

KITE: Keypoint-Conditioned Policies for Semantic Manipulation

Anonymous Author(s)

Affiliation

Address

email

1 **Abstract:** While natural language offers a convenient shared interface for humans
2 and robots, enabling robots to interpret and follow language commands remains a
3 longstanding challenge in manipulation. A crucial step to realizing a performant
4 instruction-following robot is achieving *semantic manipulation* — where a robot
5 interprets language at different specificities, from high-level instructions like ‘Pick
6 up the stuffed animal’ to more detailed inputs like ‘Grab the left ear of the elephant.’
7 To tackle this, we propose KITE: Keypoints + Instructions to Execution, a two-
8 step framework for semantic manipulation which attends to both *scene* semantics
9 (distinguishing between different objects in a visual scene) and *object* semantics
10 (precisely localizing different parts within an object instance). KITE first grounds
11 an input instruction in a visual scene through 2D image keypoints, providing a
12 highly accurate object-centric bias for downstream action inference. Provided an
13 RGB-D scene observation, KITE then executes a learned keypoint-conditioned skill
14 to carry out the instruction. The combined precision of keypoints and parameterized
15 skills enables fine-grained manipulation with generalization to scene and object
16 variations. Empirically, we demonstrate KITE in 3 real-world environments: long-
17 horizon 6-DoF tabletop manipulation, semantic grasping, and a high-precision
18 coffee-making task. In these settings, KITE achieves a 75%, 70%, and 71% overall
19 success rate for instruction-following, respectively. KITE outperforms frameworks
20 that opt for pre-trained visual language models over keypoint-based grounding, or
21 omit skills in favor of end-to-end visuomotor control, all while being trained from
22 fewer or comparable amounts of demonstrations. Supplementary material, datasets,
23 code, and videos can be found on our [website](#).¹

24 **Keywords:** Semantic Manipulation, Language Grounding, Keypoint Perception

25 1 Introduction

26 Language has the potential to serve as a powerful communication channel between humans and robots
27 in homes, workplaces, and industrial settings. However, two primary challenges prevent today’s
28 robots from handling free-form language inputs. The first is enabling a robot to reason over *what* to
29 manipulate. Instruction-following requires not only recognizing task-relevant objects from a visual
30 scene, but possibly refining visual search to specific features on a particular object. For instance,
31 telling a robot to “Open the top shelf” vs. “Yank open the bottom shelf” of a cabinet requires not only
32 parsing and resolving any liberties taken with phrasing and localizing the cabinet in the scene (*scene*
33 semantics), but also identifying the exact object feature that matters for the task — in this case the
34 top or bottom handle (*object* semantics). In this work, we refer to instruction-following with scene
35 and object awareness as *semantic manipulation*. Similarly, pick-and-place is a standard manipulation
36 benchmark [1, 2, 3, 4], knowing how to pick up a stuffed animal by the ear versus leg, or a soap bottle
37 by the dispenser versus side requires careful discernment. After identifying what to manipulate, the
38 second challenge is determining *how* the robot can accomplish the desired behavior, i.e., low-level
39 sensorimotor control. In many cases, low-level action execution requires planning in SE(3) with 6
40 degrees-of-freedom (DoF), such as reorienting the gripper sideways to grasp and pull open a drawer.

¹<https://tinyurl.com/kite-manip>

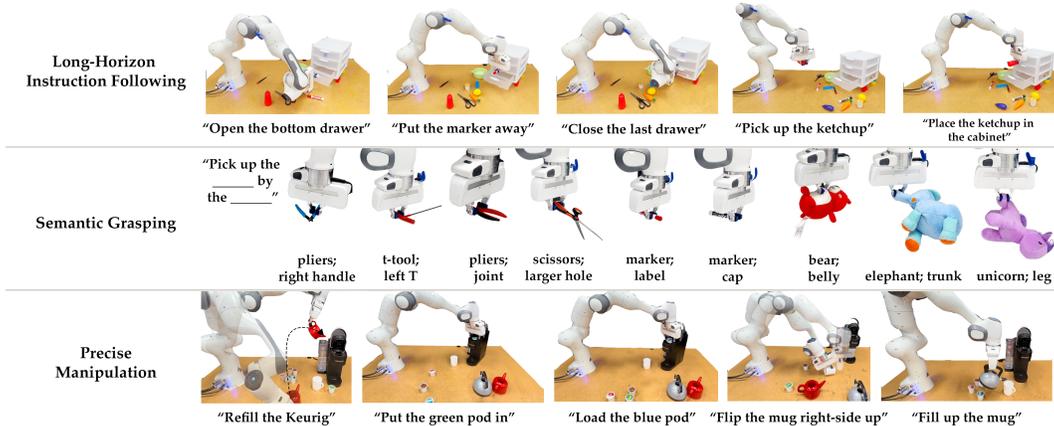


Figure 1: **Real-World Semantic Manipulation Environments:** We visualize our semantic manipulation framework KITE on three real-world environments: long-horizon instruction following, semantic grasping, and coffee-making. Using keypoint-based grounding, KITE contextualizes scene-level semantics (‘Pick up the green/red/blue/brown coffee pod’) as well as object-level semantics (‘Pick up the unicorn by the leg/ear/tail’, ‘Open the cabinet by the top/middle/bottom shelf’) and precisely executes keypoint-conditioned skills.

41 Going beyond grasping, we want robots to assist us in many daily real-world tasks which may require
 42 even finer-grained precision. Making coffee, for example, is a simple daily task for humans, but for a
 43 robot it involves complex steps like reorienting a mug from sideways to upright or carefully inserting
 44 a coffee pod into an espresso machine. Thus to achieve semantic manipulation, robots must extract
 45 scene and object semantics from input instructions and plan precise low-level actions accordingly.

46 Leveraging advances in open-vocabulary object detection [5, 6, 7, 8], prior works in language-based
 47 manipulation determine *what* to manipulate via bounding boxes or keypoints obtained from pre-
 48 trained [9] or fine-tuned vision language models (VLMs) [10]. So far, these works operate at the
 49 level of scene semantics (distinguishing amongst objects) rather than object semantics (identifying
 50 within-object features). In addition, these works do not apply VLMs to any complex manipulation
 51 beyond simple pick-and-place. To address these shortcomings, follow-up works couple the *what*
 52 and *how* subproblems together and learn end-to-end 6-DoF language-conditioned policies from
 53 demonstrations [11, 12, 13]. However, learning high dimensional action spaces from raw sensor inputs
 54 such as images or voxelized scene representations can require excessive amounts of data [12, 13] and
 55 can be difficult in high-precision tasks especially when using discretized actions [11].

56 Between approaches that leverage pre-trained visual representations from off-the-shelf VLMs, and
 57 those that plan directly from pixels or voxels, we lack an intermediate object-centric representation
 58 that can link natural language to scene and object semantics. Prior work has demonstrated that
 59 open-vocabulary VLMs can address scene semantics to some extent by locating different objects with
 60 coarse bounding boxes [9], but these representations are still too granular to precisely locate parts
 61 on objects. A suitable visual representation would be one that can represent both across-object or
 62 within-object features, and is interpretable enough to inform downstream 6-DoF action planning. We
 63 argue that keypoints provide this happy medium by offering a way to precisely pinpoint objects at
 64 the scene-level or even features within an object (*what*) and a way to condition downstream 6-DoF
 65 manipulation (*how*) on a region of interest.

