Autonomous Robots (2023) 47:1087–1102
https://doi.org/10.1007/s10514-023-10139-z

TidyBot: personalized robot assistance with large language models

Jimmy Wu[1] · Rika Antonova[2] · Adam Kan[3] · Marion Lepert[2] · Andy Zeng[4] · Shuran Song[5] · Jeannette Bohg[2] · Szymon Rusinkiewicz[1] · Thomas Funkhouser[1,4]

Received: 2 May 2023 / Accepted: 25 August 2023 / Published online: 16 November 2023

Abstract
For a robot to personalize physical assistance effectively, it must learn user preferences that can be generally reapplied to future scenarios. In this work, we investigate personalization of household cleanup with robots that can tidy up rooms by picking up objects and putting them away. A key challenge is determining the proper place to put each object, as people's preferences can vary greatly depending on personal taste or cultural background. For instance, one person may prefer storing shirts in the drawer, while another may prefer them on the shelf. We aim to build systems that can learn such preferences from just a handful of examples via prior interactions with a particular person. We show that robots can combine language-based planning and perception with the few-shot summarization capabilities of large language models to infer generalized user preferences that are broadly applicable to future interactions. This approach enables fast adaptation and achieves 91.2% accuracy on unseen objects in our benchmark dataset. We also demonstrate our approach on a real-world mobile manipulator called TidyBot, which successfully puts away 85.0% of objects in real-world test scenarios.

Keywords Service robotics · Mobile manipulation · Large language models

✉ Jimmy Wu
jw60@cs.princeton.edu

Rika Antonova
rika.antonova@stanford.edu

Adam Kan
adakan@nuevaschool.org

Marion Lepert
lepertm@stanford.edu

Andy Zeng
andyzeng@google.com

Shuran Song
shurans@cs.columbia.edu

Jeannette Bohg
bohg@stanford.edu

Szymon Rusinkiewicz
smr@princeton.edu

Thomas Funkhouser
funk@cs.princeton.edu

[1] Princeton University, Princeton, NJ, USA

[2] Stanford University, Stanford, CA, USA

[3] The Nueva School, San Mateo, CA, USA

[4] Google, Mountain View, CA, USA

[5] Columbia University, New York, NY, USA

1 Introduction

Building a robot that provides personalized assistance for physical household tasks is a long-standing goal of robotics research. In this paper, we investigate the task of tidying up a room: moving every object on the floor to its "proper place." One of the challenges in performing this task is determining the correct receptacle ("proper place") for every object. This is difficult because where objects should go is highly personal, and depends on cultural norms and individual preferences. One person may want to put shirts in a dresser drawer, another may want them on shelves, and a third may want them hanging in a closet. There is no "one size fits all" solution.

Classical approaches to the household cleanup task ask a person to specify a target location for every object (Rasch et al., 2019; Yan et al., 2021), which is tedious and impractical in an autonomous setting. Other works learn generic (non-personalized) rules about where objects typically go inside a house by averaging over many users (Kant et al., 2022; Sarch et al., 2022; Taniguchi et al., 2021). Works that focus on personalization aim to extrapolate from a few user examples given similar choices made by other users, using methods such as collaborative filtering (Abdo et al., 2015), spatial relationships (Kang et al., 2018), or learned latent

preference vectors (Kapelyukh & Johns, 2022). However, all of these approaches require collecting large datasets with user preferences or generating datasets from manually constructed, simulated scenarios. Such datasets can be expensive to acquire and may not generalize well if they are too small.

Our approach is to utilize the summarization capabilities of large language models (LLMs) to provide generalization from a small number of example preferences. We ask a person to provide a few example object placements using textual input (e.g., yellow shirts go in the drawer, dark purple shirts go in the closet, white socks go in the drawer), and then we ask the LLM to summarize these examples (e.g., light-colored clothes go in the drawer and dark-colored clothes go in the closet) to arrive at generalized preferences for this particular person.

The underlying insight is that the summarization capabilities of LLMs are a good match for the generalization requirements of personalized robotics. LLMs demonstrate astonishing abilities to perform generalization through summarization, drawing upon complex object properties and relationships learned from massive text datasets. By using the summarization provided by LLMs for generalization in robotics, we hope to produce generalized rules from a small number of examples, in a form that is human interpretable (text) and is expressed in nouns that can be grounded in images using open-vocabulary image classifiers. Using an off-the-shelf LLM also avoids expensive collection of user preference data and model training.

We investigate the proposed approach in a real-world robotic mobile manipulation system for household cleanup, which we call TidyBot (Fig. 1). Before the robot begins cleanup, we ask the user to provide a handful of example placements for specific objects, which are passed to an LLM to be summarized into a generalized set of rules (personalized to that user) mapping object categories to receptacles. The nouns of these generalized rules are provided to an open-vocabulary image classifier in order to identify objects on the floor and determine target receptacles for them using the rules. The robot will then carry out the cleanup task by repeatedly picking up objects, identifying them, and moving them to their target receptacles.

We evaluate our approach quantitatively on both a text-based benchmark dataset and our real-world robotic system. On the benchmark, we find that our approach generalizes well, achieving an accuracy of 91.2% on unseen objects across all scenarios in the benchmark. In our real-world test scenarios, we find that TidyBot correctly puts away 85.0% of objects. We also show that our approach can be easily extended to infer generalized rules for manipulation primitive selection (e.g., pick and place vs. pick and toss) in addition to inferring object placements.

Our contributions are: (i) the idea that text summarization with LLMs provides a means for generalization in robotics, (ii) a publicly released benchmark dataset for evaluating generalization of receptacle selection preferences, and (iii) implementation and evaluation of our approach on a real-world mobile manipulation system.

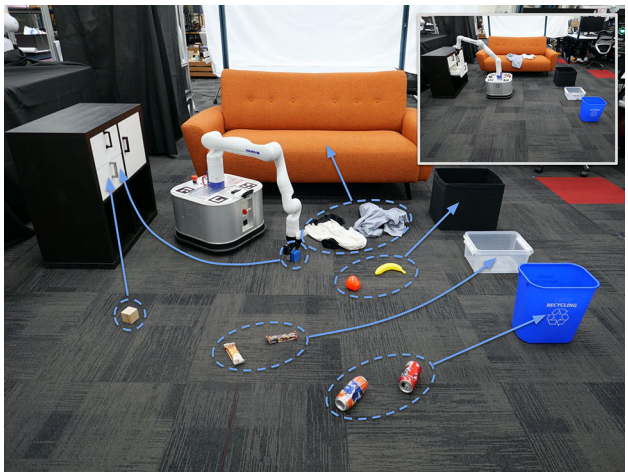

Fig. 1 We study the task of household cleanup, where each object on the floor must be picked up and put away while following user preferences

This journal paper is an extended version of a previously published conference paper (Wu et al., 2023). The new material in this journal version includes:

1. A user study that evaluates whether humans prefer the preferences learned by our approach, and whether human responses align with our benchmark's ground truth
2. Quantitative analysis of the perception component of the real-world system, including comparisons of different visual language models
3. Additional statistics of our benchmark showing representation of different sorting criteria in the dataset, along with a breakdown of baseline results according to these criteria
4. A summary of the limitations of our system

Please see our project page at https://tidybot.cs.princeton.edu for additional supplementary material, benchmark dataset and code, and qualitative videos of our real-world system TidyBot in action.

2 Related work

2.1 Household cleanup

Many recent works in Embodied AI have proposed benchmarks or methods for completing household tasks in simulated indoor environments (Kolve et al., 2017; Li et al., 2022; Puig et al., 2018; Shridhar et al., 2020, 2021; Szot et

al., 2021; Srivastava et al., 2022). For household cleanup in particular, the object rearrangement task (Batra et al., 2020; Ehsani et al., 2021; Gan et al., 2022; Puig et al., 2018; Szot et al., 2021; Weihs et al., 2021) requires an embodied agent to pick up and move objects so as to bring the environment into a specified state. Household cleanup has also been studied in robotics works, in which instructions for object rearrangement are specified via pointing gestures (Rasch et al., 2019) or target layouts (Yan et al., 2021). The drawback of these setups is that a target location must be manually specified for every object to be manipulated, which can require significant human effort. Prior works have addressed this challenge by automatically inferring object placements based on human preferences for where objects typically go inside a house (Kant et al., 2022; Taniguchi et al., 2021; Sarch et al., 2022), eliminating the need to specify where every individual object goes. However, these works predict human preferences that are generic rather than personalized. To handle the variability in preferences across different users, other works have used collaborative filtering (Abdo et al., 2015), spatial relationships (Kang et al., 2018), or learned latent preference vectors (Kapelyukh & Johns, 2022) to predict object placements that are based on personalized user preferences. These methods require the collection of large crowd-sourced datasets for human preferences, which can be expensive. By contrast, our approach uses off-the-shelf LLMs with no additional training or data collection. We are able to directly leverage the commonsense knowledge and summarization abilities of LLMs to build generalizable personalized preferences for each user.

