Talk Through It: End-user-in-the-loop Robot Manipulation Learning Framework

Carl Winge, Adam Imdieke, Bahaa Aldeeb, Dongyeop Kang, Karthik Desingh

***Abstract*— Training generalist robot agents is an immensely difficult feat due to the adaptation to specific environments. We propose selectively training robots based on end user preferences. Given a *factory model* that lets an end user instruct a robot to perform lower-level actions (e.g., 'Move left'), we show that end-users can collect demonstrations using language to train their *home model* for higher-level tasks specific to their needs (e.g., 'Open the top drawer and put the block inside'). We demonstrate this hierarchical robot learning framework on robot manipulation tasks using RLBench environments. Our method results in a 16% improvement on skill success rates compared to a baseline method. In further experiments, we explore the use of the large vision-language model (VLM), Bard, to automatically break down tasks into sequences of lower-level instructions, aiming to bypass end user involvement. While the VLM achieves some success, our findings reveal that having the end user in the loop results in superior performance.**

I. INTRODUCTION

Interactive devices such as Siri, Alexa, and Google Home have brought AI into the home, but the prospect of having embodied AI manipulating household objects remains out of reach. Robotic advancements cannot easily be applied universally. Most current robot learning research focuses on controlled environments or requires re-training to adapt to new settings highlighting the difficulty of obtaining a generalist robot. Home robots will require personalization and adaptation to their specific environments to carry out tasks. We believe that it will be the end-users who actively imbue the domestic robots with skills and behaviors pertinent to the tasks they intend their robots to assist with, thus necessitating the development of robot learning frameworks centered around *end-users*.

We involve end-users in robot manipulation learning by decomposing it into two steps: *factory* model and *home* model. We envision a user receiving a robot programmed with a *factory model* which endows it with primitive capabilities, allowing it to follow basic instructions such as "move right", "close the gripper", or "move above the green jar". The end-user would bootstrap off of these capabilities to help the robot's *factory model* evolve into a personalized *home model*. By instructing the robot through more complex tasks such as "sweep the dust into the dustpan", or "open the top drawer", the user can teach the robot to follow complex instructions and perform longer-horizon tasks.

Learning from demonstration provides an accessible method of robot training through behavior cloning. Our work presents a framework to collect demonstrations by leveraging actions already learned by a robot agent. We show that we can train a robot to perform basic motions and then use those motions to abstract away low-level control. Using natural language, we instruct our robot through skill and task demonstrations that we use to train the robot and expand its capabilities.

We include experiments using a large vision language model (VLM) to see whether it can replace a human breaking down tasks for a robot. While VLMs are getting better, we quantitatively show that end-user in the loop robot learning is still necessary to close the domain gap between low-level reasoning (grounded on the current scene and task) and high-level reasoning (common knowledge).

Through this work, we present an end-user-in-the-loop hierarchical robot learning framework, leveraging natural language for communication.

- We present a method for training a robot to respond to observation-dependent and observation-independent primitive action commands.
- We show language commands for primitive actions can be used in sequence to collect demonstrations for higher level skills and tasks.
- Our results prove that our hierarchical training method leads to performance gains over a state-of-the-art baseline method. **[mention a potential of our framework for the generalization and adaptation of robot manipulation learning? –dk]**

II. RELATED WORK

Our work entails robot manipulation learning from end-user demonstrations in a hierarchical fashion. In this section, we list related work in the areas of learning from demonstrations, human demonstration-data acquisition, using language in robot learning, hierarchical demonstration, and reasoning via large vision-language models.

A. Learning from Demonstrations

Behavior cloning (BC) has garnered significant attention when it comes to robot manipulation task learning from demonstrations [1], [2], [3], [4]. For our proposed framework, we require the policy models to be both generalizable to various skills and sample efficient when it comes to training. BC models such as [2], [5] are designed to output robot end-effector states based on expert demonstrations. RT-2 [1] leveraged vision language models (VLMs) that

C. Winge, A. Imdieke, B. Aldeeb, D. Kang, and K. Desingh are with the Minnesota Robotics Institute, University of Minnesota, Twin Cities, Minneapolis, USA {winge134, imdie022, baldeeb, dongyeop, kdesingh}@umn.edu