66 In this work, we present **KITE: Keypoints + Instructions To Execution**, a flexible framework for
 67 semantic manipulation. KITE is decoupled into a grounding policy which maps input images and
 68 language commands to task-relevant keypoints, and an acting module which employs keypoint-
 69 conditioned skills to carry out low-level 6-DoF actions. We show that KITE can be trained from just
 70 a few hundred annotated examples for the grounding model, and less than 50 demos per skill for the
 71 acting module, while outperforming and generalizing better than methods that do not make use of
 72 either keypoints or skills. We experimentally evaluate KITE on semantic manipulation across three
 73 challenging real-world scenarios with varying tiers of difficulty: 6-DoF tabletop manipulation, se-
 74 mantic grasping, and coffee-making (Figure 1). Results indicate that KITE demonstrates fine-grained
 75 instruction following, while exhibiting a capacity for long-horizon reasoning and generalization.

76 2 Related Work

77 **Language-Based Manipulation:** Many recent works ground language to manipulation skills as a
78 means for long-horizon instruction following. Several methods learn language conditioned policies
79 end-to-end using imitation learning; however, end-to-end learning can require many demonstrations
80 and can be brittle to novel scenes or objects [12, 11, 14, 15, 16, 17, 18, 19, 10, 20]. To improve sample
81 efficiency, Shridhar et al. [11] predicts robot waypoints rather than low-level actions, conditioned
82 on language and point cloud inputs. Waypoints alone do not specify *how* to go from one waypoint
83 to another, and thus fail to capture dynamic tasks such as peg insertion or pouring motions. Other
84 works take a hierarchical approach that first learn or define a library of language-conditioned skills,
85 and then plan over skills with large language models (LLM) [21, 22, 23, 24, 25, 26]. However, each
86 skill often requires hundreds of demonstrations. In addition, the LLM planner can only reason about
87 the scene at a high level, lacking *visual grounding*. Alternatively, Liang et al. [27] query the LLM to
88 generate code using an API for low-level skills, but predicting continuous parameters of these skills
89 is challenging for an ungrounded LLM limiting this approach to tasks that do not require precision or
90 dexterity. Vision-Language Models (VLM) are often proposed to ground LLM planners. For instance,
91 Stone et al. [28] leverage a pretrained VLM to identify task-relevant objects from language. However,
92 this approach has also been limited to pick and place tasks suggesting that today’s pretrained VLMs
93 struggle with more precise language instructions. In contrast, KITE maps language and vision directly
94 to desired keypoints, enabling precise and semantic manipulation over long horizons.

95 **Skill-Based Manipulation:** Many prior works study multi-task manipulation by defining *skills* to
96 represent sub-tasks, and then composing these skills over long horizons. These skills can either be
97 learned to output each action or parameterized by expert-defined features. In reinforcement learning
98 (RL), hierarchy can be imposed on the policy to learn both skills and composition end-to-end, but
99 these methods can be sample inefficient and rarely integrate well with natural language [29, 30, 31].
100 Other RL works parameterize skills to reduce the action space size for sample efficiency, but these
101 skills are usually rigid and cannot generalize to new settings [32, 33, 34]. Imitation learning (IL) aims
102 to learn skills from demonstrations in a more sample efficient manner than RL, but these skills still
103 fail to generalize to scene perturbations [35, 36]. Furthermore, in both IL and RL, connecting learned
104 skills to precise language in a generalizable fashion is an open challenge. KITE avoids learning skills
105 from scratch and instead defines a library of keypoint-conditioned skills, where the exact parameters
106 of each skill are learned from demonstration. We show that keypoint-conditioned skills are sample
107 efficient to learn and generalizable to new objects, while also easily integrated with precise language.

108 **Keypoints for Manipulation:** Keypoints have emerged in the literature as a more robust skill
109 representation for manipulation [37, 10]. Keypoints are 2D points on images that serve as a natural
110 intermediary between images and low-level behaviors. Several methods use keypoints to force the
111 model to attend to the most important features in the input images [37]. Others predict keypoints
112 and then translate keypoints into 3D points, or directly predict 3D points, to parameterize low level
113 behaviors in a general and visually-grounded fashion [38, 10, 39, 40]. For example, Shridhar et al.
114 [10] parameterize pick and place tasks using keypoints learned with image supervision, showing that
115 this keypoint abstraction generalizes better to new objects. Keypoint action spaces have also helped
116 in deformable object manipulation, for example in the domains of cloth folding, rope untying,
117 and food manipulation [41, 42, 43, 44, 45]. For many of these prior works, keypoints have yet to be
118 integrated with language, and the methods that are linked to language are limited to a small library of
119 primitives, usually focusing only on pick and place scenarios. Our approach defines a much broader
120 library of keypoint-conditioned skills, and integrates keypoints with complex language instructions.

121 3 KITE: Keypoints + Instructions To Execution

122 In this work, our goal is to train an instruction-following agent capable of performing semantic
123 manipulation at both scene and object-level granularity. We accomplish this with KITE, a sample-
124 efficient and generalizable framework operating in two stages: *grounding* language into keypoints
125 and *acting* on those keypoints. In this section, we first formalize the semantic manipulation problem
126 (Section 3.1), discuss data collection (Section 3.2), and then discuss the training procedures for the
127 grounding (Section 3.3) and acting modules (Section 3.4).

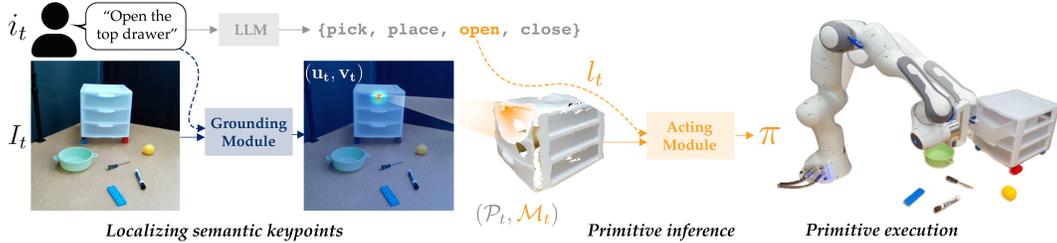


Figure 2: **KITE System Overview:** KITE receives an image observation I_t along with user instruction i_t and grounds these inputs to a 2D semantic keypoint in the image. After inferring which skill type l_t is appropriate from a set of skill labels, KITE takes an RGB-D point cloud observation \mathcal{P}_t , annotated with the deprojected keypoint \mathcal{M}_t , and infers the appropriate waypoint policy π for execution. After executing this action, KITE replans based on a new observation (I_{t+1}, i_{t+1}) and repeats the whole process.

128 3.1 Semantic Manipulation Problem Formulation

129 We aim to tackle instruction-following with scene and object-level semantic awareness using a library
 130 of *skills*. We assume each *skill* can be parameterized by 6-DoF waypoints, and we decouple each
 131 skill into a waypoint *policy* π and *controller* ρ to move between waypoints in a task-specific manner.
 132 Additionally, we assume each skill can be represented by a skill label l , e.g., `pick`, `open`, etc. We
 133 construct a library \mathcal{L} of M specialized skills where $\mathcal{L} = \{l^1 : (\pi^1, \rho^1), \dots, l^M : (\pi^M, \rho^M)\}$ maps
 134 from a skill label to an underlying policy π and controller ρ .

135 We assume access to multiple calibrated cameras that provide RGB and depth. We assume that at
 136 least one “grounding” camera can partially see all relevant objects. An observation $o_t = (I_t, \mathcal{P}_t)$
 137 where $I_t \in \mathbb{R}^{W \times H \times 3}$ is the image from the grounding camera, and $\mathcal{P}_t \in \mathbb{R}^{D \times 6}$ is a multi-view
 138 point cloud. We denote low-level robot actions $a_t = (x, y, z, \psi, \theta, \phi)$ at time t which consist of the
 139 end-effector position, yaw, pitch, and roll. We denote waypoints as κ which is also a 6-DoF pose, but
 140 represents a high-level pose (e.g., grasp pose for `pick`) rather than a low-level action.

