

A Environment Details

We evaluate our method in a real-world tabletop manipulation setup. We use a 6DOF WidowX-250 robot interacting with various objects both inside and outside of our training distribution at 5 Hz. We use one 640×480 RGB camera mounted on top of the model as set up in BridgeData [5].

We evaluate our method in the following scenes, which include:

Sweep: This scene involves an object manipulation as well as sweeping task unseen in the BridgeData’s initial training trajectories.

mint: Placing the mushroom in the pot, then sweep the mints on the right using the towel.

skittles: Instead of using mints and towel for sweeping, we use a swiffer and skittles instead.

Drawer: This scene involves using a drawer and perform manipulation within the space of the drawer.

put in: Open the drawer, and then put a purple object (beet/sweet potato) inside the drawer.

pry away: A pot is stored inside the drawer space, and the robot must use a ladle to pry away the pot within drawer.

Bowl: This scene involves object manipulation to a bowl and perform long-horizon or 6DOF manipulation.

salad: This task requires sequential object manipulation by putting a corn cob and a mushroom in the bowl.

pouring: This task requires the robot to grasp a scoop and pour almonds inside the scoop into the bowl.

Rotation: This scene involves rotating a spoon and a marker to fit into a white container not aligned with the pen/marker, and naive pick-and-place will not correctly align the object into the container.

spoon: Placing the spoon in the container placed on the left side of the table.

marker: Replacing the spoon with the marker and randomize location of the container while being misaligned.

We summarize the evaluation tasks in Table 1. Frames from each evaluation task are presented in Fig. 7.

Table 1: Task Breakdown

Scene	Task	Long-Horizon?	6DO required?	Instruction
Drawer	put in	Yes	Yes	“put the beet toy/purple thing into the drawer.”
	pry away	Yes	Yes	“pry out the pot in the drawer using the ladle.”
Bowl	salad	Yes	No	“make a salad bowl with corn and mushroom.”
	pour scoop	No	Yes	“pour the contents of the scoop into the bowl.”
Sweep	mints	Yes	No	“sweep the mints to the right after putting the mushroom in the bowl.”
	skittles	Yes	No	“sweep the skittles into the bin after putting the mushroom in the container.”
Rotation	marker	No	Yes	“put the marker into the box while aligning it.”
	spoon	No	Yes	“put the spoon into the cleaner while aligning it.”