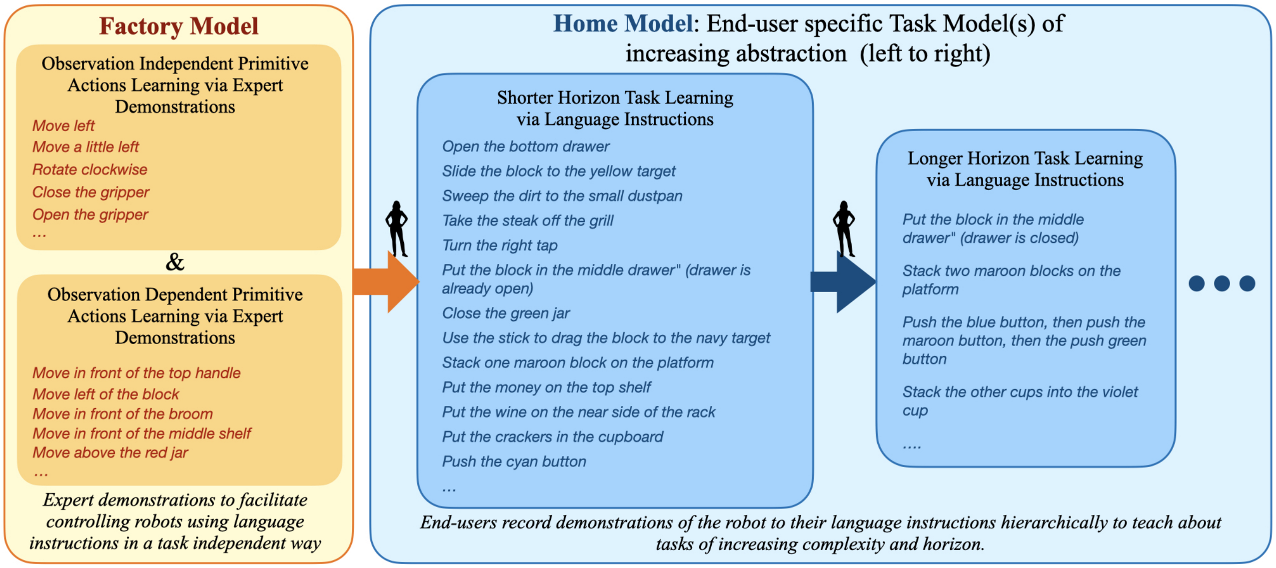


Fig. 1: A *factory model* is trained with a diverse set of motion commands. End users train the robot in their home to complete the tasks they care about. The *factory model* commands consist of primitive actions we call Level-1. These are learned from scripted demonstrations. The *home model* includes skills and tasks that build on Level-1, which we refer to as Level-2 and Level-3. The demonstrations for the *home model* are collected by the end user using language. **[Suggest to increase the font size and reduce the number of instructions. One suggestion: readers may not want to know the whole list of primitive and high-level instructions in this plot. We can only show some of them here and make separate tables/figures in the Experiment sections (A-D) –dk] [By the way, I saw Figure 1 is never mentioned in Page 1 and 2. In fact, I would suggest making a teaser version of this figure and mentioning the conceptual point of the proposed method focusing on the hierarchy of these two steps. Then, we can move this figure to the Framework section with more technical details. –dk]**

generate text containing end-effector pose. While such methods demonstrated impressive success rates, they required prohibitively large amounts of demonstration data.

To address the data collection bottleneck, Wang et al. [6] proposed using videos of human play to augment the training process. Recording the human play data is fast, but they still require data collection using robot teleoperation to transfer the skills to their robot. Shridhar et al. proposed PerAct [3], which demonstrated high accuracy and sample efficiency. The PerAct architecture is suitable for our purposes because it is sample efficient and leverages the RLBench [7] simulation environment, which includes many benchmark manipulation tasks.

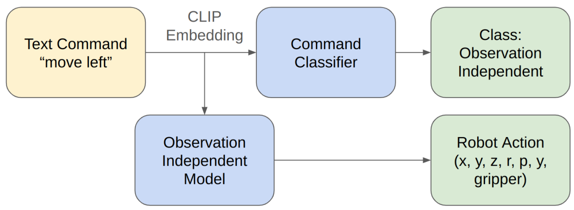
B. Demonstration Data Acquisition

BC models generally require expert demonstrations. However, the tools used for interfacing with robots are not particularly user-friendly. Teleoperation using joysticks or other controllers is commonly used to manipulate the agent's end-effector [6], with some leveraging VR [2], [8] to facilitate the data collection process. ALOHA [9] presented the idea of using a twin robot for the target robot to mimic. While these teleoperation techniques have become more intuitive, they still require expensive equipment that can be difficult to set up. Furthermore, a recent user study [10] shows that it is easier for users to control a robot when they can command meaningful actions instead of directly controlling the robot motion. In this work, we focus on exposing the robot's controls to a non-expert end-user via natural language. Specifically, the *factory model* is trained on expert demonstrations, and *home models* are trained using demonstration data collected via natural language.

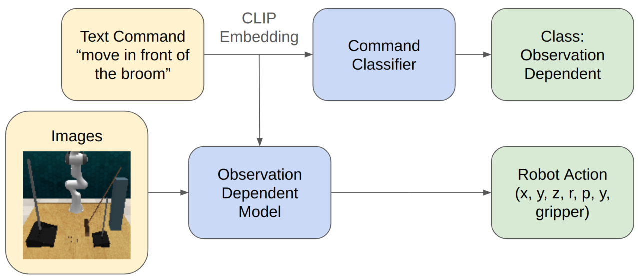
With the progress of large language models and large vision-language models, language is becoming the go-to method for task specification. Language is mostly used to expose the robot's learned skills to an end-user [3], [11], [12]. Lynch et al. [13] bridge the gap between teleoperation and task execution by allowing users to control the robot through speech in real-time. However, unlike our method, their methods do not use language to collect demonstrations.

C. Reasoning via Large Vision-Language Models:

Pre-trained large language models (LLMs) [14], [15] have demonstrated impressive reasoning, prompting a new generation of LLM-based Vision-Language Models (VLMs) that extend LLMs to allow reasoning over visual contexts [12], [1]. These VLMs demonstrated the ability to describe visual scenes and answer questions about them [12] as well as control robots [12]. Given a prompt describing a situation and intention, these VLMs demonstrated the ability to reason over tasks [12] and integrate feedback from their environments [11], [16]. We utilize the publicly available Bard model [17] in our VLM experiments.



(a) If the command classifier determines a command is observation-independent, the observation-independent model uses the text embedding to output a robot action, as shown above.



(b) If the command classifier determines a command in observation-dependent, the observation-dependent model uses the text embedding and the current image observations to output a robot action, as shown above.