141 At time t , given an instruction i_t , we want to know *which* skill to execute by associating i_t to a
 142 corresponding skill label $l_t \in \{l^1, \dots, l^M\}$. Next, we aim to infer a 2D keypoint $[u, v]$ in the current
 143 visual observation o_t which grounds i_t to an associated object or object part. Finally, the chosen
 144 skill $(\pi, \rho) = \mathcal{L}[l_t]$ is executed (Figure 2). For each skill, we want to find a waypoint policy π
 145 that takes as input the visual observation o_t and 2D keypoint $[u_t, v_t]$ and outputs K waypoints:
 146 $\pi : (o_t, [u_t, v_t]) \rightarrow \{\kappa^1, \dots, \kappa^K\}$. Then, the associated controller ρ can output a low-level trajectory
 147 between waypoints $\rho : \{\kappa^1, \dots, \kappa^K\} \rightarrow \tau = \{(o_t, a_t), \dots, (o_{t+T-1}, a_{t+T-1})\}$ (i.e. via linear
 148 interpolation, motion planning, etc.) for the robot to execute. For multi-step manipulation tasks, we
 149 restart the above process at each step with the new observation o_{t+T} and paired language input i_{t+T} .

150 We consider instructions which refer to *scene* semantics, such as specifying desired spatial rearrange-
 151 ments of objects (e.g. “Pick up the lemon”), and *object* semantics, which reference desired object
 152 parts to be manipulated (e.g. “Grab the kangaroo stuffed animal by the tail”). As we do not assume
 153 access to the interaction history, our space of feasible language inputs excludes post-hoc feedback
 154 (“Pick up the *other* marker”) or online-corrections (“No, to the *left!*”), which we leave to future work.
 155 Next we outline how KITE learns to predict keypoints (grounding) and learns each skill π (acting).

156 3.2 Demonstration Collection

157 To learn both the grounding and acting modules, we collect a dataset \mathcal{D}_π consisting of N expert
 158 demonstrations per skill. Each demonstration has an initial observation, a list of K waypoints,
 159 and a language instruction: $\mathcal{D}_\pi = \{(o_n, \{\kappa_n^1 \dots \kappa_n^K\}, i_n) : n \in \{1, \dots, N\}\}$. For instance, for a
 160 `pick` skill, we record the initial image and point cloud, provide an instruction (e.g., ‘Pick up the
 161 lemon’), and then kinesthetically move the robot and record each end-effector waypoint. We use the
 162 calibrated robot-to-camera transformation to automatically project each robot end-effector pose κ_n^j to
 163 2D coordinates $[u_n, v_n]$ in the image plane of the camera used for grounding. For each skill, we train
 164 the acting module from \mathcal{D}_π . Aggregating across all skills yields a dataset of paired images, keypoint
 165 annotations, and language instructions with which to train the grounding module.

166 **3.3 Grounding Module**

167 The grounding module learns to identify 2D keypoints from RGB images that correspond to object
 168 features mentioned in an input instruction. We draw inspiration from recent works which use explicitly
 169 supervised [10, 38] or self-supervised keypoint attention networks [37, 46] to implement a grounding
 170 model Q_{ground} . Specifically, we learn a grounding function $Q_{\text{ground}}(u, v, I_t, i_t)$ representing the
 171 likelihood of keypoint $[u, v]$ given image I_t and paired language instruction i_t . In this work, we
 172 attempt to learn a single-step look-ahead grounding function that takes a language input (e.g. “Put
 173 the lemon into the cabinet”) and outputs the most immediately relevant keypoint (e.g. the pixel for
 174 the lemon if not already grasped, otherwise the pixel for the cabinet drawer to be placed) (see Fig. 2).
 175 Given this grounding function Q_{ground} , we infer the 2D pixel in the image with highest likelihood:

$$[u_t, v_t] = \arg \max_{u, v} Q_{\text{ground}}(u, v, I_t, i_t) \quad (1)$$

176 In practice, we implement Q_{ground} using the two-stream architecture from [10] which fuses pre-
 177 trained CLIP [47] embeddings of visual and textual features in a fully-convolutional network to
 178 output a heatmap of Q_{ground} . The grounding function is trained with a binary cross-entropy loss
 179 between the predicted heatmaps and 2D Gaussian heatmaps centered at the ground-truth pixel.

180 **3.4 Acting Module**

181 Although keypoints can pinpoint both scene and object semantics, they critically lack the 3D geometric
 182 context necessary to recover precise 6-DoF actions for a given task. For instance, the command
 183 “Pick up the bowl” may result in a predicted keypoint located at the bottom of a bowl, where there
 184 is no feasible grasp. The exact 6-DoF actions are also dependent not just on the keypoint, but also
 185 language: “pick the lemon” and “cut the lemon” have similar keypoints but require completely
 186 different actions. We need a way to *refine* a predicted keypoint into candidate 6-DoF actions based on
 187 a desired language command, which we discuss next.

188 **Skill Selection:** Given a free-form language instruction, KITE first leverages the knowledge of LLMs
 189 to determine the appropriate skill label (e.g. $i_t =$ “Put the lemon in the cabinet” should result in
 190 the LLM outputting $\hat{l}_t =$ ‘pick_place’), following prior work [48, 21]. The procedure entails
 191 prompting the LLM, in our case OpenAI’s text-davinci-003 [49], with in-context examples
 192 of instructions and the appropriate skill type (see Appendix B.3 for examples of our prompting
 193 strategy). At test-time, we concatenate the example prompt with instruction i_t and generate skill label
 194 $\hat{l}_t \in \{l^1, \dots, l^M\}$ using the LLM. Then, we obtain the skill, consisting of the waypoint policy π and
 195 controller ρ via lookup in the library: $(\pi, \rho) = \mathcal{L}[\hat{l}_t]$.

196 **Learning Waypoint Policies:** Given the keypoint $[\hat{u}_t, \hat{v}_t]$ predicted by the grounding module and
 197 skill label \hat{l}_t , we need to learn a waypoint policy π to perform the skill. KITE learns π for each skill
 198 from demonstrations of keypoints $\{\kappa^1, \dots, \kappa^K\}$. The waypoint policy π takes a point cloud \mathcal{P}_t and
 199 keypoint $[\hat{u}_t, \hat{v}_t]$ as input, and aims to output K waypoints $\{\kappa^1, \dots, \kappa^K\}$ to execute the chosen skill.

200 In KITE we align both 3D point cloud and a 2D keypoint representations by “annotating” \mathcal{P}_t with the
 201 keypoint. We do this by first taking the depth image D_t from the same view as I_t , and deprojecting
 202 all nearby pixels within a radius R : $\mathcal{K}_R = \{[u, v] \in I_t, \|[u, v] - [\hat{u}_t, \hat{v}_t]\| < R\}$ to their associated
 203 3D points $\mathcal{P}_R = \{(x, y, z) = \text{deproject}(u, v), \forall (u, v) \in \mathcal{K}_R\}$. This yields a set of “candidate”
 204 points to consider for interaction. In the bowl grasping example, \mathcal{P}_R would be points on the bottom
 205 of the bowl. Next, we augment the point cloud \mathcal{P}_t with a 1-channel mask $\mathcal{M}_t \in \mathbb{R}^{N \times 1}$ (Fig. 2).
 206 For any point $(x, y, z) \in \mathcal{P}_t$, the mask channel label is 1 if $(x, y, z) \in \mathcal{P}_R$ (i.e., the point is in close
 207 proximity to the deprojected keypoint) and 0 otherwise.

