

HAMSTER: HIERARCHICAL ACTION MODELS FOR OPEN-WORLD ROBOT MANIPULATION

Anonymous authors

Paper under double-blind review

ABSTRACT

Large models have shown strong open-world generalization to complex problems in vision and language, but they have been relatively more difficult to deploy in robotics. This challenge stems primarily from the lack of scalable robotic training data since this requires expensive on-robot collection. For scalable training, these models must show considerable transfer across domains, to make use of cheaply available “off-domain” data such as videos, hand-drawn sketches, or data from simulation. In this work, we posit that hierarchical vision-language-action models can be more effective at transferring behavior across domains than standard monolithic vision-language-action models. In particular, we study a class of hierarchical vision-language-action models, where high-level vision-language models (VLMs) are trained on relatively cheap data to produce semantically meaningful intermediate predictions such as 2D paths indicating desired behavior. These predicted 2D paths serve as guidance for low-level control policies that are 3D-aware and capable of precise manipulation. In this work, we show that separating prediction into semantic high-level predictions, and 3D-aware low-level predictions allows such *hierarchical* VLA policies to transfer across significant domain gaps, from simulation to the real world or across scenes with widely varying visual appearance. Doing so allows for the usage of cheap, abundant data sources beyond teleoperated on-robot data thereby enabling broad semantic and visual generalization. We demonstrate how hierarchical architectures trained on such cheap off-domain data can enable robotic manipulation with semantic, visual, and geometric generalization through experiments in simulation and the real world.

1 INTRODUCTION

Developing general robot manipulation policies has been notoriously difficult. With the advent of large vision-language models (VLMs) that display compelling generalizations, there is an optimism that similar techniques can be helpful for robotic manipulation. Several prior works (Team et al., 2024; Kim et al., 2024; Gu et al., 2023) build open-world vision-language-action models (VLAs) by finetuning off-the-shelf, pretrained VLMs. The recipe for training many of these VLA models has been to collect and curate a large-scale robotics-specific dataset, complete with images and corresponding on-robot actions, and then finetune a VLM to directly produce actions (Kim et al., 2024; Brohan et al., 2023a). Such VLAs have shown robustness on simple tasks and controlled environmental variations. However, these models display limited generalization in terms of environment, object, task, and semantic variation. This issue could be attributed to the scarcity of diverse, in-domain training data. The data needed to train these models is expensive since it requires end-to-end image-action pairs that must all be collected directly on-robot. A solution for training VLA models must be developed to instead learn from easy-to-collect “cheap” sources of data.

On the other hand, relatively “small” imitation learning models have shown impressive dexterity and geometric robustness. Such models have demonstrated promise across a range of complex tasks involving contact-rich manipulation and 3D reasoning, spanning domains from tabletop manipulation (Shridhar et al., 2023; Goyal et al., 2023) to fine dexterous manipulation (Zhao et al., 2023). Trained on relatively small datasets, these models show local robustness and stable control but typically lack semantic or visual generalization. They are often brittle to changes in the environment, semantic description of the tasks, or changes in the objects being manipulated (Pumacay et al., 2024). This fragility can also be boiled down to scarce in-domain data collected on a robot.

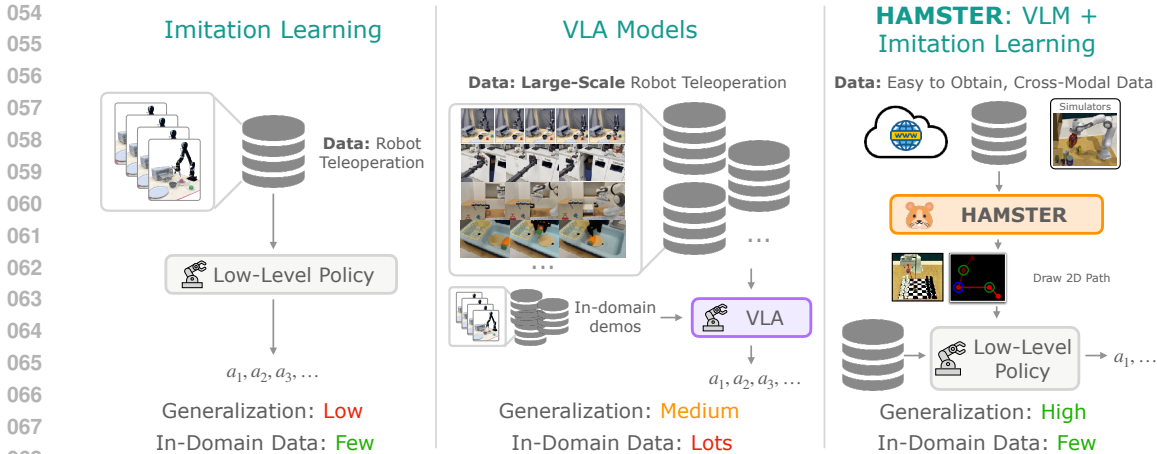


Figure 1: Overview of HAMSTER, VLAs and “smaller” imitation learning methods. HAMSTER’s hierarchical design results in better generalization with a small amount of in-domain data. HAMSTER is able to utilize cheap training sources such as videos or simulations for enhanced generalization.

Reliable, generalizable robotic learning techniques must marry the generalization benefits of large VLMs, with the efficiency, local robustness and dexterity of small imitation learning policies, all while being able to train from abundant and cheap sources of data. In this work, we ask – can we design VLA models that train on relatively abundant and cheap data sources, showing broad visual and semantic generalization, while capturing the low-level geometric and 3D understanding displayed by small imitation learning models?

We propose that a hierarchical architecture for vision-language-actions models, HAMSTER (**H**ierarchical **A**ction **M**odels with **S**eparate**D** Path **R**epresentations), can serve as an effective way to learn from abundant and cheap sources of data such as videos or simulation. We study a family of HAMSTERS, where finetuned VLMs are connected to low-level 3D policy learning methods via intermediate 2D path representations. Since these 2D paths can easily be obtained in abundance from data sources such as videos or simulations (either with point tracking, hand-sketching, or proprioceptive projection), these can be used to finetune the larger higher-level VLM in HAMSTER. These 2D paths can then serve as guidance for a low-level policy that operates on rich 3D and proprioceptive inputs, alleviating the burden of long-horizon planning and semantic reasoning, allowing low-level policies to focus on robustly generating precise, spatially-aware actions.

Representations similar to 2D paths has been explored in the robot learning literature (Gu et al., 2023), primarily as a technique for flexible task specification. However, the key hypothesis explored in this paper is distinct – we posit that using cheap data such as videos or simulation to finetune *hierarchical* path generating VLMs can enable a surprising degree of cross-domain transfer as compared to the direct transfer of monolithic vision-language-action models (Brohan et al., 2022; Kim et al., 2024). Here the focus is less on using paths as a scalable technique for task specification, and more on using hierarchy as a mechanism for robust cross-domain transfer across settings with considerable visual and semantic differences. Specifically, we find that VLMs trained to predict 2D path representation can transfer to the real world from simulations that look very different from the real world, or across real-world scenarios with widely varying appearance. Hence, the hierarchical design of HAMSTER provides a way to utilize cheaper, but perceptually varying sources of “off-domain” data (such as simulation or cross-embodiment data) to benefit real-world control policies.

The hierarchical design presented in HAMSTER can also offer additional advantages through the decoupling of VLM training and low-level action prediction. Specifically, since the higher-level VLM is predicting semantically meaningful trajectories from monocular RGB camera inputs, the lower-level control policies can operate from rich 3D and proprioceptive inputs. In doing so, HAMSTER inherits the semantic reasoning benefits of VLMs along with the 3D reasoning and spatial awareness benefits of 3D imitation learning policies (Goyal et al., 2024; Ke et al., 2024). Finally, since HAMSTER is built on both open-source VLMs and low-level policies, it can serve as a fully open-sourced enabler for the community-building vision-language-action models.

2 RELATED WORK

LLMs and VLMs for robotics. Early attempts in leveraging LLMs and VLMs for robotics are through pretrained language (Jang et al., 2022; Shridhar et al., 2023; Singh et al., 2023) and visual (Shah & Kumar, 2021; Parisi et al., 2022; Nair et al., 2023; Ma et al., 2023) models. However, these are not sufficient for complex semantic reasoning and generalization to the open world (Brohan et al., 2022; Zitkovich et al., 2023). Recent research has focused on directly leveraging open world reasoning and generalization capability of LLMs and VLMs, by prompting or fine-tuning them to, e.g., generate plans (Huang et al., 2022; 2023b; Lin et al., 2023; Liang et al., 2023; Singh et al., 2023; Brohan et al., 2023b), construct value (Huang et al., 2023a) and reward functions (Kwon et al., 2023; Sontakke et al., 2023; Yu et al., 2023; Ma et al., 2024; Wang et al., 2024). Our work is more closely related to the literature on VLA models, summarized below.

Monolithic VLA models as language-conditioned robot policies. Monolithic VLA models have been proposed to produce robot actions given task description and image observations directly (Brohan et al., 2022; Jiang et al., 2023; Zitkovich et al., 2023; Team et al., 2024; Kim et al., 2024; Radosavovic et al., 2023). Monolithic VLA models are often constructed from VLMs (Liu et al., 2024b; Bai et al., 2023; Driess et al., 2023; Lin et al., 2024), and are trained on large-scale robot teleoperation data (Brohan et al., 2022; Collaboration et al., 2023; Khazatsky et al., 2024) to predict actions as text or special tokens. However, due to the lack of coverage in existing robotics datasets, they must be finetuned in-domain on expensive teleoperated data. The most relevant monolithic VLA model is LLARVA (Niu et al., 2024), which predicts end-effector trajectories in addition to robot actions. However, LLARVA does not use trajectory prediction to control the robot; rather, it uses it as an auxiliary task to improve action prediction. Therefore, LLARVA still suffers from the limitations of monolithic VLA models. In contrast, our work takes a hierarchical approach, enabling us to use specialist lower-level policies that take in additional inputs the VLMs cannot support, such as 3D pointclouds, to enable better imitation learning. Our predicted paths then enable these lower-level policies to generalize more effectively.

VLMs for predicting intermediate representations. Our work bears connections to prior methods using vision-language models for intermediate prediction. These methods can be categorized by the choice of predicted representation:

Point-based predictions: A common intermediate prediction interface has been keypoint affordances (Stone et al., 2023; Sundaresan et al., 2023; Nasiriany et al., 2024; Yuan et al., 2024). Some examples include using open-vocabulary detectors (Minderer et al., 2022), iterative prompting of VLMs (Nasiriany et al., 2024), or fine-tuning detectors to identify certain parts of an object by semantics (Sundaresan et al., 2023). Perhaps most related, Yuan et al. (2024) finetunes a VLM to predict objects of interest as well as free space for placing an object, and Liu et al. (2024a) propose a mark-based visual prompting procedure to predict keypoint affordances as well as a fixed number of waypoints. As opposed to these, our work finetunes a VLM model to not just predict points but rather entire 2D paths, making it more broadly applicable across robotic tasks.

Trajectory-based predictions: The idea of using trajectory-based task specifications to condition low-level policies was proposed in RT-trajectory (Gu et al., 2023), largely from the perspective of flexible task specification. This work also briefly discusses the possibility of combining RT-Trajectory with trajectory sketches generated from prompting a pre-trained vision language model. Complementary to RT-Trajectory, the focus of this work is less on the use of trajectory sketches for task specification, but rather on the abilities of a hierarchical VLA model to finetune the high-level VLM on cheap and abundant sources. This could include training data such as videos or simulation data, and show transfer to test scenarios of interest with considerable visual and semantic variation. While RT-trajectory uses human effort or off-the-shelf pre-trained models to generate trajectories, we show that finetuning VLM models on cheap data sources can generate more accurate and generalizable trajectories (see Table. 5). Moreover, our instantiation of this architecture enables the incorporation of rich 3D and proprioceptive information, as compared to monocular 2D policies (Gu et al., 2023).

Leveraging simulation data for training robot policies. There has been extensive work on leveraging simulation for robot learning. Simulation data is popular in reinforcement learning (RL), as RL on real robotic systems is often impractical due to high sample complexity and safety concerns (Lee et al., 2020; Handa et al., 2023; Torne et al., 2024). Recently, simulation has been also

exploited to directly generate (Fishman et al., 2022) or bootstrap (Mandlekar et al., 2023) large-scale datasets for imitation learning, to reduce the amount of expensive robot teleoperation data needed. Our work takes a different approach - using simulation data to finetune a VLM, and showing that VLM is able to transfer the knowledge learned from simulation data to real robot systems, despite considerable visual differences. A related observation is recently made by (Yuan et al., 2024), but they use keypoint affordances as the interface between the VLM and the low-level policy as opposed to more general expressive 2D path representations.

3 BACKGROUND

Imitation Learning via Supervised Learning. The goal of imitation learning is to train a probabilistic policy $\pi_\theta(a \mid s, o, z)$ from an expert-provided dataset. This policy π_θ outputs the probability of producing action a conditioned on proprioceptive states s , perceptual observations o , and language instructions z that specify the task. In the typical imitation learning setting, a dataset of expert in-domain trajectories is provided, consisting of observation-action-language tuples $\mathcal{D} = \{(s_i, a_i, o_i, z_i)\}_{i=1}^N$. This dataset can be utilized to learn the parameters of the policy π_θ . While π can take on a variety of architectures with various training objectives (Goyal et al., 2023; Ke et al., 2024; Zhao et al., 2023; Chi et al., 2023), most imitation learning algorithms are trained via supervised learning to maximize the objective: $\mathbb{E}_{(s_i, a_i, o_i, z_i) \sim \mathcal{D}} [\log \pi_\theta(a_i \mid s_i, o_i, z_i)]$. This core objective can be modified with rich architectural choices such as 3D policy architectures (Goyal et al., 2023; Ke et al., 2024) or more expressive policy distribution classes (Zhao et al., 2023; Chi et al., 2023), but generalization to out-of-domain to settings with semantic or visual variations is still challenging. We study how vision-language models can be used to aid the generalization of such low-level imitation learning-based policies, discussed in Section 4.1.