Fig. 2: Our architecture includes a command classifier which determines whether to run an observation-dependent or observation-independent model. The *factory model* includes both models. The *home models* are fine-tuned versions of the observation-dependent model.

III. Framework

A. Framework Architecture

Our network architecture consists of an observation-dependent, and an observation-independent model, as shown in Figure 2. The command classifier determines which model to use for a given text command. Both models take in a text instruction and output a robot action. The observation-dependent model takes in RGB-D images in addition to the text instruction. The factory and home models use identical architecture, the home model is trained on more observation-independent commands.

The observation-independent model is an MLP that regresses Δx, Δy, Δz, Δroll, Δpitch, Δyaw, and gripper state (open or close) from a CLIP [18] embedding of the text instruction. The model also takes the previous output as input to maintain the previous gripper state. We train using a set of labeled commands. For example, "move left" is a 10cm move in the negative x direction, and "rotate clockwise" is a 90-degree roll. Some of these commands and labels are shown in Appendix Table VIII.

The observation-dependent model is PerAct [3] with minor modifications. This model has a preprocessing stage, which converts the RGB-D images into a voxel representation. We elected to reduce the voxel dimensions to $50\times50\times50$ from $100\times100\times100$ to save computation, enabling us to run more experiments in a shorter time. Note that our architecture is modular and hence any advancements in feature embedding and efficient observation-dependent models can be integrated easily.

B. Model Levels

The Level-1 model (aka *factory model*) is trained on primitive motions demonstrated by an expert user. In our experiments, these are scripted trajectories from the RL-Bench environment. The primitive motions model includes observation independent commands, and things like moving above or in front of a certain object. Level-2 and Level-3 are skills and task models, respectively, trained by end-users via language-commanded demonstrations. The skills model includes things like picking and placing an object, or pushing an object to a specified location. The tasks model includes things that require repeating a skill or combining multiple skills. Tasks include repeating a pick and place to stack multiple objects or pulling a drawer open, then putting a block in the drawer. Refer to the Figure 1 where Level-1 model is the *factory model* and Level-2 and beyond are *home models*

C. Environments

We selected 14 RLBench tasks, which are a subset of the 18 tasks evaluated in PerAct. Each voxel in our model is 2cm wide, which makes high precision tasks more difficult. Therefore, we eliminated the tasks of screwing in a light bulb, sorting shapes in a shape sorter, placing a ring on a peg, and hanging mugs on a mug tree because they require very high precision. To avoid confusion, we refer to the 14 RLBench tasks as environments. Some of the RLBench tasks are actually skills according to our definitions. Each environment is used to train Level-1 motions, and Level-2 skills. Some environments are also used to train Level-3 tasks.

D. Level-1 Factory Model

The basis of our approach is training a Level-1 (aka *factory model*) on demonstrations of primitive motions. The demonstrations for Level-1 all consist of a single robot motion. We created primitive action demonstrations for each RLBench environment. For example, the *open drawer* environment has 3 primitive actions: "move in front of the top handle," "move in front of the middle handle," and "move in front of the bottom handle." The *factory model* is trained with 1400 scripted demonstrations covering primitive motions across all the environments. In practice, these scripted demonstrations will be replaced by expert demonstrations in the factory setting.

E. Collecting Demonstrations with Language

Once the *factory model* is trained, we no longer need any scripted demonstrations. Demonstrations for more complex skills and tasks are collected using only language instructions typed by the user. Figure 3 shows a demonstration of the Level-1 language commands used to sweep dirt into a dustpan. The keyframes are labeled by the user during the

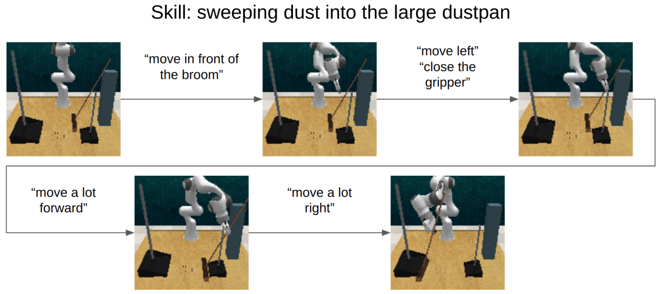


Fig. 3: Primitive motion (Level-1) commands are used to collect a skill (Level-2) demonstration of sweeping dust into the large dustpan.

language-driven demonstration so it can be used to train this skill.

F. Home Model Training

Once the user collects a few demonstrations of a skill or task they desire the robot to learn, they can have the *factory model* fine-tune these demonstrations to create their *home models* (in our case Level-2 and Level-3). Different users may care more about different tasks and only fine-tune for a few of the many possible tasks the robot could learn. This idea is illustrated in Figure 1.

IV. Experiments

In this section, we discuss the details of our experimental setups, and the results of each experiment. For all experiments, model weights are saved after every 5000 steps of training. Every saved weight is evaluated on 25 unseen rollouts of each task the model was trained on. The evaluation episodes are seeded so they initialize repeatably. The best weight from the evaluation is then tested on another set of rollouts. The test episodes are seeded to ensure they are different than the evaluation episodes.

The first experiment determines how well the *factory model* learns to complete primitive actions. We follow that with a language augmentation experiment to see if we can train a *factory model* that is robust to paraphrased inputs.

Next, we compare Level-2 *home models* with a baseline Level-2 model. The *home models* include a multi-skill model, and single-skill models. As part of our Level-2 analysis, we test the Level-2 multi-skill model with Level-1 text commands. We want to make sure the model doesn't forget earlier levels.

