Robotics: Science and Systems 2026
Sydney, Australia, July 13–17, 2026

Steerable Vision-Language-Action Policies for Embodied Reasoning and Hierarchical Control

William Chen[1], Jagdeep Singh Bhatia[1], Catherine Glossop[1], Nikhil Mathihalli[1], Ria Doshi[2], Andy Tang[2], Danny Driess[3], Karl Pertsch[3], Sergey Levine[1]
[1]U.C. Berkeley, [2]Stanford University, [3]Physical Intelligence

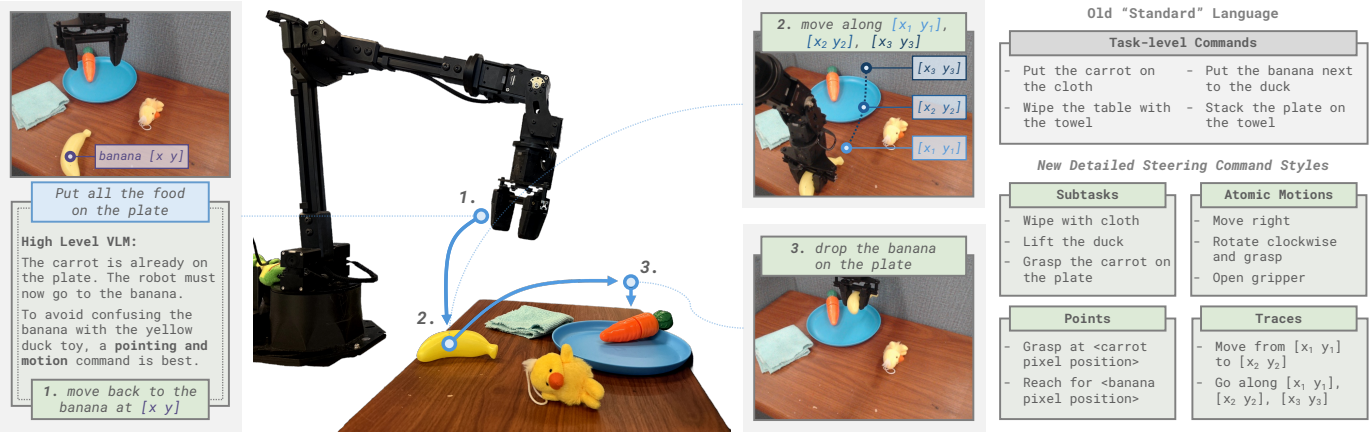


Fig. 1: We propose **Steerable Policies**: vision-language-action models that can robustly follow a wide range of detailed commands (green boxes on right), going beyond usual task language to include instructions such as motions or pixel coordinates of the gripper and objects. The flexibility afforded by Steerable Policies enables substantially improved transfer of VLMs' pretrained reasoning, semantic knowledge, and in-context learning skills to generalist robotic control in hierarchical systems.

***Abstract*—Pretrained vision-language models (VLMs) can make semantic and visual inferences across diverse settings, providing valuable common-sense priors for robotic control. However, effectively grounding this knowledge in robot behaviors remains an open challenge. Prior methods often employ a hierarchical approach where VLMs reason over high-level commands to be executed by separate low-level policies, e.g., vision-language-action models (VLAs). The interface between VLMs and VLAs is usually *natural language task instructions*, which fundamentally limits how much VLM reasoning can steer low-level behavior. We thus introduce Steerable Policies: VLAs trained on rich synthetic commands at various levels of abstraction, like subtasks, motions, and grounded pixel coordinates. By improving low-level controllability, Steerable Policies can unlock pretrained knowledge in VLMs, enabling improved task generalization. We demonstrate this benefit by controlling our Steerable Policies with both a learned high-level embodied reasoner and an off-the-shelf VLM prompted to reason over command abstractions via in-context learning. Across extensive real-world manipulation experiments, these two novel methods outperform prior embodied reasoning VLAs and VLM-based hierarchical baselines, including on challenging generalization and long-horizon tasks.**
Website: steerable-policies.github.io

I. Introduction

A long-standing goal of robotics is to develop *generalist* policies that can follow open-ended commands to perform a wide range of tasks. As scaling robot data collection remains expensive, this motivates the use of pretrained foundation vision-language models (VLMs) as a source of semantic and perceptual knowledge. Prior works often employ a hierarchy, where a high-level VLM reasons over task instructions and issues commands to a separate, language-conditioned low-level policy [1, 2]. However, effectively bringing VLM capabilities to bear for solving general robotics tasks is challenging, as doing so requires grounding their pretrained knowledge in robot behaviors. This raises a fundamental question: *how can we best unlock and leverage the generalization capabilities of pretrained foundation models for robotics?*

We posit that a major bottleneck in transferring these capabilities is insufficient policy *steerability*. Put simply, no matter how good a VLM is at determining robot behaviors needed for solving tasks, its reasoning abilities are wasted if the robot's policy cannot execute them. For example, the VLM might infer that an object needs to be grasped in a particular location, but this inference is only useful if the policy can understand commands specifying that location. Even powerful policies, such as vision-language-action models (VLA), have limited ability to follow diverse commands due to dataset limitations, where labels are often too formulaic, homogeneous, and imprecise to specify the full range of behaviors needed for solving new tasks [3]. These drawbacks make standard task-level language derived from dataset labels a poor interface for VLMs to control robot policies.

One insight is that steerability can be improved by augmenting existing datasets with more detailed synthetic language. Robot datasets already *implicitly* contain rich semantics, interactions, and behaviors, far beyond what is described in their sparse task labels. For example, a policy which learns to "wipe

the table with the towel on the left" tacitly learns what towels look like, how to grasp them, and how to move left – all of which can be used for other tasks, like "hang the towel on the hook." Training on more descriptive commands could help the policy in flexibly invoking these skills for solving new tasks.

However, naïvely densifying task descriptions is insufficient for generalization and composition. For example, expanding the descriptions of existing behaviors would struggle to teach the policy the names of fundamentally out-of-distribution objects. Still, the policy may be able to grasp these novel objects, due to their physical similarity to objects the policy *has* learned to interact with. A generalizable way to induce these behaviors is by prompting with *grounded* features, like pixel coordinates, as they can specify behaviors and concepts that are otherwise difficult to communicate to the policy (e.g., `"grasp at <novel object's position>"`). Not only are grounded features easily extracted for training, but modern VLMs are also trained to output them, thereby allowing these VLMs to transfer their pretrained knowledge to robotics.

We thus introduce **Steerable Policies**: robotic foundation models that follow a much wider range of commands than standard VLAs. Beyond typical "task-level" commands (e.g., `"put the carrot in the pot"`), we train VLAs to accept more detailed inputs, such as subtasks (`"reach for the carrot"`) or low-level atomic motions (`"move left and close the gripper"`). We also include commands with grounded features, such as gripper traces (`"move along` $[x_1, y_1], [x_2, y_2], ...$`"`) or points (`"grasp the object at` $[x, y]$`"`). Finally, we include prompts that compose all these modalities (`"move right from` $[x_1, y_1]$ `to put the mushroom on the plate at` $[x_2, y_2]$`"`). These ***steering commands*** are synthetically generated by a scalable pipeline that automatically parses and labels robot data. See Fig. 1 for more.

Compared to past hierarchical methods [1], Steerable Policies vastly expand the interface between low-level robot behaviors and high-level VLM skills, allowing high-level reasoners to flexibly choose instructions at the right level of abstraction for generalization and compositionality. We show this by proposing and evaluating two ways for VLMs to control our Steerable Policies, instantiated on a real-world Bridge WidowX setup [4, 5]. First, we show that VLMs can be fine-tuned into effective *high-level* models for instructing Steerable Policies by producing chain-of-thought reasonings (CoT [6, 7]) followed by steering commands. This outperforms past reasoning VLAs [8, 9], indicating that it makes especially good use of embodied reasoning data. Second, we show how Steerable Policies can better apply the *in-context learning* skills of API-based off-the-shelf VLMs. We find that these VLMs can apply in-context learning over past observations and commands to further improve robot behaviors by choosing commands of the correct level of abstraction. This is a novel functionality afforded by our Steerable Policies, which leads to significant performance gains over standard baselines.

In summary, we show that improving VLA steerability is critical to facilitate transfer of VLM capabilities to robotics. We do this by introducing **(1)** Steerable Policies: VLAs that can be prompted at many levels of abstraction to perform diverse manipulation skills – a paradigm which is agnostic to VLA architecture, as we demonstrate by instantiating Steerable Policies with two popular VLA frameworks (OpenVLA and $\pi_{0.5}$ [10, 11]); and **(2)** novel methods for using VLMs' embodied reasoning and in-context learning to hierarchically control our VLA to solve challenging tasks. We demonstrate that the steerability of low-level VLAs significantly improves the use of high-level VLMs' pretrained capabilities, resulting in better robot generalization.

II. Related Works

Vision-language-action models. As in vision and language, robotics has moved towards using large foundation models [12]. A popular method is to fine-tune general VLMs into *vision-language-action policies (VLAs)* [13, 10], transferring the base VLM's comprehensive pretrained representations to learn robot tasks from demonstrations. Prior works explore many variations of this recipe, such as altering action representations [13, 10, 14–16, 11, 17] or training VLAs to perform embodied reasoning [8, 9, 18]. Our approach differs from such works in that we aim to improve policy steerability via augmentation of language prompts, with the end goal of making better use of foundation model capabilities in hierarchical approaches.

