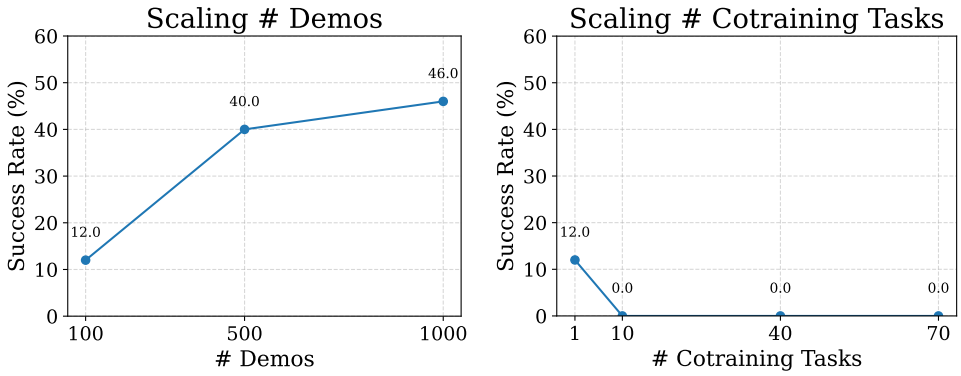
What are the limits of MimicGen data? MimicGen data has several limitations, especially for our domain with such scene variety. It is not well suited to deal with dense clutter, and even with our optimizations it sometimes outputs suboptimal trajectories. We do not claim in this paper to present a general solution for robot data generation, or that our data is optimal. Rather, we claim that MimicGen data is useful for inexpensively generating data for experiments, and that the stack ranking of policies trained on this data is predictive of real-world performance.

What types of tasks are included in the benchmark? The benchmark mainly focuses on pick and place and articulated manipulation tasks. This limited scope allows us to quickly generate data with MimicGen and train policies that achieve meaningful generalization under reasonable compute budgets.

APPENDIX B
ADDITIONAL EXPERIMENTAL RESULTS

B.1 Why Diffusion Policy Scales Poorly in Multi-Task MESA

One surprising finding is that, although we expected the diffusion policy to underperform relative to its VLA counterparts, it performs substantially worse than anticipated on the MESA benchmark, achieving an average success rate of 1.1%. We investigate this in Fig. 6, where we train a DiT policy [45] for a single task with increasing numbers of demonstrations and cotraining tasks. We see that it achieves a reasonable success rate with 1000 demonstrations, but performance drops quickly as the number of demonstrations decreases. Furthermore, we see that it especially struggles when switching from a single-task to multi-task setting, consistently achieving a 0% success rate when cotraining with other tasks. A likely explanation is that, compared to the baselines using a VLM backbone, the DiT policy has a much weaker vision-language conditioning pathway and cannot adequately fit the diverse data included in the MESA dataset.



(a) Single-task training **(b)** Co-training with 100 demonstrations

Figure 6: We study the performance of diffusion policy on a single task (`"put the lime in the bowl"`) as we scale the number of demonstrations and cotraining tasks. (a) We see that with 1000 demonstrations it achieves reasonable performance, but with 100 demonstrations, the number present for each task in the MESA, its performance drops sharply. (b) With 100 demonstrations we see that as we increase the number of cotraining tasks, performance quickly drops to zero.

Verifying with LIBERO

To verify our DiT Policy implementation we train and evaluate on LIBERO-90 [17]. We achieve a success rate of 90.7%, which is comparable to results reported elsewhere [50, 51] and *confirms that the poor DP results across MESA are due to the difficulty of the benchmark rather than implementation issues.*

B.2 Ablating Language Instructions

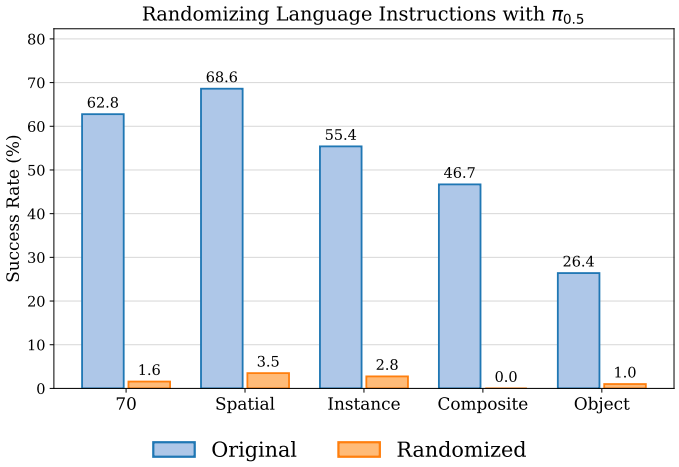
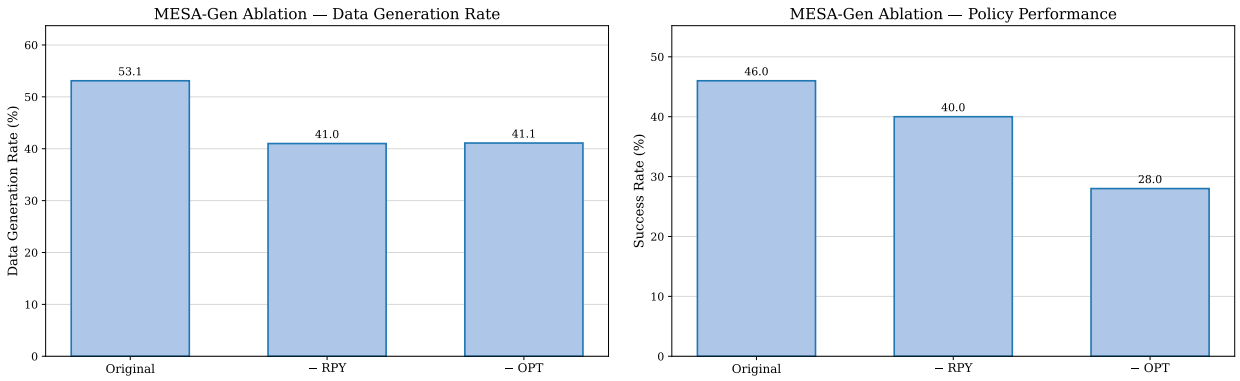


Figure 7: We evaluate $\pi_{0.5}$ with the original and randomized language instructions. This leads to a substantial performance drop, indicating the necessity of language following for MESA.

A core claim from our work is that the diverse initial configurations create a consistent requirement for language steerability. To verify this claim, we rerun evaluations for $\pi_{0.5}$, the strongest baseline, while scrambling the natural language instructions (precisely, for each evaluation we sample a random instruction from the benchmark), and present results in Fig. 7. We see that this change leads to a huge drop in success rate, with successes only occurring when the policy successfully guesses the correct instruction. This verifies our claim that accurate language steering is essential to perform well in the MESA benchmark.

B.3 Ablating MimicGen Changes



(a) Data Generation Rate

(b) Trained Policy Performance

Figure 8: We study the affects of our MimicGen changes when generating and training on 1000 demos for the "`Put the lime in the bowl`" task. We ablate the omission of roll, pitch and yaw (– RPY) and the starting point optimization procedure (– Opt). In (a) we study data generation rate, or the success rate of MimicGen at data generation time, and see that our changes lead to a consistent improvement. In (b) we study downstream policy performance and see that especially the starting point optimization leads to a substantial policy improvement.

