EXIMO: VLM Guided Exploration of VLA Policies

Bhavya Sukhija[*,1], **Oliver Groth**[1], **Mohit Shridhar**[1], **Tim Hertweck**[1], **Michael Bloesch**[1], **Markus Wulfmeier**[1], **Abbas Abdolmaleki**[1] **and Martin Riedmiller**[1]
[*]Work done as a Student Researcher in 2025, [1]Google DeepMind

arXiv:2608.19891v1 [cs.AI] 20 Aug 2026

How to efficiently finetune robot policies to learn new tasks on the fly? **State of the art robotic manipulation policies are based on behaviour cloning of large vision-language-action (VLA) models with billions of parameters on huge teleoperation datasets. While this simple approach has enabled significant advances for robotic manipulation, finetuning of VLA policies for learning new tasks still remains an open problem. In particular, collecting teleoperation datasets requires hundreds of hours of expensive human labour and the alternative, reinforcement learning (RL), can be notoriously sample-inefficient especially for long-horizon tasks. In addition, RL with VLAs imposes several challenges due to the model's size and architectural design. In this work, we propose EXIMO, an efficient algorithm for finetuning of VLA policies. EXIMO operates in three stages: *explore*, *imitate*, and *optimize*. During the explore phase, EXIMO equips the VLA with a vision language model (VLM) that acts as a planner. The VLM thinks and breaks down challenging long-horizon problems into shorter ones for the VLA. The VLM, together with the VLA, is used to collect an orchestrated dataset on new tasks. During the imitate phase, the VLA is finetuned with the orchestrated data. Finally, during the optimize stage, we use residual off-policy RL to further finetune the policy. In our experiments, we ablate all three stages of EXIMO and show that it outperforms existing approaches significantly in terms of sample-efficiency and final performance.**

Introduction

A crucial challenge in learning for robotics is to collect high quality datasets for training robot policies. Essentially, there are two approaches that have been widely applied for this purpose: (*i*) reinforcement learning (Sutton, 2018), and (*ii*) large scale behavioural cloning (LBC) (Barreiros et al., 2025; Black et al.; Brohan et al., 2022; Gemini Robotics Team, 2025). RL enables robotic agents to collect their own data and self-improve, and it has had great success in games and robotic locomotion tasks (Miki et al., 2022; Mnih, 2013) where high-fidelity and fast simulators are available. However, RL suffers from notoriously high sample complexity, especially for long-horizon problems where exploration is particularly challenging. This limits its direct application in the real world. To this end, large scale behavioural cloning is being widely adopted for robotic manipulation. Here teleoperators are used to collect high-quality and diverse motions with the robotic system and a policy is trained to match the motions from the teleoperators. This approach is simple and significantly more sample-efficient than RL. However, it suffers from generalization challenges, since the policy is trained only on the data distribution of the teleoperation and does not completely capture the diversity of the real-world. Furthermore, it is expensive since it requires significant amounts of human hours that are repeated for each new task and robotic system.

In this work, we aim to bridge the gap between the two aforementioned strategies. Concretely, we consider the following setting: we are given a pretrained, language-conditioned vision-language-action (VLA) policy, itself trained via LBC, that reliably performs a repertoire of *atomic* skills (e.g., pick and place) within its training distribution, but struggles on *new*, long-horizon, compositional, or reasoning-heavy tasks that require chaining these skills together or interpreting goals that lie outside the training

© 2026 Google DeepMind. All rights reserved

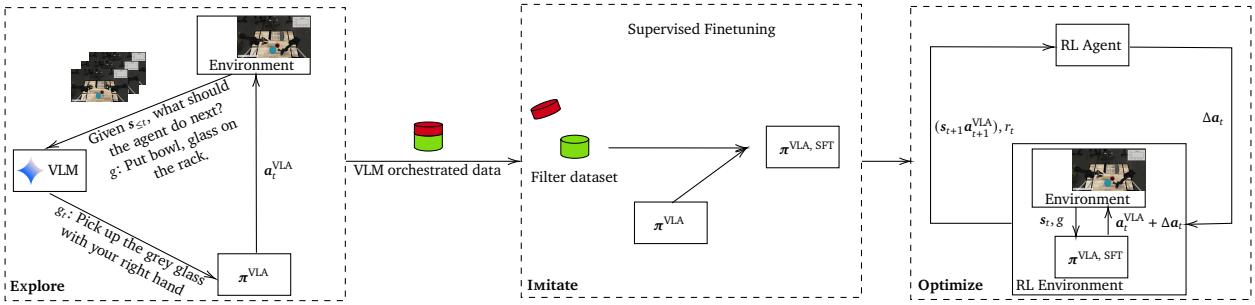

Figure 1 | EXIMO consists of three stages: Semantic exploration via VLM orchestration, where a VLM is used to command the VLA in natural language to accomplish a challenging task. The data from the VLM orchestration is used in the second phase for imitation learning, where the successful episodes are filtered and the VLA is finetuned via supervised finetuning on the filtered dataset. Finally, in the third stage, we use the finetuned VLA to perform residual reinforcement learning online.

distribution (e.g., "put the fruit the monkey likes to eat in the bowl"). Our objective is to adapt the VLA to such tasks *sample-efficiently* and *without additional teleoperation*, where each task is specified only through a natural-language goal and a success detector. Instead of relying on teleoperators, we leverage the world-knowledge embedded in state-of-the-art vision-language foundation models (VLMs) to orchestrate the VLA and collect data on these more complex, long-horizon tasks. Next, we distill the knowledge of the VLM orchestrator by supervised fine-tuning of the initial policy with the newly collected data. This enables us to teach the policy new skills without requiring expensive teleoperation hours. Finally, we further finetune the initial policy using online reinforcement learning. After the fine-tuning phase, the policy is already capable of achieving non-trivial performance, making the exploration for online RL tractable, i.e., more sample-efficient. In summary, we use a three-step approach; (*i*) data-collection with VLM orchestration, (*ii*) supervised-finetuning on the collected data, (*iii*) further finetuning of the policy via online RL. The algorithm, EXIMO, is depicted in Fig. 1.

We evaluate EXIMO on *twenty-two* different manipulation tasks in simulation that require reasoning about the object to manipulate and chaining skills together to achieve success. In our experiments, we show that VLM orchestration results in significant sample-efficiency gains, and that EXIMO outperforms both the SOTA base VLA—even when the latter is given additional RL finetuning—and the base VLA combined with VLM orchestration.

Related Work

Foundation Models as High-level Planners Several works leverage LLMs as high-level planners for robotic tasks (Ahn et al., 2022; Bhat et al., 2024; Huang et al., 2022; Mon-Williams et al., 2025; Wu et al., 2023). In particular, Huang et al. (2022) use LLMs as a receding horizon planner for long-horizon problems. The LLM generated actions are then grounded to permissible actions using a masked LLM. Generally, grounding is a crucial challenge for LLM planners. Huang et al. (2023); Mon-Williams et al. (2025); Singh et al. (2023); Wu et al. (2023) provide robot APIs in the prompt to the LLMs for grounding, whereas Ahn et al. (2022) pass a finite set of skills along with their probability of success to the LLM planner as context for grounding. Bhat et al. (2024) propose using two LLMs, one for generating action plans in natural language and another LLM for grounding the actions to robot API calls. Crucially, recent advances in VLA policies, e.g., Gemini Robotics On-Device (GROD) and Gemini Robotics (Gemini Robotics Team, 2025), enable directly interacting with the robots in natural language, alleviating the grounding challenges associated with LLM planners. Shi et al. (2025) jointly train a VLM and VLA, where the former is used to decompose high-level natural

language instructions into executable low-level ones. The low-level instructions are then passed to the VLA which generates the actions. In this work, we focus on the finetuning of VLAs using SOTA VLM models. Moreover, we leverage a natural language instructable VLA policy and study the problem of sample-efficient finetuning of the policy on challenging robotic tasks.

