Toward Grounded Commonsense Reasoning

Minae Kwon, Hengyuan Hu, Vivek Myers†, Siddharth Karamcheti, Anca Dragan†, Dorsa Sadigh
Stanford University, UC Berkeley†
{mnkwon, hengyuan, skaramcheti, dorsa}@cs.stanford.edu,
{vmyers, anca}@berkeley.edu†

***Abstract*— Consider a robot tasked with tidying a desk with a meticulously constructed Lego sports car. A human may recognize that it is not appropriate to disassemble the sports car and put it away as part of the "tidying." How can a robot reach that conclusion? Although large language models (LLMs) have recently been used to enable commonsense reasoning, grounding this reasoning in the real world has been challenging. To reason in the real world, robots must go beyond passively querying LLMs and *actively gather information from the environment* that is required to make the right decision. For instance, after detecting that there is an occluded car, the robot may need to actively perceive the car to know whether it is an advanced model car made out of Legos or a toy car built by a toddler. We propose an approach that leverages an LLM and vision language model (VLM) to help a robot actively perceive its environment to perform grounded commonsense reasoning. To evaluate our framework at scale, we release the MESSYSURFACES dataset which contains images of 70 real-world surfaces that need to be cleaned. We additionally illustrate our approach with a robot on 2 carefully designed surfaces. We find an average 12.9% improvement on the MESSYSURFACES benchmark and an average 15% improvement on the robot experiments over baselines that do not use active perception. The dataset, code, and videos of our approach can be found at https://minaek.github.io/grounded_commonsense_reasoning/.**

I. INTRODUCTION

Imagine you are asked to clean up a desk and you see a meticulously constructed Lego sports car on it. You might immediately recognize that the normative behavior is to leave the car be, rather than taking it apart and putting it away as part of the "cleaning". But how would a robot in that same position know that's the right thing to do? Traditionally, we would expect this information to be specified in the robot's objective – either learned from demonstrations [1], [2], [3] or from human feedback [4], [5], [6], [7]. While a robot could expensively query a human for their preferences on how to clean the car, we explore a different question in this work: how can we equip robots with the commonsense reasoning necessary to follow normative behavior *in the absence of personalized input from the human*? The ability to behave in a commonsense, normative manner can be an effective prior over robot behavior when personalized feedback is not present. When feedback is present, having a good prior can reduce the amount of human specification needed.

Recent work has demonstrated that large language models (LLMs) trained on internet data have enough context for commonsense reasoning [8], making moral judgements [9], [10], or acting as a proxy reward function capturing human preferences [11]. Rather than explicitly asking a human for the answer, the robot could instead ask an LLM whether it would be appropriate to clean up the car. But in real-world environments, this is easier said than done. Tapping into an LLM's commonsense reasoning skills in the real-world requires the ability to *ground language in the robot's perception of the world* – an ability that might be afforded by powerful vision-and-language models (VLMs). Unfortunately, we find that today's VLMs cannot reliably provide all the relevant information for commonsense reasoning. For instance, a VLM may not describe that the sports car is constructed from Legos, or that it contains over 1000 pieces – details that are key to making decisions. While advanced multi-modal models might alleviate this problem, a fundamental limitation is the image itself might not contain all the relevant information. If the sports car is partially occluded by a bag (as in Fig. 1), no VLM could provide the necessary context for reasoning over what actions to take. Such a system would instead need the ability to move the bag – or move *itself* – to actively gather the necessary information. Thus, in order to perform "grounded commonsense reasoning" robots must go beyond passively querying LLMs and VLMs to obtain action plans and instead *directly interact with the environment.* Our insight is that robots must reason about what additional information they need to make appropriate decisions, *and then actively perceive the environment to gather that information*.

Acting on this insight, we propose a framework to enable a robot to perform grounded commonsense reasoning by iteratively identifying details it still needs to clarify about the scene before it can make a decision (e.g. is the model car made out of intricate Lego pieces or MEGA Bloks?) and actively gathering new observations to help answer those questions (e.g. getting a close up of the car from a better angle). In this paper, we focus on the task of cleaning up real-world surfaces through commonsense reasoning. Our framework is shown in Fig. 1. Given a textual description of the desk, an LLM asks follow-up questions about the state of each object that it needs in order to make a decision of what the robot should do with that object. The robot actively perceives the scene by taking close-up photos of each object from angles suggested by the LLM. The follow-up questions and close-up photos are then given to a VLM so that it can provide more information about the scene.

This process can be repeated multiple times. The LLM then decides on an action the robot should take to clean

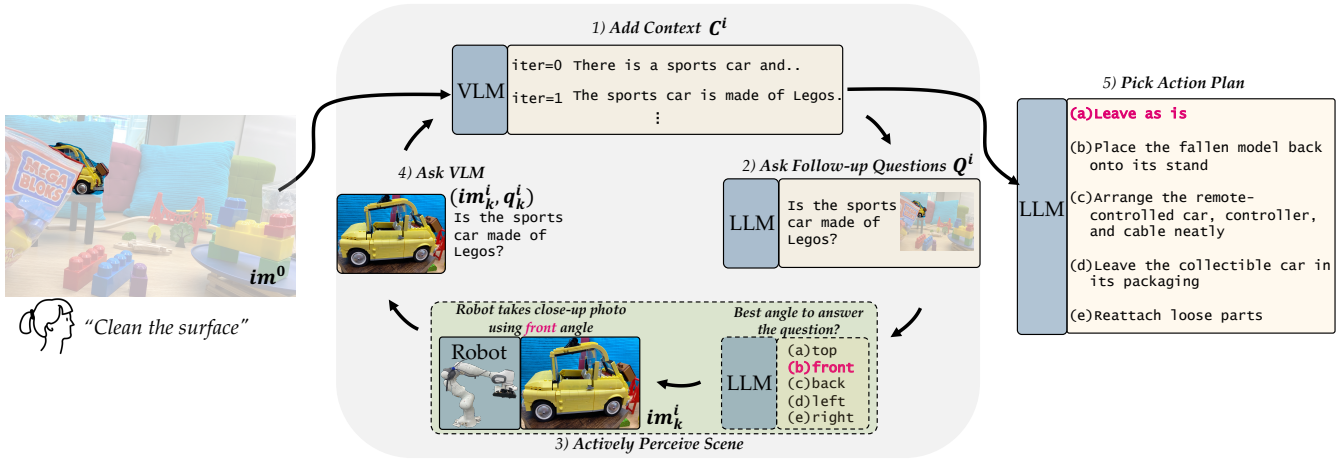


Fig. 1. **Grounded Commonsense Reasoning Framework.** We demonstrate our framework using the sports car. Blue boxes indicate the model and yellow boxes indicate its output. Our framework takes an image of the scene and an instruction as input. 1) The VLM outputs an initial description of the scene $\mathscr{C}^0$ from the initial image im^0. 2) The LLM asks follow-up questions about each object in the scene, $\mathscr{Q}^i$. 3) The robot takes a close-up image im^i_k of each object k. It is guided by an LLM that chooses the best angle that would help answer the question. 4) We pair the close-up images with the follow-up questions and ask the VLM to answer them. Answers are appended to the context. We repeat steps 1-4 to gather more information. 5) We query an LLM to choose the most appropriate way to tidy the object.

the object in an appropriate manner. For example, our robot leaves the Lego sports car intact, throws a browning half-eaten banana in the trash, but keeps an unopened can of Yerba Mate on the desk. Furthermore, we release the MESSYSURFACES dataset containing images of 70 surfaces as well an evaluation benchmark that assesses how well a robot can clean up each surface in an appropriate manner. The dataset is available here.

We evaluate our framework on our benchmark dataset as well as on a real-world robotic system. We examine each component of our framework, asking whether the robot asks useful follow-up questions, whether the robot chooses informative close-up images, and whether the images actually help a VLM more accurately answer questions. We find an average 12.9% improvement on the MESSYSURFACES benchmark and an average 15% improvement on the robot experiments over baselines that do not use active perception.

