

# Open-vocabulary Queryable Scene Representations for Real World Planning

Boyuan Chen<sup>1,†</sup>, Fei Xia<sup>2</sup>, Brian Ichter<sup>2</sup>, Kanishka Rao<sup>2</sup>, Keerthana Gopalakrishnan<sup>2</sup>, Michael S. Ryoo<sup>2</sup>, Austin Stone<sup>2</sup>, Daniel Kappler<sup>1</sup>

**Abstract**—Large language models (LLMs) have unlocked new capabilities of task planning from human instructions. However, prior attempts to apply LLMs to real-world robotic tasks are limited by the lack of grounding in the surrounding scene. In this paper, we develop NLMap, an open-vocabulary and queryable scene representation to address this problem. NLMap serves as a framework to gather and integrate contextual information into LLM planners, allowing them to see and query available objects in the scene before generating a context-conditioned plan. NLMap first establishes a natural language queryable scene representation with Visual Language models (VLMs). An LLM based object proposal module parses instructions and proposes involved objects to query the scene representation for object availability and location. An LLM planner then plans with such information about the scene. NLMap allows robots to operate without a fixed list of objects nor executable options, enabling real robot operation unachievable by previous methods. Project website: <https://nlmap-saycan.github.io>

## I. INTRODUCTION

For robots to perform varied, real-world tasks, they must be able to comprehend diverse human commands and then act on these commands in the context of their environment. Imagine a robot in a home environment tasked with “water the plants in the living room”. It has to first identify relevant objects and locations within the scene (e.g., the watering can, the sink, and each potential plant) and then plan over these objects in sequential order (get the watering can, then go the sink, and then fill it up), conditioning on its affordances (e.g., can it carry a full watering can), and conditioning on the scene (e.g., how many plants there are, and where are they). Semantic representation and downstream mobile manipulation planners capable of accessing this representation emerge as critical challenges in such a pipeline.

Semantic understanding is crucial for a robot to achieve long-horizon tasks in unstructured environments. Though a robot can avoid building a semantic representation by finding objects each time they are required, e.g., with Object Goal Navigation [1], [2], this repeated exploration can be inefficient. A persistent scene representation on the other hand avoids this exploration, but past works are generally limited to locating object categories known during the construction of the representation and may not encode the open-vocabulary objects that arise from human queries, such as in “bring me the purple unicorn plush toy”. Recent progress in contrastively trained visual language models offers a promising solution to open-ended scene presentation. Contrastive Language-Image Pre-training (CLIP) [3] models are trained on image-language associations and can provide open-vocabulary image understanding and object detection [4]. They have demonstrated impressive zero-shot classification performance and thus might be used to build a semantic representation in a zero-shot manner.

Another challenge lies in connecting the semantic scene representation to a planning algorithm that is capable of acting

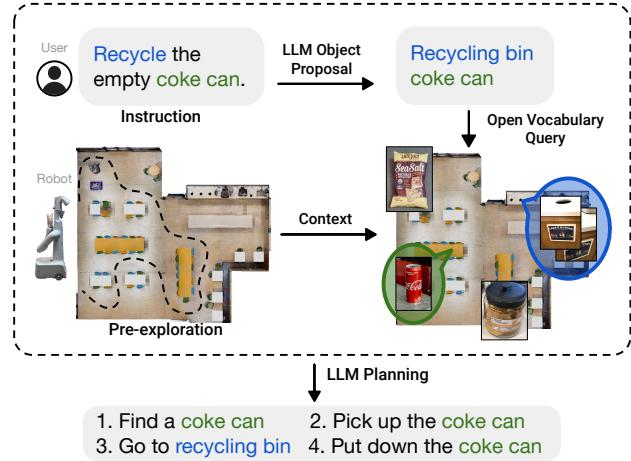


Fig. 1: **NLMap + SayCan overview.** We propose an open-vocabulary and queryable scene representation for real-world planning. A queryable scene representation is built from exploration. When the system receives a user query, it uses an LLM-based object proposal module to propose relevant objects to query the map. The returned object presence and location are used for LLM-based planning. We benchmark the method on robots from Everyday Robots.

upon it. Recent progress in large language models (LLMs), has shown impressive few-shot performance in language comprehension, semantic understanding, and reasoning, as well as application to robotics problems like planning [5]–[7] and instruction following [8]. Using such models in embodied settings can provide significant challenges, most critically because LLMs are not grounded in the physical world. For example, [5] pioneers in using LLMs for planning, but it has no grounding in environmental context. In contrast, SayCan [6] showed how value functions of learned skills can provide such a grounding through selecting options scored highly by a language model and an affordance model. However, this is limited by the options provided and hardcoded knowledge of where objects exist.

In this work, we introduce Natural-Language Map (NLMap), a flexible and language-queryable spatial semantic representation based on visual-language models including ViLD and CLIP and integrate with SayCan. We show that NLMap grounds LLM-based planners in their environments, significantly improves long-horizon planning via natural language instructions in the open-world domain, and enables new tasks prior state-of-the-art algorithms failed to address. To summarize, we make the following contributions:

- 1) We propose an open-vocabulary, queryable semantic representation based on ViLD and CLIP.
- 2) We integrate NLMap into a language-based planner to enable grounding on the context.
- 3) We benchmark NLMap + SayCan in a real-world kitchen, showing it is capable of performing 55 tasks at 61.8% success rate. Notably, 35 of these tasks are impossible with previous state-of-the-art planners that do not have access to NLMap.

<sup>1</sup>Everyday Robots, <sup>2</sup>Robotics at Google, <sup>†</sup> MIT. Emails: boyuanc@mit.edu, xiafei@google.com

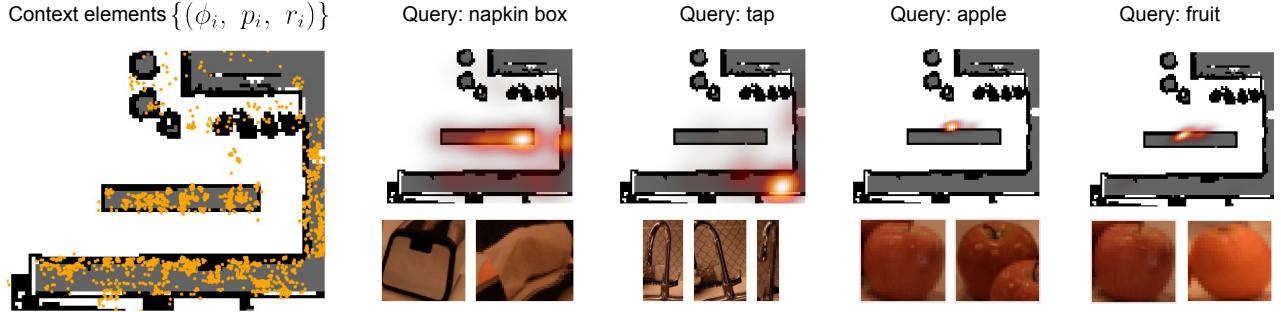


Fig. 2: **Natural Language Queryable Scene Representation.** The key design of NLMap is to establish a queryable map. First, the agent explores the scene and provides a class-agnostic bounding box proposal based on objectness. We extract 512d CLIP features and 512d ViLD features of each bounding box and represent them as a feature point cloud  $\mathcal{C} = \{(\phi_i, p_i, r_i)\}_{i=1\dots N}$ . When queried with a piece of text, we visualize the heatmap of matches based on the alignment of text and visual features. Note that we can query with a single object name, or object families, such as “snack” or “fruit”.

## II. RELATED WORK

**Semantic Scene Representations.** Scene representation is a central theme in robot perception and planning. Semantic SLAM [9]–[11] is an augmentation over traditional SLAM, it assigns semantic features over geometric features provided by SLAM (points, lines, planes). Many representations are proposed, ranging from a faithful 3D reconstruction [12] of the environment, to more object-centric ones [13], [14], such as object detection bounding boxes [15] and 3D bounding boxes [16]. Recently, topological maps [17], [18] and scene graphs [19], [20] emerge as an effective discrete representation of scenes.

One issue with those representations is that they cannot be queried with natural language. Interfacing with those scene representations requires reducing the object set to a closed set, indicating that they are not as useful for LLM-based planners and that they are limited in an open-vocabulary setting. In contrast, our work allows *the scene representation to be queried at test time with natural language*. Concurrent work VLMaps [21] also explores this concept, by fusing pretrained visual-language model features into a geometric reconstruction of the scene. The representation is then used for visual-language navigation tasks via program synthesis.

**Object Goal Navigation.** There is also a significant body of related work on object navigation, which focuses on flexible exploration to find objects in unknown scenes. A few of these algorithms construct a semantic map of the current region before planning in that region [1], [22]–[24]. Map-based methods are modular and interpretable and hence easier to deploy in the real world. Other algorithms [25]–[30] do not require a map and can decide where to go based directly on the current observations and memories, without maintaining a global representation of the environment. Recently, methods that leverage pre-trained image-text models can do zero-shot Object Goal Navigation [31], [32]. CoW [32] performs zero-shot object goal navigation by leveraging CLIP. LM-Nav [8] uses three pretrained model to perform visual language navigation. Our work differs from Object Goal Navigation since the eventual goal is not purely finding objects, but using object presence and location information for planning. Our work can use the representation from a single exploration for many downstream planning tasks without the need to run Object Goal Navigation every time.

