Look Before You Leap: Unveiling the Power of GPT-4V in Robotic Vision-Language Planning

Yingdong Hu[1,2,3*], Fanqi Lin[1,2,3*], Tong Zhang[1,2,3], Li Yi[1,2,3], Yang Gao[1,2,3]

Abstract— **In this study, we are interested in imbuing robots with the capability of physically-grounded task planning. Recent advancements have shown that large language models (LLMs) possess extensive knowledge useful in robotic tasks, especially in reasoning and planning. However, LLMs are constrained by their lack of world grounding and dependence on external affordance models to perceive environmental information, which cannot jointly reason with LLMs. We argue that a task planner should be an inherently grounded, unified multimodal system. To this end, we introduce Robotic Vision-Language Planning (ViLa), a novel approach for long-horizon robotic planning that leverages vision-language models (VLMs) to generate a sequence of actionable steps. ViLa directly integrates perceptual data into its reasoning and planning process, enabling a profound understanding of commonsense knowledge in the visual world, including spatial layouts and object attributes. It also supports flexible multimodal goal specification and naturally incorporates visual feedback. Our extensive evaluation, conducted in both real-robot and simulated environments, demonstrates ViLa's superiority over existing LLM-based planners, highlighting its effectiveness in a wide array of open-world manipulation tasks. Project page: robot-vila.github.io**

I. Introduction

Scene-aware task planning is a pivotal facet of human intelligence [1], [2]. When presented with a simple language instruction, humans demonstrate a spectrum of complex behaviors depending on the context. Take the instruction "*get a can of coke*," for example. If a coke can is visible, a person will immediately pick it up. If not, they will search locations like the refrigerator or storage cabinets. This adaptability reflects humans' deep understanding of the scene and extensive common sense, enabling them to interpret instructions contextually. In this paper, we explore how we can create an embodied agent, such as a robot, that emulates this human-like adaptability and exhibits long-horizon task planning in varying scenes.

In recent years, large language models (LLMs) [3]–[6] have showcased their remarkable capabilities in encoding extensive semantic knowledge about the world [7]–[9]. This has sparked a growing interest in leveraging LLMs for generating step-by-step plans for complex, long-horizon tasks [10]–[12]. However, a critical limitation of LLMs is their lack of world grounding — they cannot perceive and reason about the physical state of robots and their environments, including object shapes, physical properties, and real-world constraints.

To overcome this challenge, a prevalent approach involves employing external affordance models [13] to provide real-world grounding for LLMs [10], [14]. However, these modules often fail to convey the truly necessary task-dependent information in complex environments, as they serve as one-directional channels transmitting perceptual information to LLMs. In this scenario, the LLM is like a blind person, while the affordance model serves as a sighted guide. On the one hand, the blind person relies solely on their imagination and the guide's limited narrative to comprehend the world; on the other hand, the sighted guide may not accurately comprehend the blind person's purpose. This combination often leads to unfeasible or unsafe action plans. For instance, a robot tasked with taking out a Marvel model from a shelf (see Figure 1) may overlook obstacles like the coke can, leading to collisions. Consider another example of preparing art class, scissors can be perceived as hazardous objects, or as essential tools for handicrafts. This distinction is challenging for the vision module due to the lack of specific task information. These examples highlight the limitations of LLM-based planners in capturing intricate spatial layouts and fine-grained object attributes, underscoring the necessity for active joint reasoning between vision and language.

The recent advancements in vision-language models (VLMs), exemplified by GPT-4V(ision) [15], [16], have significantly broadened the horizons of research. VLMs synergize perception and language processing into a unified system, enabling direct incorporation of perceptual information into the language model's reasoning [17]–[20]. Building upon these developments, we introduce Robotic **Vi**sion-**La**nguage Planning (**ViLa**) — a simple, effective, and scalable method for long-horizon robotic planning. ViLa distinguishes itself from previous LLM-based planning methods by eschewing independent affordance models and instead directly prompting VLMs to generate a sequence of actionable steps based on visual observations of the environment and high-level language instructions. ViLa exhibits the following key properties absent in LLM-based planning methods:

- **Profound Understanding of Commonsense Knowledge Grounded in the Visual World.** ViLa excels in complex tasks that demand an understanding of spatial layouts or object attributes. This kind of commonsense knowledge pervades nearly every task of interest in robotics, but previous LLM-based planners consistently fall short in this regard.
- **Versatile Goal Specifiaction.** ViLa supports flexible multimodal goal specification approaches. It is capable

* The first two authors contributed equally.
[1] Institute of Interdisciplinary Information Sciences, Tsinghua University.
[2] Shanghai Artificial Intelligence Laboratory.
[3] Shanghai Qi Zhi Institute.

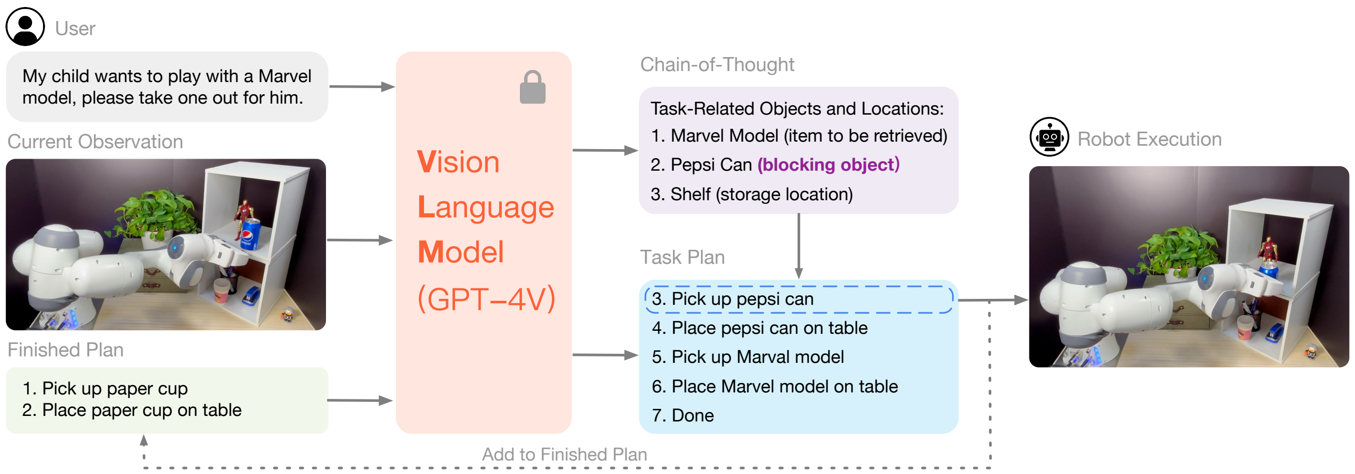

User
My child wants to play with a Marvel model, please take one out for him.
Current Observation
Finished Plan
1. Pick up paper cup
2. Place paper cup on table
Vision Language Model (GPT–4V)
Chain-of-Thought
Task-Related Objects and Locations:
1. Marvel Model (item to be retrieved)
2. Pepsi Can (blocking object)
3. Shelf (storage location)
Task Plan
3. Pick up pepsi can
4. Place pepsi can on table
5. Pick up Marval model
6. Place Marvel model on table
7. Done
Robot Execution
Add to Finished Plan

Fig. 1. **Overview of VILA.** Given a language instruction and current visual observation, we leverage a VLM to comprehend the environment scene through chain-of-though reasoning, subsequently generating a step-by-step plan. The first step of this plan is then executed by a primitive policy. Finally, the step that has been executed is added to the finished plan, enabling a closed-loop planning method in dynamic environments.

of utilizing not just language instructions but also diverse forms of goal images, and even a blend of both language and images, to define objectives effectively.

- **Visual Feedback.** VILA effectively utilizes visual feedback in an intuitive and natural way, enabling robust closed-loop planning in dynamic environments.