2.2 Object sorting

Object sorting has been studied in robotics using approaches such as clustering (Gupta & Sukhatme, 2012), active learning (Herde et al., 2018; Kujala et al., 2016), metric learning (Zeng et al., 2022), or heuristic search (Huang et al., 2019; Pan & Hauser, 2021; Song et al., 2020). These setups carry out pre-specified sorting rules using physical properties such as color (Dewi et al., 2020; Gupta & Sukhatme, 2012; Herde et al., 2018; Huang et al., 2019; Kujala et al., 2016; Pan & Hauser, 2021; Szabo & Lie, 2012; Song et al., 2020), shape (Herde et al., 2018), size (Dewi et al., 2020; Gupta & Sukhatme, 2012; Herde et al., 2018), or material (Lukka et al., 2014). Notably, they are not able to sort based on semantics or commonsense knowledge, nor are they able to automatically infer sorting rules. More recently, Høeg and Tingelstad (2022) studied whether classification of objects into general high-level categories can be improved by using an LLM to take in an object detector's prediction and output a general category for the object. In our work, we similarly tap into the commonsense knowledge of LLMs to reason about object sorting. However, whereas their setup uses pre-specified sorting rules based on a fixed set of categories, ours is able to infer generalizable sorting rules automatically.

2.3 LLMs for robotics

Large language models (LLMs) have been shown to exhibit remarkable commonsense reasoning abilities (Brown et al., 2020; Kojima et al., 2022; Madaan et al., 2022; Nye et al., 2021; Rytting & Wingate, 2021; Wei et al., 2022a, b). As a result, there has been increasing interest in harnessing the capabilities of LLMs to build more commonsense knowledge into robotic systems. Many recent works study how LLM-generated high-level robotic plans (typically produced using the few-shot learning paradigm (Brown et al., 2020)) can be grounded in the state of the environment. This can be done with value functions (Brohan et al., 2022; Lin et al., 2023), semantic translation into admissible actions (Huang et al., 2022), scene description as context (Chen et al., 2022; Mees et al., 2022; Singh et al., 2022; Zeng et al., 2022), feedback (Huang et al., 2022; Yao et al., 2022), or re-prompting (Raman et al., 2022). However, these works assume a setup in which the LLM is expected to output a single generic plan. This is not a good fit for personalized household cleanup, because a "one size fits all" plan would not address the wide variability in user preferences. Instead, our system generates personalized plans that are tailored to the preferences of a particular user. Other works in robotics have used LLMs for PDDL planning (Silver et al., 2022), code generation for robotic control policies (Liang et al., 2022), parsing navigation instructions into textual landmarks (Shah et al., 2022), room classification (Chen et al., 2022), and tool manipulation (Ren et al., 2022). These works all use LLMs as a means of integrating commonsense knowledge into robotic systems, which is also true in our case. However, unlike these works, we additionally show that the summarization ability of LLMs enables generalization in robotics.

3 Method

We use the summarization capabilities of an off-the-shelf LLM to generalize user preferences from a small number of examples. Below, we describe how we use the LLM to infer personalized rules for both receptacle selection and manipulation primitive selection, and also how we deploy the approach on a real-world mobile manipulation system for household cleanup.

3.1 Personalized receptacle selection

Our system first receives a few examples of object placements reflecting the personal preferences of a user. For instance, the

user may specify that yellow shirts and white socks go in the drawer, while dark purple shirts and black shirts go in the closet. We provide these examples to an LLM, which then infers personalized rules on where objects belong. Specifically, the LLM (i) summarizes the examples into general rules, and then (ii) uses the summary to determine where to place new objects.

Following recent work (Singh et al., 2022; Zeng et al., 2022)), we convert the user examples into LLM prompts that are structured as Pythonic code. This prompt form is advantageous because LLMs are trained on large amounts of code, and it also provides a structured output that is easy to parse. To represent the user examples, the prompt first contains a list of objects present in the scene and a list of potential receptacles (see Appendix A for full prompt with in-context examples). This is followed by a series of pick and place commands reflecting where the objects should be placed according to the user. Then, we ask the LLM to complete the last line, which is a code comment summarizing what the preceding code block does. Here is an example LLM completion where the output from the LLM is highlighted:

```
objects = ["yellow shirt", "dark purple
shirt", "white socks", "black shirt"]
receptacles = ["drawer", "closet"]
pick_and_place("yellow shirt", "drawer")
pick_and_place("dark purple shirt", "closet")
pick_and_place("white socks", "drawer")
pick_and_place("black shirt", "closet")
# Summary: Put light-colored clothes in the
drawer and dark-colored clothes in the closet.
```

In this example, the LLM summarized the provided object placements and inferred that light-colored clothes go in the drawer while dark-colored clothes go in the closet. These examples lead to a generalized rule for where objects belong, personalized to this particular user.

Next, the summary is used by the LLM to generate placements for novel, unseen objects. The prompt consists of the summary from the LLM summarization step (in the form of a code comment), a list of the unseen objects, a list of receptacles, and a partial pick and place command for the first object. We then ask the LLM to provide a placement for each object by completing the prompt:

```
# Summary: Put light-colored clothes in the
drawer and dark-colored clothes in the closet.
objects = ["black socks", "white shirt", "navy
socks", "beige shirt"]
receptacles = ["drawer", "closet"]
pick_and_place("black socks", "closet")
pick_and_place("white shirt", "drawer")
pick_and_place("navy socks", "closet")
pick_and_place("beige shirt", "drawer")
```

The output pick and place commands can then be parsed to determine where each unseen object should be placed.

3.2 Personalized primitive selection

Similar to the way we infer generalized rules for receptacle selection, we can also infer generalized rules for how to manipulate objects, again leveraging the summarization capabilities of LLMs. First, we provide a few examples of objects along with their user-preferred manipulation primitive to the LLM, and ask it to summarize. Here is an example completion where the output from the LLM is highlighted:

```
objects = ["yellow shirt", "dark purple
shirt", "white socks", "black shirt"]
pick_and_place("yellow shirt")
pick_and_place("dark purple shirt")
pick_and_toss("white socks")
pick_and_place("black shirt")
# Summary: Pick and place shirts, pick and
toss socks.
```

The summary can then be used as a generalized rule to predict the appropriate primitive to use for unseen objects:

```
# Summary: Pick and place shirts, pick and
toss socks.
objects = ["black socks", "white shirt", "navy
socks", "beige shirt"]
pick_and_toss("black socks")
pick_and_place("white shirt")
pick_and_toss("navy socks")
pick_and_place("beige shirt")
```

3.3 Real-world robotic system

Given generalized rules from LLM summarization, we can now implement these rules on a robot tasked with tidying up a household environment. To do so, we use a perception system to localize and recognize objects in the environment, and a predetermined set of manipulation primitives to move objects into receptacles. For our setup, we use `pick_and_place` and `pick_and_toss` as our primitives, as they are well-suited for household cleanup. However, other sets of primitives could also be used.

For each new user, the system will receive a set of example preferences and run the previously described LLM pipeline to get personalized rules for the user. The rules contain a set of generalized object categories produced by summarization (e.g., light-colored clothes, dark-colored clothes), each of which is matched to a preferred receptacle and manipulation primitive for that category. The robot will tidy up the environment by iteratively performing the following steps until no more objects remain on the floor: (1) localize the nearest object, (2) classify the object into a generalized category, (3) determine the appropriate receptacle and manipulation primitive for the object using generalized rules produced by the LLM, and (4) use the manipulation primitive to put the object into the receptacle. Figure 2 provides a conceptual illustra-

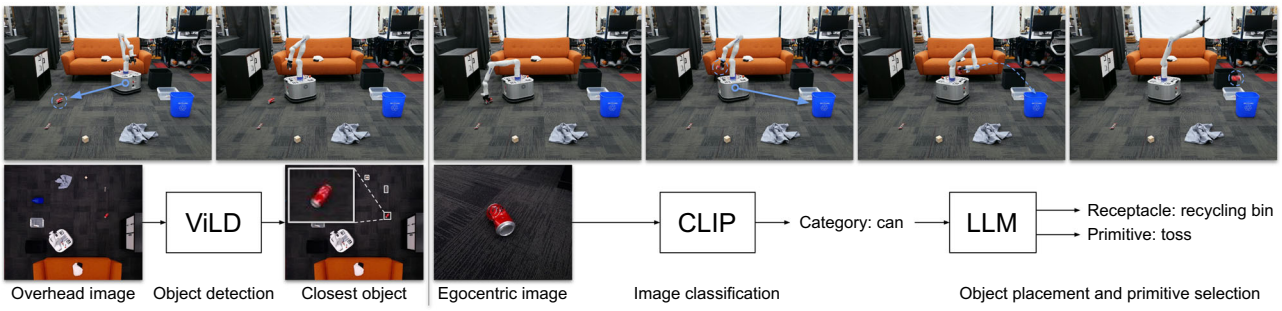

Fig. 2 System overview. Once the user's preferences have been summarized with an LLM, TidyBot will localize the closest object on the floor, move to get a close-up view with its egocentric camera, predict the object's category using CLIP, use the LLM-summarized rules to select a receptacle and manipulation primitive, and then execute the primitive to put the object into the selected receptacle, repeating this entire process until no more objects can be found on the floor

tion of this procedure, and Algorithm 1 outlines these steps in pseudocode.