**Planning with Scene Representations.** In task and motion planning, scene representations are often composed of predicates compatible with symbolic planners [33], [34]. Recent progresses

attempt to build a symbolic and geometric scene graph to facilitate task and motion planning [35]. However, they still require defining the objects in the scene. Recently LLM-based planners are more flexible [6], [7], [36] and do not require handcrafting predicates, however, they do not handle the complexity of open-vocabulary object proposal and require defining a set of objects involved in planning. They also fail to integrate perception in real robot experiments due to the difficulty of connecting unstructured natural language instruction to perception algorithms that need structured inputs.

## III. PROBLEM STATEMENT

In this work, we aim to efficiently fulfill high-level, natural-language instructions, such as “Bring me a snack” or “I spilled my coffee, can you help?”. This requires a robotic system to solve problems at the intersection of natural language comprehension, scene understanding, task planning, navigation, and manipulation. Recent work, SayCan [6], has shown how large language models can be applied to such problems through world-grounding affordance functions, allowing LLMs to understand what a robot can do from a *state*. However, SayCan did not provide scene-scale affordance grounding, and thus cannot reason over what a robot can do in a *scene*. To that end, we address two core problems (i) how to maintain open-vocabulary scene representations that are capable of locating arbitrary objects and (ii) how to merge such representations within long-horizon LLM planners to imbue them with scene understanding.

## IV. NLMAP + SAYCAN

We provide a high-level description of our algorithm in Listing 1. The design of each component is described below:

### A. Scene Representation

The scene representation is generated from an exploration phase of the unstructured scene, which our approach is agnostic to, but could be for example frontier exploration [37] or pre-determined waypoints. During this exploration, NLMap runs a class agnostic region proposal network as in ViLD [4] on all the observed RGB images. For each proposed region of interest (ROI)  $I_i \in I_{1\dots N}$ , our method uses an ensemble of VLM image encoders  $\Phi_{1\dots M}$  [3], [4] to extract image embeddings  $\phi_i = [\Phi_j(I_i) | j \in 1\dots M]$ . As shown in Fig. 2, such embedding can be queried with text at plan time since VLMs are capable of estimating the correlation

between texts and images. In our setup we leverage CLIP [3] and ViLD [4] as visual encoders  $\phi_i = [\Phi_{clip.img}(I_i), \Phi_{vild.img}(I_i)]$ , where image-text-alignment is scored with inner product of image feature and CLIP text feature. We also extract the estimated location  $p_i = (x_i, y_i, z_i)$  using depth at the center of the image as well as estimated size  $r_i$  of the object in  $I_i$ . Defining the tuple  $c_i = (\phi_i, p_i, r_i)$  as a context element, the collection  $\mathcal{C} = \{c_i\}_{i=1\dots N}$  forms our scene representation.

### B. Querying the Representation

To complete a task specified by human instruction, the robot will query the scene representation for relevant information. This is achieved by first parsing natural language instruction into a list of relevant object names, then using the names as keys to query object locations and availability. Finally, we generate executable options based on what's found in the scene, then plan and execute as instructed.

Listing 1: High-level description of NLMap + SayCan algorithm. Note we only need to build scene representation once for each scene.

```
Input: instruction
if is_new_scene():
    # construct queryable scene representation
    rgbd_images = robot.scene_explore()
    bboxes = roi_proposal(rgbd_images)
    positions, sizes = extract_3d(rgbd_images, bboxes)
    phi = VLM.encode_image(rgbd_images, bboxes)
    nl_map = Context(phi, positions, sizes)
    save_nl_map(nl_map)
else:
    nl_map = load_nl_map()
# extract relevant objects via LLM
objects = LLM.object_proposal(instruction)
# extract text features
queries = VLM.encode_text(objects)
# query the nl_map
object_scores = queries.dot_product(nl_map.Phi)
object_presence, locations
    = multiview_fusion(object_scores, nl_map)
scene_objects = objects.filter_by(object_presence)
# planning with scene objects information
LLM.plan(instruction, scene_objects)
```

1) *Object proposal*: The core challenge of querying scene information is bridging unstructured natural language input and structured representations. In order to decide what objects to look up in the scene representation, we use few-shot prompting to let LLM actively propose required objects given an instruction. Different from previous work [8] that uses LLM to extract names from a sentence, our object proposal is much more demanding in four different ways as we will discuss in Sec. V-B.

In order to achieve a reliable object proposal that addresses four requirements, we introduce example prompts for each case and use the few-shot prompting technique of LLMs to propose them. The few-shot examples can be found on our project website.

2) *Object Query*: Given a list of object names  $\{y_i\}_{i=1\dots O}$ , we then query the scene representation for object locations and availability. This is achieved by finding top k nearest neighbor elements in  $\mathcal{C}$  followed by a clustering algorithm to fuse multi-view information. A threshold on a cluster's score determines if the queried object is found. We first define a metric  $D : \mathcal{C} \times \mathcal{Y} \rightarrow \mathbb{R}$  where  $\mathcal{Y}$  is the set of possible object names. We use the maximum

ensemble of both CLIP and ViLD for the metric D defined below:

$$D : (\phi_i, p_i, r_i), y_i \mapsto \max(D_{clip}, D_{vild}), \text{ where}$$

$$D_{clip} = \langle \Phi_{clip.img}(I_i), \Phi_{clip.text}(I_i) \rangle$$

$$D_{vild} = \langle \Phi_{vild.img}(I_i), \Phi_{clip.text}(I_i) \rangle$$

Here we use both CLIP embedding and ViLD embedding because the former detects out-of-distribution objects better while the latter is more robust to common objects as shown in Fig. 5. We can directly take the maximum over the two inner products because both of them are normalized vectors designed to be queried by the inner product CLIP text encoder. Given metric  $D$ , the top k nearest neighbor elements for object name  $y_i$  can be found in the scene representation  $\mathcal{C}$ . We note that based on the value of  $D(c_i, y_i)$ , we can impose a threshold to filter out low-confidence detections. These top context elements are associated with ROIs, multiple of which may correspond to the same real-world 3D object instance. We then run a multi-view fusion algorithm to aggregate these context elements into 3d object locations and filter out objects that don't exist according to an aggregated score. Details of the algorithm can be found in Sec. VI-B.

### C. Combining NLMap and SayCan

Our method constructs a scene representation queryable by natural language. Such representation can be connected with LLM-based planners to enable robots to operate in a truly uncontrolled environment. Previously, SayCan [6] presents a framework that allows robots to plan and execute in the real world following human instructions. We highlight the difference between our work and SayCan in Fig. 3. SayCan work as follows: with few-shot prompting, SayCan uses the scoring of a language model to break down a high-level instruction like “Bring me an apple” to “1. Find the apple, 2. Pick up the apple, 3. Bring it to you, 4. Put down the apple”. Each option from a pre-defined list is scored by an LLM and an affordance prediction module. However, SayCan relies on a hard-coded list of object names, locations, and executable options so its capability is largely limited by the lack of contextual grounding.

NLMap makes up this missing component in SayCan. Our object proposal, combined with the object query, generates the relevant object names and locations conditioned on the instruction and the scene. There are two major remaining challenges.

1) *Generate executable options*: Vanilla SayCan [6] provides a list of skills associated with either 1) navigation policies to hard-coded locations 2) manipulation policies (pick and place) of objects, specified by object names. Given a detected object and its location, we can create a new skill “find the [object name]” bound to a navigation policy to that location. This means we can expand a small fixed set of navigation options to infinitely many options. On the other hand, although training manipulation policies for infinitely many objects is beyond the scope of our work, we can still augment the manipulation capability of SayCan by binding all possible references to a manipulable object with the available manipulation policies. This is achieved by finding CLIP nearest neighbor of object names. For example, given discovered objects, we can generate executable options like “pick up the red can” and “pick up a tin of coke”. Our method will bind both of them to the closest manipulation policy “pick up coke can” with CLIP. This nearest neighbor query is similar to that used with BERT in [5].

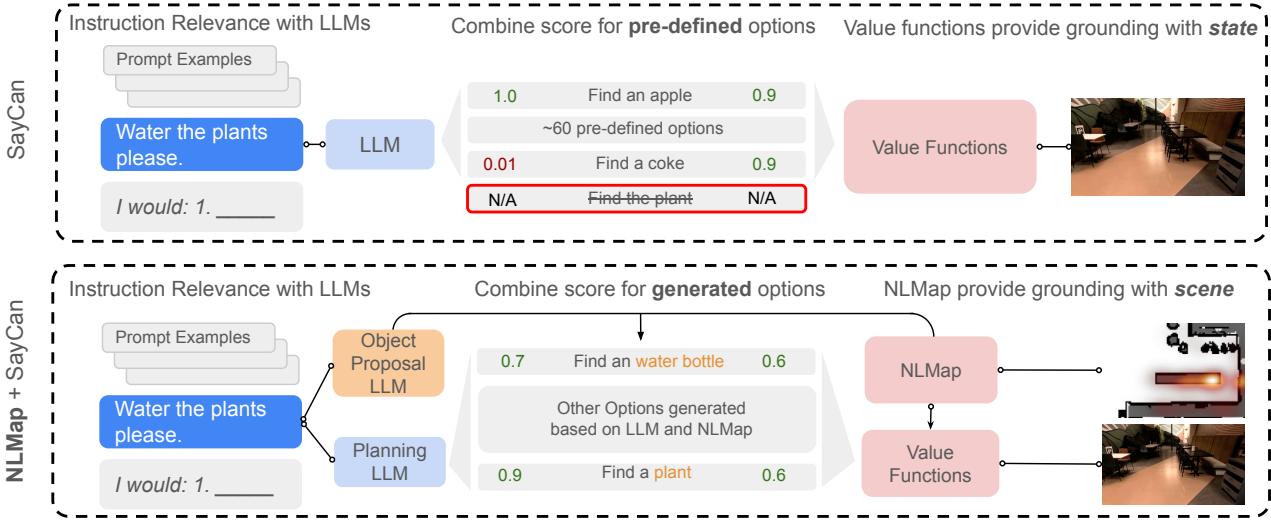


Fig. 3: **Comparison of NLMap + SayCan with SayCan**: With few-shot prompting, SayCan uses the scoring of a language model to break down a high-level instruction like “Bring me an apple” to “1. Find the apple, 2. Pick up the apple, 3. Bring it to you, 4. Put down the apple”. Each option from a pre-defined list is scored by an LLM and an affordance prediction module. It failed on “water the plant” task since plant is not contained in the pre-defined options. **NLMap + SayCan** Our method generates a list of relevant objects based on instruction with an LLM module, and then queries the NLMap to filter the object list and get object locations. A list of options is generated based on a template find/pick/place object, and then LLM-based planning module plan over these options.

