APPENDIX

A. 3D Representation Details

Given image(s) $\mathcal{I} = \{I_i\}_{i=0,1,\ldots,M}$ of one or more RGB-D camera(s), we extract 2D patch-level features $\mathcal{W}_i$ by MaskCLIP [25], including visual patch-level features $\mathcal{W}_i^f$ and vision-language similarity information $\mathcal{W}_i^s$ denoting cosine similarities between language embeddings and $\mathcal{W}_i^f$.

We generate a 3D point cloud $\mathbf{p}$ within the workspace using the camera parameters. For each point p_j of $\mathbf{p}$, we project it back to the ith camera viewpoint as the pixel u_j^i, and get its visual feature f_j^i by interpolation:

$$f_j^i = \mathcal{W}_i^f[u_j^i] \tag{1}$$

Following [18], we compute weights for each camera according to the visibility and distance of p_j relative to the ith camera. We denote the distance from p_j to the ith camera viewpoint as l_i, and compute the depth by interpolating the corresponding depth image I_i^d as $l_i' = I_i^d[u_j^i]$. Then the truncated depth difference is defined as:

$$d_i = l_i - l_i', \quad d_i' = \max(\min(d_i, \mu), -\mu), \tag{2}$$

where $\mu = 0.02$ represents the truncation threshold for the Truncated Signed Distance Function (TSDF). The visibility of p_j in the ith camera viewpoint can be represented as $v_i = \mathbb{1}_{d_i<\mu}$. Here $\mathbb{1}$ is the indicator function. We compute the weight for the ith camera viewpoint as:

$$\beta_i = \exp\left(\frac{\min(\mu - |d_i|, 0)}{\mu}\right). \tag{3}$$

where β_i decays as $|d_i|$ increases. Then, we can obtain the semantic feature f_j by fusing features from M camera viewpoints:

$$f_j = \frac{\sum_{i=1}^M \beta_i v_i f_j^i}{\epsilon + \sum_{i=1}^M v_i} \tag{4}$$

where $\epsilon = 1 \times 10^{-6}$ is to avoid numeric issues.

Similarly, we can get the similarity value s_j for p_j in the same way upon $\mathcal{W}_i^s$. Finally, we get a 3D feature cloud $\mathbf{f} = \{f_j\}$ indicating the visual features and a 3D similarity cloud $\mathbf{s} = \{s_j\}$ indicating the task-relevant information.

B. Data Collection Details

Language Instructions. During each rollout of data collection, we randomly sample a language template along with keywords (target for pick tasks, reference and relation for place tasks) to form a complete language instruction. For pick, there are five language templates: "Give me the {target}", "I need a {target}", "Grasp a {target} object", "I want a {target} object", "Get something to {target}", while there are three for place: "Put it {relation} the {reference}", "Place this {direction} the {reference}", "Move the object {direction} the {reference}". There is a total of 66 object models for data collection, with 36 language keywords categorized into three types: labels, general labels, and attributes. For spatial relations, there are 6 choices for "on" or "around" relations.

Model-based Experts. We collect data with model-based expert planners. The model-based pick expert planner selects the grasp nearest to the target objects from candidates generated by GraspNet [24]. The model-based place expert planner determines valid place regions based on the reference object and the relation. Specifically, we first obtain object region proposals from the mask image in Pybullet [69], where each pixel donates the index of the object visible in the camera. Object regions are identified as bounding boxes of pixels with the same index, and regions whose size is smaller than 5×5 are discarded. Then the valid place region is generated within the reference object for the "on" relation, or around the reference object for the "around" relation. Note that the generated "around" region should not overlap with any object regions. Fig. 11 shows example place regions for the "on" and "around" relations.

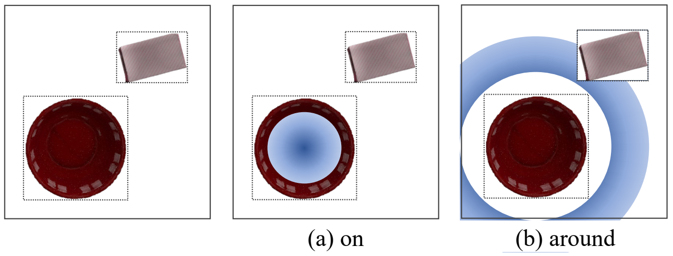

Fig. 11. Example generated place regions for (a) "on" relation and (b) "around" relation relative to the red bowl.