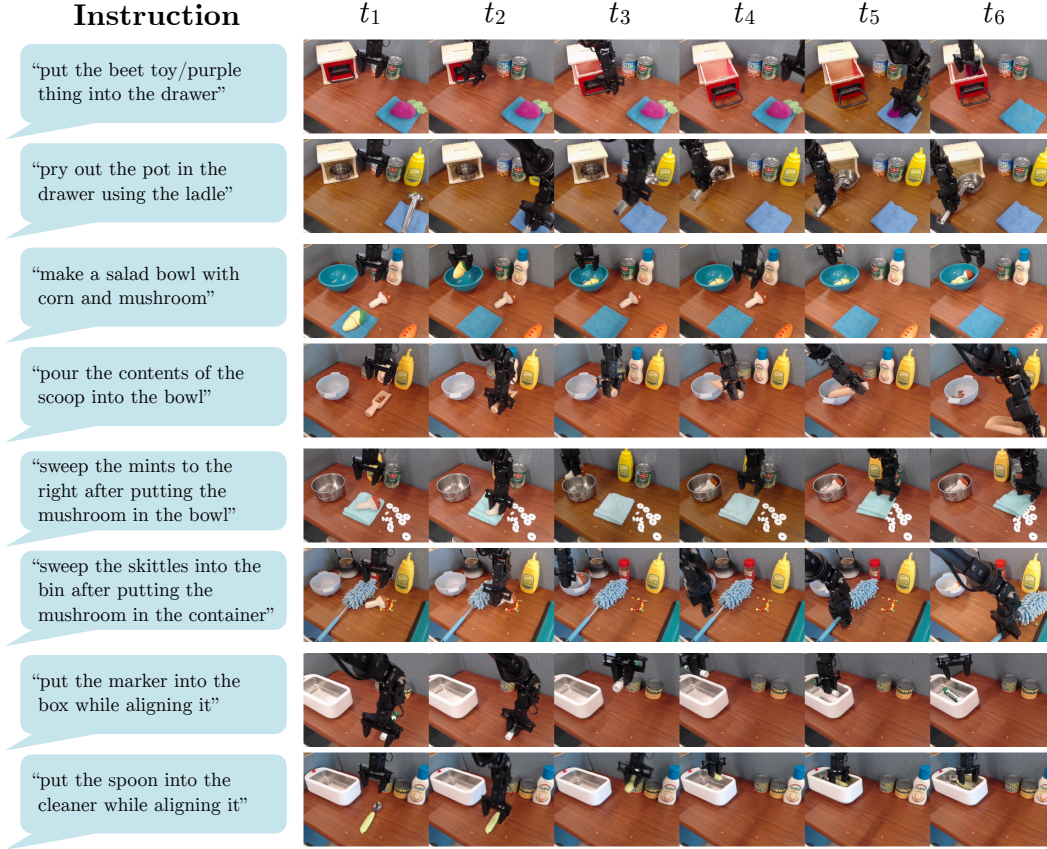


Figure 7: Frames from each evaluation task and language instruction.

B Training Details

We train on an augmented version of the BridgeDataV2 dataset [5]. We algorithmically augment the dataset with low-level instructions using heuristics designed over the proprioceptive states of the robot and incorporate language context by parsing the language instruction using a language model. We use the Adam optimizer [63] to minimize the loss function in Eq. (3).

Instead of naively looping through Algorithm 1, we batch our implementation with the exception of the outermost for loop, thus reducing time consumption during optimization by a significant margin via vectorization. We record an empirical time consumption of 470 seconds for our language optimization module on computations ran on a V4 TPU module, in which only 200 seconds are required for sampling 20000 different partitions to complete the optimization for all of the 15 sets of language instructions. We save our optimal plans for future use, thus reducing overhead even more.

We encode both language instructions using a frozen MUSE model [60] before passing them into the main ResNet with FiLM layers [61].

B.1 Hyperparameter Selection

We discuss the hyperparameters used in our method and baselines.

B.1.1 Policy Training

We set our learning rate for our Adam Optimizer [63] to $3 \cdot 10^{-4}$ and a dropout rate of 0.1 in our policy head. We employ random resizing and cropping, contrast, brightness, saturation, and hue for input images. We train our policy for 300,000 steps, in which we use the checkpoint with the lowest validation MSE. The total training time takes 12 hours when trained on 4 TPU pods.

468 **B.1.2 Language Decomposition Optimization**

469 During optimization, we sample 15 random instruction sets from GPT4-o, and we use 20,000 sam-
470 pling steps in order to find the best subtask decomposition.

471 In order to batch across demonstrations, which have different trajectory lengths, we pad our trajec-
472 tories to a certain length (200 for long-horizon tasks, 150 for non long-horizon tasks). We sum the
473 squared difference between generated action and oracle action in evaluation, thus giving a consistent
474 baseline not affected by the length of the trajectory.

475 **B.2 Baseline Details**

476 We finetune an Octo-small [28] model that is trained on BridgeData [5] in order to perform few-shot
477 learning on the collected trajectories for the baseline in Table 2. We use an Adam optimizer [63]
478 with a learning rate of $3 \cdot 10^{-4}$ and finetune our model’s action head for 5000 steps. We use the
479 hyperparameters set by Octo for the rest of the settings.

480 In order to perform tasks in long-horizon, we assign a language label for each task in order to trans-
481 plant semantic understanding from human into Octo. The same language instruction for PALO eval-
482 uation is also used for Octo finetuning.

