VLA as Tools: Exploring the Space of Agentic Robot Generalists

Dylan Goetting
University of California Berkeley
dylangoetting@berkeley.edu

Abstract: Recent advancements in Vision-Language-Action (VLA) models have shown promise in robotic manipulation tasks when provided with natural language instructions. However, these models still struggle with complex, multi-step tasks and exhibit limited generalization to new objects and scenes. In this work, I explore the space in which a central planning agent, powered by a Large Language Model (LLM), interacts with the environment by iteratively prompting a VLA model. Through a carefully designed feedback module, the LLM can observe the outcome of such robotic trajectory, and use this information for downstream tasks.

Keywords: Robotics, VLM, Agents, VLA

1 Introduction

Vision-Language-Action (VLA) models [1, 2, 3, 4] have exhibited early promise in performing diverse tasks through natural language prompting. However, as highlighted in [5], current VLA models have difficulty interpreting and executing complex tasks. To address these shortcomings, this work proposes a framework that leverages the strengths of Large Language Models (LLMs) as central planning agents in conjunction with VLA models. The core idea is to have the planning agent interact with the environment by prompting a VLA, observe the outcome of the robotic trajectory executed by the VLA, and iteratively refine the subsequent instructions. In addition to proposing such a system, this work makes the following contributions. First, I evaluate several approaches for autonomously communicating feedback and task status from the robot environment to the planning agent. Second, I present a sensitivity analysis of an off-the-shelf VLA to variations in language prompting. Third, I propose an exploration framework in which the planning agent autonomously learns how to use the VLA without any external supervision, and evaluate its performance. Lastly, I provide a qualitative example of the system working together, and and acknowledge several key shortcomings.

2 Related Work

Connecting the rich space of language commands with the world of robotic policies has been attempted in a few ways. As mentioned above, VLAs [1, 2, 3, 4], are one approach to developing generalist policies that span natural language inputs. A survey paper [5], extensively evaluates several VLAs and concludes that OpenVLA [1] achieves the highest performance, but like the other models, is very sensitive to out-of-distribution inputs.

Another line of existing research leverages LLMs as high-level planners that call low-level robotic primitives [6, 7, 8, 9, 10], which has enabled significant improvements in the range of language commands a robot can execute. However, these low-level primitives are often limited in scope, and constrain the robot to a set of pre-defined behaviors.

Vision Language Models (VLMs) have also gained traction as a method for autonomously evaluating robotic performance due to their improved visual capabilities [11, 12, 13]. This has taken the

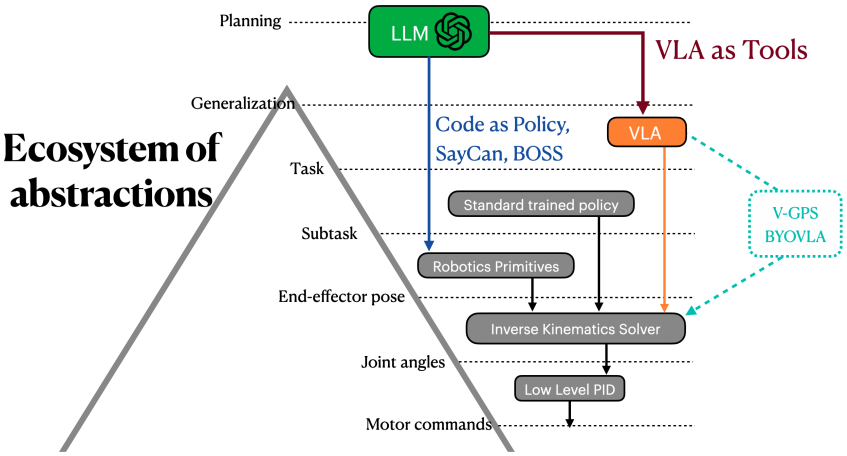

Figure 1: Robotic generalists require several layers of abstraction. VLAs are able to generalize across tasks, they fall short of the high level planning capabilities of LLMs. Works shown in blue bridge this gap by allowing LLMs to solve tasks by calling low-level robotic APIs. This work builds off of those ideas by connecting LLMs to VLAs, enabling a higher level of abstraction

form of training reward functions, estimating task completion values, and even classifying robotic trajectories.