2) *Ground LLM planner with context*: Unlike the setup in SayCan, which assumes all objects in the hard-coded list are present, our method is expected to tackle infeasible instructions, such as instructions involving objects that aren’t present. SayCan weakly addresses this problem by grounding plans with local affordance, which is only conditioned on what’s directly visible in the field of view rather than what’s available in the entire scene. NLMap gives us a list of available objects so we can add the missing global contextual grounding to SayCan. This is achieved by modifying the original few shot prompts in SayCan to also condition the plan on discovered objects, expressed in templates like “Scene: apple, coke can.” We include both positive examples when necessary objects are all present and negative examples when available objects cannot fulfill the instruction. In the former case, LLM is prompted to plan just like in vanilla SayCan; In the latter case, LLM is prompted to output the terminate signal “done” directly, indicating the task is infeasible.

With these components, we can ground SayCan with context awareness. After exploring the scene, when a human gives the robot an instruction, the robot will propose potentially involved objects in the scene and query the gathered scene representation for their locations and availability. NLMap then generates executable options, plans with LLM conditioned on what’s found and finally executes the plan in the real world under the SayCan framework.

## V. EXPERIMENTS

In this section, we evaluate NLMap and its individual components with real-world robotics tasks. We test a robot running NLMap in a real office kitchen, as shown in Fig. 4. We test the entire system in an end-to-end setting such that the robot attempts to accomplish tasks specified by humans with natural language. We list a subset of the manipulable objects in Fig. 4(a) receptacle locations in Fig. 4(b). The robot is a mobile manipulator from Everyday Robot, which has a mobile base and a 7-degree-of-freedom arm, as shown in Fig. 4(c). The main sensor is an RGBD

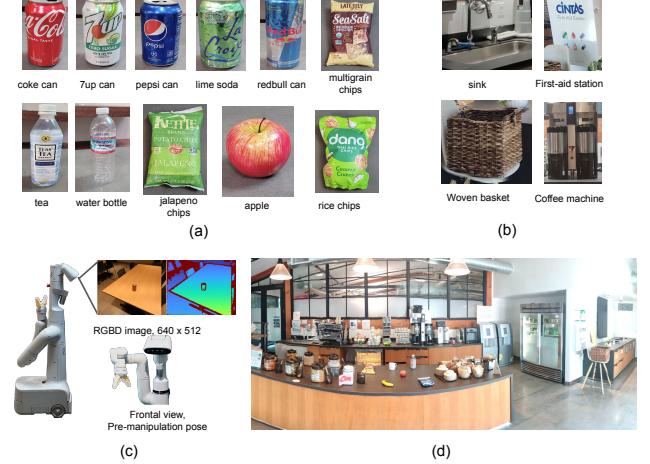


Fig. 4: (a) a representative subset of objects that are used in manipulation (b) a representative subset of objects used as receptacles (e.g. for the task putting the cup next to the coffee machine) (c) a robot from Everyday Robots used in the experiment (d) The scene where we run the experiments, it is a kitchen in an office building.

camera, which returns  $640 \times 512$  RGBD images. Similar to SayCan, we use a set of manipulation policies trained from imitation learning and PaLM 540B [38] as the LLM for all experiments, due to its good performance on new tasks with few-shot prompting. Throughout this section, all experiments share the same set of hyper-parameters and LLM prompts unless specified otherwise. A full list of test instructions can be found on the project website.

### A. Benchmarking NLMap + SayCan as a system

In this section, we demonstrate our natural language queryable representation can be combined with LLM planners to significantly augment the capability of real robot operation. We choose to combine NLMap with SayCan, a recent work that uses LLM planners to let robots plan and execute according to natural language instructions. One of the biggest limitations of SayCan, as stated in Sec. III, is that it has no global context awareness. By combining our

method with SayCan using the method described in Sec. IV-C, we free SayCan from a fixed, hard-coded set of objects, locations, or executable options. With NLMap, SayCan can now perform a great number of previously unachievable tasks. In addition, we demonstrate that our method allows SayCan to plan with the global context to identify infeasible tasks. We quantitatively evaluate the real robot performance of NLMap + SayCan in Table I with three sets of benchmarks. We compare our method with a privileged version of SayCan, which uses ground truth perception results in the scene.

1) *SayCan tasks*: We hope to understand how much performance will be lost compared to SayCan due to the addition of perception and context-aware planning. Therefore, we benchmark 18 tasks adopted from 6 of the 7 task families from the original SayCan paper with 3 random tasks from each family (except for Embodiment family). Our method achieves a success rate of 66.7% among these tasks compared to the 84% of privileged SayCan. We also tried 2 tasks with deliberate typos ‘ppsi’ ‘chpis’. Our method failed in both instructions with typos, with one failure during object proposal and one failure due to policy binding. With these two typo experiments included, our method achieves an overall success rate of 60% compared to 65% in real robot experiments compared to privileged SayCan that has hard-coded object locations. This shows our NLMap maintains a reasonable overall success rate even if multiple components like object proposal, perception, and context-conditioned planning are added.

2) *Novel objects*: SayCan relies on a hard-coded list of object names, locations, and executable options. Since the hard-coded set of objects and executable options are finite, SayCan is incapable of performing tasks that involve objects or skills outside these small sets. However, since NLMap can propose and detect objects, and generate executable options itself, NLMap can be combined with SayCan to execute infinitely many tasks that involve such novel objects as described in Sec. IV-C. As shown in Table I, SayCan fails to plan nor execute any of these tasks while our method achieves a success rate of 80% in the end-to-end execution experiment. It even succeeds in some very out-of-distribution instructions such as “I want to watch TV, can you get a bottle of tea and put it there” or “Show me where is the first aid station”. We note that manipulation policies used in this project are still limited to be with the objects that are visually similar to training objects in [6] and rely on the generalization to slightly out-of-distribution data. Therefore, the novel object names in this experiment are either used for navigation only, or for describing objects that are visually similar to training objects in [6]. Such constraint can be lifted in the future when a general text-conditioned manipulation policy is available but lifting it is beyond the scope of the project.

3) *Missing Objects*: Vanilla SayCan isn’t grounded by what’s available in the scene. If a necessary object is removed from the scene, there is no way for SayCan’s LLM planner to tell the task is infeasible. With NLMap, we can use the method in Sec. IV-C to condition SayCan planning on what’s actually detected. In this benchmark, we ask NLMap + SayCan to perform tasks that require objects not present in the scene. Instructions in the benchmark consist of size 15 subset of all instructions in the “novel object” benchmark since we cannot remove objects like “first aid station” from the wall. In a successful run, the robot is expected to not detect an object doesn’t exist and output a termination signal immedi-

Task Family	NLMap+SayCan		SayCan*	
	Planning	Execution	Planning	Execution
SayCan Tasks	0.8	0.6	0.8	<b>0.65</b>
Novel Objects	0.9	<b>0.8</b>	0.0*	0.0
Missing Objects	0.67	<b>0.4</b>	0.0*	0.0

TABLE I: **Planning and execution success rate.** NLMap +SayCan shows comparable performance as SayCan on instructions from [6] while enabling new tasks SayCan cannot do before due to its lack of contextual grounding. Planning success rate for NLMap + SayCan refers to that of generative planning. (\*SayCan uses privileged ground truth perception information, thus not able to handle objects out of the pre-defined list.)

Task Family	PaLM540B [38]	PaLM62B	PaLM8B
Instruction implication	0.92	0.84	0.72
Crowd-sourced	0.96	0.96	0.72
Detailed description	0.72	0.8	0.6
Proper granularity	0.6	0.2	0.133

TABLE II: Object proposal achieves a very high success rate for all task families except the hardest set “proper granularity”. The performance on task family “proper granularity” sharply declines when we use smaller models while other tasks families witnessed minor decline.

ately in its plan. Our method achieves a success rate of 40% in the missing object setting, where 56% of the total failure cases are due to false positive detections. Although vanilla SayCan will achieve a success rate of zero in comparison, this benchmark still indicates false positive detection is a challenge for context-aware planning.

### B. Benchmarking Object Proposal

Object proposal is a foundational component in our framework to parse unstructured instructions into structured object names. We investigate the robustness and generalization capability of object proposal from four perspectives:

- Infer objects from implication of the instruction: e.g. “Heat up the taco” (taco, microwave)
- Unstructured crowd-sourced instructions: e.g. “Redbull is my favorite drink, can I have a one please?” (redbull, human)
- Objects with fine-grained description: e.g. “turn off the macbook with yellow stickers” (macbook with yellow stickers)
- Decomposition to proper granularity: e.g. “check out what types of ingredients are available to cook a luxurious breakfast” (milk,eggs,bacon,bread,butter,cheese,ham,sausage...)