We conduct a evaluation of VILA across 16 real-world, everyday manipulation tasks, which involve a diverse range of open-set instructions and objects. VILA consistently outperforms LLM-based planners, such as SayCan [10] and Grounded Decoding [14], by a significant margin. To facilitate a more exhaustive comparison, we extend our evaluation to include 16 simulated tasks based on RAVENS [21], wherein VILA continues to show marked enhancements. All these outcomes provide compelling evidence that VILA possesses the potential to serve as a universal task planning method for general-purpose robotic systems.

II. METHOD

We first provide the formulation of the planning problem in Sec. II-A. Subsequently, we present how VILA utilize vision-language models as robot planners (Sec. II-B). Finally, we describe unique properties of VILA (Sec. II-C).

A. Problem Statement

Our robotic system takes a visual observation $\mathbf{x}_t$ of the environment and a high-level language instruction $\mathcal{L}$ (e.g. "*stack these containers of different colors steadily*") that describes a manipulation task. The language instruction $\mathcal{L}$ can be arbitrarily long-horizon or under-specified (i.e., requires contextual understanding). The central problem investigated in this work is to generate a sequence of text actions, represented as $\ell_1, \ell_2, \cdots, \ell_T$. Each text action ℓ_t is a short-horizon language instruction (e.g. "*pick up blue container*") that specifies a sub-task/primitive skill $\pi_{\ell_t} \in \Pi$. Note that our contributions do not focus on the acquisition of these skills Π; rather, we assume that all the necessary skills are already available. These skills can take the form of script policies or may have been acquired through various learning methods, including reinforcement learning and behavior cloning [22].

B. Vision-Language Models as Robot Planners

To generate feasible plans, high-level robot planning must be grounded in the physical world. While LLMs possess a wealth of structured world knowledge, their exclusive reliance on language input necessitates external components, such as affordance models, to complete the grounding process. However, these external affordance models (e.g., value functions of RL policies [10], [23], object detection models [24], and action detection models [25]) are manually designed as independent channels, operating separately from LLMs, rather than being integrated into an end-to-end system. Moreover, their role is solely transmitting high-dimensional visual perceptual information to LLMs, lacking the capability for joint reasoning. This separation of vision and language modalities results in the vision module's inability to provide comprehensive, task-relevant visual information, thereby hindering the LLM from planning based on accurate task-related visual insights.

Recent advances in VLMs offer a solution. VLMs demonstrate unprecedented ability in understanding and reasoning across both images and language [17]–[20]. Crucially, the extensive world knowledge encapsulated in VLMs is inherently grounded in the visual data they process. Therefore, we advocate for directly employing VLMs that synergizes vision and language capabilities to decompose a high-level instruction into a sequence of low-level skills.

We refer to our method as Robotic **Vi**sion-**La**nguage Planning (**VILA**). Concretely, given current visual observation $\mathbf{x}_t$ of environment and a high-level language goal $\mathcal{L}$, VILA operates by prompting the VLMs to yield a step-by-step plan $p_{1:N}$. We enable closed-loop execution by selecting the first step as the text action $\ell_t = p_1$. Once the text action ℓ_t is selected, the corresponding policy π_{ℓ_t} is executed by the robot and the VLM query is amended to include ℓ_t and the process is run again until a termination token (e.g., "done") is reached. The entire process is shown in Figure 1.

In our study, we utilizes GPT-4V(ision) [15], [16] as the VLM. GPT-4V, trained on vast internet-scale data, exhibits exceptional versatilities and extremely strong generalization capabilities. These attributes make it particularly adept at handling open-world scenarios presented in our paper. Furthermore, we find that ViLa, powered by GPT-4V, is capable of solving a variety of challenging planning problems, even when operating in a *zero-shot* mode (i.e., without requiring any in-context examples). This significantly reduces the prompt engineering efforts required in previous approaches [10], [11], [14].

C. *Intriguing Properties of* ViLa

In this section, we delve deeper into ViLa, shedding light on its differentiations from previous planning methods.

Comprehension of Common Sense in the Visual World. Previous studies primarily focus on leveraging the knowledge of LLMs for high-level planning [10], [11], often overlooking the crucial role of vision. Directly integrating images into the reasoning and planning process, such as in the case of ViLa, allows for a more intuitive understanding of commonsense knowledge grounded in the physical world. Specifically, this understanding manifests in two key aspects:

1) Spatial Layout Understanding: Describing complex geometric configurations, particularly object relationships and environmental constraints, can be challenging with just simple language. Consider a situation where the desired object is inside a container (like a cabinet). In that case, if an affordance model (like object detector) is utilized, since the desired object is not visible, the affordance model would predict a zero probability of successful retrieval. However, by directly incorporating vision into the reasoning process, ViLa can deduce that the sought object is likely inside the container. This realization necessitates opening the container as a preliminary step to accomplish the task.

2) Object Attribute Understanding: An object is defined by multiple attributes, like shape and material, etc. However, the expressive capacity of language is limited, making it a somewhat cumbersome medium for conveying these attributes comprehensively. Furthermore, note that an object's attributes is intricately tied to the specific tasks at hand. For example, scissors might be deemed hazardous for children, but they become essential tools during a paper-cutting art class. Therefore, active joint reasoning between image and language emerges as a crucial necessity when our tasks demand a thorough understanding of an object's attributes.

Versatile Goal Specification. In many complex, long-term tasks, using a goal image to represent the desired outcome is often more effective than relying solely on verbal instructions. For example, to direct a robot to tidy a desk, providing a photo of the desk arranged as desired can be more efficient. Such tasks, previously unattainable with LLM-based planning methods, are now remarkably straightforward with ViLa. Specifically, ViLa can not only accepts current observation $\mathbf{x}_n$ and instructions $\mathcal{L}$ as inputs but also incorporates a goal image $\mathbf{x}_g$. This feature sets it apart from many existing goal-conditioned RL/IL algorithms [26]–[28], as it does not require the goal and visual observation images to originate from the same domain. The goal image merely needs to convey the essential elements of the task, offering flexibility in its form – it could range from an internet photo to a child's drawing, or even an image showing a target location indicated by a pointing finger. This versatility greatly enhances the system's practicality.

Visual Feedback. The environments are inherently dynamic, making closed-loop feedback essential for robots. In an effort to incorporate environment feedback into planning methods that rely solely on LLMs, Huang et al. [12] investigate converting all feedback to language. However, this approach proves to be cumbersome and ineffective because most of the feedback is initially observed visually. We believe that providing visual feedback directly is a more intuitive and natural approach, as demonstrated in ViLa. Within ViLa, the VLM serves both as a scene descriptor to recognize object states and as a success detector. By reasoning over visual feedback, ViLa enables robots to make corrections or replan in response to changes in the environment or when a skill fails.

III. Experiments and Analysis

In this section, we first carry out extensive experiments in a real-world system to evaluate ViLa's capability (Sec. III-A). Subsequently, we conduct a comparison of ViLa against baseline methods within a simulated tabletop environment (Sec. III-B). The videos of experiment rollouts can be found on the project website: robot-vila.github.io.

A. *Real-World Manipulation Tasks*

Experimental Setup.

1) Hardware: We use a Franka Emika Panda robot and a parallel jaw gripper. For perception, we use a Logitech color camera mounted on a tripod, at an angle, pointing towards the tabletop.

2) Tasks and Evaluation: We design 16 long-horizon manipulation tasks to assess ViLa's performance in three domains: comprehension of commonsense knowledge in the visual world (8 tasks), flexibility in goal specification (4 tasks), and utilization of visual feedback (4 tasks). For comprehensive details of each task, please see Appendix A.2.

3) VLM and Prompting: We use GPT-4V from OpenAI API as our VLM. Unlike previous approaches [10], [14], we do not include any in-context examples in the prompt, but only use high-level language instructions and some simple constraints that the robot needs to meet (i.e., strict *zero-shot*). The full prompt is shown in Appendix A.3.