Additionally, some recent work explores ways in which to improve VLAs in a zero-shot setting, treating them as black-box models [14, 15]. Surprisingly, an area that has not yet been explored is employing LLMs as planners to use VLAs as black-box tools, taking advantage of the shared language space between both models. As shown in Figure 1, this work explores that space and additionally incorporates VLMs as a feedback signal.

3 VLA as Tools

The core idea of the proposed system involves a central planning agent that is able to send instructions for a VLA to execute, effectively making the VLA a *tool*. Unlike the traditional agent and tool paradigm, which has seen tremendous success in tasks such as web navigation and software development, the problem setting here poses two significant challenges: (i) the tool in this case cannot be documented and described easily, and (ii) the time horizon at which actions are executed is also unknown. To address the former, I take advantage of how LLMs are capable of learning from examples [16, 17]. To address the latter, I design a feedback module to periodically provide the agent with updates on the impact of its previous actions. An system overview can be seen in Figure 2. The VLA operates at high frequency executing some instruction provided by the planning agent, and writing the third person images to a shared memory buffer. The feedback module periodically reads from this buffer, and reports whether the current instruction was completed successfully, failed, or is still in progress. The central planning agent (green), upon receiving this feedback, decides what instruction to subsequently send to the VLA. The individual results of the attempted tasks are then stored offline for future prompting of the planning agent.

In addition to the proposed system, this work investigates three research hypotheses.

1. LLMs can visually determine the completion status of a task
2. The space of instructions that VLAs can successfully execute is limited
3. LLMs can learn to translate general instructions into such space

Setup: I utilize OpenVLA as the VLA model, given it is the best available open-source model [5]. As in [1], I use the Libero environment and task suites, which contains a diverse selection of

manipulation tasks. The environment provides rendered third person images, which is used as input to OpenVLA, and also provides ground truth success information for each task. I use Gemini-Flash for both the central planning agent and the feedback module.

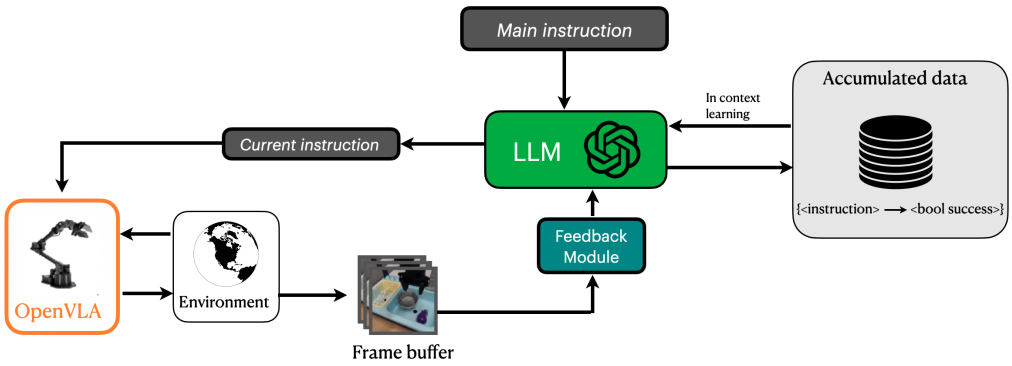

Figure 2: An overview of the proposed framework. In one process, the VLA continuously executes an instruction in its environment. Concurrently, a central planning agent chooses what instructions to send to the VLA, and a feedback module periodically sends updates to the planning agent. The trajectory results are stored offline for future prompting of the central agent

3.1 What is the best design for a feedback module?

VLMs have gained popularity in robotics in a variety of ways. [12] uses VLM feedback to train a reward function for various robotic tasks. [13] propose a method for VLMs to estimate a task completion percentage from a sequence of frames. Most similar to this work, [11] use a VLM to classify the success value of a trajectory from the last frame. Surprisingly, there has been little research on feeding videos into the VLM as opposed to just single images. To address this, I present a small-scale study on four different designs: using just the last frame as in [11], using a video of the whole trajectory, and both methods but with a prompt consisting of several example trajectories and their respective labels (ICL [16]). The VLM is then asked to output a binary success value for each robot trajectory, and this is compared to the ground-truth success value to calculate accuracy.