A summary of result of each perspective can be found in Table II.

1) *Infer objects from implication of the instruction*: In previous work [8] that use LLM to extract object names from language, all object names are nouns that are directly present in the language input. However, in the real world, humans frequently give instructions that involve objects that have to be inferred from the implication of the task. We test object proposal on 25 such instructions and evaluate whether proposed objects would complete the task. Object proposal achieved a success rate of 92% in 25 test cases including “season the steak (salt, pepper)”, “fillet the fish (fish, knife)”.

2) *Unstructured crowd-sourced instructions*: Object proposal module is expected to take in instructions from a variety of highly unstructured formats. We evaluate the robustness of our object proposal on a set of 25 test instructions adopted from crowd-sourced instructions for SayCan. Object proposal achieved a success rate of 96% in this study, including multi-step tasks like “Move an

multigrain chips to the table and an apple to the far counter”. Object proposal succeeded in all 8 out of 9 multi-step tasks in this study.

3) *Reference to objects with fine-grained description*: Human instructions often involve reference to objects with fine-grained descriptions. Such descriptions are often important to visually identify a particular instance in the scene. Thus it’s important for the object proposal to keep these fine-grained descriptions in its output. We evaluate object proposal on 25 test instructions that involve fine-grained descriptions by adjectives or clauses. The model attains a success rate of 72% in this experiment. The model even succeeded in some complicated descriptions like “mug in the shape of a donut”.

4) *Decomposition to proper granularity*: Many instructions require a different level of object proposal granularity. Certain tasks can only be accomplished if the object proposal is more fine-grained. We evaluate object proposal on 15 tasks that require expanding a category mentioned in the instruction. Overall, the object proposal achieves a success rate of 60% in this set, indicating that proper granularity is still a hard challenge for LLM due to its multi-modality nature.

### C. Benchmarking Object Queries to NLMap

In this section, we evaluate the open-vocabulary object query module on a list of 50 common objects in our testing kitchens. We run robot exploration and object query in two different kitchen scenes, each with some object deliberately missing. Our method uses both maximum ensemble metric  $D$  and multi-view fusion described in Sec. IV-B with  $k=4$ . We compare this choice with alternative embeddings and metrics like  $D_{clip}$  or  $D_{vild}$ . Maximum ensemble metric  $D$  without multi-view fusion is also evaluated as a baseline. We have  $k=1$  in the above three baselines since no multi-view fusion is happening. As shown in Table III, ViLD and CLIP embedding alone achieves a very low success rate in both environments. As illustrated in Fig. 5, we observe that ViLD embedding detects common objects like cans or apples more reliably while suffering from false negative detection of out-of-distribution objects such as “first aid station”. On the other hand, CLIP embedding gives us better results on uncommon objects but is less robust for basic objects. Additionally CLIP embeddings better captures features of text and signs. Our method uses multi-view fusion in addition to the maximum ensemble. Multiview fusion leads to a slight 2% accuracy increase in scene 1 but a significant 17% increase in the second scene. This shows that multi-view fusion can help remove outlier observations that produce high likelihood scores but are actually noise by noticing a lack of detection of it from different views. Overall, the perception success rate for our method is 82% and 64% respectively in the two kitchens. Such accuracy is limited by the low resolution and exposure of our robot camera. However, since instructions don’t always contain visually ambiguous objects like many in these test queries, perception is still reliable enough as we see in the real robot experiments Sec. V-A.

### D. Benchmarking Context Grounded Planning

Failures from perception or object proposal are coupled with planning in real robot experiments. In this section, we ablate context-aware LLM planning as a standalone component, assuming correct object proposal and detection. We test LLM

Method	Scene 1	Scene 2
ViLD embedding	0.6	0.47
CLIP embedding	0.58	0.44
Maximum Ensemble	0.8	0.47
Maximum Ensemble + Multiview fusion	<b>0.82</b>	<b>0.64</b>

TABLE III: We ablate different object query methods in two real-world scenes. Both ViLD and CLIP achieve low query success rate but the ensemble of their maximum score as well as our multi-view fusion algorithm provides a significant boost to the query success rate.



Fig. 5: **Comparison of different ROI retrieval method**. We ablate using different features to retrieval Rols with natural language and found there are unique failure cases with either CLIP or ViLD features, while maximum ensemble of features provide the best results.

planning in a generative way. A generated plan is considered correct if it will accomplish the instruction, is consistent with the available objects, and is executable. We benchmark generative planning with 80 test cases consisting of 40 instructions with 2 set of available objects for each. One set is a positive set that contains all needed objects for the task while the other set is a negative set with some necessary objects missing. To be considered successful, the planner should behave like Vanilla SayCan in the positive set while outputting the terminal signal immediately in the negative set. Our LLM planner, conditioned on available objects using the method described in Sec. IV-C, achieves a success rate of 85% and 60% on the 40 instructions with positive object set and negative set respectively. The performance gap is expected because negation is known to be a hard problem for LLM.

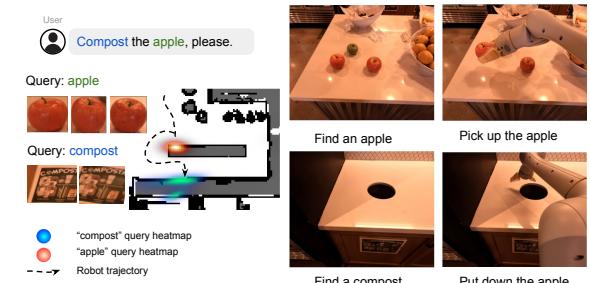


Fig. 6: Execution trajectory of proposed method on task “Compost the apple”. Note CLIP features allow the robot to understand the sign on the compost bin. The images are from the onboard camera of a robot from Everyday Robots.

## VI. CONCLUSIONS

We integrate NLMap, a flexible and queryable spatial semantic representation based on visual-language models including ViLD and CLIP with SayCan. We show that NLMap is a flexible scene representation that grounds LLM-based planners in their environments, significantly improving long-horizon planning via natural language instructions in open-worlded domain, enabling new tasks prior state-of-the-art algorithms failed to address.

**Future work.** Currently, NLMap only handles a static scene representation without dynamic objects and human, which we will leave this for future work. All the modules used in NLMap + SayCan is pre-trained and deployed zero-shot. It is a great advantage but we hope to fine-tune them for better performance. Additionally, we will look into efficient exploration algorithms to speed up the creation of scene representation.

## ACKNOWLEDGEMENTS

Special thanks to Arjun Majumdar, Andy Zeng and Karol Hausman for helpful discussions; we thank Peng Xu and Xiran Liu for helpful feedbacks on writing.