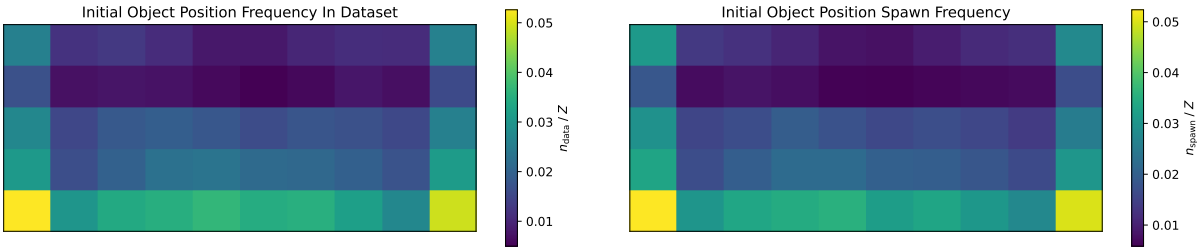
In the main draft we presented several modifications to MimicGen to improve data generation in our setting with many tasks and objects. These changes include subtask stitching and the trajectory starting point optimization procedure. The optimization procedure has two main components: (i) we remove roll, pitch and optionally yaw from the source and target object poses and (ii) we optimize the starting point of the trajectory as described in § D.3. Subtask stitching has a clear benefit – enabling data generation across over 800 tasks after only collecting data from 80 – which is difficult to ablate. We instead ablate the changes to the optimization procedure in Fig. 8. As shown in Fig. 8a, these changes both lead to an improvement in data generation rate, reducing compute requirements when generating large datasets. In Fig. 8b, we additionally observe that removing the optimization procedure leads to a substantial drop in performance. This is likely because without optimizing the start of the trajectory, the generated data can include large, unnecessary interpolations to distant regions before moving towards the target object. In effect, the data includes many suboptimal motions that can confuse the policy. This interpretation is further supported by trajectory-length statistics: removing the optimization procedure increases the average trajectory length from 141.7 to 177.9.

B.4 Bias in Generated Data Toward High Success-Rate Initial Conditions

MimicGen works by initializing the environment, attempting to solve the task using its trajectory transformations, and then reinitializing it once it exceeds some retry budget. This process can bias the dataset towards initial configurations where MimicGen is likely to succeed, creating issues for the downstream policy. We study this issue by computing normalized initial position frequency in Fig. 9. We observe minor bias in the dataset with more data in the lower-right quadrant and fewer in the upper-left quadrant. However, within each tile, the deviation from an unbiased distribution is at most about 15%, reflecting a relatively evenly distributed dataset.

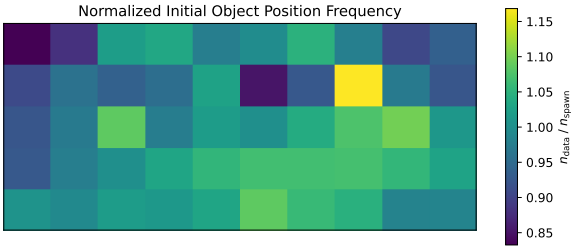
B.5 Visualizing Confusion Matrices

In Fig. 10 we visualize confusion matrices for DP and $\pi_{0.5}$, which we can compute directly using the logged distractor task completion data. One notably observation is that, despite being more performant overall, $\pi_{0.5}$ is confused more often. This is likely because it is simply more capable of successfully grasping and placing objects; as a result, even if it occasionally takes an incorrect action, it is still likely to complete the distractor task and register as a confusion. Another second observation is that the confusion matrix for $\pi_{0.5}$ is much more "spiky", with a handful of frequently repeated mistakes, whereas diffusion policy's confusions are scattered more uniformly across the confusion matrix. Importantly, the high-frequency confusions made by $\pi_{0.5}$ are intuitive. For example, peaches and tomatoes look visually similar to apples and are therefore often confused with them. This suggests that $\pi_{0.5}$ is generally making reasonable attempts to select the intended object, with errors driven by visual similarity. In contrast, diffusion policy seems to select incorrect objects more indiscriminately, consistent with behavior that resembles grabbing random objects.



(a) Frequency in the Dataset

(b) Spawn Frequency



(c) Normalized Frequency

Figure 9: We study whether MimicGen led to a biased dataset by visualizing the frequency with which different target object initial positions appear in our dataset. In (a), we show this value, and in (b) we show the frequency with which initial states spawn before MimicGen. Since these distributions are highly correlated, we show the normalized distribution in (c). We see that there is some bias towards the lower right corner of the region and away from the upper left corner. In (c), values greater than one reflect higher frequency in the dataset than an unbiased dataset would have and values less than one are the opposite.

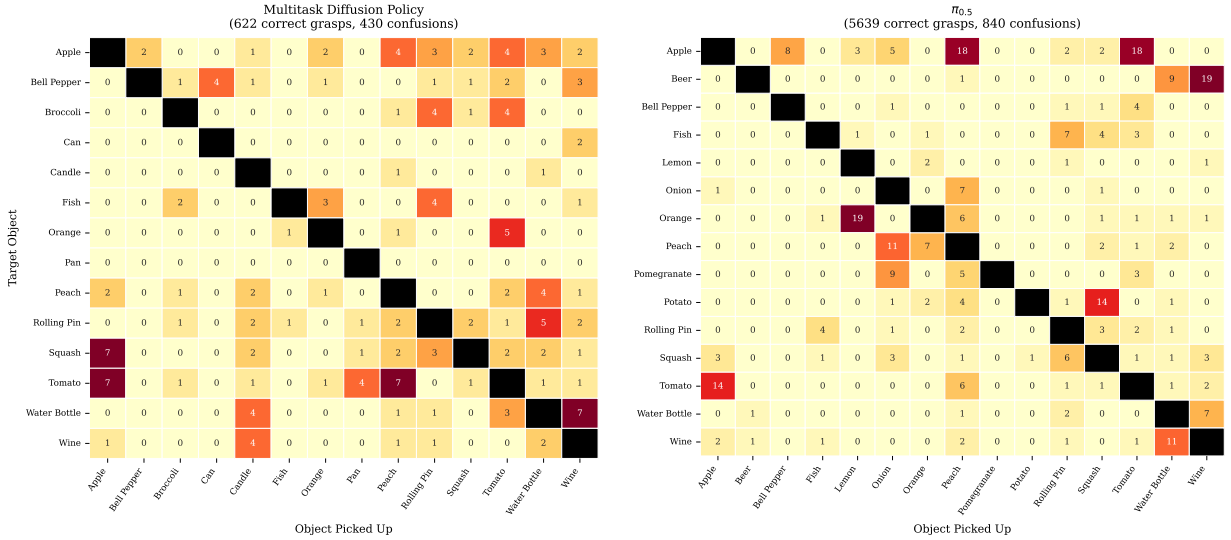


Figure 10: We compare confusion matrices for diffusion policy and $\pi_{0.5}$. Entry i, j in these plots shows the frequency with which a given policy completed a distractor task by grasping i when it was supposed to grasp j. For example, both confusion matrices show a lot of confusion for $\texttt{tomato}, \texttt{apple}$ because the policy frequently grasped an apple when it was supposed to grasp a tomato. We condensed them to a reasonable size by selecting the top 15 most confusing entries in each matrix and removing rows and columns corresponding to objects not involved in those entries. For example, no object was commonly confused for mushrooms, so its rows and columns were removed.

B.6 Per-Task Success Rates

Here we list success rates across all tasks present in the benchmark. We present success rates for MESA-70 in Table VI, MESA-Spatial in Table VII, MESA-Instance in Table VIII, MESA-Composite in Table IX, and MESA-Category in Table X.

Table VI: Success rates for all tasks in the MESA-70 task set. These are also the set of tasks in the training set.