Algorithm 1 System pipeline

Input: $E_{\text{receptacle}} = \{(o_1, r_1), (o_2, r_2), \ldots\}$
Input: $E_{\text{primitive}} = \{(o_1, p_1), (o_2, p_2), \ldots\}$
$S_{\text{receptacle}} = LLM.\text{Summarize}(E_{\text{receptacle}})$
$S_{\text{primitive}} = LLM.\text{Summarize}(E_{\text{primitive}})$
$C = LLM.\text{GetCategories}(S_{\text{receptacle}})$
robot.Initialize()
while True **do**
  I_{top} = GetOverheadImage()
  $o = ViLD.\text{GetClosestObject}(I_{\text{top}})$
  robot.MoveTo(o)
  $I_{\text{ego}} = robot.\text{GetEgocentricImage}()$
  $c = CLIP.\text{GetCategory}(I_{\text{ego}}, C)$
  $r = LLM.\text{GetReceptacle}(S_{\text{receptacle}}, c)$
  $p = LLM.\text{GetPrimitive}(S_{\text{primitive}}, c)$
  robot.PickUp(o)
  robot.MoveTo(r)
  robot.ExecutePrimitive(p)
end while

One important aspect of our approach is that the LLM summarization automatically provides candidate categories to the perception system. Nouns (or noun phrases) are extracted from the summarization text as categories, and used as the target label set for CLIP (Radford et al., 2021), the open-vocabulary image classification model we use. For example, the following LLM prompt will extract the two general categories in the summary text (light-colored clothing and dark-colored clothing):

```
# Summary: Put light-colored clothes in the
drawer and dark-colored clothes in the closet.
objects = ["light-colored clothing",
"dark-colored clothing"]
```

This combination of summarization and open-vocabulary classification is critical to the autonomy of the system, as it enables the object classifier to work with a small set of generalized object categories. The approach is (i) robust as there are only a small number of categories to differentiate between, and (ii) flexible because it supports arbitrary sets of object categories for different users. In contrast, without LLM summarization, the object classifier would have to be able to recognize all possible fine-grained object classes, which is much more difficult. Alternatively, the user would have to manually specify the list of objects present in each target scene, which would be impractical for an autonomous system.

4 Experiments

We investigate the performance of our proposed approach with two types of evaluation. For the first type of evaluation, we design a benchmark for generalization of receptacle selection using text-based examples, which enables direct comparison to alternative approaches and ablation studies, with quantitative metrics. For the second type of evaluation, we deploy our approach in a real-world mobile manipulation system for tidying up a room based on user preferences. Unless otherwise specified, the LLM we use is `text-davinci-003`, a variant of GPT-3 (Brown et al., 2020). All LLM experiments were run with temperature 0.

4.1 Benchmark dataset

In order to evaluate the proposed approach and to quantitatively compare it to alternatives, we created a benchmark dataset of object placements. The benchmark is comprised of 96 scenarios, each of which has a set of objects, a set of receptacles, a set of example "seen" object placements (preferences), and a set of "unseen" evaluation placements, all specified as text. The task is to predict the placements in the "unseen" set given the examples in the "seen" set.

Table 1 Representation of sorting criteria in benchmark

Category	Attribute	Function	Subcategory	Multiple
86/96	27/96	24/96	31/96	17/96

The benchmark scenarios are defined in 4 room types (living room, bedroom, kitchen, pantry room), with 24 scenarios per room type. Each scenario contains 2–5 receptacles (potential places to put objects, such as shelves, cabinets, etc.), 4–10 "seen" example object placements provided as input to the task, and an equal number of "unseen" object placements (distinct from the seen examples) provided for evaluation. There are 2 seen and 2 unseen object placements per receptacle. In total, there are 672 seen and 672 unseen object placements, which cumulatively reference 87 unique receptacles and 1076 unique objects.

Success on this benchmark is measured by the object placement accuracy: the number of objects placed in the correct receptacle divided by the total number of objects. We evaluate accuracy separately for seen and unseen objects, to tease apart memorization versus generalization. For each, we compute the accuracy per scenario, and then average the results across all scenarios to produce the numbers shown in the tables.

Since different people may sort items in the home in many different ways, our benchmark contains a diversity of preferences with several kinds of sorting criteria represented in the dataset:

- *Category* Sort objects based on general categories (e.g., put clothes here and toys there)
- *Attribute* Sort objects based on object attributes (e.g., put plastic items here and metal items there)
- *Function* Sort objects based on function (e.g., put winter clothes here and summer clothes there)
- *Subcategory* Sort objects such that a specific (subordinate) subcategory is separated from the general (superordinate) category (e.g., put shirts on the sofa and other clothes in the closet)
- *Multiple categories* Sort objects from multiple categories into one receptacle (e.g., put both books and toys on the shelf)

We show in Table 1 the representation of different sorting criteria in our benchmark dataset, indicated by the fraction of the 96 scenarios to which each criteria applies. Note that multiple sorting criteria may apply to a single scenario.

4.2 Baseline comparisons

In our first set of experiments, we use the benchmark to provide quantitative evaluation of our approach compared to several alternatives. The results are in Table 2. We also show in Table 3 the same results but broken down by the sorting criteria described in Sect. 4.1. Since the main challenge is to generalize from objects in the examples (seen) to those in the evaluation set (unseen), we consider a variety of baseline generalization approaches and report placement accuracy metrics only for unseen objects.

Table 2 Comparisons to baselines

Method	Accuracy (unseen) (%)
Examples only	78.5
WordNet taxonomy	67.5
RoBERTa embeddings	77.8
CLIP embeddings	83.7
Summarization (ours)	**91.2**

The following paragraphs describe each baseline and provide a discussion of how the performance compares to that of our proposed approach.

Examples only The first baseline provides a direct comparison to a system like ours if it did not use summarization. The LLM is given a list of objects, receptacles, and example placement preferences, along with a list of unseen objects for a new scene. Then, the LLM is asked to directly infer the proper placements (highlighted text) for unseen objects in the new scene, without summarization as an intermediate step:

```
objects = ["yellow shirt", "dark purple
shirt", "white socks", "black shirt"]
receptacles = ["drawer", "closet"]
pick_and_place("yellow shirt", "drawer")
pick_and_place("dark purple shirt", "closet")
pick_and_place("white socks", "drawer")
pick_and_place("black shirt", "closet")

objects = ["black socks", "white shirt", "navy
socks", "beige shirt"]
receptacles = ["drawer", "closet"]
pick_and_place("black socks", "drawer")
pick_and_place("white shirt", "closet")
pick_and_place("navy socks", "drawer")
pick_and_place("beige shirt", "closet")
```

The prediction accuracy of this method for unseen objects (78.5%) is significantly worse than that of our method (91.2%). Since the main difference between this method versus ours is that our method leverages summarization, this result presents strong evidence for our main hypothesis—i.e., summarization is useful for generalization. This finding is also consistent with recent work showing that LLMs perform better when they are asked to output intermediate steps of reasoning before the final answer (Nye et al., 2021; Wei et al., 2022a). When looking at the predictions, we find that this baseline approach generally predicts object placements

Table 3 Comparisons to baselines by sorting criteria

Method	Category (%)	Attribute (%)	Function (%)	Subcategory (%)	Multiple (%)
Examples only	80.1	72.7	75.7	77.0	81.5
WordNet taxonomy	69.1	59.8	61.4	71.3	74.1
RoBERTa embeddings	78.6	75.5	71.8	71.7	87.5
CLIP embeddings	84.6	79.8	85.5	84.7	87.9
Summarization (ours)	**91.0**	**85.6**	**93.9**	**90.1**	**93.5**

that are sensible but may not be consistent with the user's preferences.