483 **C Augmentation Details**

484 We augment our training data by decomposing into segments of translation, rotation, and gripper
485 movement data. To achieve this, we use proprioception and use a language model to extract the
486 target object to create context of language instruction.

487 We chunk actions within training data into segments of length 4 and evaluate the low level instruction
488 within these segments and append them into the training data.

489 **C.1 Proprioception**

490 We use standard deviation of each action against the metadata of BridgeData [5] and determine how
491 to describe the proprioception of the label. We determine the biggest direction in which the gripper
492 is moving (up, down, left, right, forward, backward) and the orientation it is rotating (up, down, left,
493 right, clockwise, counterclockwise), and determine whether the movement is unambiguous enough
494 by checking the largest z-score in translation and rotation. We then combine the movement as well
495 as the keywords extracted to form language primitive commands.

496 **C.2 Target Object**

497 We identify the target object using a prompt heuristic to be fed into GPT3.5-Turbo [9] by taking
498 advantage of the fact that BridgeData consists of mainly object manipulation data. We extract two
499 keywords: the object to be manipulated and the destination of the object, based on the fact that much
500 of BridgeData is focused on object manipulation. The precise prompt can be found at Appendix E

501 **C.3 Data Filtering**

502 We filter low-level instruction on two occasions: when the movement itself is ambiguous and when
503 the language model gives inconsistent results. We check the former by looking up the norm of
504 the translation and the norm of rotation, and we check the latter by using regular expression to see
505 if the result was against the desired format and manually filtering out some common keywords of
506 inadmissible GPT query. On the former occasion, we use an empty string as the low level instruction,
507 and on the second occasion, we use only proprioceptive information for low-level instruction.

508 **D Ablation Details**

509 We ablate our experiment in progressive manners, going from full implementation to using only the
510 barebone hierarchical policy network.

- PALO w/o high level instruction: while running PALO, we derive both high and low level instruction sets. However, during inference on robot, we mask out the high level instruction and feed in zero embeddings.
- PALO w/o low level instruction: mask out the low level instruction and replace them with zero embeddings during inference.
- Fixed Time During Optimization: for each trajectory that has corresponding length H_1, H_2, \dots, H_i , we choose fixed $u_i = [\frac{H_i}{k}, \frac{2H_i}{k}, \dots, \frac{(k-1)H_i}{k}]$ during optimization. We implement no u sampling, which reduce PALO into an arg max operation.
- Zero-Shot Plan Generation: instead of sampling 15 plans, we sample only one plan from VLM and examine the behavior of the robot using that specific plan.
- No VLM Guidance: We use only ℓ as our high level instruction, and mask out low level instruction with zero embeddings during inference.

E Prompting Methods

We employ a keyword decomposition prompt in our augmentation method and a planning prompt to generate VLM outputs. They are listed below:

Keyword Decomposition Prompt

User: "You are presented with a text for high level instruction for a robot, and you need to extract keywords in the task description text.
In this instruction, the first keyword is the object being moved, and the second keyword, if applicable, what is the moving taking this to (either another object or a location) within the instruction.
Only return the first and second keyword, and they should be separated by a comma. If the instruction is in another language, write your response in English.
For example, if the text instruction says "Pick up the silver lid on the left to the middle of two burners", return "silver lid, middle of two burners".
Or if the instruction says: "Move the object to the top middle side of the table.", your response should be "object, top middle side of the table".
Or if the instruction says : "Move the red greenish thing on the towel to the right.", return "red greendish thing on the towel, the right".
Try your best to find the two key phrases, but if you can't find the second keyword within the instruction sentence, write "N/A".
For example, if the instruction is "Move the pot lid.", the response should be "pot lid, N/A".
There might be some other description regarding confidence at the end, you are safe to ignore it.\n The specific task description for you to analyze is: \n {instruction} \n "

Planning Prompt

User: Here is an image observed by the robot in a tabletop robot manipulation environment. The gripper situated at the top of the center of table and perpendicular to it.
Now plan for the the list of subtasks and skills the robot needs to perform in order to {instrs}.