Training for steerability. Past works often improve policy steerability by training on synthetic language labels that are more diverse [19], detailed [20, 21], or composable [22, 23]. They can also be trained on non-text steering modalities, like gripper traces [24, 25] or visual marks [26].

In contrast, our method trains on *many* input types, rather than finding a one-size-fits-all steering modality. Additionally, all our steering commands are expressed in text tokens, allowing our VLA to interface with generative VLMs more easily than other models trained on many prompt modalities, such as OmniVLA [27]. Similar to our method, MolmoAct trains on standard text with gripper trace CoTs, enabling inference-time steering via two modalities (changing either the trace *or* the language) [28]. However, unlike our work, they limit language steering to task-level prompts only (no other abstractions, like subtasks, motions, or points, as we use). Additionally, MolmoAct's gripper trace steering can only be used *in conjunction* with text commands, which can be a drawback when text introduces adverse biases (e.g., if the policy systematically misidentifies an object by name). This disadvantage is not shared by our policy.

In summary, the core novelty of our Steerable Policies is that they accept steering commands spanning a spectrum of abstractions, not just one or two, as MolmoAct and other prior works do. In particular, giving high-level VLMs the flexibility to *choose* how to steer the low-level VLAs is necessary for improved generalization. We find in Sec. VI-A that this is critical for performance, as the effectiveness of different steering abstractions depends heavily on the task setting.

Synthetic data for model controllability. Our method of training policies on synthetic language parallels a trend in text-

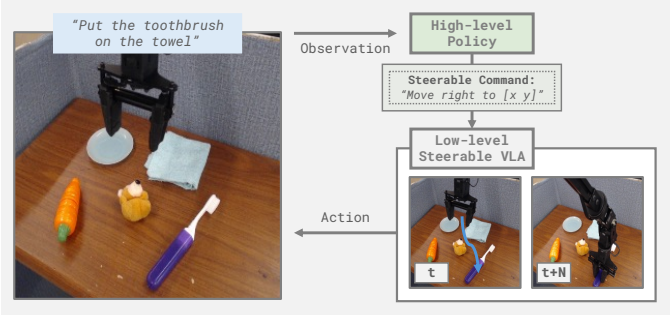


Fig. 2: The hierarchical policy inference loop, where a high-level model sends commands to the low-level Steerable Policy.

to-image generation: while image corpora often contain the diversity needed for detailed image generation, the accompanying captions are too imprecise for fine-grained text controllability, necessitating detailed synthetic labels [29]. Similarly, our core insight is that *robot datasets already contain diverse behaviors, but their text labels are too sparse, biased, and succinct to induce desirable actions for novel tasks*. While steerability in the image domain is valuable for aligning with user preferences, we aim to make VLAs more steerable to better interface with high-level policies, enabling improved use of their pretrained capabilities for solving new control tasks.

Hierarchical robot learning. One way to improve generalization is to hierarchically decompose robot tasks into atomic skills [30]. These techniques broadly divide action prediction into (1) fast and reactive control and (2) slow and methodical reasoning or planning, akin to the System I and II model of human cognition [31]. In robot learning, these methods either are trained end-to-end (e.g., policies with an intermediate "chain-of-thought reasoning" [8, 11, 28, 32]) or, as we focus on in Sec. V, are factorized into separate models [1, 33, 2, 24].

Several works use off-the-shelf VLMs as the high-level policy [1, 34–41], while others finetune them to improve their alignment with a given domain or low-level policy [33, 42–44, 24, 28, 45, 46]. However, these works usually select a *single* modality to interface between high and low levels, such as subtask-level language prompts [1, 36, 37, 44, 45, 39, 35, 38, 43, 46] or grounded representations [24, 28].

Steerable Policies enable a fundamentally new capability absent from these works: the ability to reason about and choose which abstraction to use for steering VLAs. This not only yields high performance (Sec. VI-A), but also provides novel ways to apply broad VLM capabilities – scene understanding, reasoning, and in-context learning – to robotics (Sec. VI-C). Of course, non-steerable VLAs only accept a single type of (usually text) prompt, and do not reap the same benefits.

III. Preliminaries

Vision-language-action models. Given task prompt l and observations o (e.g., proprioception and images), *behavioral cloning (BC)* learns a policy $\pi(a|o,l)$ that solves tasks by matching the behavior distribution of expert demonstrators. *Vision-language-action models* are pretrained *vision-language models* fine-tuned into π. Our Steerable Policy (Sec. IV) is a VLA, though trained to accept diverse steering modalities.

Hierarchical policies. One way to improve the generalization of learned policies is with hierarchical methods. This factorizes robot instruction-following into two steps: first, a high-level policy takes the overall task l and observation o to output an appropriate intermediate subtask, plan, or goal g; then, the low-level policy samples actions a conditioned on g and o. This is the framework we adopt (Fig. 2): our Steerable Policy acts as the low-level controller (where g are steering commands), while we explore the novel high-level policy capabilities they enable in Sec. V.

IV. Steerable Vision-Language-Action Policies

We now introduce Steerable Policies. We propose training VLAs on ***steering commands*** that span many abstractions, such as subtasks, motions, or grounded coordinates (Sec. IV-A). After generating them at scale (Sec. IV-B), these prompts randomly replace the standard task-level labels used for BC. Training on steering commands yields a policy that can execute diverse compositional skills when prompted via a range of modalities (Sec. VI-A), in contrast to past works that use single steering inputs [24, 25, 22]. This section solely focuses on training *low-level* Steerable Policies; then, in Sec. V, we discuss how to use these VLAs with *high-level* VLM policies that perform embodied reasoning or in-context learning to choose appropriate steering commands.

A. Styles of Steering Commands

To obtain VLAs that can exhibit diverse behavior when instructed with a wide range of abstractions, we train on a diverse mix of different ***steering command styles***, satisfying several desiderata: commands should be (1) *versatile* for inducing a range of generalizable behaviors for solving novel tasks; (2) *conducive* to being issued by human operators, high-level reasoners, and in-context learning VLMs; and (3) *scalable* for synthetic labeling – i.e., they can be automatically generated from robot trajectories by using foundation models. The categories of steering commands that we train on are as follows (see Fig. 1 and App. A for more examples):

1) ***Tasks***: The default labels used for training VLAs. They allow Steerable Policies to also solve tasks that standard VLAs can. E.g., `"put the carrot in the pot"`.
2) ***Subtasks***: These help compose existing semantic skills for novel tasks. E.g., `"reach for the carrot"`.
3) ***Atomic motions***: These let the VLA follow granular movements in language, without referencing the semantic contents of the scene. E.g., `"move left"` or `"open gripper"`.
4) ***Gripper traces***: These provide a list of pixels for the gripper to follow. E.g., `"move from` $[x_1, y_1]$ `to` $[x_2, y_2]$`"`.
5) ***Points***: These indicate positions of objects or locations of interest. E.g., `"lift above <pot position>"` or `"grasp at <carrot position>"`. This is subtly distinct from gripper trace commands, as pointing does *not* necessarily indicate the exact pixels the gripper must move to, merely visual positions relevant to the task.
6) ***Combinations***: Hybrids of these styles. E.g., `"move left from` $[x, y]$ `to the carrot at <carrot position>"`.

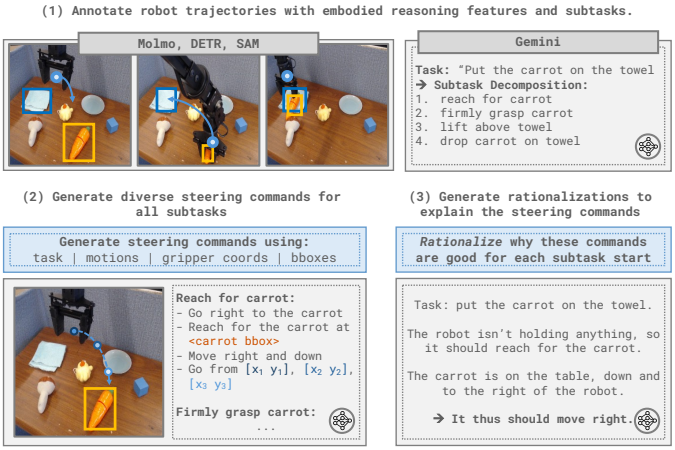


Fig. 3: Our automated pipeline for annotating robot data with synthetic steering commands at scale. **1:** We use a suite of foundation models to extract subtasks and grounded features (bounding boxes, motions, and gripper traces) from each trajectory. **2:** We query a VLM to generate diverse steering commands for training Steerable Policies. These commands may reference features extracted in the first step, which we provide in the prompt. **3:** To train high-level embodied reasoners, we also generate *rationalizations* for why particular commands are appropriate for given observations (Sec. V-A).

The last three styles can reference 2D pixel coordinates, which VLMs can easily specify [47, 18]. They also allow fine-grained steering when language is insufficient: e.g., the policy might fail to pick up some out-of-distribution object specified by name, but would succeed if told to "`pick up the object at` $[x, y]$". Likewise, if there are multiple of instances an object (making language underspecified), the policy can determine the correct one to interact with from a pointing command.