WordNet taxonomy This baseline uses a hand-crafted lexical ontology called WordNet (Miller, 1995) to generalize placements from seen to unseen objects. For each unseen object, we place it in the same receptacle as the most similar seen object, where similarity is measured using the shortest path between two objects in the taxonomy. Since WordNet is a hand-crafted taxonomy, it does not contain all possible object names. For the 694 objects in our benchmark that are missing from WordNet, we manually mapped each of them to the closest WordNet object name. Even with the manual mapping, the performance of this WordNet baseline for unseen objects (67.5%) is far worse than that of our method (91.2%). This shows that LLM summarization provides better generalization than using the hierarchy provided by a hand-crafted ontology. When looking at the breakdown in Table 3, we see that this baseline performs worse on the two criteria that are not related to object categorization (attribute and function). We hypothesize that WordNet is not able to generalize well along these dimensions because it was constructed mainly based on semantic relationships between categories.

Text embedding This baseline uses pretrained text embeddings to assist with generalization. For each unseen object, we place it in the receptacle provided for the most similar seen object, where similarity is defined by cosine similarity between encoded object names in the RoBERTa (Liu et al., 2019) or CLIP (Radford et al., 2021) embedding space. For RoBERTa, we use the pretrained Sentence-BERT (Reimers & Gurevych, 2019) model from the SentenceTransformers library. Specifically, we use the `all-distilroberta-v1` variant which is a distilled (Sanh et al., 2019) version of the RoBERTa (Liu et al., 2019) model that is fine-tuned on a dataset of 1 billion sentence pairs. For CLIP, we use the pretrained model provided by OpenAI. In either case, the generalization performance for predicting placements of unseen objects does not reach the performance of our proposed summarization approach (77.8% for RoBERTa and 83.7% for CLIP, versus 91.2% for ours). Although text embeddings trained on large datasets encode many types of object similarities, particularly for related object categories, they may not encode the object attributes relevant to the preferences of a particular user (e.g., light objects go here, heavy object go there). In contrast, our summarization approach is able to correctly encode a larger variety of user preferences.

Table 4 Ablation studies

Method	Seen (%)	Unseen (%)
Commonsense	45.0	45.6
Summarization	91.8	91.2
Human summary	97.1	97.5

4.3 Ablation studies

In the second set of experiments, we use the benchmark to evaluate the performance of several variants to our method. The goal of these experiments is to compare its performance to alternatives with far less information (using only common sense, without preferences) or far more information (using human-generated summarizations). We also study the impact of using different LLMs. The benchmark metrics for both seen and unseen objects are provided in Tables. 4 and 5.

Commonsense Our first ablation study measures how well an LLM can perform the benchmark tasks using only commonsense reasoning—i.e., without using the preferences at all. For each benchmark scene, we give the LLM the list of objects and list of receptacles, and then ask it to generate object placements (highlighted text) without using the provided user preferences:

```
# Put objects into their appropriate
receptacles.
objects = ["black socks", "white shirt", "navy
socks", "beige shirt"]
receptacles = ["drawer", "closet"]
pick_and_place("black socks", "drawer")
pick_and_place("white shirt", "closet")
pick_and_place("navy socks", "drawer")
pick_and_place("beige shirt", "closet")
```

This baseline performs poorly, even for seen objects (45.0%), due to the high variability of object placement preferences in the benchmark. The predicted object placements are sensible but are not reflective of the particular user's preferences. In contrast, our method can learn preferences from examples via summarization and performs much better for both seen and unseen objects (91.8% and 91.2%).

Table 5 Comparison of different LLMs

Model	Commonsense		Summarization	
	Seen (%)	Unseen (%)	Seen (%)	Unseen (%)
text-davinci-003	45.0	45.6	**91.8**	**91.2**
text-davinci-002	41.8	37.5	84.1	75.7
code-davinci-002	41.4	39.4	88.6	83.2
PaLM 540B	**45.5**	**49.6**	84.6	75.7

Human summary This ablation studies how the summaries provided by the LLM compare to summaries crafted manually by a human. For each benchmark scenario, a human-written summary was used by the LLM (in place of the LLM-produced summary) to predict object placements for the test objects. The results achieved with this "oracle" summarization are better than the LLM summarization by 6% for both seen and unseen objects. This result suggests that the LLM summarizations are already quite good, and that improvements to LLM summarization could enable further gains for our method in the future.

Different LLMs Table 5 reports our performance on the benchmark using different LLMs. We find that `text-davinci-002` and `code-davinci-002` (Chen et al., 2021), which are older variants of GPT-3, are not as good as the newest one (`text-davinci-003`). In particular, there is a much larger gap between seen and unseen objects. This is because the older models are more likely to generate summaries that list out individual objects in the seen set, which does not generalize well to the unseen objects. For PaLM 540B (Chowdhery et al., 2022), we find that while it shows slightly higher performance on commonsense reasoning, it does not do as well as `text-davinci-003` on summarization, particularly in scenarios where there is a larger number of receptacles to choose from.

4.4 Human evaluation

To evaluate whether humans prefer the preferences learned by our method, we conduct a user study based on the scenarios in our benchmark dataset. The study asks participants to compare the object placements generated by our method to those of CLIP embeddings, which is the strongest baseline. The study has 2 objectives:

1. Evaluate whether humans prefer the object placements generated by our LLM summarization method over those of the CLIP embeddings baseline
2. Evaluate whether human-preferred object placements align with the ground truth placements in our benchmark

Study setup We recruited 40 participants (24 males and 16 females) consisting of affiliates from author institutions and asked them to fill out an online survey. Each participant was assigned 24 scenarios randomly selected from the 96 scenarios in the benchmark. Each scenario in the benchmark is evaluated by 10 participants, giving 960 evaluations in total.

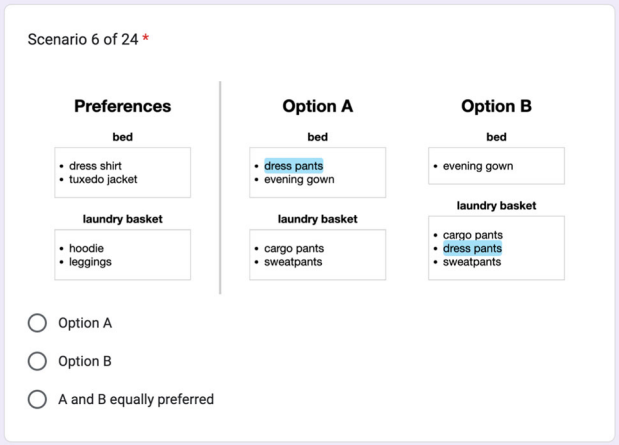

Fig. 3 Example user study question. This screenshot shows an example survey question from our user study. On the left are preferences, on the right are two placement options corresponding to the two methods being compared. The participant is asked to select the option that is best aligned with the given preferences

For each scenario, we provide (i) example placements of "seen" objects indicating user preferences, and (ii) placements of "unseen" objects from both our LLM summarization method and the CLIP embeddings method (example shown in Fig. 3). The participant is then asked to specify which of the two object placement options better aligns with the given preferences, or if they are equally preferable. For the convenience of the participants, we highlight the object placements that differ between the two methods. We randomize the order of scenarios as well as the order of methods for each scenario (the participant is unaware of which option goes with which method). For some of the scenarios, both methods give the exact same object placements, so we preselect the third "equally preferred" option and exclude them from the surveys given to participants.

Results Our results across all 960 evaluations are shown in Table 6. Overall, we find that our LLM summarization method is preferred over the CLIP embeddings baseline 46.9% of the time, whereas the baseline is preferred 19.1% of the time, and both methods are equally preferred 34.1% of

Table 6 User study results by sorting criteria

Method	Category (%)	Attribute (%)	Function (%)	Subcategory (%)	Multiple (%)	Overall (%)
CLIP embeddings	19.7	23.7	11.2	22.6	21.2	19.1
Summarization (ours)	**47.4**	**41.9**	**60.0**	**46.1**	**40.6**	**46.9**
Equally preferred	32.9	34.4	28.8	31.3	38.2	34.1

the time. When considering the results broken down by sorting criteria, we find that our method performs particularly well relative to the baseline for the function criteria (*e.g.*, formal vs. casual clothes). Even though the corresponding benchmark accuracy is relatively high (CLIP embeddings in Table 3), the baseline method usually sorts by object category (as described in Sec. 4.2) which can lead to egregiously wrong placements (e.g. store dress pants with sweatpants) when the intended sorting criteria is function.

We ran a statistical analysis with the following null hypothesis (H0): There is no significant difference between the preference for our method versus the baseline method. In other words, the mean fraction of time participants prefer our method over the baseline is equal to 0.5. For each study participant, we calculated the fraction of time our method was preferred over the baseline method across the 24 scenarios for that participant. For scenarios where both methods were equally preferred, we gave them both equal weight. We then conducted a paired t-test, and found a significant difference between our method and the baseline method, with a calculated t-statistic of 9.93 (df $= 39$), $p < 0.001$, indicating strong evidence to reject the null hypothesis and suggesting that the observed difference in human preference between our method and the baseline is unlikely to have occurred due to random chance.

