# Supplementary Material for "PolarNet: 3D Point Clouds for Language-Guided Robotic Manipulation"

**Anonymous Author(s)**
Affiliation
Address
email

In the supplementary material, we first illustrate all the evaluated RLBench tasks in Sec 1. Then we describe more details of the models in Sec 2. Finally, in Sec 3, we analyze success and failures cases of our model. The supplementary video provides more testing examples.

## 1 RLBench Tasks

We consider RLBench tasks for three different learning setups: single-task single-variation, multi-task single-variation and multi-task multi-variation. Table 1 lists the task names with the corresponding variation type, the number of variations and examples of instructions.

**Single-task and multi-task single-variation details.** Following [1], we select the 10 tasks depicted in Figure 1. We use the original RLBench code[1] to collect training data and run evaluation.

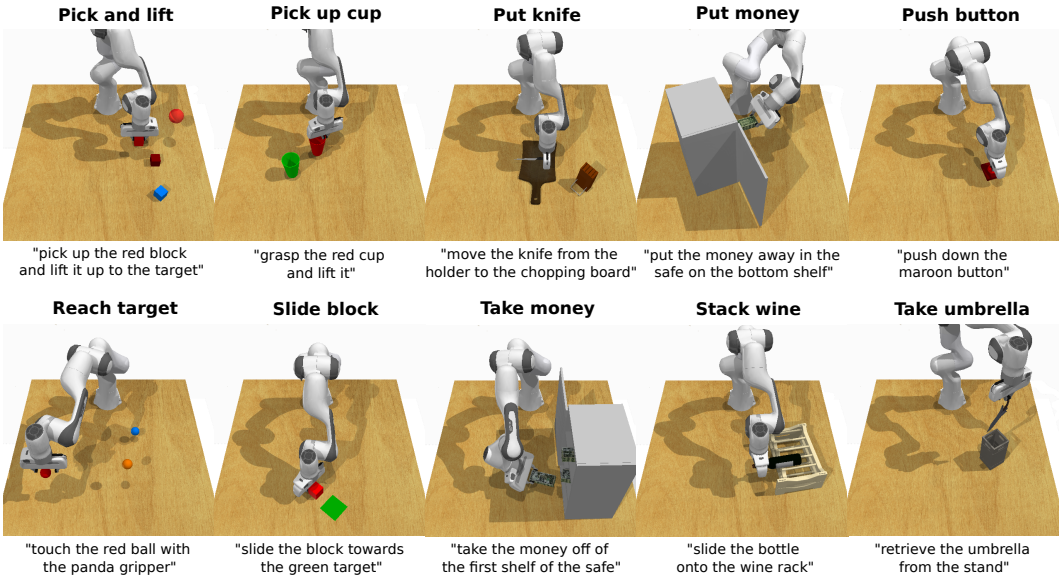


**Figure 1:** 10 tasks with corresponding instructions in single-task single-variation and multi-task single-variation settings.

**Multi-task multi-variation tasks details.** To have a fair comparison with state-of-the-art methods in the multi-task multi-variation setting, we use the setup proposed in PerAct [2] containing 18 tasks with 249 unique task variations. We use the RLBench codebase[2] from [2] which modified

[1] https://github.com/stepjam/RLBench
[2] https://github.dev/MohitShridhar/RLBench/blob/peract/

Submitted to the 7th Conference on Robot Learning (CoRL 2023). Do not distribute.