Method	Accuracy	Accuracy w ICL	Latency
Last frame	56%	61% (+5%)	**1.5s**
Video	**60%**	**78% (+18%)**	6s

Table 1: VLMs as a classifier: Using video with in-context learning proves to be the strongest method, but is limited by its high latency

As seen in Table 1, VLMs are indeed able to reason about robotic videos, likely due to recent advances in context length and vision capabilities. Videos also capture important information that may not be visible in the last frame. Surprisingly, adding few-shot examples improves the video approach by a much larger margin, which is contrary to Gemini's documentation that recommends only using one video per prompt. However, the video method comes with a significant efficiency drawback, as most of the additional latency comes with sending the video over the network. From these results, I choose to use a video-based feedback module with few-shot prompting.

3.2 How sensitive is OpenVLA to variations in prompting language?

To investigate the space of instructions OpenVLA is able to successfully complete, I use an LLM to generate a new dataset of tasks, where each original task mapped to several slighly rephrased tasks. For example, the original task *put the wine bottle on the rack* might get rephrased to *locate the wine bottle and transfer it to the wine rack.*

Figure 3 shows how the success rate on the rephrased instructions compares to the original success rate. Evidently, these small changes in the instruction language lead to very large decreases in performance, especially on the harder task suites of *goal* and *long*. While these rephrased instructions are out of the model's training distribution, the performance drop is still concerning, given that the instruction tokens are all passed through a language encoder.

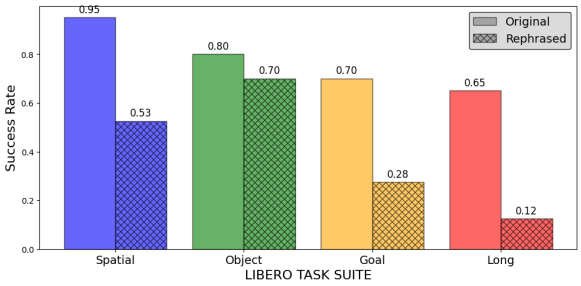

Figure 3: OpenVLA is very sensitive to the specific phrasing used in the instruction

3.3 Can a planning agent learn to use a VLA as tools autonomously?

With the findings from Sec 3.2, the natural question to ask is if the planning agent can *learn* to translate instructions into the space the VLA can successfully complete. To investigate this, I design a prompt that includes many instruction-success tuples sampled from the other task suites, which were collected in the previous experiment. The LLM then reads the current (rephrased) instruction, along with these sampled results, and uses its reasoning capabilities to output an instruction for the VLA to execute. With this method, the average success rate across the task suites **increases by 17%**, indicating that the planning agent can learn the space of successful VLA instructions to some degree.

However, this study relies on ground-truth success information from the simulator, which is not practical for real-world deployment, and furthermore the examples were not chosen by the agent. To investigate whether the previous results can be reproduced without any supervision or privileged information, I propose the following exploration procedure: First, the planning agent is given an image of the scene, and is told to learn the capabilities of the robot. It then outputs a list of tasks for the VLA to execute, importantly involving no knowledge of what tasks the VLA was trained on. The VLA then attempts each proposed task, and a success value is derived from the VLM feedback module, instead of the environment. These instruction-success tuples are then saved and sampled in the same prompting mechanism.

Method	Success Rate
Original	77.5%
Rephrased	40.6%
Rephrased ICL	57.5%
Exploration ICL	49.4%

Table 2: The success rate for different instruction methods

As seen in Table 2, this exploration method leads to a **9% increase** in success rate. While not as strong of an increase as using the ground-truth data, this result shows a promise of autonomous learning. I believe this method to be most limited by the VLM feedback signal, as I observe it to be unreliable at times, especially when the agent proposes strange tasks.

4 Limitations

I was not able to obtain quantitative results proving the efficacy of the entire system running together. Both the VLA and feedback module are limited by the real-to-sim gap, as OpenVLA was trained entirely on real-world images, and this is evident in many failure cases. A qualitative example showing the system at play can be found here. The video highlights several flaws, notably the unreliability of the feedback module, and the VLA's sensitivity to OOD tasks. Additionally, the high latency of the video-based feedback module is impractical for real-world deployment. I believe using VLAs as tools is an exciting area with lots of opportunity for future research, and this work provides a brief but meaningful exploration into such space.

References

[1] M. J. Kim, K. Pertsch, S. Karamcheti, T. Xiao, A. Balakrishna, S. Nair, R. Rafailov, E. Foster, G. Lam, P. Sanketi, et al. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

[2] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, X. Chen, K. Choromanski, T. Ding, D. Driess, A. Dubey, C. Finn, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023.