TABLE I. **Quantitative evaluation results in tasks requiring rich commonsense knowledge.**

Task	**SayCan**	**GD**	**VILA**
Pour Chips	20%	40%	80%
Bring Pepsi Can	40%	30%	90%
Bring Empty Plate	0%	0%	100%
Take Out Marvel Model	0%	10%	70%
Righteous Characters	0%	10%	80%
Pick Fresh Fruits	20%	30%	80%
Stack Plates Steadily	20%	10%	70%
Prepare Art Class	0%	30%	70%
Total	13%	20%	80%

TABLE II. **Quantitative evaluation results of VILA in tasks featuring multimodal goals.**

Task	**Goal Type**	**Succ. %**
Arrange Sushi	Real Image	80%
Arrange Gigsaw Pieces	Drawing	100%
Pick Vegetables	Pointing Finger	100%
Tidy Up Study Desk	Image + Language	60%

4) Primitive Skills: We use five categories of primitive skills that lend themselves to complex behaviors through composition and planning. These include "pick up `object`", "place `object` in/on `object`", "open `object`", "close `object`", and "pour `object` into/onto `object`". We concentrate on high-level planning rather than acquiring low-level primitive skills, which is orthogonal to our study. Therefore, we employ script policies as the primitive skills. Additional details of primitive skills are in Appendix A.4.

VILA can understand commonsense knowledge in the visual world. In Table I, we compare the planning success rates on tasks that require understanding of spatial layouts and object attributes. VILA stands out with an average success rate of 80% across 8 tasks, significantly surpassing the performances of SayCan [10] and Grounded Decoding (GD) [14], which achieve success rates of only 13% and 20%, respectively. Particularly in intricate and challenging tasks such as `Take Out Marvel Model` (it's crucial to avoid the cup and coke can) and `Righteous Characters`, SayCan and GD's success rates are close to *zero*. These tasks all necessitate the integration of images into the reasoning and planning processes and a deep understanding of commonsense knowledge in the visual world. We present a failure breakdown analysis in Appendix A.6.

VILA supports flexible multimodal goal specification. We introduce a suite of 4 tasks, each with distinct goal types. The quantitative results are shown in Table II, where VILA demonstrates strong capabilities across all tasks. Utilizing the internet-scale knowledge imbued in GPT-4V, VILA exhibits the remarkable ability to understand a variety of goal images. This includes interpreting vibrant children's drawings for puzzle completion, preparing a sushi platter by referencing a photograph of the dish, and even accurately identifying the intended arrangement of vegetables as indicated by a human finger.

TABLE III. **Open-loop VILA vs. closed-loop VILA.**

Task	**Open-Loop**	**w/ Feedback**
Stack Blocks	20%	90%
Pack Chip Bags	0%	100%
Find Stapler	30%	90%
Human-Robot Interaction	20%	80%

TABLE IV. **Average success rate in simulated environment.** See Appendix C.5 for a detailed breakdown.

	CLIPort		**LLM**		**GD**	**VILA**
Tasks	Short	Long	Llama 2	GPT-4		
Seen Tasks						
Blocks & Bowls	3.3%	68.3%	1.7%	0%	18.3%	78.3%
Letters	0%	40.0%	25.0%	25.0%	51.7%	88.3%
Unseen Tasks						
Blocks & Bowls	6.0%	6.0%	20.0%	22.0%	23.0%	81.0%
Letters	1.0%	0%	15.0%	15.0%	42.0%	82.0%

VILA can leverage visual feedback naturally. We design 4 tasks that require real-time visual feedback for successful execution. In the `Stack Blocks` task, we inject Gaussian noise into the joint position controller, which increases the likelihood of failure in the primitive policy. For the `Pack Chip Bags` task, task progress is reverted by an experimenter who takes out previously packed chip bags from the box. In the `Find Stapler` task, the stapler's location varies among three potential places: the top drawer, the bottom drawer, or the cabinet. The `Human-Robot Interaction` task requires the robot to pause until a person retrieves the cola it has picked up. We evaluate the performance of VILA against an open-loop variant that formulates a plan based solely on the initial observation. The quantitative results, presented in Table III, reveal that the open-loop variant struggles with these dynamic tasks that demand continuous replanning, while the closed-loop VILA significantly outperforms it.

B. Simulated Tabletop Rearrangement

The experimental setup is in Appendix C. We present the results in Table IV, where each method is evaluated over 20 episodes per task within each category. We observe that CLIPort-based methods have a limited capacity for generalizing to novel, unseen tasks. Given that GD requires access to the output token probabilities of LLMs, we employ Llama 2 instead of GPT-4 for GD. As depicted in Table IV, both Llama 2 and GPT-4 exhibit comparable performances across all tasks, ensuring a fair comparison between GD and VILA (utilizing GPT-4V). While GD surpasses other LLM-based planning methods by leveraging an external affordance model, it significantly lags behind VILA. This finding further highlights the benefits of synergistic reasoning between vision and language for high-level robotic planning.

References

[1] R. Wilensky, "Planning and understanding: A computational approach to human reasoning," 1983.

[2] L. A. Suchman, *Plans and situated actions: The problem of human-machine communication*. Cambridge university press, 1987.

[3] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[4] OpenAI, "Gpt-4 technical report," 2023.

[5] A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H. W. Chung, C. Sutton, S. Gehrmann, *et al.*, "Palm: Scaling language modeling with pathways," *arXiv preprint arXiv:2204.02311*, 2022.

[6] R. Bommasani, D. A. Hudson, E. Adeli, R. Altman, S. Arora, S. von Arx, M. S. Bernstein, J. Bohg, A. Bosselut, E. Brunskill, *et al.*, "On the opportunities and risks of foundation models," *arXiv preprint arXiv:2108.07258*, 2021.

[7] F. Petroni, T. Rocktäschel, P. Lewis, A. Bakhtin, Y. Wu, A. H. Miller, and S. Riedel, "Language models as knowledge bases?" *arXiv preprint arXiv:1909.01066*, 2019.

[8] Z. Jiang, F. F. Xu, J. Araki, and G. Neubig, "How can we know what language models know?" *Transactions of the Association for Computational Linguistics*, vol. 8, pp. 423–438, 2020.

[9] W. Gurnee and M. Tegmark, "Language models represent space and time," *arXiv preprint arXiv:2310.02207*, 2023.

[10] 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.

[11] 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*. PMLR, 2022, pp. 9118–9147.

[12] W. Huang, F. Xia, T. Xiao, H. Chan, J. Liang, P. Florence, A. Zeng, J. Tompson, I. Mordatch, Y. Chebotar, *et al.*, "Inner monologue: Embodied reasoning through planning with language models," *arXiv preprint arXiv:2207.05608*, 2022.

[13] J. J. Gibson, "The theory of affordances," *Hilldale, USA*, vol. 1, no. 2, pp. 67–82, 1977.

[14] W. Huang, F. Xia, D. Shah, D. Driess, A. Zeng, Y. Lu, P. Florence, I. Mordatch, S. Levine, K. Hausman, *et al.*, "Grounded decoding: Guiding text generation with grounded models for robot control," *arXiv preprint arXiv:2303.00855*, 2023.

[15] OpenAI, "Gpt-4v(ision) system card," https://cdn.openai.com/papers/GPTV_System_Card.pdf, 2023.

[16] Z. Yang, L. Li, K. Lin, J. Wang, C.-C. Lin, Z. Liu, and L. Wang, "The dawn of lmms: Preliminary explorations with gpt-4v (ision)," *arXiv preprint arXiv:2309.17421*, vol. 9, 2023.

[17] H. Liu, C. Li, Q. Wu, and Y. J. Lee, "Visual instruction tuning," *arXiv preprint arXiv:2304.08485*, 2023.