Each step in the plan can be selected from the available skills below:

*movement direction:
 *forward. This skill moves the robot gripper away from the camera by a small distance.
 *backward. This skill moves the robot gripper towards the camera by a small distance.

567 *left. This skill moves the robot gripper to the left of the
568 image by a small distance.
569 *right. This skill moves the robot gripper to the right of the
570 image by a small distance.
571 *up. This skill moves the robot gripper upward until a safe
572 height.
573 *down. This skill moves the robot gripper downward to the
574 table surface.
575
576 *rotation direction:
577 *left. This skill tilts the gripper to an angle to the left.
578 *right. This skill tilts the gripper to an angle to the right.
579 *down. This skill tilts the gripper to an angle facing up.
580 *up. This skill tilts the gripper to an angle facing down.
581 *clockwise. This skill rotates the gripper and the object it
582 is holding clockwise.
583 *counterclockwise. This skill rotates the gripper and the
584 object it is holding counterclockwise.
585
586 *gripper movement:
587 *close the gripper. This skill controls the robot gripper to
588 close to grasp an object.
589 *open the gripper. This skill controls the robot gripper to
590 open and release the object in hand.
591
592 You may choose between using one of movement direction, rotation
593 direction, or gripper movement.
594 If you were to choose to use movement direction, you may use one
595 or two directions and include a target object, and you should
596 format it like this:
597 "move the gripper x towards z" or "move the gripper x and y
598 towards z" where x and y are the directions and z is the
599 target object.
600 You also must start your command with "move the gripper".
601 Therefore, instead of saying something like "down" or "up",
602 you should phrase it like "move the gripper down" and "move
603 the gripper up". Make sure to include at least one direction
604 in your command since otherwise this command format won't make
605 sense.
606
607 If you were to choose to use gripper movement, you should format
608 the command as "close the gripper to pick up x" or "open the
609 gripper to release x", where x is the target object.
610 You may discard the target object if necessary. In that case use "
611 close the gripper" or "open the gripper".
612 If you think the gripper is close to the target object, then you
613 must choose to use gripper movement to grasp the target object
614 to maintain efficiency.
615
616 If you were to choose gripper rotation, you should format the
617 command as "rotate the gripper x", where x is the target
618 rotation direction. You need to make sure that in pouring
619 tasks, the opening of the container is aligned with the pot.
620 For example, if the object is aligned vertically but you want it
621 to align it horizontally, then you should call "rotate the
622 gripper counterclockwise". If you want to tilt the object in
623 the gripper to pour it, you should call "rotate the gripper
624 left"
625
626 Pay close attention to these factors:
627 *Which task are you doing.
628 *Whether the gripper is closed.
629 *Whether the gripper is holding the target object.
630 *How far the two target objects are. If they are across the table,
631 then duplicate the commands with a copy of it.

632 *Where the gripper is. After the end of each subtask, it is
633 reasonable to assume that the gripper will not be at where it
634 originally was in the image, but somewhere close to the last
635 target object.

636

637 Especially pay attention to the actual direction between the
638 gripper and the target object. Remember that the robot’s angle
639 is roughly the same as the camera’s angle.

640 To determine whether the gripper should move forward or backward,
641 look into the edge of the table. If the target object is
642 closer to the edge of the table that is near the top of the
643 image, you should move forward, and if it is closer to the
644 edge that is near the bottom of the image, you should move
645 backward.

646 At the end of each subtask, you need to use the skill "move the
647 gripper back to neutral. This will move the gripper back to
648 the original position of the image after completing the task.

649

650 Start by looking at what objects are in the image, and then plan
651 with the direction of the objects in mind. The tasks should be
652 completed sequentially, therefore you need to consider the
653 position of the gripper after each task before planning the
654 next task.

655 You should return a json dictionary with the following fields:

656 - subtask: this should be the key of the dictionary. It should
657 contain the only the verbal description of the subtask the
658 robot needs to perform sequentially in order to finish the
659 task, and they should be ordered in the same way the task is
660 completed.