## REFERENCES

- [1] D. S. Chaplot, D. Gandhi, A. Gupta, and R. Salakhutdinov, “Object goal navigation using goal-oriented semantic exploration,” in *In Neural Information Processing Systems (NeurIPS)*, 2020.
- [2] P. Anderson, A. Chang, D. S. Chaplot, A. Dosovitskiy, S. Gupta, V. Koltun, J. Kosecka, J. Malik, R. Mottaghi, M. Savva *et al.*, “On evaluation of embodied navigation agents,” *arXiv preprint arXiv:1807.06757*, 2018.
- [3] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark *et al.*, “Learning transferable visual models from natural language supervision,” in *International Conference on Machine Learning*, 2021.
- [4] X. Gu, T.-Y. Lin, W. Kuo, and Y. Cui, “Open-vocabulary object detection via vision and language knowledge distillation,” *arXiv preprint arXiv:2104.13921*, 2021.
- [5] W. Huang, P. Abbeel, D. Pathak, and I. Mordatch, “Language models as zero-shot planners: Extracting actionable knowledge for embodied agents,” *arXiv preprint arXiv:2201.07207*, 2022.
- [6] M. Ahn, A. Brohan, N. Brown, Y. Chebotar, O. Cortes, B. David, C. Finn, C. Fu, K. Gopalakrishnan, K. Hausman, A. Herzog, D. Ho, J. Hsu, J. Ibarz, B. Ichter, A. Irpan, E. Jang, R. J. Ruano, K. Jeffrey, S. Jesmonth, N. Joshi, R. Julian, D. Kalashnikov, Y. Kuang, K.-H. Lee, S. Levine, Y. Lu, L. Luu, C. Parada, P. Pastor, J. Quiambao, K. Rao, J. Reettinghouse, D. Reyes, P. Sermanet, N. Sievers, C. Tan, A. Toshev, V. Vanhoucke, F. Xia, T. Xiao, P. Xu, S. Xu, M. Yan, and A. Zeng, “Do as i can and not as i say: Grounding language in robotic affordances,” in *arXiv preprint arXiv:2204.01691*, 2022.
- [7] W. Huang, F. Xia, T. Xiao, H. Chan, J. Liang, P. Florence, A. Zeng, J. Tompson, I. Mordatch, Y. Chebotar, P. Sermanet, N. Brown, T. Jackson, L. Luu, S. Levine, K. Hausman, and B. Ichter, “Inner monologue: Embodied reasoning through planning with language models,” in *arXiv preprint arXiv:2207.05608*, 2022.
- [8] D. Shah, B. Osinski, B. Ichter, and S. Levine, “Lm-nav: Robotic navigation with large pre-trained models of language, vision, and action,” *arXiv preprint arXiv:2207.04429*, 2022.
- [9] J. Civera, D. Gálvez-López, L. Riazuelo, J. D. Tardós, and J. M. M. Montiel, “Towards semantic slam using a monocular camera,” in *2011 IEEE/RSJ international conference on intelligent robots and systems*, 2011.
- [10] L. Zhang, L. Wei, P. Shen, W. Wei, G. Zhu, and J. Song, “Semantic slam based on object detection and improved octomap,” *IEEE Access*, 2018.
- [11] S. L. Bowman, N. Atanasov, K. Daniilidis, and G. J. Pappas, “Probabilistic data association for semantic slam,” in *2017 IEEE international conference on robotics and automation (ICRA)*, 2017.
- [12] F. Xia, A. R. Zamir, Z.-Y. He, A. Sax, J. Malik, and S. Savarese, “Gibson env: real-world perception for embodied agents,” in *Computer Vision and Pattern Recognition (CVPR), 2018 IEEE Conference on*, 2018.
- [13] M. Runz, M. Buffier, and L. Agapito, “Maskfusion: Real-time recognition, tracking and reconstruction of multiple moving objects,” in *2018 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, 2018.
- [14] J. McCormac, R. Clark, M. Bloesch, A. Davison, and S. Leutenegger, “Fusion++: Volumetric object-level slam,” in *2018 international conference on 3D vision (3DV)*, 2018.
- [15] C. R. Qi, X. Chen, O. Litany, and L. J. Guibas, “Imvotenet: Boosting 3d object detection in point clouds with image votes,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020.
- [16] D. Xu, D. Anguelov, and A. Jain, “Pointfusion: Deep sensor fusion for 3d bounding box estimation,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018.
- [17] K. Chen, J. K. Chen, J. Chuang, M. Vázquez, and S. Savarese, “Topological planning with transformers for vision-and-language navigation,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021.
- [18] K. Chen, J. P. de Vicente, G. Sepulveda, F. Xia, A. Soto, M. Vázquez, and S. Savarese, “A behavioral approach to visual navigation with graph localization networks,” *arXiv preprint arXiv:1903.00445*, 2019.
- [19] S.-C. Wu, J. Wald, K. Tateno, N. Navab, and F. Tombari, “Scenegraphfusion: Incremental 3d scene graph prediction from rgb-d sequences,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021.
- [20] I. Armeni, Z.-Y. He, J. Gwak, A. R. Zamir, M. Fischer, J. Malik, and S. Savarese, “3d scene graph: A structure for unified semantics, 3d space, and camera,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019.
- [21] C. Huang, O. Mees, A. Zeng, and W. Burgard, “Visual language maps for robot navigation,” *arXiv preprint arXiv:2210.05714*, 2022.
- [22] D. S. Chaplot, D. Gandhi, S. Gupta, A. Gupta, and R. Salakhutdinov, “Learning to explore using active neural slam,” in *International Conference on Learning Representations (ICLR)*, 2020.
- [23] D. S. Chaplot, R. Salakhutdinov, A. Gupta, and S. Gupta, “Neural topological slam for visual navigation,” in *CVPR*, 2020.
- [24] K. Fang, F.-F. Li, S. Savarese, and A. Toshev, “Scene memory transformer for embodied agents in long time horizon tasks,” in *CVPR 2019*, 2019.
- [25] A. Talukder, S. Goldberg, L. Matthies, and A. Ansar, “Real-time detection of moving objects in a dynamic scene from moving robotic vehicles,” in *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)(Cat. No. 03CH37453)*, 2003.
- [26] J. Santos-Victor and G. Sandini, “Visual-based obstacle detection: a purposive approach using the normal owl,” in *Proc. of the International Conference on Intelligent Autonomous Systems, Karlsruhe, Germany*, 1995.
- [27] A. Mousavian, A. Toshev, M. Fišer, J. Košecká, A. Wahid, and J. Davidson, “Visual representations for semantic target driven navigation,” in *2019 International Conference on Robotics and Automation (ICRA)*, 2019.
- [28] T. Chen, S. Gupta, and A. Gupta, “Learning exploration policies for navigation,” in *International Conference on Learning Representations*, 2019.
- [29] S. K. Ramakrishnan, D. S. Chaplot, Z. Al-Halah, J. Malik, and K. Grauman, “Poni: Potential functions for objectgoal navigation with interaction-free learning,” in *Computer Vision and Pattern Recognition (CVPR), 2022 IEEE Conference on*, 2022.
- [30] A. Wahid, A. Stone, K. Chen, B. Ichter, and A. Toshev, “Learning object-conditioned exploration using distributed soft actor critic,” *CoRR*, 2020.
- [31] A. Majumdar, G. Aggarwal, B. Devnani, J. Hoffman, and D. Batra, “Zson: Zero-shot object-goal navigation using multimodal goal embeddings,” *arXiv preprint arXiv:2206.12403*, 2022.
- [32] S. Y. Gadre, M. Wortsman, G. Ilharco, L. Schmidt, and S. Song, “Clip on wheels: Zero-shot object navigation as object localization and exploration,” *arXiv preprint arXiv:2203.10421*, 2022.
- [33] C. R. Garrett, T. Lozano-Pérez, and L. P. Kaelbling, “Ffrob: An efficient heuristic for task and motion planning,” in *Algorithmic Foundations of Robotics XI*.
- [34] ———, “Pddlstream: Integrating symbolic planners and blackbox samplers via optimistic adaptive planning,” in *Proceedings of the International Conference on Automated Planning and Scheduling*, 2020.
- [35] Y. Zhu, J. Tremblay, S. Birchfield, and Y. Zhu, “Hierarchical planning for long-horizon manipulation with geometric and symbolic scene graphs,” in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021.
- [36] A. Zeng, A. Wong, S. Welker, K. Choromanski, F. Tombari, A. Purohit, M. Ryoo, V. Sindhwani, J. Lee, V. Vanhoucke *et al.*, “Socratic models: Composing zero-shot multimodal reasoning with language,” *arXiv preprint arXiv:2204.00598*, 2022.
- [37] B. Yamauchi, “Frontier-based exploration using multiple robots,” in *Proceedings of the second international conference on Autonomous agents*, 1998.
- [38] A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H. W. Chung, C. Sutton, S. Gehrmann *et al.*, “Palm: Scaling language modeling with pathways,” *arXiv preprint arXiv:2204.02311*, 2022.

## APPENDIX

### A. Context-aware SayCan Algorithm

Our context-aware SayCan algorithm is similar to [6], it expands the last line `LLM.plan(instruction, scene_objects)` in Listing 1. Compared to the original SayCan [6], our context-aware version needs a list of detected object names  $\mathcal{M}$ , along with a list of template functions  $\mathcal{F}$  as extra input. A template function maps an object name to an option name such as  $x \rightarrow \text{"pick up } [x]\text{"}$ . We note that the template function is used here because training manipulation policies beyond pick-and-place are beyond the scope of our project. If we have a language-conditioned policy in the future, we don't need to use template functions anymore. Trusting LLM for new options will suffice in that case. A full pseudo-code can be found in Algo 1.

---

#### Alg. 1 Context-Aware SayCan

---

```

1: Input: A high level instruction  $i$ , a list of detected scene object
   names  $\mathcal{M}$ , a list of template functions  $\mathcal{F}$ , state  $s_0$ , and a set
   of skills  $\Pi$  and their affordance functions  $V_\Pi$  along with their
   language descriptions  $d_\Pi$ .
2:  $\ell_{\mathcal{A}} \leftarrow \{\text{"done"}\}$ 
3:  $\text{translate} \leftarrow \{\}$ 
4: for  $o \in \mathcal{M}$  do
5:   for  $f \in \mathcal{F}$  do
6:      $\ell_{\mathcal{A}}.\text{append}(f(o))$             $\triangleright$  Create executable options
7:      $\pi_{nn} \leftarrow \text{argmax}_{\pi \in \Pi}(\text{clip}(d_\pi), \text{clip}(f(o)))$ 
8:      $\text{translate}[f(o)] = \pi_{nn}$          $\triangleright$  Bind options to policies
9:    $n \leftarrow 1$ 
10:  while  $\ell_{\mathcal{A}_{n-1}} \neq \{\text{"done"}\}$  do
11:     $\mathcal{Q} = \emptyset$ 
12:    for  $a \in \mathcal{A}$  and  $\ell_a \in \ell_{\mathcal{A}}$  do
13:       $\pi = \text{translate}[\ell_a]$ 
14:       $q_a^{\text{LLM}} = p(\ell_a | i, \mathcal{M}, \ell_{a_{n-1}}, \dots, \ell_{a_1})$            $\triangleright$  LLM score
15:       $q_a^{\text{affordance}} = V_\pi(s_n)$                                  $\triangleright$  Affordance
16:       $q_a^{\text{combined}} = q_a^{\text{affordance}} q_a^{\text{LLM}}$ 
17:       $\mathcal{Q} = \mathcal{Q} \cup q_a^{\text{combined}}$ 
18:     $a_n = \text{argmax}_{a \in \mathcal{A}} \mathcal{Q}$ 
19:     $\pi_n = \text{translate}[\ell_{a_n}]$ 
20:    Execute  $\pi_n(s_n)$  in the environment, updating state  $s_{n+1}$ 
21:     $n = n + 1$ 

```

---

### B. Multi-view fusion algorithm

In this section, we describe details of the multi-view fusion algorithm mentioned in Sec. IV-B. In the gathered scene representation  $\mathcal{C}$ , multiple context elements may be associated with the same object. Each context element  $c_i$  contains an estimation of object centroid  $p_i$  and along with a object width  $r_i$ . To simplify formulation, we use cylindrical bounding volumes to model 3d objects. We create such bounding boxes with center  $p_i$  and radius  $r_i$  in an upright position. Given each queried object name  $y_i$ , we can quickly narrow down bounding box candidates by finding the top k nearest neighbors with metric  $D$ . We now have a problem similar to post-processing in object detection - for each real object instance, we may have overlapping bounding box predictions,

which are supposed to be aggregated together. In computer vision, this is achieved by the NMS algorithm that group predictions based on the intersection over union(IOU) of the bounding box followed by keeping only the bounding box with the highest confidence in each group. We made three major changes to the NMS algorithm by noticing the special structure of our problem.