II. RELATED WORK

a) Commonsense Reasoning: Large language models are trained on internet-scale data, making them effective commonsense reasoners [12], [13], [14], [15]. Prior works have studied whether LLMs' commonsense reasoning aligns with human values [9], [10], [16], [11]. There is evidence that when LLMs make moral or social judgements, they align with the normative beliefs of the population that generated their training data [17]. In addition, prior work show commonsense reasoning models can align with conventional beliefs [18], [19], [20], [21]. *Our approach is in line with commonsense reasoning; instead of adapting to individual preferences, we show we can take commonsense actions to clean up a scene.*

b) Learning Human Preferences: Past work on aligning with human preferences has focused on using human feedback to infer rewards and policies by designing queries for active preference learning [22], [4], [6], [23], performing inverse reinforcement learning [24], [25], or recovering reward signals from language feedback [11], [26], [27], [28], [29]. Policies defined via LLMs have also been directly tuned with language feedback by approaches like RLHF [30]. Instead of querying humans, we leverage normative values from pre-trained models. While some works use normative values from LLMs in negotiations and games [31], these are not grounded in the real world. *In this work, we do not focus on particular human preferences, though the normative responses of LLMs could be fine-tuned for particular applications.*

c) Active Perception: When robots must use commonsense reasoning like humans, active information gathering may be important [32]. Approaches like TidyBot actively zoom-in on objects to better categorize them [33]. Other approaches such as Inner Monologue seek out additional environment information, but need aid from a human annotator or assume access to simulators [34], [35]. VLMs have also been used for active perception in navigation [36], [37], [38]. *In this work, we show that active perception is necessary for grounded commonsense reasoning, enabled by the semantic knowledge in an LLM.*

d) LLMs for Robotics: Past work uses semantic knowledge in LLMs for task planning. Methods like SayCan decompose natural language tasks into primitive action plans [39], [40], [41]. In addition, approaches such as Code as Policies [42], [43] use LLMs to write Python programs that plan with executable robot policy code. Other approaches use multimodal sequence models to reason about language-conditioned manipulation [44], [45], [46], [47]. *We use the semantic awareness of an LLM to reason about action plans. Unlike the above works, an LLM interactively queries an off-the-shelf VLM to ground the scene.*

III. Grounding Commonsense Reasoning

We propose a framework that combines existing foundation models in a novel way to enable active information gathering, shown in Fig. 1. Our framework makes multiple calls to an LLM and VLM to gather information. The LLM plays a number of distinct roles in our framework that we distinguish below: generating informative follow-up questions, guiding active perception, and choosing an action plan. In every call, the LLM takes in and outputs a string $\texttt{LLM}: A^* \to A^*$, and the VLM takes in an image, string pair and outputs a string $\texttt{VLM}: \mathscr{I} \times A^* \to A^*$, where A^* is the set of all strings and $\mathscr{I}$ is the set of all images. The context $\mathscr{C}^i \in A^*$ contains information about the scene that the robot has gathered up to iteration i of the framework. Initially, the inputs to our framework are an image of the scene $im^0 \in \mathscr{I}$ (i.e., an unblurred image from Fig. 1) and an instruction (e.g., "clean the surface").

VLM Describes the Scene. Our framework starts with the VLM producing an initial description $\mathscr{C}^0$ of the scene from the scene image im^0. The description can contain varying amounts of information — in the most uninformative case, it may simply list the objects that are present. In our experiments, this is the description that we use.

LLM Generates Follow-Up Questions. To identify what information is missing from $\mathscr{C}^0$, we use an LLM to generate informative follow-up questions as shown in stage (2) of Fig. 1. We prompt an LLM with $\mathscr{C}^0$ and ask the LLM to produce a set of follow-up questions $\mathscr{Q}^i = \{q_1^i, \ldots, q_K^i\}$ for the K objects. LLMs are apt for this task because of their commonsense reasoning abilities. We use Chain-of-Thought prompting [48] where we first ask the LLM to reason about the appropriate way to tidy each object before producing a follow-up question (see examples in the supplementary). For example, the LLM could reason that the sports car should be put away if it is a toy but left on display if someone built it. The resulting follow-up question asks whether the sports car is built with Lego blocks. We assume that the information in $\mathscr{C}^0$ is accurate (i.e., correctly lists the names of all the objects) to prevent the LLM from generating questions based on inaccurate information.

Robot Actively Perceives the Scene. At this stage, one might normally query the VLM with the original scene image im^0. However if the object-in-question is obstructed or too small to see, the scene image might not provide enough information for the VLM to answer the follow-up question accurately (e.g., the sports car is obstructed in Fig. 1). Instead, we would like to provide an unobstructed close-up image $im_k^i \in \mathscr{I}$ of the object k to "help" the VLM accurately answer the generated questions. Taking informative close-up images requires interaction with the environment — something we can use a robot for.

To actively gather information, the robot should proceed based on some notion of "informativeness" of camera angles. To determine "informativeness", we can again rely on the commonsense knowledge of LLMs. Although LLMs don't have detailed visual information about the object, they can suggest reasonable angles that will be, on average, informative. For instance, an LLM will choose to take a photo from the top of an opaque mug, instead of its sides, to see its content. In practice, we find that this approach works well and can improve the informativeness of an image by 8%. We query an LLM to choose a close-up angle of the object from a set of angles {`<FRONT>`, `<BACK>`, `<LEFT>`, `<RIGHT>`, `<TOP>`} that would give an unobstructed view. We then pair the close-up images with their questions $\{(im_1^i, q_1^i), \ldots, (im_k^i, q_K^i)\}$ and query the VLM for answers to these questions in step (4) of our framework. We concatenate the VLM's answers for each object and append them to our context $\mathscr{C}^i$ to complete the iteration. To gather more information about each object, steps $1-4$ can be repeated where the number of iterations is a tunable parameter.

LLM Chooses an Action Plan. In the final step, for each object, we prompt the LLM with the context $\mathscr{C}^i$ and a multiple choice question that lists different ways to tidy an object. The LLM is then instructed to choose the most appropriate option. The multiple choice options come from the MessySurfaces benchmark questions, a bank of 308 multiple-choice questions about how to clean up real-life objects found on messy surfaces. For example, in Fig. 1, the LLM chooses to leave the sports car as is because it infers that the sports car must be on display. To map the natural language action to robot behavior, we implement a series of hand-coded programmatic skill primitives that define an API the LLM can call into. See §V for more details.

IV. The MessySurfaces Dataset

To assess a robot's ability to perform commonsense reasoning in grounded environments, we introduce the MessySurfaces dataset. The dataset consists of images of 308 objects across 70 real-world surfaces that need to be cleaned. An average of 68% of objects are occluded in scene-level images[1], so we also provide 5 close-up images as a way for the robot to "actively perceive" the object, see Fig. 2 for an example. MessySurfaces also includes a benchmark evaluation of multiple choice questions for each object where each option corresponds to different ways to tidy the object. Through a consensus of 5 human annotators, we determine which one of the choices is the most appropriate. To do well, a robot must reason about the appropriate way to clean each object from the images alone. Since no human preferences are given, the robot must identify relevant attributes of each object from the images (e.g., is the sports car built out of Legos or MEGA Bloks?) and then reason about how to tidy the object using this information. MessySurfaces contains 45 office desks, 4 bathroom counters, 5 bedroom tables, 8 kitchen counters, 4 living room tables and 4 dining tables.

a) Data Collection Process: We recruited 51 participants to provide images of cluttered surfaces. Each participant was asked to pick 4 – 6 objects on a surface. They were

[1]Computed as the average number of times annotators indicated a question cannot be answered by the scene image.

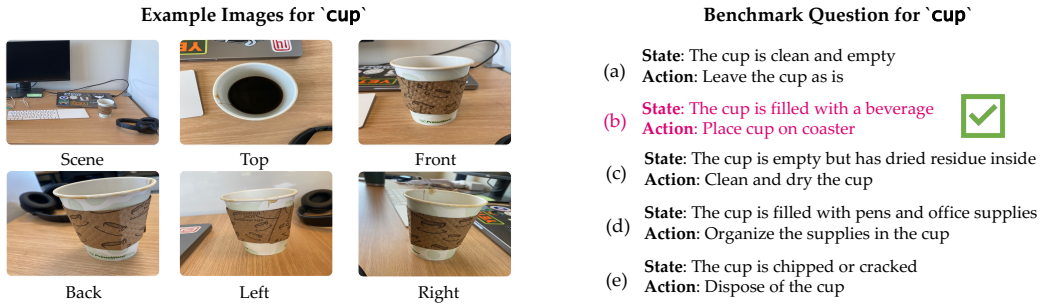


Fig. 2. **MESSYSURFACES Example.** Each object in MESSYSURFACES is represented by a scene image and 5 close-up images. Each object also has a benchmark question that presents 5 options to tidy the object; each option is constructed by producing a cleaning action conditioned on a hypothetical object state.

then asked to take a photo of the scene-level view as well as close-up photos of each object from the top, right, left, front, and back angles – the offline equivalent of having a robot actively navigate a scene. The task took approximately 15 – 30 minutes. After receiving the photos, we post-processed each image and cropped out any identifiable information.

b) Benchmark Evaluation: The benchmark questions consist of 5 LLM-generated multiple choice options about how to manipulate each object to clean the surface in an appropriate manner. To make the options diverse, we asked the LLM to first identify 5 states the object could be in and then queried it to come up with a cleaning action for each of those states (see Fig. 2 for an example). For each question, we recruited 5 annotators to choose the correct state-action pair based on the scene and close-up images of the object. Annotators were also given an option to indicate if none of the choices were a good fit. We used the majority label as our answer and omitted 16 questions (out of 324) where a majority thought none of the choices were a good fit. For questions that had two equally popular answers, we counted both as correct. Our annotators agreed on average 67% of the time. To evaluate the quality of our multiple choice options, we asked annotators to rate how appropriate each cleaning action is for each object state. Annotators gave each option an average rating of 4.1 out of 5. The average rating for the correct option was 4.4 out of 5. *Annotators.* In total, we recruited 350 English-speaking annotators from Prolific that were based in the U.S. or U.K. with an approval rating of at least 98%. Our study is IRB-approved.

