KITE: Keypoints + Instructions To Execution Supplementary Material

In this section, we outline additional details regarding the implementation of KITE, the real-world environments studied, and qualitative results of all methods. Please refer to our website to see the task diversity we evaluate on and for additional results and videos of KITE performing real-world semantic manipulation.

A Implementation Details

A.1 KITE

As discussed in Section 3.4, KITE trains a PointNet++ model to output all K relative waypoints and a one-hot waypoint index for each point in the point cloud. The auxiliary one-hot classification output layer classifies *which* waypoint each point in the point cloud is most relevant for manipuation (nearest to). In practice, many skills like `pick` or `place` can be parameterized by just $K = 1$ waypoint (where to grasp, where to place). In this case, the 1-hot classification output is reduced to binary classification of which points are near the graspable object or target location, respectively. For more general skills parameterized by K waypoints, we can supervise the K-th end-effector pose predictions per-point by taking the loss of the predicted and ground truth gripper pose for that point compared to ground truth. This loss is provided in the main text in Eq. (2).

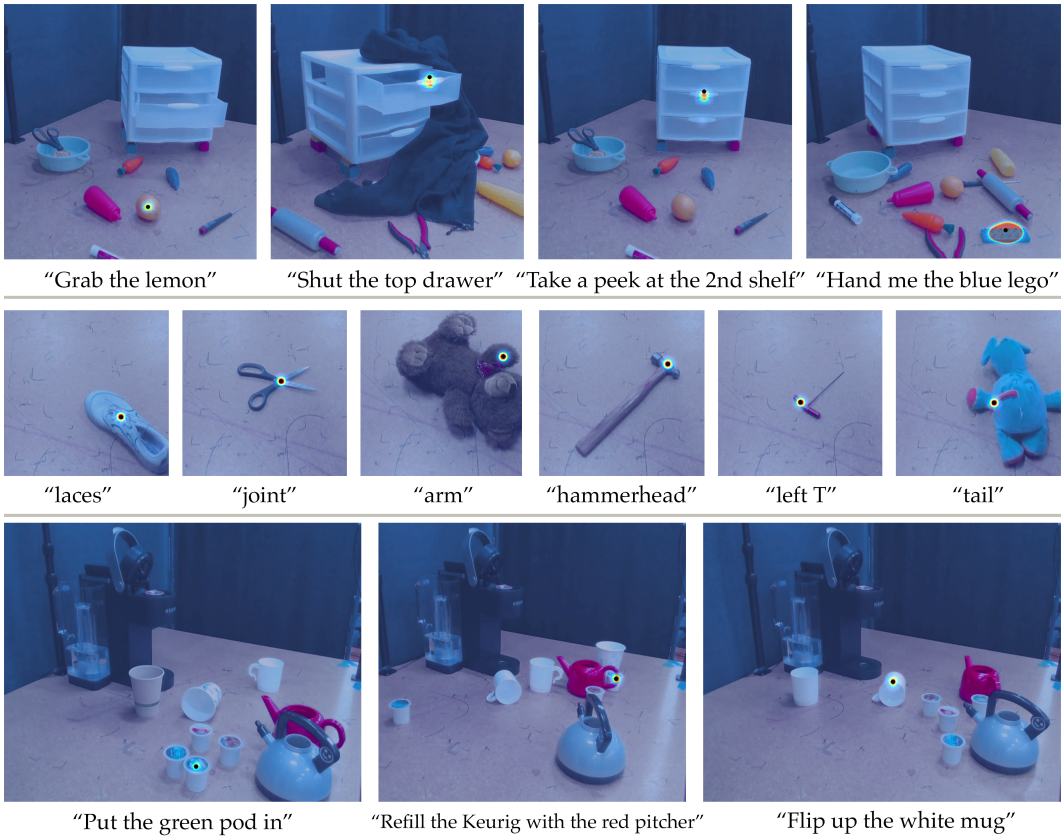

Figure 4: **KITE Grounding Predictions:** KITE's grounding model is able to accurately predict keypoints for both scene semantic instructions (e.g., "grab the lemon" and "put the green pod in") and object semantic instructions (e.g., "shut the top drawer" and "take a peek at the 2nd shelf".