661 - list of skills: this should be the value of the dictionary. It
662 should be a list of skills the robot needs to perform in order
663 to finish the corresponding subtask.

664 Be concise, and do not return any other comments other than the
665 dictionary mentioned above. Do not put "subtask: " or "list of
666 skills: " in the key and value of the dictionary you generate
667 . Remember only the description and list should be returned.

668 **F Execution Breakdown**

669 In this section, we provide more details for PALO during inference.

670 **F.1 Inference Details**

671 During inference, we chunk each low-level instruction into length 8 intervals, switching to the new
672 set of low-level (and high-level, if applicable) after these 8 steps. We chose a fixed interval instead of
673 a dynamically allocated one due to the policy choosing to mostly stay put after finishing the action
674 prescribed by the low-level instruction.

675 **F.2 Success Cases**

676 We show the full breakdowns of success cases here. Fig. 8 and Fig. 9 gives detailed description of
677 the robot’s action primitives generated by PALO during inference.

678 **F.3 Failures Cases**

679 We discuss failure cases in more details here.

680 **F.3.1 Failure During Full Inference**

681 While PALO is robust in generating language primitives that help achieve the task, it does not
682 guarantee a successful execution of the policy as shown in Fig. 10.

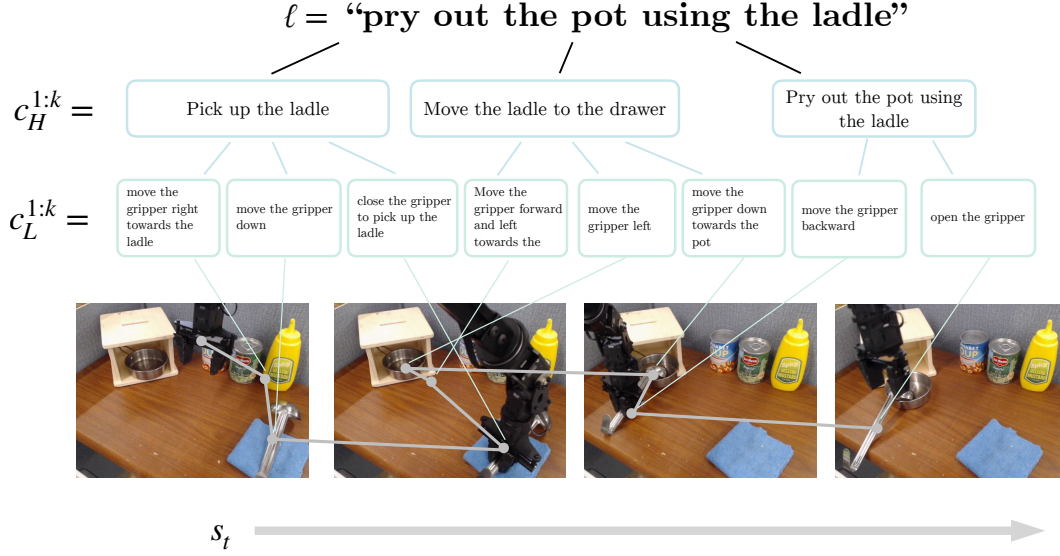


Figure 8: An execution of our method on the task “Pry out the pot using the ladle.”