V. EXPERIMENTS

We examine how well our approach can perform grounded commonsense reasoning on the MESSYSURFACES dataset as well as a real-world robotic system.

Primary Metric. We use accuracy on the benchmark questions as our primary metric. Each benchmark question presents 5 options on how to tidy the object, with accuracy defined as the percentage by which our framework selects the most appropriate option (as indicated by our annotators).

Baselines. Key to our approach (■ **Ours-LLM**) is the ability to supplement missing information by asking questions and actively perceiving the environment. To evaluate this, we compare the following:

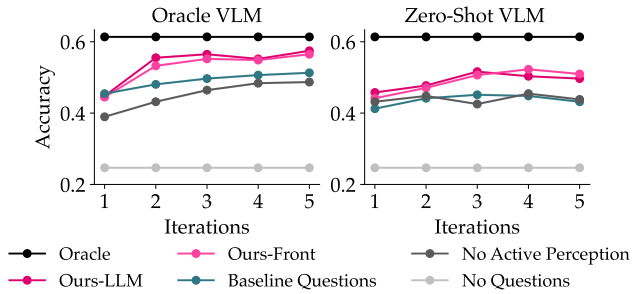


Fig. 3. **MESSYSURFACES Benchmark Accuracy.** For both the Oracle VLM and InstructBLIP, on average, our approach outperforms all baselines on the MESSYSURFACES benchmark. Accuracy is given by the percentage by which our framework selects the most appropriate (as indicated by our annotators) way to tidy each object.

- ■ **Oracle.** We ask a human annotator to answer the benchmark questions where they can actively perceive the scene using all angles.
- ■ **Ours-LLM.** Our approach as described in §III.
- ■ **Ours - Front.** Inspired by TidyBot [33], this is a variant of our approach wherein we simulate "zooming" into the image, using the "front" angle image as input to the VLM. The "front" angles can be the most informative angle in many cases, making it an effective heuristic.
- ■ **Baseline Questions.** This baseline evaluates the need for normative commonsense reasoning by asking more factual questions (e.g., "What color is the cup?").
- ■ **No Active Perception.** This baseline evaluates the need for active perception in our framework by allowing the robot to ask questions *that are answered solely from the scene image*.
- ■ **No Questions.** This baseline requires the robot to perform grounded commonsense reasoning from an initial description of the scene. The robot does not ask questions or actively perceive the environment, instead operating in an open-loop fashion akin to methods like SayCan [39].

Implementation Details. We use GPT-4 with temperature 0 as our LLM and InstructBLIP [49] (Flan-T5-XXL) as our VLM. We also report "oracle" results where a human

answers questions instead of the VLM to simulate results our approach could achieve if the VLM were near-perfect (denoted as the "Oracle VLM"). Further implementation details (e.g., prompts, model usage) are in the supplementary.

A. *Evaluation on* MESSYSURFACES

We evaluate our method on the 308 benchmark questions across 5 iterations of our framework. After each iteration, the robot is evaluated on the information it has accumulated up until that point. We measure accuracy on each question and report results using both the Oracle VLM and zero-shot performance on InstructBLIP. Although **No Question** and **Oracle** are "open-loop" methods that do not require iteration, we plot their results as a constant for comparison.

After 5 iterations, for both the Oracle VLM and InstructBLIP, our approaches outperform all baselines: **No Question**, **No Active Perception**, and **Baseline Questions**. Notably, **Ours-LLM** significantly outperforms **No Question** by an average of 27.7% across the two VLM types, $p < 0.01$. **Ours-LLM** also outperforms **Baseline Questions** by an average of 5% across the VLM types, $p > 0.05$ and outperforms **No Active Perception** by an average of 6%, $p > 0.05$. Using an Oracle VLM allows **Ours-LLM** to close the gap with the **Oracle** by an average of 5% more than using InstructBLIP. Although our approach outperforms baselines using both VLMs, we suspect that InstructBLIP gives lower accuracies because the MESSYSURFACES images – especially the close-up images – are out of distribution. For this reason, we presume that our approach gives a smaller advantage over other baseline methods when using InstructBLIP.

These results suggest that asking questions and actively perceiving the environment can enable grounded commonsense reasoning; with better VLMs, we can reach close to human-level performance. However, we were puzzled why the human **Oracle** was not more accurate. We hypothesize that in some situations, it is unclear what the most appropriate way to clean an object would be – our annotators agreed 67% of the time. To obtain higher accuracy, commonsense reasoning may sometimes not be enough and we must query user preferences to personalize the cleaning action; we explore this further in §VI and the supplementary. We now analyze each component of our framework.

Does the LLM Ask Good Follow-Up Questions? We first evaluate the LLM's follow-up questions and the reasoning used to produce those questions. On average, 82% of users agreed that the reasoning was valid and 87% agreed that the reasoning was appropriate. To evaluate the follow-up questions, we asked users to rate each question's usefulness and relevance for tidying the surface on a 5-point Likert scale. We compared against **Baseline Questions**, where we removed the constraint that LLM-generated questions must be relevant for commonsense reasoning about normative values. An example baseline question is, "Does the cup have a logo?" All prompts and example questions are in the supplementary. **Users rated our questions to be significantly more useful and relevant for tidying surfaces compared to the baseline** ($p < 0.01$, Fig. 4). However, across iterations, the average usefulness and relevance of our questions decreased. This result may be because there are not many useful and relevant questions to ask about simple objects such as a keyboard without interacting with them or people in the room.

Does the LLM Suggest Informative Close-Up Angles? We next focus on whether the close-up angles suggested by the LLM are informative. For each object, we asked users whether the object's follow-up question is answerable from the close-up angle chosen by the LLM by showing them the corresponding close-up image. We also do this for the "front" angle. As our main baseline, we ask users whether questions are answerable from the scene-level view. Additionally, we compare against angles that the LLM did not choose ("Non-LLM Angles"), as well as non-front angles. **Across 5 iterations we find that, on average, 35.5% more questions are answerable by LLM-chosen angles and 31% more questions are answerable by the front angles compared to the scene, $p < 0.01$. The LLM-chosen angles and front angle are also significantly more informative than the non-LLM-chosen angles and non-front angles respectively.** This trend holds for each iteration (Fig. 5).

Do Our Close-Up Angles Improve VLM Accuracy? Using VLMs for grounded commonsense reasoning pose challenges when there are obstructions in the image (e.g., a bag blocking the sports car) or when they are not able to describe relevant details. We hypothesized that providing a close-up image would "help" a VLM answer follow-up questions more accurately. We evaluate whether close-up images can actually improve VLM accuracy on follow-up questions. From the results in Table I, we see that **having access to close-up angles greatly improves the zero-shot prediction accuracy for both VLM variants.** More importantly, the front angles and the LLM proposed angles generally outperform other angles. These results show that it is beneficial to have both active perception and correct angles for our tasks.

B. *Evaluation on Real-World Robotic System*

To assess the performance of our system on a real-world robot, we assemble 2 surfaces with 11 objects that require complex commonsense reasoning to tidy up. Importantly, we design these surfaces so that the commonsense way to tidy each object would be unambiguous. The first surface resembles a child's play area, with toys of ranging complexities (e.g., a MEGA Bloks structure, a partially built toy train set, and a to-scale Lego model of an Italian sports car). The robot must understand which toys to clean up and which toys should be left on display. The second surface, shown in §C, consists of trash that a robot must sort through and decide whether to recycle, put in landfill, or keep on the desk.

Grounding Language in Robot Behavior. Following the active perception component of our framework, we use a robot arm (equipped with a wrist camera) to servo to angles produced by the LLM and take photos. To map the LLM-produced angles and natural-language action plans to robot behavior, we implement a series of programmatic skill primitives (e.g., `relocate(''block'')`). In this work, each "view" and "action" primitive is defined assuming

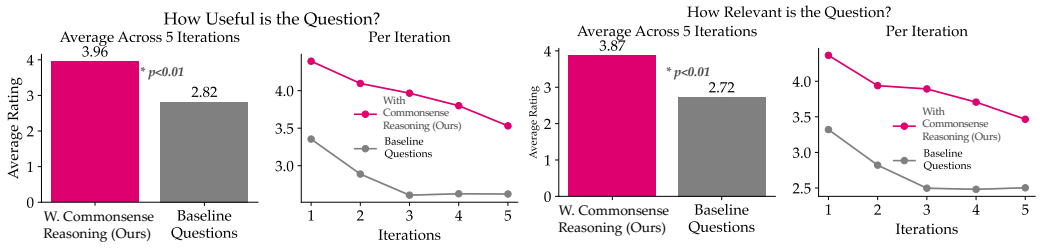


Fig. 4. **How Good are the Follow-Up Questions?** Users rated our questions to be significantly more useful and relevant compared to baseline questions, $p < 0.01$. However, the average usefulness and relevance of questions decreased over iterations.