To artificially scale the data KITE's grounding module is trained on, we apply various random colorspace and affine transformations to the dataset collected with all skills to 8X the overall data size before training. We train the grounding module and each skill policy using the Adam optimizer with learning rate 0.0001, which takes 3 hours and 1 hour on an NVIDIA GeForce GTX 1070 GPU, respectively.

A.2 PerAct

For each evaluation environment, we consolidate each of the skill datasets used to train KITE into one multi-task dataset with which to train PerAct. The input to PerAct is a 75^3 voxel grid (although the original PerAct implementation used a 100^3 voxel resolution, we adjust our workspace bounds accordingly to retain the same voxel resolution). We represent waypoints as 1-hot encodings in this voxel grid, end-effector orientations as a discrete prediction over 5 bins each for yaw, pitch, and roll, and the gripper open/closed state as a binary indicator variable as in [11]. For each environment, we train PerAct for 7200 iterations.

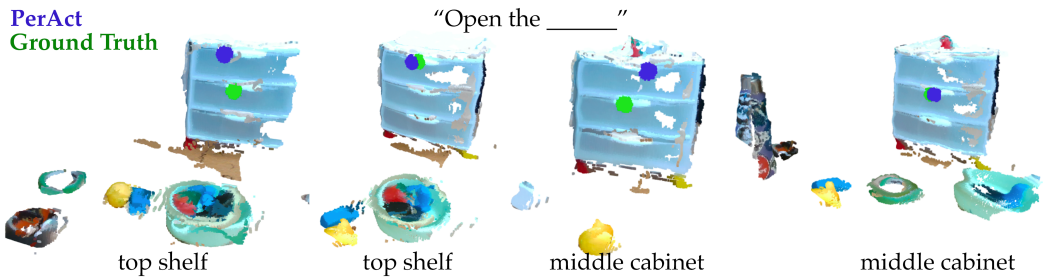

Figure 5: **PerAct Predictions:** We visualize PerAct predictions on the task of opening a cabinet with multiple drawers. Although PerAct exhibits some reasonable predictions (last column), it struggles with localizing the correct handle (1st, 3rd columns). Even when localizing the correct handle (2nd column), the slight imprecision of the predict vs. ground truth action can lead to downstream manipulation failure.

A.3 RobotMoo

We note that the original implementation of RobotMoo leveraged the RT-1 [13] skill learning framework. This set of skills were trained with months of data collection, amassing thousands of trajectories for 16 object categories, and RobotMoo further extended these policies to 90 diverse object categories. As this is not reproducible in our setting, we implement RobotMoo by using KITE's library of skills, but conditioning them on VLM predictions instead of our keypoints. Specifically, while the original RobotMoo implementation used OwLViT, we use the more recent state-of-the-art open vocabulary object detectors Grounding DINO [8] and Segment Anything [52] in conjunction to obtain segmentation masks for objects referenced in an input instruction. We take the center pixel coordinate of these segmentation masks as input to our acting module rather the output of Q_{ground}.

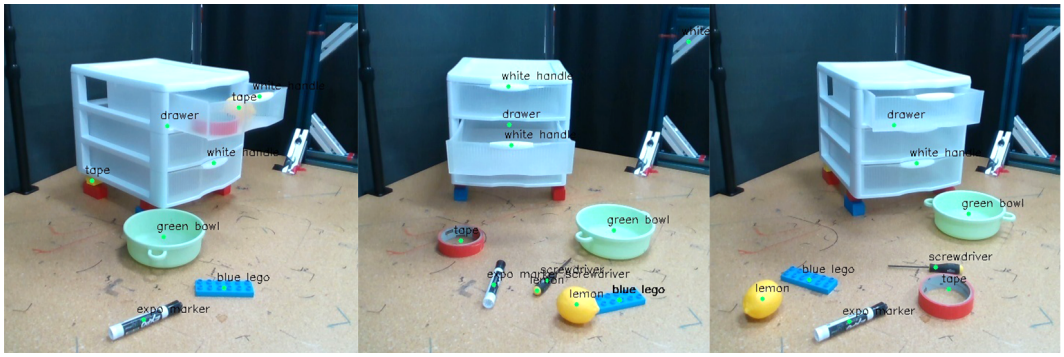

Figure 6: **RobotMoo Predictions:** We visualize the predictions for RobotMoo's perception stack on tabletop instruction following images. Although RobotMoo exhibits decent scene awareness and an ability to localize different object instances, it struggles with object semantics. Specifically, RobotMoo struggles to localize all the different drawers in the scene, let alone distinguish amongst the *top* vs. *middle* vs. *bottom* handles.