208 Given the pointcloud and keypoint mask, KITE predicts all K waypoints *relative* to individual points
 209 in the point cloud. For each point, we classify which of the K waypoints it is nearest to, along
 210 with the offset to the desired 7-DoF end-effector pose (position and quaternion) for each of the K
 211 waypoints. To do so, we adapt the PointNet++ [50] architecture, and define $Q_\pi : (\mathcal{P}_t, \mathcal{M}_t) \rightarrow \mathcal{P}_\pi$
 212 where $\mathcal{P}_\pi \in \mathbb{R}^{N \times d}$ and $d = K \times (1 + 3 + 4)$. Continuing the example of grasping a bowl, the
 213 predicted pose offsets for each point on the bottom of the bowl (ungraspable) should lead to the
 214 bowl rim (graspable). See Appendix A for more details about the actor. We supervise Q_π using the

215 following per-point loss:

$$L_{\text{skill}} = \lambda_{\text{cls}} CE(\hat{k}, k) + \lambda_{\text{ori}}(1 - \langle \hat{q}_k, q_k \rangle) + \lambda_{\text{pos}} L1([\hat{x}_k, \hat{y}_k, \hat{z}_k], [x_k, y_k, z_k]) \quad (2)$$

216 The first term corresponds to the 1-hot cross-entropy classification loss between the predicted
217 waypoint index \hat{k} and the true nearest waypoint index k . The remaining terms supervise the predicted
218 gripper orientation and position using waypoint κ^k for only the points that have classification label k ,
219 so as to only penalize points that matter (are in close proximity) to the κ^k .

220 **Action Module Inference:** At test time, given a point cloud \mathcal{P}_t with associated keypoint mask
221 \mathcal{M}_t , we use Q_π to obtain $\hat{\mathcal{P}}_\pi$. By taking the highest likelihood point for each of the K indices
222 (representing the points nearest each of the K waypoints). Then, we index the predicted end-effector
223 poses in $\hat{\mathcal{P}}_\pi$ by these K indices, resulting in K waypoints $\{\hat{\kappa}^1, \dots, \hat{\kappa}^K\}$. Finally, we obtain the final
224 trajectory to carry out the skill with using the skill-specific controller: $\tau = \rho(\{\hat{\kappa}^1, \dots, \hat{\kappa}^K\})$.

225 In summary, KITE’s full pipeline first grounds a language command i_t in an observation o_t via
226 Q_{ground} to infer keypoints (Section 3.3), infers the skill label l_t (Section 3.4), maps the label to a
227 skill and controller $(\pi, \rho) = \mathcal{L}[l_t]$, then and executes π and ρ , and finally replans.

228 4 Experiments

229 In this section, we aim to answer the following questions: (Q1) How well does KITE handle scene
230 semantic instructions? (Q2) How well does KITE handle precise object-semantic instructions? (Q3)
231 Does KITE’s scene and object semantic awareness generalize to unseen object instances? and (Q4)
232 Can KITE’s primitives capture precise and dexterous motions beyond pick and place? We first outline
233 KITE’s key implementation details and the baseline methods we benchmark against. Finally, we
234 analyze KITE’s comparative and overall performance across three real-world environments which
235 stress test scene and object-aware semantic manipulation (Section 4.1).

236 **Implementation Details:** Across all evaluation environments, we specify a library of skills and
237 collect 50 kinesthetic demonstrations per-skill. In order to improve precision of the grounding module,
238 and because keypoint supervision is easy to obtain compared to kinesthetic teaching, we supplement
239 the grounding dataset obtained from kinesthetic data collection by manually labeling a small amount
240 of images with paired language instructions (0.75:1 supplemental to original samples ratio). We
241 implement the grounding module according to the architecture from [10] and each waypoint policy
242 in the acting module as a PointNet++ backbone [50] with a point cloud resolution of 20K points. See
243 Appendix A for more details.

244 **Baselines:** We benchmark KITE’s performance against two state-of-the-art instruction-following
245 frameworks. The first is PerAct [11], which trains a PerceiverIO [51] transformer backbone to
246 predict waypoint poses end-to-end conditioned on a voxelized scene representation and language. In
247 comparing against PerAct, we hope to understand whether KITE’s use of keypoint-parameterized
248 skills can offer better precision over end-to-end actions. To understand the value of keypoint-based
249 grounding over frozen representations obtained from VLMs, we compare to RobotMoo [9], which
250 extends a library of language-conditioned skills to additionally condition on segmentation masks
251 from an open-vocabulary object detector. Since exact models and data were not released, we use a
252 state-of-the-art VLM and our set of learn skills for RobotMoo. See Appendix A.3 for more details.

253 4.1 Real-World Evaluation

254 We explore three real-world manipulation environments that provide a rich testbed to explore KITE’s
255 sensitivity to scene and object semantics. Task variations are detailed in Appendix B.1. Across all
256 experimental trials, we use a Franka Emika 7DoF robot and 3 Realsense D435 RGB-D cameras.

257 **Tabletop Instruction-Following:** We train a library of four skills: {pick, place, open, close}
258 to reorganize a tabletop environment with 15 different household objects and an articulated storage
259 organizer with three pull-out drawers (see Figure 1, Appendix B.1). While we do not test on
260 completely unseen objects, we randomly vary the positions of objects on the table and the degree of
261 clutter by adding distractor objects to the scene.

262 Table 1 compares KITE against PerAct and RobotMoo in this setting. We evaluate all approaches
263 with 12 trials of instruction-following across three tiers of difficulty, ranging from a few objects on

264 the table and fairly straightforward language instructions (Tier 1), a visually cluttered table (Tier 2),
 265 and a cluttered table with more ambiguous instructions (Tier 3). See Appendix B.1 for examples of
 266 the objects considered and variations across tiers.

267 We first evaluate individual actions (open,
 268 close, pick), finding KITE to be the most ro-
 269 bust and repeatable. KITE’s use of precise key-
 270 point grounding enables scene semantic aware-
 271 ness (Q1) over different objects (pick, place)
 272 and object semantic understanding (Q2) by dis-
 273 tinguishing amongst different drawer handles
 274 with the open and close skills. PerAct’s disad-
 275 vantage is its discrete visual space, where any
 276 slight 1-off voxel predictions can make it diffi-
 277 cult to grasp objects or cabinet handles. Due to
 278 the weak classification objectives it is trained
 279 on, its most common failure mode is misclassi-
 280 fied gripper opening/closing actions. These fail-
 281 ures are alleviated by the parameterized skills
 282 used in KITE and RobotMoo. Unsurprisingly,
 283 RobotMoo does well at grasping different ob-
 284 jects referenced in language, as VLMs trained
 285 on internet-scale data have strong object priors.
 286 Still, RobotMoo struggles with object semantics
 287 like the distinction amongst top, middle, or bottom
 288 drawers when opening or closing. We find that KITE is also the most competitive framework for
 289 long-horizon sequential reasoning (last two columns in Table 1), and the most common failures still
 290 include grasping the wrong object or with slightly misaligned gripper poses. RobotMoo’s inability to
 291 reason over multiple cabinet handles for opening and closing impedes its long-horizon performance,
 whereas PerAct’s compounding precision errors render long horizon tasks especially difficult.

