Talk Through It: End User Directed Manipulation Learning

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

***Abstract*— Training robots to perform a huge range of tasks in many different environments is immensely difficult. Instead, we propose selectively training robots based on end-user preferences.**

Given a vision and language conditioned *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'). Our method results in a 13% improvement in task success rates compared to a baseline method.

We also 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. The VLM is unable to break tasks down to our lowest level, but does achieve good results breaking high-level tasks into mid-level skills. We have a supplemental video and additional results at `talk-through-it.github.io`.

I. INTRODUCTION

Interactive devices such as Alexa and Google Home have brought AI into the home, but the prospect of having embodied AI manipulating household objects remains out of reach. Large Vision-Language Models (VLMs) demonstrated the capacity to effectively control mobile robots. However, those models have yet to demonstrate highly successful 3D object manipulation. We believe using a VLM on top of a robotic manipulation model is currently the best strategy.

We can't expect to pre-train any model to complete every task an end user might want. We believe the robotic manipulation model must be capable of learning from the end user, necessitating the development of robot learning frameworks centered around *end users*. We take a two step approach: creating a *factory model* and creating a *home model*, as shown in Figure 1. We envision a user receiving a robot programmed with a *factory model* which endows it with primitive capabilities, allowing it to follow basic natural language instructions such as "move right", "close the gripper", or "move above the green jar". The end user would bootstrap off of these capabilities to direct the robot's *factory model* to evolve into a personalized *home model*. By instructing the robot through more complex skills such as "sweep the dust into the dustpan", or "open the top drawer", the user can collect demonstrations and teach the robot to follow more complex instructions.

We include experiments using the VLM Bard to see whether it can replace a human breaking down tasks for a robot. While the VLM can break a task into skills, it cannot break skills into primitive actions.

Our main contributions are as follows.

- We present a method for training a robot to respond to observation-dependent and observation-independent language commands.
- We show language commands for primitive actions can be used in sequence to collect demonstrations for training higher-level skills and tasks. This technique allows non-expert end users to train a robot.
- Our results prove that our hierarchical training method leads to performance gains over a state-of-the-art baseline method.
- We demonstrate a VLM can successfully chain skills learned by our model to complete longer-horizon tasks.

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

II. RELATED WORK

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 a policy that can learn many skills in a sample efficient manner. Models such as BC-Z [2] and MOO [5] are designed to output robot end-effector states based on expert demonstrations. While such methods demonstrated impressive success rates, they require 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 human play data is fast, but they still require robot teleoperation demonstrations to transfer the skills to their robot. Shridhar et al. proposed PerAct [3], which demonstrated high accuracy and sample efficiency in the RLBench simulation environment. We chose to utilize the PerAct architecture in our framework.

B. Demonstration Data Acquisition

BC models require expert demonstrations. The tools used for interfacing with robots are not particularly user-friendly. Teleoperation using game controllers is commonly used [6], with some works leveraging VR [2], [7] to facilitate data collection. ALOHA [8] uses a twin robot for the target robot to mimic. While teleoperation techniques can be intuitive, they still require expensive equipment that can be difficult to set up. In our work, we expose the robot's controls via natural language.