We then compare Level-3 *home models* with a baseline Level-3 model. The *home models* include a multi-skill model, and single-skill models.

Last, we experiment with a VLM to see if we can avoid having a human in the loop. We try using the VLM to break down Level-2 skills into Level-1 primitive action commands, and to break down Level-3 tasks into Level-2 skill commands.

A. Learning Level-1 Primitive Actions

For primitive actions, success is determined by the robot end effector reaching within 2.5cm of the target position. The model is given up to 5 action predictions to reach the target position. On average, the robot reaches the correct location 87% of the time. This Level-1 model is a strong foundation for collecting demonstrations for higher level models via language.

B. Language Augmentation

We can't expect end users to give commands identical to those we used. In order to test variations in language, we created 6 total paraphrases of the Level-1 motion instructions. We refer to these as variations $i = 0, 1, 2, 3, 4, 5$. We use a Level-1 model that was only trained on variation $i = 0$ as the default model. We compare it to a Level-1 model that was trained on variations $i = 0, 1, 2, 3$. Variations $i = 4, 5$ were not seen by either model. Variation 4 uses a novel combination of words, but all the individual words were seen in variations $i = 0, 1, 2, 3$. Variation $i = 5$ includes at least one word that was not seen in any of the other variations.

The augmented model shows an 8% improvement in success rate compared to the default model on both test variations, shown in Table I. The success rate is averaged across all the motion commands in all 14 RLBench environments. This shows a model trained on more language variations is more likely to succeed when given paraphrased language commands unseen in training.

Model	Variation	Average Success Rate
Default	4	76
Augmented	4	84
Default	5	75
Augmented	5	83

Table I: Comparison between a Level-1 model trained with no language variation, and an augmented Level-1 model trained with 4 language variations. Variation 4 and variation 5 were not seen by either model during training.

C. Baseline Models

We create baseline models for Level-2 and Level-3 using a more traditional approach. We generate 10 demos for each skill using scripted waypoints in RLBench. Then we train a model on the scripted demos for 100,000 steps. The key differences are that our method trains the model with Level-1 demos first, and our method uses Level-2 demos collected with language instead of using scripted waypoints. The same is true for the Level-3 baseline; it is trained on 10 demos for each task using scripted waypoints.

D. Learning Level-2 Skills from Level-1 Actions

From our 14 RLBench environments, we define 14 Level-2 skills and 4 Level-3 tasks. The skills are listed in Table III. The tasks build on the skills. For example, *put in drawer* includes the skills of *open drawer*, and *put in drawer*.

We train two types of Level-2 models - a) multi-skill model and b) single-skill model. We envision that an end-user would want to train for single-skill models than multi-skill model. So we experiment to understand the performance when multiple skills are learnt at the same time vs. single-skills. We use 10 demos for each skill to train these models. The training starts with the Level-1 weights and fine-tunes for 100,000 steps. During the fine-tuning, each batch samples three times from the Level-1 dataset and once from the Level-2 dataset. This sampling prevents the model from forgetting Level-1 actions. We find that the Level-2 model achieves an average success rate 7% higher than the Level-1 model when tested on Level-1 actions, as shown in Table II, meaning it improves rather than forgets.

Model	Average Success Rate on Level-1 Actions
Level-1	87
Level-2	94

Table II: We test the Level-2 model on Level-1 actions to determine whether it is forgetting Level-1 commands. Rather than forgetting, it shows improvement over the Level-1 model.

The multi-skill and single-skill models are evaluated every 5000 steps on 25 episodes of the skills. The models that perform best on the validation episodes are used for testing, once again on 25 episodes per skill. Our method produces a multi-skill model that achieves an average success rate of 39% compared to the baseline of 23%, as seen in Table III. We believe many important features for learning a skill are learned from the Level-1 actions model. In the open drawer skill, the Level-1 model already learned the concepts of top, middle, and bottom from Level-1. In the push buttons skill, the Level-1 model already learned the 18 button color variations.

E. Learning Level-3 Tasks from Level-1 & 2

To learn Level-3 tasks, we fine-tune for 100,000 steps with Level-3 demos on the best Level-2 models. Fine-tuning here means each batch takes 2 samples from Level-1, 1 sample from Level-2, and 1 sample from Level-3. We train single-task and multi-task models, and the results are shown in Tables V and VII. The baseline multi-task model achieves an average success rate of 10%, whereas our multi-task model achieves 23% using the same number of Level-3 demos. Even our multi-taks model trained on only 5 Level-3 demos achieves a 7% better success rate than the baseline. Curiously, our model trained on 5 demos appears to beat the more thoroughly trained models on the *stack blocks* task, but we attribute that result to chance and deem it statistically insignificant.

F. Using Large Vision Language Models

Inspired by PaLM-E [12], which demonstrated the application of pre-trained LLM-based VLMs to control downstream primitive policies, we explore the possibility of augmenting our Level-1 action and Level-2 skill models with an off-the-shelf VLM to achieve zero-shot task performance. We use Bard (*version 2023.10.30*) as it is the most advanced accessible VLM available for this work, but GPT4 [19] or any other VLM could be used as a substitute. Taking inspiration from [16], we prompt Google's Bard [14] with a list of possible actions and an image containing a front view and the gripper view, as shown in Figure 4. We prompt the VLM to output a description of the image, a list of past executed actions, and whether or not the task has been successfully executed. The VLM is then asked to state the feasibility of each potential next action and choose the best one. This idea is similar to the idea of inner monologue [16], where the prompt draws attention to the image and previous actions. The list of possible actions is compressed to only the relevant actions for a given task. We observe that an off-the-shelf VLM model which has not been trained on any manipulation task fails at lower-level grounded reasoning, but excels in higher-level task planning.