TABLE I
VLM MULTIPLE-CHOICE PREDICTION ACCURACY (ZERO-SHOT) UNDER DIFFERENT ANGLES OVER 5 ITERATIONS

	Scene	Non-front Angles	Front Angle	Non-LLM Angles	LLM Angle
InstructBLIP (Vicuna)	47.98	51.06	**52.64**	50.94	**53.21**
InstructBLIP (Flan-T5)	51.95	53.99	**56.74**	54.08	**56.30**

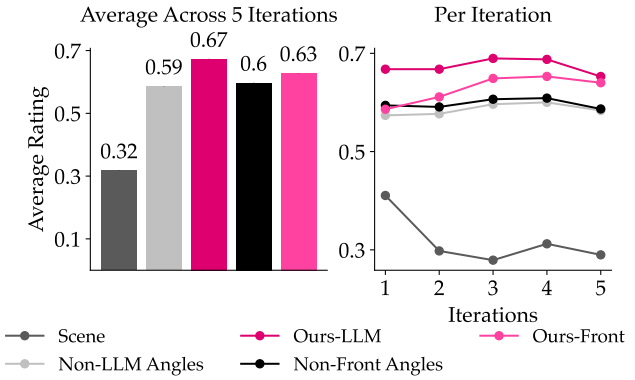


Fig. 5. **Do We Choose Informative Close-Up Angles?** An average of 33.25% more questions are answerable by the LLM-chosen angles and front angles compared to the scene, $p < 0.01$. The LLM-chosen angles and front angle are also significantly more informative than the non-LLM-chosen angles and non-front angles respectively.

access to the ground-truth object class and position. These programmatic skill primitives define an API that the LLM can call into, similar to the process introduced by [42]. Each action plan is translated to a sequence of these programmatic skills, which are then executed in an open loop (further implementation details are in the supplementary).

Benchmark Evaluation Results. To evaluate our method, we designed benchmark questions for each of the 11 objects in a similar manner to that outlined in §IV. We recruited 5 annotators on Prolific to choose the correct answer and took the majority label. We report results for both the Oracle VLM and InstructBLIP after running 5 iterations of our framework (see Figure in the supplementary). **Across both types of VLMs, Ours-LLM beats Baseline Questions by an average of** 13.5%**, beats No Active Perception by an average of** 18%**, and beats No Questions by an average of** 13.5%**.** With the Oracle VLM, we achieve **Oracle** performance. With InstructBLIP, our method produces a smaller advantage over baselines.

VI. DISCUSSION

The purpose of this work is to equip robots with basic grounded commonsense reasoning skills to reduce the need for human specification. These reasoning skills can later be personalized towards an individual's preferences. To this end, we conduct a preliminary study to explore how we can add personalization on top of our framework. We analyzed questions that the human **Oracle** got incorrect in §V and found that object attributes such as "dirtiness" can indeed be subjective. This may have caused the **Oracle** to incorrectly answer some questions. We experimented with adding personalization information to 8 questions where both the **Oracle** and our framework chose the same incorrect answer. **We found an average** 86% **improvement in accuracy, suggesting that preference information helps further enable grounded commonsense reasoning.** See the supplementary for more details.

Limitations and Future Work. While our work presents a first step towards actively grounded commonsense reasoning, there are some limitations to address. One limitation is our reliance on heuristics to guide our active perception pipeline – while the five specified angles are enough for most of the questions in the MESSYSURFACES dataset, there are cases where objects may be occluded, or otherwise require more granular views to answer questions; future work might explore learned approaches for guiding perception based on uncertainty, or developing multi-view, queryable scene representations [50], [51]. Similarly, we are limited by an inability to *interact with objects dynamically* – e.g., opening boxes, removing clutter. Finally, while we focus on commonsense behaviors, there are times where the "right" thing to do is to ask for human preferences.

Acknowledgements. This work was supported by DARPA YFA, NSF Award #2006388, #2125511, #2218760, AFOSR YIP, JP Morgan, ONR, and TRI.

REFERENCES

[1] S. Ross, G. Gordon, and D. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*. JMLR Workshop and Conference Proceedings, 2011, pp. 627–635.

[2] D. S. Brown, W. Goo, and S. Niekum, "Better-than-demonstrator imitation learning via automatically-ranked demonstrations," Oct. 2019.

[3] M. Palan, N. C. Landolfi, G. Shevchuk, and D. Sadigh, "Learning Reward Functions by Integrating Human Demonstrations and Preferences," June 2019.

[4] D. Sadigh, A. D. Dragan, S. S. Sastry, and S. A. Seshia, "Active preference-based learning of reward functions," in *Proceedings of Robotics: Science and Systems (RSS)*, July 2017.

[5] M. Li, A. Canberk, D. P. Losey, and D. Sadigh, "Learning human objectives from sequences of physical corrections," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 2877–2883.

[6] E. Biyik, D. P. Losey, M. Palan, N. C. Landolfi, G. Shevchuk, and D. Sadigh, "Learning Reward Functions from Diverse Sources of Human Feedback: Optimally Integrating Demonstrations and Preferences," 2021.

[7] T. Fitzgerald, P. Koppol, P. Callaghan, R. Q. Wong, R. Simmons, O. Kroemer, and H. Admoni, "INQUIRE: INteractive Querying for User-aware Informative REasoning."

[8] A. Talmor, O. Yoran, R. L. Bras, C. Bhagavatula, Y. Goldberg, Y. Choi, and J. Berant, "CommonsenseQA 2.0: Exposing the limits of AI through gamification," *arXiv preprint arXiv:2201.05320*, 2022.

[9] L. Jiang, J. D. Hwang, C. Bhagavatula, R. L. Bras, J. Liang, J. Dodge, K. Sakaguchi, M. Forbes, J. Borchardt, S. Gabriel, *et al.*, "Can Machines Learn Morality? The Delphi Experiment," July 2022.

[10] D. Hendrycks, C. Burns, S. Basart, A. Critch, J. Li, D. Song, and J. Steinhardt, "Aligning ai with shared human values," *arXiv preprint arXiv:2008.02275*, 2020.

[11] M. Kwon, S. M. Xie, K. Bullard, and D. Sadigh, "Reward design with language models," *arXiv preprint arXiv:2303.00001*, 2023.

[12] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, *et al.*, "Language Models are Few-Shot Learners," 2020.

[13] C. M. Rytting and D. Wingate, "Leveraging the Inductive Bias of Large Language Models for Abstract Textual Reasoning."

[14] B. Zhang and H. Soh, "Large Language Models as Zero-Shot Human Models for Human-Robot Interaction," Mar. 2023.

[15] X. Zhou, Y. Zhang, L. Cui, and D. Huang, "Evaluating Commonsense in Pre-Trained Language Models," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 05, pp. 9733–9740, Apr. 2020.

[16] Z. Jin, S. Levine, F. Gonzalez Adauto, O. Kamal, M. Sap, M. Sachan, R. Mihalcea, J. Tenenbaum, and B. Schölkopf, "When to Make Exceptions: Exploring Language Models as Accounts of Human Moral Judgment," *Advances in Neural Information Processing Systems*, vol. 35, pp. 28 458–28 473, Dec. 2022.

[17] K. C. Fraser, S. Kiritchenko, and E. Balkir, "Does Moral Code Have a Moral Code? Probing Delphi's Moral Philosophy," May 2022.

[18] P. Ammanabrolu, L. Jiang, M. Sap, H. Hajishirzi, and Y. Choi, "Aligning to Social Norms and Values in Interactive Narratives," May 2022.

[19] D. Hendrycks, M. Mazeika, A. Zou, S. Patel, C. Zhu, J. Navarro, D. Song, B. Li, and J. Steinhardt, "What Would Jiminy Cricket Do? Towards Agents That Behave Morally," Feb. 2022.

[20] D. Hendrycks, C. Burns, S. Basart, A. Critch, J. Li, D. Song, and J. Steinhardt, "Aligning AI With Shared Human Values," Feb. 2023.

[21] R. Zellers, Y. Bisk, A. Farhadi, and Y. Choi, "From Recognition to Cognition: Visual Commonsense Reasoning," Mar. 2019.

[22] R. Akrour, M. Schoenauer, and M. Sebag, "April: Active preference learning-based reinforcement learning," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2012, pp. 116–131.

[23] M. Cakmak, S. S. Srinivasa, M. K. Lee, J. Forlizzi, and S. Kiesler, "Human preferences for robot-human hand-over configurations," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2011, pp. 1986–1993.