B. Generating Steering Commands at Scale

The default task commands found in large robot datasets are usually human-annotated. While this yields the gold standard in terms of data quality, acquiring these labels at scale is expensive and labor-intensive, especially when language commands reference grounded features, such as pixel coordinates. We thus turn to synthetically-generated annotations.

Our pipeline for producing steering commands is shown in Fig. 3. We first extract grounded features from a robot dataset, then compile them into subtasks. Specifically, we follow Zawalski et al. [8] to programmatically extract motion language, then use Molmo to extract task-relevant object names and map them to segmentation masks [47]. Next, we use SAM2 to track and propagate these masks across the entire trajectory video [48], thereby producing temporally-consistent open-vocabulary bounding boxes. Separately, we call DETR to extract the robot's gripper traces [49]. Finally, given the overall task, motions, and objects, we query Gemini 2.0 [50] to break the episode into semantic subtasks.

After grounded feature extraction and subtask decomposition, we query Gemini *again* to restate each subtask into equivalent steering commands of all styles in Sec. IV-A). Gripper traces, motions, and object centroids are provided in the prompt, as these grounded features are useful for generating some steering commands. This pipeline allows us to expand from 38k "standard" Bridge task-level language labels to 206k subtasks and nearly 2M total steering commands. See App. A for full details and all prompts used.

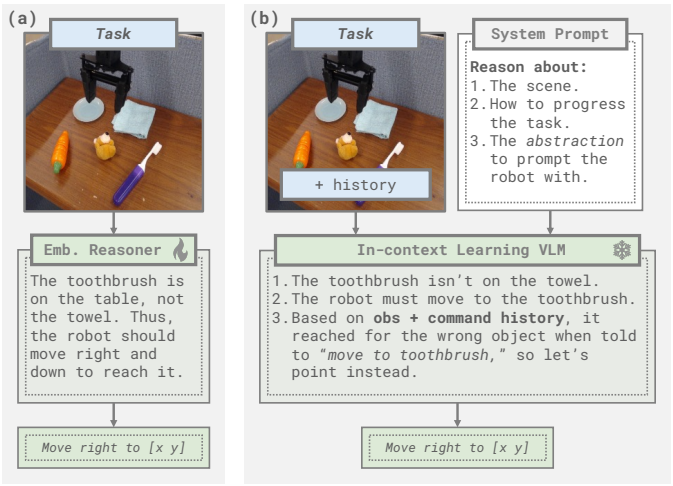


Fig. 4: Our two novel high-level policies (Sec. V). **(a)** fine-tunes a VLM into an embodied reasoner that issues steering commands, while **(b)** queries an off-the-shelf VLM to determine appropriate commands via in-context reasoning.

The generated commands are used to train Steerable Policies by simply replacing the standard language during BC with a corresponding steering command (sampled uniformly at random across all commands for each frame). The resulting VLA can thus follow all command styles in Sec. IV-A necessary to produce generalizable behaviors, which we confirm in Sec. VI-A. In particular, we show that different command styles are suited to different tasks and situations, suggesting Steerable Polices generalize better. Indeed, we also find that choosing steering commands intelligently yields a substantial performance boost over a non-steerable baseline.

V. Hierarchical Methods with Steerable Policies

To fully realize the benefits of Steerable Policies, they must be integrated with VLMs that understand their affordances. We introduce two *high-level* VLM policies that effectively leverage semantic knowledge and physical reasoning to issue steering commands. First, we investigate how embodied reasoning training can aid lightweight open-source VLMs to leverage their pretrained representations for controlling Steerable Policies. Second, we use the zero-shot in-context learning abilities of off-the-shelf VLMs for multi-step robotic problem-solving. Both methods are depicted in Fig. 4.

A. Training High-level Embodied Reasoners

Our first method trains a high-level embodied reasoning model to decompose tasks into steering commands, as shown in Fig. 4 (a). Specifically, we fine-tune a VLM to output appropriate steering commands for accomplishing a given task based on a task instruction and the current observation. Note that our command generation pipeline (Sec. IV-B) provides a mapping from tasks to appropriate commands, yielding the exact supervision necessary to train this model.

To make better use of the base VLMs' reasoning abilities and improve generalization, we also produce post-hoc rationales for *why* each command is appropriate. For each subtask in every trajectory, we query Gemini with its starting frame, asking the VLM to explain why that subtask is needed to make progress (Fig. 3). See App. A-B for the full prompt used and more details. We then train the high-level policy via next-token prediction to reason about what the robot should do in each frame. During inference, the model autoregressively predicts a reasoning followed by a corresponding steering command, which our Steerable Policy follows for $N = 5$ environment steps before re-querying the high-level reasoner.

We expect this approach to be particularly effective for three main reasons. First, as the high-level reasoner's learning objective (mapping from task language to natural language reasonings and steering commands) is similar to the base VLM's pretraining objective, we expect it to readily transfer its generalizable representations to this new task. Second, since the low-level VLA only attends to steering commands, it learns fewer spurious biases between the task, reasoning, and action, compared to most unfactorized end-to-end models. Finally, since the high-level reasoner is queried less frequently than the VLA, it speeds up inference compared to standard embodied reasoning, even without any compilation techniques [51, 52].

B. In-context Reasoning for Choosing Steering Abstractions

Steerable Policies also enable a novel functionality: they permit many command styles, letting high-level models *choose* which level of abstraction is best for inducing behaviors that progress a given task. Thus, in addition to reasoning about *what* the robot should do, hierarchical systems using Steerable Policies have the added dimension of reasoning about *how* to get the robot to do it. Our approach in Sec. V-A fine-tunes a model to map tasks to steering commands, but not necessarily those of the optimal style. Doing so requires (1) intuition about the strengths and weaknesses of each command style and (2) the ability to learn when each style is effective on the fly. We thus hypothesize that *off-the-shelf* VLMs would excel at selecting command styles, due to their zero-shot proficiency with in-context learning and reasoning.

We test this idea with the implementation shown in Fig. 4 (b). We query an API-based off-the-shelf VLM with the robot's observation and an overall task. The VLM is told to issue commands to our Steerable Policy, based on examples of all the command styles and brief descriptions of their strengths and weaknesses. The model receives a history of images and executed commands in context, and is asked to (1) parse the current scene, (2) determine what the robot should do next, and (3) reason about the best level of abstraction for commanding the policy. Finally, it predicts a steering command for the Steerable Policy to follow for $N = 20$ steps before the high-level VLM is re-queried. See App. B for more details.

Beyond emitting steering commands at a range of abstraction styles, our approach has several key innovations. Most significantly, the VLM can use *in-context learning* to adapt its command choices, since it receives a sequential history of past observations and commands it has chosen. After selecting a steering abstraction, the VLM can observe the robot's resulting behavior, then adjust the abstraction or command as needed. Critically, because in-context data points are produced by the VLM, it does not need hand-crafted examples of robot behaviors. Note that as in-context learning is used here to predict steering commands, not robot actions, the VLM's task is thus akin to the "standard" vision-language in-context learning that VLMs are known to excel at. This obviates domain-specialized training or structured scene descriptions needed in prior robotic in-context learning works [53, 54].

Another benefit of our approach is that the VLM can use its scene understanding in conjunction with descriptions of each command style to reason about optimal steering abstractions. For instance, the VLM might produce visual inferences such as "the scene is cluttered," leading to the selection of a pointing or motion command which allows for improved specificity.

Naturally, as the above benefits stem from choosing steering abstractions, they are *not* applicable to prior methods where VLMs control standard policies that accept only a single conditioning modality (e.g., task language [13, 10, 22] or traces [24–26]). Only Steerable Policies can fully bring these VLM capabilities to bear for robotic problem-solving.

VI. Experiments

Research questions. We now evaluate our Steerable Policies, both individually and as part of our two hierarchical control methods. We aim to answer: **(1)** Can steering commands effectively induce diverse compositional behaviors in Steerable Policies, yielding improved task performance? **(2)** How can Steerable Policies enable trained high-level VLMs to leverage embodied reasoning data for generalization? **(3)** How can Steerable Policies let us better apply in-context learning, scene understanding, and reasoning competencies in off-the-shelf VLMs for solving long-horizon tasks?

Experimental setup. We train Steerable Policies on the Bridge WidowX real-world robot manipulation dataset [4, 5, 55], allowing for evaluations consistent with past generalist policies [56, 10, 8, 9]. Following these works, we adapt the OpenVLA codebase to train our Steerable Policy (and embodied reasoner) with the standard Prismatic VLM 7B architecture [57, 10], but modified to map *steering commands* (rather than standard task labels) and third-person RGB images to end effector actions. To show that Steerable Policies are applicable to many VLAs, we also fine-tune one from $\pi_{0.5}$ [11], which we use in Sec. VI-B. See App. F for more details.

For experiments in Sec. VI-A and Sec. VI-B, we measure performance over common generalization axes [10, 8, 9]: in-distribution, motion, spatial, and semantic generalization. When evaluating our in-context learning high-level policy formulation in Sec. VI-C, we additionally include unseen longer-horizon tasks. See App. C for full task details.