First, since our bounding volumes are not cubes, IOU is hard to compute. We instead use KL divergence of Gaussian distributions to model. For each cylindrical bounding box  $(p_i, r_i)$  with a circular projection on the 2d plane, define Gaussian distribution  $G_i = \mathcal{N}(p_i, \alpha \cdot r_i)$ . The 2d Gaussian will have its center at the estimated centroid and standard deviation proportional to the width of the object. KL divergence measures how different two distributions are so it acts like the IOU for gaussian distributions. When estimations have very different centers or sizes, they will be considered to correspond to two different object instances by our algorithm. Second, different from the setup in 2d object detection, different estimations of the same object in our problem are considered valid, independent data points that contribute to a better estimation of object location. Therefore, we don't discard non-maximum estimations in each clustered group, but rather use their score as importance weights to derive the final estimation through weighted average. Third, bounding boxes are directly filtered out based on a threshold on confidence score in 2d detection. In our setup, we give confidence scores a bonus based on how many elements there are by noticing available objects should be detected from multiple view points.

We then offer a formal algorithm box for multi-view fusion in Algo 2. Given object name  $y$ , we can use metric  $D$  to score each context element in  $\mathcal{C}$  and find the top k ones. Denote the indices of top k context elements as  $\mathcal{K}$ , sorted in descending order by score. For each context element  $c_i = (\phi_i, p_i, r_i)$ , define Gaussian distribution  $\mathcal{N}_i = \mathcal{N}(p_i, \alpha \cdot r_i)$ . In our experiments, we choose the monotonic increasing function  $f$  to be in the form  $f(x) = 1 + t - \frac{t}{x}$  where  $t$  is some hyper-parameter.

---

#### Alg. 2 Multi-view Fusion in NLMap

---

```

1: Input: Sorted indices  $\mathcal{K}$ , Scores  $S$  for context elements,
   Gaussian distributions for context elements  $N$ , KL threshold
    $\lambda$ , score threshold  $\beta$ , monotonic increasing function  $f$ .
2:  $\text{Groups} \leftarrow [[\mathcal{K}[0]]]$ 
3: for  $i \in \mathcal{K}$  do
4:   for  $j \in \mathcal{K}$  do
5:     if  $\forall G \in \text{Groups}, \forall z \in G, i \neq z \wedge j \neq z$  then
6:       if  $KL(\mathcal{N}_i, \mathcal{N}_j) > \lambda$  then
7:          $\text{Groups}[-1].\text{append}(j)$ 
8:       else
9:          $\text{Groups}.\text{append}([i])$ 
10:     $P \leftarrow []$ 
11:    for  $G \in \text{Groups}$  do
12:      if  $G[0] \cdot f(|G|) > \beta$  then
13:         $P.\text{append}(\frac{\sum_{i \in G} p_i \exp(S_i)}{\sum_{i \in G} \exp(S_i)})$ 
14: return  $P$ 

```

---

The algorithm then outputs clustered locations for objects

queried by name *y*.

### C. Prompt used for object proposal and for planning

#### Listing 2: Object proposal prompt in NLMap + SayCan.

The task 'hold the snickers' may involve the following objects:snickers.  
The task 'wipe the table' may involve the following objects: table, napkin, sponge, towel.  
The task 'put a water bottle and an oatmeal next to the microwave' may involve the following objects:water bottle, oatmeal, microwave.  
The task 'place the mug in the cardboard box' may involve the following objects:mug, cardboard box.  
The task 'go to the fridge' may involve the following objects:fridge.  
The task 'put a grapefruit from the table into the bowl' may involve the following objects:grapefruit, table, bowl.  
The task 'can you open the glass jar' may involve the following objects:glass jar.  
The task 'heat up the taco and bring it to me' may involve the following objects:taco, human, microwave oven, fridge.  
The task 'hold the fancy plate with flower pattern' may involve the following objects:fancy plate with flower pattern.  
The task 'put the fruits in the fridge' may involve the following objects:fridge, apple, orange, banana, peach, grape, blueberry.  
The task 'get a sponge from the counter and put it in the sink' may involve the following objects:sponge, counter, sink.  
The task 'empty the water bottle' may involve the following objects:water bottle, sink.  
The task 'i am hungry, give me something to eat' may involve the following objects:human, candy, snickers, chips, apple, banana, orange.  
The task 'go to the trash can for bottles' may involve the following objects:trash can for bottles.  
The task 'put the apple in the basket and close the door' may involve the following objects:apple, basket, door.  
The task 'help me make a cup of coffee' may involve the following objects:cup, coffee, mug, coffee machine.  
The task 'check what time is it now' may involve the following objects:clock, watch.  
The task 'let go of the banana' may involve the following objects:banana, trash can.  
The task 'put the grapes in the bowl and then move the cheese to the table' may involve the following objects: grape, bowl, cheese.  
The task 'find a coffee machine' may involve the following objects:coffee machine.  
The task 'clean up the spilled coke' may involve the following objects:spilled coke, towel, mop, napkin, sponge.  
The task 'bring me some soft drinks' may involve the following objects:human, pepsi, coke, sprite, fanta, 7 up.  
The task 'boil some water' may involve the following objects: water, kettle, sink, tap.  
The task 'wash the dishes' may involve the following objects: sink, tap, mug, plate, bowl, fork, spoon, knife.  
The task 'place a knife and a banana to the table' may involve the following objects:knife, banana, table.

#### Listing 3: Task planning prompt in NLMap + SayCan.

**Robot:** Hi there, I'm a robot operating in an office kitchen. You can ask me to do various tasks and I'll tell you the sequence of actions I would do to accomplish your task.  
**Human:** Hold the snickers  
**Available objects** are: snickers.  
**Explanation:** Hold on means to pick it up. I will pick up the snickers.  
**Robot:** 1. pick up the snickers  
2. done.  
**Human:** Put the trailmix here  
**Available objects** are: trailmix.  
**Explanation:** This is a single action of putting down the trailmix.  
**Robot:** 1. put down the trailmix  
2. done.

**Human:** Put a water bottle and an oatmeal next to the microwave  
**Available objects** are: water bottle, oatmeal, microwave.  
**Explanation:** I will find and move a water bottle to the microwave , then find and move an oatmeal to the microwave.  
**Robot:** 1. find the water  
2. pick up the water  
3. go to microwave  
4. put down the water  
5. find the oatmeal  
6. pick up the oatmeal  
7. go to the microwave  
8. put down the oatmeal  
9. done.  
**Human:** Put the mug in the cardboard box  
**Available objects** are: .  
**Explanation:** There is no mug nor cardboard box available so I will do nothing.  
**Robot:** 1. done.  
**Human:** Put a grapefruit from the table into the bowl  
**Available objects** are: grapefruit, bowl, table.  
**Explanation:** I will find and pick up the grapefruit , then find the bowl and put the grapefruit in it.  
**Robot:** 1. find the grapefruit  
2. pick up the grapefruit  
3. go to the bowl  
4. put down the grapefruit  
5. done.  
**Human:** Get a sponge from the counter and put it in the sink  
**Available objects** are: sponge, sink.  
**Explanation:** I will find a sponge first. Then I will pick up the sponge and put it down in the sink.  
**Robot:** 1. find the sponge  
2. pick up the sponge  
3. go to the sink  
2. pick up the sponge  
3. go to the sink  
4. put down the sponge  
5. done.  
**Human:** Move the water from the table to the counter  
**Available objects** are: water, table, counter.  
**Explanation:** The user has requested moving the water to the counter. I will find the water and pick it up, then put it on the counter.  
**Robot:** 1. find the water  
2. pick up the water  
3. go to the counter  
4. put down the water  
5. done.  
**Human:** Bring me some snacks  
**Available objects** are: human, twix.  
**Explanation:** The user has requested some snacks, twix is a type of snack. I will bring the user some twix.  
**Robot:** 1. find the twix  
2. pick up the twix  
3. find the human  
4. put down the twix  
5. find the dried fruit  
6. pick up the dried fruit  
7. find the human  
8. put down the dried fruit  
9. done.  
**Human:** Bring me an orange  
**Available objects** are: human.  
**Explanation:** There is no orange available so I will do nothing.  
**Robot:** 1. done.  
**Human:** Give me something to drink  
**Available objects** are: human, pepsi, 7 up, milk, juice.  
**Explanation:** The user has requested some drinks, pepsi is a type of drink. I will bring the user some pepsi.  
**Robot:** 1. find the pepsi  
2. pick up the pepsi  
3. find the human  
4. put down the pepsi  
5. done.  
**Human:** Put the apple in the basket  
**Available objects** are: basket.  
**Explanation:** There is no apple available so I will do nothing.  
**Robot:** 1. done.  
**Human:** Deliver a can of coke to me  
**Available objects** are: coke can.  
**Explanation:** The user has requested

a coke, I will find a coke and bring it to the user.  
**Robot:** 1. find the coke  
2. pick up the coke  
3. find the human  
4. put down the coke  
5. done.

**Human:** I finished my drink  
, can you throw it away and bring me a mountain dew  
**Available objects** are: coke can, mountain dew.

**Explanation:** The user has requested to throw his drink  
(which appears to be a coke) away. I will put the coke  
in the trash, and then bring the user a mountain dew.