[24] B. D. Ziebart, A. L. Maas, J. A. Bagnell, and A. K. Dey, "Maximum entropy inverse reinforcement learning." in *Aaai*, vol. 8, 2008, pp. 1433–1438.

[25] N. D. Ratliff, J. A. Bagnell, and M. A. Zinkevich, "Maximum margin planning," Pittsburgh, Pennsylvania, 2006.

[26] L. Fan, G. Wang, Y. Jiang, A. Mandlekar, Y. Yang, H. Zhu, A. Tang, D.-A. Huang, Y. Zhu, and A. Anandkumar, "MineDojo: Building Open-Ended Embodied Agents with Internet-Scale Knowledge," June 2022.

[27] S. Singh and J. H. Liao, "Concept2Robot 2.0: Improving Learning of Manipulation Concepts Using Enhanced Representations."

[28] L. Shao, T. Migimatsu, Q. Zhang, K. Yang, and J. Bohg, "Concept2Robot: Learning manipulation concepts from instructions and human demonstrations," *The International Journal of Robotics Research*, vol. 40, no. 12-14, pp. 1419–1434, Dec. 2021.

[29] S. Mirchandani, S. Karamcheti, and D. Sadigh, "Ella: Exploration through learned language abstraction," Oct. 2021.

[30] D. M. Ziegler, N. Stiennon, J. Wu, T. B. Brown, A. Radford, D. Amodei, P. Christiano, and G. Irving, "Fine-Tuning Language Models from Human Preferences," Jan. 2020.

[31] H. Hu and D. Sadigh, "Language instructed reinforcement learning for human-ai coordination," in *40th International Conference on Machine Learning (ICML)*, 2023.

[32] J. Bohg, K. Hausman, B. Sankaran, O. Brock, D. Kragic, S. Schaal, and G. Sukhatme, "Interactive Perception: Leveraging Action in Perception and Perception in Action," *IEEE Transactions on Robotics*, vol. 33, no. 6, pp. 1273–1291, Dec. 2017.

[33] J. Wu, R. Antonova, A. Kan, M. Lepert, A. Zeng, S. Song, J. Bohg, S. Rusinkiewicz, and T. Funkhouser, "TidyBot: Personalized Robot Assistance with Large Language Models," May 2023.

[34] W. Huang, F. Xia, T. Xiao, H. Chan, J. Liang, P. Florence, A. Zeng, J. Tompson, I. Mordatch, Y. Chebotar, *et al.*, "Inner Monologue: Embodied Reasoning through Planning with Language Models," July 2022.

[35] X. Zhao, M. Li, C. Weber, M. B. Hafez, and S. Wermter, "Chat with the Environment: Interactive Multimodal Perception using Large Language Models," Mar. 2023.

[36] J. Yu, Y. Xu, J. Y. Koh, T. Luong, G. Baid, Z. Wang, V. Vasudevan, A. Ku, Y. Yang, B. K. Ayan, *et al.*, "Scaling autoregressive models for content-rich text-to-image generation," *arXiv*, 2022.

[37] C. Huang, O. Mees, A. Zeng, and W. Burgard, "Visual Language Maps for Robot Navigation," Mar. 2023.

[38] D. Shah, B. Osinski, B. Ichter, and S. Levine, "LM-Nav: Robotic Navigation with Large Pre-Trained Models of Language, Vision, and Action," July 2022.

[39] M. Ahn, A. Brohan, N. Brown, Y. Chebotar, O. Cortes, B. David, C. Finn, C. Fu, K. Gopalakrishnan, K. Hausman, *et al.*, "Do As I Can, Not As I Say: Grounding Language in Robotic Affordances," Aug. 2022.

[40] M. Attarian, A. Gupta, Z. Zhou, W. Yu, I. Gilitschenski, and A. Garg, "See, Plan, Predict: Language-guided Cognitive Planning with Video Prediction," Oct. 2022.

[41] W. Huang, P. Abbeel, D. Pathak, and I. Mordatch, "Language Models as Zero-Shot Planners: Extracting Actionable Knowledge for Embodied Agents," Mar. 2022.

[42] J. Liang, W. Huang, F. Xia, P. Xu, K. Hausman, B. Ichter, P. Florence, and A. Zeng, "Code as Policies: Language Model Programs for Embodied Control," May 2023.

[43] D. Surís, S. Menon, and C. Vondrick, "ViperGPT: Visual Inference via Python Execution for Reasoning," Mar. 2023.

[44] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, J. Hsu, *et al.*, "RT-1: ROBOTICS TRANSFORMER FOR REAL-WORLD CONTROL AT SCALE," Dec. 2022.

[45] D. Driess, F. Xia, M. S. M. Sajjadi, C. Lynch, A. Chowdhery, B. Ichter, A. Wahid, J. Tompson, Q. Vuong, T. Yu, *et al.*, "PaLM-E: An Embodied Multimodal Language Model," Mar. 2023.

[46] S. Reed, K. Zolna, E. Parisotto, S. G. Colmenarejo, A. Novikov, G. Barth-maron, M. Giménez, Y. Sulsky, J. Kay, J. T. Springenberg, *et al.*, "A Generalist Agent," *Transactions on Machine Learning Research*, Nov. 2022.

[47] Y. Jiang, A. Gupta, Z. Zhang, G. Wang, Y. Dou, Y. Chen, L. Fei-Fei, A. Anandkumar, Y. Zhu, and L. Fan, "VIMA: General Robot Manipulation with Multimodal Prompts," Oct. 2022.

[48] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. Chi, Q. Le, and D. Zhou, "Chain of thought prompting elicits reasoning in large language models," *arXiv*, 2022.

[49] W. Dai, J. Li, D. Li, A. M. H. Tiong, J. Zhao, W. Wang, B. Li, P. Fung, and S. Hoi, "Instructblip: Towards general-purpose vision-language models with instruction tuning," *arXiv preprint arXiv:2305.06500*, 2023.

[50] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "Nerf: Representing scenes as neural radiance fields for view synthesis," *Communications of the ACM*, vol. 65, no. 1, pp. 99–106, 2021.

[51] J. Kerr, C. M. Kim, K. Goldberg, A. Kanazawa, and M. Tancik, "LERF: Language Embedded Radiance Fields," Mar. 2023.

[52] M. A. Research, "Polymetis: A real-time pytorch controller manager," https://github.com/facebookresearch/fairo/tree/main/polymetis, 2021–2023.

[53] J. Carpentier, G. Saurel, G. Buondonno, J. Mirabel, F. Lamiraux, O. Stasse, and N. Mansard, "The pinocchio c++ library – a fast and flexible implementation of rigid body dynamics algorithms and their analytical derivatives," in *IEEE International Symposium on System Integrations (SII)*, 2019.

[54] J. Carpentier, F. Valenza, N. Mansard, *et al.*, "Pinocchio: fast forward and inverse dynamics for poly-articulated systems," https://stack-of-tasks.github.io/pinocchio, 2015–2023.

APPENDIX

A. Data Collection

Our data collection consists of three components:

1) Collecting the MESSYSURFACES dataset photos.
2) Asking crowdworkers to choose the most appropriate action in our benchmark questions.
3) Asking crowdworkers to evaluate parts of our framework.

1) Survey Interface: We show the survey interface we used to complete the 2nd and 3rd crowdsourcing components below:

The survey consists of a set of questions that we ask about each object, with a single page per object. An example page for the "mug" object is shown in Fig. 6 and Fig. 7. The first part of the survey asks users to rate the follow-up questions generated by the LLM; results are reported in Section 5 – Experiments in main body of the paper, under *"Does the LLM Ask Good Follow-Up Questions?"* The second part of the survey asks users to evaluate the informativeness of each close-up angle; results are also reported in Section 5, under *"Does the LLM Suggest Informative Close-Up Angles?"* The third part of the survey asks users to give ground truth answers to the follow-up questions based on all six images collected of the object; these answers are used as the Oracle VLM when evaluating our framework. The final part of the survey asks users to evaluate the appropriateness of each multiple choice option in the MESSYSURFACES benchmark and asks them to indicate the most appropriate way to tidy the object. These results are used to determine the correct answer for our benchmark questions as described in Section 4 of the main paper. We designed our survey using Flask and Python and hosted it on an AWS server.

2) Prolific Details: We recruited crowdworkers from Prolific to complete our study. The study took an average of 10 minutes and each crowdworker was paid $2 ($12/hour). We required workers to be fluent English speakers, based in the U.S. or U.K., and have a minimum approval rating of 98%. Each worker was in charge of answering survey questions about all objects belonging to a desk. We have a total of 70 desks and ran our framework 5 times, resulting in the recruitment of 350 Prolific workers.

B. Framework Implementation Details

In this section, we describe the design choices and implementation details of our framework.