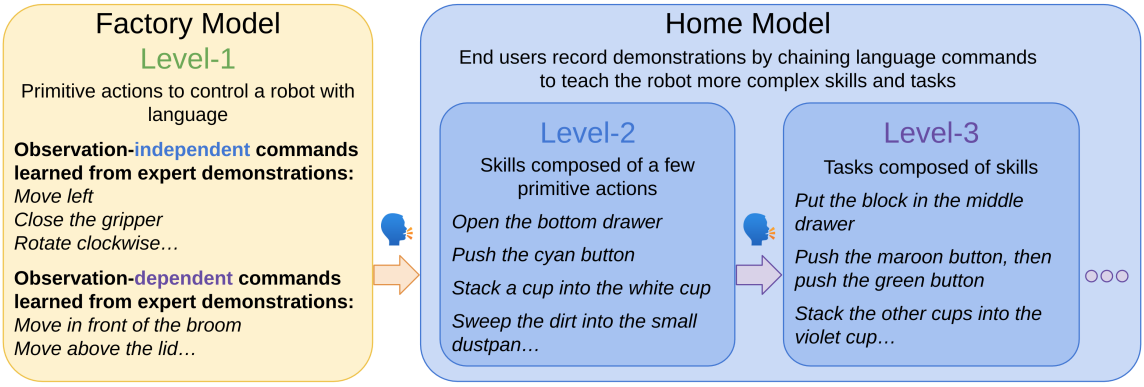


Fig. 1: The Level-1 *factory model* is trained on scripted demonstrations to perform primitive actions from language commands. An end user trains the robot to perform Level-2 skills in their home by using the Level-1 action commands to collect demonstrations of desired skills. They can then train Level-3 tasks by utilizing Level-1 action commands and Level-2 skill commands to collect demonstrations of desired tasks. These demonstrations collected by the end user only use natural language; no programming or special hardware is required. Different end users may choose to train different skills and tasks according to their needs.

C. Reasoning via Large Vision-Language Models

LLM-based VLMs extend LLMs to allow reasoning over visual contexts [9], [1]. VLMs demonstrate the ability to describe visual scenes and answer questions about them, as well as control robots [9]. Given a prompt describing a situation and intention, VLMs demonstrated the ability to reason over tasks [9] and integrate feedback from their environments [10], [11]. We utilize Bard [12] for evaluating the need for human intervention and hierarchical learning.

III. Framework

A. Framework Architecture

Our 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. This architecture applies to the *factory model* and the *home model* illustrated in Figure 1.

The observation-independent model is a multi-layer perceptron that regresses Δx, Δy, Δz, Δroll, Δpitch, Δyaw, and gripper state from a CLIP [13] embedding of the 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, whereas "move a little left" prompts a 5cm move instead. "Rotate clockwise" is a 90-degree roll. More commands can be seen in Appendix Table IV.

The observation-dependent model is PerAct [3] with 2cm voxel size. Note that our architecture is modular so future improved models can be easily integrated.

B. Model Levels

The Level-1 model is trained on primitive actions demonstrated by a scripted expert in RLBench. Primitive actions include both observation-independent commands (e.g. 'Move a little right'), and observation-dependent commands (e.g. 'Move above the block'). Level-2 and Level-3 are skill and task models, respectively, trained by end users via language-commanded demonstrations. The skills model includes things like picking and placing, or pushing an object. The tasks model includes things that require repeating a skill or combining multiple skills.

C. Environments

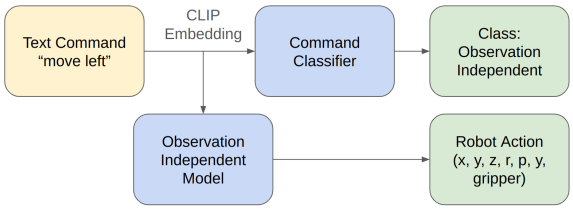
We selected 14 RLBench tasks, which are a subset of the 18 tasks evaluated in PerAct. Each voxel in our model is 2cm wide, which makes high precision tasks more difficult. Therefore, we eliminated 4 high precision tasks we wouldn't succeed on. To avoid confusion, we refer to the 14 RLBench tasks as environments. Some of the RLBench tasks are actually skills according to our definitions.

D. Level-1 Factory Model

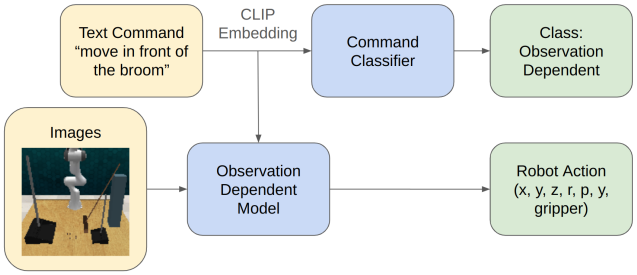
The basis of our approach is training a Level-1 model on demonstrations of primitive actions. The demonstrations for Level-1 all consist of a single robot motion. We created primitive action demonstrations for each RLBench environment. For example, the *open drawer* environment has 3 primitive actions: "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 actions 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 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



(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 is 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* and *home model* include both models. The observation-dependent model is fine-tuned in the *home model*.