Vision Language Models. Typical vision language models (VLMs) (Lin et al., 2024; Liu et al., 2024b) are large transformers (Vaswani et al., 2023) that take vision & text tokens as input and produce text responses. These models are pre-trained on large multimodal datasets (Zhu et al., 2023; Byeon et al., 2022), and then finetuned on targeted high-quality datasets (Shen et al., 2021; Lu et al., 2022). These models tokenize each modality into a shared space to produce a sequence of output tokens corresponding to text or other output modalities. In this work, we assume access to a pre-trained, text and image input VLM (Lin et al., 2024; Liu et al., 2024b), that autoregressively outputs a sequence of text tokens conditioned on an image and previous text tokens. These pretrained VLMs can typically be finetuned using a supervised prediction loss that minimizes the negative log-likelihood of the answer text tokens.

4 HAMSTER: HIERARCHICAL ACTION MODELS FOR ROBOTIC LEARNING

In this work, we examine how VLA models can be trained on relatively abundant data to demonstrate cross-domain transfer capabilities, as opposed to training on expensive image-action data collected on a robot. HAMSTER is a family of hierarchical action models designed for this purpose, exhibiting generalizable and robust manipulation. It consists of two interconnected models: first, a higher-level VLM that is fine-tuned on large-scale, cross-modal data to produce intermediate guidance (detailed in Section 4.1), and second, a low-level policy that produces actions conditioned on the VLM’s predicted guidance (detailed in Section 4.2). The finetuned VLM and the low-level policy communicate using a 2D path representation. Figure 2 provides an overview of HAMSTER’s design. Crucially, we study the ability of such a hierarchical design to enable training on cheap, abundantly available data such as simulation and videos.

Problem Definition. Rather than operating in the pure imitation learning setting as described in Section 3, we study a scenario where cross-domain data is utilized to train VLA models. While the typical imitation learning setting uses a dataset of optimal in-domain, on-robot tuples $\mathcal{D} = \{(s_t, a_t, o_t, z_t)\}_{t=1}^N$ to learn a near-optimal policy π_θ , in this setting we additionally assume access to a much larger dataset(s) of “off-domain” approximately optimal data $\mathcal{D}_{\text{off}} = \{(o_i^o, z_i^o)\}_{i=1}^M$, where $M \gg N$, such as video or simulation data. This “off-domain” data \mathcal{D}_{off} is different from in-domain data \mathcal{D} in several important ways: 1) Off-domain perceptual observations o_i^o may be considerably different than in-domain perceptual observations o_i , even when the underlying physical state of the system is similar. An illustrative example of this is the marked difference between simulation and

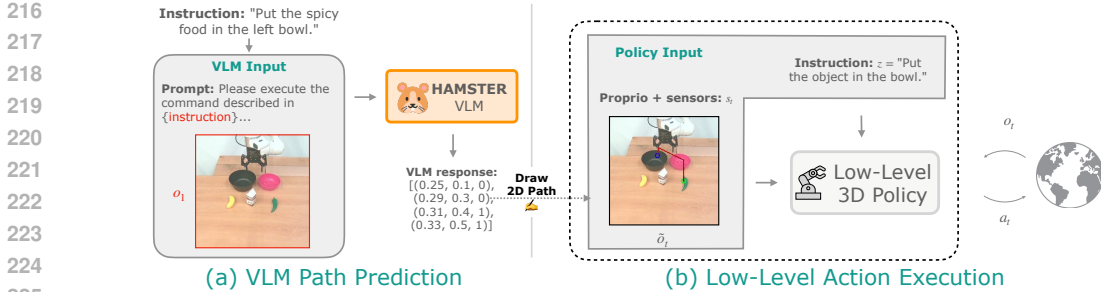


Figure 2: Depiction of HAMSTER’s execution. The high-level VLM is called once to generate the 2D path. The low-level policy is conditioned on the 2D path and interacts with the environment sequentially to execute low-level actions. The path predicted by the VLM enhances the low-level policy generalization capability.

real-world scene appearance (see Figure 6). 2) The underlying physical dynamics of the system can be potentially different, i.e., the transition dynamics may be different between off-domain sources such as video or simulation than the test-time deployment setting. While the dynamics may show level differences, we assume the higher-level coarse strategies to solve the task remain invariant. 3) Off-domain data may not have access directly to actions a or proprioception state s , for instance in video based datasets. This poses challenges to directly applying the standard imitation learning paradigm for these datasets.

The goal is to leverage the combination of a small amount of “expensive” in-domain data \mathcal{D} and a large amount of relatively “cheap” off-domain data \mathcal{D}_{off} to obtain a generalizable policy π_{θ} that can be successfully deployed over various initial conditions, task variations, and visual variations in the in-domain robot environment. Without additional assumptions, this problem is arduous due to the lack of alignment between the in-domain and off-domain settings. In this work, we assume access to an intermediate *path-labeler* $p_i = h(o_i, z_i)$ at training time, that accepts an observation o_i and a language instruction z_i from either the off-domain or in-domain datasets, to produce an intermediate path label p_i that indicates *how* to optimally perform the task z_i from the observation o_i . In this work, we choose this intermediate path label p_i to be a sequence of points, a 2D path, on the image that indicates coarse end-effector motion to solve the designated task. This path-labeler at training time can come from different sources – a projection of known proprioception if available, human-drawn trajectory annotations on images, point-tracked end-effector or hand positions from video, and so on. Applying such a path labeler to the off-domain dataset yields $\tilde{\mathcal{D}}_{\text{off}} = \{(o_i^o, z_i^o, p_i^o)\}_{i=1}^M$.

4.1 HAMSTER’S VLM FOR PRODUCING 2D PATHS TRAINED FROM OFF-DOMAIN DATA

The first stage of building a HAMSTER VLA model is finetuning a high-level VLM that predicts coarse 2D paths p given a language instruction z and observation o . This path represents the approximate trajectory of the robot end-effector on the input camera image. It also contains information about the gripper state (where to open the gripper and where to close it) as subsequently explained.

Although, conceptually, any VLM can be used to predict such a 2D path by casting an appropriate prompt, we find that standard pre-trained VLMs struggle with predicting such a path in a zero-shot manner (see Table 5). Therefore, we finetune pre-trained VLMs on datasets that ground VLMs to robot scenes and path predictions collected from easier-to-obtain sources, i.e., internet visual-question-answering data, robot data from other modalities, and simulation data. The primary advantages of finetuning such a hierarchical VLM that produces intermediate representations as opposed to directly producing actions a with a monolithic model (Kim et al., 2024; Zitkovich et al., 2023) are twofold: 1) the lack of actions in certain off-domain datasets (such as videos) makes it impossible to even train monolithic pixel-to-action models, 2) we find empirically that hierarchical VLMs producing intermediate cross-domain predictions generalize more effectively than monolithic VLA models.

Finetuning Objective and Datasets. We use VILA-1.5-13b (Lin et al., 2024), a 13-billion-parameter vision language model trained on interleaved image-text datasets and video captioning data, as our base VLM. We then curate a multi-domain dataset to finetune this model for effective 2D path prediction. Predicting the 2D path of the end-effector requires understanding *what* objects

to manipulate in a given task in terms of their pixel positions, but also reasoning about *how* a robot should perform the task. To enable this understanding, we collate a diverse off-domain dataset \mathcal{D}_{off} from a wide range of modalities, including real-world data, visual question-answering data, and simulation data. Importantly, *none* of this off-domain data used to train the VLM comes from the deployment environment, thereby emphasizing generalizability. However, as outlined in Section 4.2, the predictions of this trained VLM are used to guide a low-level policy at inference time.

We assemble a dataset $\tilde{\mathcal{D}}_{\text{off}} = \{(o_i^o, z_i^o, p_i^o)\}_{i=1}^M$ of image inputs o_i^o , language prompts z_i^o , and path labels p_i^o consisting of three types of data: (1) pixel point prediction tasks (*what*); (2) simulated robotics tasks (*what and how*); (3) a real robot dataset consisting of trajectories (*what and how*). We detail each dataset below; see Figure 7 for visualization of each dataset’s prompts and labels.

Pixel Point Prediction. For pixel point prediction, we use the dataset released by RoboPoint (Yuan et al., 2024) with 1.4 million VQA tasks, with most answers represented as a list of 2D points corresponding to locations on the image. A sample consists of a prompt z^o like Find all instances of cushions, an input image o^o and labels p^o like $[(0.25, 0.11), (0.22, 0.19), (0.53, 0.23)]$.¹ This dataset consists of data automatically generated in simulation and collected from existing real-world datasets; its diversity and tasks enable the HAMSTER VLM to reason about pixel-object relationships across diverse scenes while retaining its semantic generalization capabilities.

Robot Simulation Data. We additionally generate a dataset of simulated robotics tasks from RL-Bench (James et al., 2020), a simulator of a Franka robot performing tabletop manipulation for a wide array of both prehensile and non-prehensile tasks. We use the simulator’s built-in planning algorithms to automatically generate successful manipulation trajectories and construct ground-truth 2D path labels p^o . Each trajectory contains a sequence of 3D coordinates of the robot’s gripper in world space, as well as whether the gripper is open or closed at a given time step. We use known camera intrinsics and extrinsics to project these points on the front image and construct labels $p^o = [(x_{\text{image}}, y_{\text{image}}, \text{gripper_open}), \dots]$ where $x_{\text{image}}, y_{\text{image}} \in [0, 1]$ are *relative pixel locations* of the end effector’s position on the image. The front camera image of the initial state forms the image input o^o and the prompt z^o for the VLM is to provide a sequence of points denoting the trajectory of the robot gripper to achieve the given instruction (see Figure 2) We generate 1000 episodes for each of 79 robot manipulation tasks in RL-Bench, each episode with ~ 4 language instructions, for a total of $\sim 300\text{k}$ (o^o, z^o, p^o) tuples for $\tilde{\mathcal{D}}_{\text{off}}$.

Real Robot Data. Using real robot data allows us to ensure the VLM can reason about objects and robot gripper paths when conditioned on scenes, including real robot arms. We use existing, online robot datasets *not from the deployment environment* to enable this VLM ability. We source 10k trajectories from the Bridge dataset (Walke et al., 2023; Collaboration et al., 2023) consisting of a WidowX arm performing manipulation tasks and 45k trajectories from DROID (Khazatsky et al., 2024). For both datasets, we use the given end-effector trajectories and given (or estimated) camera matrices to convert robot gripper trajectories to 2D paths p^o . We use a camera image from the first timestep of each robot trajectory as o^o and a similar text prompt z^o as the simulation dataset. Note that we essentially utilize the robot data as video data, where the end effector is tracked over time. In principle, this could be done with any number of point-tracking methods (Doersch et al., 2023) on raw video as well, with no action or proprioceptive labels.

VLM Training. We finetune the HAMSTER VLM on all three datasets by randomly sampling from all samples in the entire dataset with equal weight. One problem with directly training on the path labels p^o is that many paths may be extremely long, e.g., exceeding one hundred points. Since we want the HAMSTER VLM to reason at a *high level* instead of on the same scale as the low-level control policy. Therefore, we simplify the paths p^o with the Ramer-Douglas-Peucker algorithm (Ramer, 1972; Douglas & Peucker, 1973) that reduces curves composed of line segments to similar curves composed of fewer points. We train with the standardized supervised prediction loss to maximize the log-likelihood of the language labels p^o : $\mathbb{E}_{(o_i^o, z_i^o, p_i^o) \sim \tilde{\mathcal{D}}_{\text{off}}} \log \text{VLM}(p_i^o | z_i^o, o_i^o)$.

¹Note that this is not a temporally ordered path, but rather simply a set of unordered points of interest in an image. We overload notation here for the sake of notational convenience.

4.2 PATH GUIDED LOW-LEVEL POLICY LEARNING

After training the HAMSTER VLM to predict paths, we train a low-level policy to utilize these paths to predict actions. While a low-level control policy *can* learn to solve the task without access to 2D path predictions, providing it with 2D paths can make the task easier. The paths allow the low-level policy to forgo long-horizon and semantic reasoning and focus on local and geometric predictions to produce low-level actions. As we find empirically (see Figure. 3), 2D paths allow for considerably improved visual and semantic generalization of low-level policies. We train low-level policies based on rich 3-D perceptual information, available at test time on a robotic platform with standard depth cameras. Then the question becomes—how do we incorporate 2D path information \hat{p} produced by the VLM in Section 4.1 onto the 3D inputs to enable generalizable robot manipulation?

Conditioning on Paths. We convert 2D paths of the form $p = \{(x_i, y_i, \text{gripper_open})\}_{i=1}^L$ into a format that is easy to incorporate into any language (z), proprioception (s), and image (o) conditioned policy $\pi_\theta(a | s, o, z)$. While one could concatenate the path with the proprioception or language input, paths are of varied lengths, and this could prevent the integration of such paths into existing policy architectures that cannot take in varied proprioceptive or language inputs. Instead, we directly draw the 2D path points onto the image input to the policy, which is not only generalizable across policy architectures but also may provide easier-to-follow path guidance as the policy does not have to learn how to associate path points with their corresponding image locations (Gu et al., 2023). During training, we use oracle paths constructed by projecting end-effector points to the camera plane as described for simulation and real robot data in Section 4.1.

Formally, we iterate through each trajectory $\tau_i = \{s_i^t, a_i^t, o_i^t, z_i\}_{t=1}^T$ on the in-domain dataset \mathcal{D} to obtain the path p_i . Gu et al. (2023) proposed using colored trajectories to guide a policy’s actions, and we largely follow their method of coloring trajectories to indicate gripper status and progression through time. These paths are drawn onto all images in the trajectory $o_i^1 \dots o_i^T$ by drawing points at each (x, y) and connecting them with line segments to obtain $\{\tilde{o}_i^t\}_{t=1}^T$. We use a color gradient to indicate progression through time (see Figure 2(b) for an example). We plot circles for change in gripper status: e.g., **green** for closing the gripper and **blue** for opening. This constructs the final in-domain path-labeled dataset $\mathcal{D}_{\text{path}} = \{(s_i, a_i, \tilde{o}_i, z_i)\}_{i=1}^N$.