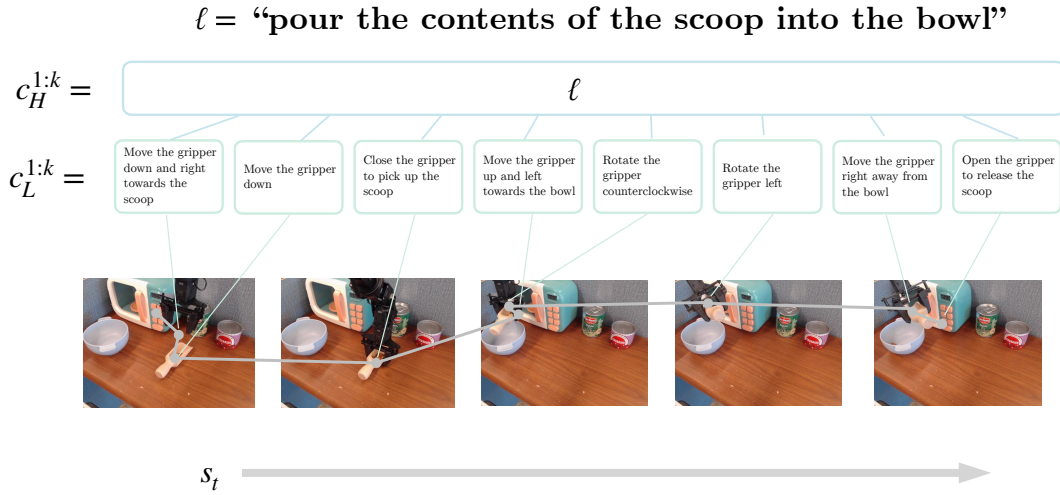


Figure 9: An execution of our method on the task “pour the contents of the scoop into the bowl.”. Note that the high level instruction is ℓ itself, as the best-proposed language decomposition does not create additional subtasks.

683 F.3.2 Failure During Ablation

684 Within ablations of PALO, more critical errors, such as grounding and reasoning errors, may happen.
 685 Fig. 11 demonstrates a case of grounding failure when c_H is masked out.

686 G Evaluation Results

687 We present detailed results of our method across four tasks in the studied scenes in Table 2. We also
 688 present ablation results in Table 3. We evaluate each entry of the result for 10 trials, shifting the
 689 starting location of both target and background objects randomly.

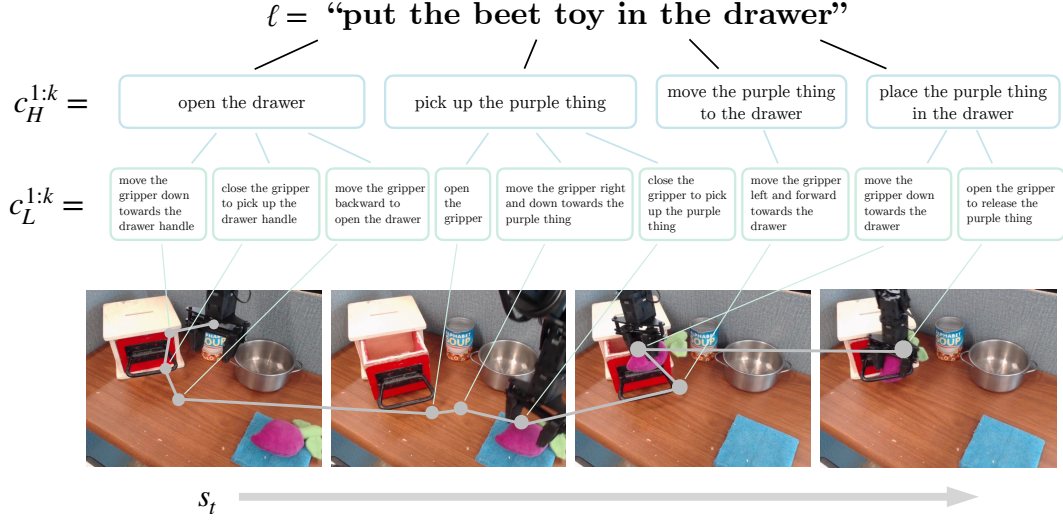


Figure 10: Failure in execution: while the robot completed every subtask correctly up until the last subtask, it did not achieve it due to errors within the policy.