Foundation Models for Exploration Dalal et al. (2024) use LLMs to decompose challenging long-horizon problems into shorter-horizon subtasks and train an RL agent to execute the LLM generated plan. They show that leveraging the planner provides significant sample-efficiency gains. However, they also focus on the grounding of LLM generated plans and train a vision policy from scratch via RL. Instead, we study the finetuning of SOTA VLA policies which are instructed by the VLM directly in natural language. We show that VLM orchestrated exploration yields significant sample efficiency gains in the finetuning of the VLA without requiring any additional grounding.

Foundation models are also used to guide intrinsic exploration of agents. In particular, Tam et al. (2022) use representations present in pretrained VLMs to quantify novelty for intrinsic exploration. Sancaktar et al. (2025) use VLMs to measure the interestingness of states for exploration and learn an intrinsic reward function via preference learning from the VLM feedback. Du et al. (2023) study goal-conditioned RL and leverage causal LLMs to propose goals for exploration and masked LLMs to reward accomplishing the proposed goals. Similarly, Zhang et al. (2023) use LLMs to explore a sequence of skills from a finite skill set.

RL Finetuning of VLAs RL enables agents to self-improve by directly interacting with the environment. However, SOTA VLA policies are typically large diffusion and/or transformer models (Barreiros et al., 2025; Black et al.; Brohan et al., 2023) that are trained to predict action chunks (Brohan et al., 2023). This makes training these models via RL extremely challenging. To this end, Mark et al. (2024) learn a Q-function to optimize actions instead of training the policy, whereas Ankile et al. (2025a,b) learn a residual policy to correct actions of the VLA. We take a similar approach as Ankile et al. (2025a) and learn a residual policy via off-policy RL. However, we additionally improve sample-efficiency of our method by integrating VLM guided exploration.

Method

In the following, we describe **E**xplore **IM**itate and **O**ptimize (**EXIMO**), our three-step training pipeline for sample-efficient robot learning.

Explore: VLM-Guided Data Collection

We assume access to an initial goal-conditioned manipulation policy that is capable of performing fundamental skills such as pick and place. Moreover, we use the 3B variant of Gemini Robotics On-Device (GROD, 2025) as our initial policy. GROD is a vision-language-action (VLA) model based on the PaliGemma (Beyer et al., 2024) VLM backbone and diffusion policy head. GROD is trained on manipulation tasks with teleoperated data for the Aloha robot in both real and simulation (Aldaco et al., 2024). Given the state measurement $\boldsymbol{s} \in \mathcal{S}$ and a goal in *natural language* $g \in \mathcal{G} \subseteq \mathcal{T}^{L}$[1], e.g., "pick up the yellow banana" the VLA returns an action $\boldsymbol{a} \in \mathcal{A}$ attempting to accomplish the goal. That is $\boldsymbol{a} \sim \boldsymbol{\pi}^{VLA}(\cdot|\boldsymbol{s}, g)$. However, while the model has high success rates in tasks that lie within the training distribution, e.g., "put the banana in the bowl", its performance drops in goals and states that lie outside of training data, e.g., "put the fruit the monkey likes to eat in the bowl" or "put the

[1]Here $\mathcal{T}$ represents the set of tokens in the vocabulary of the model and $\mathcal{T}^{L}$ a sequence of tokens.

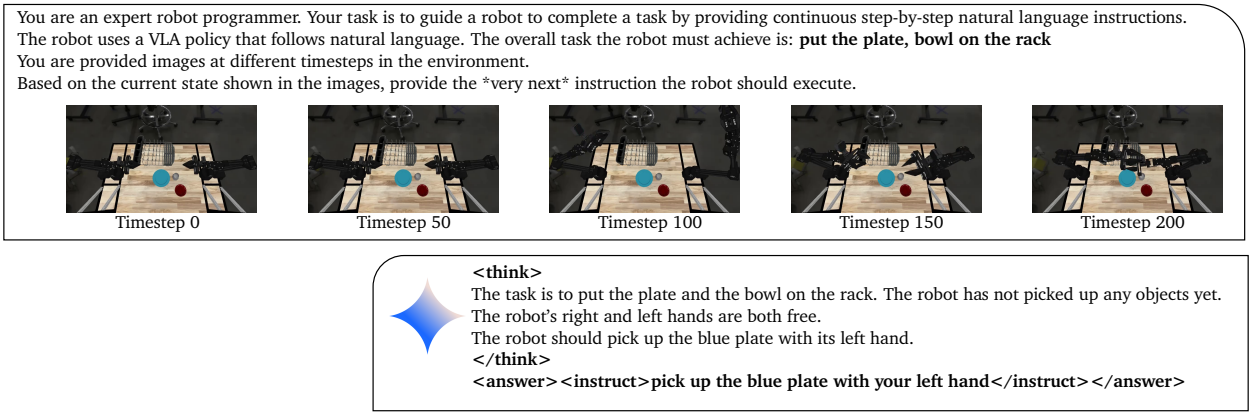

Figure 2 | Example interaction of the VLM during the explore phase of EXIMO. The VLM is given a sequence of images from the environment along with the task description in the prompt. The VLM analyzes the provided information within a <think> </think> block and provides an instruction in the <answer></answer> block for the VLA to execute.

banana and lime in the bowl". In general, collecting data for all plausible tasks via teleoperation is intractable. However, since the model is capable of performing fundamental skills, we combine it with a state-of-the-art VLM, in particular Gemini (Team et al., 2023), which decomposes the goal g into intermediary goals/skills that the VLA is capable of executing. Moreover, the VLM acts as an orchestration policy which, at time t, given the history of states $\boldsymbol{s}_{\leq t}$ and the natural goal g provides tractable goals g_t for the VLA to reach, i.e., $g_t \sim \boldsymbol{\pi}^{VLM}(\cdot|\boldsymbol{s}_{\leq t}, g)$ (see Fig. 2). The VLA is then conditioned on the goal provided by the VLM. We integrate the VLM and VLA interaction in a closed-loop manner, allowing the VLM to change and adapt the goals provided to the VLA as and when needed.

This approach provides a natural separation between fundamental physical skills that require high quality and domain specific teleoperated datasets with which VLAs are trained and semantic understanding of the underlying task for which SOTA VLMs can be leveraged off-the-shelf. We rollout the VLM-orchestrated policy to collect data for new unseen tasks and store the rollouts that successfully accomplished the task in a data buffer, which we use for supervised fine-tuning. Whether a rollout was successful is determined by the environment's ground-truth success detector: at the end of each episode we query the task-success signal and only add episodes marked successful to the buffer, thereby filtering out unsuccessful orchestration attempts.[2]

IMITATE: Supervised Finetuning on VLM Orchestrated Data

We collect a dataset consisting of states, goals, and the corresponding action, i.e., $(\boldsymbol{s}, g, \boldsymbol{a})$ and finetune the VLA to predict the action $\boldsymbol{a}$ given $(\boldsymbol{s}, g)$[3]. Crucially, here instead of keeping the VLM orchestrated-goal, we train the VLA to directly predict the actions conditioned on the actual task goal g. Through this we distill the knowledge of the VLM orchestrated policy back into the VLA. Concretely, finetuning here refers to continuing to train the pretrained GROD policy on the filtered orchestrated dataset using the same behaviour-cloning objective with which the VLA was originally trained, i.e., we update the model weights to predict the action chunk $\boldsymbol{a}_{0:K-1}$ from $(\boldsymbol{s}, g)$ via gradient descent. We report the exact VLM orchestration prompt in Fig. 9 and the full task suite in Table 1. This approach has several

[2]We assume access to a ground-truth success detector, as is standard in simulation. Replacing it with a VLM-based success detector is a promising direction discussed in the Future Work section.