In the following sections, we picture the VLM as a reasoning tool and our Level-1 or Level-2 model as a policy function that the VLM can utilize.

VLM Reasoning over Level-1 Policy: We first provide the VLM with a *factory model* (Level-1) and observe that the VLM fails at lower-level grounded reasoning and thus struggles to perform zero-shot tasks using the *factory model*. In this experiment, the VLM is given a list of Level-1 commands to use to complete the skill. We observe that the VLM fails to perform 3D spatial reasoning and provides poor justification as to why it elected to perform an action. We hypothesize that the VLM reasons at a high level, which is corroborated by the way it describes a scene when prompted. Similarly, the VLM was not trained to comprehend the robot's state and tends to predict that the robot is carrying an object even when the gripper is open. This justifies the need for allowing the user to extend a *factory model* into a skill model (aka Level-2) before pre-trained VLMs can be leveraged to perform zero-shot tasks.

VLM Reasoning over Level-2 Policy: In the second experiment, we provide the VLM with a skill model (Level-2) and observe that it is capable of utilizing it to achieve some success in zero-shot task execution. The VLM is able to complete 2 of the 4 Level-3 tasks shown in Table V. We do not attempt *stack blocks* or *stack cups* since the provided Level-2 policy already has low success rates on the prerequisite skills (9% and 15%, respectively).

Notably, the VLM was able to complete *push buttons* with a 90% success rate over 20 roll-outs, significantly better than the 56% success rate seen by our single-task Level-3 model. This result suggests that the VLM can reason well over a high-level task given a prompt. The VLM was also able to complete *put in drawer*, but achieved only a 25% success rate over 20 roll-outs, which is 15% worse than our best Level-3 model. We observe that a primary reason for the VLM's failure on this task is that the Level-2 policy fails to open the drawer, and the VLM fails to recognize that and does not retry. The robot may also push the drawer closed as it moves to pick up the block, indicating that the skills are not trained well by our approach to be chained together

Model	Average	Open Drawer	Slide Block	Sweep to Dustpan	Meat Off Grill	Turn Tap	Put in Drawer Lv2
Ours 5 demos	30±2	**79±6**	24±8	33±22	5±2	**52±8**	60±16
Ours 10 demos	**39±2**	**79±6**	40±4	**68±11**	15±15	51±14	**84±7**
Baseline 10 demos	23±3	49±9	**43±5**	23±21	**41±2**	8±7	44±31

Close Jar	Drag Stick	Stack Blocks Lv2	Put in Safe	Place Wine	Put in Cupboard	Push Buttons Lv2	Stack Cups Lv2
13±2	**56±7**	5±5	3±2	7±2	0	73±6	8±4
28±7	51±27	**9±5**	16±7	8±7	0	**79±10**	**15±2**
3±2	24±7	7±6	**21±6**	**23±8**	0	24±11	9±5

Table III: Results for the multi-skill Level-2 model compared to a baseline. Results are an average and standard deviation from 3 training runs with different initializations.

Average	Open Drawer	Slide Block	Sweep to Dustpan	Meat Off Grill	Turn Tap	Put in Drawer Lv2	Close Jar
50	84	60	80	36	68	92	32
Drag Stick	Stack Blocks Lv2	Put in Safe	Place Wine	Put in Cupboard	Push Buttons Lv2	Stack Cups Lv2	
80	4	28	32	0	80	20	

Table IV: Performance for Level-2 models fine-tuned for a single task

Model	Average	Put in Drawer	Stack Blocks	Push Buttons	Stack Cups
Ours 5 demos	17±2	28±11	**3±5**	39±6	0
Ours 10 demos	**23±6**	**40±8**	0	**53±17**	0
Baseline 10 demos	10±2	0	0	39±9	0

Table V: Success rates for Level-3 tasks with multitask models. Numbers are the average and standard deviation from 3 models with different initializations.

Demos	Average	Put in Drawer	Stack Blocks	Push Buttons	Stack Cups
5 demos	25	40	4	56	0
10 demos	24	36	0	56	4

Table VI: Performance for Level-3 models fine-tuned for a single task

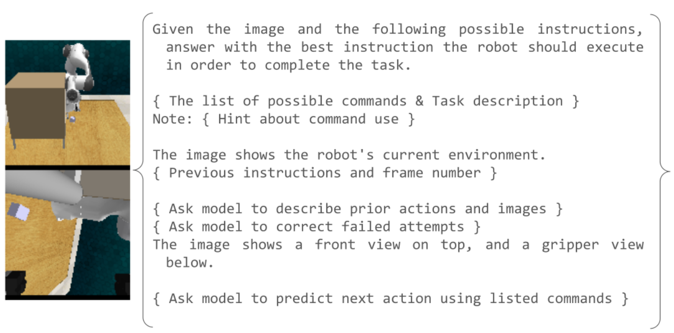


Fig. 4: The prompt template shown above is used to query the VLM for the next actions. Every action proposed is executed by the policy for 8 steps. The images are updated after every execution.

directly.