We also evaluate how well the participant responses align with the ground truth in our benchmark. For each scenario, we identify which of the two methods is closer to the benchmark ground truth based on unseen object placement accuracy on that scenario. We then calculate the percent of human responses that prefer the method that is closer to the ground truth. Overall, across the 40 participants, we find that human responses were aligned with benchmark ground truth 82.2% $\pm$ 7.7% of the time, or 95.4% $\pm$ 4.1% if "equally preferred" is treated as a wildcard.

4.5 Real-world experiments

In our final set of experiments, we test the proposed approach on a robot performing a cleanup task in the real world (Fig. 1). The robot base is a holonomic vehicle capable of any 3-degree-of-freedom motion on the ground plane. This maneuverability comes from the vehicle's Powered-Caster Drive System (Holmberg & Khatib, 2000), which consists of four caster wheels that are powered to roll and steer as needed to achieve the desired vehicle motion. The robot manipulator is a Kinova Gen3 7-DoF arm mounted on top of the mobile base with a Robotiq 2F-85 parallel jaw gripper as its end effector.

The robot is placed inside a room with various objects and receptacles on the floor and is then tasked with picking up all the objects and putting them into the correct receptacles according to user preferences. The preferences are provided as a set of textual examples for a particular user (as in the benchmark). As described in Sect. 3.3 and illustrated in Fig. 2, the robot iteratively locates the closest object on the floor, navigates to it, recognizes its category, picks it up, determines the appropriate receptacle for the object, navigates to the receptacle, and then puts the object inside.

Implementation The robot uses two overhead cameras for 2D robot pose estimation (x, y, θ) and 2D object localization (x, y). The pose of the robot base is estimated using ArUco fiducial markers (Garrido-Jurado et al., 2014) mounted on its top plate (see Fig. 1). The object locations are detected in the overhead camera using ViLD (Gu et al., 2021), while the receptacle locations are hard-coded for each scenario. We found that these design choices work well for our mobile robot system. However, other pose trackers and object detectors could also be used instead.

To navigate in the scene, the robot calculates the shortest collision-free path to the target position using an occupancy map that includes obstacles in the scene such as receptacles. It then uses the pure pursuit algorithm (Coulter, 1992) to follow the computed path.

After the robot arrives at the closest object, it uses a camera mounted on its base (and pointed forward at the ground) to take a close-up, centered image of the object, then determines the object category using cosine similarity between text and image features in the CLIP embedding space (Radford et al., 2021). The set of object categories in the LLM summary is automatically extracted and used as the target label set for CLIP. Note that without these categories from LLM summarization, a human would have to manually specify a list of fine-grained object classes potentially present in the target scene in order to use CLIP for object classification.

After the object category is identified, the system uses the LLM summarization to predict the appropriate receptacle and manipulation primitive for the object. The robot then moves the object into the receptacle with a sequence of two high-level manipulation primitives: (i) pick and (ii) place or

Fig. 4 Real-world scenarios. We evaluate our mobile manipulation system in 8 real-world scenarios, encompassing a wide variety of objects and receptacles

Fig. 5 Real-world objects. 70 unique "unseen" test objects are represented in our real-world scenarios

Fig. 6 Real-world receptacles. 11 unique receptacles are represented in our real-world scenarios

toss. The "pick" primitive uses the gripper to grasp at the center of the detected object. The "place" primitive moves the gripper to a location just above the selected receptacle and drops the grasped object in. The "toss" primitive swings the robot arm and releases the gripper with timing that results in tossing (Zeng et al., 2020) of the grasped object into the selected receptacle.

Real-world evaluation Using this mobile robot system, we ran tests on 8 real-world scenarios as shown in Fig. 4, each with its own set of 10 objects, 2–5 receptacles, 4–10 "seen" examples indicating preferences for which objects should go into which receptacles and which primitive should be used to put them there, as well as 10 "unseen" test objects. Across all 8 scenarios, 70 unique "unseen" test objects (Fig. 5) and 11 unique receptacles (Fig. 6) are represented.

For each scenario, we asked the robot to perform 3 runs of the cleanup task and measured its success throughout operation. Overall, the system was able to put 85.0% of the objects into the correct receptacle during these tests. For qualitative examples, please refer to the supplementary material and additional videos at https://tidybot.cs.princeton.edu.

Looking at the results in more detail, there were 240 objects to be cleaned up in total (8 scenarios, 10 objects per scenario, 3 runs per scenario). We observed that the overhead camera was able to localize 92.5% of the objects, and the object classifier correctly identified the object category for 95.5% of the localized objects. Given the predicted object category, the LLM selected the appropriate receptacle and manipulation primitive for 100% of localized objects. Additionally, the robot succeeded in executing the chosen primitive for 96.2% of the localized objects. In terms of speed, the robot took on average 15–20 s to pick up and put away each object.

Visual language model (VLM) evaluation In this section, we perform a quantitative comparison of different visual language models (VLMs) within our real robot system. Recall that for each object successfully localized by the overhead camera, the real robot will first use its egocentric camera to take a close-up image of the object before picking it up. This image is given to a VLM to determine the category of the object. To conduct our analysis, we save all egocentric images from our real world evaluation (222 in total across all test runs) and annotate them.

To evaluate a VLM, we run all 222 images through the model and determine the fraction of images in which the centered foreground object is correctly recognized. We compare

Table 7 Comparison of different VLMs

	CLIP (%)	ViLD (%)	OWL-ViT (%)
Summarized categories	**95.5**	76.1	45.9
Scenario object names	**70.7**	59.9	24.8
All object names	**52.3**	36.5	18.5

along two axes: (i) model type and (ii) vocabulary used for the target label set. The model types we consider (all open-vocabulary) are (i) CLIP (Radford et al., 2021), which was the image classifier used in our final system, and two alternatives, (ii) ViLD (Gu et al., 2021) and (iii) OWL-ViT (Minderer et al., 2022). The vocabulary options we consider are (i) the set of categories output by LLM summarization (e.g., clothing, fruit,...), which was used in our final system, (ii) a list of human-annotated names for all objects in the current scenario (e.g., blue jeans, apple,...), and (iii) a list of human-annotated object names across all scenarios (instead of just one scenario). Note that the human-annotated options for the vocabulary are for analysis only, as it would be infeasible to ask a human to annotate every object encountered during robot operation. Results are shown in Table 7.

Looking at the results comparing different VLMs (columns of Table 7), we find that CLIP performs the best out of all the models. One reason is that CLIP will always output a prediction, whereas the object detectors (ViLD and OWL-ViT) will sometimes detect no objects in the image. Additionally, ViLD and OWL-ViT are derived from CLIP, and it is possible that the process of adapting the models to localize bounding boxes degrades their performance on object classification.

Qualitatively, the main failure mode of CLIP is reporting the class of an object in the background rather than that of the foreground object. This is expected since CLIP performs an image-wide classification. We also observe that CLIP is often not able to consider noun phrases as complete units. For example, the phrase "white socks" may match strongly with anything that looks white.

For ViLD and OWL-ViT (both object detectors), we use the bounding box closest to the center of the image as the detection, since the egocentric camera is pointed directly at the pick location on the floor. We expected that this localization would improve accuracy since foreground objects can be isolated from background objects (unlike with CLIP). However, we find that quantitatively, both ViLD and OWL-ViT perform worse than CLIP. Qualitatively, ViLD works well with small rigid objects, but struggles with larger deformable objects (such as clothes or stuffed animals), outputting many extraneous detections corresponding to parts of objects. Additionally, for both ViLD and OWL-ViT, we find that the foreground object is sometimes not detected at all, even though it is always prominently placed in the center of the image.

When looking at results for different vocabularies (rows of Table 7), we find that using the categories from the LLM summary performs the best. This is partly because the VLM has to differentiate between a much smaller number of options (2–5 categories vs. 10 or 65 object names). Note again that the use of object names is not actually feasible in a real system due to the human annotation burden. By contrast, our use of LLM summarized categories allows the system to directly generalize to novel objects as the VLM only needs to correctly identify the closest category rather than what the specific object is.

4.6 Limitations

4.6.1 LLM summarization

While LLMs are generally able to summarize preferences well, we find that there are still cases in which the generated summary is not quite right. The most common failure mode is when the generated summary simply lists out the seen objects rather than summarizing into categories. Summaries of that nature are too specific and do not generalize well to unseen objects. Another failure mode is when the LLM summarizes receptacles by grouping them together (e.g., top drawer and bottom drawer might be summarized as drawers), resulting in poor performance when using the summary for receptacle selection.