Generating the Initial Description. In the first step of our framework (Section 3 in the main paper), we generate an initial description of the scene and append it to our context $\mathscr{C}^0$. The initial description is a list of all the objects in the scene. To ensure that we list the objects accurately, we generate the initial description using ground truth names of objects (see Listing 1 for an example).

```
"""
These are the objects on the desk:
    'scrunchie', 'lotion', 'vaseline', 'brush'.
"""
```

Listing 1. Example Initial Description

Structuring Follow-Up Questions. In the second step of our framework, we prompt an LLM to generate follow-up questions about information that it is missing in its context. We structure our follow-up questions to be yes-or-no questions where the LLM also has an option to choose "Cannot answer from image". We choose a yes-or-no question format to make it easier to evaluate the VLM's answers to these question. See subsubsection F.1 for the actual prompts used for the LLM.

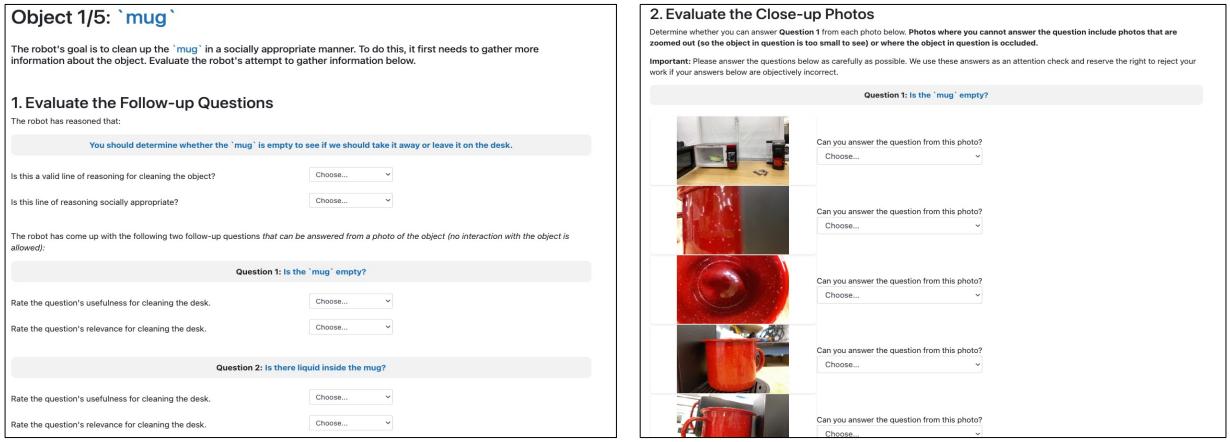
Eliciting Informative Close-Up Angles from an LLM. In the third step of our framework, we prompt an LLM to generate informative close-up angles that guide a photo-taking robot. We restrict the close-up angles the LLM can choose to a set of 5 angles: `<FRONT>`, `<BACK>`, `<LEFT>`, `<RIGHT>`, `<TOP>`. When querying the LLM, we format the prompt as a multiple choice question where the options are the five specified angles. See subsubsection F.1 for further prompting details.

C. Real-World Robot Evaluation

When implementing our grounded commonsense reasoning system on physical robot hardware (Fig. 8), there are two operating modes, reflecting the *active perception* and *skill execution* components of our approach respectively. As a preliminary, for the real-robot experiments, we assume that the object poses (in the coordinate frame of the robot's end-effector) are known a priori. While in this work we assume these poses are hand-specified by an expert, one could also use off-the-shelf perception systems that predict 6-DoF object poses or bounding boxes directly, as in prior work [33].

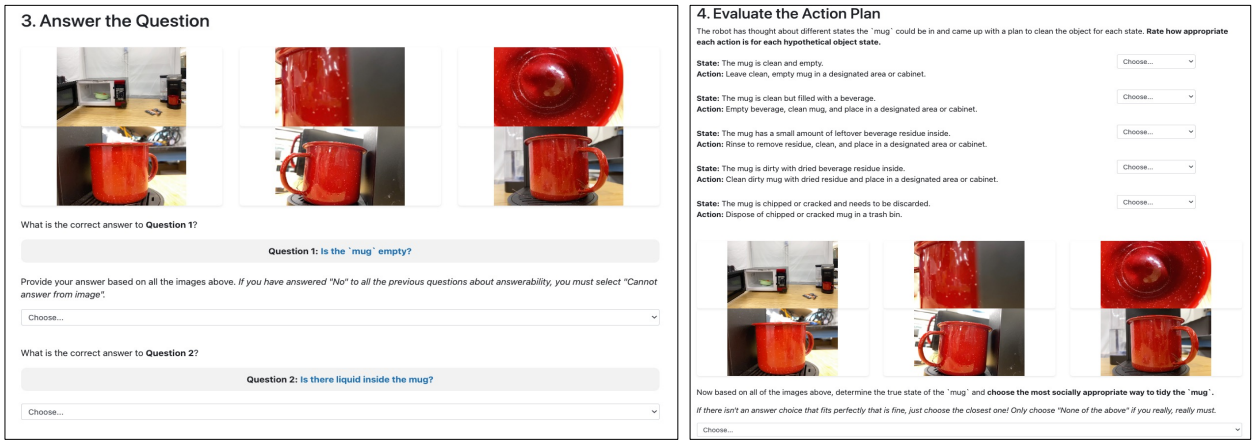
Active Perception Primitives. The active perception component of our framework requires the robot to execute on two types of behaviors, which we codify as functional primitives `move_<direction>(<object>)` and `take_photo()`. While the latter behavior is well-defined (capture an image at the robot's current position), the directional movement primitives vary *per-object*. As each object in our experiments is of different scale and composition, we manually define a set of pose transformations $p_{\text{dir}} \in \text{SE}(3)$ for each object and direction `<FRONT>`, `<BACK>`, `<LEFT>`, `<RIGHT>`, `<TOP>`. Given this dictionary of pose offsets, we implement `move_direction` for a specified object and desired direction by planning and executing a min-jerk trajectory from the robot's current location to the resulting pose after applying p_{dir} to the known object's pose.

Implementing Object-Centric Manipulation Skills. Similar to the perception primitives, we define each manipulation skill on a per-object basis as well; this is both due to the variety in object scale and properties, but also due to the variance in grasp locations for different desired behaviors. For example, the location where the robot should grasp an object such as a soda can may differ greatly depending on whether we want to throw the soda can away into a recycling bin (in which case the robot should grasp the soda can across the top), or if we want to relocate the can to a shelf (in which case the robot should grasp the soda can along the



Part 1/4 Part 2/4

Fig. 6. **Parts 1 and 2 of Survey Interface.**



Part 3/4 Part 4/4

Fig. 7. **Parts 3 and 4 of Survey Interface.**

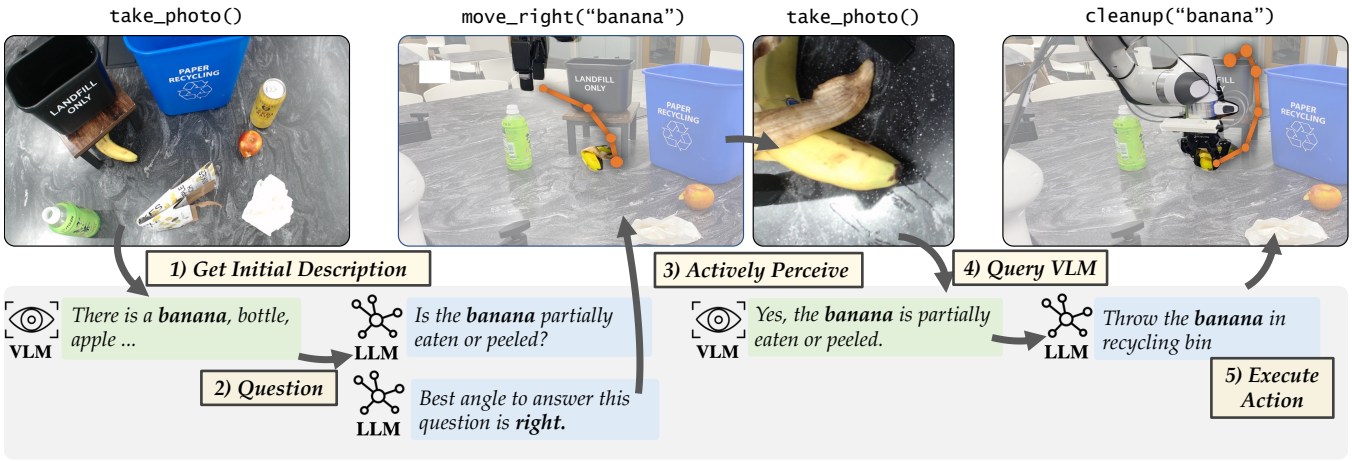


Fig. 8. **Real-World Commonsense Reasoning.** We outline the steps of our framework with a robot. Notably, the LLM generates questions and "angles" for the arm to servo to (e.g., *right of the banana*). We also use the LLM to generate an *action plan* for each object – each plan is converted to a sequence of skill primitives that are then executed by the robot.

side, to aid in insertion). To formalize this, we define a fixed interface depicted in Fig. 9. The provided API defines functions for each skill – for example, `relocate()` and `cleanup()` – at the *object-level*, with a stateful function

`set_designated()` that provides a compositional way to set target locations (i.e., "receptacles"). Fig. 9 (Right) shows the actual invoked API calls for the Kitchen Cleanup Desk depicted in Fig. 15.