Demos	Put in Drawer	Stack Blocks	Push Buttons	Stack Cups
Best L3	40	4	56	4
VLM	25	-	90	-

Table VII: Performance of best Level-3 models fine-tuned for a single-task against VLM using Level-2 motions

V. Conclusion

We present a framework that is designed for end-users to train a robot to perform a variety of skills and tasks. By providing a *factory model* capable of following language instructions for primitive motions, we show long horizon demonstrations can be collected using only natural language. *Home models* shows superior performance when compared to a model trained on skill and task demonstrations directly. Our hierarchical training method achieves a 1.7x improvement on Level-2 skills, and a 2.3x improvement on Level-3 tasks compared to the baseline method. We hope that the findings of this work can encourage research toward end-user-in-the-loop robot training methods.

We attempt to replace the human in the loop with a VLM, but find the VLM falls short. The VLM lacks the spatial reasoning abilities to make fine adjustments to the robot position to complete Level-2 skills using Level-1 primitive action commands. The VLM shows better results using Level-2 skill commands to complete Level-3 tasks. These results show potential benefits of coupling our system with a VLM.

One limitation of this work is the limited selection of training environments. The *factory model* must be trained with data from a wide variety of tasks and environments to function in homes. Increasing the variety of training environments is a potential future work.

References

[1] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, X. Chen, K. Choromanski, T. Ding, D. Driess, A. Dubey, C. Finn, P. Florence, C. Fu, M. G. Arenas, K. Gopalakrishnan, K. Han, K. Hausman, A. Herzog, J. Hsu, B. Ichter, A. Irpan, N. Joshi, R. Julian, D. Kalashnikov, Y. Kuang, I. Leal, L. Lee, T.-W. E. Lee, S. Levine, Y. Lu, H. Michalewski, I. Mordatch, K. Pertsch, K. Rao, K. Reymann,

M. Ryoo, G. Salazar, P. Sanketi, P. Sermanet, J. Singh, A. Singh, R. Soricut, H. Tran, V. Vanhoucke, Q. Vuong, A. Wahid, S. Welker, P. Wohlhart, J. Wu, F. Xia, T. Xiao, P. Xu, S. Xu, T. Yu, and B. Zitkovich, "Rt-2: Vision-language-action models transfer web knowledge to robotic control," in *arXiv preprint arXiv:2307.15818*, 2023.

[2] E. Jang, A. Irpan, M. Khansari, D. Kappler, F. Ebert, C. Lynch, S. Levine, and C. Finn, "BC-z: Zero-shot task generalization with robotic imitation learning," in *5th Annual Conference on Robot Learning*, 2021.

[3] M. Shridhar, L. Manuelli, and D. Fox, "Perceiver-actor: A multi-task transformer for robotic manipulation," in *Proceedings of the 6th Conference on Robot Learning (CoRL)*, 2022.

[4] T. Gervet, Z. Xian, N. Gkanatsios, and K. Fragkiadaki, "Act3d: Infinite resolution action detection transformer for robotic manipulation," *arXiv preprint arXiv:2306.17817*, 2023.

[5] A. Stone, T. Xiao, Y. Lu, K. Gopalakrishnan, K.-H. Lee, Q. Vuong, P. Wohlhart, S. Kirmani, B. Zitkovich, F. Xia, C. Finn, and K. Hausman, "Open-world object manipulation using pre-trained vision-language model," in *arXiv preprint*, 2023.

[6] C. Wang, L. Fan, J. Sun, R. Zhang, L. Fei-Fei, D. Xu, Y. Zhu, and A. Anandkumar, "Mimicplay: Long-horizon imitation learning by watching human play," *arXiv preprint arXiv:2302.12422*, 2023.

[7] S. James, Z. Ma, D. Rovick Arrojo, and A. J. Davison, "Rlbench: The robot learning benchmark & learning environment," *IEEE Robotics and Automation Letters*, 2020.

[8] H. M. Clever, A. Handa, H. Mazhar, K. Parker, O. Shapira, Q. Wan, Y. Narang, I. Akinola, M. Cakmak, and D. Fox, "Assistive tele-op: Leveraging transformers to collect robotic task demonstrations," 2021.

[9] T. Z. Zhao, V. Kumar, S. Levine, and C. Finn, "Learning fine-grained bimanual manipulation with low-cost hardware," 2023.

[10] K. Mahadevan, Y. Chen, M. Cakmak, A. Tang, and T. Grossman, "Mimic: In-situ recording and re-use of demonstrations to support robot teleoperation," in *Proceedings of the 35th Annual ACM Symposium on User Interface Software and Technology*, UIST '22, (New York, NY, USA), Association for Computing Machinery, 2022.

[11] 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.

[12] D. Driess, F. Xia, M. S. Sajjadi, C. Lynch, A. Chowdhery, B. Ichter, A. Wahid, J. Tompson, Q. Vuong, T. Yu, *et al.*, "Palm-e: An embodied multimodal language model," *arXiv preprint arXiv:2303.03378*, 2023.

[13] C. Lynch, A. Wahid, J. Tompson, T. Ding, J. Betker, R. Baruch, T. Armstrong, and P. Florence, "Interactive language: Talking to robots in real time," 2022.