4.6.2 Real-world system

Our implementation of the real-world system contains simplifications such as the use of hand-written manipulation primitives, use of top-down grasps, and assumption of known receptacle locations. These limitations could be addressed by incorporating more advanced primitives into our system and expanding the capabilities of the perception system. Additionally, since the mobile robots cannot drive over objects, the system would not work well in excessive clutter. It would be interesting to incorporate more advanced high-level planning, so that instead of always picking up the closest object, the robot could reason about whether it needs to first clear itself a path to move through the clutter.

5 Conclusion

In this work, we showed that the summarization capabilities of large language models (LLMs) can be used to generalize user preferences for personalized robotics. Given a handful of example preferences for a particular person, we use LLM summarization to infer a generalized set of rules

to manipulate objects according to the user's preferences. We show that our summarization approach outperforms several strong baselines on our benchmark, and we also evaluate our approach on a real-world mobile manipulator called TidyBot, which can successfully clean up test scenarios with a success rate of 85.0%. Our approach provides a promising direction for developing personalized robotic systems that can learn generalized user preferences quickly and effectively from only a small set of examples. Unlike classical approaches that require costly data collection and model training, we show that LLMs can be directly used off-the-shelf to achieve generalization in robotics, leveraging the powerful summarization capabilities they have learned from vast amounts of text data.

Acknowledgements The authors would like to thank William Chong, Kevin Lin, and Jingyun Yang for fruitful technical discussions, and Bob Holmberg for mentorship and support in building up the mobile platforms.

Author Contributions JW, RA, AK, ML, and AZ contributed to system implementation, experiments, or analysis. RA, AZ, SS, JB, SR, and TF supervised the project. All authors contributed to the manuscript.

Funding This work was supported in part by the Princeton School of Engineering, Toyota Research Institute, and the National Science Foundation under CCF-2030859, DGE-1656466, and IIS-2132519.

Declarations

Conflict of interest The authors have no competing interests to declare that are relevant to the content of this article.

Appendix A: LLM prompts

This section contains the full prompts used for all LLM text completion tasks. Each prompt consists of 1–3 in-context examples in gray followed by a test example that we ask the LLM to complete. The portion of the test example that is generated by the LLM is highlighted. We use the same in-context examples across all scenarios in both the benchmark and the real-world system. For each scenario, only the final test example is modified.

A.1 Summarization for receptacle selection

```
objects = ["dried figs", "protein bar",
"cornmeal", "Macadamia nuts", "vinegar",
"herbal tea", "peanut oil", "chocolate bar",
"bread crumbs", "Folgers instant coffee"]
receptacles = ["top rack", "middle rack",
"table", "shelf", "plastic box"]
pick_and_place("dried figs", "plastic box")
pick_and_place("protein bar", "shelf")
pick_and_place("cornmeal", "top rack")
pick_and_place("Macadamia nuts", "plastic
box")
pick_and_place("vinegar", "middle rack")
pick_and_place("herbal tea", "table")
pick_and_place("peanut oil", "middle rack")
pick_and_place("chocolate bar", "shelf")
pick_and_place("bread crumbs", "top rack")
pick_and_place("Folgers instant coffee",
"table")
# Summary: Put dry ingredients on the top
rack, liquid ingredients in the middle rack,
tea and coffee on the table, packaged snacks
on the shelf, and dried fruits and nuts in the
plastic box.

objects = ["yoga pants", "wool sweater",
"black jeans", "Nike shorts"]
receptacles = ["hamper", "bed"]
pick_and_place("yoga pants", "hamper")
pick_and_place("wool sweater", "bed")
pick_and_place("black jeans", "bed")
pick_and_place("Nike shorts", "hamper")
# Summary: Put athletic clothes in the hamper
and other clothes on the bed.

objects = ["Nike sweatpants", "sweater",
"cargo shorts", "iPhone", "dictionary",
"tablet", "Under Armour t-shirt", "physics
homework"]
receptacles = ["backpack", "closet", "desk",
"nightstand"]
pick_and_place("Nike sweatpants", "backpack")
pick_and_place("sweater", "closet")
pick_and_place("cargo shorts", "closet")
pick_and_place("iPhone", "nightstand")
pick_and_place("dictionary", "desk")
pick_and_place("tablet", "nightstand")
pick_and_place("Under Armour t-shirt",
"backpack")
pick_and_place("physics homework", "desk")
# Summary: Put workout clothes in the
backpack, other clothes in the closet, books
and homeworks on the desk, and electronics on
the nightstand.
```

```
objects = ["jacket", "candy bar", "soda can",
"Pepsi can", "jeans", "wooden block",
"orange", "chips", "wooden block 2", "apple"]
receptacles = ["recycling bin", "plastic
storage box", "black storage box", "sofa",
"drawer"]
pick_and_place("jacket", "sofa")
pick_and_place("candy bar", "plastic storage
box")
pick_and_place("soda can", "recycling bin")
pick_and_place("Pepsi can", "recycling bin")
pick_and_place("jeans", "sofa")
pick_and_place("wooden block", "drawer")
pick_and_place("orange", "black storage box")
pick_and_place("chips", "plastic storage box")
pick_and_place("wooden block 2", "drawer")
pick_and_place("apple", "black storage box")
# Summary: Put clothes on the sofa, snacks in
the plastic storage box, cans in the recycling
bin, wooden blocks in the drawer, and fruits
in the black storage box.
```

A.2 Receptacle selection

```
# Summary: Put clothes in the laundry basket
and toys in the storage box.
objects = ["socks", "toy car", "shirt", "Lego
brick"]
receptacles = ["laundry basket", "storage
box"]
pick_and_place("socks", "laundry basket")
pick_and_place("toy car", "storage box")
pick_and_place("shirt", "laundry basket")
pick_and_place("Lego brick", "storage box")

# Summary: Put clothes on the sofa, snacks in
the plastic storage box, cans in the recycling
bin, wooden blocks in the drawer, and fruits
in the black storage box.
objects = ["jacket", "candy bar", "soda can",
"Pepsi can", "jeans", "wooden block",
"orange", "chips", "wooden block 2", "apple"]
receptacles = ["recycling bin", "plastic
storage box", "black storage box", "sofa",
"drawer"]
pick_and_place("jacket", "sofa")
pick_and_place("candy bar", "plastic storage
box")
pick_and_place("soda can", "recycling bin")
pick_and_place("Pepsi can", "recycling bin")
pick_and_place("jeans", "sofa")
pick_and_place("wooden block", "drawer")
pick_and_place("orange", "black storage box")
pick_and_place("chips", "plastic storage box")
pick_and_place("wooden block 2", "drawer")
pick_and_place("apple", "black storage box")
```

A.3 Summarization for primitive selection

```
objects = ["granola bar", "hat", "toy car",
"Lego brick", "fruit snacks", "shirt"]
pick_and_toss("granola bar")
pick_and_place("hat")
pick_and_place("toy car")
pick_and_place("Lego brick")
pick_and_toss("fruit snacks")
pick_and_place("shirt")
# Summary: Pick and place clothes and toys,
pick and toss snacks.

objects = ["jacket", "candy bar", "soda can",
"Pepsi can", "jeans", "wooden block",
"orange", "chips", "wooden block 2", "apple"]
pick_and_place("jacket")
pick_and_toss("candy bar")
pick_and_toss("soda can")
pick_and_toss("Pepsi can")
pick_and_place("jeans")
pick_and_place("wooden block")
pick_and_toss("orange")
pick_and_toss("chips")
pick_and_place("wooden block 2")
pick_and_toss("apple")
# Summary: Pick and place clothes and wooden
blocks, pick and toss snacks and drinks.
```

A.4 Primitive selection

```
# Summary: Pick and place clothes, pick and
toss snacks.
objects = ["granola bar", "hat", "toy car",
"Lego brick", "fruit snacks", "shirt"]
pick_and_toss("granola bar")
pick_and_place("hat")
pick_and_place("toy car")
pick_and_place("Lego brick")
pick_and_toss("fruit snacks")
pick_and_place("shirt")

# Summary: Pick and place granola bars, hats,
toy cars, and Lego bricks, pick and toss fruit
snacks and shirts.
objects = ["clothing", "snack"]
pick_and_place("clothing")
pick_and_toss("snack")

# Summary: Pick and place clothes and wooden
blocks, pick and toss snacks and drinks.
objects = ["jacket", "candy bar", "soda can",
"Pepsi can", "jeans", "wooden block",
"orange", "chips", "wooden block 2", "apple"]
pick_and_place("jacket")
pick_and_place("jeans")
```

```
pick_and_place("wooden block")
pick_and_place("wooden block 2")
pick_and_toss("candy bar")
pick_and_toss("soda can")
pick_and_toss("Pepsi can")
pick_and_toss("orange")
pick_and_toss("chips")
pick_and_toss("apple")
```