MESA-70	DP	PG-Bin	PG-FM	GR00T-N1.6	π_0	FAST	$\pi_{0.5}$
put the apple on the tray	2.0	6.0	54.0	62.0	**66.0**	64.0	**66.0**

MESA-70	DP	PG-Bin	PG-FM	GR00T-N1.6	π_0	FAST	$\pi_{0.5}$
put the avocado in the basket	0.0	2.0	58.0	18.0	50.0	70.0	**88.0**
put the bagel on the cutting board	8.0	0.0	26.0	36.0	48.0	36.0	**86.0**
put the bar in the bottom drawer of the cabinet and close it	0.0	0.0	20.0	12.0	22.0	30.0	**46.0**
put the bar soap in the top drawer of the cabinet and close it	0.0	0.0	24.0	20.0	24.0	58.0	**60.0**
put the beer in the slide cabinet and close it	0.0	0.0	6.0	6.0	10.0	10.0	**34.0**
put the bell pepper in the box and close it	0.0	0.0	10.0	10.0	14.0	26.0	**52.0**
put the bottled water on the tray	4.0	2.0	36.0	30.0	42.0	66.0	**84.0**
put the bowl in the microwave and close it	0.0	0.0	22.0	10.0	34.0	12.0	**40.0**
put the broccoli in the pan	4.0	2.0	56.0	52.0	64.0	62.0	**72.0**
put the candle on the tray	6.0	8.0	48.0	30.0	52.0	48.0	**62.0**
put the carrot in the bowl	0.0	0.0	30.0	16.0	44.0	50.0	**68.0**
put the cheese on the plate	0.0	0.0	32.0	24.0	46.0	70.0	**84.0**
put the corn on the cutting board	6.0	0.0	56.0	30.0	72.0	76.0	**88.0**
put the croissant in the slide cabinet and close it	0.0	0.0	16.0	10.0	24.0	28.0	**36.0**
put the cucumber in the bottom drawer of the cabinet and close it	0.0	0.0	8.0	12.0	16.0	24.0	**48.0**
put the cup in the right side of the dish drainer	2.0	0.0	32.0	20.0	42.0	48.0	**62.0**
put the egg in the bowl	0.0	0.0	24.0	24.0	40.0	42.0	**74.0**
put the eggplant on the cutting board	0.0	0.0	38.0	28.0	44.0	68.0	**78.0**
put the fish in the pan	2.0	2.0	40.0	24.0	46.0	56.0	**72.0**
put the fish on the plate	0.0	0.0	40.0	14.0	48.0	52.0	**78.0**
put the garlic in the middle drawer of the cabinet and close it	0.0	0.0	56.0	54.0	52.0	50.0	**70.0**
put the garlic in the pan	2.0	4.0	68.0	32.0	76.0	58.0	**78.0**
put the jam in the basket	0.0	2.0	46.0	20.0	64.0	68.0	**88.0**
put the jug in the basket	0.0	0.0	10.0	20.0	10.0	32.0	**64.0**
put the kiwi in the top drawer of the cabinet and close it	0.0	0.0	28.0	28.0	42.0	34.0	**60.0**
put the lemon on the plate	0.0	0.0	36.0	32.0	48.0	48.0	**74.0**
put the lime in the bowl	0.0	10.0	70.0	60.0	80.0	84.0	**96.0**
put the mango in the box and close it	0.0	0.0	20.0	2.0	24.0	34.0	**56.0**
put the mug in the left side of the dish drainer	0.0	0.0	22.0	22.0	24.0	42.0	**48.0**
put the mushroom in the bowl	0.0	0.0	38.0	30.0	38.0	48.0	**66.0**
put the onion on the tray	0.0	2.0	56.0	52.0	64.0	66.0	**70.0**
open the top drawer of the cabinet and put the avocado in it	0.0	0.0	10.0	6.0	28.0	40.0	**58.0**
open the slide cabinet and put the book in it	6.0	0.0	58.0	32.0	68.0	64.0	**70.0**
open the bottom drawer of the cabinet and put the broccoli in it	0.0	0.0	0.0	4.0	16.0	0.0	**22.0**
open the box and put the can in it	2.0	0.0	22.0	20.0	32.0	42.0	**62.0**
open the slide cabinet and put the candle in it	2.0	0.0	24.0	14.0	12.0	20.0	**32.0**
open the microwave and put the canned food in it	0.0	0.0	24.0	20.0	18.0	24.0	**52.0**
open the top drawer of the cabinet and put the carrot in it	0.0	0.0	12.0	12.0	42.0	34.0	**64.0**
open the box and put the cheese in it	0.0	0.0	14.0	8.0	30.0	38.0	**58.0**

MESA-70	DP	PG-Bin	PG-FM	GR00T-N1.6	π_0	FAST	$\pi_{0.5}$
open the microwave and put the eggplant in it	0.0	0.0	14.0	4.0	28.0	18.0	**50.0**
open the middle drawer of the cabinet and put the lemon in it	0.0	0.0	12.0	6.0	40.0	46.0	**62.0**
open the bottom drawer of the cabinet and put the mushroom in it	0.0	0.0	2.0	2.0	10.0	2.0	**20.0**
open the middle drawer of the cabinet and put the onion in it	2.0	0.0	14.0	6.0	28.0	32.0	**46.0**
open the top drawer of the cabinet and put the peach in it	2.0	0.0	6.0	18.0	20.0	22.0	**52.0**
open the microwave and put the squash in it	0.0	0.0	6.0	4.0	10.0	28.0	**64.0**
open the slide cabinet and put the tomato in it	4.0	0.0	44.0	60.0	80.0	68.0	**90.0**
open the slide cabinet and put the water bottle in it	4.0	0.0	16.0	54.0	60.0	60.0	**72.0**
open the slide cabinet and put the wine in it	0.0	0.0	18.0	22.0	34.0	32.0	**46.0**
put the orange on the cutting board	2.0	0.0	22.0	14.0	54.0	48.0	**56.0**
put the orange on the plate	2.0	2.0	40.0	36.0	64.0	70.0	**74.0**
put the peach in the basket	0.0	0.0	52.0	24.0	66.0	64.0	**88.0**
put the peach on the cutting board	2.0	2.0	52.0	32.0	50.0	52.0	**58.0**
put the peach on the tray	0.0	0.0	50.0	46.0	60.0	56.0	**74.0**
put the plate in the left side of the dish drainer	0.0	0.0	2.0	2.0	8.0	14.0	**16.0**
put the potato in the microwave and close it	0.0	0.0	8.0	8.0	24.0	22.0	**38.0**
put the rolling pin in the right side of the dish drainer	0.0	0.0	10.0	8.0	20.0	22.0	**36.0**
put the rolling pin in the pan	2.0	2.0	34.0	30.0	58.0	56.0	**60.0**
put the rolling pin on the tray	2.0	0.0	30.0	26.0	**62.0**	50.0	50.0
put the sponge in the middle drawer of the cabinet and close it	0.0	0.0	12.0	0.0	10.0	12.0	**24.0**
put the squash in the pan	0.0	2.0	26.0	28.0	46.0	**66.0**	64.0
put the squash on the tray	2.0	0.0	24.0	26.0	56.0	**68.0**	66.0
put the tomato on the cutting board	8.0	4.0	72.0	58.0	**88.0**	76.0	86.0
put the tomato in the pan	8.0	6.0	64.0	**92.0**	86.0	82.0	88.0
put the tomato on the tray	8.0	0.0	72.0	74.0	90.0	70.0	**96.0**
put the water bottle in the basket	4.0	0.0	34.0	20.0	52.0	66.0	**78.0**
put the water bottle on the tray	10.0	2.0	30.0	38.0	50.0	50.0	**74.0**
put the wine in the box	0.0	2.0	34.0	18.0	58.0	70.0	**76.0**
put the wine on the tray	2.0	0.0	30.0	28.0	44.0	66.0	**68.0**
put the yogurt in the basket	2.0	0.0	26.0	10.0	44.0	48.0	**56.0**

Table VII: Success rates for all tasks in the MESA-Spatial task set.

