Supplementary Materials: Recognizing object affordances to support scene reasoning for manipulation tasks

1. Computing grasping location in 3D from 2D

In the physical experiments, the manipulator performs top-down graspings. Therefore, to determine a 3D grasp, one needs to determine grasp location (x, y, z) and grasp orientation theta, in the robot coordinate system.

As shown in Figure 1.(a), we consider three coordinate systems: robot coordinate (O_R), camera coordinate (O_C) and aruco tag coordinate (O_A). With the aruco tag in the view, the transformation from camera to aruco (M_C^A) is known. The transformation from aruco to robot (M_A^R) is known and fixed. Therefore, the transformation from camera to robot is determined (M_C^R).

To determine grasp location (x, y, z) corresponding to the robot coordinate, we first consider the pixels predicted graspable from the 2D affordance map (red part in Figure 1.(b)). Since the sensor provides depth information, the point cloud (3D coordinates of corresponding 2D pixels) in the camera coordinate is generated. The average 3D coordinate of the pixels associated to grasp affordance in the 2D map is used as the grasp location. We transformed the 3D coordinate in the camera coordinate (O_C) to the robot coordinate (O_R) via the aruco coordinate ((O_A)).

To determine the grasp orientation, we first fit a line to pixels predicted as graspable in 2D (pixels in red in Figure 1.(b)) By projecting the line from the camera plane to the table plane (aruco plane) via M_C^A as shown in Figure 1.(c), the slope of the line based on the aruco tag coordinate (O_A) is obtained, so as to the angle θ in the aruco coordinate. Because the end-effector is top-down, the rotation angle for top-down grasping is θ (assuming the initial gripper opening direction is along y-axis of the aruco coordinate).

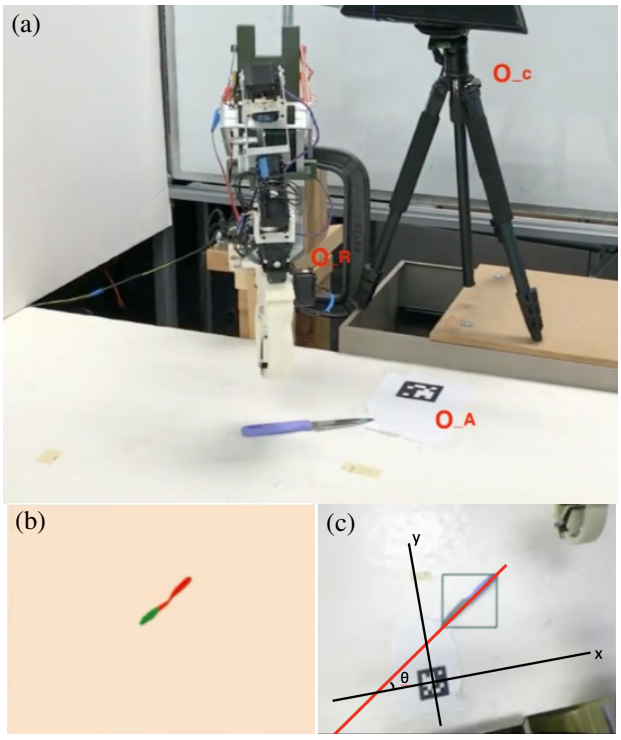

Figure 1: Visualization of the physical experiment setting. (a): Overview of the setting with a RGBD camera, robotic manipulator and aruco tag on the table; (b): Predicted affordance map from the RGBD camera view. Segmentation in red represents grasping area. (best view in color); (c): Fitted line and corresponding in camera coordinate.

2. PDDL in Physical Manipulation

PDDL is a widely used language for planning. Given an initial state and a goal state, PDDL generates a sequence of states. To utilize PDDL, a **problem description** covering the initial state and the goal state needs to be provided. For instance, a simple initial state could be: *ball is graspable; robot can grasp*, and a simple goal state could be: *robot grasps a ball*. Then the PDDL could take the problem description and output a sequence of actions. The output actions lead from initial state to the goal state. The set of **domain-specific actions** are predefined, with corresponding preconditions and effects on state changes per action is triggered. For example, an action *pickup* could result in *X grasps a graspable Y*, if *condition 1: X can grasp* and *condition 2: Y is graspable* are both satisfied.

Figure. 2 illustrates a simple example of **problem description** and **domain-specific actions** required for PDDL.