[3]In practice, instead of learning to predict the immediate action $\boldsymbol{a}$, we train the VLA to predict a chunk of next actions, i.e., $\boldsymbol{a}_{0:K-1}$ where K is the action chunk size. See Zhao et al. (2023) for more detail.

benefits: first, it enables efficient real-time control on the system, since the VLM latency issues are circumvented by distilling the knowledge directly into the much smaller VLA; furthermore, it also significantly minimizes VLM calls, as the VLM is not queried any further after data collection. Finally, it simplifies the downstream RL pipeline because the RL agent now only interacts with the VLA. This also answers a natural question—why not simply keep the VLM orchestrator at evaluation time? Beyond removing the VLM's latency, distillation even improves task performance over orchestrating at evaluation time, as we show in Fig. 3 (GROD + SFT vs. GROD + VLM-Orchestration).

Optimize: Finetuning via Online RL

In the final stage, we further finetune the policy via online RL. RL finetuning of VLAs is particularly challenging due to their large size and implicit policy distribution induced from the diffusion process. Therefore, we instead use a residual policy similar to Ankile et al. (2025a). The policy outputs a residual action $\Delta \boldsymbol{a}$. This residual action is then combined with the action returned by the VLA and then executed on the system, i.e., $\boldsymbol{a} = \boldsymbol{a}^{\text{VLA}} + \Delta \boldsymbol{a}$. The residual policy is conditioned on the current state, goal, and the VLA action, that is, $\Delta \boldsymbol{a} \sim \boldsymbol{\pi}^{\text{ref}}(\cdot|\boldsymbol{s}, g, \boldsymbol{a}^{\text{VLA}})$. Accordingly, the residual MDP has the following state $\boldsymbol{x} = (\boldsymbol{s}, \boldsymbol{a}^{\text{VLA}})$ and given the action $\Delta \boldsymbol{a}$, the next state $\boldsymbol{x}' = (\boldsymbol{s}', \boldsymbol{a}'^{\text{VLA}})$ is obtained via the following transition dynamics

$$\begin{aligned} \boldsymbol{s}' &\sim T(\cdot|\boldsymbol{s}, \boldsymbol{a}^{\text{VLA}} + \Delta \boldsymbol{a}) \\ \boldsymbol{a}'^{\text{VLA}} &\sim \boldsymbol{\pi}^{VLA}(\cdot|\boldsymbol{s}', g) \end{aligned}$$

Here T represents the transition kernel of the underlying system.

We train the residual policy to optimize the underlying probability of success

$$\max_{\boldsymbol{\pi}^{\text{ref}} \in \Pi} \underbrace{\mathbb{E}_{\boldsymbol{\pi}^{\text{ref}}, \boldsymbol{s}_0 \sim \rho} \left[\sum_{t=0}^{\infty} \gamma^t \mathbb{1}_{\boldsymbol{s}_t \in \text{Success}(g)} \right]}_{J(\boldsymbol{\pi}^{\text{ref}}, \rho)} \tag{1}$$

The set $\text{Success}(g) \subset \mathcal{S}$ denotes the states for which the goal g is accomplished. For most goals, this set is much smaller than $\mathcal{S}$ and therefore $\mathbb{1}_{\boldsymbol{s}_t \in \text{Success}(g)}$ represents a sparse reward, making the problem in Eq. (1) extremely challenging for RL. However, if the VLA policy is able to accomplish non-trivial success rates on the task, this improves the residual policy learning and exploration. We use the MPO algorithm (Abdolmaleki et al., 2018) for the RL finetuning of the residual controller.

Experiment

We empirically validate our approach across several manipulation tasks for the ALOHA (Aldaco et al., 2024) robotic platform in simulation. In particular, we ablate each step of EXIMO and investigate the following questions: (*i*) Does VLM orchestration with GROD yield better exploration and efficient data collection? (*ii*) Does finetuning of GROD using the orchestrated data improve the model's performance? (*iii*) Does online RL with the finetuned GROD result in more sample-efficient online learning?

Does VLM orchestration with GROD yield better exploration?

We compare the performance of the base VLA (no orchestration) to the VLA with VLM orchestration. As performance metrics, we consider the model's success rate and time-to-success. The results are

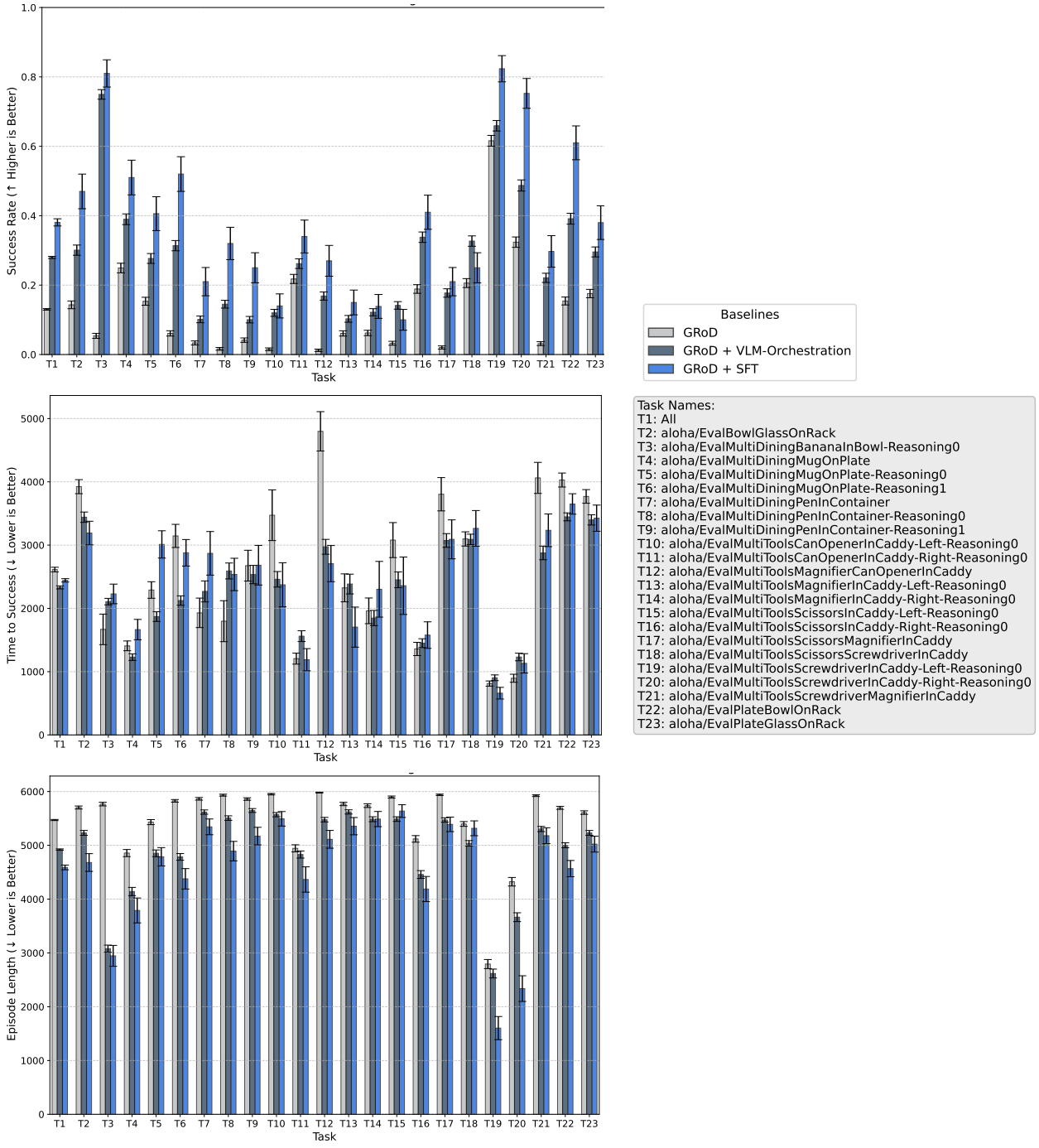

Figure 3 | Success rate (top), time to success (middle), and episode length (bottom) of VLM orchestrated GROD, GROD with no orchestration, and GROD finetuned on the data collected from VLM orchestration. The baselines and the per-task legend (shared across all three plots) are shown on the right. Across the tasks, we observe that VLM orchestration achieves higher success rate than the model without orchestration. This illustrates the benefits of semantically guided exploration from the VLM. Furthermore, performing filtered SFT on the VLM orchestrated data gives additional performance gains in the success rate across the task, showcasing the advantages of distilling the VLM orchestrated trajectories into the VLA. The VLM orchestrated VLA also has a lower episode length and is therefore more data efficient. Note that time to success is only meaningful for tasks with non-zero success rate.

reported in Fig. 3. We evaluate both methods across twenty two different tasks for *1000* episodes. Each episode is terminated if the policy successfully solves the task.