Imitation Learning. Finally, we train a policy $\pi_\theta(a | s, \tilde{o}, z)$ conditioned on proprioception and other sensor information s , path-annotated image observations \tilde{o} , and a task language instruction z on $\mathcal{D}_{\text{path}}$. HAMSTER’s general path-conditioning framework allows for using arbitrary lower-level control policies as they do not need to condition on the same inputs as the VLM. Therefore, we train 3D low-level policies, such as RVT-2 (Goyal et al., 2024) and 3D-DA (Ke et al., 2024), for low-level control. Here, we assume s includes additional sensor information (i.e., depth), which 3D-DA and RVT-2 utilize to construct point clouds and virtual camera renderings, respectively, for more accurate control and data-efficient imitation learning. We directly train these policies, with no necessary major architectural modifications, with their supervised imitation learning objectives on $\mathcal{D}_{\text{path}}$ to maximize log-likelihoods of the dataset actions: $\mathbb{E}_{(s_t, a_t, \tilde{o}_t, z_t) \sim \mathcal{D}_{\text{path}}} \log \pi_\theta(a | s_t, \tilde{o}_t, z_t)$. For further implementation details, see Appendix B.

Online Evaluation. Standard VLA architectures query the VLM for every low-level action (Kim et al., 2024; Brohan et al., 2023a), which can be very expensive with large VLMs—for example, OpenVLA’s 7B param VLA only runs at 6Hz on an RTX 4090 (Kim et al., 2024). Instead, HAMSTER’s hierarchical design allows us to query the VLM just once at the beginning of the episode to generate a 2D path \hat{l} that we draw onto every subsequent image.² Therefore, HAMSTER can be scaled to large VLM backbones without needing end-users to be concerned about inference speed.

5 EXPERIMENTAL EVALUATION

To test the hypotheses proposed in Section 4, we perform empirical evaluations in both simulation and the real world. The experiments primarily aim to answer the following questions: (1) do hierarchical VLA models enable behavioral generalization to unseen scenarios? (2) do hierarchical VLA models show more effective cross-domain generalization than monolithic VLA models or low-level

²HAMSTER is not inherently limited to being queried once per episode, but for simplicity and computational efficiency we query just once per episode in our experiments.

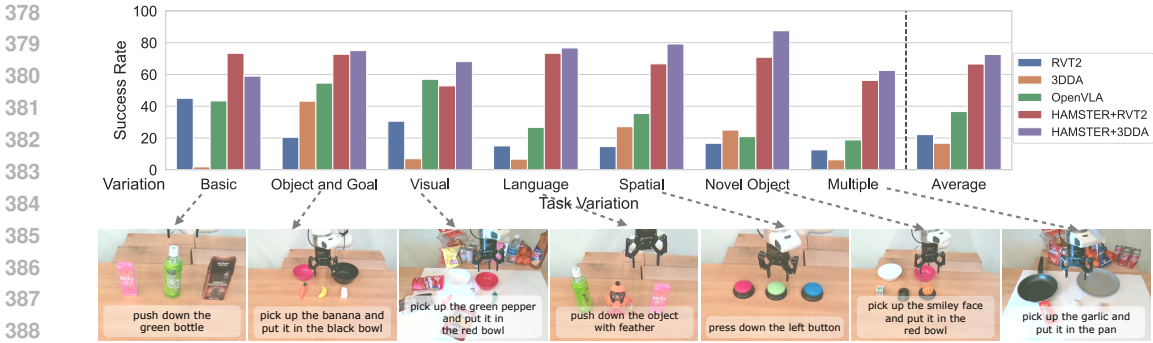


Figure 3: Depiction of quantitative real-world policy execution results on a real-world robot, evaluated across different axes of generalization and across both prehensile and non-prehensile tasks. Across all generalization axes, HAMSTER outperforms monolithic VLAs and the base 3D imitation learning policies.

imitation learning methods? (3) is behavior learned by hierarchical VLA models robust to significant degrees of visual and semantic variations? (4) does including cross-domain data from settings like simulation really help with model generalization? (5) does explicitly finetuning the high-level VLM yield benefits in terms of spatial and semantic reasoning?

5.1 REAL WORLD EVALUATION ON TABLETOP MANIPULATION

Our real-world evaluation experiments aim to test the generalization capability of hierarchical VLA models across significant semantic and visual variations. In particular, we consider a variant of HAMSTER that uses a VLM (VILA-1.5-13b) finetuned on the data mixture in Section 4.1 as the high-level predictor, with two 3D policy architectures - RVT-2 (Goyal et al., 2024) and 3D Diffuser Actor (3D-DA) (Ke et al., 2024) as the choice of low-level policy, as described in Section 4.2. The low-level 3D policies are trained with 320 episodes collected via teleoperation directly on the table-top manipulation setup shown in Fig. 7. Importantly, the high-level VLM in HAMSTER is not finetuned on any in-domain data and is directly transferred only from the cheap data sources described in Section 4.1. This suggests that any generalization that the VLM sees does not result from in-domain training data rather than from cross-domain transfer.

Baseline comparisons. We compare HAMSTER to a state-of-the-art monolithic VLA, OpenVLA (Kim et al., 2024), as well as a non-VLM 3D imitation learning policies. For fair comparison, we finetune OpenVLA on the collected in-domain trajectory data described above since OpenVLA showed poor zero-shot generalization. The 3D imitation learning policy (RVT-2, 3D-DA) baselines are trained with the same teleoperation data used to train the low-level policy in HAMSTER but without the intermediate 2D path representation from HAMSTER’s VLM.

Results. Figure 3 summarizes our real-world results. We compile results for multiple task types, including ‘pick and place,’ and nonprehensile tasks such as ‘push buttons’ and ‘knock down objects.’ Similar to prior work (Kim et al., 2024), we test generalization across various axes: *obj and goal*: unseen object-goal combinations; *visual*: visual changes in table texture, lighting, distractor objects; *language*: unseen language instructions (e.g., candy → sweet object); *spatial*: unseen spatial object relationships in the instruction; *novel object*: unseen objects; and lastly, *multiple*: a combination of multiple variations. In total, we evaluate each model on 74 tasks for 222 total evaluations.

We find that HAMSTER significantly outperforms monolithic VLA models and 3D imitation learning methods by over **2x** and **3x**, respectively, on average. This is significant because this improved performance is in the face of considerable visual and semantic changes in the test setting, showing the ability of HAMSTER to transfer much more effectively than monolithic VLA models or non-VLM base models. We further group results by task type in Table 6, where we see HAMSTER outperforms OpenVLA across all task types (pick and place, press button, and knock down). See Appendix C for evaluation conditions, a task list, and other experiment details, and Appendix E for failure modes.

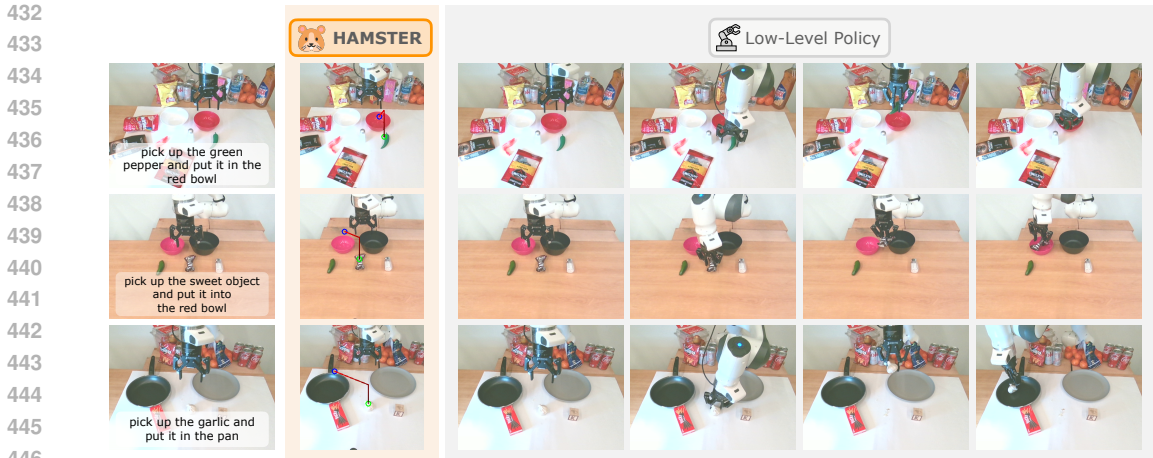


Figure 4: Example real-world HAMSTER rollouts demonstrate its strong performance in novel scenes achieved by leveraging VLMs’ generalization capabilities and the robust execution of low-level 3D policies.

	Avg.	no var	bac tex	cam pos	distractor	lig col	man obj col	man obj siz
3D-DA[Ke et al.]	0.35 ± 0.04	0.43 ± 0.06	0.34 ± 0.07	0.35 ± 0.11	0.39 ± 0.11	0.44 ± 0.13	0.41 ± 0.04	0.41 ± 0.11
HAMSTER (w 3D-DA)	0.46 ± 0.04	0.57 ± 0.03	0.48 ± 0.08	0.39 ± 0.06	0.41 ± 0.05	0.59 ± 0.04	0.57 ± 0.08	0.51 ± 0.10
	man obj tex	rec obj col	rec obj siz	rec obj tex	rlb and col	rlb var	tab col	tab tex
3D-DA[Ke et al.]	0.27 ± 0.04	0.34 ± 0.10	0.36 ± 0.05	0.36 ± 0.12	0.07 ± 0.03	0.45 ± 0.12	0.42 ± 0.06	0.23 ± 0.04
HAMSTER (w 3D-DA)	0.48 ± 0.06	0.48 ± 0.05	0.40 ± 0.05	0.56 ± 0.09	0.11 ± 0.10	0.58 ± 0.04	0.56 ± 0.03	0.35 ± 0.07

Table 1: Simulation evaluation of HAMSTER across different visual variations. We test vanilla 3D Diffuser Actor and HAMSTER across variations in Colosseum (Pumacay et al., 2024) and find that HAMSTER generalizes more effectively than 3D Diffuser Actor. Avg. indicates mean across variations, including no variation.

5.2 SIMULATION EVALUATION

We also perform controlled experiments in simulation. We use Colosseum (Pumacay et al., 2024) as the benchmark as it displays considerable visual and semantic variations. In simulation, we paired our high-level VLM with 3D Diffuser Actor (Ke et al., 2024) as the low-level policy, since this is one of the state-of-the-art models on RL Bench. We compare HAMSTER with a vanilla 3D Diffuser Actor implementation without path guidance. Table 1 summarizes our results in simulation across 5 seeds. HAMSTER significantly outperforms vanilla 3D-DA by 31%. This shows that the 2D paths produced by the VLM in HAMSTER can help low-level policies to generalize better to novel unseen variations. We refer readers to Pumacay et al. (2024) for details on the variations and Appendix F for further simulation experiment details.

5.3 GENERALIZATION AND ABLATION STUDIES

Finally, we perform additional experiments testing HAMSTER’s ability to generalize to novel views, various ways to represent the paths, and finally, the demonstration efficiency of HAMSTER.

View Invariance and Path Representation. We test camera view invariance with a new camera angle, as pictured in Figure 5, by evaluating HAMSTER+RVT2 against OpenVLA on the new camera angle across 10 separate pick and place task trials with 6 training objects and 3 training containers. Additionally, we also compare HAMSTER+RVT2 (Concat), where instead of drawing the path onto the RGB image given as input to RVT2, we modify RVT2 to accept a 6-channel input image consisting of the original RGB image concatenated with a second RGB image that only contains the drawn path. This approach is less easily applied to arbitrary imitation learning policies (for example, it cannot be easily applied to 3D-DA as it uses a pre-trained CLIP image encoder expecting 3 input channels), but allows us to represent paths in a different way.



Figure 5: The camera angle invariance setup: old camera on the right, new camera angle on the left.

Method	Original Camera		Novel Camera	
	Cumulative Score	# Success	Cumulative Score	# Success
OpenVLA	6	3	2.25	0
HAMSTER+RVT2	8.25	7	7.25	4
HAMSTER+RVT2 (Concat)	10	10	9.75	9

Table 2: Real world results comparing HAMSTER, HAMSTER where paths are concatenated with RGB instead of drawn onto the image, and OpenVLA on a setup with the new camera angle as shown in Figure 5. We report cumulative completion scores out of 10 (10 trials) and total number of fully successful executions. HAMSTER performs better with both path drawing settings.

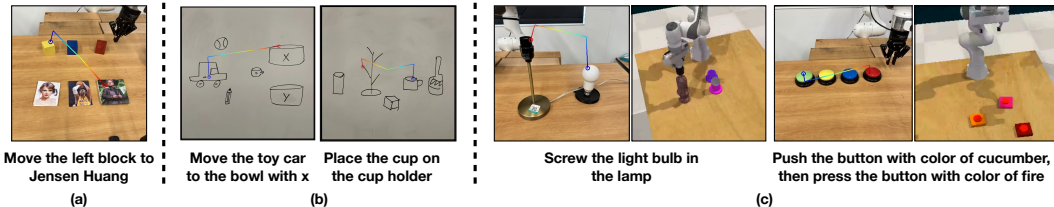


Figure 6: HAMSTER’s VLM demonstrates considerable generalization and cross-domain learning to scenarios not encountered in the training set. From left to right: (a) it can effectively utilize world knowledge to generalize to tasks specified by people; (b) it generalizes to highly out-of-domain input images, such as human-drawn sketches; (c) when trained on diverse simulated data it shows transfer to related, but visually distinct tasks in the real world.