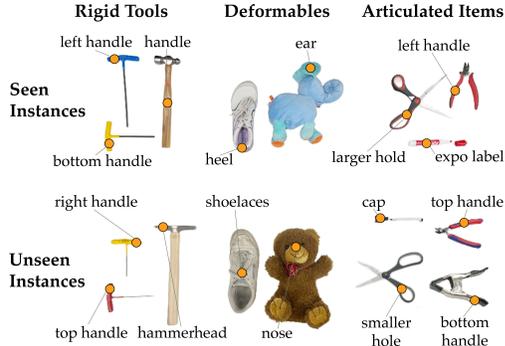


Figure 3: **Semantic Grasping Experimental Setup:** We evaluate KITE on semantic grasping across rigid tools, deformable objects, and articulated items. We show 17 of the 20 objects tested along with ground-truth semantic labels for different features. The top row includes objects seen during grounding module training, and the bottom consists of unseen object instances.

		open	close	pick	pick → place	open → pick → place → close
Tier 1	KITE	1	0.92	0.83	0.75	0.75
	RobotMoo	0.33	0.41	0.75	0.41	0.08
	PerAct	0.08	0.5	0.33	0.08	N/A
Tier 2	KITE	1	0.83	0.76	0.66	0.42
	RobotMoo	0.36	0.36	0.55	0.36	0.09
Tier 3	KITE	0.75	0.83	0.58	0.66	0.58
	RobotMoo	0.33	0.42	0.5	0.42	0.0

Table 1: **Tabletop Instruction Following Results:** Across 12 trials per method per tier, KITE outperforms both RobotMoo and PerAct for individual actions (open, close, pick) and chaining together up to four actions in sequence. We test all approaches on Tier 1 (fewer objects, straightforward language), Tier 2 (more objects, straightforward language), and Tier 3 (more objects, more free-form language). KITE’s use of parameterized skills gives it an edge with precision over PerAct, which is highly susceptible to one-off voxel predictions. This makes skills like opening and picking especially hard, and renders the approach virtually ineffective for higher complexity tiers (2 and 3). RobotMoo is the most competitive approach to KITE, but its main pitfall is a lack of object semantic awareness such as distinguishing amongst different-level cabinet handles.

292 **Semantic Grasping:** Aside from recognizing and manipulating different objects, we explore in
 293 greater detail whether KITE can perform *object-semantic* manipulation (Q2). We evaluate KITE
 294 on the task of *semantic grasping*, with instructions of the form “Pick up the X by the Y” (i.e.
 295 ‘stuffed bear’ and ‘ear’; ‘marker’ and ‘cap’; ‘shoe’ and ‘laces’) (examples in Fig. 1). For these trials,
 296 we train Q_{ground} on a subset of rigid tools, deformable items, and articulated items (Fig. 3) and
 297 retain the keypoint-conditioned pick skill from Section 4.1. We summarize the findings in Table 2
 298 with 26 trials per category of items, noting that KITE can achieve precise semantic grasping with
 299 generalization to unseen object instances (Q2, Q3). We omit a comparison to PerAct as its difficulties

300 with pick-and-place in the tabletop environment are only exacerbated in the semantic grasping setting
 301 where specific intra-object features matter. In the trials summarized in Table 2, KITE outperforms
 302 RobotMoo, suggesting the utility of keypoints to pinpoint specific object parts compared to coarse
 303 segmentation masks or bounding boxes output by VLMs. We also observe that the majority of KITE’s
 304 failures in this setting are due to misinterpretations with symmetry (i.e. grasping the left instead of
 305 right handle of the pliers), rather than a completely erroneous keypoint as is common in RobotMoo.
 306 We posit that this could be alleviated with more diverse data of object semantic variations.

		Rigid Tools	Deformable Objects	Articulated Items	Failures		
					A	B	C
Seen Instances	KITE	0.77	0.77	0.70	5	3	3
Unseen Instances	KITE	0.70	0.54	0.70	4	5	4
All	RobotMoo	0.23	0.35	0.19	2	36	7

Table 2: **Semantic Grasping Results:** Across 20 total objects, 3 diverse object categories, and 26 trials per method per category, KITE achieves the highest rate of pick success for various object semantic features (Fig. 3), and with the least severity of failures. We categorize failure modes as follows, with (A) denoting a symmetry error (picking the left instead of right handle), (B) representing a grounding error with an erroneous keypoint prediction, and (C) indicating a manipulation failure (wrong inferred orientation or slip during grasping).

307 **Coffee-Making:** Finally, we
 308 seek to answer whether KITE
 309 can execute fine-grained be-
 310 haviors from instructions (Q4)
 311 by studying a coffee-making

	reorient_mug	pour_cup	refill_keurig	load_pod
KITE	8/12	9/12	8/12	9/12

Table 3: **Coffee-Making Results:** KITE handles fine-grained manipulation across 4 skills requiring highly precise manipulation.

312 scenario with four skills: {reorient_mug, pour_cup, refill_keurig, load_pod} (examples in
 313 Fig. 1). We evaluate on the same object instances seen in training, but subject to spatial variations
 314 and language variations (i.e. ‘Place the blue/red/green/brown pod in the machine’, ‘Pour the red/grey
 315 pitcher into the mug/Keurig refill area, ‘Place the cup/mug that’s sideways right-side-up.’). Even for
 316 these very fine-grained motions, KITE is able to follow instructions with 67-75% success (Table 3).
 317 The main failures reside with low-level control errors rather than grounding, such as partial coffee
 318 pod insertion, misaligned mugs and pitchers during pouring, or slippage during mug reorientation.
 319 This suggests that the individual skills could benefit from scaling up demonstration collection, while
 320 retaining the existing grounding modules.

321 5 Discussion

322 **Summary** In this work, we present KITE a framework for semantic manipulation that identifies 2D
 323 task-relevant keypoints and extrapolates 6-DoF actions accordingly. By leveraging 2D keypoints
 324 to precisely localize semantic concepts, KITE is adept at recognizing semantic labels both across
 325 different object instances and on different regions within the same object. KITE does action-planning
 326 by drawing from a library of parameterized skills. Empirically, we find that KITE surpasses existing
 327 language-based manipulation frameworks along the axes of scene semantic awareness and object
 328 semantic awareness. We also find that KITE can be trained from orders of magnitude less data and
 329 with large precision gains over end-to-end approaches, while exhibiting an ability to generalize and
 330 operate over extended horizons. Finally, we show that KITE offers a flexible interface for instruction-
 331 following, including tabletop rearrangement, fine-grained grasping, and dexterous manipulation.

332 **Limitations and Future Work** One limiting factor of KITE is its reliance on building a library
 333 of skills. However, we show that a relatively small library of keypoint-parameterized skills is
 334 expressive enough to accomplish many standard manipulation tasks with object variations over an
 335 extended horizon. Additionally, KITE requires less than 50 demonstrations per new skill, meaning
 336 that adding new skills is fairly straightforward. We also note that KITE’s grounding module is
 337 trained from scratch. As VLMs continually improve and in the future may be able to pinpoint
 338 keypoints in images, it would be interesting to replace or enhance KITE’s grounding module with
 339 these models. Additionally, we acknowledge that KITE currently executes skills in an open-loop
 340 manner as parameterized by waypoints. In, the future, we are excited to extend KITE’s skills with
 341 closed-loop feedback and extend the complexity of these skills to even more dexterous settings.