A. Diverse Steering Commands Induce Performant and Compositional Behaviors in Steerable Policies

To improve the performance of learned hierarchical systems, the underlying policies must respond to a wider range of com-

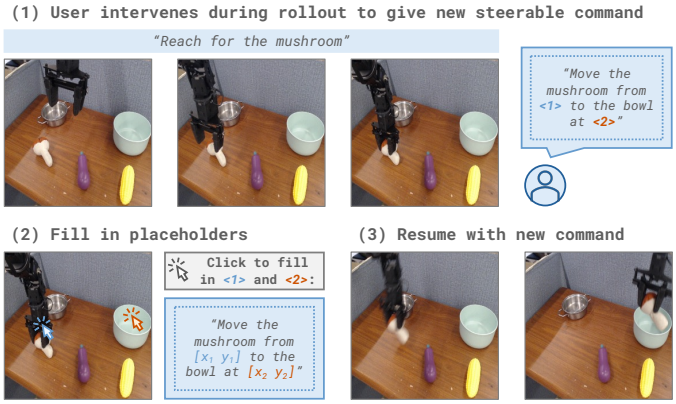


Fig. 5: Interactive interface for querying humans for oracle steering commands. **1:** The operator can interrupt the rollout to issue a new steering command. To facilitate giving commands with pixel coordinates, they can add textual placeholder markers. **2:** If any are given, a GUI is opened displaying the current robot observation, allowing the user to click to fill the markers in. **3:** Finally, the rollout resumes with the new command.

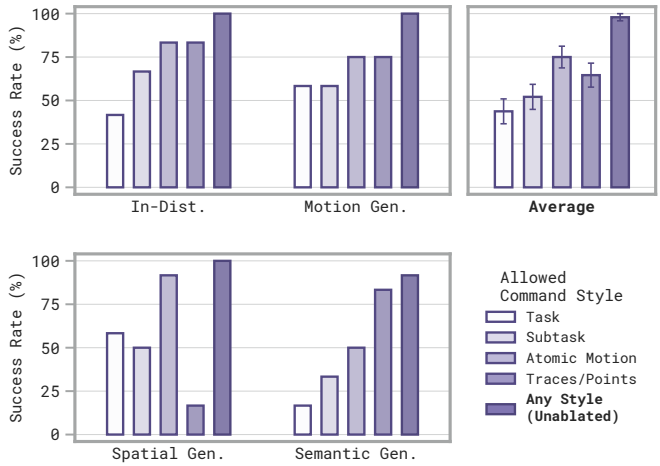


Fig. 6: Allowing the oracle human user to issue *any* command style to our Steerable Policy **nearly saturates performance** on Bridge. By restricting the user to each style alone, we find each one is suited to different task types. All individual styles are better than directly providing the task-level label that is used by regular VLAs. Error bars denote $\pm$1StdErr.

mands beyond standard task-level language. Our first experiment evaluates whether our proposed steering commands are sufficiently expressive to induce the compositional behaviors necessary for Steerable Policies to solve a range of novel tasks.

Specifically, we establish an achievable upper bound on performance by having a human act as an "oracle" high-level policy. This experiment serves as a didactic proof of existence showing that, given human-level scene understanding and reasoning, steering commands enable a Steerable Policy to attain high task performance. Following Shi et al. [33], the user may intervene by modifying the policy's free-form steering command as needed (Fig. 5). However, they must wait ≥ 2 seconds between interventions to reflect practical hierarchical systems that query the high-level policy less frequently than the low-level. This also prevents the user from "fully teleoperating" the robot by repeatedly changing the command.

We additionally repeat this experiment while restricting the user to a single command style at a time, including the 'task-level" prompts used by standard VLAs. This isolates whether *individual* steering modalities are sufficient for strong performance, and clarifies *when* each style is most effective.

Our results demonstrate that **a human oracle issuing unrestricted steering commands can solve all tasks at nearly 100% success rate** (Fig. 6). Even when constrained to a single command style, the user consistently outperforms task-level language alone. Additionally, we find that no single style dominates across settings: each exhibits complementary strengths and weaknesses. Optimal performance empirically requires access to a spectrum of prompt abstraction, supporting the claim that no one-size-fits-all steering modality exists.

We observe several trends for when each style is effective. First, ***trace/point*** commands dominate in semantic generalization, as pointing allows the policy to know which out-of-distribution objects to interact with, even if it fails to identify them by name. Next, ***atomic motion*** commands are best in tasks that reference spatial relations. When following motion commands, the policy remains on a manifold of "reasonable" actions – e.g., when prompted to "move left," it servos left *towards an object*, rather than moving left unconditionally (see App. E-B for more examples). As a result, when using a single steering modality, motions are the most effective. Finally, while ***task/subtask*** commands never perform best in any evaluation, they are especially reliable for *in-distribution* parts of an overall task (e.g., reaching for non-novel objects).

Overall, we find our Steerable Policy to be both responsive to different command styles and highly performant, particularly when given suitable steering commands leveraging *all* command styles. However, selecting effective commands requires reasoning about the task, the scene, and the VLA's capabilities. This motivates the novel hierarchical methods presented in Sec. V, which we evaluate below.

B. Steerable Policies Enable VLMs to Effectively Use Embodied Reasoning Training

We now evaluate if fine-tuning a VLM into a high-level policy is an effective way to leverage embodied reasoning to control Steerable Policies. Since our method involves fine-tuning the high-level to produce embodied reasonings and intermediate goals, we compare against three prior methods that use reasoning data to improve VLAs (all built on OpenVLA): ***Embodied Chain-of-Thought Reasoning (ECoT)*** [8] and the two ***ECoT-Lite*** variants, which learn representations from robot reasonings, but do not generate them during inference [9]. We also include a non-reasoning ablation, where the fine-tuned VLM outputs steering commands directly. Finally, we compare against equivalent non-reasoning ***"standard" OpenVLA and*** $\pi_{0.5}$ ***baselines*** [10, 11]. All methods are evaluated on the same Bridge tasks used in ECoT-Lite, which were chosen to probe several axes of generalization. For fairness, we control for training demonstrations and architectures; the policies differ only in how embodied reasoning is incorporated.

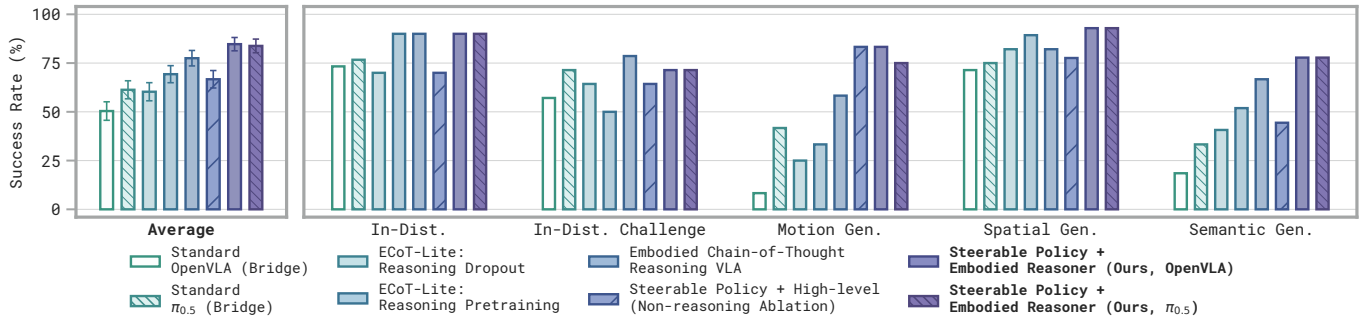


Fig. 7: Our approach of controlling Steerable Policies with learned high-level embodied reasoning VLMs outperforms five baselines: the equivalent standard Bridge OpenVLA and $\pi_{0.5}$ [10, 11], the Reasoning Pretraining and Dropout ECoT-Lite methods [9], and full Embodied Chain-of-Thought Reasoning [8]. Error bars denote $\pm$1StdErr. We adopt the same Bridge task suite as ECoT-Lite [9], enabling a direct comparison with other methods for training VLAs with embodied reasoning data.

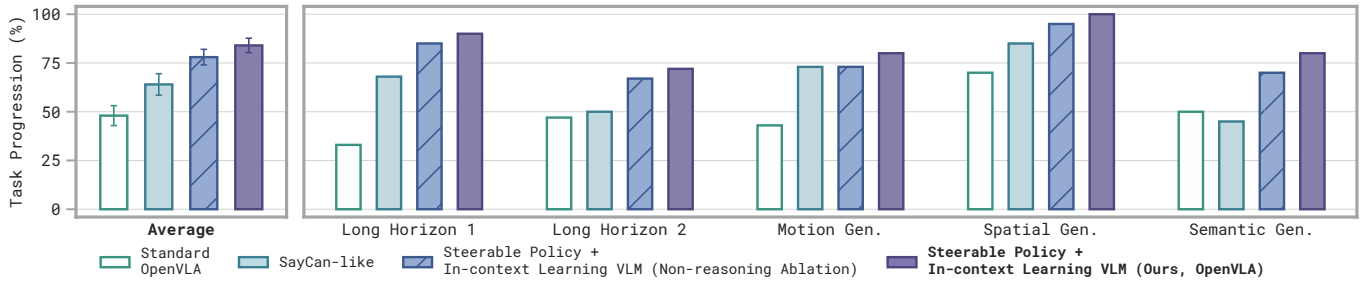


Fig. 8: In-context learning VLMs can effectively select abstractions for instructing our Steerable Policy. Error bars denote $\pm$1StdErr. Steerability allows VLMs to better apply their reasoning and in-context learning skills, in ways the standard SayCan-like paradigm (where the VLM interfaces with the VLA via subtasks alone) fails to take advantage of (Sec. VI-C). To better highlight these capabilities, we measure task progression on challenging *multi-step* Bridge tasks.