The results in Table 2 demonstrate that HAMSTER far outperforms OpenVLA and is generally robust to a new camera angle. HAMSTER with concatenated image paths performs the best, which demonstrates this other path representation can work well with RVT2, although it is less general and cannot be easily integrated with 3D-DA.

HAMSTER with Fewer Demonstrations. Finally, we also test HAMSTER’s ability to work well with limited demonstrations. We test on a subset of 5 Colosseum tasks, namely, SLIDE_BLOCK_TO_TARGET, PLACE_WINE_AT_RACK_LOCATION, INSERT_ONTO_SQUARE_PEG, STACK_CUPS, SETUP_CHESS. Results in Table 3 demonstrate that HAMSTER+3D-DA with just 50% of the data still achieves 2x the success rate of standard 3D-DA with HAMSTER is demonstration-efficient for the demonstrator imitation learning tasks.

Method	Success
3D-DA	0.18 ± 0.10
HAMSTER+3D-DA (50%)	0.36 ± 0.04
HAMSTER+3D-DA	0.43 ± 0.05

Table 3: Colosseum results demonstrate that HAMSTER is demonstration-efficient, doubling 3D-DA’s success rate even with just 50% of the data.

Finally, we visualize example HAMSTER path drawings in Figure 6, demonstrating HAMSTER effectively generalizes to new tasks. We further investigate design decisions on VLM performance in Appendix D.1, where we find that (1) HAMSTER outperforms zero-shot path generation from closed-source VLMs (Gu et al., 2023; Liang et al., 2023) and (2) that inclusion of simulation data improves HAMSTER’s real-world performance. See Appendix D.1 for further details.

6 CONCLUSION AND LIMITATIONS

In summary, HAMSTER studies the potential of hierarchical VLA models, achieving robust generalization in robotic manipulation. It consists of a finetuned VLM that accurately predicts 2D paths for robotic manipulation and a low-level policy that learns to generate actions using the 2D paths. This two-step architecture enables visual generalization and semantic reasoning across considerable domain shifts, while enabling data-efficient specialist policies, like ones conditioned on 3D inputs, to perform low-level action execution.

This work represents an initial step towards developing versatile, hierarchical VLA methods, with numerous opportunities for future improvement and expansion. The proposed work only generates points in 2D space, without making native 3D predictions. This prevents the VLM from having true spatial 3D understanding. Moreover, the interface of just using 2D paths is a bandwidth limited one, which cannot communicate nuances such as force or rotation. In the future, investigating learnable intermediate interfaces is a promising direction. Moreover, training these VLMs directly from large-scale human video datasets would also be promising.

REFERENCES

- 540
541
542 OpenAI Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni
543 Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor
544 Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mo Bavarian, Jeff Bel-
545 gum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bog-
546 donoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles
547 Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea
548 Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen,
549 Ruby Chen, Jason Chen, Mark Chen, Benjamin Chess, Chester Cho, Casey Chu, Hyung Won
550 Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah
551 Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien
552 Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Sim'on Posada Fish-
553 man, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun
554 Gogineni, Gabriel Goh, Raphael Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray,
555 Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Har-
556 ris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter
557 Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain,
558 Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto,
559 Billie Jonn, Heewoo Jun, Tomer Kaftan, Lukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Ni-
560 tish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik
561 Kim, Hendrik Kirchner, Jamie Ryan Kiros, Matthew Knight, Daniel Kokotajlo, Lukasz Kon-
562 draciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo,
563 Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li,
564 Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia
565 Lue, Anna Adeola Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski,
566 Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine
567 McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke
568 Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel P. Mossing,
569 Tong Mu, Mira Murati, Oleg Murk, David M'ely, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak,
570 Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Ouyang Long, Cullen O'Keefe, Jakub W.
571 Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish,
572 Emy Parparita, Alexandre Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila
573 Belbute Peres, Michael Petrov, Henrique Pondé de Oliveira Pinto, Michael Pokorny, Michelle
574 Pokrass, Vitchyr H. Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul
575 Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rim-
576 bach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario D. Saltarelli, Ted Sanders,
577 Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Sel-
578 sam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor,
579 Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin D. Sokolowsky,
580 Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang,
581 Nikolas A. Tezak, Madeleine Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Pre-
582 ston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cer'on Uribe, Andrea Vallone, Arun Vi-
583 jayvergiya, Chelsea Voss, Carroll L. Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang,
584 Jonathan Ward, Jason Wei, CJ Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian
585 Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren
586 Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming
587 Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tian-
588 hao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. Gpt-4 technical report. In *arxiv
589 preprint*, 2023. URL <https://arxiv.org/pdf/2303.08774>.
- 586 Jinze Bai, Shuai Bai, Shusheng Yang, Shijie Wang, Sinan Tan, Peng Wang, Junyang Lin, Chang
587 Zhou, and Jingren Zhou. Qwen-vl: A frontier large vision-language model with versatile abilities.
588 *arXiv preprint arXiv:2308.12966*, 2023.
- 589 Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn,
590 Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, et al. Rt-1: Robotics
591 transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022.
- 592 Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choro-
593 manski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, Pete Florence, Chuyuan Fu,

- 594 Montse Gonzalez Arenas, Keerthana Gopalakrishnan, Kehang Han, Karol Hausman, Alex Herzog,
595 Jasmine Hsu, Brian Ichter, Alex Irpan, Nikhil Joshi, Ryan Julian, Dmitry Kalashnikov,
596 Yuheng Kuang, Isabel Leal, Lisa Lee, Tsang-Wei Edward Lee, Sergey Levine, Yao Lu, Henryk
597 Michalewski, Igor Mordatch, Karl Pertsch, Kanishka Rao, Krista Reymann, Michael Ryoo,
598 Grecia Salazar, Pannag Sanketi, Pierre Sermanet, Jaspiar Singh, Anikait Singh, Radu Soricut,
599 Huang Tran, Vincent Vanhoucke, Quan Vuong, Ayzaan Wahid, Stefan Welker, Paul Wohlhart,
600 Jialin Wu, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. Rt-
601 2: Vision-language-action models transfer web knowledge to robotic control. In *arXiv preprint*
602 *arXiv:2307.15818*, 2023a.
- 603 Anthony Brohan, Yevgen Chebotar, Chelsea Finn, Karol Hausman, Alexander Herzog, Daniel Ho,
604 Julian Ibarz, Alex Irpan, Eric Jang, Ryan Julian, et al. Do as i can, not as i say: Grounding
605 language in robotic affordances. In *Conference on robot learning*, pp. 287–318. PMLR, 2023b.
- 606
607 Minwoo Byeon, Beomhee Park, Haecheon Kim, Sungjun Lee, Woonhyuk Baek, and Saehoon
608 Kim. Coyo-700m: Image-text pair dataset. [https://github.com/kakaobrain/
609 coyo-dataset](https://github.com/kakaobrain/coyo-dataset), 2022.
- 610 Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran
611 Song. Diffusion policy: Visuomotor policy learning via action diffusion. In Kostas E. Bekris,
612 Kris Hauser, Sylvia L. Herbert, and Jingjin Yu (eds.), *Robotics: Science and Systems XIX, Daegu,*
613 *Republic of Korea, July 10-14, 2023*, 2023. doi: 10.15607/RSS.2023.XIX.026. URL <https://doi.org/10.15607/RSS.2023.XIX.026>.
- 614
615
616 Open X-Embodiment Collaboration, Abby O’Neill, Abdul Rehman, Abhinav Gupta, Abhiram Mad-
617 dukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay
618 Mandlekar, Ajinkya Jain, Albert Tung, Alex Bewley, Alex Herzog, Alex Irpan, Alexander Khaz-
619 atsky, Anant Rai, Anchit Gupta, Andrew Wang, Andrey Kolobov, Anikait Singh, Animesh Garg,
620 Aniruddha Kembhavi, Annie Xie, Anthony Brohan, Antonin Raffin, Archit Sharma, Arefeh
621 Yavary, Arhan Jain, Ashwin Balakrishna, Ayzaan Wahid, Ben Burgess-Limerick, Beomjoon Kim,
622 Bernhard Schölkopf, Blake Wulfe, Brian Ichter, Cewu Lu, Charles Xu, Charlotte Le, Chelsea
623 Finn, Chen Wang, Chenfeng Xu, Cheng Chi, Chenguang Huang, Christine Chan, Christopher
624 Chia, Chuer Pan, Chuyuan Fu, Coline Devin, Danfei Xu, Daniel Morton, Danny Driess, Daphne
625 Chen, Deepak Pathak, Dhruv Shah, Dieter Büchler, Dinesh Jayaraman, Dmitry Kalashnikov,
626 Dorsa Sadigh, Edward Johns, Ethan Foster, Fangchen Liu, Federico Ceola, Fei Xia, Feiyu Zhao,
627 Felipe Vieira Frujeri, Freek Stulp, Gaoyue Zhou, Gaurav S. Sukhatme, Gautam Salhotra, Ge Yan,
628 Gilbert Feng, Giulio Schiavi, Glen Berseth, Gregory Kahn, Guangwen Yang, Guanzhi Wang, Hao
629 Su, Hao-Shu Fang, Haochen Shi, Henghui Bao, Heni Ben Amor, Henrik I Christensen, Hiroki
630 Furuta, Homanga Bharadhwaj, Homer Walke, Hongjie Fang, Huy Ha, Igor Mordatch, Ilija Ra-
631 dosavovic, Isabel Leal, Jacky Liang, Jad Abou-Chakra, Jaehyung Kim, Jaimyn Drake, Jan Peters,
632 Jan Schneider, Jasmine Hsu, Jay Vakil, Jeannette Bohg, Jeffrey Bingham, Jeffrey Wu, Jensen
633 Gao, Jiaheng Hu, Jiajun Wu, Jialin Wu, Jiankai Sun, Jianlan Luo, Jiayuan Gu, Jie Tan, Jihoon
634 Oh, Jimmy Wu, Jingpei Lu, Jingyun Yang, Jitendra Malik, João Silvério, Joey Hejna, Jonathan
635 Booher, Jonathan Tompson, Jonathan Yang, Jordi Salvador, Joseph J. Lim, Junhyek Han, Kaiyuan
636 Wang, Kanishka Rao, Karl Pertsch, Karol Hausman, Keegan Go, Keerthana Gopalakrishnan, Ken
637 Goldberg, Kendra Byrne, Kenneth Oslund, Kento Kawaharazuka, Kevin Black, Kevin Lin, Kevin
638 Zhang, Kiana Ehsani, Kiran Lekkala, Kirsty Ellis, Krishan Rana, Krishnan Srinivasan, Kuan
639 Fang, Kunal Pratap Singh, Kuo-Hao Zeng, Kyle Hatch, Kyle Hsu, Laurent Itti, Lawrence Yun-
640 liang Chen, Lerrel Pinto, Li Fei-Fei, Liam Tan, Linxi ”Jim” Fan, Lionel Ott, Lisa Lee, Luca
641 Weihs, Magnum Chen, Marion Lepert, Marius Memmel, Masayoshi Tomizuka, Masha Itkina,
642 Mateo Guaman Castro, Max Spero, Maximilian Du, Michael Ahn, Michael C. Yip, Mingtong
643 Zhang, Mingyu Ding, Minh Ho, Mohan Kumar Srirama, Mohit Sharma, Moo Jin Kim, Naoaki
644 Kanazawa, Nicklas Hansen, Nicolas Heess, Nikhil J Joshi, Niko Suenderhauf, Ning Liu, Nor-
645 man Di Palo, Nur Muhammad Mahi Shafiullah, Oier Mees, Oliver Kroemer, Osbert Bastani,
646 Pannag R Sanketi, Patrick ”Tree” Miller, Patrick Yin, Paul Wohlhart, Peng Xu, Peter David
647 Fagan, Peter Mitrano, Pierre Sermanet, Pieter Abbeel, Priya Sundareshan, Qiuyu Chen, Quan
648 Vuong, Rafael Rafailov, Ran Tian, Ria Doshi, Roberto Mart’in-Mart’in, Rohan Bajjal, Rosario
649 Scalise, Rose Hendrix, Roy Lin, Runjia Qian, Ruohan Zhang, Russell Mendonca, Rutav Shah,
650 Ryan Hoque, Ryan Julian, Samuel Bustamante, Sean Kirmani, Sergey Levine, Shan Lin, Sherry
651 Moore, Shikhar Bahl, Shivin Dass, Shubham Sonawani, Shubham Tulsiani, Shuran Song, Sichun