We implement each object-oriented skill – `relocate()` and `cleanup()` – for a given object o_i and receptacle r_j as a tuple of pick-and-place waypoints defined as ($\text{pick}_{o_i} \in \text{SE}(3)$, $\text{place}_{r_j} \in \text{SE}(3)$; each pick/place point is defined as a transformation relative to the origin of the robot's reference frame. To execute on a "pick" waypoint, we plan a collision-free min-jerk trajectory to the given pose, and execute a blocking grasp action; similarly, to execute on a "place" waypoint, we plan a similar trajectory to the given receptacle pose, and execute a blocking open-gripper action. We run all experiments with a 7-DoF Franka Emika Panda robot manipulator equipped with a Robotiq 2F-85 gripper, using Polymetis [52] to facilitate real-time control and Pinocchio [53], [54] for trajectory planning.

Grounding Language to Skills. While the API definition deterministically defines robot behavior and skill execution in a given environment, we need a way of mapping natural langauge action plans generated by the LLM to sequence of API calls – for example, mapping the language action "dispose of the coffee cup" to the corresponding API calls `robot.set_designated(''recycling bin'');` `robot.cleanup(''coffee cup'');` `robot.done()`. To do this, we follow a similar procedure as in prior work using LLMs for code generation, prompting an LLM with the API definition, a series of in-context examples, and a continuation prompt for generating a valid sequence of API calls. The continuation prompt contains the set of known objects in the environment and valid receptacles defined for each skill, following prior work [39], [42]. The full prompt is in subsubsection F.5.

Evaluation. We add Fig. 10 to supplement our results described in Section 5 of the main paper.

D. VLM Details

We use pretrained visual-and-language models (VLMs) trained on massive internet scale images and texts to answer the questions generated by LLM. Following §B, we prompt the LLM so that it generates queries that can be easily answered by *yes*, *no* or *unknown*; these queries (and the respective images) are the inputs to the VLM.

To make it easier to parse the predictions of the VLM question-answerer, we rank the three answer options conditioned on the image and text input, rather than allowing the VLM to generate free-form responses. Specifically, we set the text prompt following Fig. 11. We use InstructBLIP [49] as our VLM and select the output with the highest predicted probability $P(\texttt{answer} \mid \texttt{prompt}, \texttt{image})$ for $\texttt{answer} \in \{\textit{yes}, \textit{no}, \textit{unknown}\}$ as the final answer. As InstructBLIP can use multiple LLM backbones, we evaluate both the Vicuna-13B and Flan-T5-XXL (11B) variants, finding Flan-T5-XXL to work better for our tasks. We have also experimented with further finetuning InstructBLIP on the in-domain data from the MessySurfaces dataset, but have not seen any noticeable performance gains; as a result, we use the off-the-shelf pretrained models in this work.

E. Personalization Analysis

We explore the hypothesis that incorporating personal preferences on how to clean objects can lead to a higher accuracy on our benchmark, as discussed in Sections 5 and 6 of the main paper. We studied questions that the human **Oracle** got incorrect in Section 5 of the main paper. Qualitatively, we found that some attributes of an object such as its "dirtiness" can be subjective, lending support to our hypothesis. This may have caused the **Oracle** to incorrectly answer some questions. For instance, in Question 6 of Fig. 12, the **Oracle** did not consider a keyboard that had a small amount of dust on it to be "dirty" enough and chose to "leave it as is". However, the majority of annotators preferred that the keyboard "should be cleaned".

We explored whether adding preferences would improve our framework's accuracy. We selected 9 questions where both the **Oracle** and our framework, **LLM-Ours**, chose the same incorrect answer. The full list of questions is shown in Fig. 12 and Fig. 13. We recruited a participant and, for each question, asked them whether the **Oracle** could have chosen the incorrect answer because of a lack of preference information. If the participant agreed that there was as lack of preference information, we asked them what the preference would be. For instance, in Question 6 of Fig. 12, the user noted that the disagreement between the human **Oracle** and human annotators could have been due to a lack of preference information, such as "It's not acceptable for objects to have any signs of dirtiness". The participant indicated that the **Oracle** could have incorrectly answered 8 out of the 9 questions due to a lack of preference information. Question 9 in Fig. 13 is an example of a question where the user thought the **Oracle** was incorrect due to noise.

For the remaining 8 questions, our goal was to see if adding preferences to the LLM's prompt would help the LLM choose the "correct" action plan as indicated by the annotators' majority label. We used 1 question to tune the prompt and evaluated the LLM on the remaining 7 questions (Questions $2-8$ in Fig. 12 and Fig. 13). We prompted the LLM by appending preference information to the prompt for choosing an action plan (described in subsubsection F.3). An example prompt is shown in Listing 2:

```
"""
The owner of the object has a preference on how you
     should tidy the 'candle': Don't trim the wick.
     It doesn't matter whether the burnt part of
    the candle wick is excessively long because I
    can still light it.

The best option is:
"""
```

Listing 2. Example Prompt for Generation an Action Plan with Preference Information

We found an average 86% improvement in accuracy, lending support to the hypothesis that preference information helps further enable grounded commonsense reasoning.

```
interface RobotManipulationInterface {
  // Leaves the <object> alone
  func leave_alone(object: str) -> None;

  // Sets the "designated receptacle" for following actions --> *stateful*
  func set_designated(receptacle: str) -> None;

  // Relocates / gathers the <object> and moves it to the designated receptacle
  func relocate(object: str) -> None;

  // Discards the <object> by placing it in the designated receptacle
  func cleanup(object: str) -> None;

  // Signals end of execution
  func done();
}

// Create a `robot` (callable instance of the interface)
robot = RobotManipulationInterface();
```

```
// "Clean up the kitchen counter"
objects = ["cardboard", "banana", "tissue", "bottle", "apple", "can"]
receptacles = {
  "cardboard": {"relocate": "countertop", "cleanup": "recycling bin"},
  "banana": {"relocate": "countertop", "cleanup": "trash can"},
  ...
}

// "Skill Execution"
robot.set_designated("recycling bin"); robot.cleanup("cardboard");
robot.set_designated("trash can"); robot.cleanup("banana");
robot.set_designated("trash can"); robot.cleanup("tissue");
robot.set_designated("recycling bin"); robot.cleanup("bottle");
robot.set_designated("trash can"); robot.cleanup("apple");

robot.leave_alone("can");

robot.done();
```

Fig. 9. **Code as Policies Interface for Real-Robot Execution.** We define a simple programmatic interface for specifying robot skill primitives on in an *object-oriented fashion*. The interface is stateful; for robot primitives such as `cleanup()` and `relocate()`, the robot sets a designated receptacle via the special function `set_designated()`. On the right, we provide the actual execution trace produced by the LLM for the Kitchen Cleanup Desk (see Fig. 15).

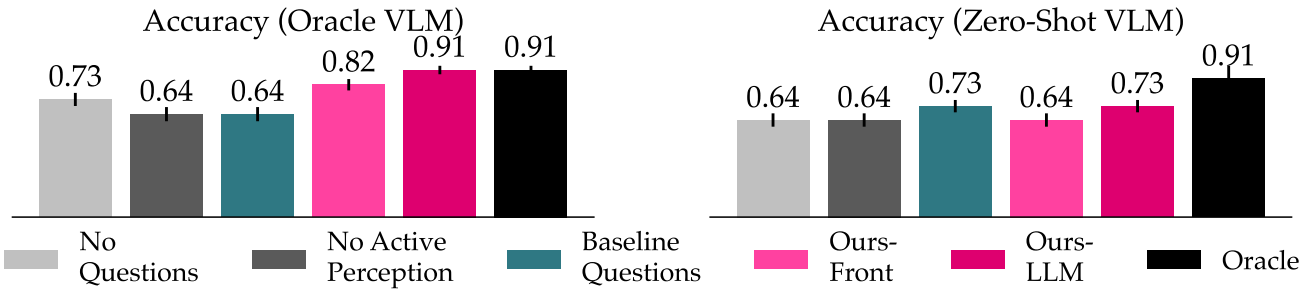


Fig. 10. **Real Robot Benchmark Accuracy**. We construct benchmark questions for objects used with the real robot in similar manner to Section 4 in the main paper. Across both types of VLMs, our **Ours-LLM** beats **Baseline Questions** by an average of 13.5%, beats **No Active Perception** by an average of 18%, and beats **No Questions** by an average of 13.5%.

```
"""
Given the image, please answer the following
    question in yes, no, or unknown.
Question: Is the bagel sandwich partially eaten?
Answer:
"""
```

Fig. 11. Example of VLM Text Prompt and Image Input.