[18] W. Dai, J. Li, D. Li, A. M. H. Tiong, J. Zhao, W. Wang, B. Li, P. Fung, and S. Hoi, "Instructblip: Towards general-purpose vision-language models with instruction tuning," 2023.

[19] J. Bai, S. Bai, S. Yang, S. Wang, S. Tan, P. Wang, J. Lin, C. Zhou, and J. Zhou, "Qwen-vl: A frontier large vision-language model with versatile abilities," *arXiv preprint arXiv:2308.12966*, 2023.

[20] D. Zhu, J. Chen, X. Shen, X. Li, and M. Elhoseiny, "Minigpt-4: Enhancing vision-language understanding with advanced large language models," *arXiv preprint arXiv:2304.10592*, 2023.

[21] A. Zeng, P. Florence, J. Tompson, S. Welker, J. Chien, M. Attarian, T. Armstrong, I. Krasin, D. Duong, V. Sindhwani, *et al.*, "Transporter networks: Rearranging the visual world for robotic manipulation," in *Conference on Robot Learning*. PMLR, 2021, pp. 726–747.

[22] D. A. Pomerleau, "Alvinn: An autonomous land vehicle in a neural network," *Advances in neural information processing systems*, vol. 1, 1988.

[23] D. Kalashnikov, J. Varley, Y. Chebotar, B. Swanson, R. Jonschkowski, C. Finn, S. Levine, and K. Hausman, "Mt-opt: Continuous multi-task robotic reinforcement learning at scale," *arXiv preprint arXiv:2104.08212*, 2021.

[24] M. Minderer, A. Gritsenko, A. Stone, M. Neumann, D. Weissenborn, A. Dosovitskiy, A. Mahendran, A. Arnab, M. Dehghani, Z. Shen, *et al.*, "Simple open-vocabulary object detection with vision transformers. arxiv 2022," *arXiv preprint arXiv:2205.06230*.

[25] M. Shridhar, L. Manuelli, and D. Fox, "Cliport: What and where pathways for robotic manipulation," in *Proceedings of the 5th Conference on Robot Learning (CoRL)*, 2021.

[26] A. V. Nair, V. Pong, M. Dalal, S. Bahl, S. Lin, and S. Levine, "Visual reinforcement learning with imagined goals," *Advances in neural information processing systems*, vol. 31, 2018.

[27] B. Eysenbach, T. Zhang, S. Levine, and R. R. Salakhutdinov, "Contrastive learning as goal-conditioned reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 35, pp. 35 603–35 620, 2022.

[28] Y. Ding, C. Florensa, P. Abbeel, and M. Phielipp, "Goal-conditioned imitation learning," *Advances in neural information processing systems*, vol. 32, 2019.

[29] Y. Zhu, A. Joshi, P. Stone, and Y. Zhu, "Viola: Imitation learning for vision-based manipulation with object proposal priors," *arXiv preprint arXiv:2210.11339*, 2022.

[30] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, *et al.*, "Llama: Open and efficient foundation language models," *arXiv preprint arXiv:2302.13971*, 2023.

[31] N. H. Matthias Minderer, Alexey Gritsenko, "Scaling open-vocabulary object detection," *NeurIPS*, 2023.

[32] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou, *et al.*, "Chain-of-thought prompting elicits reasoning in large language models," *Advances in Neural Information Processing Systems*, vol. 35, pp. 24 824–24 837, 2022.

[33] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, *et al.*, "Training language models to follow instructions with human feedback," *Advances in Neural Information Processing Systems*, vol. 35, pp. 27 730–27 744, 2022.

Appendix

A. *Real-World Environment*

1) Hardware Setup: We use a Franka Emika Panda robot (a 7-DoF arm) and a 1DoF parallel jaw gripper. The robot is operated using the joint controller from Deoxys [29]. For perception, we use a Logitech Brio color camera mounted on a tripod, at an angle, pointing towards the tabletop. This camera offers high-resolution images at 1920×1080, ensuring maximum detail retention.

2) Tasks and Evaluation: We design 16 long-horizon tasks, categorized into three domains: *(i)* understanding commonsense knowledge in the visual world (8 tasks, detailed in Table A.1); *(ii)* flexibility in goal specification (4 tasks, detailed in Table A.2); and *(iii)* utilization of visual feedback (4 tasks, detailed in Table A.3). Figure A.2 illustrates a selection of 12 tasks, drawn from the first two domains. For each task, we perform 10 evaluations under different variations of the environment, accounting for changes in scene configuration, lighting conditions, etc.

3) Prompts: We do not include any in-context examples in the prompt, but only use high-level language instructions and some simple constraints that the robot needs to meet (i.e., strict zero-shot). The full prompt is shown in Figure A.1.

You are highly skilled in robotic task planning, breaking down intricate and long-term tasks into distinct primitive actions.
If the object is in sight, you need to directly manipulate it. If the object is not in sight, you need to use primitive skills to find the object first. If the target object is blocked by other objects, you need to remove all the blocking objects before picking up the target object. At the same time, you need to ignore distracters that are not related to the task. And remember your last step plan needs to be "done".
Consider the following skills a robotic arm can perform. In the descriptions below, think of [sth] as an object:
1. pick up [sth]
2. place [sth] in/on [sth]
3. pour [sth] into/onto [sth]
4. open [sth]
5. close [sth]
You are only allowed to use the provided skills. It's essential to stick to the format of these basic skills. When creating a plan, replace these placeholders with specific items or positions without using square brackets or parentheses. You can first itemize the task-related objects to help you plan.

[Initial Environment Image]

[Task Instruction]

[Reply from GPT-4V]

[Environment Image after Executing Some Steps]

This image displays a scenario after you have executed some steps from the plan generated earlier. When interacting with people, sometimes the robotic arm needs to wait for the person's action. If you do not find the target object in the current image, you need to continue searching elsewhere.

[Reply from GPT-4V]

Fig. A.1. **Prompt for real-world environment.**

4) Primitive Skills: We use five categories of primitive skills that lend themselves to complex behaviors through composition and planning. These include "pick up `object`", "place `object` in/on `object`", "open `object`", "close `object`", and "pour `object` into/onto `object`". We concentrate on high-level, temporally extended planning rather than acquiring low-level primitive skills, which is orthogonal to our study. Therefore, we employ script policies as the primitive skills. For simple tasks like "pick up `object`", we teleoperate the robots by operating a 3D SpaceMouse. For more intricate, contact-rich tasks such as "open drawer", kinematic teaching is employed. These skills are tailored to the tasks considered in our study, developing a generalizable and robust set of primitive skills is an important area for future exploration and research.

5) Baselines: We compare with SayCan [10] and Grounded Decoding (GD) [14], which both ground LLMs with external affordance models. Implementing these baselines necessitates accessing output token probabilities from LLMs. However, since OpenAI API currently does not return these probabilities, we employ the open-source Llama 2 70B [30] as an alternative. For the affordance models, we utilize the open-vocabulary detector OWL-ViT [24], [31], following Huang et al [14].