Despite GROD being a SOTA VLA, in Fig. 3 we see that VLM orchestration significantly increases the success rate of GROD while maintaining similar time to success. The gain in success rate is particularly observable in long horizon tasks such as PlateBowlOnRack, that require chaining skills of the base VLA together. Furthermore, the VLM orchestration also improves the performance of the base VLA on reasoning tasks (e.g., BananaInBowl-Reasoning), which require the agent to reason about the objects in the scene. This underlines the benefits of leveraging SOTA foundation models for semantic exploration and VLA for accomplishing individual base skills. Moreover, as shown in Fig. 3 (bottom), the VLM orchestrated policy has much smaller episode lengths and accordingly is more sample-efficient in data collection.

In summary, via VLM orchestration, we can collect higher quality episodes, i.e., those with higher success rate, more data efficiently (since each episode is effectively shorter with VLM orchestration). In the following, we investigate whether this data can be leveraged for efficient finetuning of the base VLA.

Does finetuning of GROD using the orchestrated data improve performance?

Next, we compare the finetuned GROD agent with the VLM orchestrated and non-orchestrated agent. The results are presented in Fig. 3. After finetuning of the base model on the orchestrated data, we improve the performance of the base VLA drastically. Moreover, the finetuned VLA also outperforms the orchestrated agent, showcasing its ability to learn novel tasks/behaviours from the orchestrated data. We attribute these gains to two factors: (*i*) the SFT dataset contains only the successful, filtered orchestration episodes, providing high-quality supervision, and (*ii*) distillation compiles the multi-step, VLM-guided behaviour into a single policy conditioned directly on the task goal, so that the resulting VLA no longer depends on online VLM queries at evaluation time. From this, we conclude that VLMs can control and guide the exploration of VLAs, and that distilling the newly acquired skills back into the base VLA model yields a significant performance boost. This result paves the way for a new paradigm in VLA training, where VLAs trained on basic robotic skills can be used in conjunction with VLMs to acquire new skills without any additional teleoperation hours.

Is online RL with the finetuned GROD more sample-efficient?

We compare the performance after RL of the finetuned GROD VLA, GROD + SFT, with the base GROD model. Moreover, Ankile et al. (2025a) propose RL finetuning of BC policies using off-policy residual RL. We adopt the same approach for finetuning GROD. To have a fair comparison between the SFT GROD model and the base model, we run RL on the base VLA for more episodes to compensate for the additional data collection performed during the Explore stage of EXIMO. We report the average success rate and time-to-success across all twenty tasks in Fig. 4. From the figure, we conclude that online RL improves the performance of both models. However, GROD + SFT starts at a higher success rate than the base model and also obtains higher performance at convergence compared to the base GROD model. Moreover, even though we run the base GROD model for more environment steps, it is not able to obtain the same performance as GROD + SFT. This is because GROD + SFT is trained on the higher quality data collected via VLM orchestration during the explore phase of EXIMO. The same behaviour is present for time-to-success.

In Fig. 5 we report the final performance after SFT and after SFT + RL across all tasks. We see that online RL further boosts the success rate of the agent consistently across the tasks. This demonstrates the benefits of the optimize phase of EXIMO. Furthermore, on several tasks SFT on

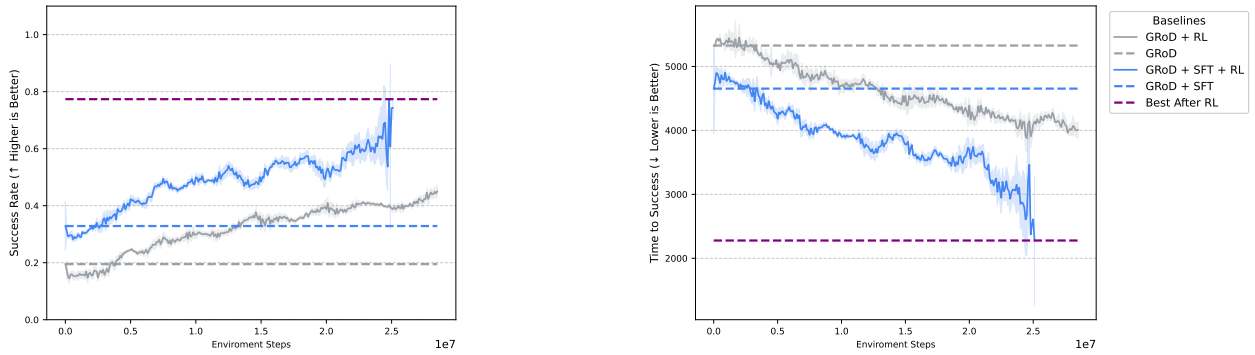

Figure 4 | Online RL performance of GROD + SFT and the base GROD model averaged across twenty tasks. We run the base GROD model for more timesteps to compensate for the additional data collected via VLM orchestration for the SFT. Due to the finetuning on VLM orchestrated data, GROD + SFT starts with a higher success rate. The model also converges to higher success rates and the base GROD model does not achieve the same performance, despite running online RL for significantly more environment steps. We observe the same behaviour for time-to-success. The performance is averaged across five seeds and we report the mean with two standard errors. Note that time to success is only meaningful for tasks with non-zero success rate.

only VLM orchestrated data outperforms the RL finetuned base VLA, despite the latter collecting significantly more data. This illustrates the benefits of VLM orchestrated data collection and distillation into the VLA model.

Conclusion

In this work, we showcase the benefits of combining the general knowledge embedded in VLMs with the sensory-motor skills of VLAs. We use the VLM to explore efficiently together with the VLA by breaking down complex tasks into intermediary steps that the VLA can execute. We show that this approach yields a higher success rate than using the base VLA alone. Furthermore, we finetune the VLA to distill the trajectories collected with VLM orchestration, thereby improving the performance of the VLA on new tasks. Next, we further improve the policy via off-policy residual RL. Across several tasks on the Aloha benchmark suite, we show that our approach significantly outperforms the baselines in terms of both sample-efficiency and performance.

Future Work

In the current setup, we assume access to ground truth success detectors. However, in principle, VLMs can also be used to detect whether a task has been completed. Future work will focus on leveraging VLMs not only as orchestrators but also as success detectors for learning. Furthermore, we also plan to leverage the VLMs for resetting the environment by orchestrating the agent to “undo the task”. For reversible tasks, this would enable a fully autonomous learning loop where the VLM is used for both reward modelling, orchestration, and resets.