Visual Representation Filtering. We exclude the table points from the visual representations (*i.e.* 3D feature cloud and 3D similarity cloud) for pick tasks while retaining them for place tasks. Specifically, table points are removed by height filtering of the point cloud in world coordinates. This is because the policy does not require the feature information of the table for pick action planning, and the filtering helps the policy focus on the objects.

C. Simulation Experiment Details

Test Case Visualizations. We collect test cases with 66 seen objects and 17 unseen objects. More example test cases across all categories are presented in Fig. 12.

Baseline Implementations. For the two neural field based pick policies, we train the feature fields for each step of action planning and select the grasp pose with the maximum query score of language instructions from a given set generated by GraspNet [24]. For language queries, we use the object as the positive query (*e.g.* "pear", "something to drink"), empty string as the negative query, and "body" as the part query.

For LERF-TOGO [19], we strictly follow the training and querying codes[1] provided by the authors. For input data, there are RGB-D images of 53 views including the 3 used by ours and 50 additional different views to provide more visual information, with a format aligned to their example.

For GraspSplats [14], we used the open-sourced codes[2] for static scene grasping, as it is claimed in the paper to achieve better success rates than dynamic scene grasping. The ground-truth poses for each viewpoint are obtained from the

[1]https://github.com/lerftogo/lerftogo
[2]https://github.com/jimazeyu/GraspSplats

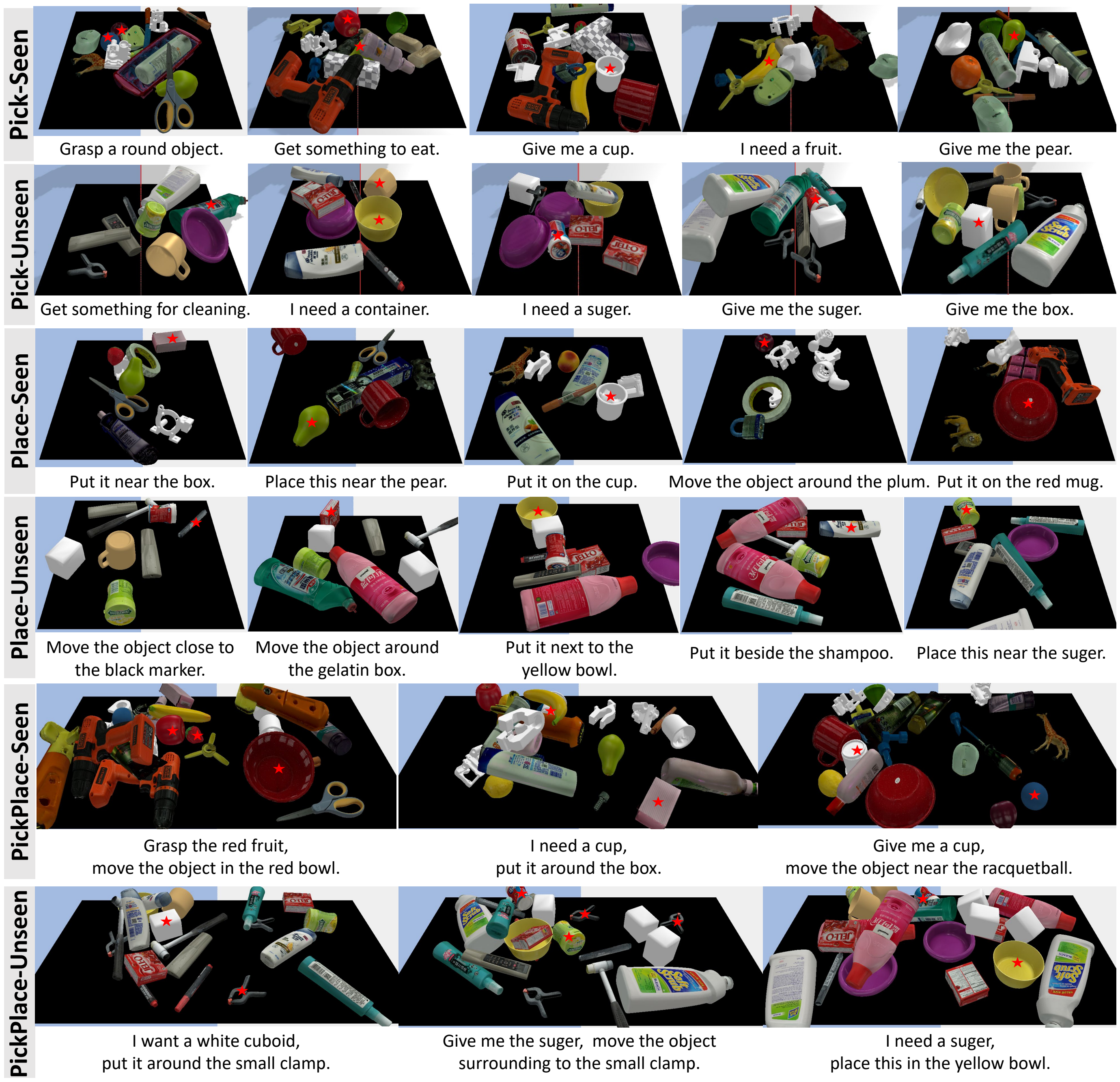
Pick-Seen
Grasp a round object.
Get something to eat.
Give me a cup.
I need a fruit.
Give me the pear.
Pick-Unseen
Get something for cleaning.
I need a container.
I need a suger.
Give me the suger.
Give me the box.
Place-Seen
Put it near the box.
Place this near the pear.
Put it on the cup.
Move the object around the plum.
Put it on the red mug.
Place-Unseen
Move the object close to the black marker.
Move the object around the gelatin box.
Put it next to the yellow bowl.
Put it beside the shampoo.
Place this near the suger.
PickPlace-Seen
Grasp the red fruit, move the object in the red bowl.
I need a cup, put it around the box.
Give me a cup, move the object near the racquetball.
PickPlace-Unseen
I want a white cuboid, put it around the small clamp.
Give me the suger, move the object surrounding to the small clamp.
I need a suger, place this in the yellow bowl.