B Real-World Experimental Details

B.1 Task Variations

For each real environment used in evaluation, we stress-test all methods across task variations ranging from diversity in the input language instructions to amount of clutter and distractor objects. We summarize each axis of variation for long-horizon tabletop instruction following (Table 4), semantic grasping (Table 5), and coffee-making (Table 6) below.

B.2 Primitive Instantiations

In this section, we describe the instantiation of our library of skills for each real-world environment:

Tier	Skill	Variations	
		Language	Scene
1	`open`	'Open the [top/middle/bottom] cabinet'	randomized position of cabinet
	`close`	'Close the [top/middle/bottom] drawer'	randomized position of cabinet
	`pick`	'Pick up the [lemon/screwdriver/lego/bowl/expo marker]	randomized object positions
	`place`	'Put the [...] [away, in the [...] drawer, in the bowl]'	randomized object positions
1	`open`	'Open the [top/middle/bottom] cabinet'	randomized position + distractor objects (clothes strewn)
	`close`	'Close the [top/middle/bottom] drawer'	randomized position + distractor objects (clothes strewn)
	`pick`	'Pick up the [lemon/screwdriver/lego/bowl/expo marker, eggplant, carrot, corn, lime, scissors, ketchup, coffee pod]	randomized position + scene clutter
	`place`	'Put the [...] [away, in the [...] drawer, in the bowl]'	randomized object positions + scene clutter
3	`open`	'Yank open the top drawer' 'Give the 2nd drawer a tug' 'Take a peek at the 3rd drawer pls' 'Can you check the top drawer?'	randomized position + distractor objects (clothes strewn)
	`close`	'Close it' 'Give it a push' 'Let's shut the drawer' 'Go ahead and close the top drawer' 'Close any open drawer' 'The one 3rd from the bottom needs to be shut'	distractor objects (clothes strewn)
	`pick`	'Grab me the [...]' 'Do me a favor and get me the [...]' 'Can you pass me the [...]?' 'Get the [...]' 'Locate the [...]' 'Could you hand me the [...] please'	randomized object positions + scene clutter
	`place`	'Grab the [...] and put it [...]' 'Take the [...] and place it [...]' 'Pick up the [...] and put it [...]' 'Fetch the [...] and drop it [...]' 'Plop the [...] into the bowl'	randomized object positions + scene clutter

Table 4: **Tabletop Instruction Following Environment Variations**

Category	Object	Language Variations
Rigid Tools	hammer	middle, end, tooltip, hammerhead, metal, wooden handle, center, tip
	T-tool	left side, left T, right side, right T, bottom handle, top handle
Deformable Objects	shoe	heel, toe, back, front, shoelaces, laces, lace-up area
	stuffed animal	head, nose, ear, tail, belly, foot, arm, leg, tummy, elephant trunk
Articulated Items	pliers	joint, left handle, right handle, top handle, bottom handle
	clamp	joint, left handle, right handle, top handle, bottom handle
	scissors	joint, left hole, right hole, smaller hold, bigger hold, larger hold
	marker + twist-off cap	cap, expo label, center, label, end

Table 5: **Semantic Grasping Environment Variations**

Tabletop Instruction Following: We parameterize each skill in the tabletop manipulation setting via a single waypoint $\kappa = (x, y, z, \psi, \theta, \phi)$ specifying the primary point of interaction.