MESA-Spatial	DP	PG-Bin	PG-FM	GR00T-N1.6	π_0	FAST	$\pi_{0.5}$
open the slide cabinet and put the tomato in it	2.0	0.0	34.0	30.0	58.0	54.0	**74.0**
open the slide cabinet and put the water bottle in it	0.0	0.0	12.0	6.0	16.0	28.0	**46.0**
put the rolling pin in the right side of the dish drainer	0.0	0.0	4.0	6.0	8.0	32.0	**38.0**
put the rolling pin in the pan	4.0	4.0	12.0	36.0	26.0	46.0	**62.0**

MESA-Spatial	DP	PG-Bin	PG-FM	GR00T-N1.6	π_0	FAST	$\pi_{0.5}$
put the rolling pin on the tray	4.0	2.0	28.0	22.0	28.0	54.0	**72.0**
put the tomato on the cutting board	0.0	2.0	54.0	28.0	58.0	72.0	**78.0**
put the tomato in the pan	0.0	14.0	64.0	42.0	76.0	74.0	**90.0**
put the tomato on the tray	0.0	8.0	54.0	46.0	68.0	76.0	**86.0**
put the water bottle in the basket	0.0	0.0	40.0	6.0	44.0	58.0	**76.0**
put the water bottle on the tray	0.0	0.0	48.0	10.0	40.0	42.0	**64.0**

Table VIII: Success rates for all tasks in the MESA-Instance task set.

MESA-Instance	DP	PG-Bin	PG-FM	GR00T-N1.6	π_0	FAST	$\pi_{0.5}$
put the apple on the tray	10.0	4.0	38.0	46.0	48.0	56.0	**64.0**
put the beer in the slide cabinet and close it	0.0	0.0	6.0	4.0	10.0	6.0	**14.0**
put the bell pepper in the box and close it	0.0	0.0	10.0	4.0	10.0	14.0	**30.0**
put the cheese on the plate	0.0	0.0	4.0	2.0	2.0	12.0	**28.0**
put the fish in the pan	4.0	0.0	32.0	54.0	60.0	76.0	**80.0**
put the jam in the basket	0.0	0.0	50.0	24.0	52.0	44.0	**78.0**
put the mushroom in the bowl	0.0	2.0	28.0	24.0	46.0	46.0	**72.0**
open the top drawer of the cabinet and put the carrot in it	2.0	0.0	16.0	16.0	28.0	30.0	**64.0**
open the top drawer of the cabinet and put the peach in it	0.0	0.0	8.0	2.0	22.0	30.0	**60.0**
open the microwave and put the squash in it	0.0	0.0	2.0	2.0	8.0	26.0	**30.0**
open the slide cabinet and put the wine in it	2.0	0.0	2.0	8.0	10.0	12.0	**40.0**
put the orange on the plate	2.0	0.0	22.0	32.0	40.0	44.0	**56.0**
put the peach in the basket	0.0	2.0	38.0	36.0	74.0	58.0	**82.0**
put the peach on the cutting board	0.0	0.0	40.0	48.0	40.0	50.0	**64.0**
put the peach on the tray	0.0	0.0	54.0	50.0	74.0	**78.0**	**78.0**
put the potato in the microwave and close it	0.0	0.0	2.0	2.0	18.0	12.0	**30.0**
put the squash in the pan	2.0	0.0	12.0	30.0	20.0	44.0	**56.0**
put the squash on the tray	4.0	0.0	24.0	26.0	36.0	36.0	**52.0**
put the wine in the box	2.0	2.0	22.0	18.0	34.0	40.0	**50.0**
put the wine on the tray	10.0	2.0	44.0	26.0	48.0	68.0	**80.0**

Table IX: Success rates for all tasks in the MESA-Composite task set.

MESA-Composite	DP	PG-Bin	PG-FM	GR00T-N1.6	π_0	FAST	$\pi_{0.5}$
put the apple on the plate	0.0	0.0	66.0	54.0	70.0	64.0	**84.0**
put the broccoli and mushroom on the cutting board	0.0	0.0	0.0	0.0	0.0	2.0	**4.0**
put the carrot in the pan	2.0	0.0	30.0	42.0	58.0	60.0	**74.0**
put the cheese in the basket	0.0	0.0	28.0	6.0	36.0	**62.0**	**62.0**
put the cheese and mug on the tray	0.0	0.0	0.0	0.0	**2.0**	0.0	**2.0**
put the cup in the left side of the dish drainer	4.0	0.0	22.0	**34.0**	24.0	28.0	**34.0**
put the fish and garlic in the pan	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
put the lemon on the cutting board	0.0	0.0	34.0	40.0	54.0	44.0	**76.0**
put the lemon and lime in the basket	0.0	0.0	0.0	0.0	2.0	6.0	**12.0**
put the mushroom on the tray	0.0	4.0	52.0	48.0	52.0	56.0	**74.0**
put the onion in the bowl	0.0	0.0	40.0	20.0	70.0	62.0	**80.0**

MESA-Composite	DP	PG-Bin	PG-FM	GR00T-N1.6	π_0	FAST	$\pi_{0.5}$
open the middle drawer of the cabinet and put the apple in it	0.0	0.0	14.0	12.0	38.0	24.0	**50.0**
open the box and put the apple in it and close it	0.0	0.0	0.0	0.0	4.0	2.0	**44.0**
open the box and put the beer in it	0.0	0.0	6.0	10.0	10.0	14.0	**38.0**
open the middle drawer of the cabinet and put the broccoli in it	0.0	0.0	14.0	6.0	30.0	34.0	**66.0**
open the top drawer of the cabinet and put the carrot in it and close it	0.0	0.0	0.0	4.0	10.0	14.0	**46.0**
open the slide cabinet and put the jam in it	0.0	0.0	20.0	16.0	52.0	48.0	**74.0**
open the middle drawer of the cabinet and put the mushroom in it and close it	0.0	0.0	2.0	0.0	6.0	10.0	**40.0**
open the microwave and put the potato in it and close it	0.0	0.0	0.0	0.0	0.0	0.0	**2.0**
put the sponge in the right side of the dish drainer	0.0	0.0	26.0	8.0	32.0	48.0	**72.0**

Table X: Success rates for all tasks in the MESA-Category task set.

MESA-Category	DP	PG-Bin	PG-FM	GR00T-N1.6	π_0	FAST	$\pi_{0.5}$
put the beet in the pot	0.0	0.0	0.0	12.0	8.0	10.0	**38.0**
put the brussel sprout in the pan	0.0	0.0	2.0	6.0	**8.0**	4.0	**8.0**
put the chili pepper in the bowl	0.0	0.0	20.0	**38.0**	22.0	4.0	26.0
put the cupcake in the basket	2.0	0.0	14.0	**28.0**	14.0	8.0	26.0
put the ginger on the cutting board	0.0	0.0	10.0	12.0	0.0	**26.0**	20.0
put the grapes in the basket	2.0	0.0	12.0	16.0	24.0	8.0	**28.0**
put the ice cream in the bowl	0.0	0.0	4.0	2.0	**34.0**	4.0	32.0
put the kebabs in the pan	0.0	0.0	20.0	18.0	12.0	16.0	**44.0**
put the olive oil in the basket	0.0	0.0	4.0	2.0	6.0	6.0	**46.0**
put the pear in the pot	4.0	0.0	0.0	**20.0**	12.0	2.0	16.0
put the pomegranate in the top drawer of the cabinet	0.0	0.0	**8.0**	6.0	2.0	2.0	4.0
put the potato on the baking sheet	2.0	0.0	4.0	16.0	34.0	8.0	**42.0**
put the radish on the cutting board	2.0	0.0	4.0	8.0	8.0	8.0	**32.0**
put the raspberry in the basket	0.0	0.0	14.0	10.0	8.0	6.0	**18.0**
put the salt shaker in the box and close it	0.0	0.0	2.0	2.0	4.0	4.0	**34.0**
put the sausage on the baking sheet	2.0	0.0	8.0	**20.0**	6.0	0.0	14.0
put the scone on the plate	0.0	0.0	0.0	12.0	2.0	**14.0**	12.0
put the strawberry in the bowl	0.0	0.0	28.0	48.0	20.0	34.0	**74.0**
put the sushi on the plate	0.0	0.0	0.0	2.0	8.0	2.0	**10.0**
put the thermos in the slide cabinet	0.0	0.0	0.0	**4.0**	0.0	2.0	**4.0**

APPENDIX C
MESA DETAILS

C.1 Implemented Task Skeletons

We implement task skeletons as simple Python classes specifying task-relevant objects, initialization predicates, demonstration predicates, and goal predicates. Our pipeline currently includes the `PickAndPlace` skeleton, which encompasses all pick and place tasks, and `MultiStepArticulatedManipulation`, which encompasses tasks involving articulated manipulation, such as opening them, closing them, and placing objects in them.