As shown in Fig. 7, *our approach outperforms all baselines*. The gains are most pronounced on motion and semantic generalization tasks, likely because atomic motion, trace, and pointing commands allow the generalizable reasoner to reliably control the Steerable Policy under distribution shift.

High-level VLM reasoning benefits Steerable Policies based on different VLA architectures. While the "standard" Bridge $\pi_{0.5}$ baseline outperforms the OpenVLA equivalent (likely due to having better base representations for control), applying our method to $\pi_{0.5}$- *or* OpenVLA-based Steerable Policies still outperforms both baselines, showing that *steerability unlocks performance gains from VLM reasoning, even for modern VLA architectures*. Both Steerable Policies achieve similar performance when using our approach.

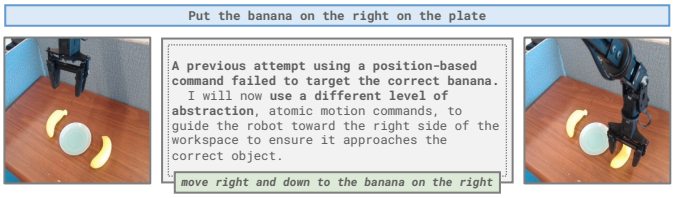
Fine-tuned high-level VLMs benefit from reasoning. While not as effective as our full approach, ablating reasoning from our method still greatly outperforms standard OpenVLA and is on-par with ECoT-Lite, despite being trained on the same robot interactions. This suggests that there are performance benefits to hierarchical control schemes with Steerable Policies over end-to-end policies, *even if the high-level is not explicitly trained to reason*. Still, we find that embodied reasoning is important for generalization and performance. Note that VLMs' off-the-shelf reasoning abilities can be used to achieve the same benefit, which we discuss below.

C. Steerable Policies Unlock In-context Reasoning in VLMs

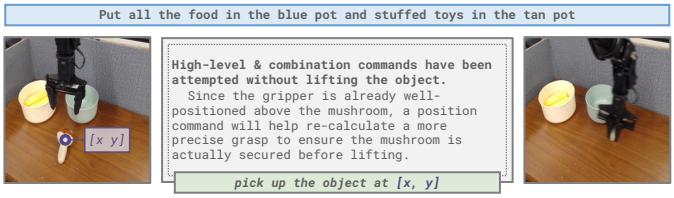
Finally, we investigate if off-the-shelf VLMs can better use their in-context reasoning faculties when interfacing with Steerable Policies. We define a distinct suite of multi-step tasks that demand reasoning over long-horizon executions. See App. C-C for task details and rubrics.

The key advantage of using Steerable Policies is that they allow high-level VLMs to reason about and flexibly choose the best abstraction for prompting low-level VLAs. Our primary baseline is therefore a ***SayCan-like*** [1] method, representing a standard approach where an off-the-shelf VLM commands the VLA through subtask language alone. For fairness, this baseline uses the same history and system prompt as our method, differing only in that the VLM is restricted to a single level of steering abstraction. We expect this to hinder the impact of in-context reasoning, lowering performance. We also include a ***standard Bridge OpenVLA*** comparison to assess if our tasks are sufficiently challenging for non-hierarchical policies. Finally, to isolate the role of explicit reasoning, we include a ***non-reasoning ablation*** where the VLM outputs steering commands without any intermediate generation. All hierarchical methods use Gemini 3.0 [50] as the VLM. More details and prompts for all methods are shown in App. B.

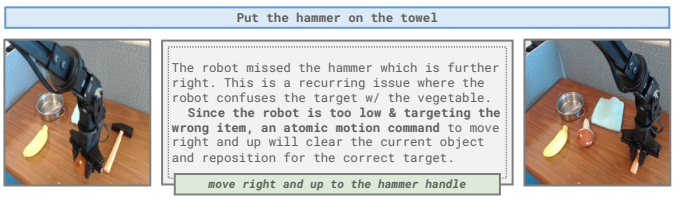
As shown in Fig. 8, our approach universally outperforms OpenVLA and SayCan-like baselines. Since the latter differs only in being restricted to subtask-level prompts, the performance gap primarily reflects the benefit of allowing the VLM to access the full spectrum of steering abstractions during in-context reasoning. Our method also outperforms the non-reasoning ablation, though the gap is smaller. This suggests that, while explicit reasoning is helpful, in-context learning



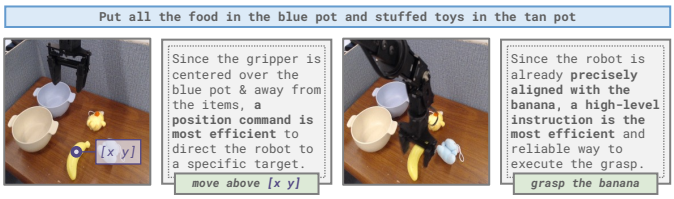
(a) The VLM uses in-context learning to "discover" what steering abstraction is best. A pointing command leads to incorrect behavior, so the VLM employs a motion command instead.



(b) The VLM uses in-context learning to adjust the abstraction when task progress stalls. Subtask (`"lift the mushroom up"`) and motion (`"move down and close gripper onto the mushroom"`) commands were ineffective, so the VLM points instead.



(c) The VLM demonstrates fine-grained semantic and physical understanding to issue a corrective command.



(d) Given an observation, the VLM reasons about which steering abstraction is most appropriate, based on each one's strengths.

Fig. 9: Example reasonings when using an in-context learning VLM for controlling our Steerable Policies.

VLMs are already adept at issuing effective steering commands without intermediate "thinking." We further observe that using in-context reasoning to steer our VLA leads to several qualitative advantages, shown below and in Fig. 9.

In-context learning enables corrective steering. Even when the VLM initially picks poor steering commands, we find it can use in-context learning to improve commands by changing the abstraction level. This is especially salient when a command leads to an error (Fig. 9a) or is ineffective and fails to progress the task (Fig. 9b). In either case, the VLM reasons over history to learn the affordances of the VLA and change its prompting strategy to complete the task.

Steerability better applies VLM understanding. Interfacing with a Steerable Policy allows the VLM to translate its scene understanding and in-context history into actionable corrections that are difficult to induce in a standard VLA accepting task-level language alone. In Fig. 9c, the VLM uses *semantic* understanding to observe that the robot grasps the wrong object when instructed to reach for the hammer. It further reasons with granular *physical* understanding, recognizing that the end effector must have sufficient clearance to disengage the current object and move above the correct one. It uses this understanding in conjunction with past behavioral and semantic failure modes to ground its final steering command: `"move right and up to the hammer handle"`.

Crucially, the VLM often cannot capitalize on such insights with subtask commands alone. A common failure mode of the SayCan-like baseline is that, even when the VLM correctly diagnoses errors, subtasks lack the specificity required to resolve them. In the same example (Fig. 9c), the VLM may recognize that the robot has confused a mushroom for the hammer and issue `"drop the mushroom"`. However, since a non-steerable VLA cannot understand motion commands such as `"move up and right"`, the VLM can only reissue the subtask command `"reach for the hammer"`, causing the robot to repeat the same mistake. In short, even if the VLM has the capacity for nuanced scene understanding, it cannot act on this knowledge with subtask commands alone, requiring a low-level Steerable Policy instead (see App. E-C).

VLMs can reason about steering abstractions. Beyond reasoning about *what* the robot should do, VLMs can also reason about *how* to best induce the desired behavior (i.e., what steering style to use). In Fig. 9d, when reaching for a banana in a cluttered scene, the VLM determines that a pointing command is the least ambiguous option and issues `"move above <banana position>"`. After observing that the robot is correctly above the banana, it switches to a higher-level instruction, `"grasp the banana"`. This demonstrates that VLMs can dynamically select and sequence steering abstractions based on the current state.

VII. Discussion and Future Work

We introduce Steerable Policies: VLAs that support a wide range of command abstractions including subtasks, motions, and grounded pixel coordinates. We show how improved steerability enables better transfer of VLM capabilities into robotics by (1) fine-tuning high-level VLMs on embodied reasoning data to solve generalization tasks with our Steerable Policy, and (2) using the in-context learning abilities of off-the-shelf VLMs for multi-step robotic problem-solving.

As robot datasets have trended toward scaling their behavioral diversity, we expect Steerable Policies to only increase in applicability. The compositionality afforded by steering also becomes more important as task complexity increases, e.g., when moving to open-world settings. To support this, VLMs require an even better grasp of the affordances provided by Steerable Policies. Beyond zero-shot prompting, VLMs could *learn* these affordances via reinforcement learning, as rollouts would allow models to elucidate when each steering style is effective. This may also provide the data needed for cross-task in-context learning, where the VLM transfers its understanding of affordances to new situations. We hope that Steerable Policies will represent a step toward finally bringing the "disembodied" capabilities of VLMs to the physical world.