**Table 1: Set of language-guided RLBench tasks.**

| Task | Var. type | # vars. | Example of language template |
|---|---|---|---|
| *10 tasks used in single-task single-variation and multi-task single-variation settings* | | | |
| pick and lift | color | 20 | ”pick up the __ block and lift it up to the target” |
| pick up cup | color | 20 | ”grasp the __ cup and lift it” |
| put knife | none | 1 | ”put the knife on the chopping board” |
| put money | placement | 3 | ”put the money away in the safe on the __ shelf” |
| push button | color | 18 | ”push down the __ button” |
| reach target | color | 20 | ”touch the __ ball with the panda gripper” |
| slide block | none | 1 | ”slide the block to target” |
| stack wine | none | 1 | ”slide the bottle onto the wine rack” |
| take money | placement | 3 | ”take the money out of the __ shelf and place it on the table” |
| take umbrella | none | 1 | ”take umbrella out of umbrella stand” |
| *18 tasks with 249 variations used in the multi-task multi-variation setting* | | | |
| open drawer | placement | 3 | “open the __ drawer” |
| slide block | color | 4 | “slide the block to __ target” |
| sweep to dustpan | size | 2 | “sweep dirt to the __ dustpan” |
| meat off grill | category | 2 | “take the __ off the grill” |
| turn tap | placement | 2 | “turn __ tap” |
| put in drawer | placement | 3 | “put the item in the __ drawer” |
| close jar | color | 20 | “close the __ jar” |
| drag stick | color | 20 | “use the stick to drag the cube onto the __ target” |
| stack blocks | color, count | 60 | “stack __ __ blocks” |
| screw bulb | color | 20 | “screw in the __ light bulb” |
| put in safe | placement | 3 | “put the money away in the safe on the __ shelf” |
| place wine | placement | 3 | “stack the wine bottle to the __ of the rack” |
| put in cupboard | category | 9 | “put the __ in the cupboard” |
| sort shape | shape | 5 | “put the __ in the shape sorter” |
| push buttons | color | 50 | “push the __ button, [then the __ button]” |
| insert peg | color | 20 | “put the ring on the __ spoke” |
| stack cups | color | 20 | “stack the other cups on top of the __ cup” |
| place cups | count | 3 | “place __ cups on the cup holder” |

some tasks to have more variations. To match the training and evaluation setup from [2] we use the datasets provided in PerAct code repository[3].

## 2 Implementation Details

**Keysteps actions.** For robotic control, we use macro steps [3] – key turning points in action trajectories where the gripper changes its state (open/close) or velocities of joints are close to zero. In this way, the sequence length of an episode is significantly reduced from hundreds of small steps to typically less than 10 macro steps.

**Point removal in point cloud preprocessing.** In Figure 2, we show the raw point cloud generated from multi-view cameras, the point cloud with background and table removal. As the background such as wall and floor are geometrically far away from the robot workspace and the table is in a fixed height, we could simply pre-define a 3D bounding box to select relevant points from the raw point cloud. This is more efficient than performing object segmentation to obtain points of objects.

**Hiveformer [4] re-implementation.** We implemented Hiveformer [4] by ourselves and achieve similar performance as they reported in the paper on the multi-task single-variation setting. To use Hiveformer on the challenging multi-task setups, we enlarge the model capacity by doubling

[3] https://github.com/peract/peract

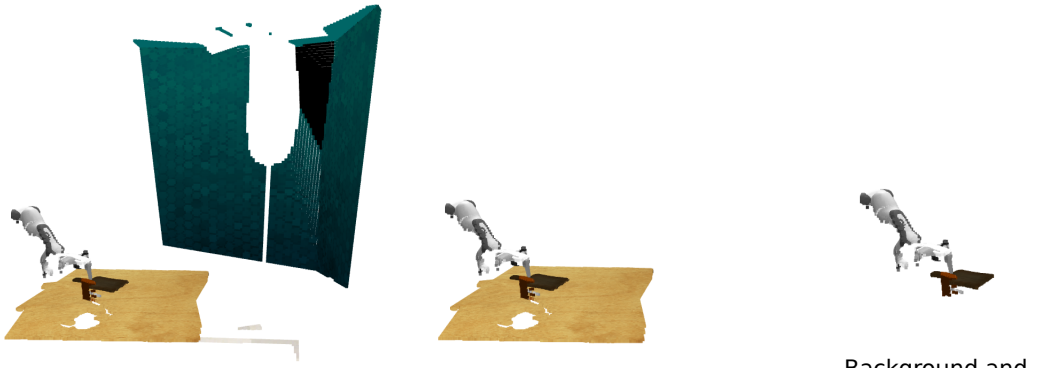


**Figure 2: Illustration of point cloud processing.** We represent the raw point cloud, point cloud with background removal and point cloud with background and table removal for put knife task.

the hidden size and train longer (250K iterations) until the model converges, which leads to better performance than using the hyper-parameters as in the original paper.