C.2 Simulation Platform

We build on MimicLabs [41], which integrates infrastructure from robosuite [52], MimicGen [28], LIBERO [17] and RoboCasa [22], and uses the Mujoco simulator [53]. We use the Franka Panda embodiment [54] asset from robosuite with a Robotiq 2F-85 gripper whose asset was sourced from Mujoco Menagerie [55]. Our environment includes left and right shoulder cameras and a wrist camera, but we only use the the left shoulder and wrist perspectives for our experiments.

C.3 Ensuring Visual Confusion via Distractor Sampling

This section expands the distractor sampling procedure referenced in the main paper, which aims to maintain high observation-conditioned task entropy $H(\mathcal{T}|\mathcal{I})$ and reduce causal confusion.

Each scene contains a mixture of task-relevant objects and task-irrelevant distractor objects. For example, for the task "`put the apple in the basket`", the task-relevant objects would be the apple and the basket, and the task-irrelevant objects might objects such as a plate. The goal is that for each initial state $i \in \mathcal{I}$ corresponding to a task $t \in \mathcal{T}$, there exists some some task $t' \in \mathcal{T}$ such that the task-relevant objects for t' appear in i. To achieve this, we define $\mathcal{O}$ as the set of graspable objects (e.g., apples) and $\mathcal{D}$ to be the set of destination objects (e.g., baskets). For each object $o \in \mathcal{O}$, we define a set $\mathcal{D}_o \subset \mathcal{D}$ as the set of destination objects in which o is placed during some task in MESA-70. Similarly, for each destination $d \in \mathcal{D}$, we define $\mathcal{O}_d \subset \mathcal{O}$ to be the set of objects placed in d. From here, we start by placing the task-relevant objects and then iteratively add more objects, sampling from the combined sets according to the placed items, until we have the desired number of distractor objects. This process is summarized for pick and place tasks in Alg. 1, and we apply a similar algorithm to choose valid articulated objects.

Algorithm 1 Distractor sampling with visual confusion guarantees.

Require: target distractor object counts n_O, n_D, task relevant objects o, d, maps between objects and destinations $\mathcal{O}_d$, $\mathcal{D}_o$
Ensure: list of distractor object keys

$\mathcal{O}_{\text{placed}} \leftarrow \{o\}$, $\mathcal{D}_{\text{placed}} \leftarrow \{d\}$
while $|\mathcal{O}_{\text{placed}}| < n_O$ or $\mathcal{D}_{\text{placed}} < n_D$ **do**
  if $|\mathcal{O}_{\text{placed}}| < n_O$ **then**
    Sample object $o \in \bigcup_{d \in \mathcal{D}_{\text{placed}}} \mathcal{O}_d \setminus \mathcal{O}_{\text{placed}}$
    $\mathcal{O}_{\text{placed}} \leftarrow \mathcal{O}_{\text{placed}} \cup \{o\}$
  if $|\mathcal{D}_{\text{placed}}| < n_D$ **then**
    Sample destination $d \in \bigcup_{o \in \mathcal{O}_{\text{placed}}} \mathcal{D}_o \setminus \mathcal{D}_{\text{placed}}$
    $\mathcal{D}_{\text{placed}} \leftarrow \mathcal{D}_{\text{placed}} \cup \{d\}$
    $c_D +$
return $\mathcal{O}_{\text{placed}}, \mathcal{D}_{\text{placed}}$

C.4 Example: `PickAndPlace` Skeleton

This section provides a sample `PickAndPlace` skeleton.

```
from dataclasses import dataclass, field
from typing import List, Optional, Tuple

from .objects import (
    get_grasp_object_spec,
    get_dest_object_spec,
)

from .task import BaseTask, BaseTaskConfig, register_task
from .utils import (
    make_named_grasp_object_spec,
    make_named_dest_object_spec,
    NamedGraspObjectSpec,
```

```
    NamedDestObjectSpec,
    NamedArticulatedObjectSpec,
)

@dataclass(frozen=True, kw_only=True)
class PickAndPlaceTaskConfig(BaseTaskConfig):
    """Configuration for generating pick-and-place task suites."""

    dest_object: str
    pick_objects: str | List[str]
    dest_region: str
    goal_predicate: str
    class_name: str = "PickAndPlaceTask"


@register_task("pick_and_place", config_cls=PickAndPlaceTaskConfig)
class PickAndPlaceTask(BaseTask):
    def __init__(self, config: PickAndPlaceTaskConfig) -> None:
        super().__init__(config=config)

        if isinstance(config.pick_objects, str):
            pick_objects = [config.pick_objects]
        else:
            pick_objects = config.pick_objects
        pick_objects = [get_grasp_object_spec(object_name) for object_name in pick_objects]
        self._pick_objects: List[NamedGraspObjectSpec] = []
        for i, spec in enumerate(pick_objects):
            object_name = f"object_{i}"
            self._pick_objects.append(make_named_grasp_object_spec(object_name, spec))

        dest_object = get_dest_object_spec(config.dest_object)
        self._dest_object: NamedDestObjectSpec = make_named_dest_object_spec("dest_object",
        ↪  dest_object)
        self._dest_region = config.dest_region
        self._goal_predicate = config.goal_predicate

    def build_task_objects(
        self
    ) -> Tuple[List[NamedGraspObjectSpec], List[NamedDestObjectSpec],
    ↪  List[NamedArticulatedObjectSpec]]:
        return self._pick_objects, [self._dest_object], []

    def build_goal_state(
        self
    ) -> List[object]:
        goal_state: List[object] = ["and"]
        for pick_object in self._pick_objects:
            if self._goal_predicate == "on":
                goal_state.append(["on", pick_object.id, self._dest_object.id])
            else:
                assert self._dest_region is not None
                contain_region_name = f"{self._dest_object.id}_{self._dest_region}"
                goal_state.append(["in", pick_object.id, contain_region_name])
        return goal_state

    def build_demonstration_states(
        self,
    ) -> List[list]:
        steps: List[list] = []
        for pick_object in self._pick_objects:
            steps.append(["grasp", pick_object.id])
            if self._goal_predicate == "on":
                steps.append(["on", pick_object.id, self._dest_object.id])
            else:
                assert self._dest_region is not None
                contain_region_name = f"{self._dest_object.id}_{self._dest_region}"
                steps.append(["in", pick_object.id, contain_region_name])
        return steps
```

C.5 Example: BDDL File

Here we provide a sample BDDL file for the task "`open the box and put the apple in it and close it`". Note that whereas prior work saves these in the `.bddl` file format, we found this to add unnecessary implementation complexity since they are ultimately parsed into Python dictionaries. To avoid this, we simply save them as JSON files as we see below. This BDDL file shows how we instantiate several distractor objects, including a bell pepper, tray, bottled water, pan, tomato and squash. In `distractor_goal_specs` we see how the possible distractor tasks are enumerated. In `demonstration_states` we see the predicates that are used when decomposing the task for MimicGen.