- 648 Xu, Siddhant Haldar, Siddharth Karamcheti, Simeon Adebola, Simon Guist, Soroush Nasiriany,
649 Stefan Schaal, Stefan Welker, Stephen Tian, Subramanian Ramamoorthy, Sudeep Dasari, Suneel
650 Belkhale, Sungjae Park, Suraj Nair, Suvir Mirchandani, Takayuki Osa, Tanmay Gupta, Tatsuya
651 Harada, Tatsuya Matsushima, Ted Xiao, Thomas Kollar, Tianhe Yu, Tianli Ding, Todor Davchev,
652 Tony Z. Zhao, Travis Armstrong, Trevor Darrell, Trinity Chung, Vidhi Jain, Vikash Kumar, Vin-
653 cent Vanhoucke, Wei Zhan, Wenxuan Zhou, Wolfram Burgard, Xi Chen, Xiangyu Chen, Xiaolong
654 Wang, Xinghao Zhu, Xinyang Geng, Xiyuan Liu, Xu Liangwei, Xuanlin Li, Yansong Pang, Yao
655 Lu, Yecheng Jason Ma, Yejin Kim, Yevgen Chebotar, Yifan Zhou, Yifeng Zhu, Yilin Wu, Ying
656 Xu, Yixuan Wang, Yonatan Bisk, Yongqiang Dou, Yoonyoung Cho, Youngwoon Lee, Yuchen
657 Cui, Yue Cao, Yueh-Hua Wu, Yujin Tang, Yuke Zhu, Yunchu Zhang, Yunfan Jiang, Yunshuang
658 Li, Yunzhu Li, Yusuke Iwasawa, Yutaka Matsuo, Zehan Ma, Zhuo Xu, Zichen Jeff Cui, Zichen
659 Zhang, Zipeng Fu, and Zipeng Lin. Open X-Embodiment: Robotic learning datasets and RT-X
660 models. <https://arxiv.org/abs/2310.08864>, 2023.
- 661 Carl Doersch, Yi Yang, Mel Vecerik, Dilara Gokay, Ankush Gupta, Yusuf Aytar, Joao Carreira,
662 and Andrew Zisserman. Tapir: Tracking any point with per-frame initialization and temporal
663 refinement. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp.
664 10061–10072, 2023.
- 665 David H Douglas and Thomas K Peucker. Algorithms for the reduction of the number of
666 points required to represent a digitized line or its caricature. *Cartographica*, 10(2):112–
667 122, 1973. doi: 10.3138/FM57-6770-U75U-7727. URL [https://doi.org/10.3138/
668 FM57-6770-U75U-7727](https://doi.org/10.3138/FM57-6770-U75U-7727).
- 669 Danny Driess, Fei Xia, Mehdi SM Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter,
670 Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, et al. Palm-e: An embodied mul-
671 timodal language model. In *International Conference on Machine Learning*, pp. 8469–8488.
672 PMLR, 2023.
- 673 Adam Fishman, Adithyavairavan Murali, Clemens Eppner, Bryan Peele, Byron Boots, and Dieter
674 Fox. Motion policy networks. In Karen Liu, Dana Kulic, and Jeffrey Ichnowski (eds.), *Conference
675 on Robot Learning, CoRL 2022, 14-18 December 2022, Auckland, New Zealand*, volume 205
676 of *Proceedings of Machine Learning Research*, pp. 967–977. PMLR, 2022. URL [https://
677 proceedings.mlr.press/v205/fishman23a.html](https://proceedings.mlr.press/v205/fishman23a.html).
- 678 Ankit Goyal, Jie Xu, Yijie Guo, Valts Blukis, Yu-Wei Chao, and Dieter Fox. Rvt: Robotic view
679 transformer for 3d object manipulation. In *Conference on Robot Learning*, pp. 694–710. PMLR,
680 2023.
- 681 Ankit Goyal, Valts Blukis, Jie Xu, Yijie Guo, Yu-Wei Chao, and Dieter Fox. Rvt2: Learning precise
682 manipulation from few demonstrations. *RSS*, 2024.
- 683 Jiayuan Gu, Sean Kirmani, Paul Wohlhart, Yao Lu, Montserrat Gonzalez Arenas, Kanishka Rao,
684 Wenhao Yu, Chuyuan Fu, Keerthana Gopalakrishnan, Zhuo Xu, Priya Sundaesan, Peng Xu,
685 Hao Su, Karol Hausman, Chelsea Finn, Quan Vuong, and Ted Xiao. Rt-trajectory: Robotic task
686 generalization via hindsight trajectory sketches, 2023.
- 687 Ankur Handa, Arthur Allshire, Viktor Makoviychuk, Aleksei Petrenko, Ritvik Singh, Jingzhou Liu,
688 Denys Makoviichuk, Karl Van Wyk, Alexander Zhurkevich, Balakumar Sundaralingam, et al.
689 Dextreme: Transfer of agile in-hand manipulation from simulation to reality. In *2023 IEEE
690 International Conference on Robotics and Automation (ICRA)*, pp. 5977–5984. IEEE, 2023.
- 691 Wenlong Huang, Pieter Abbeel, Deepak Pathak, and Igor Mordatch. Language models as zero-shot
692 planners: Extracting actionable knowledge for embodied agents. In *International conference on
693 machine learning*, pp. 9118–9147. PMLR, 2022.
- 694 Wenlong Huang, Chen Wang, Ruohan Zhang, Yunzhu Li, Jiajun Wu, and Li Fei-Fei. Voxposer:
695 Composable 3d value maps for robotic manipulation with language models. In *Conference on
696 Robot Learning*, pp. 540–562. PMLR, 2023a.
- 697 Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan
698 Tompson, Igor Mordatch, Yevgen Chebotar, et al. Inner monologue: Embodied reasoning through
699
700
701

- 702 planning with language models. In *Conference on Robot Learning*, pp. 1769–1782. PMLR,
703 2023b.
- 704
- 705 Stephen James, Zicong Ma, David Rovick Arrojo, and Andrew J Davison. Rlbench: The robot
706 learning benchmark & learning environment. *IEEE Robotics and Automation Letters*, 5(2):3019–
707 3026, 2020.
- 708 Eric Jang, Alex Irpan, Mohi Khansari, Daniel Kappler, Frederik Ebert, Corey Lynch, Sergey Levine,
709 and Chelsea Finn. Bc-z: Zero-shot task generalization with robotic imitation learning. In *Confer-*
710 *ence on Robot Learning*, pp. 991–1002. PMLR, 2022.
- 711
- 712 Yunfan Jiang, Agrim Gupta, Zichen Zhang, Guanzhi Wang, Yongqiang Dou, Yanjun Chen, Li Fei-
713 Fei, Anima Anandkumar, Yuke Zhu, and Linxi Fan. Vima: General robot manipulation with
714 multimodal prompts. In *International Conference on Machine Learning*, 2023.
- 715
- 716 Tsung-Wei Ke, Nikolaos Gkanatsios, and Katerina Fragkiadaki. 3d diffuser actor: Policy diffusion
717 with 3d scene representations. In *First Workshop on Vision-Language Models for Navigation and*
718 *Manipulation at ICRA 2024*, 2024.
- 719 Alexander Khazatsky, Karl Pertsch, Suraj Nair, Ashwin Balakrishna, Sudeep Dasari, Siddharth
720 Karamcheti, Soroush Nasiriany, Mohan Kumar Srirama, Lawrence Yunliang Chen, Kirsty Ellis,
721 Peter David Fagan, Joey Hejna, Masha Itkina, Marion Lepert, Yecheng Jason Ma, Patrick Tree
722 Miller, Jimmy Wu, Suneel Belkhale, Shivin Dass, Huy Ha, Arhan Jain, Abraham Lee, Young-
723 woon Lee, Marius Memmel, Sungjae Park, Ilija Radosavovic, Kaiyuan Wang, Albert Zhan, Kevin
724 Black, Cheng Chi, Kyle Beltran Hatch, Shan Lin, Jingpei Lu, Jean Mercat, Abdul Rehman, Pan-
725 nag R Sanketi, Archit Sharma, Cody Simpson, Quan Vuong, Homer Rich Walke, Blake Wulfe,
726 Ted Xiao, Jonathan Heewon Yang, Arefeh Yavary, Tony Z. Zhao, Christopher Agia, Rohan Bai-
727 jal, Mateo Guaman Castro, Daphne Chen, Qiuyu Chen, Trinity Chung, Jaimyn Drake, Ethan Paul
728 Foster, Jensen Gao, David Antonio Herrera, Minh Heo, Kyle Hsu, Jiaheng Hu, Donovan Jack-
729 son, Charlotte Le, Yunshuang Li, Kevin Lin, Roy Lin, Zehan Ma, Abhiram Maddukuri, Suvir Mir-
730 chandani, Daniel Morton, Tony Nguyen, Abigail O’Neill, Rosario Scalise, Derick Seale, Victor
731 Son, Stephen Tian, Emi Tran, Andrew E. Wang, Yilin Wu, Annie Xie, Jingyun Yang, Patrick Yin,
732 Yunchu Zhang, Osbert Bastani, Glen Berseth, Jeannette Bohg, Ken Goldberg, Abhinav Gupta,
733 Abhishek Gupta, Dinesh Jayaraman, Joseph J Lim, Jitendra Malik, Roberto Martín-Martín, Sub-
734 ramanian Ramamoorthy, Dorsa Sadigh, Shuran Song, Jiajun Wu, Michael C. Yip, Yuke Zhu,
735 Thomas Kollar, Sergey Levine, and Chelsea Finn. Droid: A large-scale in-the-wild robot manip-
736 ulation dataset. 2024.
- 737
- 738 Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair,
739 Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. Openvla: An open-source
740 vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.
- 741
- 742 Minae Kwon, Sang Michael Xie, Kalesha Bullard, and Dorsa Sadigh. Reward design with language
743 models. In *The Eleventh International Conference on Learning Representations*, 2023.
- 744
- 745 Joonho Lee, Jemin Hwangbo, Lorenz Wellhausen, Vladlen Koltun, and Marco Hutter. Learning
746 quadrupedal locomotion over challenging terrain. *Science robotics*, 5(47):eabc5986, 2020.
- 747
- 748 Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and
749 Andy Zeng. Code as policies: Language model programs for embodied control. In *2023 IEEE*
750 *International Conference on Robotics and Automation (ICRA)*, pp. 9493–9500. IEEE, 2023.
- 751
- 752 Ji Lin, Hongxu Yin, Wei Ping, Pavlo Molchanov, Mohammad Shoeybi, and Song Han. Vila: On pre-
753 training for visual language models. In *Proceedings of the IEEE/CVF Conference on Computer*
754 *Vision and Pattern Recognition (CVPR)*, pp. 26689–26699, June 2024.
- 755
- 756 Kevin Lin, Christopher Agia, Toki Migimatsu, Marco Pavone, and Jeannette Bohg. Text2motion:
757 From natural language instructions to feasible plans. *Autonomous Robots*, 47(8):1345–1365,
758 2023.
- 759
- 760 Fangchen Liu, Kuan Fang, Pieter Abbeel, and Sergey Levine. Moka: Open-vocabulary robotic
761 manipulation through mark-based visual prompting. *arXiv preprint arXiv:2403.03174*, 2024a.

- 756 Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *Advances*
757 *in neural information processing systems*, 36, 2024b.
- 758
- 759 Pan Lu, Swaroop Mishra, Tony Xia, Liang Qiu, Kai-Wei Chang, Song-Chun Zhu, Oyvind Tafjord,
760 Peter Clark, and Ashwin Kalyan. Learn to explain: Multimodal reasoning via thought chains for
761 science question answering. In *The 36th Conference on Neural Information Processing Systems*
762 *(NeurIPS)*, 2022.
- 763 Yecheng Jason Ma, Shagun Sodhani, Dinesh Jayaraman, Osbert Bastani, Vikash Kumar, and Amy
764 Zhang. Vip: Towards universal visual reward and representation via value-implicit pre-training.
765 In *The Eleventh International Conference on Learning Representations*, 2023.
- 766
- 767 Yecheng Jason Ma, William Liang, Guanzhi Wang, De-An Huang, Osbert Bastani, Dinesh Jayara-
768 man, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Eureka: Human-level reward design via
769 coding large language models. In *The Twelfth International Conference on Learning Representa-*
770 *tions*, 2024.
- 771 Ajay Mandlekar, Soroush Nasiriany, Bowen Wen, Iretoiyo Akinola, Yashraj Narang, Linxi Fan,
772 Yuke Zhu, and Dieter Fox. Mimicgen: A data generation system for scalable robot learning using
773 human demonstrations. In *Conference on Robot Learning*, pp. 1820–1864. PMLR, 2023.
- 774
- 775 Matthias Minderer, Alexey Gritsenko, Austin Stone, Maxim Neumann, Dirk Weissenborn, Alexey
776 Dosovitskiy, Aravindh Mahendran, Anurag Arnab, Mostafa Dehghani, Zhuoran Shen, et al. Sim-
777 ple open-vocabulary object detection. In *European Conference on Computer Vision*, pp. 728–755.
778 Springer, 2022.
- 779 Matthias Minderer, Alexey A. Gritsenko, and Neil Houlsby. Scaling open-vocabulary object de-
780 tection. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL
781 <https://openreview.net/forum?id=mQPNcBWjGc>.
- 782
- 783 Suraj Nair, Aravind Rajeswaran, Vikash Kumar, Chelsea Finn, and Abhinav Gupta. R3m: A univer-
784 sal visual representation for robot manipulation. In *Conference on Robot Learning*, pp. 892–909.
785 PMLR, 2023.
- 786 Soroush Nasiriany, Fei Xia, Wenhao Yu, Ted Xiao, Jacky Liang, Ishita Dasgupta, Annie Xie, Danny
787 Driess, Ayzaan Wahid, Zhuo Xu, et al. Pivot: Iterative visual prompting elicits actionable knowl-
788 edge for vlms. In *International Conference on Machine Learning*, 2024.
- 789
- 790 Dantong Niu, Yuvan Sharma, Giscard Biamby, Jerome Quenum, Yutong Bai, Baifeng Shi, Trevor
791 Darrell, and Roei Herzig. LLARVA: Vision-action instruction tuning enhances robot learning.
792 In *8th Annual Conference on Robot Learning*, 2024. URL [https://openreview.net/](https://openreview.net/forum?id=Q21GXMZCv8)
793 [forum?id=Q21GXMZCv8](https://openreview.net/forum?id=Q21GXMZCv8).
- 794 Simone Parisi, Aravind Rajeswaran, Senthil Purushwalkam, and Abhinav Gupta. The unsurprising
795 effectiveness of pre-trained vision models for control. In *international conference on machine*
796 *learning*, pp. 17359–17371. PMLR, 2022.
- 797
- 798 Wilbert Pumacay, Ishika Singh, Jiafei Duan, Ranjay Krishna, Jesse Thomason, and Dieter Fox. The
799 colosseum: A benchmark for evaluating generalization for robotic manipulation. *arXiv preprint*
800 *arXiv:2402.08191*, 2024.
- 801 Ilija Radosavovic, Baifeng Shi, Letian Fu, Ken Goldberg, Trevor Darrell, and Jitendra Malik. Robot
802 learning with sensorimotor pre-training. In *Conference on Robot Learning*, pp. 683–693. PMLR,
803 2023.
- 804
- 805 Urs Ramer. An iterative procedure for the polygonal approximation of plane curves. *Computer*
806 *Graphics and Image Processing*, 1(3):244–256, 1972. ISSN 0146-664X. doi: [https://doi.org/10.](https://doi.org/10.1016/S0146-664X(72)80017-0)
807 [1016/S0146-664X\(72\)80017-0](https://www.sciencedirect.com/science/article/pii/S0146664X72800170). URL [https://www.sciencedirect.com/science/](https://www.sciencedirect.com/science/article/pii/S0146664X72800170)
808 [article/pii/S0146664X72800170](https://www.sciencedirect.com/science/article/pii/S0146664X72800170).
- 809 Rutav M Shah and Vikash Kumar. Rrl: Resnet as representation for reinforcement learning. In
International Conference on Machine Learning, pp. 9465–9476. PMLR, 2021.