A.5 Category extraction for real-world system

```
# Summary: Put shirts on the bed, jackets and
pants on the chair, and bags on the shelf.
objects = ["shirt", "jacket or pants", "bag"]

# Summary: Put pillows on the sofa, clothes on
the chair, and shoes on the rack.
objects = ["pillow", "clothing", "shoe"]

# Summary: Put clothes on the sofa, snacks in
the plastic storage box, cans in the recycling
bin, wooden blocks in the drawer, and fruits
in the black storage box.
objects = ["clothing", "snack", "can",
"wooden block", "fruit"]
```

A.6 Receptacle selection for real-world system

```
# Summary: Put clothes in the laundry basket
and toys in the storage box.
objects = ["socks", "toy car", "shirt", "Lego
brick"]
receptacles = ["laundry basket", "storage
box"]
pick_and_place("socks", "laundry basket")
pick_and_place("toy car", "storage box")
pick_and_place("shirt", "laundry basket")
pick_and_place("Lego brick", "storage box")

# Summary: Put clothes on the sofa, snacks in
the plastic storage box, cans in the recycling
bin, wooden blocks in the drawer, and fruits
in the black storage box.
objects = ["clothing", "snack", "can", "wooden
block", "fruit"]
receptacles = ["recycling bin", "plastic
storage box", "black storage box", "sofa",
"drawer"]
pick_and_place("clothing", "sofa")
pick_and_place("snack", "plastic storage box")
pick_and_place("can", "recycling bin")
pick_and_place("wooden block", "drawer")
pick_and_place("fruit", "black storage box")
```

A.7 Primitive selection for real-world system

```
# Summary: Pick and place clothes, pick and
toss snacks.
objects = ["granola bar", "hat", "toy car",
"Lego brick", "fruit snacks", "shirt"]
pick_and_toss("granola bar")
pick_and_place("hat")
pick_and_place("toy car")
pick_and_place("Lego brick")
pick_and_toss("fruit snacks")
pick_and_place("shirt")

# Summary: Pick and place granola bars, hats,
toy cars, and Lego bricks, pick and toss fruit
snacks and shirts.
objects = ["clothing", "snack"]
pick_and_place("clothing")
pick_and_toss("snack")

# Summary: Pick and place clothes and wooden
blocks, pick and toss snacks and drinks.
objects = ["clothing", "snack", "can", "wooden
block", "fruit"]
pick_and_place("clothing")
pick_and_place("wooden block")
pick_and_toss("snack")
pick_and_toss("can")
pick_and_toss("fruit")
```

References

Abdo, N., Stachniss, C., Spinello, L., & Burgard, W. (2015). Robot, organize my shelves! tidying up objects by predicting user preferences. In *2015 IEEE international conference on robotics and automation (ICRA)*.

Batra, D., Chang, A. X., Chernova, S., Davison, A. J., Deng, J., Koltun, V., Levine, S., Malik, J., Mordatch, I., & Mottaghi, R., et al. (2020). Rearrangement: A challenge for embodied ai. arXiv preprint arXiv:2011.01975

Brohan, A., Chebotar, Y., Finn, C., Hausman, K., Herzog, A., Ho, D., Ibarz, J., Irpan, A., Jang, E., & Julian, R. (2022). Do as i can, not as i say: Grounding language in robotic affordances. In *6th annual conference on robot learning*.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. (2020). Language models are few-shot learners. *Advances in Neural Information Processing Systems, 33*, 1877–1901.

Chen, W., Hu, S., Talak, R., & Carlone, L. (2022). Leveraging large language models for robot 3d scene understanding. arXiv preprint arXiv:2209.05629

Chen, M., Tworek, J., Jun, H., Yuan, Q., Pinto, H.P.d.O., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., & Brockman, G., et al. (2021). Evaluating large language models trained on code. arXiv preprint arXiv:2107.03374

Chen, B., Xia, F., Ichter, B., Rao, K., Gopalakrishnan, K., Ryoo, M.S., Stone, A., & Kappler, D. (2022). Open-vocabulary queryable scene representations for real world planning. arXiv preprint arXiv:2209.09874

Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., Barham, P., Chung, H.W., Sutton, C., & Gehrmann, S., et al. (2022). Palm: Scaling language modeling with pathways. arXiv preprint arXiv:2204.02311

Coulter, R. C. (1992). Implementation of the pure pursuit path tracking algorithm. Technical report, Carnegie-Mellon UNIV Pittsburgh PA Robotics INST.

Dewi, T., Risma, P., & Oktarina, Y. (2020). Fruit sorting robot based on color and size for an agricultural product packaging system. *Bulletin of Electrical Engineering and Informatics, 9*(4), 1438–1445.

Ehsani, K., Han, W., Herrasti, A., VanderBilt, E., Weihs, L., Kolve, E., Kembhavi, A., & Mottaghi, R. (2021). Manipulathor: A framework for visual object manipulation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*.

Gan, C., Zhou, S., Schwartz, J., Alter, S., Bhandwaldar, A., Gutfreund, D., Yamins, D. L., DiCarlo, J. J., McDermott, J., & Torralba, A. (2022). The threedworld transport challenge: A visually guided task-and-motion planning benchmark towards physically realistic embodied ai. In *2022 International conference on robotics and automation (ICRA)*.

Garrido-Jurado, S., Muñoz-Salinas, R., Madrid-Cuevas, F. J., & Marín-Jiménez, M. J. (2014). Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recognition, 47*(6), 2280–2292.

Gu, X., Lin, T.-Y., Kuo, W., & Cui, Y. (2021). Open-vocabulary object detection via vision and language knowledge distillation. In *International conference on learning representations*.

Gupta, M., & Sukhatme, G. S. (2012). Using manipulation primitives for brick sorting in clutter. In *2012 IEEE international conference on robotics and automation*.

Herde, M., Kottke, D., Calma, A., Bieshaar, M., Deist, S., & Sick, B. (2018). Active sorting: An efficient training of a sorting robot with active learning techniques. In *2018 international joint conference on neural networks (IJCNN)*.

Høeg, S. H., & Tingelstad, L. (2022). More than eleven thousand words: Towards using language models for robotic sorting of unseen objects into arbitrary categories. In *Workshop on language and robotics at CoRL 2022*.

Holmberg, R., & Khatib, O. (2000). Development and control of a holonomic mobile robot for mobile manipulation tasks. *The International Journal of Robotics Research, 19*(11), 1066–1074.

Huang, W., Abbeel, P., Pathak, D., & Mordatch, I. (2022). Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. arXiv preprint arXiv:2201.07207

Huang, E., Jia, Z., & Mason, M. T. (2019). Large-scale multi-object rearrangement. In *2019 international conference on robotics and automation (ICRA)*.

Huang, W., Xia, F., Xiao, T., Chan, H., Liang, J., Florence, P., Zeng, A., Tompson, J., Mordatch, I., & Chebotar, Y., et al. (2022). Inner monologue: Embodied reasoning through planning with language models. arXiv preprint arXiv:2207.05608

Kang, M., Kwon, Y., & Yoon, S.-E. (2018). Automated task planning using object arrangement optimization. In *2018 15th international conference on ubiquitous robots (UR)*, IEEE.

Kant, Y., Ramachandran, A., Yenamandra, S., Gilitschenski, I., Batra, D., Szot, A., & Agrawal, H. (2022). Housekeep: Tidying virtual households using commonsense reasoning. arXiv preprint arXiv:2205.10712

Kapelyukh, I., & Johns, E. (2022). My house, my rules: Learning tidying preferences with graph neural networks. In *Conference on robot learning*.

Kojima, T., Gu, S.S., Reid, M., Matsuo, Y., & Iwasawa, Y. (2022). Large language models are zero-shot reasoners. arXiv preprint arXiv:2205.11916

Kolve, E., Mottaghi, R., Han, W., VanderBilt, E., Weihs, L., Herrasti, A., Gordon, D., Zhu, Y., Gupta, A., & Farhadi, A. (2017). Ai2-thor: An interactive 3d environment for visual ai. arXiv preprint arXiv:1712.05474

Kujala, J. V., Lukka, T. J., & Holopainen, H. (2016). Classifying and sorting cluttered piles of unknown objects with robots: A learning approach. In *2016 IEEE/RSJ international conference on intelligent robots and systems (IROS)*.

Li, C., Xia, F., Martín-Martín, R., Lingelbach, M., Srivastava, S., Shen, B., Vainio, K.E., Gokmen, C., Dharan, G., & Jain, T. (2022). igibson 2.0: Object-centric simulation for robot learning of everyday household tasks. In *Conference on robot learning*.