More broadly, the exploration and imitation phases of EXIMO can be viewed as a post-training procedure for VLAs based on self-distillation (Hübotter et al., 2026; Lu and Lab, 2025). Our approach distills additional contextual information derived from task breakdowns produced by the VLM back into the VLA, yielding a stronger standalone policy without requiring orchestration at deployment time. In this work, we restrict ourselves to an off-policy setting, in which trajectories collected during

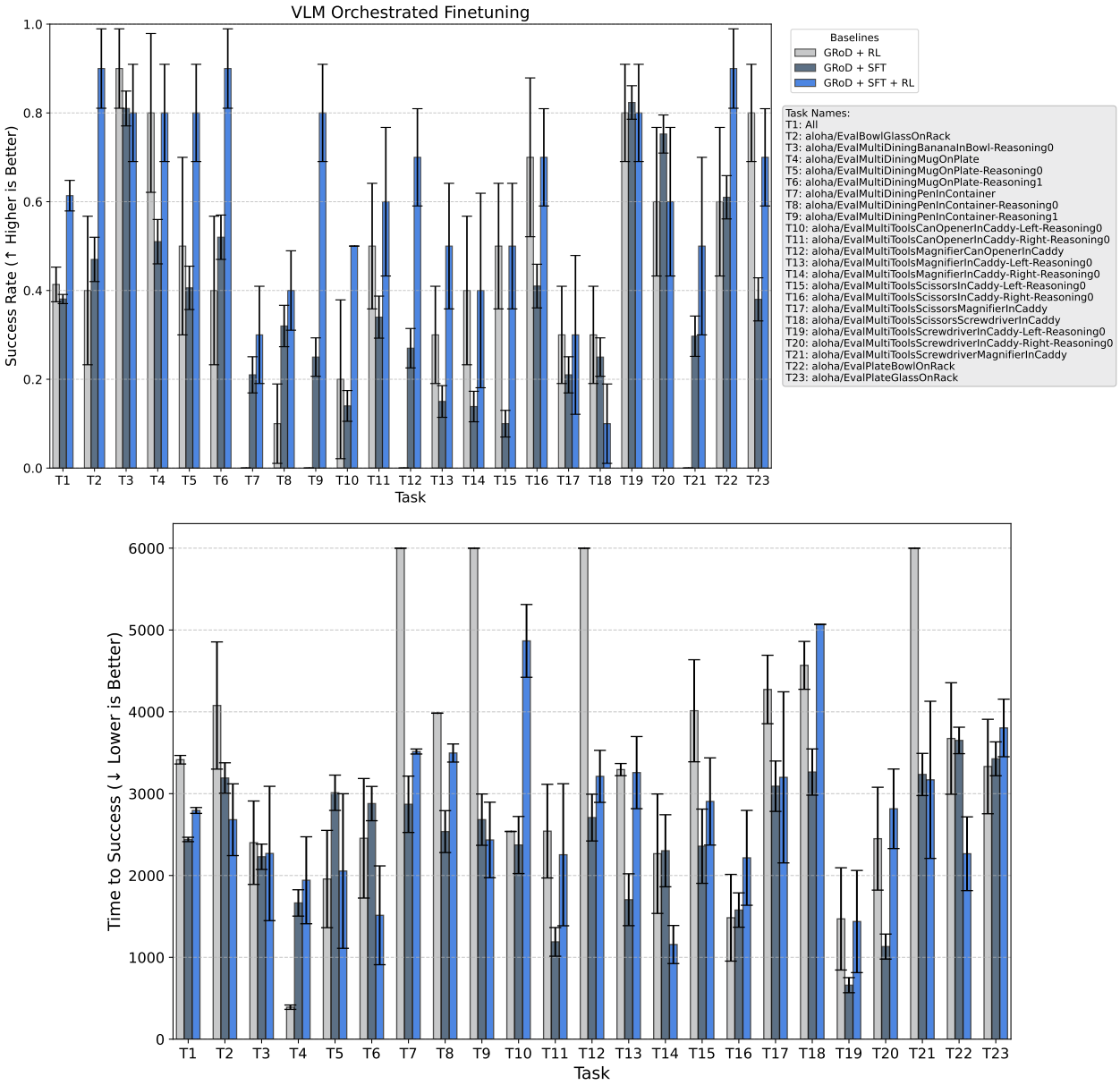

Figure 5 | Success rate and time to success post RL finetuning. We compare the performance of the VLA after SFT on VLM orchestrated data, after SFT + RL and the base GROD model after RL. Across all tasks, we observe that the model after SFT + RL outperforms the other two baselines. Note that time to success is only meaningful for tasks with non-zero success rate.

exploration are reused for supervised fine-tuning during imitation. A promising direction for future work is to extend this framework to on-policy distillation methods (Hübotter et al., 2026; Shenfeld et al., 2026).

References

A. Abdolmaleki, J. T. Springenberg, Y. Tassa, R. Munos, N. Heess, and M. Riedmiller. Maximum a posteriori policy optimisation. *arXiv preprint arXiv:1806.06920*, 2018.

M. Ahn, A. Brohan, N. Brown, Y. Chebotar, O. Cortes, B. David, C. Finn, C. Fu, K. Gopalakrishnan, K. Hausman, et al. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*, 2022.

J. Aldaco, T. Armstrong, R. Baruch, J. Bingham, S. Chan, K. Draper, D. Dwibedi, C. Finn, P. Florence, S. Goodrich, et al. Aloha 2: An enhanced low-cost hardware for bimanual teleoperation. *arXiv preprint arXiv:2405.02292*, 2024.

L. Ankile, Z. Jiang, R. Duan, G. Shi, P. Abbeel, and A. Nagabandi. Residual off-policy rl for finetuning behavior cloning policies. *arXiv preprint arXiv:2509.19301*, 2025a.

L. Ankile, A. Simeonov, I. Shenfeld, M. Torne, and P. Agrawal. From imitation to refinement-residual rl for precise assembly. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pages 01–08. IEEE, 2025b.

J. Barreiros, A. Beaulieu, A. Bhat, R. Cory, E. Cousineau, H. Dai, C.-H. Fang, K. Hashimoto, M. Z. Irshad, M. Itkina, et al. A careful examination of large behavior models for multitask dexterous manipulation. *arXiv preprint arXiv:2507.05331*, 2025.

L. Beyer, A. Steiner, A. S. Pinto, A. Kolesnikov, X. Wang, D. Salz, M. Neumann, I. Alabdulmohsin, M. Tschannen, E. Bugliarello, et al. Paligemma: A versatile 3b vlm for transfer. *arXiv preprint arXiv:2407.07726*, 2024.

V. Bhat, A. U. Kaypak, P. Krishnamurthy, R. Karri, and F. Khorrami. Grounding llms for robot task planning using closed-loop state feedback. *arXiv preprint arXiv:2402.08546*, 2024.

K. Black, N. Brown, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, L. Groom, K. Hausman, B. Ichter, et al. $\pi 0$: A vision-language-action flow model for general robot control. corr, abs/2410.24164, 2024. doi: 10.48550. *arXiv preprint ARXIV.2410.24164*.

A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, J. Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022.

A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, J. Hsu, J. Ibarz, B. Ichter, A. Irpan, T. Jackson, S. Jesmonth, N. J. Joshi, R. Julian, D. Kalashnikov, Y. Kuang, I. Leal, K.-H. Lee, S. Levine, Y. Lu, U. Malla, D. Manjunath, I. Mordatch, O. Nachum, C. Parada, J. Peralta, E. Perez, K. Pertsch, J. Quiambao, K. Rao, M. Ryoo, G. Salazar, P. Sanketi, K. Sayed, J. Singh, S. Sontakke, A. Stone, C. Tan, H. Tran, V. Vanhoucke, S. Vega, Q. Vuong, F. Xia, T. Xiao, P. Xu, S. Xu, T. Yu, and B. Zitkovich. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2023.

M. Dalal, T. Chiruvolu, D. Chaplot, and R. Salakhutdinov. Plan-seq-learn: Language model guided rl for solving long horizon robotics tasks. *arXiv preprint arXiv:2405.01534*, 2024.

Y. Du, O. Watkins, Z. Wang, C. Colas, T. Darrell, P. Abbeel, A. Gupta, and J. Andreas. Guiding pretraining in reinforcement learning with large language models. In *International Conference on Machine Learning*, pages 8657–8677. PMLR, 2023.