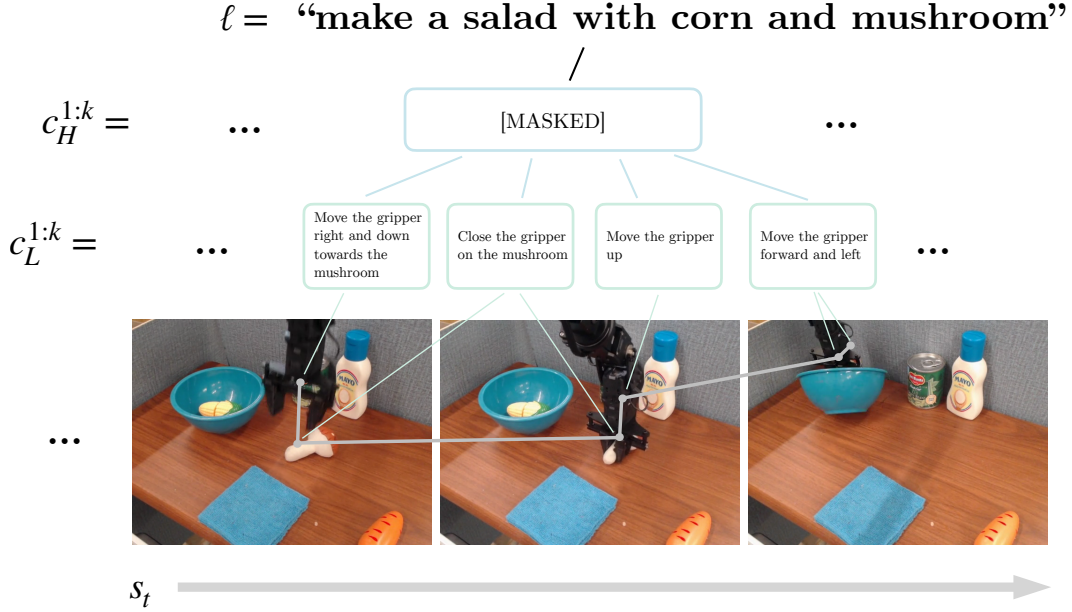


Figure 11: In this instance, we mask out the high level instructions, and the policy is only conditioned on the low-level instructions. We see that the low-level instruction “move the gripper forward and left.” causes the robot to overshoot its trajectory and causes failure in execution.

690 H Code

691 We make our code publicly available at [https://anonymous.4open.science/r/](https://anonymous.4open.science/r/palo-robot-0FCF)
 692 [palo-robot-0FCF](https://anonymous.4open.science/r/palo-robot-0FCF).

Table 2: Method Comparisons

Scene	Task	PALO (Ours)	RT-2-X	FT-Octo	Octo	GRIF	LCBC
Drawer	put in pry away	0.7	0.0	0.0	0.2	0.1	0.1
		0.6	0.2	0.2	0.1	0.0	0.0
Bowl	salad pour scoop	0.7	0.5	0.0	0.3	0.4	0.0
		0.5	0.1	0.2	0.3	0.0	0.0
Sweep	mints skittles	0.7	0.3	0.1	0.2	0.0	0.0
		0.8	0.4	0.0	0.4	0.3	0.2
Rotation	marker spoon	0.9	0.4	0.0	0.1	0.3	0.0
		0.8	0.2	0.1	0.1	0.1	0.0
Average		0.713	0.263	0.1	0.213	0.15	0.08

Table 3: Ablations

Scene	Task	Ours	No c_H	No c_L	Fixed Times	Zero-shot	No VLM
Drawer	put in pry open	0.7	0.2	0.4	0.4	0.3	0.0
		0.6	0.4	0.2	0.1	0.4	0.1
Bowl	salad pour scoop	0.7	0.4	0.5	0.4	0.2	0.0
		0.5	0.1	0.4	0.4	0.2	0.0
Sweep	mints skittles	0.7	0.5	0.3	0.5	0.0	0.0
		0.8	0.7	0.2	0.5	0.4	0.2
Rotation	marker spoon	0.9	0.6	0.3	0.3	0.1	0.3
		0.8	0.6	0.1	0.2	0.3	0.2