6) Failure Breakdown: In Figure A.3, we present a failure breakdown analysis. "Response structure error" here refers to errors of LLMs and VLMs in generating plan steps that fall outside our predefined set of primitive skills. In the case of baselines, "perception error" denotes failures within the open-vocab detector [31]. While VLMs lack a separate perception module, their output, as observed in the chain-of-thought process [32], occasionally fails to recognize some objects. The dominant error in baseline models is "understanding error", which involves errors in understanding the complex spatial

`Pour Chips`	**Instruction:** "My child is hungry, please pour him a plate of chips." **Description:** In five of the evaluation episodes, the chips are stored inside a cabinet, requiring the robot arm to first open the cabinet in order to locate the chips. For the remaining five episodes, the chips are directly visible, the robot arm should immediately pick up the chip bag.
`Bring Pepsi Can`	**Instruction:** "I'm very thirsty, can you help me get a can of cola and put it on the table?" **Description:** In five of the evaluation episodes, the Pepsi can is placed inside a refrigerator, requiring the robot arm to first open the refrigerator to locate the Pepsi can. In the other five episodes, the Pepsi can is directly visible, the robot arm should immediately grasp and pick it up.
`Bring Empty Plate`	**Instruction:** "Please pass me the blue empty plate." **Description:** In each evaluation episode, one or two objects are placed on a blue plate, surrounded by several distractor objects on the table. The robot arm is required to first remove the objects on the plate and then hand the plate to a human, while disregarding the distractor objects.
`Take Out Marvel Model`	**Instruction:** "My child wants to play with a Marvel model, please take one out for him." **Description:** The Marvel model is placed on the upper shelf, blocked by one or two objects, with additional distractor objects placed on the lower shelf, on top of the shelf, or around the shelf. The robot arm is required to first remove the objects blocking the Marvel model, then pick up the model and place it on the table, while ignoring the distractor objects.
`Righteous Characters`	**Instruction:** "I want to pick some righteous character models for my child, but I am not familiar with these characters. Which color toys should I put in the box?" **Description:** There are three Marvel character models: Iron Man (righteous), Captain America (righteous), and Thanos (unrighteous). Due to the constraints in the instruction stating, "I am not familiar with these characters" and "which color toys should I ...", the robot plan must not explicitly mention the names of the characters (such as Iron Man), but is limited to referencing models by their color (like red model).
`Pick Fresh Fruits`	**Instruction:** "I want to buy some fruits. Help me pick the fresh fruits from this pile of fruits and put them into the orange box." **Description:** There are some rotten fruits and some incomplete fruits (such as a half-peeled orange). The robot arm needs to disregard these distracting fruits and accurately select the fresh fruits.
`Stack Plates Steadily`	**Instruction:** "Steadily stack these containers of different colors." **Description:** There are several containers of varying sizes and colors. The robot arm must accurately discern the relative sizes of these containers and stack them steadily in order of size.
`Prepare Art Class`	**Instruction:** "We are having an art class, please prepare an area for the children. Please put any inappropriate items on the table into the box." **Description:** Certain objects are unsuitable for an art class setting (such as screwdrivers and fruit knives), while others (like glue and colored paper) are appropriate. Classifying scissors is challenging as they can be viewed as either hazardous or a craft tool for cutting paper. In this specific context, with paper cuttings present, scissors should be retained for this task. This task requires the task planner to ground objects within the specific scene to determine their attributes.

TABLE A.1. **A list of 8 tasks requiring understanding commonsense knowledge in the visual world.** The first four tasks are centered on comprehending spatial layouts, while the subsequent four are dedicated to understanding object attributes. For every task, we provide the instruction as used in our experiments and a detailed description of the task.

layouts and object attributes in the physical world, such as occlusions and context-specific attributes. VILA significantly reduces the "understanding error" by seamlessly integrating vision and language reasoning, thereby resulting in the lowest overall error. Furthermore, we suggest that careful prompt engineering (i.e., providing examples in the prompt) [3], [33] could steer VLM outputs towards admissible primitive skills, thereby reducing "response structure error".

`Arrange Sushi`	**Instruction:** "In the second picture, arrange the sushi on a specific side of the plate similar to the one in the first picture." **Goal Type**: Real Image **Description:** The planner needs to identify the types of sushi and their arrangement in the goal image, and then, based on the observed image from the experiment, place the sushi onto a specific location on the sushi plate.
`Arrange Jigsaw Pieces`	**Instruction:** "The first picture is my child's drawing. In the second picture, arrange the jigsaw pieces on the corners of the whiteboard similar to the landscape image shown in the first picture." **Goal Type**: Drawing **Description:** The planner needs to identify the positions of elements in the goal image, and then, based on the observed image from the experiment, place the jigsaw pieces in a specific corner of the whiteboard.
`Pick Vegetables`	**Instruction:** "I need to put the two vegetables in picture 2 onto the plate pointed by the finger in picture 1." **Goal Type**: Pointing Finger **Description:** In the goal image, there are multiple plates of different colors. The planner is required to identify the plate being pointed at by a finger, and based on the image observed in the experiment, place the vegetables from the scene onto the indicated plate.
`Tidy Up Study Desk`	**Instruction:** "Study the arrangement in the first picture. Replicate it in the second picture, yet switching the cup and pen holder's positions this time." **Goal Type**: Image + Language **Description:** The planner must precisely identify the arrangement of objects in the goal image, while also considering the instruction to switch the positions of the cup and the pen holder.

TABLE A.2. **A list of 4 tasks featuring multimodal goals.** For every task, we provide the instruction as used in our experiments, the goal type, and a detailed description of the task.

`Stack Blocks`	**Instruction:** "Stack all the blocks." **Description:** In this task, we inject noise during the execution of the primitive skill "Place a block on another block". In 4 out of 10 evaluation episodes, the primitive skill fails when stacking blocks for the first time. In another 4 episodes, the failure occurs during the second stacking attempt. For the remaining 2 episodes, the primitive skill does not fail.
`Pack Chip Bags`	**Instruction:** "Put the chip bag on the table in the gift box." **Description:** This task involves human intervention, where a person removes the chip bag placed in the gift box by a robot arm. In five of the evaluation episodes, the chip bag that is placed in the gift box for the first time is removed and placed on the table by human. In the other five episodes, the chip bag that is placed in the gift box for the second time is removed and placed on the table.
`Find Stapler`	**Instruction:** "Put the stapler on the table." **Description:** In this task, the target object (stapler) may be placed in the top drawer, bottom drawer, or cabinet. The planner is required to locate the target object based on visual feedback. In three evaluation episodes, the target object is placed in the top drawer; in four episodes, it is placed in the bottom drawer; and in the remaining three episodes, it is placed inside the cabinet.
`Human-Robot Interaction`	**Instruction:** "Pass me a can of cola." **Description:** In this task, the robot arm can only execute "Place can of cola in human hand" after detecting a human hand. Before that, the robot repeatedly waits and checks every five seconds for the hand's appearance. In two of the evaluation episodes, the human hand appears directly in the observation; in the remaining eight episodes, the human hand appears in the observation several seconds after the robot arm picks up the can of cola.

TABLE A.3. **A list of 4 tasks requiring visual feedback.** For every task, we provide the instruction as used in our experiments and a detailed description of the task.