- 810 Zejiang Shen, Kyle Lo, Lucy Lu Wang, Bailey Kuehl, Daniel S. Weld, and Doug Downey. Incorporating visual layout structures for scientific text classification. *ArXiv*, abs/2106.00676, 2021.
811 URL <https://arxiv.org/abs/2106.00676>.
812
- 813 Mohit Shridhar, Lucas Manuelli, and Dieter Fox. Perceiver-actor: A multi-task transformer for
814 robotic manipulation. In *Conference on Robot Learning*, pp. 785–799. PMLR, 2023.
815
- 816 Ishika Singh, Valts Blukis, Arsalan Mousavian, Ankit Goyal, Danfei Xu, Jonathan Tremblay, Dieter
817 Fox, Jesse Thomason, and Animesh Garg. Progprompt: Generating situated robot task plans using
818 large language models. In *2023 IEEE International Conference on Robotics and Automation*
819 *(ICRA)*, pp. 11523–11530. IEEE, 2023.
820
- 821 Sumedh Anand Sontakke, Jesse Zhang, Séb Arnold, Karl Pertsch, Erdem Biyik, Dorsa Sadigh,
822 Chelsea Finn, and Laurent Itti. Roboclip: One demonstration is enough to learn robot policies. In
823 *NeurIPS*, 2023.
- 824 Austin Stone, Ted Xiao, Yao Lu, Keerthana Gopalakrishnan, Kuang-Huei Lee, Quan Vuong, Paul
825 Wohlhart, Sean Kirmani, Brianna Zitkovich, Fei Xia, et al. Open-world object manipulation using
826 pre-trained vision-language models. In *Conference on Robot Learning*, pp. 3397–3417. PMLR,
827 2023.
828
- 829 Priya Sundareshan, Suneel Belkhale, Dorsa Sadigh, and Jeannette Bohg. Kite: Keypoint-conditioned
830 policies for semantic manipulation. In *Conference on Robot Learning*, pp. 1006–1021. PMLR,
831 2023.
- 832 Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep
833 Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, et al. Octo: An open-source generalist robot
834 policy. *arXiv preprint arXiv:2405.12213*, 2024.
835
- 836 Marcel Torne, Anthony Simeonov, Zechu Li, April Chan, Tao Chen, Abhishek Gupta, and Pulkit
837 Agrawal. Reconciling reality through simulation: A real-to-sim-to-real approach for robust ma-
838 nipulation. *Robotics: Science and Systems*, 2024.
- 839 Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez,
840 Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023. URL <https://arxiv.org/abs/1706.03762>.
841
- 842 Homer Walke, Kevin Black, Abraham Lee, Moo Jin Kim, Max Du, Chongyi Zheng, Tony Zhao,
843 Philippe Hansen-Estruch, Quan Vuong, Andre He, Vivek Myers, Kuan Fang, Chelsea Finn, and
844 Sergey Levine. Bridgedata v2: A dataset for robot learning at scale. In *Conference on Robot*
845 *Learning (CoRL)*, 2023.
846
- 847 Yufei Wang, Zhanyi Sun, Jesse Zhang, Zhou Xian, Erdem Biyik, David Held, and Zackory Erick-
848 son. RL-vlm-f: Reinforcement learning from vision language foundation model feedback. In
849 *International Conference on Machine Learning*, 2024.
850
- 851 Wenhao Yu, Nimrod Gileadi, Chuyuan Fu, Sean Kirmani, Kuang-Huei Lee, Montserrat Gonzalez
852 Arenas, Hao-Tien Lewis Chiang, Tom Erez, Leonard Hasenclever, Jan Humplik, et al. Language
853 to rewards for robotic skill synthesis. In *Conference on Robot Learning*, pp. 374–404. PMLR,
854 2023.
- 855 Wentao Yuan, Jiafei Duan, Valts Blukis, Wilbert Pumacay, Ranjay Krishna, Adithyavairavan Murali,
856 Arsalan Mousavian, and Dieter Fox. Robopoint: A vision-language model for spatial affordance
857 prediction in robotics. In *8th Annual Conference on Robot Learning*, 2024. URL <https://openreview.net/forum?id=GVX6jzpZOhU>.
858
- 859 Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual
860 manipulation with low-cost hardware. In Kostas E. Bekris, Kris Hauser, Sylvia L. Herbert, and
861 Jingjin Yu (eds.), *Robotics: Science and Systems XIX, Daegu, Republic of Korea, July 10-14,*
862 *2023*, 2023. doi: 10.15607/RSS.2023.XIX.016. URL [https://doi.org/10.15607/RSS.](https://doi.org/10.15607/RSS.2023.XIX.016)
863 *2023.XIX.016*.

864 Wanrong Zhu, Jack Hessel, Anas Awadalla, Samir Yitzhak Gadre, Jesse Dodge, Alex Fang, Young-
865 jae Yu, Ludwig Schmidt, William Yang Wang, and Yejin Choi. Multimodal C4: An open, billion-
866 scale corpus of images interleaved with text. *arXiv preprint arXiv:2304.06939*, 2023.
867

868 Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart,
869 Stefan Welker, Ayzan Wahid, et al. Rt-2: Vision-language-action models transfer web knowledge
870 to robotic control. In *Conference on Robot Learning*, pp. 2165–2183. PMLR, 2023.
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917

For extended supplementary details and results, please see <https://sites.google.com/view/hamster-iclr>.

A VLM FINETUNING DATASET DETAILS

Pixel Point Pred Data. Our point prediction dataset comes from Robopoint (Yuan et al., 2024). Most data in our point prediction dataset contains labels given as a set of unordered points such as $p^o = [(0.25, 0.11), (0.22, 0.19), (0.53, 0.23)]$. However, data in RoboPoint also contains answers that are instead in natural language for VQA queries such as “what is the person feeding the cat?” We keep these data as is because these VQA queries are likely to benefit a VLM’s semantic reasoning on a visual generalization capabilities; we fine-tune HAMSTER’s VLM on the entire Robopoint dataset as given.

Simulation Data. We selected 79 RL Bench tasks out of 100 to generate data by removing the tasks with poor visibility on the `front_cam` view in RL Bench. We use the first image in each episode combined with each language instruction. The final dataset contains around 320k trajectories.

Real Robot Data. For the Bridge (Walke et al., 2023) dataset, which only provides RGB images, we extract trajectories by iteratively estimating the extrinsic matrix for each episode. In each scene, we randomly sample a few frames and manually label the center of the gripper fingers. Using the corresponding end-effector poses, we compute the 3D-2D projection matrix with a PnP (Perspective-n-Point) approach. We then apply this projection matrix to the episodes and manually check for any misalignments between the projected gripper and the actual gripper. Episodes exhibiting significant deviations are filtered out, and a new round is started to estimate their extrinsic matrix.

For DROID (Khazatsky et al., 2024), a large portion of the dataset contains noisy camera extrinsics information that do not result in good depth alignment. Therefore, we filter out trajectories with poor-quality extrinsics as measured by the alignment between the projected depth images and the RGB images. This results in $\sim 45k$ trajectories ($\sim 22k$ unique trajectories as trajectories each have 2 different camera viewpoints) which we use for constructing the VLM dataset \mathcal{D}_{off} as described in Section 4.1.

B IMPLEMENTATION AND ARCHITECTURE DETAILS

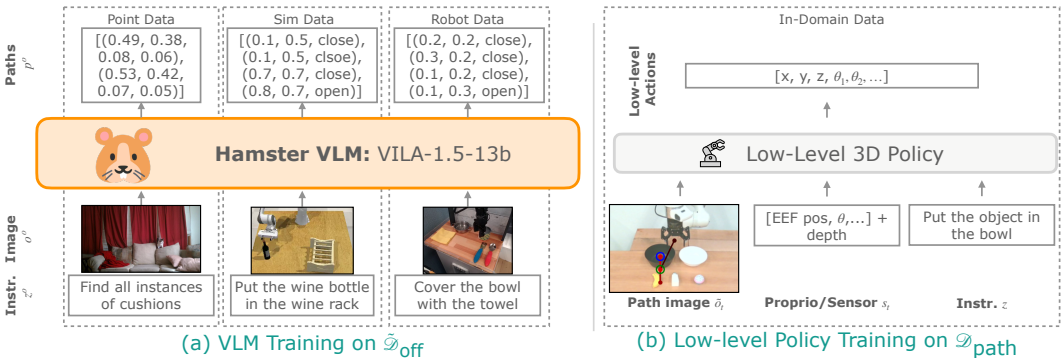


Figure 7: (a): Examples of training data in $\tilde{\mathcal{D}}_{\text{off}}$ used to train HAMSTER’s VLM. (b): The data used to train HAMSTER’s low-level policies.

B.1 VLM IMPLEMENTATION DETAILS

VLM Prompt. We list the prompt for both fine-tuning on sim and real robot data and evaluation in Figure 8. We condition the model on an image and the prompt, except when training on Pixel Point Prediction data (i.e., from Robopoint (Yuan et al., 2024)) where we used the given prompts from the dataset. Note that we ask the model to output gripper changes as separate language tokens, i.e.,

972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025

HAMSTER Prompt

In the image, please execute the command described in `<quest>{quest}</quest>`.
 Provide a sequence of points denoting the trajectory of a robot gripper to achieve the goal.
 Format your answer as a list of tuples enclosed by `<ans>` and `</ans>` tags. For example:
`<ans>[(0.25, 0.32), (0.32, 0.17), (0.13, 0.24), <action>Open
 Gripper</action>, (0.74, 0.21), <action>Close Gripper</action>
 ...]</ans>`
 The tuple denotes the x and y location of the end effector of the gripper in the image. The action tags
 indicate the gripper action.
 The coordinates should be floats ranging between 0 and 1, indicating the relative locations of the points
 in the image.

Figure 8: The full text prompt we use to train HAMSTER with on simulation and real robot data (Section 4.1). We also use this prompt for inference.

Open Gripper/Close Gripper, as opposed to as a numerical value as shown in simplified depictions like Figure 2.

VLM Trajectory Processing. As mentioned in Section 4.1, one problem with directly training on the path labels p^o is that many paths may be extremely long. Therefore, we simplify the paths p^o with the Ramer-Douglas-Peucker algorithm (Ramer, 1972; Douglas & Peucker, 1973) that reduces curves composed of line segments to similar curves composed of fewer points. We run this algorithm on paths produced by simulation and real robot data to generate the labels p^o for \mathcal{D}_{off} . We use tolerance $\epsilon = 0.05$, resulting in paths that are around 2-5 points for each short horizon task.

VLM Training Details. We train our VLM, VILA1.5-13B Lin et al. (2024), on a node equipped with eight NVIDIA A100 GPUs, each utilizing approximately 65 GB of memory. The training process takes about 30 hours to complete. We use an effective batch size of 256 and a learning rate of 1×10^{-5} . During fine-tuning, the entire model—including the vision encoder—is updated.

B.2 LOW-LEVEL POLICY TRAINING DETAILS

We train RVT2 (Goyal et al., 2024) and 3D-DA (Ke et al., 2024) as our lower-level policies. We keep overall architecture and training hyperparameters the same as paper settings. Specific details about how the inputs were modified other than the 2D path projection follow.

For low-level policy training, we train the policies on ground truth paths constructed by projecting trajectory end-effector points to the camera image. In order to also ensure the policies are robust to possible error introduced by HAMSTER VLM predictions during evaluation, we add a small amount of random noise ($N(0, 0.01)$) to the 2D path (x, y) image points during training to obtain slightly noisy path drawings. No noise was added to the gripper opening/closing indicator values.

RVT2 (Goyal et al., 2024). We remove the language instruction for RVT-2 when conditioning on HAMSTER 2D paths.

3D-DA (Ke et al., 2024). In simulated experiments in Colosseum, no changes were needed. In fact, we saw a performance drop for HAMSTER+3D-DA when removing language for Colosseum tasks and a small drop in performance when using simplified language instructions. This is likely due to 3D-DA’s visual attention mechanism which cross attends CLIP language token embeddings with CLIP visual features, therefore detailed language instructions are beneficial.

In real-world experiments, we simplify the language instruction in the same way as for RVT2 when conditioning on HAMSTER 2D paths to encourage following the trajectory more closely with limited data. In addition, we reduced the embedding dimension of the transformer to 60 from 120, removed proprioception information from past timesteps, and reduced the number of transformer heads to 6 from 12 in order to prevent overfitting.

1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038

RT-Trajectory GPT-4o Prompt

In the image, please execute the command described in '{quest}'.

Provide a sequence of keypoints denoting a trajectory of a robot gripper to achieve the goal. Keep in mind these are keypoints, so you do not need to provide too many points.

Format your answer as a list of tuples enclosed by <ans> and </ans> tags. For example:
<ans>[(0.25, 0.32), (0.32, 0.17), (0.13, 0.24), <action>Open Gripper</action>, (0.74, 0.21), <action>Close Gripper</action>, ...]</ans>

The tuple denotes point x and y location of the end effector of the gripper in the image. The action tags indicate the gripper action.

The coordinates should be floats ranging between 0 and 1, indicating the relative locations of the points in the image.

The current position of the robot gripper is: {current_position}. Do not include this point in your answer.

1039
1040
1041

Figure 9: The full text prompt we use to prompt RT-Trajectory with GPT4-o.