Acknowledgements

We thank Qiyang Li for assistance with the teaser diagram. We also thank Ameesh Shah, Arhan Jain, and members of the RAIL Lab for insightful discussions. This research was partly supported by ONR N00014-25-1-2060, NSF IIS-2246811, and DARPA ANSR, with additional support from Google and the NVIDIA Academic Grant Program.

References

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

[2] Lucy Xiaoyang Shi, Brian Ichter, Michael Equi, Liyiming Ke, Karl Pertsch, Quan Vuong, James Tanner, Anna Walling, Haohuan Wang, Niccolo Fusai, Adrian Li-Bell, Danny Driess, Lachy Groom, Sergey Levine, and Chelsea Finn. Hi robot: Open-ended instruction following with hierarchical vision-language-action models, 2025. URL https://arxiv.org/abs/2502.19417.

[3] Selma Wanna, Agnes Luhtaru, Jonathan Salfity, Ryan Barron, Juston Moore, Cynthia Matuszek, and Mitch Pryor. Limited linguistic diversity in embodied ai datasets, 2026. URL https://arxiv.org/abs/2601.03136.

[4] Frederik Ebert, Yanlai Yang, Karl Schmeckpeper, Bernadette Bucher, Georgios Georgakis, Kostas Daniilidis, Chelsea Finn, and Sergey Levine. Bridge data: Boosting generalization of robotic skills with cross-domain datasets, 2021.

[5] Homer Walke, Kevin Black, Abraham Lee, Moo Jin Kim, Max Du, Chongyi Zheng, Tony Zhao, Philippe Hansen-Estruch, Quan Vuong, Andre He, Vivek Myers, Kuan Fang, Chelsea Finn, and Sergey Levine. Bridgedata v2: A dataset for robot learning at scale. In *Conference on Robot Learning (CoRL)*, 2023.

[6] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models, 2023.

[7] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners, 2023.

[8] Michał Zawalski, William Chen, Karl Pertsch, Oier Mees, Chelsea Finn, and Sergey Levine. Robotic control via embodied chain-of-thought reasoning. In *Conference on Robot Learning*, 2024.

[9] William Chen, Suneel Belkhale, Suvir Mirchandani, Oier Mees, Danny Driess, Karl Pertsch, and Sergey Levine. Training strategies for efficient embodied reasoning, 2025.

[10] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Pannag Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. Openvla: An open-source vision-language-action model. 2024.

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

[12] Rishi Bommasani, Drew A. Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S. Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, Erik Brynjolfsson, Shyamal Buch, Dallas Card, Rodrigo Castellon, Niladri Chatterji, Annie Chen, Kathleen Creel, Jared Quincy Davis, Dora Demszky, Chris Donahue, Moussa Doumbouya, Esin Durmus, Stefano Ermon, John Etchemendy, Kawin Ethayarajh, Li Fei-Fei, Chelsea Finn, Trevor Gale, Lauren Gillespie, Karan Goel, Noah Goodman, Shelby Grossman, Neel Guha, Tatsunori Hashimoto, Peter Henderson, John Hewitt, Daniel E. Ho, Jenny Hong, Kyle Hsu, Jing Huang, Thomas Icard, Saahil Jain, Dan Jurafsky, Pratyusha Kalluri, Siddharth Karamcheti, Geoff Keeling, Fereshte Khani, Omar Khattab, Pang Wei Koh, Mark Krass, Ranjay Krishna, Rohith Kuditipudi, Ananya Kumar, Faisal Ladhak, Mina Lee, Tony Lee, Jure Leskovec, Isabelle Levent, Xiang Lisa Li, Xuechen Li, Tengyu Ma, Ali Malik, Christopher D. Manning, Suvir Mirchandani, Eric Mitchell, Zanele Munyikwa, Suraj Nair, Avanika Narayan, Deepak Narayanan, Ben Newman, Allen Nie, Juan Carlos Niebles, Hamed Nilforoshan, Julian Nyarko, Giray Ogut, Laurel Orr, Isabel Papadimitriou, Joon Sung Park, Chris Piech, Eva Portelance, Christopher Potts, Aditi Raghunathan, Rob Reich, Hongyu Ren, Frieda Rong, Yusuf Roohani, Camilo Ruiz, Jack Ryan, Christopher Ré, Dorsa Sadigh, Shiori Sagawa, Keshav Santhanam, Andy Shih, Krishnan Srinivasan, Alex Tamkin, Rohan Taori, Armin W. Thomas, Florian Tramèr, Rose E. Wang, William Wang, Bohan Wu, Jiajun Wu, Yuhuai Wu, Sang Michael Xie, Michihiro Yasunaga, Jiaxuan You, Matei Zaharia, Michael Zhang, Tianyi Zhang, Xikun Zhang, Yuhui Zhang, Lucia Zheng, Kaitlyn Zhou, and Percy Liang. On the opportunities and risks of foundation

models, 2022.

[13] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, Pete Florence, Chuyuan Fu, Montse Gonzalez Arenas, Keerthana Gopalakrishnan, Kehang Han, Karol Hausman, Alexander Herzog, Jasmine Hsu, Brian Ichter, Alex Irpan, Nikhil Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Lisa Lee, Tsang-Wei Edward Lee, Sergey Levine, Yao Lu, Henryk Michalewski, Igor Mordatch, Karl Pertsch, Kanishka Rao, Krista Reymann, Michael Ryoo, Grecia Salazar, Pannag Sanketi, Pierre Sermanet, Jaspiar Singh, Anikait Singh, Radu Soricut, Huong Tran, Vincent Vanhoucke, Quan Vuong, Ayzaan Wahid, Stefan Welker, Paul Wohlhart, Jialin Wu, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. Rt-2: Vision-language-action models transfer web knowledge to robotic control, 2023.

[14] Karl Pertsch, Kyle Stachowicz, Brian Ichter, Danny Driess, Suraj Nair, Quan Vuong, Oier Mees, Chelsea Finn, and Sergey Levine. Fast: Efficient action tokenization for vision-language-action models, 2025. URL https://arxiv.org/abs/2501.09747.

[15] Suneel Belkhale and Dorsa Sadigh. Minivla: A better vla with a smaller footprint, 2024. URL https://ai.stanford.edu/blog/minivla/.

[16] Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A vision-language-action flow model for general robot control, 2024. URL https://arxiv.org/abs/2410.24164.

[17] Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-tuning vision-language-action models: Optimizing speed and success, 2025. URL https://arxiv.org/abs/2502.19645.

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

[19] Catherine Glossop, William Chen, Arjun Bhorkar, Dhruv Shah, and Sergey Levine. Cast: Counterfactual labels improve instruction following in vision-language-action models, 2025. URL https://arxiv.org/abs/2508.13446.

[20] Ted Xiao, Harris Chan, Pierre Sermanet, Ayzaan Wahid, Anthony Brohan, Karol Hausman, Sergey Levine, and Jonathan Tompson. Robotic skill acquisition via instruction augmentation with vision-language models, 2022.

[21] Laura Smith, Alex Irpan, Montserrat Gonzalez Arenas, Sean Kirmani, Dmitry Kalashnikov, Dhruv Shah, and Ted Xiao. Steer: Flexible robotic manipulation via dense language grounding, 2024. URL https://arxiv.org/abs/2411.03409.

[22] Jesse Zhang, Karl Pertsch, Jiahui Zhang, and Joseph J. Lim. Sprint: Scalable policy pre-training via language instruction relabeling, 2024. URL https://arxiv.org/abs/2306.11886.

[23] Corey Lynch, Ayzaan Wahid, Jonathan Tompson, Tianli Ding, James Betker, Robert Baruch, Travis Armstrong, and Pete Florence. Interactive language: Talking to robots in real time, 2022.

[24] Yi Li, Yuquan Deng, Jesse Zhang, Joel Jang, Marius Memmel, Raymond Yu, Caelan Reed Garrett, Fabio Ramos, Dieter Fox, Anqi Li, Abhishek Gupta, and Ankit Goyal. Hamster: Hierarchical action models for open-world robot manipulation, 2025. URL https://arxiv.org/abs/2502.05485.

[25] Jiayuan Gu, Sean Kirmani, Paul Wohlhart, Yao Lu, Montserrat Gonzalez Arenas, Kanishka Rao, Wenhao Yu, Chuyuan Fu, Keerthana Gopalakrishnan, Zhuo Xu, Priya Sundaresan, Peng Xu, Hao Su, Karol Hausman, Chelsea Finn, Quan Vuong, and Ted Xiao. Rt-trajectory: Robotic task generalization via hindsight trajectory sketches, 2023. URL https://arxiv.org/abs/2311.01977.

[26] Ruijie Zheng, Yongyuan Liang, Shuaiyi Huang, Jianfeng

Gao, Hal Daumé III, Andrey Kolobov, Furong Huang, and Jianwei Yang. Tracevla: Visual trace prompting enhances spatial-temporal awareness for generalist robotic policies, 2025. URL https://arxiv.org/abs/2412.10345.