## 3 Analysis of Qualitative Results

**Success cases.** We show successful cases for multi-variation multi-task setting in Figure 3. In drag stick task, the policy is able to use a stick to drag the cube to the colored target given by the language instruction. We observed that policy can adapt to the cube trajectory drift to complete the task. Policy is also able to generalize to placement variation such as place wine task. We provide more complex and additional success cases on the supplementary video.

**Failure cases.** In Figure 4 we show different failure cases of our PolarNet. We can observe how the policy fails to insert the ring on the blue spoke due to imprecise position prediction in insert peg task. This task requires fine-grained manipulation which is difficult to have when controlling the robot with keysteps. Place cups is a task that can be solved by placing the cups on the tree in any order. Due to the multimodality nature of this task, we observe that the robot fails to grasp a cup going in between two cups. Finally, in some tasks such as sort shape, the motion planner fails causing the gripper to collide with other objects in the scene failing to complete the task.

## References

[1] S. Liu, S. James, A. J. Davison, and E. Johns. Auto-Lambda: Disentangling dynamic task relationships. *TMLR*, 2022.

[2] M. Shridhar, L. Manuelli, and D. Fox. Perceiver-actor: A multi-task transformer for robotic manipulation. In *CoRL*, 2022.

[3] S. James and A. J. Davison. Q-Attention: Enabling efficient learning for vision-based robotic manipulation. *RA-L*, 2022.

[4] P.-L. Guhur, S. Chen, R. Garcia-Pinel, M. Tapaswi, I. Laptev, and C. Schmid. Instruction-driven history-aware policies for robotic manipulations. In *CoRL*, 2022.