Fig. 12. More example test cases in simulation. The target objects or reference objects are labeled with stars.

simulation, and written directly to the COLMAP [76] database as required. We run COLMAP for point cloud initialization with ground-truth camera poses. To investigate the influence of image input, we test Graspsplats with 23 view RGB, 3 view RGB-D, and 23 view RGB-D supervision respectively, as shown in Table VII. In these experiments, the input images contain the 3 input views used to evaluate our method. It is worth noting that with the default parameters, GraspSplats fails in most of the cases to select grasp poses because its default distance threshold of $0.02m$ excludes most grasp poses generated by GraspNet. To address this, we increased the distance threshold to $0.08m$.

In the original implementation[2], GraspSplats first queries for the object, then crops the point cloud using a hard-coded workspace limit. This can result in failures when the queried object lies outside the workspace, leaving the cropped point cloud devoid of the target. To mitigate this issue, in Table VII we first crop the point cloud and then query the object, ensuring the target remains within the workspace limit. Therefore, the results of 23 RGB images in Table VII are better than others, which avoids some failures of the queried object outside the workspace.

We analyze failure cases of GraspSplats and identify several key issues. A portion of failures stemmed from the collisions with other objects that cause the target object to drop from the gripper. Also, although GraspSplats can correctly segment

TABLE VII
RESULTS OF GRASPSPLATS WITH DIFFERENT INPUTS

Data	Seen	Unseen
23 RGB	68.0/1.89	31.3/2.19
3 RGB-D	55.0/3.48	34.6/1.36
23 RGB-D	58.0/2.05	37.3/1.667

* Metrics are presented as Task Success Rate / Planning Steps.

the queried object, the selected grasp point might be grasping other surrounding objects since objects are closely packed in the clutter. Additionally, GraspSplats is more sensitive to typos in language instructions, which are included in the test cases.