Notice in `regions` that the initialization regions all cover the same area of the table. Unlike prior works such as LIBERO [17], which contains each object to a small region without inter-object overlap, we allow objects to spawn across the full table and filter out overlapping initial configurations. To do this, we start by labeling each object with a 2D bounding box of its projection onto the table, which is already done for RoboCasa [22] objects and we do manually for other objects. Next, we sort objects by the sizes of their bounding boxes and iterate through them, randomly placing an object until it doesn't intersect with any already placed objects.


```
{
    "problem_name": "mimiclabs_lab1_tabletop_manipulation",
    "fixtures": {
        "table": [
            "table"
        ],
        "sliding_top_box_631": [
            "sliding_top_box_631_1"
        ]
    },
    "regions": {
        "table_sliding_top_box_631_1_init_region": {
            "target": "table",
            "ranges": [
                [
                    0.05,
                    -0.4,
                    0.4,
                    0.4
                ]
            ],
            "extra": [],
            "yaw_rotation": [
                -1.5707963267948966,
                1.5707963267948966
            ],
            "rgba": [
                0.0,
                0.0,
                1.0,
                0.0
            ]
        },
        "table_object_1_init_region": {
            "target": "table",
            "ranges": [
                [
                    0.05,
                    -0.4,
                    0.4,
                    0.4
                ]
            ],
            "extra": [],
            "yaw_rotation": [
                -1.5707963267948966,
                1.5707963267948966
            ],
            "rgba": [
                0.0,
```

```
                0.0,
                1.0,
                0.0
            ]
        },
        "table_robocasa_bell_pepper_1_1_1_init_region": {
            "target": "table",
            "ranges": [
                [
                    0.05,
                    -0.4,
                    0.4,
                    0.4
                ]
            ],
            "extra": [],
            "yaw_rotation": [
                -1.5707963267948966,
                1.5707963267948966
            ],
            "rgba": [
                0.0,
                0.0,
                1.0,
                0.0
            ]
        },
        "table_wooden_tray_1_2_init_region": {
            "target": "table",
            "ranges": [
                [
                    0.05,
                    -0.4,
                    0.4,
                    0.4
                ]
            ],
            "extra": [],
            "yaw_rotation": [
                -1.5707963267948966,
                1.5707963267948966
            ],
            "rgba": [
                0.0,
                0.0,
                1.0,
                0.0
            ]
        },
        "table_robocasa_bottled_water_12_1_3_init_region": {
            "target": "table",
            "ranges": [
                [
                    0.05,
                    -0.4,
                    0.4,
                    0.4
                ]
            ],
            "extra": [],
            "yaw_rotation": [
                -1.5707963267948966,
                1.5707963267948966
            ],
            "rgba": [
                0.0,
                0.0,
                1.0,
```

```
                0.0
            ]
        },
        "table_robocasa_pan_24_1_4_init_region": {
            "target": "table",
            "ranges": [
                [
                    0.05,
                    -0.4,
                    0.4,
                    0.4
                ]
            ],
            "extra": [],
            "yaw_rotation": [
                -1.5707963267948966,
                1.5707963267948966
            ],
            "rgba": [
                0.0,
                0.0,
                1.0,
                0.0
            ]
        },
        "table_robocasa_tomato_0_1_5_init_region": {
            "target": "table",
            "ranges": [
                [
                    0.05,
                    -0.4,
                    0.4,
                    0.4
                ]
            ],
            "extra": [],
            "yaw_rotation": [
                -1.5707963267948966,
                1.5707963267948966
            ],
            "rgba": [
                0.0,
                0.0,
                1.0,
                0.0
            ]
        },
        "table_robocasa_squash_1_1_6_init_region": {
            "target": "table",
            "ranges": [
                [
                    0.05,
                    -0.4,
                    0.4,
                    0.4
                ]
            ],
            "extra": [],
            "yaw_rotation": [
                -1.5707963267948966,
                1.5707963267948966
            ],
            "rgba": [
                0.0,
                0.0,
                1.0,
                0.0
            ]
```

```
        },
        "sliding_top_box_631_1_contain_region": {
            "target": "sliding_top_box_631_1",
            "ranges": [],
            "extra": [],
            "yaw_rotation": [
                0,
                0
            ],
            "rgba": [
                0,
                0,
                1,
                0
            ]
        }
    },
    "objects": {
        "robocasa_apple_0": [
            "object_1"
        ],
        "robocasa_bell_pepper_1": [
            "robocasa_bell_pepper_1_1_1"
        ],
        "wooden_tray": [
            "wooden_tray_1_2"
        ],
        "robocasa_bottled_water_12": [
            "robocasa_bottled_water_12_1_3"
        ],
        "robocasa_pan_24": [
            "robocasa_pan_24_1_4"
        ],
        "robocasa_tomato_0": [
            "robocasa_tomato_0_1_5"
        ],
        "robocasa_squash_1": [
            "robocasa_squash_1_1_6"
        ]
    },
    "textures": {},
    "camera": {
        "rightshoulder": {
            "r": 1.2,
            "theta": 1.0471975511965976,
            "phi": 0.7853981633974483,
            "r_radius": 0.0,
            "theta_radius": 0.0,
            "phi_radius": 0.0
        },
        "leftshoulder": {
            "r": 1.2,
            "theta": 1.0471975511965976,
            "phi": -0.7853981633974483,
            "r_radius": 0.0,
            "theta_radius": 0.0,
            "phi_radius": 0.0
        }
    },
    "lighting": {},
    "styles": {},
    "scene_properties": {},
    "initial_state": [
        [
            "on",
            "sliding_top_box_631_1",
            "table_sliding_top_box_631_1_init_region"
        ],
```

```
            [
                "on",
                "object_1",
                "table_object_1_init_region"
            ],
            [
                "on",
                "robocasa_bell_pepper_1_1_1",
                "table_robocasa_bell_pepper_1_1_1_init_region"
            ],
            [
                "on",
                "wooden_tray_1_2",
                "table_wooden_tray_1_2_init_region"
            ],
            [
                "on",
                "robocasa_bottled_water_12_1_3",
                "table_robocasa_bottled_water_12_1_3_init_region"
            ],
            [
                "on",
                "robocasa_pan_24_1_4",
                "table_robocasa_pan_24_1_4_init_region"
            ],
            [
                "on",
                "robocasa_tomato_0_1_5",
                "table_robocasa_tomato_0_1_5_init_region"
            ],
            [
                "on",
                "robocasa_squash_1_1_6",
                "table_robocasa_squash_1_1_6_init_region"
            ]
        ],
        "goal_state": [
            "and",
            [
                "close",
                "sliding_top_box_631_1"
            ],
            [
                "in",
                "object_1",
                "sliding_top_box_631_1_contain_region"
            ]
        ],
        "distractor_goal_specs": [
            [
                "on",
                "object_1",
                "wooden_tray_1_2"
            ],
            [
                "on",
                "object_1",
                "robocasa_pan_24_1_4"
            ],
            [
                "in",
                "robocasa_bell_pepper_1_1_1",
                "sliding_top_box_631_1_contain_region"
            ],
            [
                "on",
                "robocasa_bell_pepper_1_1_1",
                "wooden_tray_1_2"
```

```
            ],
            [
                "on",
                "robocasa_bell_pepper_1_1_1",
                "robocasa_pan_24_1_4"
            ],
            [
                "in",
                "robocasa_bottled_water_12_1_3",
                "sliding_top_box_631_1_contain_region"
            ],
            [
                "on",
                "robocasa_bottled_water_12_1_3",
                "wooden_tray_1_2"
            ],
            [
                "on",
                "robocasa_bottled_water_12_1_3",
                "robocasa_pan_24_1_4"
            ],
            [
                "in",
                "robocasa_pan_24_1_4",
                "sliding_top_box_631_1_contain_region"
            ],
            [
                "on",
                "robocasa_pan_24_1_4",
                "wooden_tray_1_2"
            ],
            [
                "on",
                "robocasa_pan_24_1_4",
                "robocasa_pan_24_1_4"
            ],
            [
                "in",
                "robocasa_tomato_0_1_5",
                "sliding_top_box_631_1_contain_region"
            ],
            [
                "on",
                "robocasa_tomato_0_1_5",
                "wooden_tray_1_2"
            ],
            [
                "on",
                "robocasa_tomato_0_1_5",
                "robocasa_pan_24_1_4"
            ],
            [
                "in",
                "robocasa_squash_1_1_6",
                "sliding_top_box_631_1_contain_region"
            ],
            [
                "on",
                "robocasa_squash_1_1_6",
                "wooden_tray_1_2"
            ],
            [
                "on",
                "robocasa_squash_1_1_6",
                "robocasa_pan_24_1_4"
            ]
        ],
        "demonstration_states": [
```

```
            [
                "open",
                "sliding_top_box_631_1"
            ],
            [
                "grasp",
                "object_1"
            ],
            [
                "in",
                "object_1",
                "sliding_top_box_631_1_contain_region"
            ],
            [
                "close",
                "sliding_top_box_631_1"
            ]
        ],
        "language_instruction": [
            "open",
            "the",
            "box",
            "and",
            "put",
            "the",
            "apple",
            "in",
            "it",
            "and",
            "close",
            "it"
        ]
}
```