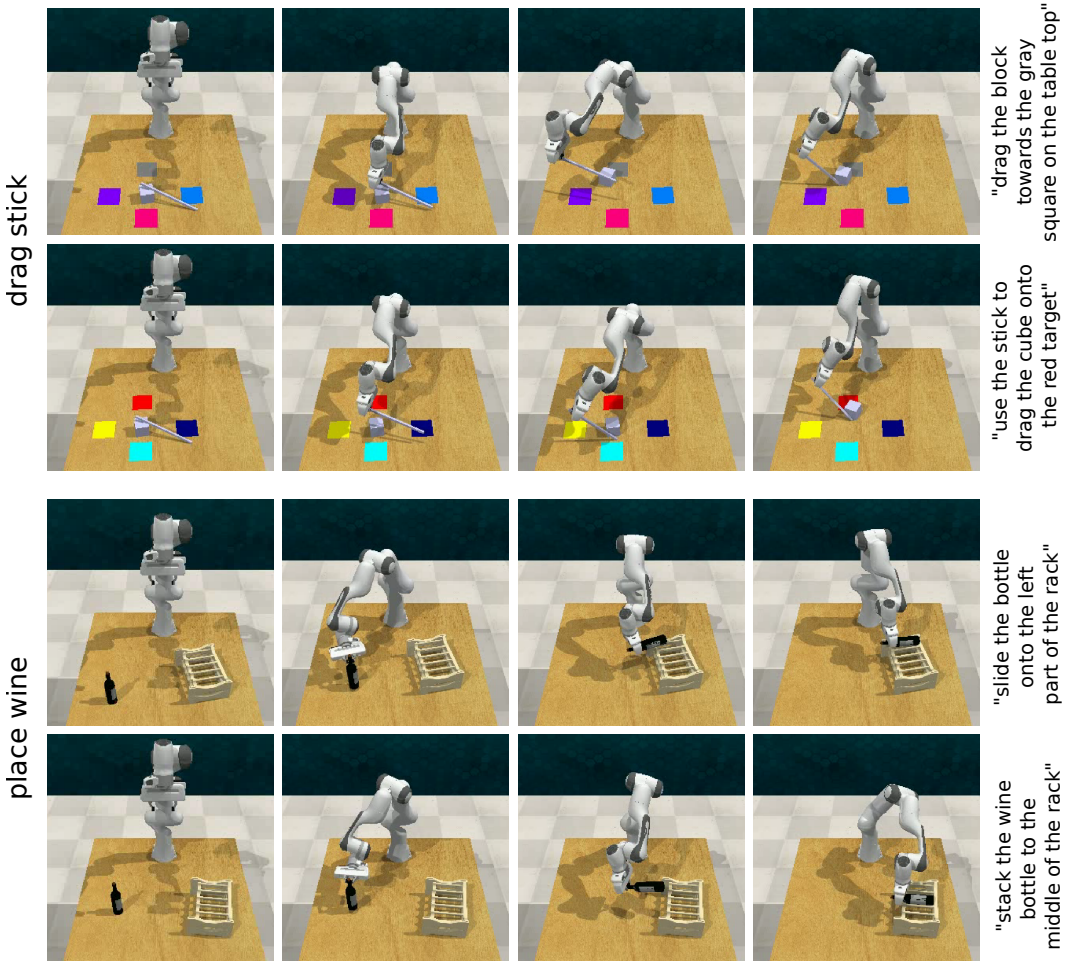


**Figure 3: PolarNet success cases on multi-task multi-variation setting.** We show policy running examples of the drag stick and place wine tasks.

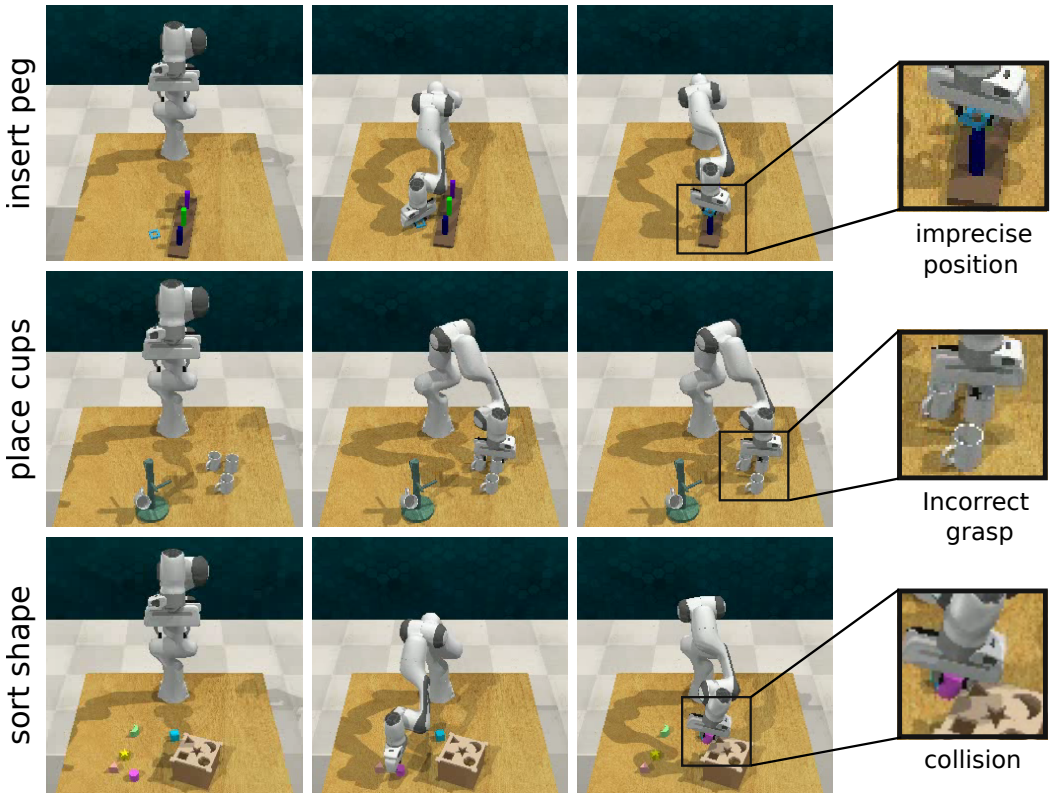


**Figure 4: PolarNet failure cases.** We illustrate failure cases for insert peg, place cups and sort shape.