Gemini Robotics Team. Gemini robotics 1.5: Pushing the frontier of generalist robots with advanced embodied reasoning, thinking, and motion transfer. *arXiv preprint arXiv:2510.03342*, 2025.

GROD. Gemini Robotics On-Device (GROD). https://deepmind.google/models/gemini-robotics/gemini-robotics-on-device/, 2025. Accessed: 2025-06-30.

S. Huang, Z. Jiang, H. Dong, Y. Qiao, P. Gao, and H. Li. Instruct2act: Mapping multi-modality instructions to robotic actions with large language model. *arXiv preprint arXiv:2305.11176*, 2023.

W. Huang, P. Abbeel, D. Pathak, and I. Mordatch. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. In *International conference on machine learning*, pages 9118–9147. PMLR, 2022.

J. Hübotter, F. Lübeck, L. Behric, A. Baumann, M. Bagatella, D. Marta, I. Hakimi, I. Shenfeld, T. K. Buening, C. Guestrin, et al. Reinforcement learning via self-distillation. *arXiv preprint arXiv:2601.20802*, 2026.

K. Lu and T. M. Lab. On-policy distillation. *Thinking Machines Lab: Connectionism*, 2025. doi: 10.64434/tml.20251026. https://thinkingmachines.ai/blog/on-policy-distillation.

M. S. Mark, T. Gao, G. G. Sampaio, M. K. Srirama, A. Sharma, C. Finn, and A. Kumar. Policy agnostic rl: Offline rl and online rl fine-tuning of any class and backbone. *arXiv preprint arXiv:2412.06685*, 2024.

T. Miki, J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter. Learning robust perceptive locomotion for quadrupedal robots in the wild. *Science robotics*, 7(62):eabk2822, 2022.

V. Mnih. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

R. Mon-Williams, G. Li, R. Long, W. Du, and C. G. Lucas. Embodied large language models enable robots to complete complex tasks in unpredictable environments. *Nature Machine Intelligence*, pages 1–10, 2025.

X. B. Peng, A. Kumar, G. Zhang, and S. Levine. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *arXiv preprint arXiv:1910.00177*, 2019.

C. Sancaktar, C. Gumbsch, A. Zadaianchuk, P. Kolev, and G. Martius. Sensei: Semantic exploration guided by foundation models to learn versatile world models. *arXiv preprint arXiv:2503.01584*, 2025.

I. Shenfeld, M. Damani, J. Hübotter, and P. Agrawal. Self-distillation enables continual learning. *arXiv preprint arXiv:2601.19897*, 2026.

L. X. Shi, B. Ichter, M. Equi, L. Ke, K. Pertsch, Q. Vuong, J. Tanner, A. Walling, H. Wang, N. Fusai, et al. Hi robot: Open-ended instruction following with hierarchical vision-language-action models. *arXiv preprint arXiv:2502.19417*, 2025.

I. Singh, V. Blukis, A. Mousavian, A. Goyal, D. Xu, J. Tremblay, D. Fox, J. Thomason, and A. Garg. Progprompt: program generation for situated robot task planning using large language models. *Autonomous Robots*, 47(8):999–1012, 2023.

R. S. Sutton. Reinforcement learning: An introduction. *A Bradford Book*, 2018.

A. Tam, N. Rabinowitz, A. Lampinen, N. A. Roy, S. Chan, D. Strouse, J. Wang, A. Banino, and F. Hill. Semantic exploration from language abstractions and pretrained representations. *Advances in neural information processing systems*, 35:25377–25389, 2022.

G. Team, R. Anil, S. Borgeaud, J.-B. Alayrac, J. Yu, R. Soricut, J. Schalkwyk, A. M. Dai, A. Hauth, K. Millican, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.

J. Wu, R. Antonova, A. Kan, M. Lepert, A. Zeng, S. Song, J. Bohg, S. Rusinkiewicz, and T. Funkhouser. Tidybot: Personalized robot assistance with large language models. *Autonomous Robots*, 47(8): 1087–1102, 2023.

J. Zhang, J. Zhang, K. Pertsch, Z. Liu, X. Ren, M. Chang, S.-H. Sun, and J. J. Lim. Bootstrap your own skills: Learning to solve new tasks with large language model guidance. *arXiv preprint arXiv:2310.10021*, 2023.

T. Z. Zhao, V. Kumar, S. Levine, and C. Finn. Learning fine-grained bimanual manipulation with low-cost hardware. *arXiv preprint arXiv:2304.13705*, 2023.

Additional Experiments

Do free form instructions from VLM affect performance? A key advantage of using GROD as the base VLA is that it can be commanded in free form natural language by the VLM and does not require any additional grounding[4]. To illustrate these capabilities of GROD, we compare it with a grounded baseline, where the VLM is restricted to only giving pick and place commands, i.e., pick up the glass[5]. In Fig. 6 we compare the free form instruction VLM with the grounded one. We see that both the free form and pick&place instructions lead to comparable performance across the five tasks considered. This aligns with our intuition of the GROD VLA, which is trained to follow natural language instructions and therefore can be commanded by the VLM directly without any additional grounding.

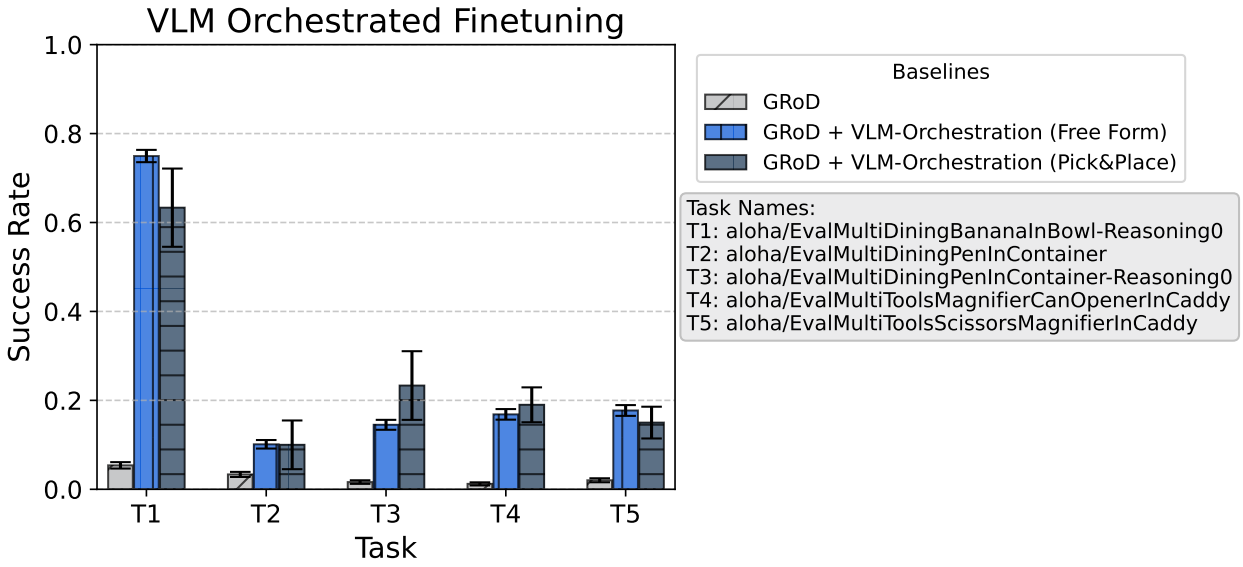

Figure 6 | Performance of VLM orchestrated GROD with free form orchestration and only Pick&Place commands. The former commands the model in natural language, whereas the latter only provides pick and place commands. We find that GROD benefits from the VLM orchestrators and is robust to the types of orchestration, performing on par for both free form and Pick&Place orchestration.