References

- 342
- 343 [1] Y.-W. Chao, W. Yang, Y. Xiang, P. Molchanov, A. Handa, J. Tremblay, Y. S. Narang, K. Van Wyk, U. Iqbal,
344 S. Birchfield, et al. Dexycb: A benchmark for capturing hand grasping of objects. In *Proceedings of the*
345 *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9044–9053, 2021.
- 346 [2] M. Sundermeyer, A. Mousavian, R. Triebel, and D. Fox. Contact-graspnet: Efficient 6-dof grasp generation
347 in cluttered scenes. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages
348 13438–13444. IEEE, 2021.
- 349 [3] J. Mahler, J. Liang, S. Niyaz, M. Laskey, R. Doan, X. Liu, J. A. Ojea, and K. Goldberg. Dex-net 2.0:
350 Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics. *arXiv preprint*
351 *arXiv:1703.09312*, 2017.
- 352 [4] N. Correll, K. E. Bekris, D. Berenson, O. Brock, A. Causo, K. Hauser, K. Okada, A. Rodriguez, J. M.
353 Romano, and P. R. Wurman. Analysis and observations from the first amazon picking challenge. *IEEE*
354 *Transactions on Automation Science and Engineering*, 15(1):172–188, 2016.
- 355 [5] X. Gu, T.-Y. Lin, W. Kuo, and Y. Cui. Open-vocabulary object detection via vision and language knowledge
356 distillation. *arXiv preprint arXiv:2104.13921*, 2021.
- 357 [6] M. Minderer, A. Gritsenko, A. Stone, M. Neumann, D. Weissenborn, A. Dosovitskiy, A. Mahendran,
358 A. Arnab, M. Dehghani, Z. Shen, et al. Simple open-vocabulary object detection. In *Computer Vision–*
359 *ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part X*, pages
360 728–755. Springer, 2022.
- 361 [7] X. Zhou, R. Girdhar, A. Joulin, P. Krähenbühl, and I. Misra. Detecting twenty-thousand classes using
362 image-level supervision. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel,*
363 *October 23–27, 2022, Proceedings, Part IX*, pages 350–368. Springer, 2022.
- 364 [8] S. Liu, Z. Zeng, T. Ren, F. Li, H. Zhang, J. Yang, C. Li, J. Yang, H. Su, J. Zhu, et al. Grounding dino:
365 Marrying dino with grounded pre-training for open-set object detection. *arXiv preprint arXiv:2303.05499*,
366 2023.
- 367 [9] A. Stone, T. Xiao, Y. Lu, K. Gopalakrishnan, K.-H. Lee, Q. Vuong, P. Wohlhart, B. Zitkovich, F. Xia,
368 C. Finn, et al. Open-world object manipulation using pre-trained vision-language models. *arXiv preprint*
369 *arXiv:2303.00905*, 2023.
- 370 [10] M. Shridhar, L. Manuelli, and D. Fox. Cliport: What and where pathways for robotic manipulation. In
371 *Conference on Robot Learning*, pages 894–906. PMLR, 2022.
- 372 [11] M. Shridhar, L. Manuelli, and D. Fox. Perceiver-actor: A multi-task transformer for robotic manipulation.
373 In *Conference on Robot Learning*, pages 785–799. PMLR, 2023.
- 374 [12] E. Jang, A. Irpan, M. Khansari, D. Kappler, F. Ebert, C. Lynch, S. Levine, and C. Finn. Bc-z: Zero-shot
375 task generalization with robotic imitation learning. In *Conference on Robot Learning*, pages 991–1002.
376 PMLR, 2022.
- 377 [13] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, K. Gopalakrishnan, K. Hausman,
378 A. Herzog, J. Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint*
379 *arXiv:2212.06817*, 2022.
- 380 [14] S. Stepputtis, J. Campbell, M. Phielipp, S. Lee, C. Baral, and H. Ben Amor. Language-conditioned
381 imitation learning for robot manipulation tasks. *Advances in Neural Information Processing Systems*, 33:
382 13139–13150, 2020.
- 383 [15] O. Mees, L. Hermann, and W. Burgard. What matters in language conditioned robotic imitation learning.
384 *arXiv preprint arXiv:2204.06252*, 2022.
- 385 [16] F. Codevilla, M. Müller, A. López, V. Koltun, and A. Dosovitskiy. End-to-end driving via conditional
386 imitation learning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages
387 4693–4700, 2018. doi:10.1109/ICRA.2018.8460487.
- 388 [17] L. Shao, T. Migimatsu, Q. Zhang, K. Yang, and J. Bohg. Concept2robot: Learning manipulation concepts
389 from instructions and human demonstrations. *The International Journal of Robotics Research*, 40(12-14):
390 1419–1434, 2021.
- 391 [18] A. Akakzia, C. Colas, P.-Y. Oudeyer, M. Chetouani, and O. Sigaud. Grounding language to autonomously-
392 acquired skills via goal generation. In *ICLR 2021-Ninth International Conference on Learning Representa-*
393 *tion*, 2021.

- 394 [19] P. Goyal, R. J. Mooney, and S. Niekum. Zero-shot task adaptation using natural language. *CoRR*,
395 abs/2106.02972, 2021. URL <https://arxiv.org/abs/2106.02972>.
- 396 [20] H. Hu, D. Yarats, Q. Gong, Y. Tian, and M. Lewis. Hierarchical decision making by generating and
397 following natural language instructions. *Advances in neural information processing systems*, 32, 2019.
- 398 [21] A. Brohan, Y. Chebotar, C. Finn, K. Hausman, A. Herzog, D. Ho, J. Ibarz, A. Irpan, E. Jang, R. Julian, et al.
399 Do as i can, not as i say: Grounding language in robotic affordances. In *Conference on Robot Learning*,
400 pages 287–318. PMLR, 2023.
- 401 [22] W. Huang, F. Xia, T. Xiao, H. Chan, J. Liang, P. Florence, A. Zeng, J. Tompson, I. Mordatch, Y. Chebotar,
402 P. Sermanet, T. Jackson, N. Brown, L. Luu, S. Levine, K. Hausman, and brian ichter. Inner monologue:
403 Embodied reasoning through planning with language models. In *6th Annual Conference on Robot Learning*,
404 2022. URL <https://openreview.net/forum?id=3R3Pz5i0tye>.
- 405 [23] I. Singh, V. Blukis, A. Mousavian, A. Goyal, D. Xu, J. Tremblay, D. Fox, J. Thomason, and A. Garg.
406 Progprompt: Generating situated robot task plans using large language models. *arXiv preprint*
407 *arXiv:2209.11302*, 2022.
- 408 [24] W. Huang, P. Abbeel, D. Pathak, and I. Mordatch. Language models as zero-shot planners: Extracting
409 actionable knowledge for embodied agents. In *International Conference on Machine Learning*, pages
410 9118–9147. PMLR, 2022.
- 411 [25] K. Lin, C. Agia, T. Migimatsu, M. Pavone, and J. Bohg. Text2motion: From natural language instructions
412 to feasible plans. *arXiv preprint arXiv:2303.12153*, 2023.
- 413 [26] Y. Jiang, A. Gupta, Z. Zhang, G. Wang, Y. Dou, Y. Chen, L. Fei-Fei, A. Anandkumar, Y. Zhu, and L. Fan.
414 Vima: General robot manipulation with multimodal prompts. *arXiv preprint arXiv:2210.03094*, 2022.
- 415 [27] J. Liang, W. Huang, F. Xia, P. Xu, K. Hausman, B. Ichter, P. Florence, and A. Zeng. Code as policies:
416 Language model programs for embodied control. In *arXiv preprint arXiv:2209.07753*, 2022.
- 417 [28] A. Stone, T. Xiao, Y. Lu, K. Gopalakrishnan, K.-H. Lee, Q. Vuong, P. Wohlhart, B. Zitkovich, F. Xia,
418 C. Finn, and K. Hausman. Open-world object manipulation using pre-trained vision-language model. In
419 *arXiv preprint*, 2023.
- 420 [29] P. Dayan and G. E. Hinton. Feudal reinforcement learning. *Advances in neural information processing*
421 *systems*, 5, 1992.
- 422 [30] L. X. Shi, J. J. Lim, and Y. Lee. Skill-based model-based reinforcement learning. In *6th Annual Conference*
423 *on Robot Learning*, 2022. URL <https://openreview.net/forum?id=iVxy2e0601U>.
- 424 [31] S. Mirchandani, S. Karamcheti, and D. Sadigh. Ella: Exploration through learned language ab-
425 straction. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, editors, *Ad-*
426 *vances in Neural Information Processing Systems*, volume 34, pages 29529–29540. Curran Asso-
427 *ciates, Inc.*, 2021. URL [https://proceedings.neurips.cc/paper_files/paper/2021/file/](https://proceedings.neurips.cc/paper_files/paper/2021/file/f6f154417c4665861583f9b9c4afafa2-Paper.pdf)
428 [f6f154417c4665861583f9b9c4afafa2-Paper.pdf](https://proceedings.neurips.cc/paper_files/paper/2021/file/f6f154417c4665861583f9b9c4afafa2-Paper.pdf).
- 429 [32] C. Agia, T. Migimatsu, J. Wu, and J. Bohg. Stap: Sequencing task-agnostic policies. *arXiv preprint*
430 *arXiv:2210.12250*, 2022.
- 431 [33] S. Nasiriany, H. Liu, and Y. Zhu. Augmenting reinforcement learning with behavior primitives for diverse
432 manipulation tasks. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2022.
- 433 [34] M. Dalal, D. Pathak, and R. Salakhutdinov. Accelerating robotic reinforcement learning via parameterized
434 action primitives. In *NeurIPS*, 2021.
- 435 [35] C. Lynch, M. Khansari, T. Xiao, V. Kumar, J. Tompson, S. Levine, and P. Sermanet. Learning latent plans
436 from play. *CoRR*, abs/1903.01973, 2019. URL <http://arxiv.org/abs/1903.01973>.
- 437 [36] S. Belkhale and D. Sadigh. Plato: Predicting latent affordances through object-centric play. In *Conference*
438 *on Robot Learning*, pages 1424–1434. PMLR, 2023.
- 439 [37] S. James and A. J. Davison. Q-attention: Enabling efficient learning for vision-based robotic manipulation.
440 *IEEE Robotics and Automation Letters*, 7(2):1612–1619, 2022.
- 441 [38] L. Manuelli, W. Gao, P. Florence, and R. Tedrake. kpm: Keypoint affordances for category-level robotic
442 manipulation. In *Robotics Research: The 19th International Symposium ISRR*, pages 132–157. Springer,
443 2022.
- 444 [39] C. Wang, M. Chai, M. He, D. Chen, and J. Liao. Clip-nerf: Text-and-image driven manipulation of neural
445 radiance fields. *CoRR*, abs/2112.05139, 2021. URL <https://arxiv.org/abs/2112.05139>.