Li, C., Zhang, R., Wong, J., Gokmen, C., Srivastava, S., Martín-Martín, R., Wang, C., Levine, G., Lingelbach, M., & Sun, J. (2022). Behavior-1k: A benchmark for embodied ai with 1000 everyday activities and realistic simulation. In *6th annual conference on robot learning*.

Liang, J., Huang, W., Xia, F., Xu, P., Hausman, K., Ichter, B., Florence, P., & Zeng, A. (2022). Code as policies: Language model programs for embodied control. arXiv preprint arXiv:2209.07753

Lin, K., Agia, C., Migimatsu, T., Pavone, M., Bohg, J. (2023). Text2motion: From natural language instructions to feasible plans. arXiv preprint arXiv:2303.12153

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., & Stoyanov, V. (2019). Roberta: A robustly optimized bert pretraining approach. arXiv preprint arXiv:1907.11692

Lukka, T. J., Tossavainen, T., Kujala, J. V., & Raiko, T. (2014). Zenrobotics recycler–robotic sorting using machine learning. In *Proceedings of the international conference on sensor-based sorting (SBS)*.

Madaan, A., Zhou, S., Alon, U., Yang, Y., & Neubig, G. (2022). Language models of code are few-shot commonsense learners. arXiv preprint arXiv:2210.07128

Mees, O., Borja-Diaz, J., & Burgard, W. (2022). Grounding language with visual affordances over unstructured data. arXiv preprint arXiv:2210.01911

Miller, G. A. (1995). Wordnet: A lexical database for english. *Communications of the ACM, 38*(11), 39–41.

Minderer, M., Gritsenko, A., Stone, A., Neumann, M., Weissenborn, D., Dosovitskiy, A., Mahendran, A., Arnab, A., Dehghani, M., & Shen, Z., et al. (2022). Simple open-vocabulary object detection with vision transformers. arXiv preprint arXiv:2205.06230

Nye, M., Andreassen, A. J., Gur-Ari, G., Michalewski, H., Austin, J., Bieber, D., Dohan, D., Lewkowycz, A., Bosma, M., & Luan, D., et al. (2021). Show your work: Scratchpads for intermediate computation with language models. arXiv preprint arXiv:2112.00114

Pan, Z., Hauser, K. (2021). Decision making in joint push-grasp action space for large-scale object sorting. In *2021 IEEE international conference on robotics and automation (ICRA)*.

Puig, X., Ra, K., Boben, M., Li, J., Wang, T., Fidler, S., & Torralba, A. (2018). Virtualhome: Simulating household activities via programs. In *Proceedings of the IEEE conference on computer vision and pattern recognition*.

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., & Clark, J. (2021). Learning transferable visual models from natural language supervision. In *International conference on machine learning*.

Raman, S. S., Cohen, V., Rosen, E., Idrees, I., Paulius, D., & Tellex, S. (2022). Planning with large language models via corrective re-prompting. arXiv preprint arXiv:2211.09935

Rasch, R., Sprute, D., Pörtner, A., Battermann, S., & König, M. (2019). Tidy up my room: Multi-agent cooperation for service tasks in

smart environments. *Journal of Ambient Intelligence and Smart Environments, 11*(3), 261–275.
Reimers, N., & Gurevych, I. (2019). Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*.
Ren, A. Z., Govil, B., Yang, T.-Y., Narasimhan, K., & Majumdar, A. (2022). Leveraging language for accelerated learning of tool manipulation. arXiv preprint arXiv:2206.13074
Rytting, C., & Wingate, D. (2021). Leveraging the inductive bias of large language models for abstract textual reasoning. *Advances in Neural Information Processing Systems, 34*, 17111–17122.
Sanh, V., Debut, L., Chaumond, J., & Wolf, T. (2019). Distilbert, a distilled version of bert: Smaller, faster, cheaper and lighter. arXiv preprint arXiv:1910.01108
Sarch, G., Fang, Z., Harley, A.W., Schydlo, P., Tarr, M.J., Gupta, S., & Fragkiadaki, K. (2022). Tidee: Tidying up novel rooms using visuo-semantic commonsense priors. In *European conference on computer vision*.
Shah, D., Osinski, B., Ichter, B., & Levine, S. (2022). LM-Nav: Robotic navigation with large pre-trained models of language, vision, and action. arXiv preprint arXiv:2207.04429
Shridhar, M., Thomason, J., Gordon, D., Bisk, Y., Han, W., Mottaghi, R., Zettlemoyer, L., & Fox, D. (2020). Alfred: A benchmark for interpreting grounded instructions for everyday tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.
Shridhar, M., Yuan, X., Côté, M.-A., Bisk, Y., Trischler, A., & Hausknecht, M. J. (2021). Alfworld: Aligning text and embodied environments for interactive learning. In *ICLR*.
Silver, T., Hariprasad, V., Shuttleworth, R. S., Kumar, N., Lozano-Pérez, T., & Kaelbling, L. P. (2022). Pddl planning with pretrained large language models. In *NeurIPS 2022 foundation models for decision making workshop*.
Singh, I., Blukis, V., Mousavian, A., Goyal, A., Xu, D., Tremblay, J., Fox, D., Thomason, J., & Garg, A. (2022). Progprompt: Generating situated robot task plans using large language models. arXiv preprint arXiv:2209.11302
Song, H., Haustein, J. A., Yuan, W., Hang, K., Wang, M.Y., Kragic, D., Stork, J. A. (2020). Multi-object rearrangement with monte Carlo tree search: A case study on planar nonprehensile sorting. In *2020 IEEE/RSJ international conference on intelligent robots and systems (IROS)*.
Srivastava, S., Li, C., Lingelbach, M., Martín-Martín, R., Xia, F., Vainio, K. E., Lian, Z., Gokmen, C., Buch, S., & Liu, K. (2022). Behavior: Benchmark for everyday household activities in virtual, interactive, and ecological environments. In *Conference on robot learning*.
Szabo, R., Lie, I. (2012). Automated colored object sorting application for robotic arms. In *2012 10th international symposium on electronics and telecommunications*.
Szot, A., Clegg, A., Undersander, E., Wijmans, E., Zhao, Y., Turner, J., Maestre, N., Mukadam, M., Chaplot, D. S., Maksymets, O., et al. (2021). Habitat 2.0: Training home assistants to rearrange their habitat. *Advances in Neural Information Processing Systems, 34*, 251–266.
Taniguchi, A., Isobe, S., El Hafi, L., Hagiwara, Y., & Taniguchi, T. (2021). Autonomous planning based on spatial concepts to tidy up home environments with service robots. *Advanced Robotics, 35*(8), 471–489.
Wei, J., Tay, Y., Bommasani, R., Raffel, C., Zoph, B., Borgeaud, S., Yogatama, D., Bosma, M., Zhou, D., & Metzler, D., et al. (2022). Emergent abilities of large language models. arXiv preprint arXiv:2206.07682
Wei, J., Wang, X., Schuurmans, D., Bosma, M., Chi, E., Le, Q., & Zhou, D. (2022). Chain of thought prompting elicits reasoning in large language models. arXiv preprint arXiv:2201.11903
Weihs, L., Deitke, M., Kembhavi, A., & Mottaghi, R. (2021). Visual room rearrangement. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*.
Wu, J., Antonova, R., Kan, A., Lepert, M., Zeng, A., Song, S., Bohg, J., Rusinkiewicz, S., & Funkhouser, T. (2023). Tidybot: Personalized robot assistance with large language models. In *IEEE/rsj international conference on intelligent robots and systems (IROS)*.
Yan, Z., Crombez, N., Buisson, J., Ruichck, Y., Krajnik, T., & Sun, L. (2021). A quantifiable stratification strategy for tidy-up in service robotics. In *2021 IEEE international conference on advanced robotics and its social impacts (ARSO)*.
Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., & Cao, Y. (2022). React: Synergizing reasoning and acting in language models. arXiv preprint arXiv:2210.03629
Zeng, A., Wong, A., Welker, S., Choromanski, K., Tombari, F., Purohit, A., Ryoo, M., Sindhwani, V., Lee, J., & Vanhoucke, V., et al. (2022). Socratic models: Composing zero-shot multimodal reasoning with language. arXiv preprint arXiv:2204.00598
Zeng, A., Song, S., Lee, J., Rodriguez, A., & Funkhouser, T. (2020). Tossingbot: Learning to throw arbitrary objects with residual physics. *IEEE Transactions on Robotics, 36*(4), 1307–1319.
Zeng, A., Song, S., Yu, K.-T., Donlon, E., Hogan, F. R., Bauza, M., Ma, D., Taylor, O., Liu, M., Romo, E., et al. (2022). Robotic pick-and-place of novel objects in clutter with multi-affordance grasping and cross-domain image matching. *The International Journal of Robotics Research, 41*(7), 690–705.