In the context of goal-oriented manipulations, a goal state and domain actions could be predefined and therefore only the initial state needs to be determined. In our case, with affordance identification, a list of objects and corresponding affordances are predicted by the vision models

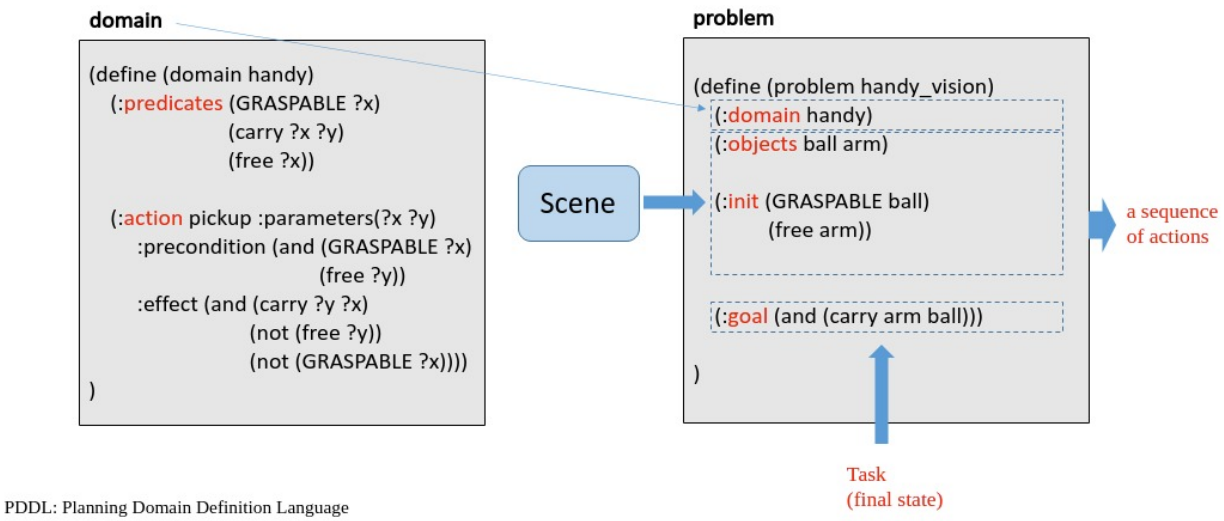

Figure 2: High-level illustration of PDDL pipeline. Left: a domain description is required for PDDL to define a list of candidate actions. For each of the actions, pre-condition of states and effects on states need to be specified; right: a problem description covers the initial state and goal state. In the context of this work, the initial state is obtained via vision information and converted into text, while the goal state is directly specified in text.

and converted to initial states in the **problem description**. To map the PDDL outcomes (a series of actions) to physical execution, a set of atomic robotic actions are implemented for robot executions. To get the location for execution, the point clouds of each detected affordance is preserved during processing the initial state. In this case, the *pickup* action (and other actions such as *drop-off*) could be physically executed by the robotic manipulator.

In our experiments with the object detector and category-agnostic affordance detector, the object label and the associated affordance are detected separately and associated based on the overlapping bounding box locations and similarity of sizes. The detected object label and affordances (e.g. ball is graspable or bowl is containable) are then converted into initial states (as well as the affordance point clouds).

State keeper keeps track of all the symbolic states and spatial parameters of the domain (e.g. bowl is containable, bowl contains ball, location of bowl, location of ball) after the PDDL- generated plan is executed by the robot. In the actual implementation, a textitState keeper could be a dictionary with the key to be the object labels and value to be corresponding state in text and spatial parameters in values. In the experiments with compound tasks which involves more than a single phase, loading scene states of the last phase to the initial state of current phase is necessary. For instance, in the *place objects into empty containers* task, after the first robot execution, the state keeper would specify one of the two containers to be occupied. And therefore only one valid container left for the second remaining object. The *state keeper* is suitable to extend PDDL for multi-goal tasks.

3. More visualizations in physical experiments

In this section, more visualization are provided to illustrate the visual differences between selected instance in physical experiments and instances in the UMD dataset. Figure 3 depicts the instances used in **Affordance on Seen Categories** experiments. The first and second columns illustrate the selected *similar instances*, while the forth and fifth columns show the *dissimilar instances*. An example instance of the same category is listed in the third column.

Figure 4 illustrates examples for the **Affordance with Multiple Objects** setting. As shown the in figures, the *dissimilar set* in the *Seen Categories* are utilized. In each trial in the physical experiment, only one target affordance is evaluated (e.g. contain, support or grasp). After a target affordance is determined, additional nuisance objects without common affordances are put in the scene for the experiment. From left to right, figure 4 illustrates the settings for *contain*, *support*, *grasp* and *grasp*, respectively.

Figure 5 illustrate visuals for the **Affordance on Unseen Categories** experiments. All the instance are in the categories outside the taxonomy of the UMD dataset.