- 446 [40] J. Kerr, C. M. Kim, K. Goldberg, A. Kanazawa, and M. Tancik. Lrf: Language embedded radiance fields,
447 2023.
- 448 [41] X. Ma, D. Hsu, and W. S. Lee. Learning latent graph dynamics for deformable object manipulation. *CoRR*,
449 abs/2104.12149, 2021. URL <https://arxiv.org/abs/2104.12149>.
- 450 [42] H. Ha and S. Song. Flingbot: The unreasonable effectiveness of dynamic manipulation for cloth unfolding.
451 In *Conference on Robotic Learning (CoRL)*, 2021.
- 452 [43] V. Viswanath, J. Grannen, P. Sundaresan, B. Thananjeyan, A. Balakrishna, E. Novoseller, J. Ichnowski,
453 M. Laskey, J. E. Gonzalez, and K. Goldberg. Disentangling dense multi-cable knots. In *2021 IEEE/RSJ*
454 *International Conference on Intelligent Robots and Systems (IROS)*, pages 3731–3738, 2021. doi:10.1109/
455 IROS51168.2021.9636397.
- 456 [44] P. Sundaresan, J. Grannen, B. Thananjeyan, A. Balakrishna, J. Ichnowski, E. R. Novoseller, M. Hwang,
457 M. Laskey, J. E. Gonzalez, and K. Goldberg. Untangling dense non-planar knots by learning manipulation
458 features and recovery policies. *CoRR*, abs/2107.08942, 2021. URL [https://arxiv.org/abs/2107.](https://arxiv.org/abs/2107.08942)
459 [08942](https://arxiv.org/abs/2107.08942).
- 460 [45] P. Sundaresan, S. Belkhale, and D. Sadigh. Learning visuo-haptic skewering strategies for robot-assisted
461 feeding. In *6th Annual Conference on Robot Learning*, 2022. URL [https://openreview.net/forum?](https://openreview.net/forum?id=1Lq09gVoaTE)
462 [id=1Lq09gVoaTE](https://openreview.net/forum?id=1Lq09gVoaTE).
- 463 [46] C. Wang, R. Wang, A. Mandlekar, L. Fei-Fei, S. Savarese, and D. Xu. Generalization through hand-eye
464 coordination: An action space for learning spatially-invariant visuomotor control. In *2021 IEEE/RSJ*
465 *International Conference on Intelligent Robots and Systems (IROS)*, pages 8913–8920. IEEE, 2021.
- 466 [47] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin,
467 J. Clark, et al. Learning transferable visual models from natural language supervision. In *International*
468 *conference on machine learning*, pages 8748–8763. PMLR, 2021.
- 469 [48] J. Wu, R. Antonova, A. Kan, M. Lepert, A. Zeng, S. Song, J. Bohg, S. Rusinkiewicz, and T. Funkhouser.
470 Tidybot: Personalized robot assistance with large language models. *arXiv preprint arXiv:2305.05658*,
471 2023.
- 472 [49] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry,
473 A. Askell, et al. Language models are few-shot learners. *Advances in neural information processing*
474 *systems*, 33:1877–1901, 2020.
- 475 [50] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a
476 metric space. *Advances in neural information processing systems*, 30, 2017.
- 477 [51] A. Jaegle, S. Borgeaud, J.-B. Alayrac, C. Doersch, C. Ionescu, D. Ding, S. Koppula, D. Zoran, A. Brock,
478 E. Shelhamer, et al. Perceiver io: A general architecture for structured inputs & outputs. *arXiv preprint*
479 *arXiv:2107.14795*, 2021.
- 480 [52] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg,
481 W.-Y. Lo, et al. Segment anything. *arXiv preprint arXiv:2304.02643*, 2023.

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 manipulation (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. \(2\)](#).