C.6 Example: Initial State Distribution

In Fig. 11 we visualize 20 initialization states for the "`open the box and put the apple in it and close it`" task. The top left image corresponds to the BDDL file shown in § C.5. Notice that each one has a unique set of distractor objects, posing substantial vision-language grounding challenges.

Figure 11: We visualize 20 initialization states for the "`open the box and put the apple in it and close it`" task.

APPENDIX D
DATA GENERATION DETAILS

D.1 Subtask Stitching for Unseen Tasks

This section provides details referenced in the main paper for composing demonstrations for unseen tasks by stitching object-centric subtasks sourced from other tasks with compatible predicates.

Demonstration Lookup System: We construct a hierarchical index offline by parsing all demonstration HDF5 files and extracting subtask-level metadata. This index is organized primarily by predicate type (e.g., *grasp, in, on, open, close, stack, turnon, turnoff*), and further refined by using object-centric semantic attributes such as shape category, object category, insertion type, destination region, and object identity. The exact hierarchy depth and key structure depend on the predicate semantics; for example, *grasp* predicates are indexed by the grasped object's shape and category, whereas *in* predicates additionally encode both the source object and destination container properties.

Each leaf node of the index contains a list of concrete subtask segments extracted from demonstrations, including the path to the source HDF5 file, the demonstration identifier, object references involved in the interaction, the subtask's index within the original trajectory, and the termination signal defining the subtask boundary. This is sufficient to recover the full state-action segment corresponding to the subtask and to re-contextualize it for a new task.

At generation time, we query this index using a predicate-specific set of search keys from the current subtask specification. These keys are computed using object metadata functions that map object instances to semantic categories like shape class or insertion type. Retrieval is performed by a recursive tree walk that follows the exact key if present. If a key is not present, we take the union across all children at that depth, which broadens the search to semantically related subtasks.

From the retrieved candidate list, we select using either *RandomStrategy* or one of several nearest-neighbor strategies: *NearestNeighborObjectStrategy* (weighted object pose distance), *NearestNeighborRobotDistanceStrategy* (robot travel distance to the first transformed pose), or *NearestNeighborEefObjectTranslationStrategy* (relative end-effector-object translation). Nearest-neighbor selection computes distances for all candidates and samples uniformly among the top-k closest (default $k = 3$).

Stitching Procedure: Each subtask is first trimmed from its source demonstration using start and end indices derived from predicate-specific termination signals stored in the HDF5 logs. For non-final subtasks, boundaries are detected via the first valid rising edge of the corresponding terminal signal following the previous subtask; the final subtask extends to the end of the demonstration. To increase diversity, subtask end boundaries may be randomly offset within a task-specified range, except for the final subtask, which is fixed to prevent truncation.

When concatenating subtasks, the system does not resample trajectories globally but instead inserts short transition segments to ensure temporal continuity. Transitions optionally insert i) a linear interpolation segment that smoothly moves the end-effector from the final pose of the previous subtask to the initial pose of the next, and ii) a fix-pose hold segment that maintains the first pose of the next subtask for a fixed number of steps. To avoid duplicate commands, the first waypoint of each appended segment is removed prior to concatenation. After all subtasks are merged, their actions are concatenated into a single action sequence.

The environment is reset once at the beginning of the stitched trajectory, and all subsequent subtasks execute sequentially without intermediate resets. We transform subtask trajectories to the current scene by preserving the relative pose between the end-effector and the reference object. When objects differ in canonical rotation axes, an additional rotation correction is applied to align coordinate conventions.

Demonstrations with missing or invalid subtask termination signals are skipped during dataset parsing. We enforce non-empty subtasks, strictly increasing subtask indices, and valid ordering of tasks. On generation, if no compatible subtasks are found even after using the broader semantic fallbacks mentioned previously, generation terminates.

D.2 Source Demonstration Collection Protocol

We collect human demonstrations for source tasks to seed MimicGen and subtask stitching. In total, we collect 1125 demonstrations across 80 tasks using a Meta Quest VR setup [56]. During data collection, control commands are executed at 20 Hz. For the Quest controller, translational and rotational inputs are normalized to ± 1.0 along axes and passed directly to the controller without action clipping, using a gain of 1.5. Source tasks are selected to broadly cover the objects and predicates supported in the benchmark. We filter tasks by a data generation rate cutoff of 10% after collecting 5 demonstrations per task.

D.3 Trajectory Optimizations for Large Pose Differences

As mentioned in the main paper, we make several changes to the MimicGen pipeline to handle situations with large changes between source and target object pose.

Removing Roll and Pitch in MimicGen Transforms We ignore roll and pitch when computing MimicGen transformations to improve stability for objects without stable resting poses. Concretely, given an input rotation $R \in SO(3)$, we extract static XYZ Euler angles (ϕ, θ, ψ) (roll, pitch, yaw) via `mat2euler(R, axes="sxyz")`, and reconstruct a pure-yaw rotation by setting roll and pitch to zero. We enable this when grasping or placing objects and ignore it for other predicates (for example opening

or closing articulated objects). Additionally, we ignore yaw for *on* predicates, or when interacting with objects rotationally symmetric about the vertical axis. These include, for example, bottles, cans and apples.

Trajectory Start Optimization We expand upon the discussion of the starting point optimization described in the main paper.

Let $\{\tau_i\}_0^N$ be a sequence of end-effector poses outputted from MimicGen, and let $T_o(0)$ and $T_e(0)$ be the poses of the target object and end-effector at the beginning of the rollout. For each pose τ_i, we solve for an angle $\theta_i \in (-\pi, \pi]$ defining a rotation about the vertical axis through $T_o(0)$ that best aligns the rotated pose with the current end-effector position:

$$\tilde{\tau}_i(\theta) \;=\; T_o(0)\, R_z(\theta)\, T_o(0)^{-1}\, \tau_i, \tag{1}$$

$$\theta_i \;=\; \arg\min_{\theta \in (-\pi,\pi]} \left\| p(\tilde{\tau}_i(\theta)) - p(T_e(0)) \right\|_2, \tag{2}$$

where $R_z(\theta)$ denotes a rotation about the world vertical axis and $p(\cdot)$ extracts the translational component of an $SE(3)$ transform. We then select the starting index as the waypoint from the first half of the trajectory that can be made closest to the current end-effector position under such a rotation:

$$i^\star \;=\; \arg\min_{i \in \{0,\ldots,\frac{N}{2}\}} \min_{\theta_i \in (-\pi,\pi]} \left\| p(\tilde{\tau}_i(\theta_i)) - p(T_e(0)) \right\|_2, \tag{3}$$

and execute the rotated, truncated trajectory $\{\tilde{\tau}_i(\theta_{i^*})\}_{i=i^\star}^N$. This ensures execution begins from the point along the optimally rotated trajectory that is closest to the current end-effector pose, avoiding large initial deviations.