1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061

RT-Trajectory Code as Policies Prompt

Task Instruction: {task_instruction}

Robot Constraints:

- The robot arm takes as input 2D poses with gripper open/closing status of the form $(x, y, gripper_open == 1)$
- The gripper can open and close with only binary values
- The workspace is a 1×1 square centered at $(0.5, 0.5)$
- The x-axis points rightward and y-axis points downward.

Please write Python code that generates a list of 2D poses and gripper statuses for the robot to follow. Include Python comments explaining each step. Assume you can use `numpy` or standard Python libraries, just make sure to import them.

Enclose the start and end of the code block with <code> and </code> so that it can be parsed. Make sure that it is a self-contained script such that when executing the code string, there is a variable named `robot_poses` which is a list of poses of the form: `[(x, y, gripper), (x, y, gripper), ...]`.

Scene Description:

```
<code>
{scene_description}
</code>
```

1062
1063
1064
1065
1066

Figure 10: The full text prompt we use for RT-Trajectory with Code-as-Policies on top of GPT4-o. The scene description at the bottom comes from an open-vocabulary object detector describing each detected object and its bounding box in the image based on the task instruction.

1067
1068
1069
1070
1071

C REAL WORLD EXPERIMENT DETAILS

1072
1073
1074
1075
1076
1077
1078
1079

C.1 TRAINING TASKS AND DATA COLLECTION

For our real-world experiments, we collected all data using a Franka Panda arm through human teleoperation, following the setup described in Khazatsky et al. (2024). Below, we describe the training tasks:

Pick and place. We collected 220 episodes using 10 toy objects. In most of the training data, 2 bowls were placed closer to the robot base, while 3 objects were positioned nearer to the camera. The language goal for training consistently followed the format: `pick up the {object}` and `put it in the {container}`.

Knock down objects. We collected 50 episodes with various objects of different sizes. Typically, 3 objects were arranged in a row, and one was knocked down. The language goal for training followed the format: `push down the {object}`.

Press button. We collected 50 episodes with 4 colored buttons. In each episode, the gripper was teleoperated to press one of the buttons. The language goal followed the format: `press the {color} button`.

When training RVT2, which requires keyframes as labels, in addition to labeling frames where the gripper performs the `open gripper` and `close gripper` actions, we also included frames that capture the intermediate motion as the gripper moves toward these keyframes.

C.2 BASELINE TRAINING DETAILS

OpenVLA (Kim et al., 2024). Following Kim et al. (2024), we only utilize parameter efficient fine-tuning (LoRA) for all of our experiments, since they showed that it matches full fine-tuning performance while being much more efficient. We follow the recommended default rank of $r=32$. We opt for the resolution of 360×360 to match all of the baseline model’s resolutions. We also follow the recommended practice of training the model until it surpasses 95% token accuracy. However, for some fine-tuning datasets, token accuracy converged near 90%. We selected the model checkpoints when we observed that the token accuracy converged, which usually required 3,000 to 10,000 steps using a global batch size of either 16 or 32. Training was conducted with 1 or 2 A6000 gpus (which determined the global batch size of 16 or 32). Empirically, we observed that checkpoints that have converged showed very similar performance in the real world. For example, when we evaluate checkpoint that was trained for 3,000 steps and showed convergence, evaluating on a checkpoint trained for 5,000 steps of the same run resulted in a very similar performance.

RT-Trajectory (Gu et al., 2023). We implement two versions of RT-Trajectory for the comparison in Table 5. The first (0-shot GPT-4o) directly uses GPT-4o to generate 2D paths with a prompt very similar to the one we use for HAMSTER, displayed in Figure 9.

The second version implements RT-Trajectory on top of a Code-as-Policies (Liang et al., 2023), as described in RT-Trajectory. We use OWLv2 (Minderer et al., 2023) to perform open-vocabulary object detection on the image to generate a list of objects as the scene description and then prompt RT-Trajectory with the prompt shown in Figure 10. We also use GPT-4o as the backbone for this method.

C.3 EVALUATION TASKS

We evaluate our method on the tasks of `pick` and `place`, `knock down object`, and `press button` across various generalization challenges, as illustrated in Figure 3. Detailed results are available in Appendix C.3. Following (Kim et al., 2024), we assign points for each successful sub-action. For VLM, human experts are employed to assess the correctness of the predicted trajectories.

D EXTENDED RESULTS

D.1 IMPACT OF DESIGN DECISIONS ON VLM PERFORMANCE

To better understand the transfer and generalization performance of the proposed hierarchical VLA model, we analyze the impact of various decisions involved in training the high-level VLM. We conduct a human evaluation of different variants of a trained high-level VLM on a randomly collected dataset of real-world test images, as shown in Figure 6. We ask each model to generate 2D path traces corresponding to instructions such as “move the block on the right to Taylor Swift” or “screw the light bulb in the lamp” (the full set is in Appendix D.2). We then provide the paths generated by each method to human evaluators who have not previously seen any of the models’ predictions. The human evaluators then rank the predictions for each method; we report the average rank across the samples in Table 5.

1134

1135

1136

1137

1138

1139

1140

1141

1142

1143

1144

1145

1146

1147

1148

1149

1150

1151

1152

1153

1154

1155

1156

1157

1158

1159

1160

1161

1162

1163

1164

1165

1166

1167

1168

1169

1170

1171

1172

1173

1174

1175

1176

1177

1178

1179

1180

1181

1182

1183

1184

1185

1186

1187

Category	Task	OpenVLA	RVT2	RVT2+Sketch	3DDA	3DDA+Sketch
Basic	pick up the corn and put it in the black bowl	1	1	1	0	0.25
Basic	pick up the grape and put it in the white bowl	1	0.75	1	0	1
Basic	pick up the milk and put it in the white bowl	0	1	1	0	0.25
Basic	pick up the salt bottle and put it in the white bowl	0.75	0.5	1	0	0
Basic	pick up the shrimp and put it in the red bowl	0.75	0.5	1	0	1
Basic	pick up the cupcake and put it in the red bowl	0	0.5	0.5	0.25	1
Basic	press down the red button	0.5	0	1	0	1
Basic	press down the green button	0	1	0	0	0.25
Basic	press down the yellow button	0	0	1	0	1
Basic	press down the blue button	0.5	0	1	0	0.5
Basic	push down the green bottle	0.5	0	0.5	0	1
Basic	push down the pocky	0	1	1	0	0.5
Basic	push down the red bag	0.5	0.5	0	0	0.5
Basic	push down the bird toy	0	0	0	0	0.5
Basic	push down the yellow box	1	0	1	0	0.5
Object and Goal	pick up the salt bottle and put it in the white bowl	1	1	1	0.5	1
Object and Goal	pick up the banana and put it in the black bowl	0.25	0.25	1	0.5	1
Object and Goal	pick up the grape and put it in the black bowl	1	0.25	0.5	1	1
Object and Goal	pick up the carrot and put it in the red bowl	0.75	0	1	0.5	1
Object and Goal	pick up the milk and put it in the white bowl	0.25	0	1	0	0.25
Object and Goal	pick up the shrimp and put it in the white bowl	0.25	0.75	0.5	0.25	1
Object and Goal	pick up the cupcake and put it in the black bowl	0.25	0	1	0.5	0.75
Object and Goal	pick up the icecream and put it in the black bowl	0.25	0	0.5	0.5	1
Object and Goal	pick up the corn and put it in the red bowl	1	0	1	1	1
Object and Goal	pick up the green pepper and put it in the red bowl	0.75	0	0.5	0	0.25
Object and Goal	pick up the orange and put it in the white bowl	0.25	0	0	0	0
Visual(Table Texture)	pick up the salt bottle and put it in the white bowl	1	1	1	0	1
Visual(Table Texture)	pick up the banana and put it in the black bowl	0.25	0.25	0.75	0.5	0.75
Visual(lightning)	pick up the grape and put it in the black bowl	0.25	0	0.5	0.25	0
Visual(lightning)	pick up the carrot and put it in the red bowl	0.75	0	1	0	0.75
Visual(clutter)	pick up the milk and put it in the white bowl	0.75	0.25	1	0.25	1
Visual(clutter)	pick up the shrimp and put it in the red bowl	0.75	0.5	0	0	0.5
Visual(mix)	pick up the green pepper and put it in the red bowl	0.25	0	1	0	0.25
Visual(mix)	pick up the salt bottle and put it in the white bowl	0.25	0	0.25	0.25	1
Visual(appearance change)	pick up the green pepper and put it in the black bowl	1	0	0.5	0	1
Visual(appearance change)	pick up the salt bottle and put it in the black bowl	1	1	1	0	1
Visual(Table Texture)	press down the red button	1	1	0	0	0.5
Visual(lightning)	press down the green button	1	0	0.5	0	0.5
Visual(clutter)	press down the yellow button	0	0	0.5	0	0.5
Visual(mix)	press down the blue button	0	0	0	0	0.5
Visual(Table Texture)	push down the pocky	0	1	0	0	0
Visual(clutter)	push down the green bottle	1	0.5	1	0	1
Visual(clutter)	push down the chocolate box	1	0	0	0	1
Visual(mix)	push down the green bottle	0	0	0.5	0	1
Language	pick up the sweet object and put it in the red bowl	1	1	1	0	1
Language	pick up the spicy object and put it in the red bowl	1	0	1	0	0.75
Language	pick up the salty object and put it in the red bowl	0	0	1	0	1
Language	pick up the object with color of cucumber and put it in the red bowl	0	0	1	0.25	0.75
Language	pick up the object with color of lavender and put it in the black bowl	0	0	1	0	1
Language	pick up the object with the color of sky and put it in the container with the color of coal	1	0	0	0.25	1
Language	pick up the block with the color of sunflower and put it in the container with the color of enthusiasm	0	0.25	1	0	1
Language	press the button with the color of fire	0.5	0	1	0	0.5
Language	press the button with the color of cucumber	0	0	1	0	0.5
Language	press the button with the color of sky	0	0	0	0.5	1
Language	press the button with the color of banana	0	0	0	0	0.5
Language	push down the object with color of leaf	0	1	1	0	0
Language	push down the box contains cruchy biscuit	0	0	0	0	1
Language	push down the bag with color of fire	0	0	1	0	0.5
Language	push down the object with feather	0.5	0	1	0	1
Spatial	pick up the left object and put it in the left bowl	0	1	1	0.25	1
Spatial	pick up the middle object and put it in the left bowl	0	0	1	0	1
Spatial	pick up the right object and put it in the left bowl	1	0	0.5	0.25	0.5
Spatial	pick up the left object and put it in the right bowl	0.25	0.25	1	0.25	1
Spatial	pick up the middle object and put it in the right bowl	0	0	1	0	1
Spatial	pick up the right object and put it in the right bowl	0.5	0	1	0	1
Spatial	press down the left button	0.5	0	0	0	0.5
Spatial	press down the middle button	0	0	1	1	0.5
Spatial	press down the right button	0	0	1	1	1
Spatial	push down the left object	0.5	0	0	0	0
Spatial	push down the middle object	1	0.5	0	0	1
Spatial	push down the right object	0.5	0	0.5	0.5	1
Novel Object	pick up the "R" and put it in the red bowl	0	0	1	0	1
Novel Object	pick up the boxed juice and put it in the red bowl	0	0.75	0.75	1	1
Novel Object	pick up the cholate bar and put it in the white bowl	0.25	0	0.5	0.5	1
Novel Object	pick up the smile face and put it in the red bowl	1	0	1	0	1
Novel Object	pick up the mouse and put it in the red bowl	0	0.25	1	0	1
Novel Object	pick up the 5 and put it in the white bowl	0	0	0	0	0.25
Multiple	pick up the lays chip and put it in the pan	0.25	0.25	0.75	0	1
Multiple	pick up the garlic and put it in then pan	0.25	0	1	0	0.25
Multiple	pick up the "K" and put it in the pan	0.25	0	0.5	0	1
Multiple	pick up the pocky and put it in the pan	0	0.25	0	0.25	0.25

Table 4: Detailed results of real-world evaluation. The first column indicates the variation category, while the second column presents the language instruction. For the `pick` and `place` task, 0.25 points are awarded for each successful action: reaching the object, picking it up, moving it to the target container, and placing it inside. For the `knock down` task, 0.5 points are awarded for touching the correct object and successfully knocking it down. For the `press button` task, 0.5 points are awarded for positioning the gripper above the correct button and successfully pressing it.

1188
 1189
 1190
 1191
 1192
 1193
 1194
 1195
 1196
 1197
 1198
 1199
 1200
 1201
 1202
 1203
 1204
 1205
 1206
 1207
 1208
 1209
 1210
 1211
 1212
 1213
 1214
 1215
 1216
 1217
 1218
 1219
 1220
 1221
 1222
 1223
 1224
 1225
 1226
 1227
 1228
 1229
 1230
 1231
 1232
 1233
 1234
 1235
 1236
 1237
 1238
 1239
 1240
 1241

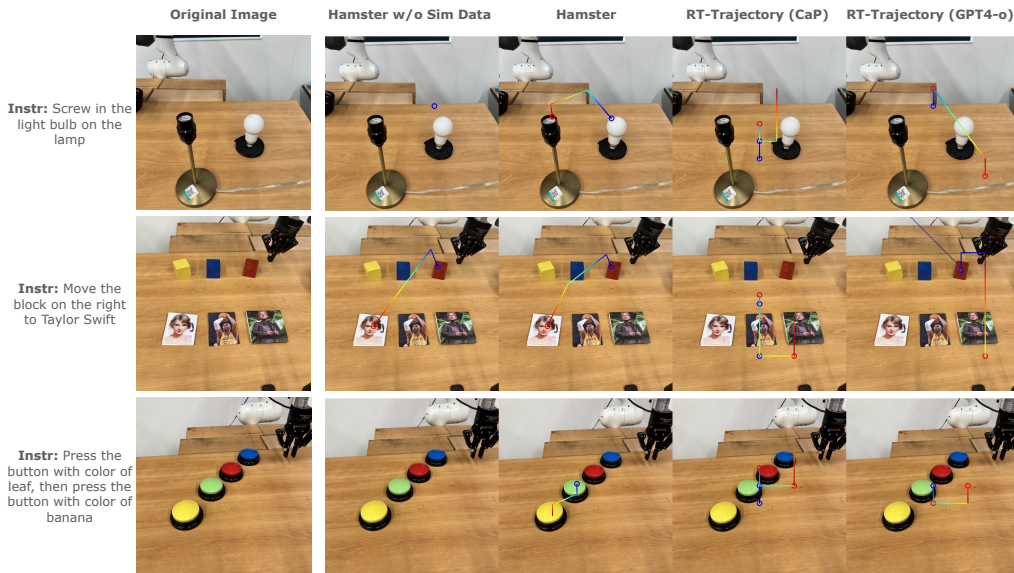


Figure 11: Human VLM evaluation example images and instructions along with corresponding trajectories from HAMSTER without any finetuning on (RLBench) simulation data, HAMSTER finetuned on all the data in Section 4.1, RT-Trajectory (Gu et al., 2023) with Code-as-Policies (Liang et al., 2023) powered by GPT-4o (Achiam et al., 2023), and RT-Trajectory powered by GPT-4o directly.