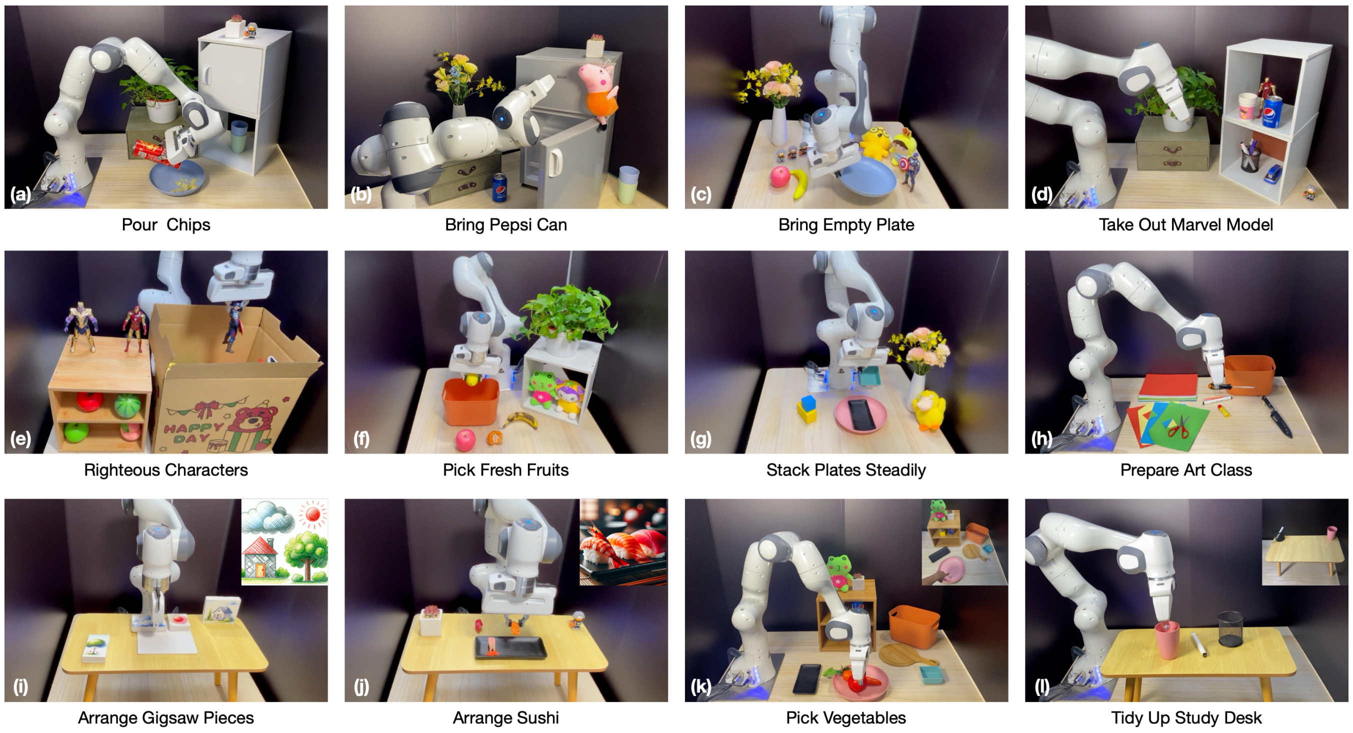


Fig. A.2. VILA can leverage the wealth of commonsense knowledge grounded in the visual world. This results in remarkable performance in tasks that demand an understanding of spatial layouts (top row), object attributes (middle row), and tasks with multimodal goals (bottom row).

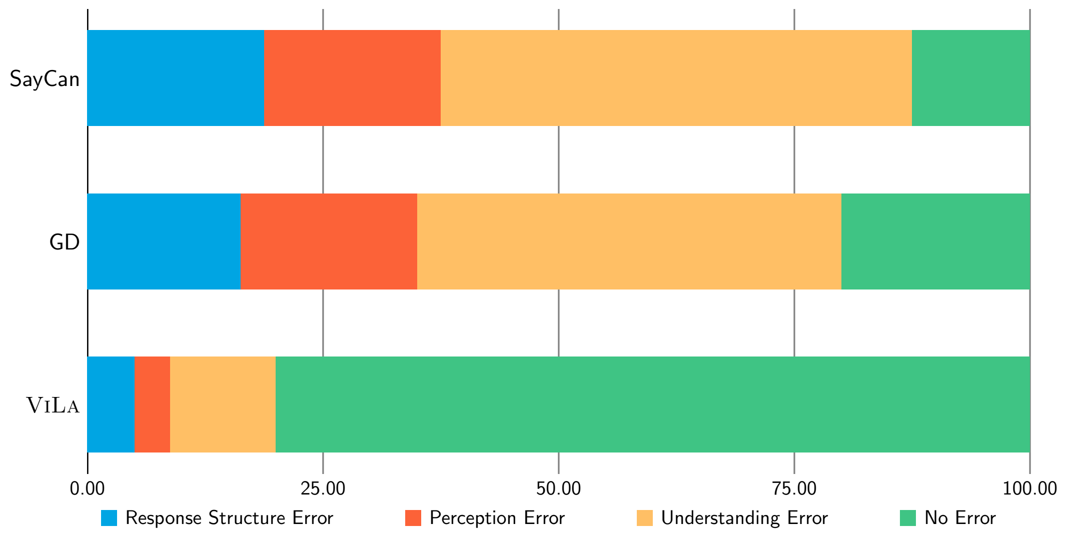


Fig. A.3. **Error breakdown of VILA and baselines.** By leveraging commonsense knowledge grounded in the visual world, VILA significantly reduces understanding error.

B. Robot Rollouts

Figure B.1 shows two environment rollouts comparing ViLa with SayCan. Figure B.2 shows two environment rollouts on image goal-conditioned tasks. Figure B.3 shows one environment rollout on a task that requires real-time visual feedback.

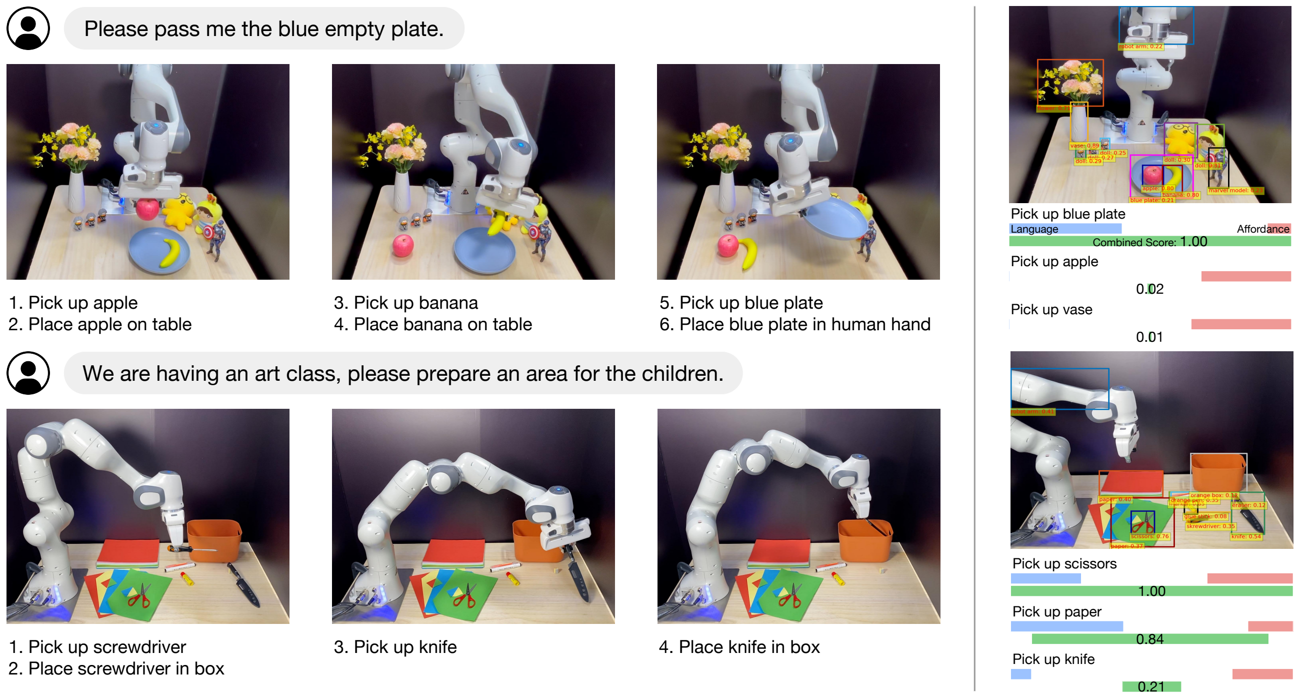
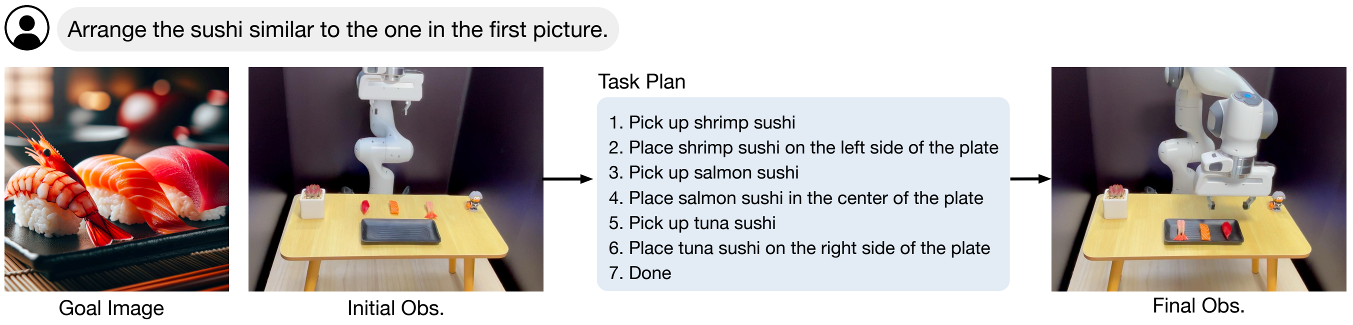
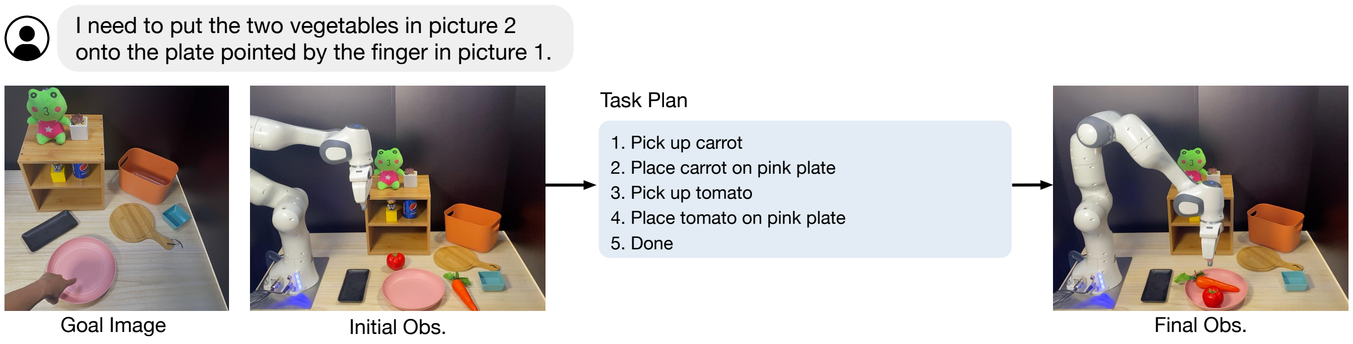