While this operation leads to more natural trajectory starting points, it also may lead to incorrect grasps for objects where end-effector yaw rotation is important, but will lead without modification to natural motions for objects where this is not the case. To alleviate this, we use an LLM to label objects based on whether they are geometrically symmetric under yaw rotation, and slerp end-effector rotation back to the original MimicGen output when they are not.

The next issue is that the current end-effector rotation is sometimes far from the rotation at the beginning of the MimicGen trajectory, even after optimizing the starting pose, causing unnecessary rapid rotations at the beginning of the subtask. To alleviate this, we compute an offset rotation R_{offset} aligning the beginning of the trajectory with the current end-effector rotation. We then compute a sequence of offsets slerping from R_{offset} to the identity and left multiply these by the end-effector orientations in our trajectory.

APPENDIX E
MESA-BENCH DETAILS

All MESA tasks are listed in § B.6.

E.1 Suite Construction Details

Here we list details about all of our tasks suites.

MESA-70

This evaluation set is composed of the 70 tasks from the MESA-70 training dataset. Notably, it includes in-distribution variants of all the tasks from MESA-Spatial and MESA-Instance

MESA-Spatial

By default, x coordinates are sampled from $\mathrm{Uniform}(0.05, 0.4)$ and y from $\mathrm{Uniform}(-0.4, 0.4)$, which we empirically found to be the subset of the table where the robot could reliably interact with objects. We partition the table into two regions, $\mathcal{R}_{\mathrm{ID}}$ for the training set and MESA-70 evaluation set, and $\mathcal{R}_{\mathrm{OOD}}$ for the MESA-Spatial out-of-distribution evaluation set. For $\mathcal{R}_{\mathrm{ID}}$ we instead sample y coordinates from $\mathrm{Uniform}(-0.4, 0.0)$ and for $\mathcal{R}_{\mathrm{OOD}}$ we sample them from $\mathrm{Uniform}(0.0, 0.4)$.

MESA-Instance

For each task, we partition the set of object assets into disjoint in-distribution and out-of-distribution subsets, training exclusively on the in-distribution assets and evaluating on held-out instances at test time. As with MESA-Spatial, the in-distribution asset variants are included in MESA-70, enabling direct comparison under changes in object appearance and geometry.

For three objects, namely the peach, squash and wine bottle, we hold out some assets from the RoboCasa-objaverse object set. For the rest, we replace the original object instance with RoboCasa AI-generated assets, which are never seen during training.

MESA-Composite

These are unseen composite tasks composed of subtasks seen in the MESA-70 dataset.

We include 12 tasks with familiar structure. Some are easy, such as "`put the apple on the plate`", which composes skills from seen tasks "`put the apple on the tray`" and "`put the cheese on the plate`". Others have familiar structure but are designed adversarially. For example, "`put the cup in the left side of the dish drainer`" is similar to the seen task "`put the cup in the right side of the dish drainer`", but to complete it correctly the policy must notice this small change in task description and adjust the placement accordingly.

We include 8 tasks with unfamiliar structure. Four of these are two-step pick and place tasks, such as "`put the fish and garlic in the pan`", which are unfamiliar since the training data only includes one-step pick and place tasks such as "`put the fish in the pan`". The other four are articulated manipulation tasks involving opening objects, placing things in them, and then closing them. These are unfamiliar since the training tasks only opened or closed objects, but never both.

MESA-Category

These tasks all involve unseen grasp objects sourced from the RoboCasa AI-generated assets. In addition, some include the `pot` and `baking sheet` placement locations, which are also unseen. For example, for the task "`put the beet in the pot`", the policy has never seen a beet or pot during finetuning.

E.2 Training and Auxiliary Datasets

Our main MESA-70 training set includes 7000 demonstrations across the 70 tasks in MESA, which is about 1M frames. We also include an auxiliary dataset of 100 demonstrations per task for 736 tasks, and train on a subset of this with 10 demonstrations per task, which also has about 1M frames.

E.3 Evaluation Details

To minimize bias caused by unlucky starting configurations, we generate a fixed set of starting configurations. We use these configurations for all of our experiments and include them in our open-source code release. When generating these starting configurations, we ensure that the target object is not occluded or out of frame and that the target task has not already been completed.

We evaluate each policy for 50 rollouts for each task, with each rollout consisting of a different set of distractor objects. Note that since the distractor objects and initial positions are sampled completely independently of the training configurations, *every single initial observation in the benchmark is unique*, and solving this benchmark through memorization alone is infeasible.

APPENDIX F
TRAINING DETAILS FOR BASELINES

Here we share additional details about our baselines.

F.1 Shared Details

Unless noted otherwise, all VLA / VLM-based policies except for GR00T are trained for 50k gradient steps with a batch size of 128. For GR00T, The PaliGemma baselines and π models are trained using the JAX version of the official `openpi` repository, and GR00T is trained using the official `Isaac-GR00T` repository. We use the default optimization parameters from these repositories for all experiments. All models are trained with a left shoulder camera image and wrist camera image. All models are evaluated using the final model at the end of training. All models use relative joint angle action space, which is a common choice in the pretraining data for our main VLA baselines. Unless otherwise noted, all models are trained with a chunk size matching the control frequency of 20 Hz and we replan after executing the first 5 actions. In general, we didn't tune hyperparameters unless performance was surprisingly poor.

F.2 Baseline-Specific Details

- **Multitask Diffusion Policy (DP)**: We use the DiT Block Policy from Dasari et al. [45] with 6 blocks, an embed dimension of 512 and feedforward dim of 3200, yielding a total of 80M parameters. This architecture uses a ResNet-18 [57] backbone with FiLM [58] to extract 7×7 featrues vectors from each image. These images are concatenated with the projected state and language embedding to before being fed to a transformer encoder. The outputs of the encoder are pooled and used together with the diffusion timestep as AdaLN conditioning [46] for the diffusion transformer decoder. We use the AdamW [59] optimizer with a learning rate of 1×10^{-4}, weight decay of 1×10^{-6} and cosine annealing learning rate decay.
- **PaliGemma-Binning (PG-Bin)**: We use the implementation in the `openpi` repository, originally intended for use as a baseline in Atreya et al. [49]. The baseline discretizes each action dimension into 256 bins. Since this baseline is extremely token-inefficient, we only predict chunks of 5 actions to save compute. The model autoregressively predicts these binned actions.
- **PaliGemma-Flow Matching (PG-FM)**: This is implemented in the `openpi` repository and is identical to the π_0 baseline, but using PaliGemma pretrained weights for the backbone and initializing the action head from scratch.
- **GR00T-N1.6**: This is implemented in the official `Isaac-GR00T` repository and is trained using the default configurations. Notably, this includes leaving the backbone frozen. We originally trained with a batch size of 128 and retrained with a batch size of 1024, which we found to lead to a modest improvement.
- $\boldsymbol{\pi_0}$, $\boldsymbol{\pi_0}$-FAST, $\boldsymbol{\pi_{0.5}}$: These were all trained using the default configuration from the `openpi` repository. For $\pi_{0.5}$ we finetuned with and without knowledge insulation, found this not to make much difference, and report the version without.

F.3 Training Compute Usage

An important goal for our framework is to allow academic researchers to quickly iterate with minimal compute requirements. We find that MESA's dataset was small enough to achieve saturation after a relatively quick training cycle. For the `openpi` repository models, the full 50k training steps took 48 hours on four L40S GPUs or 21 hours on two H100s. We also saw that $\pi_{0.5}$ achieved strong performance even after 5k gradient steps, and that it achieved strong performance with only 10% of our data, meaning it may be possible to shrink the budget even further. For GR00T, the version with a batch size 128 took about 12 hours on one H100 and the version with batch size 1024 took about 30 hours with four H100s.