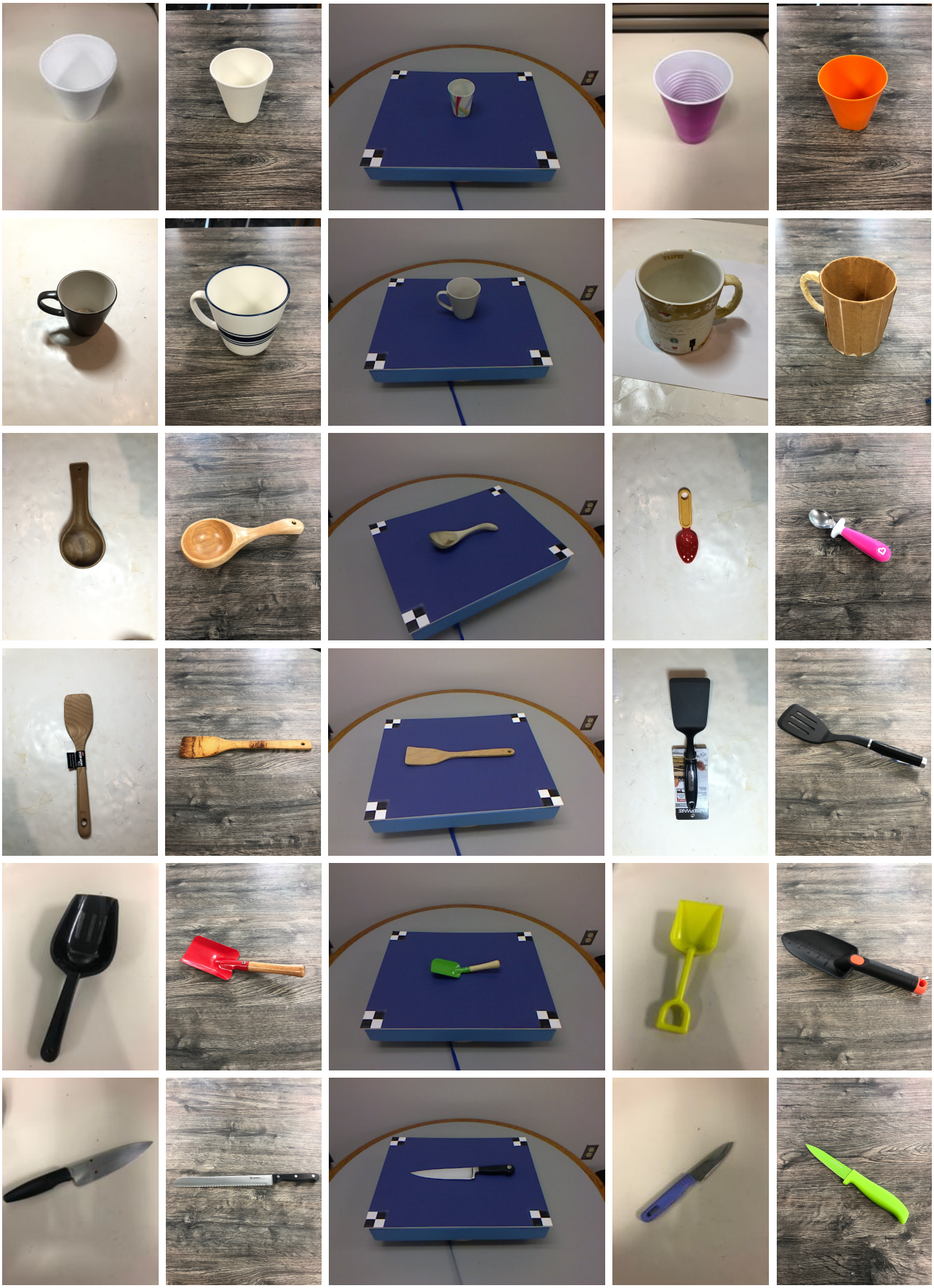

Figure 3: Figures of instances used in physical experiments. The first and the second columns show the found instances similar to UMD dataset (the third column); the forth and the fifth columns show the instances dissimilar to the instances in the UMD dataset.

Figure 4: Examples to illustrate the experimental setting for multiple-objects. Objects are selected from the dissimilar set in the seen objects. In each trial, only one target affordance is evaluated. Additional nuisance objects are in the scene without common affordances. (a) contain (b) support (c) grasp (d) grasp.

Figure 5: Figures of instances used in physical experiments for unseen categories in the UMD dataset. (a) flipper (b) griddle turner (c) grill spatula (d) juice box (e) screwdriver (f) mouse (g) pie server (h) plate (i) pot (j) wrench.