**Robot:** 1. find the sprite  
2. pick up the sprite  
3. go to the trash  
4. put down the sprite  
5. find the mountain dew  
6. pick up the mountain dew  
7. find the human  
8. put down the mountain dew  
9. done.

**Human:** Let go of the banana  
**Available objects** are: banana.

**Explanation:** Let go  
simply means to put down. I will put down the banana.

**Robot:** 1. put down the banana  
2. done.

**Human:** Put the grapes  
in the bowl and then move the cheese to the table  
**Available objects** are: grape, bowl, cheese.

**Explanation:** This is a query  
with two parts. The first part requires finding grapes  
and then moving them to the bowl. The second part  
requires finding the cheese and moving it to the table  
. I will make sure to finish both parts of the task.

**Robot:** 1. find the grapes  
2. pick up the grapes  
3. go to the bowl  
4. put down the grapes  
5. pick up the cheese  
6. go to the table  
7. put down the cheese  
8. tell you I'm done.

**Human:** Pick up the peanuts  
**Available objects** are: human, peanut.

**Explanation:** This is a simple  
task of a single step, I will just pick up the peanuts.

**Robot:** 1. pick up the peanuts  
2. done.

**Human:** Pick up the water bottle

**Available objects** are: .

**Explanation:** There  
is no water bottle available so I will do nothing.

**Robot:** 1. done.

**Human:** Bring me the peanuts

**Available objects** are: peanut.

**Explanation:** The user has requested peanuts  
, I will find peanuts and bring them to the user.

**Robot:** 1. pick up the peanuts  
2. find the human  
3. put down the peanuts  
4. done.

**Human:** Throw away a coffee cup

**Available objects** are: coffee cup, trash can.

**Explanation:** The user has  
requested me to throw away a coffee cup. Throwing away  
means putting something in the trash can. I will find a  
coffee cup, pick that up and then put it in the trash.

**Robot:** 1. find the coffee cup  
2. pick up the coffee cup  
3. go to the trash  
4. put down the coffee cup  
5. done.

**Human:** Place a knife and a banana to the table

**Available objects** are: knife, table.

**Explanation:** : There is no banana available so I will do nothing.

**Robot:** 1. done.

**Human:** Throw away the fruits

**Available objects** are: apple, orange, banana, lime.

**Explanation:** : The user has requested me to throw away the fruits.  
Throwing away means putting something in the trash can  
. Banana is a type of fruit that's available. I will  
find banana, pick that up and then put it in the trash.

**Robot:** 1. find the banana  
2. pick up the banana  
3. go to the trash  
4. put down the banana  
5. done.

#### D. Object proposal experiment task list

Listing 4: Object proposal task list, where robot needs to infer objects from tasks

```
make lasagna
cook chicken tikka masala
make a sandwich
recycle the coke can
freeze the ice cream in the shopping bag
blend pineapples and mangos to make some smoothies
fillet the fish
find some container to serve the steak
compost the apple
water the plant
slice the sausages and put them into a bowl
microwave the to go box
give me something to brush my teeth
light up the room
season the steak
cook an egg
bake the apple pie
fill the paper cup with water
cut the paper in half
wash away the dusts on the cutting board
drain the rice
stir fry the bok choy
steam the dumplings
sharpen the knife
throw away the yogurt cup
```

Listing 5: Object proposal task list, where the robot needs to understand complex human language inputs

```
I opened a pepsi earlier. bring me an open can
I spilled my coke, can you bring me a replacement
I spilled my coke, can you bring me something to clean it up
I accidentally dropped that jalapeno chips after eating it.
Would you mind throwing it away
I like fruits, can you bring me something I would like
There is a close counter, a far counter, and a table. visit
all the locations
There is a close counter, a trash can, and a table. visit
all the locations
Redbull is my favorize drink, can I have a one please
Would you bring me a coke can
Please, move the pepsi to the close counter
Can you move the coke can to the far counter
Would you throw away the bag of chips for me
Put an energy bar and water bottle on the table
Bring me a lime soda and a bag of chips
Can you throw away the apple and bring me a coke
Bring me a 7up can and a tea
Move an multigrain chips to the table and an apple to the
far counter
Move the lime soda, the sponge, and the water bottle to the
table
Bring me two sodas
Move three cokes to the trash can
Throw away two cokes
Bring me two different sodas
Bring me an apple, a coke, and water bottle
I spilled my coke on the table, throw it away and then bring
me something to help clean
I just worked out, can you bring me a drink and a snack to
recover
```

Listing 6: Object proposal task list, where reference to objects contains fine grained descriptions

```

put the red can in the trash bin
put the brown multigrain chip bag in the woven basket
find the succulent plant
pick up the up side down mug
put put the apple on the macbook with yellow stickers
use the dyson vacuum cleaner
bring me the kosher salt
put the used towels in washing machine
move the used mug to the dish washer
place the pickled cucumbers on the shelf
find my mug with the shape of a donut
put the almonds in the almond jar
fill the zisha tea pot with water
take the slippery floor sign with you
give me my slippers that have holes on them
find the mug on the mini fridge
bring me the mint flavor gum
find some n95 masks
grab the banana with most black spots
fill the empty bottle with lemon juice
throw away the apple that's about to rot
throw away the rotting banana
take the box of organic blueberries out of the fridge
give a can of diet coke
open the drawer labelled as utensils

```

Listing 7: Object proposal task list, where robot needs to infer objects from categories and decompose it to the right granularity

```

list some different types of masks in the house
find out what types of pastries are there in the kitchen
tell me what type of spices we have in the kitchen
find some appropriate storages for mugs
what are some protein rich food
check out what types of ingredients are available to cook a
luxurious breakfast
bring me a bunch of flowers
find me some different types of Chinese dumplings in the
freezer
give me a bunch of different flowers
put different kinds of common cheeses in the fridge
list all available vegetables in the fridge
give me some sweet snacks
give me some savory snacks
give me some first-aid items
mix all types of wines in the cabinet

```

#### E. Robot experiment task list

Listing 8: Task List used in experiment. The scene setup is the same as in SayCan [6].

```

put the coke can in the your gripper
let go of the coke can
come to the table
deliver the red bull to the close counter
throw away the water bottle
put the apple back on the far counter
bring me something to quench my thirst
bring me a fruit
bring me a bag of chips from close counter
pick up the 7up and bring it to me
pick up the water bottle and move it to the trash
pick up the apple and move it to the far counter
Please, move the pepsi to the close counter
Would you throw away the bag of chips for me
Redbull is my favorite drink, can I have one please
Can you throw away the apple and bring me a coke
How would you bring me an apple, a coke, and water bottle
I just worked out, can you bring me a drink and a snack to
recover?
Please, move the ppsi to the close cuontner
Would you throw away the bag of chpis for me

```

Listing 9: Task List used in experiment, the scene set up is an office kitchen full of objects, plus testing objects: multigrain chip, basket, plant, sink, apple, first aid station, coke, sink, rice chip bag, coffee machine, water bottle, paper cup, lime sparkling water(green), yellow sign, snack jars of nuts, snack jar of dried fruits, snack jar of gums, snack jar of candy, mug, water fountain machine, tv, tea bottle, box of tea, energy cup, paper bowl, clipboard, compost bin.

```

Put the brown multigrain chip in the woven basket
Water the potted plant
Wash the apple
Move a can of soda to the first aid station
Put the red can in the sink
Put the green chip bag in front of the coffee machine
Help me put bottled water near the paper cup
Compost the apple
Show me where is the first aid station
Put the green can besides the yellow sign
Go to the snack jar of nuts
Microwave the apple
Moisterize the green plant
I want to watch TV, can you get a bottle of tea and put it
there
Place the plastic bottle next to the snack jar
Put the chip bag in front of the box of tea
Move the apple to the plant and water the plant
Find a disposable cup
Please help me put the energy bar in the paper bowl
Can you help me find where my clipboard is

```

Listing 10: Task List used in experiment. The scene setup is an office kitchen full of other common objects but without coke, green can, protein bar, green chip bag, bottle of tea, apple, coke can, multigrain chip.

```

Put the brown multigrain chip in the woven basket
Water the potted plant
Wash the apple
Move a can of soda to the first aid station
Put the red can in the sink
Put the green chip bag in front of the coffee machine
Help me put bottled water near the paper cup
Compost the apple
Put the green can besides the yellow sign
Microwave the apple
Moisterize the green plant
I want to watch TV, can you get a bottle of tea and put it
there
Put the chip bag in front of the box of tea
Move the apple to the plant and water the plant
Please help me put the energy bar in the paper bowl

```

#### F. Additional qualitative experiment results

We show additional qualitative experiment results in Fig. 7, Fig. 8 and Fig. 9.