[14] R. Anil, A. M. Dai, O. Firat, M. Johnson, D. Lepikhin, A. Passos, S. Shakeri, E. Taropa, P. Bailey, Z. Chen, E. Chu, J. H. Clark, L. E. Shafey, Y. Huang, K. Meier-Hellstern, G. Mishra, E. Moreira, M. Omernick, K. Robinson, S. Ruder, Y. Tay, K. Xiao, Y. Xu, Y. Zhang, G. H. Abrego, J. Ahn, J. Austin, P. Barham, J. Botha, J. Bradbury, S. Brahma, K. Brooks, M. Catasta, Y. Cheng, C. Cherry, C. A. Choquette-Choo, A. Chowdhery, C. Crepy, S. Dave, M. Dehghani, S. Dev, J. Devlin, M. Díaz, N. Du, E. Dyer, V. Feinberg, F. Feng, V. Fienber, M. Freitag, X. Garcia, S. Gehrmann, L. Gonzalez, G. Gur-Ari, S. Hand, H. Hashemi, L. Hou, J. Howland, A. Hu, J. Hui, J. Hurwitz, M. Isard, A. Ittycheriah, M. Jagielski, W. Jia, K. Kenealy, M. Krikun, S. Kudugunta, C. Lan, K. Lee, B. Lee, E. Li, M. Li, W. Li, Y. Li, J. Li, H. Lim, H. Lin, Z. Liu, F. Liu, M. Maggioni, A. Mahendru, J. Maynez, V. Misra, M. Moussalem, Z. Nado, J. Nham, E. Ni, A. Nystrom, A. Parrish, M. Pellat, M. Polacek, A. Polozov, R. Pope, S. Qiao, E. Reif, B. Richter, P. Riley, A. C. Ros, A. Roy, B. Saeta, R. Samuel, R. Shelby, A. Slone, D. Smilkov, D. R. So, D. Sohn, S. Tokumine, D. Valter, V. Vasudevan, K. Vodrahalli, X. Wang, P. Wang, Z. Wang, T. Wang, J. Wieting, Y. Wu, K. Xu, Y. Xu, L. Xue, P. Yin, J. Yu, Q. Zhang, S. Zheng, C. Zheng, W. Zhou, D. Zhou, S. Petrov, and Y. Wu, "Palm 2 technical report," 2023.

[15] 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.

[16] W. Huang, F. Xia, T. Xiao, H. Chan, J. Liang, P. Florence, A. Zeng, J. Tompson, I. Mordatch, Y. Chebotar, P. Sermanet, N. Brown, T. Jackson, L. Luu, S. Levine, K. Hausman, and B. Ichter, "Inner monologue: Embodied reasoning through planning with language models," 2022.

[17] Google AI, "Bard, a large language model developed by Google AI." https://www.bard.google.com, 2023. Version 2023.10.30.

[18] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever, "Learning transferable visual models from natural language supervision," in *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event* (M. Meila and T. Zhang, eds.), vol. 139 of *Proceedings of Machine Learning Research*, pp. 8748–8763, PMLR, 2021.

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

APPENDIX

A. Training Data

This section provides more information on the training of the level 1 factory model. Table VIII shows examples of labels for observation-independent commands. Table IX gives the success rate breakdown of the primitive actions in each RLBench environment. The motions for the *turn tap* environment have a low success rate because the robot is trying to move directly to a grasp position. We train a pre-grasp position for most environments, but in the *turn tap* environment, the tap handles are at angles difficult to describe with language. The actions for the *put in cupboard* environment also have a low success rate. This is because the grocery items are not easy to distinguish in our low-resolution voxel space. Some of the box shaped or cylinder shaped items are easily confused with each other.

Command	Δx	Δy	Δz	Δroll	Δpitch	Δyaw	gripper
move left	-10	0	0	0	0	0	1
move a little left	-5	0	0	0	0	0	1
move a lot left	-20	0	0	0	0	0	1
move a tiny bit left	-1	0	0	0	0	0	1
move forward	0	10	0	0	0	0	1
move up	0	0	10	0	0	0	1
move backward and down	0	-10	-10	0	0	0	1
rotate clockwise	0	0	0	90	0	0	1
turn left	0	0	0	0	0	-90	1
turn up	0	0	0	0	-90	0	1
close the gripper	0	0	0	0	0	0	0

Table VIII: Selected examples of commands and labels used to train the observation-independent model

Environment	Motions	Success Rate
Open Drawer	move in front of the {top, middle, bottom} handle	96
Slide Block	move {in front of, behind, left of, right of} the block	100
Sweep to Dustpan	move in front of the broom	100
Meat Off Grill	move above the {steak, chicken}	100
Turn Tap	move to the {left, right} tap	36
Put in Drawer	move above the block	92
Close Jar	move above the {color} jar, move above the lid	92
Drag Stick	move above the sick	100
Put in Safe	move above the money, move in front of the {top, middle, bottom} shelf	96
Place Wine	move in front of the wine bottle, move in front of the {near side, middle, far side} of the rack	96
Put in Cupboard	move above the {item}, move in front of the cupboard	48
Push Buttons	move above the {color} button	92
Stack Cups	move above the left edge of the {color} cup	88

Table IX: Level 1 motions model success rates for 25 test episodes in each environment

B. VLM Experiment Details

This section provides examples and details of our VLM experiments. Figure 5 shows a successful use of the VLM whereas Figure 6 shows a failure case. Figure 4 shows an example of a full detailed prompt used in our VLM experiment.