Fig. B.1. **Illustration of the execution of ViLa (left) and the decision-making process of SayCan (right)**. In the `Bring Empty Plate` task, the robot must first relocate the apple and banana from the blue plate. However, SayCan's initial step is to directly pick up the blue plate. In the `Prepare Art Class` task, while the scissor is supposed to remain on the table, SayCan erroneously picks up the scissor and places it in a box.



Task Plan

1. Pick up shrimp sushi
2. Place shrimp sushi on the left side of the plate
3. Pick up salmon sushi
4. Place salmon sushi in the center of the plate
5. Pick up tuna sushi
6. Place tuna sushi on the right side of the plate
7. Done


I need to put the two vegetables in picture 2 onto the plate pointed by the finger in picture 1.
Goal Image
Initial Obs.
Final Obs.

Task Plan

1. Pick up carrot
2. Place carrot on pink plate
3. Pick up tomato
4. Place tomato on pink plate
5. Done

Fig. B.2. **Illustration of the execution of ViLA on image goal-conditioned tasks.** In the `Arrange Sushi` task, ViLA generates a plan to arrange sushi based on a reference image. In the `Pick Vegetables` task, the scenario involves a table set with a pink plate, a black sushi plate, a pizza plate, and a green snack plate. Here, ViLA deduces from pointing finger in the goal image that the vegetables should be placed on the pink plate.

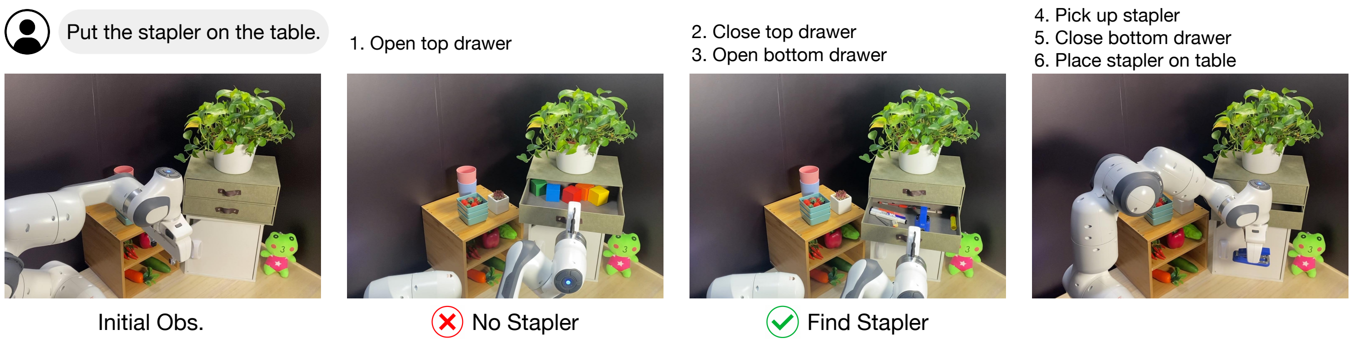


Fig. B.3. **Illustration of the execution of ViLA on the `Find Stapler` task.** By incorporating visual feedback and replanning at every step, ViLA is able to continue exploring the bottom drawer when it does not find the stapler in the top drawer, thereby successfully locating the stapler.

C. Simulated Environment

1) Tasks: Drawing inspiration from the Grounded Decoding [14] setting, we develop 16 tasks based on the RAVENS environment [21] (listed in Table C.1). Each task requires a UR5 robot to rearrange the objects on the table in some desired configuration, specified by high-level language instructions. A camera is employed to capture a top-down view for task planning purposes. The tasks are categorized into two categories: (i) `Blocks & Bowls` (8 tasks), which focus on rearranging or combining blocks and bowls. (ii) `Letters` (8 tasks), which involve rearranging alphabetical letters. Upon each reset, task-relevant objects, along with some distractor objects, are randomly distributed across the workspace. We employ a binary reward function using ground-truth state of the objects to facilitate automatic evaluations. Additionally, for certain tasks, the attributes mentioned in the instructions are also randomized, details of which are provided below:

- **corner/side**: top left corner, top side, top right corner, left side, right side, bottom right corner, bottom side, bottom left corner
- **word**: cat, dog, red, blue, pink, gold, yoga, fork, soap, milk, dance, bread, knife, chair, peach, white, brown, plate, brush, table

2) Prompts: The prompt for `Blocks & Bowls` is shown in Figure C.1. We incorporate three in-context examples (seen tasks) into the prompt. This approach addresses the substantial domain gap between simulated images of blocks and bowls and their real-world counterparts. Due to this gap, GPT-4V is unable to recognize and comprehend these objects in a zero-shot setting. Conversely, for `Letters`, we omit in-context examples from the prompt (see Figure C.2), as GPT-4V demonstrates proficient recognition and understanding of all letters.

3) Primitive Skills: In prior work, such as Grounded Decoding, CLIPort policies are utilized as low-level primitive skills. However, our findings suggest that this approach does not accurately represent the capabilities of the high-level planner. We observe that, in many tests, CLIPort policies correctly interact with objects even when the planner generate an *incorrect* step. To address this, we shift to employing script policies as our primitive skills. These policies can directly access the ground-truth states of objects within the simulator, ensuring a noise-free outcome and providing a more accurate measure of the planner's success rate.