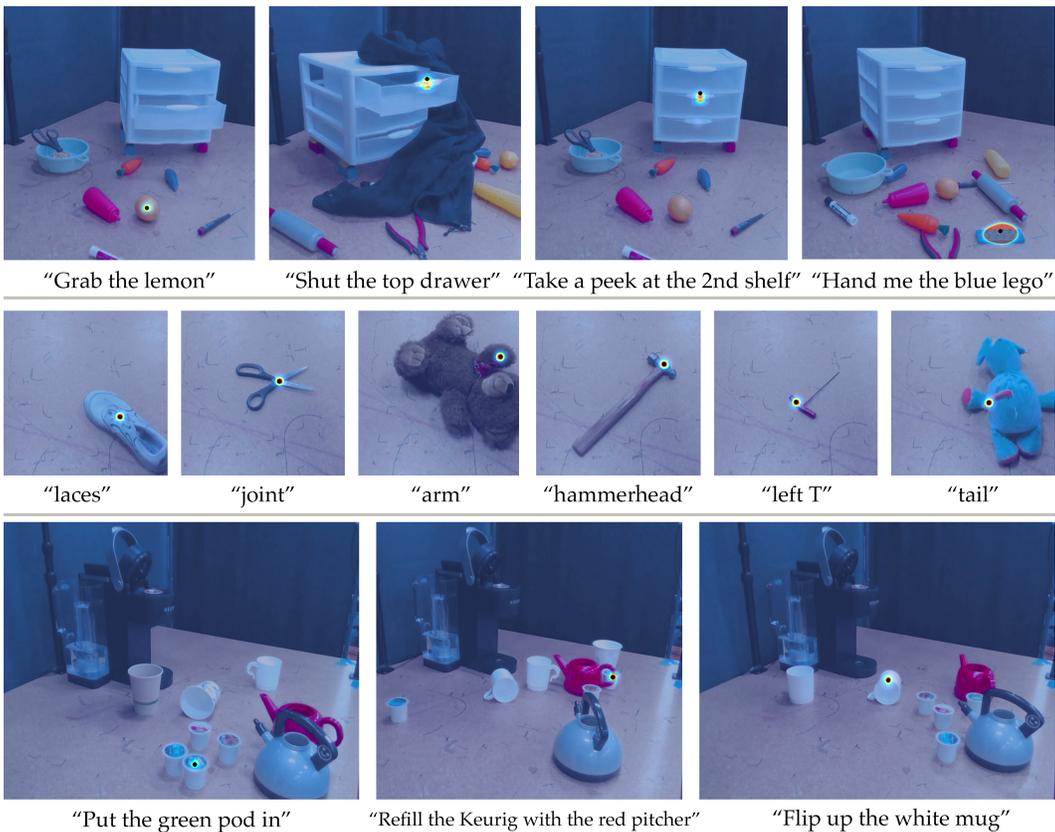


Figure 4: **KITE Grounding Predictions:** KITE’s grounding model is able to accurately predict keypoints for both scene semantic instructions (e.g., “grab the lemon” and “put the green pod in”) and object semantic instructions (e.g., “shut the top drawer” and “take a peek at the 2nd shelf”).

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.

504 **A.2 PerAct**

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

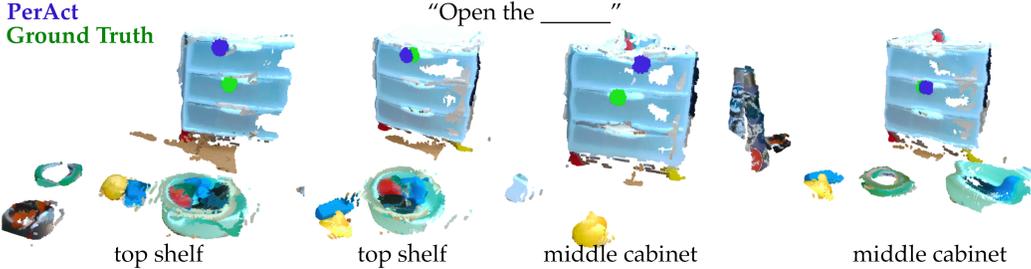


Figure 5: **PerAct Predictions:** We visualize PerAct predictions on the task of opening a cabinet with multiple drawers. Although PerAct exhibits some reasonable predictions (last column), it struggles with localizing the correct handle (1st, 3rd columns). Even when localizing the correct handle (2nd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

512 **A.3 RobotMoo**

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

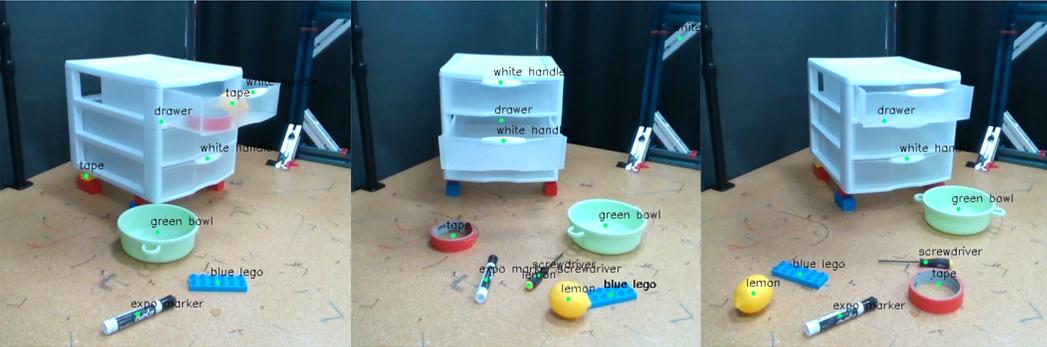


Figure 6: **RobotMoo Predictions:** We visualize the predictions for RobotMoo’s perception stack on tabletop instruction following images. Although RobotMoo exhibits decent scene awareness and an ability to localize different object instances, it struggles with object semantics. Specifically, RobotMoo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

522 **B Real-World Experimental Details**

523 **B.1 Task Variations**

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

528 **B.2 Primitive Instantiations**

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

Tier	Skill	Language	Variations	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
3	open	‘Yank open the top drawer’		randomized position + distractor objects (clothes strewn)
		‘Give the 2nd drawer a tug’		
		‘Take a peek at the 3rd drawer pls’		
		‘Can you check the top drawer?’		
	close	‘Close it’		distractor objects (clothes strewn)
		‘Give it a push’		
		‘Let’s shut the drawer’		
		‘Go ahead and close the top drawer’		
	pick	‘Close any open drawer’		randomized object positions + scene clutter
		‘The one 3rd from the bottom needs to be shut’		
		‘Grab me the [...]’		
		‘Do me a favor and get me the [...]’		
place	‘Can you pass me the [...]?’		randomized object positions + scene clutter	
	‘Get the [...]’			
	‘Locate the [...]’			
	‘Could you hand me the [...] please’			
	‘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’			

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

Category	Object	Language Variations
Rigid Tools	hammer	middle, end, tootip, 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**

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

- 532
- **open:** With its gripper open, the robot approaches 5cm. away from a closed drawer handle at position (x, y, z) with orientation (ψ, θ, ϕ) . 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.
- 536
- **close:** The robot approaches 5cm. away from an opened drawer handle at position (x, y, z) with orientation (ψ, θ, ϕ) , then executes a linear push towards (x, y, z) close the drawer.
- 537
- **pick:** The robot approaches the object located at (x, y, z) , closes its gripper, and lifts 5cm.
- 538
- **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.
- 539
- 540
- 541

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

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

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

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

Table 6: Coffee-Making Variations

561 B.3 LLM Prompting

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

Listing 1: LLM Prompting for Skill Label Inference

```

565 i_t = input("Enter instruction: ")
566 """
567 Input: "Pick up the lemon"
568 Output: ["pick"]
569
570 Input: "Put the screwdriver away"
571 Output: ["pick", "place"]
572
573 Input: "Pls grab me the screwdriver and put it away"
574 Output: ["pick", "place"]
575
576 Input: "Grab the green bowl"
577 Output: ["grasp"]
578
579 Input: "Put the lemon in the bowl"
580 Output: ["pick", "place"]
581
582

```

```
58318 Input: _"Open_the_top_drawer"  
58419 Output: _["open"]  
58520  
58621 Input: _"Pls_shut_the_drawer"  
58722 Output: _["close"]  
58823  
58924 Input: _"put_the_expo_marker_away"  
59025 Output: _["pick",_"place"]  
59126  
59227 Input: _"put_the_Blue_lego_in_the_cabinet"  
59328 Output: _["pick",_"place"]  
59429  
59530 Input: _'%s'  
59631 Output:  
59732 ""_%i_t
```