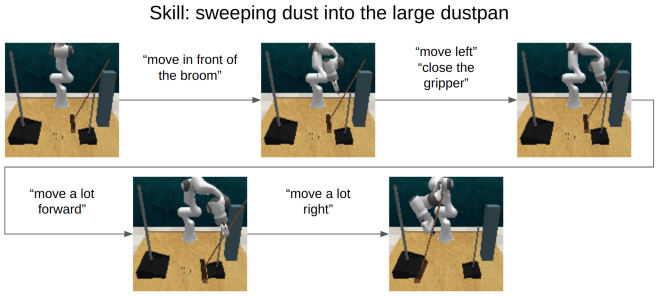


Fig. 3: Primitive motion (Level-1) commands are used to collect a skill (Level-2) demonstration of sweeping dust into the large dustpan.

the Level-1 language commands used to sweep dirt into a dustpan.

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*.

IV. Experiments & Results

For all experiments, model weights are saved after every 5000 steps of training. Every saved weight is evaluated on 25 unseen rollouts of each skill or task the model was trained on. The best weight from the evaluation is then tested on another set of unseen rollouts.

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

In order to test variations in language, we created 6 total paraphrases of the Level-1 action instructions. 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 $i = 4$ uses a novel combination of words from earlier variations. Variation $i = 5$ includes at least one unseen word.

The augmented model shows an 8% improvement in success rate compared to the default model on both test variations, showing a model trained on more language variations is more likely to succeed when given paraphrased commands.

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.

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 I. The tasks build on the skills. For example, *put in drawer* includes the skills of *open drawer*, and *put in drawer*.

We train a Level-2 multi-skill model, and Level-2 single skill models. We use 10 demos for each skill to fine-tune the Level-1 model to learn the Level-2 skills. Our multi-skill model achieves an average success rate of 39% compared to the baseline of 23%, as seen in Table I. 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. When trained on a single skill, success rates are 11% higher on average. Therefore, users who want to train fewer skills will likely see better success rates.

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

To learn Level-3 tasks, we fine-tune with Level-3 demos on the best Level-2 models. The multi-task model results are shown in Table II. The baseline multi-task model achieves an average success rate of 10%, whereas our multi-task model achieves 23%.

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 I: Success rates for the multi-skill Level-2 model compared to a baseline. Success rates are the mean and standard deviation from tests on 3 models initialized randomly before training.

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 II: Success rates for Level-3 tasks with multi-task models. Success rates are the mean and standard deviation from tests on 3 models initialized randomly before training.

Model	Average	Put in Drawer	Push Buttons
Best L3	48	40	56
VLM	57.5	25	90

Table III: Success rates of Level-3 single-task models compared to the VLM using a Level-2 multi-skill policy

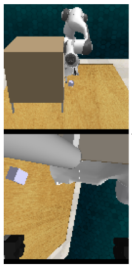
F. Using Large Vision-Language Models

We explore utilizing VLM-generated lower-level instructions to complete higher-level skills or tasks. We prompt the VLM Bard (*version 2023.10.30*) with a list of possible actions and an image containing a front view and a gripper view, as shown in Figure 4. The VLM is then asked to state the feasibility of each potential next action and choose the best one. The list of possible actions is reduced to only the relevant actions for a given task.

VLM Reasoning over Level-1 Policy: In this experiment, the VLM is given a list of Level-1 commands to use to complete a Level-2 skill. Each command it selects is executed by our trained Level-1 model. 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.

VLM Reasoning over Level-2 Policy: We provide the VLM with a Level-2 multi-skill model and observe that it is capable of utilizing it to achieve some success in zero-shot task execution. The VLM is able to complete the *put in drawer* and *push buttons* tasks. We do not attempt *stack blocks* or *stack cups* since the provided Level-2 policy already has low success rates on the prerequisite skills.

On average, the VLM performs better than the trained Level-3 models, as shown in Table III. This result suggests that the VLM can reason well over a high-level task given a prompt. We observe that a primary reason for the VLM's failures on *put in drawer* is that the Level-2 policy fails to open the drawer, and the VLM fails to recognize that and does not retry. An example is shown in Appendix Figures 5 and 6.



```
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 & Task description }
Note: { Hint about command use }

The image shows the robot's current environment.
{ Previous instructions and frame number }

{ Ask model to describe prior actions and images }
{ Ask model to correct failed attempts }
The image shows a front view on top, and a gripper view
  below.

{ Ask model to predict next action using listed commands }
```

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.

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 actions, we show longer-horizon demonstrations can be collected using only natural language. Our hierarchical training method produces *home models* that achieve a 1.7x improvement on Level-2 skills, and a 2.3x improvement on Level-3 tasks compared to the baseline method.

We attempt to replace the human in the loop with a VLM, but find the VLM falls short on low-level reasoning. 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.