[27] Noriaki Hirose, Catherine Glossop, Dhruv Shah, and Sergey Levine. Omnivla: An omni-modal vision-language-action model for robot navigation, 2025. URL https://arxiv.org/abs/2509.19480.

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

[29] James Betker, Gabriel Goh, Li Jing, Tim Brooks, Jianfeng Wang, Linjie Li, LongOuyang, Juntang Zhuang, Joyce Lee, Yufei Guo, Wesam Manassra, Prafulla Dhariwal, Casey Chu, Yunxin Jiao, and Aditya Ramesh. Improving image generation with better captions. 2023.

[30] Caelan Reed Garrett, Rohan Chitnis, Rachel Holladay, Beomjoon Kim, Tom Silver, Leslie Pack Kaelbling, and Tomás Lozano-Pérez. Integrated task and motion planning, 2020.

[31] Daniel Kahneman. *Thinking, fast and slow*. Farrar, Straus and Giroux, 2011.

[32] Jyh-Jing Hwang, Runsheng Xu, Hubert Lin, Wei-Chih Hung, Jingwei Ji, Kristy Choi, Di Huang, Tong He, Paul Covington, Benjamin Sapp, Yin Zhou, James Guo, Dragomir Anguelov, and Mingxing Tan. Emma: End-to-end multimodal model for autonomous driving, 2024. URL https://arxiv.org/abs/2410.23262.

[33] Lucy Xiaoyang Shi, Zheyuan Hu, Tony Z. Zhao, Archit Sharma, Karl Pertsch, Jianlan Luo, Sergey Levine, and Chelsea Finn. Yell at your robot: Improving on-the-fly from language corrections, 2024.

[34] Boyi Li, Philipp Wu, Pieter Abbeel, and Jitendra Malik. Interactive task planning with language models, 2025. URL https://arxiv.org/abs/2310.10645.

[35] Dhruv Shah, Blazej Osinski, Brian Ichter, and Sergey Levine. Lm-nav: Robotic navigation with large pre-trained models of language, vision, and action, 2022. URL https://arxiv.org/abs/2207.04429.

[36] Wenlong Huang, Pieter Abbeel, Deepak Pathak, and Igor Mordatch. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents, 2022. URL https://arxiv.org/abs/2201.07207.

[37] Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, Pierre Sermanet, Noah Brown, Tomas Jackson, Linda Luu, Sergey Levine, Karol Hausman, and Brian Ichter. Inner monologue: Embodied reasoning through planning with language models, 2022.

[38] Andy Zeng, Maria Attarian, Brian Ichter, Krzysztof Choromanski, Adrian Wong, Stefan Welker, Federico Tombari, Aveek Purohit, Michael Ryoo, Vikas Sindhwani, Johnny Lee, Vincent Vanhoucke, and Pete Florence. Socratic models: Composing zero-shot multimodal reasoning with language, 2022.

[39] Danny Driess, Fei Xia, Mehdi S. M. Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, Wenlong Huang, Yevgen Chebotar, Pierre Sermanet, Daniel Duckworth, Sergey Levine, Vincent Vanhoucke, Karol Hausman, Marc Toussaint, Klaus Greff, Andy Zeng, Igor Mordatch, and Pete Florence. Palm-e: An embodied multimodal language model, 2023.

[40] Jesse Zhang, Jiahui Zhang, Karl Pertsch, Ziyi Liu, Xiang Ren, Minsuk Chang, Shao-Hua Sun, and Joseph J. Lim. Bootstrap your own skills: Learning to solve new tasks with large language model guidance, 2023. URL https://arxiv.org/abs/2310.10021.

[41] Huy Ha, Pete Florence, and Shuran Song. Scaling up and distilling down: Language-guided robot skill acquisition, 2023.

[42] Suneel Belkhale, Tianli Ding, Ted Xiao, Pierre Sermanet, Quon Vuong, Jonathan Tompson, Yevgen Chebotar, Debidatta Dwibedi, and Dorsa Sadigh. Rt-h: Action hierarchies using language, 2024.

[43] Yi Yang, Jiaxuan Sun, Siqi Kou, Yihan Wang, and Zhijie Deng. Lohovla: A unified vision-language-action model for long-horizon embodied tasks, 2025. URL https://arxiv.org/abs/2506.00411.

[44] Markus Peschl, Pietro Mazzaglia, and Daniel Dijkman. From code to action: Hierarchical learning of diffusion-vlm policies, 2025. URL https://arxiv.org/abs/2509.24917.

[45] Huang Fang, Mengxi Zhang, Heng Dong, Wei Li, Zixuan Wang, Qifeng Zhang, Xueyun Tian, Yucheng Hu, and Hang Li. Robix: A unified model for robot interaction, reasoning and planning, 2025. URL https://arxiv.org/abs/2509.01106.

[46] Tao Jiang, Tianyuan Yuan, Yicheng Liu, Chenhao Lu, Jianning Cui, Xiao Liu, Shuiqi Cheng, Jiyang Gao, Huazhe Xu, and Hang Zhao. Galaxea open-world dataset and g0 dual-system vla model, 2025. URL https://arxiv.org/abs/2509.00576.

[47] Matt Deitke, Christopher Clark, Sangho Lee, Rohun Tripathi, Yue Yang, Jae Sung Park, Mohammadreza Salehi, Niklas Muennighoff, Kyle Lo, Luca Soldaini, Jiasen Lu, Taira Anderson, Erin Bransom, Kiana Ehsani, Huong Ngo, YenSung Chen, Ajay Patel, Mark Yatskar, Chris Callison-Burch, Andrew Head, Rose Hendrix, Favyen Bastani, Eli VanderBilt, Nathan Lambert, Yvonne Chou, Arnavi Chheda, Jenna Sparks, Sam Skjonsberg, Michael Schmitz, Aaron Sarnat, Byron Bischoff, Pete Walsh, Chris Newell, Piper Wolters, Tanmay Gupta, Kuo-Hao Zeng, Jon Borchardt, Dirk Groeneveld, Crystal Nam, Sophie Lebrecht, Caitlin Wittlif, Carissa Schoenick, Oscar Michel, Ranjay Krishna, Luca Weihs, Noah A. Smith, Hannaneh Hajishirzi, Ross Girshick, Ali Farhadi, and Aniruddha Kembhavi. Molmo and pixmo: Open weights

and open data for state-of-the-art vision-language models, 2024. URL https://arxiv.org/abs/2409.17146.

[48] Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, Eric Mintun, Junting Pan, Kalyan Vasudev Alwala, Nicolas Carion, Chao-Yuan Wu, Ross Girshick, Piotr Dollár, and Christoph Feichtenhofer. Sam 2: Segment anything in images and videos, 2024. URL https://arxiv.org/abs/2408.00714.

[49] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers, 2020. URL https://arxiv.org/abs/2005.12872.

[50] Gemini Team. Gemini: A family of highly capable multimodal models, 2024.

[51] William Chen, Michał Zawalski, Karl Pertsch, Oier Mees, Chelsea Finn, and Sergey Levine. Tensorrt-openvla, 2025. URL https://github.com/rail-berkeley/tensorrt-openvla.

[52] NVIDIA. Tensorrt-llm. https://github.com/NVIDIA/TensorRT-LLM?tab=readme-ov-file, 2024.

[53] Letian Fu, Huang Huang, Gaurav Datta, Lawrence Yunliang Chen, William Chung-Ho Panitch, Fangchen Liu, Hui Li, and Ken Goldberg. In-context imitation learning via next-token prediction. *arXiv preprint arXiv:2408.15980*, 2024.

[54] Yida Yin, Zekai Wang, Yuvan Sharma, Dantong Niu, Trevor Darrell, and Roei Herzig. In-context learning enables robot action prediction in llms, 2025. URL https://arxiv.org/abs/2410.12782.