[3] O. M. Team, D. Ghosh, H. Walke, K. Pertsch, K. Black, O. Mees, S. Dasari, J. Hejna, T. Kreiman, C. Xu, J. Luo, Y. L. Tan, L. Y. Chen, P. Sanketi, Q. Vuong, T. Xiao, D. Sadigh, C. Finn, and S. Levine. Octo: An open-source generalist robot policy, 2024. URL https://arxiv.org/abs/2405.12213.

[4] K. Black, N. Brown, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, L. Groom, K. Hausman, B. Ichter, S. Jakubczak, T. Jones, L. Ke, S. Levine, A. Li-Bell, M. Mothukuri, S. Nair, K. Pertsch, L. X. Shi, J. Tanner, Q. Vuong, A. Walling, H. Wang, and U. Zhilinsky. π_0: A vision-language-action flow model for general robot control, 2024. URL https://arxiv.org/abs/2410.24164.

[5] Z. Wang, Z. Zhou, J. Song, Y. Huang, Z. Shu, and L. Ma. Towards testing and evaluating vision-language-action models for robotic manipulation: An empirical study, 2024. URL https://arxiv.org/abs/2409.12894.

[6] M. Ahn, A. Brohan, N. Brown, Y. Chebotar, O. Cortes, B. David, C. Finn, C. Fu, K. Gopalakrishnan, K. Hausman, A. Herzog, D. Ho, J. Hsu, J. Ibarz, B. Ichter, A. Irpan, E. Jang, R. J. Ruano, K. Jeffrey, S. Jesmonth, N. J. Joshi, R. Julian, D. Kalashnikov, Y. Kuang, K.-H. Lee, S. Levine, Y. Lu, L. Luu, C. Parada, P. Pastor, J. Quiambao, K. Rao, J. Rettinghouse, D. Reyes, P. Sermanet, N. Sievers, C. Tan, A. Toshev, V. Vanhoucke, F. Xia, T. Xiao, P. Xu, S. Xu, M. Yan, and A. Zeng. Do as i can, not as i say: Grounding language in robotic affordances, 2022. URL https://arxiv.org/abs/2204.01691.

[7] B. Wang, J. Zhang, S. Dong, I. Fang, and C. Feng. Vlm see, robot do: Human demo video to robot action plan via vision language model, 2024. URL https://arxiv.org/abs/2410.08792.

[8] J. Liang, W. Huang, F. Xia, P. Xu, K. Hausman, B. Ichter, P. Florence, and A. Zeng. Code as policies: Language model programs for embodied control, 2023. URL https://arxiv.org/abs/2209.07753.

[9] 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, 2023. URL https://arxiv.org/abs/2310.10021.

[10] H. Ha, P. Florence, and S. Song. Scaling up and distilling down: Language-guided robot skill acquisition, 2023. URL https://arxiv.org/abs/2307.14535.

[11] Z. Zhou, P. Atreya, A. Lee, H. Walke, O. Mees, and S. Levine. Autonomous improvement of instruction following skills via foundation models, 2024. URL https://arxiv.org/abs/2407.20635.

[12] Y. Wang, Z. Sun, J. Zhang, Z. Xian, E. Biyik, D. Held, and Z. Erickson. Rl-vlm-f: Reinforcement learning from vision language foundation model feedback, 2024. URL https://arxiv.org/abs/2402.03681.

[13] Y. J. Ma, J. Hejna, A. Wahid, C. Fu, D. Shah, J. Liang, Z. Xu, S. Kirmani, P. Xu, D. Driess, T. Xiao, J. Tompson, O. Bastani, D. Jayaraman, W. Yu, T. Zhang, D. Sadigh, and F. Xia. Vision language models are in-context value learners, 2024. URL https://arxiv.org/abs/2411.04549.

[14] A. J. Hancock, A. Z. Ren, and A. Majumdar. Run-time observation interventions make vision-language-action models more visually robust, 2024. URL https://arxiv.org/abs/2410.01971.

[15] M. Nakamoto, O. Mees, A. Kumar, and S. Levine. Steering your generalists: Improving robotic foundation models via value guidance, 2024. URL https://arxiv.org/abs/2410.13816.

[16] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. Language models are few-shot learners, 2020. URL https://arxiv.org/abs/2005.14165.

[17] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. Chi, Q. Le, and D. Zhou. Chain-of-thought prompting elicits reasoning in large language models, 2023. URL https://arxiv.org/abs/2201.11903.