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] 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.

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

[9] 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.

[10] 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.

[11] 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.

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

[13] 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.

APPENDIX

A. Observation-Independent Model

To endow the robot with primitive capabilities we train it to follow the "move" and "turn" instructions listed in Table IV. Those commands shape the capacity of our low-level *factory model*.

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

B. Training Data

This section provides more information on the training of the Level-1 *factory model*. Table IV shows examples of labels for observation-independent commands. Table V 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.

C. VLM Experiment Details

This section provides examples and details of our VLM experiments. These experiments involved prompting the VLM with task objective details, possible actions to perform, a list of previously performed actions, and a description of the requested output. Alongside the description, images showing the current state of the robot are also provided. Figure 7 shows an example of a complete prompt used in our VLM experiment.

D. VLM Experiment Results

As mentioned in our results section, a pre-trained VLM does well at decomposing level-3 (high-level) tasks into level-2 skills but some failure cases remain. In some cases, the pre-trained VLM fails to recognize a missing step in the task causing it to fail. Figure 5 shows a successful use of the VLM whereas Figure 6 shows a failure case.

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 stick	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 V: Level-1 actions model success rates for 25 test episodes in each environment

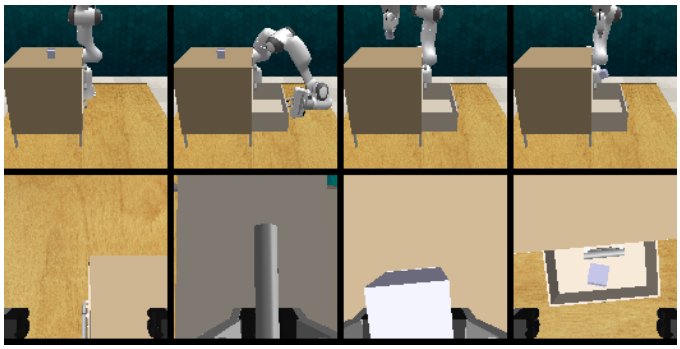


Fig. 5: VLM success for *Put in Drawer*. The VLM correctly predicts the sequence of commands (left to right): *open the bottom drawer*, *put the block in the bottom drawer*, *move a lot left* (irrelevant since the task is complete).

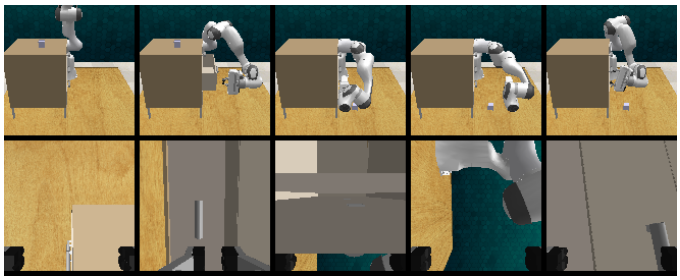


Fig. 6: VLM failure for *Put in Drawer*. The VLM correctly predicts the sequence of commands, but the Level-2 model fails to execute them correctly. Predicted commands (left to right): *open the middle drawer*, *put the block in the middle drawer*, *move a lot left*, *put the block in the middle drawer*.

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.

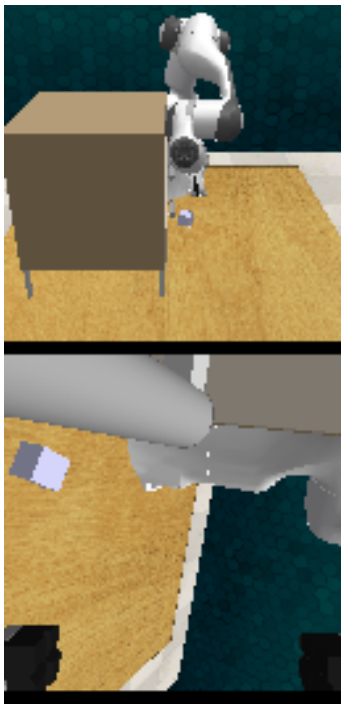


Fig. 7: Text prompt for VLM tasks. The VLM is given a prompt with the list of possible actions, and the history of previous actions, along with a task in each prompt. Each prompt includes the current images from the front view and gripper view. The output of the VLM is sent to the Level-2 skills model. The Level-2 skills model is allowed to run for 8 steps, then the VLM is prompted with the updated images.