Distilling VLM Orchestration to the Residual Policy We investigate distilling the VLM orchestrator into the residual policy instead of the GROD VLA. In order to achieve this, we first collect the orchestrated dataset using the base VLA and store the tuple: $(\boldsymbol{x}, \Delta\boldsymbol{a}, \boldsymbol{x}', r)$, where $\boldsymbol{x} = (\boldsymbol{s}, \boldsymbol{a}^{VLA})$, $\Delta\boldsymbol{a} = \boldsymbol{a}^{VLA-VLM} - \boldsymbol{a}^{VLA}$, and $\boldsymbol{x}' = (\boldsymbol{s}', \boldsymbol{a}'^{VLA})$. Moreover, we define the residual action such that the policy learns to correct the GROD VLA with the residual orchestrated actions. We train the policy with advantage weighted BC (AWBC, Peng et al. (2019)) and then transition to the online RL phase. We evaluate this approach on the bowl and glass on rack environment and report the evaluated performance (without VLM) of the trained policy in Fig. 7. Here we observe that while the residual policy improves with the orchestrated data, when transitioning to the online RL phase, the policy learns significantly more slowly than the pure online RL baseline. We believe this is due to the distribution shift between the online RL and the offline RL/VLM distillation phase.

[4]In our experiments, the VLM gives natural language commands such as “move the yellow banana further into the light blue bowl with your left hand”

[5]The free form version can instead describe the object, provide additional specifications, or use different wordings, e.g., pick up the grey glass with your left hand or drop the grey glass.

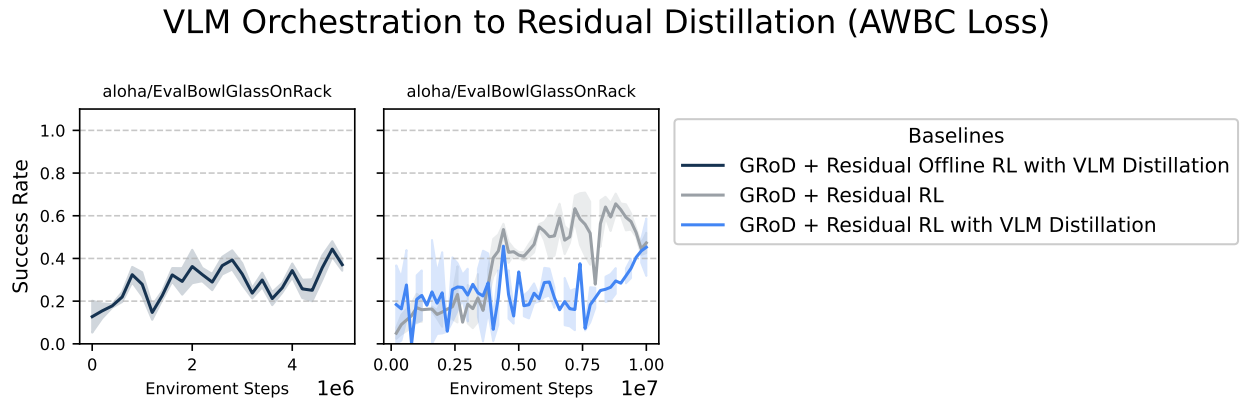

Figure 7 | Performance of VLM to residual policy distillation using AWBC for offline RL and residual RL during the online phase. Left: Performance during offline RL as a function of the offline dataset size. Right: Performance during the online RL phase. We observe that while the residual policy learns during the offline RL phase, it fails to benefit from it during the online RL one. This is most likely due to the distribution shift between the VLM orchestrated data (offline) and online rollouts. These results are obtained across three seeds and the mean performance with standard deviation is reported.

Online RL with VLM Orchestration In order to overcome the distributional shift, we train the residual policy with online RL and include the VLM orchestrator during the rollout phase. However, to overcome the distributional shift between train and evaluation phase (without the VLM), we only use the VLM for $p \times 100\%$ of the episodes. Therefore, during training the VLM is only used for a fraction of the episodes collected. This allows us to collect on-policy data from the residual policy and also bridge the gap between the train and evaluation setting. We store the tuple $(\boldsymbol{x}, \Delta\boldsymbol{a}, \boldsymbol{x}', r)$, for RL training in the data buffer. Here $\boldsymbol{x} = (\boldsymbol{s}, \boldsymbol{a}^{VLA-VLM})$, and $\boldsymbol{x}' = (\boldsymbol{s}', \boldsymbol{a}'^{VLA-VLM})$. Note that when the VLM is not used $\boldsymbol{a}^{VLA-VLM} = \boldsymbol{a}^{VLA}$. The evaluation performance of the agent (without VLM) is reported in Fig. 8 (top row). Here we observe that the method without the VLM performs the best across the tasks. We believe this is because the NO VLM baseline trains directly in the eval setting, i.e., learns to correct the base GROD model $\boldsymbol{a}^{VLA}$, where as p transitions in the data buffer of the orchestrated method are for correcting the VLM orchestrated action $\boldsymbol{a}^{VLA-VLM}$. Despite the VLM orchestrated agent performing better during data collection (Fig. 8 bottom row), the residual policy does not benefit from the orchestrated tuples in the data and therefore performs worse than the non-orchestrated one during evaluation.

Implementation Details

Task Suite

Table 1 lists the 22 manipulation tasks used in our evaluation together with their natural-language goals. The task identifiers (T2–T23) match the per-task labels in Fig. 3 and Fig. 5 (T1 in those plots denotes the average over all tasks). The tasks span several categories: (i) *dish-placement* tasks that require chaining multiple pick-and-place skills to place two items on a rack (e.g., a bowl and a glass), (ii) *reasoning* tasks, in which the target object is not named explicitly but must be inferred from a semantic description (e.g., “the item a monkey can eat”), (iii) *spatial-understanding* tasks that require placing a tool in the correct (left or right) compartment of a caddy, and (iv) *chained* tasks that require placing multiple tools into the caddy in sequence. The base VLA reliably solves the underlying atomic skills within its training distribution, but the compositional, reasoning, and spatial aspects of these