We evaluate the following VLM models: (1) zero-shot state-of-the-art closed-source models such as GPT-4o using a similar prompt to ours (shown in Figure 9), (2) zero-shot state-of-the-art closed-source models such as GPT-4o but using Code-as-Policies (Liang et al., 2023) to generate paths as described in Gu et al. (2023) (prompt in Figure 10), (3) finetuned open-source models (VILA-1.5-13b) on the data sources described in Section 4.1, but excluding the simulation trajectories from the RLBench dataset, (4) finetuned open-source models (VILA-1.5-13b) on the data sources described in Section 4.1, including path sketches from the RLBench dataset. The purpose of these evaluations is to first compare with closely related work that generates 2D trajectories using pretrained closed source VLMs Gu et al. (2023) (Comparison (1) and (2)). The comparison between (3) and (4) (our complete method) is meant to isolate the impact of including the simulation path sketches from the RLBench dataset. In doing so, we analyze the ability of the VLM to predict intermediate paths to transfer across significantly varying domains (from RLBench to the real world).

The results suggest that: (1) zero-shot path generation, even from closed-source VLMs Gu et al. (2023) such as GPT-4o with additional help through Code-as-Policies (Liang et al., 2023), underperforms VLMs finetuned on cross-domain data as in HAMSTER; (2) inclusion of significantly different training data such as low-fidelity simulation during finetuning improves the real-world performance of the VLM. This highlights the transferability displayed by HAMSTER across widely varying domains. These results emphasize that the hierarchical VLA approach described in HAMSTER can effectively utilize diverse sources of cheap prior data for 2D path predictions, despite considerable perceptual differences.

D.2 VLM REAL WORLD GENERALIZATION STUDY

The full list of task descriptions for this study is below (see Appendix D.1 for the main experiment details). Duplicates indicate different images for the same task. We plot some additional comparison examples in Figure 11. Note that the path drawing convention in images for this experiment differ from what is given to the lower-level policies as described in Section 4.2 as this multi-colored line is easier for human evaluators to see.

1. screw in the light bulb on the lamp

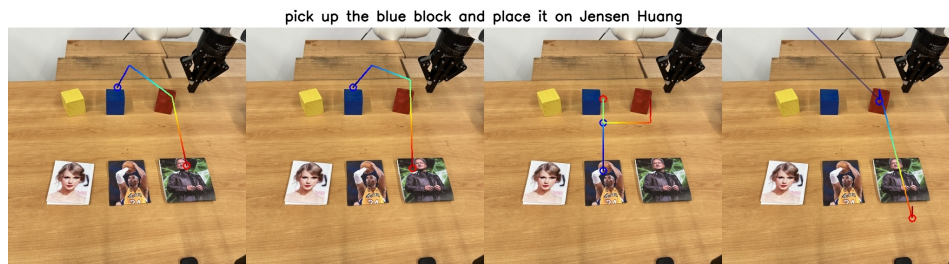
Method	VLM	Finetuning Data	Rank Exc. Real RLB.	Rank Real RLB.	Rank All
RT-Traj.	0-shot GPT-4o	-	3.40	3.63	3.47
RT-Traj.	CaP GPT-4o	-	3.57	3.36	3.41
HAMSTER	VILA	Our Exc. Sim RLB.	1.78	2.39	2.13
HAMSTER	VILA	Our	1.59	1.28	1.40

Table 5: Ranking-based human evaluation of different VLMs, averaged across various real-world evaluation tasks. Results indicate that HAMSTER including simulation data is most effective since it captures both spatial and semantic information across diverse tasks from RL Bench. This significantly outperforms zero-shot VLM-based trajectory generation, as described in Gu et al. (2023)

2. screw in the light bulb on the lamp
3. screw in the light bulb on the lamp
4. screw out the light bulb and place it on the holder
5. screw out the light bulb and place it on the holder
6. screw in the light bulb
7. screw in the light bulb on the lamp
8. move the blue block on Taylor Swift
9. pick up the left block and put it on Jensen Huang
10. move the block on the right to Taylor Swift
11. place the yellow block on Kobe
12. pick up the blue block and place it on Jensen Huang
13. move the red block to Kobe
14. press the button on the wall
15. press the button to open the left door
16. press the button to open the right door
17. open the middle drawer
18. open the bottom drawer
19. open the top drawer
20. open the middle drawer
21. open the bottom drawer
22. press the button
23. press the button
24. press the orange button
25. press the orange button with black base
26. press the button
27. pick up the SPAM and put it into the drawer
28. pick up the orange juice and put it behind the red box
29. pick up the tomato soup and put it into the drawer
30. pick up the peach and put it into the drawer
31. move the mayo to the drawer
32. move the dessert to the drawer
33. pick up the object on the left and place it on the left
34. pick up the fruit on the left and put it on the plate
35. pick up the milk and put it on the plate

- 1296 36. press the button with the color of cucumber, then press the button with color of fire
- 1297 37. press the button with color of banana
- 1298 38. press the button with color of leaf
- 1300 39. press the button with color of leaf, then press the one with color of banana
- 1301 40. press left button
- 1302 41. pick up the left block on the bottom and stack it on the middle block on top
- 1303 42. make I on top of C
- 1304 43. put number 2 over number 5
- 1305 44. stack block with lion over block with earth
- 1306 45. pick up the left block on the bottom and stack it on the middle block on top
- 1307 46. stack the leftest block on the rightest block
- 1308 47. stack the block 25 over block L
- 1309 48. put the left block on first stair

1313 D.3 HUMAN RANKING



1324 Figure 12: An example of results for human ranking. The trajectory is from blue to red with blue
 1325 circle and red circle denotes gripper close point and open point respectively. The grader is asked to
 1326 provide a rank to these trajectory about which trajectory has highest chance to succeed.

1327
 1328 Due to the variety of possible trajectories that accomplish the same task, we use human rankings
 1329 to compare how likely produced trajectories are to solve the task instead of quantitative metrics
 1330 such as MSE. To do that, we generate trajectories for 48 image-question pairs with HAMSTER w/o
 1331 RL Bench, HAMSTER, Code-as-Policy (Liang et al., 2023), and GPT4o (Achiam et al., 2023). See
 1332 Figure 12 for an example.

1333 We recruit 5 human evaluators, who are robot learning researchers that have not seen the path outputs
 1334 of HAMSTER, to grade these 4 VLMs based on the instruction: “Provide a rank for each method (1
 1335 for best and 4 for worst). In your opinion, which robot trajectory is most likely to succeed. Traj goes
 1336 from blue to red, blue circle means close gripper, red circle means open gripper.” The evaluators
 1337 are allowed to give multiple trajectories the same score if they believe those trajectories are tied.
 1338 As they are robot learning researchers, they are familiar with the types of trajectories that are more
 1339 likely to succeed. Therefore, these rankings act as a meaningful trajectory quality metric.

1341 E FAILURE ANALYSIS

1342
 1343 This section outlines the failure modes observed during our experiments and provides a detailed
 1344 breakdown of the causes. Failures can be attributed to issues in **trajectory prediction**, **trajectory**
 1345 **adherence**, and **action execution**.

1347 E.1 DIFFERENT FAILURE MODES

1348
 1349 **Trajectory Prediction Failures** The Vision-Language Model (VLM) may fail to predict the cor-
 rect trajectory due to several factors:

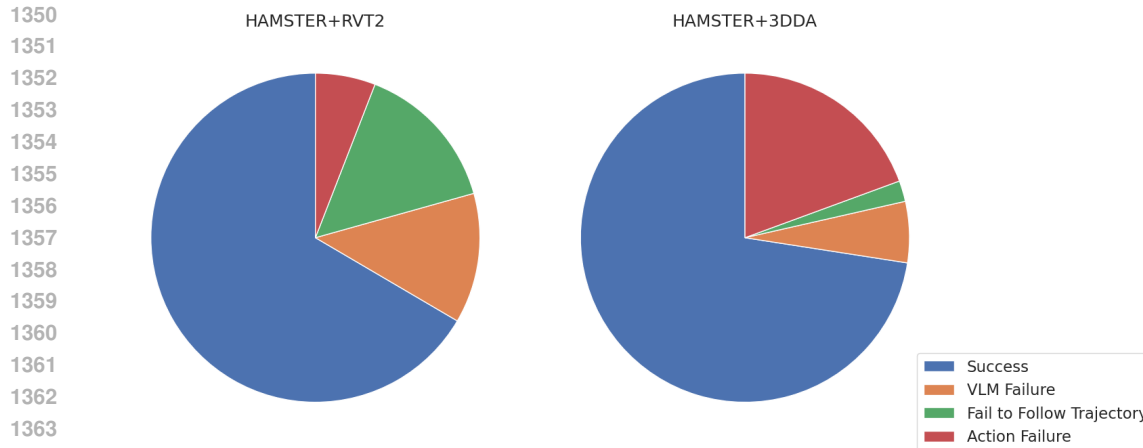


Figure 13: Performance Distribution of RVT2+Sketch and 3DDA+Sketch

1368 - *Failure to understand the language goal:* Although the VLM demonstrates strong capabilities in
1369 handling diverse task descriptions, it struggles when the training set lacks similar tasks. This can
1370 cause the model to misunderstand the goal and make inaccurate predictions.

1371 - *Incorrect trajectory prediction:* In some cases, the VLM predicts an incorrect trajectory, either by
1372 interacting with the wrong objects or misinterpreting the direction of the affordance.

1373 - *Dynamic changes in the environment:* Since trajectories are generated at the beginning of a task,
1374 significant environmental changes during execution can lead to failure. The model lacks the ability
1375 to dynamically adjust the trajectory or reidentify the object initially referenced.
1376

1377 **Trajectory Adherence Failures** Failures in adhering to the predicted trajectory arise primarily
1378 due to:

1379 - *3D ambiguity:* The use of 2D trajectory predictions introduces ambiguities, such as determining
1380 whether a point is positioned above or behind an object, leading to execution errors.

1381 - *Incorrect object interaction:* The low-level action model is not explicitly constrained to strictly
1382 follow the predicted trajectory. As a result, it may deviate, interacting with the wrong object and
1383 causing task failures.
1384

1385 **Action Execution Failures** Even when the trajectory is correctly predicted and adhered to, action
1386 execution may still fail due to:

1387 - *Execution-specific issues:* Despite training on a diverse set of actions, the model may fail during
1388 execution. For example, in grasping tasks, an incorrect grasp angle can cause the object to slip,
1389 resulting in a failed grasp.
1390

1391 E.2 FAILURE ANALYSIS

1392 Our analysis in Figure 13 reveals distinct failure tendencies across methods.

1393 For RVT, 72% of failures stemmed from the low-level model failing to follow the trajectory, while
1394 28% were due to execution failures. In contrast, for 3DDA, only 10% of failures were related to
1395 trajectory adherence, with 90% attributed to execution failures.

1396 We hypothesize that this discrepancy arises because RVT incorporates a re-projection step, compli-
1397 cating trajectory adherence. In contrast, 3DDA leverages a vision tower that processes the original
1398 2D image, simplifying trajectory interpretation.
1399

1400 F SIMULATION EXPERIMENT DETAILS

Our simulation experiments are performed on Colosseum (Pumacay et al., 2024), a simulator built upon RL Bench (James et al., 2020) containing a large number of visual and task variations to test the generalization performance of robot manipulation policies (see Figure 14 for a visualization of a subset of the variations). We use the `front_camera` and remove all tasks in which the camera does not provide a clear view of the objects in the task, resulting in 14 out of 20 colosseum tasks (we remove `basketball.in.hoop`, `empty.drawer`, `get.ice.from.fridge`, `move.hanger`, `open.drawer`, `turn.oven.on`).

Colosseum contains 100 training episodes for each task, without any visual variations, and evaluates on 25 evaluation episodes for each variation. We follow the same procedure other than using just the `front_camera` instead of multiple cameras. We report results in Table 1 after removing variations with no visual variations (e.g., object friction).

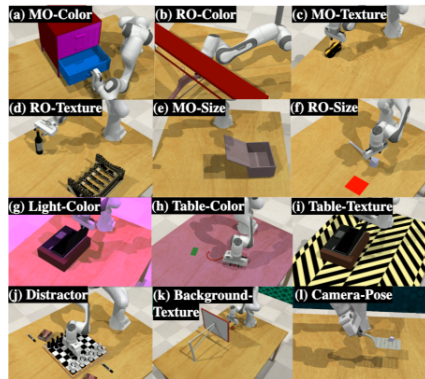


Figure 14: Colosseum benchmark variations. Figure from Pumacay et al. (2024), taken with permission.

Task	openvla	HAMSTER+RVT2	HAMSTER+3DDA
pick and place	0.46	0.79	0.78
press button	0.25	0.50	0.63
knock down	0.41	0.47	0.66

Table 6: Real world average success rates grouped by task type.