[55] Embodiment Collaboration, Abby O'Neill, Abdul Rehman, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, Albert Tung, Alex Bewley, Alex Herzog, Alex Irpan, Alexander Khazatsky, Anant Rai, Anchit Gupta, Andrew Wang, Andrey Kolobov, Anikait Singh, Animesh Garg, Aniruddha Kembhavi, Annie Xie, Anthony Brohan, Antonin Raffin, Archit Sharma, Arefeh Yavary, Arhan Jain, Ashwin Balakrishna, Ayzaan Wahid, Ben Burgess-Limerick, Beomjoon Kim, Bernhard Schölkopf, Blake Wulfe, Brian Ichter, Cewu Lu, Charles Xu, Charlotte Le, Chelsea Finn, Chen Wang, Chenfeng Xu, Cheng Chi, Chenguang Huang, Christine Chan, Christopher Agia, Chuer Pan, Chuyuan Fu, Coline Devin, Danfei Xu, Daniel Morton, Danny Driess, Daphne Chen, Deepak Pathak, Dhruv Shah, Dieter Büchler, Dinesh Jayaraman, Dmitry Kalashnikov, Dorsa Sadigh, Edward Johns, Ethan Foster, Fangchen Liu, Federico Ceola, Fei Xia, Feiyu Zhao, Felipe Vieira Frujeri, Freek Stulp, Gaoyue Zhou, Gaurav S. Sukhatme, Gautam Salhotra, Ge Yan, Gilbert Feng, Giulio Schiavi, Glen Berseth, Gregory Kahn, Guanzhi Wang, Hao Su, Hao-Shu Fang, Haochen Shi, Henghui Bao, Heni Ben Amor, Henrik I Christensen, Hiroki Furuta, Homer Walke, Hongjie Fang, Huy Ha, Igor Mordatch, Ilija Radosavovic, Isabel Leal, Jacky Liang, Jad Abou-Chakra, Jaehyung Kim, Jaimyn Drake, Jan Peters, Jan Schneider, Jasmine Hsu, Jeannette Bohg, Jeffrey Bingham, Jeffrey Wu, Jensen Gao, Jiaheng Hu, Jiajun Wu, Jialin Wu, Jiankai Sun, Jianlan Luo, Jiayuan Gu, Jie Tan, Jihoon Oh, Jimmy Wu, Jingpei Lu, Jingyun Yang, Jitendra Malik, João Silvério, Joey Hejna, Jonathan Booher, Jonathan Tompson, Jonathan Yang, Jordi Salvador, Joseph J. Lim, Junhyek Han, Kaiyuan Wang, Kanishka Rao, Karl Pertsch, Karol Hausman, Keegan Go, Keerthana Gopalakrishnan, Ken Goldberg, Kendra Byrne, Kenneth Oslund, Kento Kawaharazuka, Kevin Black, Kevin Lin, Kevin Zhang, Kiana Ehsani, Kiran Lekkala, Kirsty Ellis, Krishan Rana, Krishnan Srinivasan, Kuan Fang, Kunal Pratap Singh, Kuo-Hao Zeng, Kyle Hatch, Kyle Hsu, Laurent Itti, Lawrence Yunliang Chen, Lerrel Pinto, Li Fei-Fei, Liam Tan, Linxi "Jim" Fan, Lionel Ott, Lisa Lee, Luca Weihs, Magnum Chen, Marion Lepert, Marius Memmel, Masayoshi Tomizuka, Masha Itkina, Mateo Guaman Castro, Max Spero, Maximilian Du, Michael Ahn, Michael C. Yip, Mingtong Zhang, Mingyu Ding, Minho Heo, Mohan Kumar Srirama, Mohit Sharma, Moo Jin Kim, Naoaki Kanazawa, Nicklas Hansen, Nicolas Heess, Nikhil J Joshi, Niko Suenderhauf, Ning Liu, Norman Di Palo, Nur Muhammad Mahi Shafiullah, Oier Mees, Oliver Kroemer, Osbert Bastani, Pannag R Sanketi, Patrick "Tree" Miller, Patrick Yin, Paul Wohlhart, Peng Xu, Peter David Fagan, Peter Mitrano, Pierre Sermanet, Pieter Abbeel, Priya Sundaresan, Qiuyu Chen, Quan Vuong, Rafael Rafailov, Ran Tian, Ria Doshi, Roberto Mart'in-Mart'in, Rohan Baijal, Rosario Scalise, Rose Hendrix, Roy Lin, Runjia Qian, Ruohan Zhang, Russell Mendonca, Rutav Shah, Ryan Hoque, Ryan Julian, Samuel Bustamante, Sean Kirmani, Sergey Levine, Shan Lin, Sherry Moore, Shikhar Bahl, Shivin Dass, Shubham Sonawani, Shuran Song, Sichun Xu, Siddhant Haldar, Siddharth Karamcheti, Simeon Adebola, Simon Guist, Soroush Nasiriany, Stefan Schaal, Stefan Welker, Stephen Tian, Subramanian Ramamoorthy, Sudeep Dasari, Suneel Belkhale, Sungjae Park, Suraj Nair, Suvir Mirchandani, Takayuki Osa, Tanmay Gupta, Tatsuya Harada, Tatsuya Matsushima, Ted Xiao, Thomas Kollar, Tianhe Yu, Tianli Ding, Todor Davchev, Tony Z. Zhao, Travis Armstrong, Trevor Darrell, Trinity Chung, Vidhi Jain, Vincent Vanhoucke, Wei Zhan, Wenxuan Zhou, Wolfram Burgard, Xi Chen, Xiangyu Chen, Xiaolong Wang, Xinghao Zhu, Xinyang Geng, Xiyuan Liu, Xu Liangwei, Xuanlin Li, Yansong Pang, Yao Lu, Yecheng Jason Ma, Yejin Kim, Yevgen Chebotar, Yifan Zhou, Yifeng Zhu, Yilin Wu, Ying Xu, Yixuan Wang, Yonatan Bisk, Yoonyoung Cho, Youngwoon Lee, Yuchen Cui, Yue Cao, Yueh-Hua Wu, Yujin Tang, Yuke Zhu, Yunchu Zhang, Yunfan Jiang, Yunshuang Li, Yunzhu Li, Yusuke Iwasawa, Yutaka Matsuo, Zehan Ma, Zhuo Xu, Zichen Jeff Cui, Zichen Zhang, Zipeng Fu,

and Zipeng Lin. Open x-embodiment: Robotic learning datasets and rt-x models, 2024.

[56] Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Charles Xu, Jianlan Luo, Tobias Kreiman, You Liang Tan, Lawrence Yunliang Chen, Pannag Sanketi, Quan Vuong, Ted Xiao, Dorsa Sadigh, Chelsea Finn, and Sergey Levine. Octo: An open-source generalist robot policy. In *Proceedings of Robotics: Science and Systems*, Delft, Netherlands, 2024.

[57] Siddharth Karamcheti, Suraj Nair, Ashwin Balakrishna, Percy Liang, Thomas Kollar, and Dorsa Sadigh. Prismatic vlms: Investigating the design space of visually-conditioned language models, 2024.

[58] Lucas Beyer, Andreas Steiner, André Susano Pinto, Alexander Kolesnikov, Xiao Wang, Daniel Salz, Maxim Neumann, Ibrahim Alabdulmohsin, Michael Tschannen, Emanuele Bugliarello, Thomas Unterthiner, Daniel Keysers, Skanda Koppula, Fangyu Liu, Adam Grycner, Alexey Gritsenko, Neil Houlsby, Manoj Kumar, Keran Rong, Julian Eisenschlos, Rishabh Kabra, Matthias Bauer, Matko Bošnjak, Xi Chen, Matthias Minderer, Paul Voigtlaender, Ioana Bica, Ivana Balazevic, Joan Puigcerver, Pinelopi Papalampidi, Olivier Henaff, Xi Xiong, Radu Soricut, Jeremiah Harmsen, and Xiaohua Zhai. Paligemma: A versatile 3b vlm for transfer, 2024. URL https://arxiv.org/abs/2407.07726.

[59] Prafulla Dhariwal and Alex Nichol. Diffusion models beat gans on image synthesis, 2021. URL https://arxiv.org/abs/2105.05233.

[60] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance, 2022.

[61] Yanwei Wang, Lirui Wang, Yilun Du, Balakumar Sundaralingam, Xuning Yang, Yu-Wei Chao, Claudia Perez-D'Arpino, Dieter Fox, and Julie Shah. Inference-time policy steering through human interactions, 2024.

[62] Mitsuhiko Nakamoto, Oier Mees, Aviral Kumar, and Sergey Levine. Steering your generalists: Improving robotic foundation models via value guidance. *Conference on Robot Learning (CoRL)*, 2024.

[63] Andrew Wagenmaker, Mitsuhiko Nakamoto, Yunchu Zhang, Seohong Park, Waleed Yagoub, Anusha Nagabandi, Abhishek Gupta, and Sergey Levine. Steering your diffusion policy with latent space reinforcement learning, 2025. URL https://arxiv.org/abs/2506.15799.

[64] Kevin Frans, Seohong Park, Pieter Abbeel, and Sergey Levine. Diffusion guidance is a controllable policy improvement operator, 2025. URL https://arxiv.org/abs/2505.23458.

[65] Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation, 2021. URL https://arxiv.org/abs/2101.00190.

[66] Jacky Kwok, Christopher Agia, Rohan Sinha, Matt Foutter, Shulu Li, Ion Stoica, Azalia Mirhoseini, and Marco Pavone. Robomonkey: Scaling test-time sampling and verification for vision-language-action models, 2025. URL https://arxiv.org/abs/2506.17811.

[67] Maximilian Du and Shuran Song. Dynaguide: Steering diffusion policies with active dynamic guidance. In *Proceedings of the 39th Conference on Neural Information Processing Systems (NeurIPS)*, 2025.

[68] Yilin Wu, Ran Tian, Gokul Swamy, and Andrea Bajcsy. From foresight to forethought: Vlm-in-the-loop policy steering via latent alignment, 2025. URL https://arxiv.org/abs/2502.01828.

[69] Noah D. Goodman and Michael C. Frank. Pragmatic language interpretation as probabilistic inference. *Trends in Cognitive Sciences*, 20(11):818–829, 2016. ISSN 1364-6613. doi: https://doi.org/10.1016/j.tics.2016.08.005. URL https://www.sciencedirect.com/science/article/pii/S136466131630122X.

[70] Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning, 2023. URL https://arxiv.org/abs/2306.03310.

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

[72] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models, 2023.

[73] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mahmoud Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. Dinov2: Learning robust visual features without supervision, 2024.

[74] Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and

Lucas Beyer. Sigmoid loss for language image pre-training, 2023.

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

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