- `open`: With its gripper open, the robot approaches 5cm. away from a closed drawer handle at position (x, y, z) with orientation $\psi, \theta, \phi)$. Next, the robot moves to (x, y, z) in the same orientation and closes the gripper to grasp the cabinet handle. Finally, it executes a linear pull, keeping the orientation fixed, for 5cm before releasing the handle.
- `close`: The robot approaches 5cm. away from an opened drawer handle at position (x, y, z) with orientation $\psi, \theta, \phi)$, then executes a linear push towards (x, y, z) close the drawer.
- `pick`: The robot approaches the object located at (x, y, z), closes its gripper, and lifts 5cm.
- `place`: While holding an object grasped with the `pick` primitive, the robot moves 5cm above the desired place location (x, y, z) with orientation (ψ, θ, ϕ) and opens the gripper to its maximum width, releasing the object.

Coffee-Making In the coffee-masking tasks, we implement a library of 4 skills which test KITE's ability to handle precise or dynamic movements. Since `pour_cup`, `refill_keurig`, and `load_pod`

all involve grasping, we finetune the `pick` skill from tabletop instruction-following with 50 demonstrations across pitchers and coffee pods, respectively. Then, we can parameterize each skill with a single waypoint $\kappa = (x, y, z, \psi, \theta, \phi)$ as follows:

- `reorient_mug`: The robot attempts to grasp a mug, initially oriented sideways, with pose κ before resetting to a canonical upright (untilted) end-effector pose.
- `pour_cup` / `refill_keurig`: After grasping a pitcher, the robot moves to position (x, y, z) denoting the position of the vessel to be poured into (cup or refill compartment of Keurig). Starting from an untilted end-effector pose, the robot gradually rotates at a constant velocity to (ψ, θ, ϕ), denoting the final pour orientation.
- `load_pod`: After grasping a coffee pod with the `pick` primitive, the robot moves 2 cm. above (x, y, z), the sensed position of the K-cup slot with orientation (ψ, θ, ϕ). Next, the robot releases its grasp to drop the pod into the compartment. As this task requires high precision, it is often the case that after releasing the pod, it is not completely inserted or properly aligned. Thus, the `load_pod` primitive moves downward an additional 2cm in attempt to push the pod into place. We note that we do not evaluate this skill with real liquids for safety reasons, but measure success in terms of visual alignment between the pitcher and vessel.

Skill	Language	Scene
`reorient_mug`	'Flip the mug right-side up' 'Put the mug upright' 'Grab the mug and put it it right-side-up' 'Can you place the mug right side up?' 'Get the mug that's laying flat and flip it upright'	randomized mug position, roll $(-\pi/2, \pi/2)$
`pour_cup`	'Fill up the mug that's right-side up' 'Pour me a glass' 'Pour the red pitcher into the mug' 'Grab the silver-handle pitcher and fill up the brown cup' 'Refill the Dixie cup with the red pitcher'	randomized pitcher (red / gray) + cups (compostable, Dixie, mug)
`refill_keurig`	'Refill the espresso machine' 'Grab the red/silver pitcher and 'fill up the water compartment'	randomized pitcher (red / gray) + randomized Keurig pose
`load_pod`	'Load the blue K-cup' 'Can you put the red pod in?' 'Insert the green pod' 'Start a brew with the brown pod'	randomized coffee pod (red/blue/green/brown) + randomized Keurig

Table 6: **Coffee-Making Variations**

B.3 LLM Prompting

Skill Label Inference: In this section, we briefly outline how KITE retrieves the skill label l_t for input instruction i_t via LLMs. Below, we provide a sample prompt which we feed as input to `text-davinci-003` to obtain l_t in tabletop instruction following setting.

Listing 1: LLM Prompting for Skill Label Inference

```
i_t = input("Enter instruction:")
"""
Input: "Pick up the lemon"
Output: ["pick"]

Input: "Put the screwdriver away"
Output: ["pick", place"]

Input: "Pls grab me the screwdriver and put it away"
Output: ["pick", "place"]

Input: "Grab the green bowl"
Output: ["grasp"]

Input: "Put the lemon in the bowl"
Output: ["pick", "place"]

```

```
Input: "Open the top drawer"
Output: ["open"]

Input: "Pls shut the drawer"
Output: ["close"]

Input: "put the expo marker away"
Output: ["pick", "place"]

Input: "put the Blue lego in the cabinet"
Output: ["pick", "place"]

Input: '%s'
Output:
"""%i_t
```