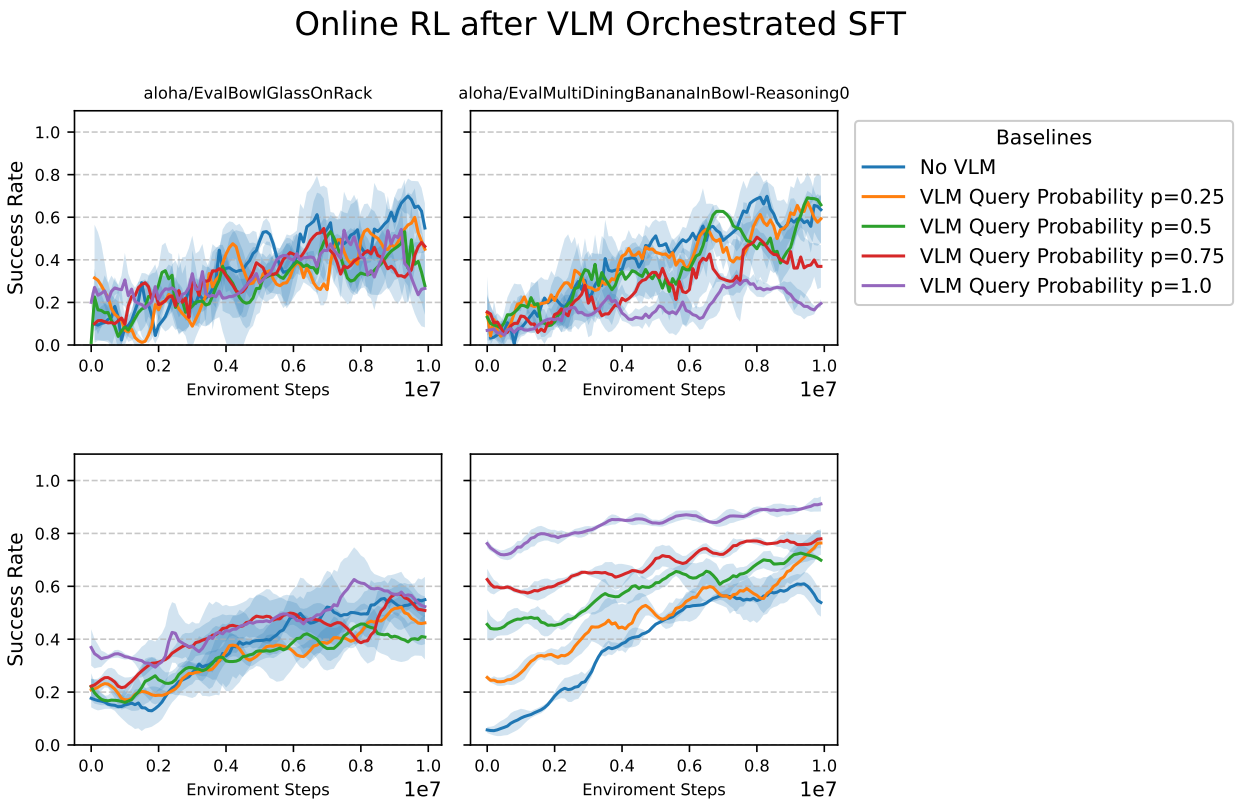

Figure 8 | Performance of VLM to residual policy distillation where the VLM is only used with probability p during the rollouts. Top row: We observe that not using the VLM yields the best performance across the tasks during evaluation, despite the VLM orchestration leading to significantly more success in the initial phases of data collection (bottom row). We believe this is because the residual policy cannot benefit from the VLM orchestrated tuples. These results are obtained across three seeds and the mean performance with standard deviation is reported.

tasks lie outside its training distribution, which is precisely where VLM orchestration helps.

VLM Orchestration Prompt

Figure 9 shows the exact prompt template used by the VLM orchestrator (cf. Fig. 2). At each step, the placeholders `[TASK]`, `{base_information_description}`, and `{instruction_guidelines}` are filled in with the overall task goal, a description of the observation the VLM receives, and the constraints on the instructions it may emit, respectively. The latter two sub-templates are shown in Fig. 10.

ID	Task	Natural-language goal
T2	BowlGlassOnRack	put the bowl and glass on the rack
T3	BananaInBowl-Reasoning0	put the item that a monkey can eat into the bowl
T4	MugOnPlate	put the mug on the plate
T5	MugOnPlate-Reasoning0	put the object you pour coffee in on the plate
T6	MugOnPlate-Reasoning1	put the object with a handle on top of the flat object
T7	PenInContainer	put the pen into the white container
T8	PenInContainer-Reasoning0	put the object you use to write into the white container
T9	PenInContainer-Reasoning1	put the thinnest object into the white container
T10	CanOpenerInCaddy-Left-Reasoning0	place the can opener in the left compartment of the caddy
T11	CanOpenerInCaddy-Right-Reasoning0	place the can opener in the right compartment of the caddy
T12	MagnifierCanOpenerInCaddy	put the magnifier and can opener in the caddy
T13	MagnifierInCaddy-Left-Reasoning0	place the magnifier in the left compartment of the caddy
T14	MagnifierInCaddy-Right-Reasoning0	place the magnifier in the right compartment of the caddy
T15	ScissorsInCaddy-Left-Reasoning0	place the scissors in the left compartment of the caddy
T16	ScissorsInCaddy-Right-Reasoning0	place the scissors in the right compartment of the caddy
T17	ScissorsMagnifierInCaddy	put the scissors and magnifier in the caddy
T18	ScissorsScrewdriverInCaddy	put the scissors and screwdriver in the caddy
T19	ScrewdriverInCaddy-Left-Reasoning0	place the screwdriver in the left compartment of the caddy
T20	ScrewdriverInCaddy-Right-Reasoning0	place the screwdriver in the right compartment of the caddy
T21	ScrewdriverMagnifierInCaddy	put the screwdriver and magnifier in the caddy
T22	PlateBowlOnRack	put the plate and bowl on the rack
T23	PlateGlassOnRack	put the plate and glass on the rack

Table 1 | Manipulation tasks used in our evaluation and their natural-language goals. The task identifiers (T2–T23) correspond to the per-task labels in Fig. 3 and Fig. 5; T1 in those plots denotes the average over all tasks. *Reasoning* variants replace the explicit object name with a semantic description that the agent must ground to the correct object, while the left/right caddy tasks additionally require spatial understanding.

```
You are an expert robot programmer. Your task is to guide a robot to complete a task by providing
↪ continuous step-by-step natural language instructions. The robot uses a VLA policy that follows
↪  natural language.

The overall task the robot must achieve is: [TASK]

{base_information_description}

Based on the current state shown in the images, provide the *very next* instruction the robot
↪ should execute.

**Instruction Guidelines:**
{instruction_guidelines}

**Key Directives:**
1. Analyze the scene and the robot's state.
2. Determine the most immediate action to progress towards the task [TASK].
3. Provide *one* clear instruction per turn.
4. Explain your reasoning inside a <thinking> </thinking> block. Inside the <thinking> block,
↪ reason step-by-step:
      a. Review all camera viewpoints to understand the current scene.
      b. Identify all objects, their properties, and their spatial relationships relevant to the
↪ task: [TASK].
      c. Describe the robot's current state (e.g., hand positions, what it's holding).
      d. What is the overall task?
      e. Based on the goal and the current state, what is the single most logical and effective
↪ action the robot should take next to make progress?
      f. Formulate this action as a clear natural language instruction, following the Instruction
↪ Guidelines.
      g. Explain why this instruction is the appropriate next step.
5. Return your instruction inside a <score><instruct> block.

Example Output Format:
<thinking>
The task is to stack the red block on the blue block. The robot's right hand is free. It should
↪ first pick up the red block.
</thinking>
<score><instruct>pick up the red block with your right hand</instruct></score>

Now, analyze the current situation for the task: [TASK].
```

Figure 9 | The main VLM orchestration prompt template used in EXIMO (cf. Fig. 2). It queries the VLM for the next natural-language instruction given the current observations. The `{base_information_description}` and `{instruction_guidelines}` sub-templates (Fig. 10) are substituted into this template at runtime.

```
# {base_information_description}
You are provided images at different timesteps in the environment. For each timestep, you are
↪ given images from different camera viewpoints. You will also be given the history of your
↪ interactions with the user.
```

```
# {instruction_guidelines}
The instructions you provide must follow this structure:
1. Allowed commands are:
   a. ‘pick the [object description]‘
   b. ‘put the [object description] in/inside the [container/location description]‘
2. Describing Objects and Locations:
   a. Be specific. Always include color and the type of object or container (e.g., "the red block
↪ ", "the blue bowl", "the wooden crate").
   b. Avoid generic terms like "thing" or "object" when a more specific noun is available.
   c. If there are multiple objects that fit the description, add details to disambiguate. Use
↪ positional language (e.g., "the leftmost green apple", "the block closest to the robot") or
↪ relational descriptions ("the cup to the right of the yellow banana").
   d. Do not refer to parts of the robot.
3. Provide the instruction in the <score><instruct> block.
   GOOD: <score><instruct>pick the red block</instruct></score>
4. One Action Per Instruction:
   a. Each instruction block must contain only a single ‘pick‘ or a single ‘put‘ command.
   b. Do not chain actions using "and", "then", or commas within a single instruction.
5. If the task is complete, only provide the <score>0</score>
```

Figure 10 | The two sub-templates substituted into the main VLM orchestration prompt (Fig. 9): `{base_information_description}` describes the observations passed to the VLM, and `{instruction_guidelines}` constrains the instructions the VLM may emit.