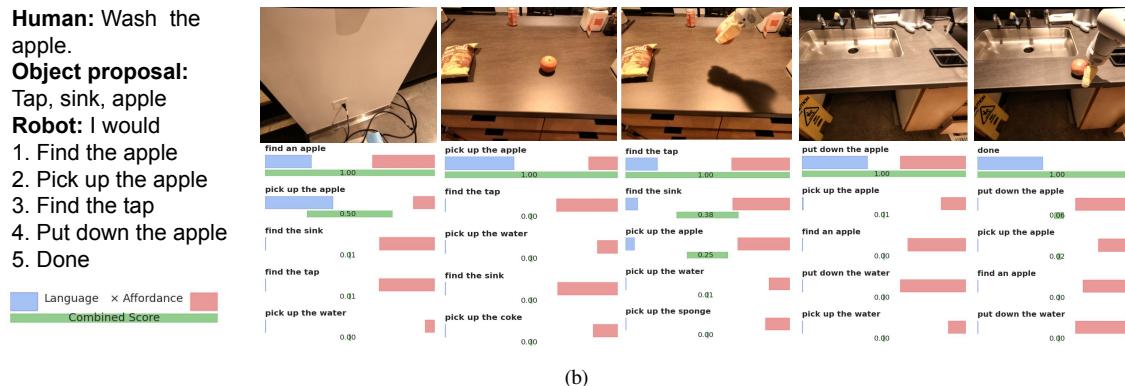
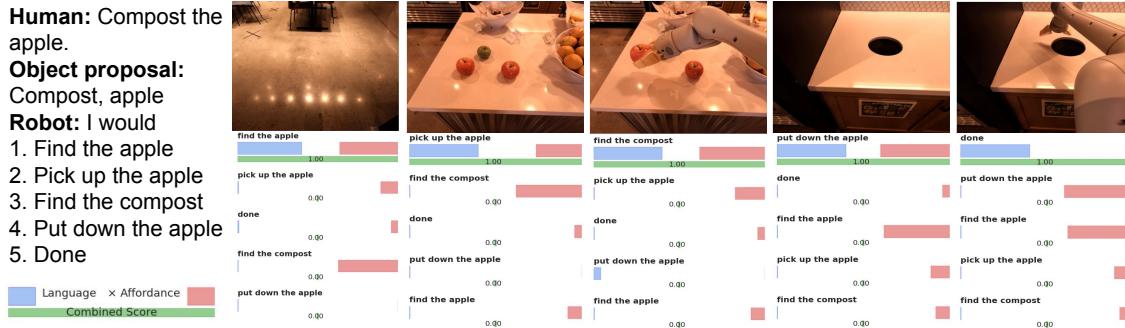


Fig. 7: Examples execution of NLMap + SayCan on two tasks. (a) The robot is able to find apple and compost, based on CLIP and ViLD features, and plan a sequence to put the apple in the compost. (b) Based on the instruction “Wash the apple”, the robot is able to bring the apple to the tap. Turning on the tap is out of scope of this project and left for future work. The images are from the onboard camera of a robot from Everyday Robots.

**Human:** Move the apple to the plant and water the plant

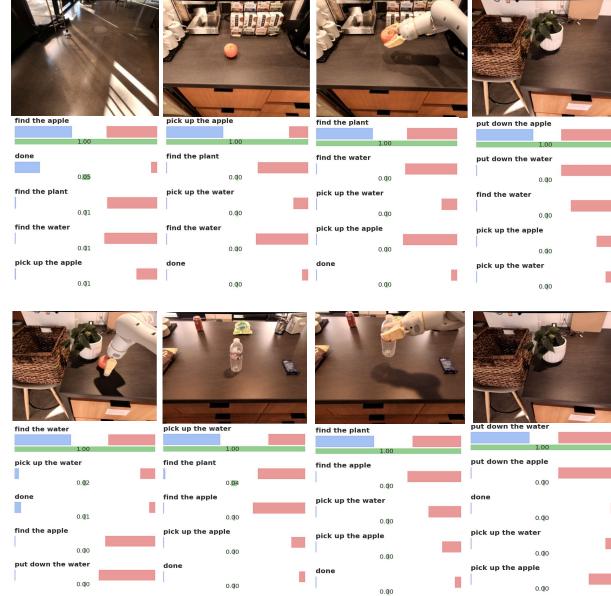
**Object proposal:**

Apple, water, plant

**Robot:** I would

1. Find the apple
2. Pick up the apple
3. Find the plant
4. Put down the apple
5. Find the water
6. Pick up the water
7. Find the plant
8. Put down the water
9. Done

Language x Affordance  
Combined Score



(a)

**Human:** Can you throw away the apple and bring me a coke

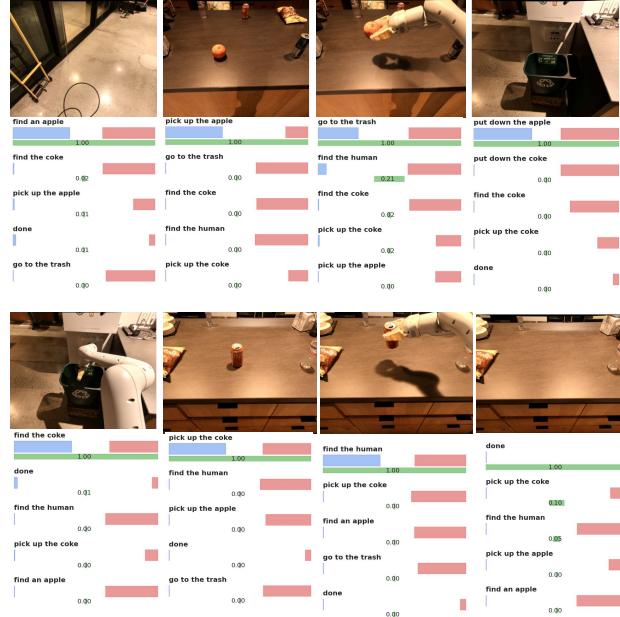
**Object proposal:**

Apple, trash, coke, human

**Robot:** I would

1. Find the apple
2. Pick up the apple
3. Find the trash
4. Put down the apple
5. Find the coke
6. Pick up the coke
7. Find the human
8. Done

Language x Affordance  
Combined Score



(b)

Fig. 8: Examples execution of NLMap + SayCan on long horizon tasks. The proposed method is able to propose objects and plan a sequence for long horizon tasks ((a) has 9 steps and (b) has 8 steps). The images are from the onboard camera of a robot from Everyday Robots.

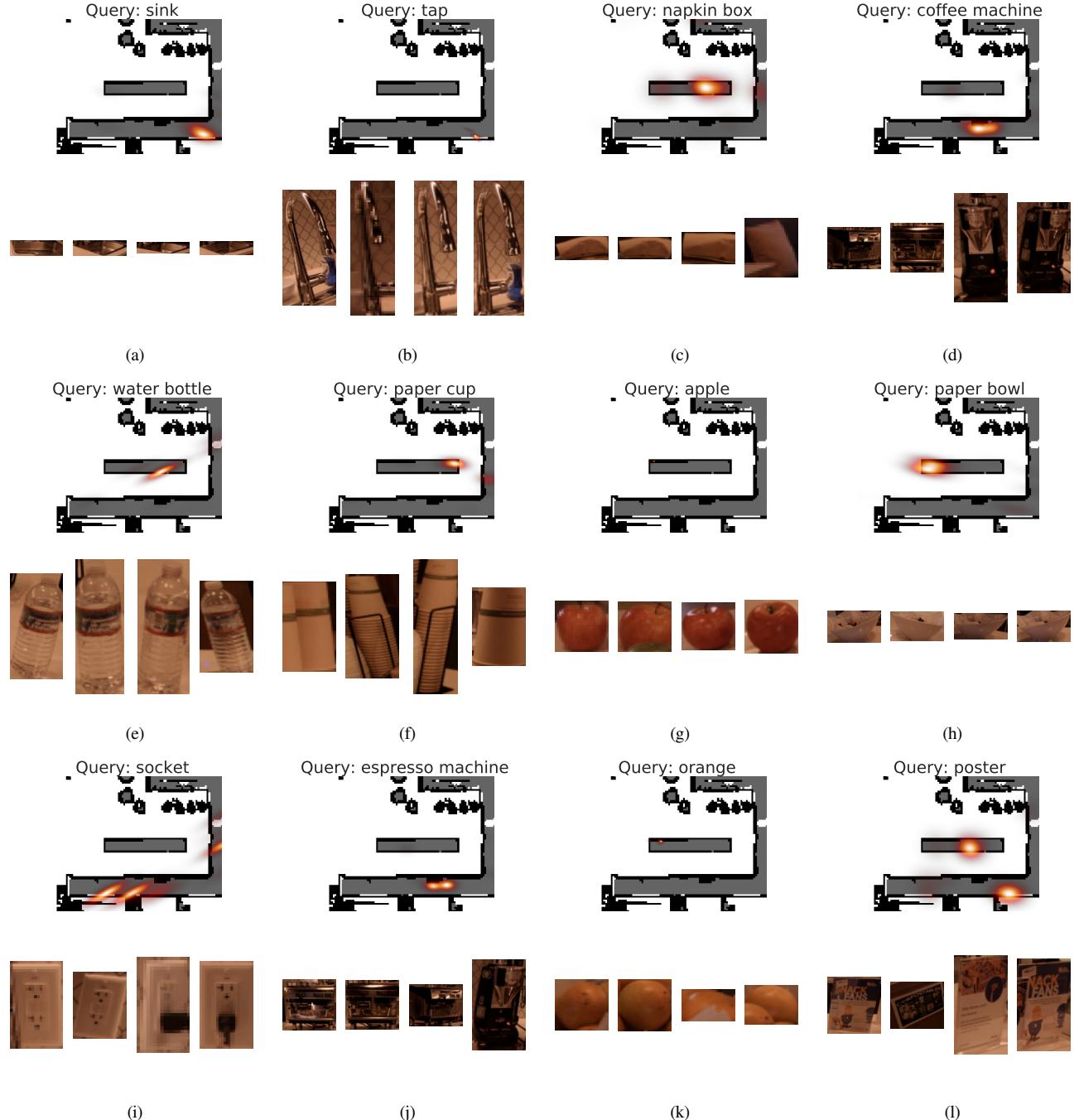


Fig. 9: **More examples of NLMap queries.** Similar to the visualization in Fig. 2, We show the query word as title of each plot, and the heatmap of matches overlaid on the map. The top 4 ROI matches are shown below the map.