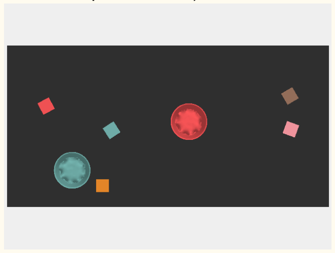
4) Baselines: Our comparison encompasses three baseline categories: *(i)* CLIPort [25], a language-conditioned imitation learning agent that directly take in the high-level language instructions without a planner. We consider two variants: "Short", trained on single-step pick-and-place instructions, and "Long", trained on high-level instructions. During evaluation, both CLIPort (Short) and CLIPort (Long) receive only the high-level instructions. These baselines aim to evaluate whether solitary language-conditioned policies can perform well on long-horizon tasks and generalize to new task instructions. Our implementation adheres closely to the description in the Grounded Decoding paper [14]; for more details, please refer to this paper. *(ii)* An LLM-based planner that does not relay on any grounding/affordance model. We evaluate Llama 2 70B and GPT-4. Our evaluation includes Llama 2 and GPT-4. The inclusion of Llama 2 stems from its use in our reimplementation of Grounded Decoding. Grounded Decoding requires access to the output token probabilities from LLMs. However, with the OpenAI API not providing these probabilities, we are constrained to using the open-source Llama 2. *(iii)* Grounded Decoding (GD), which integrates an LLM with an affordance model for enhanced planning. Here, Llama 2 is used as the LLM. For the `Blocks & Bowls` scenario, affordances are deduced from CLIPort's predicted logits, as outlined in the GD paper. For `Letters`, we resort to ground-truth affordance values from simulation due to the limited generalization capability of CLIPort's predicted logits on unseen letters. We employ the beam search variant of GD.

5) Full Results on Simulated Environments: In Table C.1, we show the full list of tasks in simulated environment, alongside their corresponding experimental results. The tasks are categorized by background color: those with a blue background are 'seen' tasks, while those with an orange background are 'unseen' tasks. 'Seen' tasks are used for training for supervised baselines (CLIPort), or included in prompts for high-level planners. However, in the case of the VILA' prompt within the `Letters` category, we do not include any 'seen' tasks.

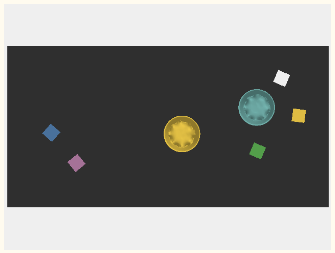
	CLIPort		LLM		GD	ViLa
Tasks	Short	Long	Llama 2	GPT-4		
Blocks & Bowls						
Stack all the blocks	10%	90%	0%	0%	30%	90%
Put all the blocks on the *corner/side*	0%	65%	0%	0%	10%	90%
Put all the blocks in the bowls with matching colors	0%	50%	5%	0%	15%	55%
Put the blocks in the bowls with mismatched colors	10%	25%	0%	0%	0%	80%
Put all the blocks on different corners	0%	0%	0%	0%	20%	90%
Stack only the blocks of cool colors	5%	0%	0%	20%	5%	60%
Stack only the blocks of warm colors	15%	5%	20%	10%	15%	80%
Stack only the primary color blocks on the left side	0%	0%	80%	80%	75%	95%
Total	5.0%	29.4%	13.1%	13.8%	21.3%	80.0%
Letters						
Put the letters on the tables in alphabetical order	0%	30%	0%	0%	25%	95%
Spell as much of *word* as you can	0%	55%	55%	60%	80%	75%
Sort the vowels from the remaining letters to the bottom side	0%	35%	20%	15%	50%	95%
Put the letters on the tables in reverse alphabetical order	0%	0%	0%	0%	10%	95%
Correctly spell out a sport using the present letters	0%	0%	0%	0%	35%	85%
Sort the geometrically vertically symmetrical letters to the bottom side	0%	0%	0%	0%	55%	40%
Sort the consonants from the remaining letters to the bottom side	0%	0%	0%	0%	10%	90%
Sort the letters less than “D” according to ASCII to the bottom side	5%	0%	75%	75%	100%	100%
Total	0.6%	15.0%	18.8%	18.8%	45.6%	84.4%

TABLE C.1. **Full experimental results in simulation on seen tasks and unseen tasks.** Each entry represents success rate averaged across 20 episodes.

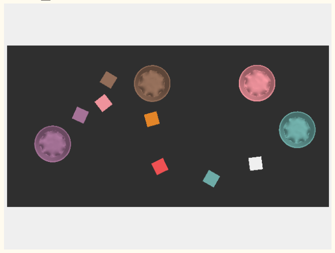
You excel at counting and identifying colors and objects in images, as well as strategizing for robotic tabletop rearrangement tasks.
I will provide three examples with corresponding tasks and robot action plans. Please first itemize all objects in each image and then detail the plan. Ensure you use the command "pick up the [color] block and place it on the [place]", and you can only pick up one block at a time, not multiple stacked blocks. Stick to the color palette: ['blue', 'red', 'green', 'yellow', 'brown', 'cyan', 'orange', 'purple', 'pink', 'white']. (Please note that your plan can only contain the instructions of each step, can not have any superfluous explanations with notes and parentheses)



This is the first example.
Task: Stack all the blocks
Blocks: red block, cyan block, orange block, pink block, brown block
Bowls: cyan bowl, red bowl
START_PLAN
Step 1: pick up the brown block and place it on the pink block
Step 2: pick up the cyan block and place it on the brown block
Step 3: pick up the orange block and place it on the cyan block
Step 4: pick up the red block and place it on the orange block
Step 5: done
END_PLAN



This is the second example.
Task: Put all the blocks on the bottom left corner
Blocks: blue block, purple block, green block, yellow block, white block
Bowls: yellow bowl, cyan bowl
START_PLAN
Step 1: pick up the white block and place it on the bottom left corner
Step 2: pick up the yellow block and place it on the bottom left corner
Step 3: pick up the green block and place it on the bottom left corner
Step 4: pick up the blue block and place it on the bottom left corner
Step 5: pick up the purple block and place it on the bottom left corner
Step 6: done
END_PLAN



This is the third example.
Task: Put all the blocks on the bowls with matching colors
Blocks: purple block, pink block, brown block, orange block, red block, cyan block, white block
Bowls: purple bowl, brown bowl, pink bowl, cyan bowl
START_PLAN
Step 1: pick up the cyan block and place it on the cyan bowl
Step 2: pick up the purple block and place it on the purple bowl
Step 3: pick up the brown block and place it on the brown bowl
Step 4: pick up the pink block and place it on the pink bowl
Step 5: done
END_PLAN

This is a new task.
[Initial Environment Image]
[Task Instruction]

Fig. C.1. **Prompt for `Blocks & Bowls` in simulated environment.**

You excel at counting and identifying letters in images, as well as planning for robotic table tasks by simplifying complex tasks into primitive skills.
The primitive skills that robot arm can execute is:
1. pick up the letter [A capital English letter] and place it on the bottom left corner / bottom side
2. pick up the letter [A capital English letter] and place it on the right of [previous letter]

It's esential to stick to the template of primitive skills. Constrains are as follows:
1. For each task, you need to first select a letter and invoke the first skill, where for the sort letters task, the first letter should be placed on the bottom side, and for other tasks (e.g. put letters in order or word spelling tasks), the letter should be placed in the bottom left corner
2. Each subsequent step in the plan requires selecting other letters and placing them to the right of the previous letters
3. In the plan, there should be no parentheses, square brackets, annotations, or explanations (e.g. skip this step)
4. You need to use the symbols "START PLAN" and "FINISH PLAN" to indicate the beginning and end of your plan
Explaining the relevant concepts in the task instruction may help your planning. Please first itemize all letters in each image and then detail the plan With These Letters. Do not include any letters that are not in the image in your plan.

[Initial Environment Image]
[Task Instruction]

Fig. C.2. **Prompt for `Letters` in simulated environment.**