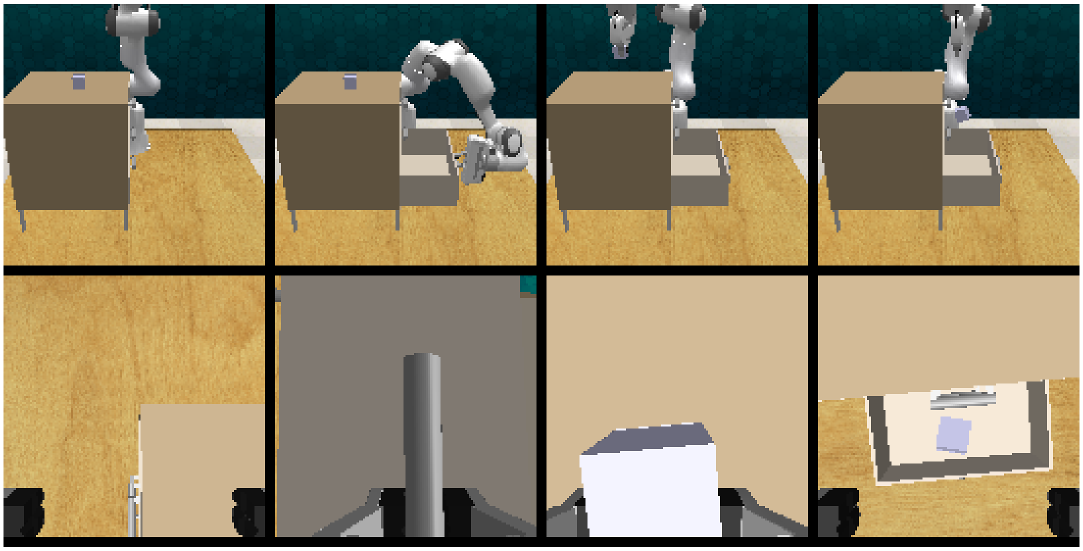


Fig. 5: VLM success for *Put in Drawer*. The VLM correctly predicts the correct sequence of commands (left to right): *open the bottom drawer*, *put the block in the bottom drawer*, *move a lot left*

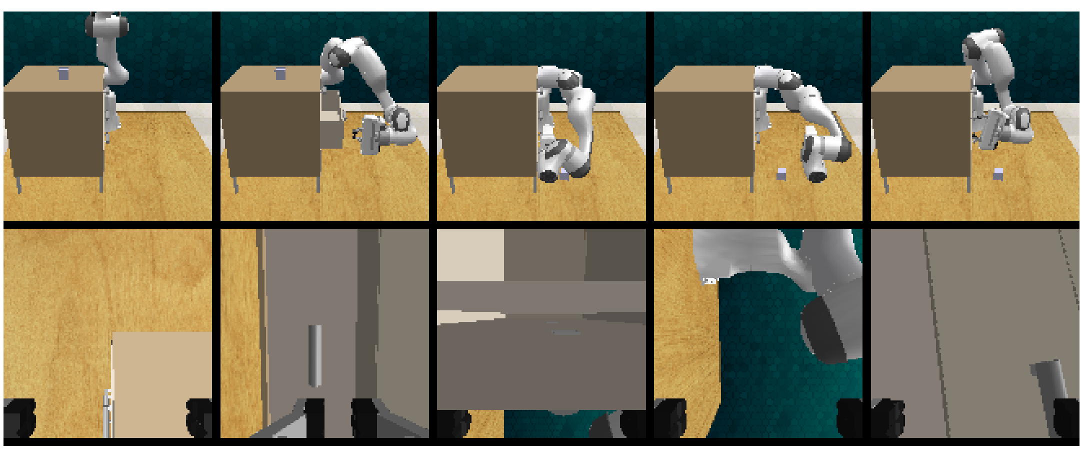


Fig. 6: VLM failure for *Put in Drawer*. The VLM correctly predicts the correct sequence of commands, but the level 2 model fails to execute them correctly. Predicted commands (left to right): *open the middle drawer*, *put the block in the middle drawer*, *move a lot left*, *put the block in the middle drawer*.

Given the image and the following possible instructions, answer with the best instruction the robot should execute in order to complete the task.

The list of possible commands are:
open the ["top", "middle", "bottom"] drawer
move a lot left
put the block in the ["top", "middle", "bottom"] drawer

The task is to "put the block in the top drawer".
Note: *move a lot left* is a command that moves the robot away from the drawer, so it does not bump into it when putting the block in the drawer.

The image shows the robot's current environment.
Previous instructions in order of execution, the number represents the frame number when the command was given:
"open the top drawer"; 1,
"put the block in the top drawer"; 104,
The current time is 385.

List your previous actions, and describe if you think they were completed correctly, then describe the image with relation to those actions.
If previous instructions were not executed correctly, predict an action that will correct the mistake.
The image shows a front view on top, and a gripper view below.

After that, please predict the next correct action and explain why it is the best for the task: "put the block in the top drawer".
Use only the commands listed above.

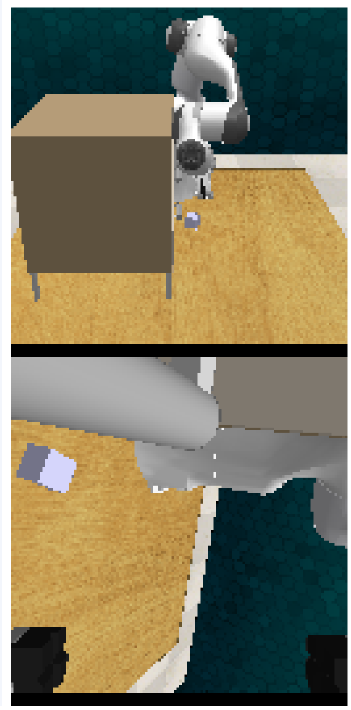


Fig. 7: Text prompt for VLM tasks. The VLM is given a prompt with the list of possible actions, and the history of previous actions, along with a task in each prompt. With each prompt, the images for the current gripper and front view are included. The output of the VLM is used in the L2 or L3 models. Once the L2 or L3 action is completed, the process repeats.