F. Prompts & In-Context Examples for LLM Inference

In this section, we provide the comprehensive set of prompts used to elicit the desired behavior from the LLM (via the OpenAI API) across the multiple functionalities described in our approach, from generating follow-up questions, to synthesizing code for real-robot execution.

1) Prompt for Generating Follow-Up Questions: In the second step of our proposed framework (see Section 3 of the main paper), we one-shot prompt the LLM to generate follow-up questions about a list of objects on a surface using the prompt in Listing 3.

```
"""
These are the objects on the desk:
   'scrunchie', 'lotion', 'vaseline', 'brush'.

Your goal is to tidy the desk in a socially
    appropriate manner.
Ask a new follow-up question about each object to
    gather
more information. Only ask questions that can be
    answered by
taking a picture of the object. For example, DO NOT
     ask whether
the object is currently being used.
"""
```

Listing 3. Instruction For Generating Follow-Up Questions

To guide follow-up question generation, we provide the following (Listing 4) as the sole in-context example before having the LLM generate a continuation:

```
"""
These are the objects on the desk:
  'apple', 'charging cable', 'empty water bottle',
  'book', 'calendar', 'coffee cup'.

Your goal is to tidy the desk in a socially
    appropriate manner.
Ask a new follow-up question about each object to
    gather
more information. Only ask questions that can be
    answered by
taking a picture of the object. For example, DO NOT
     ask
whether the object is currently being used.

- 'Apple':
  Socially motivated reasoning: You should throw
    away the
```

Question 1: `tablet` (used as "training example" to tune prompt)

Oracle/LLM Answer	Detach accessories, put in sleep mode, and place tablet and accessories in a designated area.
Annotator Majority Answer (Correct Label)	Ensure the tablet in sleep mode is in a designated area.
Disagreement Due to Lack of Preferences?	Yes
Missing Preference	I prefer to keep my tablet accessories attached so I can continue charging them.

Question 2: `cup`

Oracle/LLM Answer	Wash and dry the cup with residue or stains.
Annotator Majority Answer (Correct Label)	Leave the empty, clean cup as is.
Disagreement Due to Lack of Preferences?	Yes
Missing Preference	Leave cups that don't appear visibly dirty.

Question 3: `controller`

Oracle/LLM Answer	Clean the controller with a soft cloth or cleaning solution, then place it in a designated area.
Annotator Majority Answer (Correct Label)	Leave the controller as is on the stand or designated area.
Disagreement Due to Lack of Preferences?	Yes
Missing Preference	It's acceptable to have some dust on objects.

Question 4: `keyboard`

Oracle/LLM Answer	Leave the properly placed and connected keyboard as is.
Annotator Majority Answer (Correct Label)	Clean the dirty or dusty keyboard and place it in a convenient location.
Disagreement Due to Lack of Preferences?	Yes
Missing Preference	It's not acceptable for objects to have any signs of dirtiness.

Question 5: `mouse`

Oracle/LLM Answer	Leave the properly placed and connected mouse as is.
Annotator Majority Answer (Correct Label)	Clean the dirty or dusty mouse and place it in a convenient location.
Disagreement Due to Lack of Preferences?	Yes
Missing Preference	It's not acceptable for objects to have any signs of dirtiness.

Question 6: `keyboard`

Oracle/LLM Answer	Leave keyboard as is, ensuring it is placed neatly.
Annotator Majority Answer (Correct Label)	Clean the dirty keyboard and place it neatly.
Disagreement Due to Lack of Preferences?	Yes
Missing Preference	It's not acceptable for objects to have any signs of dirtiness.

Fig. 12. **Questions Used for Personalization Analysis (1/2).** We display questions where both **Oracle** and **Ours-LLM** chose the same incorrect answer. We recruited a participant to indicate whether the **Oracle** could have incorrectly answered these questions due to a lack of preference information, and if so, what the preference would be.

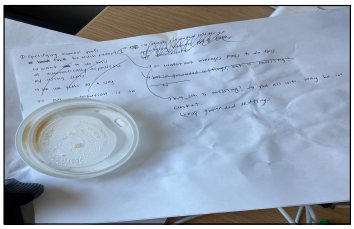
Question 7: `lamp`

Oracle/LLM Answer	Turn on the lamp if needed.
Annotator Majority Answer (Correct Label)	Set the turned-off lamp upright.
Disagreement Due to Lack of Preferences?	Yes
Missing Preference	It's better to keep the light off!

Question 8: `candle`

Oracle/LLM Answer	Trim the burnt wick and place the used candle in a designated area.
Annotator Majority Answer (Correct Label)	Leave the clean, unlit candle as is and place it in a designated area.
Disagreement Due to Lack of Preferences?	Yes
Missing Preference	It doesn't matter whether the burnt part of the wick is excessively long because I can still light it.

Question 9: `papers` (omitted)



Oracle/LLM Answer	Flatten crumpled papers or trim torn edges, organize, and store them in a designated area.
Annotator Majority Answer (Correct Label)	Gather, organize, and store the scattered papers in a designated area.
Disagreement Due to Lack of Preferences?	No, the Oracle/LLM answer is incorrect
Missing Preference	N/A

Fig. 13. **Questions Used for Personalization Analysis (2/2).** We display questions where both **Oracle** and **Ours-LLM** chose the same incorrect answer. We recruited a participant to indicate whether the **Oracle** could have incorrectly answered these questions due to a lack of preference information, and if so, what the preference would be.

```
    'apple' if it is partially eaten, but not if it
     is intact.

  Resulting question (that can be answered by
    taking a
    picture of object): Is the 'apple' partially
    eaten?

    (a) Yes (b) No (c) Cannot answer from image

-'Charging cable':
  Socially motivated reasoning: You should coil the
    'charging cable' and store it neatly if it is
    not in use,
    but leave it in place if it is connected to a
    device that
    needs charging.

  Resulting question (that can be answered by
    taking a
  picture of object): Is the 'charging cable'
    connected to a device?

  (a) Yes (b) No (c) Cannot answer from image

...
"""
```

Listing 4. In-Context Example For Generating Follow-Up Questions

Notably, we use Chain-of-Thought prompting to encourage the LLM to generate questions that are motivated by commonsense reasoning. We also encourage the LLM to ask questions that can be answered by an image of the object.

Prompt for Generating Baseline Follow-Up Questions.

To generate baseline questions, we use the following prompt (Listing 5):

```
"""
Ask one yes-or-no question for each object on the
    desk. Only ask
yes-or-no questions that can be answered by taking
    a picture of the object.

These are the objects on the desk:
    'scrunchie', 'lotion', 'vaseline', 'brush'.

Format your answer in the following format: '
    object_name': question
"""
```

Listing 5. Instruction For Generating Baseline Follow-Up Questions

In our baseline question prompt, we do not specify that the goal for the LLM is to tidy the desk nor do we require the LLM to generate commonsense-motivated questions.

2) Prompt for Choosing a Close-Up Angle: In the third step of our proposed framework, we few-shot prompt the LLM to generate informative close-up angles that would guide a robot. In the prompt, we include a list of objects on the current surface, the follow-up question about an object, and a multiple choice question with options corresponding to the five predefined close-up angles: `<FRONT>`, `<BACK>`, `<LEFT>`, `<RIGHT>`, `<TOP>`. We use the following prompt (Listing 6):

```
"""
Description: These are the objects on the desk:
    'computer monitor', 'cup', 'computer wires', '
    apple'.

Follow-up Question: Are the 'computer wires'
    connected to anything?
    (a) Yes (b) No

You are instructing a robot to take a close-up
    picture of the object
to help answer the follow-up question.

Which of the following angles would yield a close-
    up picture that can
best answer the question?

(a) Top of the object
(b) Right side of the object
(c) Left side of the object
(d) Front of the object
(e) Behind the object

Response: A top-down view would give an unoccluded
    view since the
wires might be tangled.

(a) Top of the object

Description: These are the objects on the desk:
    'monitor', 'stack of papers', 'cups'.

Follow-up Question: Are the 'cups' empty?
    (a) Yes (b) No

You are instructing a robot to take a close-up
    picture of the object
to help answer the follow-up question.

Which of the following angles would yield a close-
    up picture that can
best answer the question?

(a) Top of the object
(b) Right side of the object
(c) Left side of the object
(d) Front of the object
(e) Behind the object

Response: The cups might be opaque so the best
    angle would be

(a) Top of the object

Description: These are the objects on the desk:
    'keyboard', 'whiteboard marker', 'stack of
    papers', 'vase'.

Follow-up Question: Are the 'stack of papers'
    straightened?
    (a) Yes (b) No

You are instructing a robot to take a close-up
    picture of the object
to help answer the follow-up question.

Which of the following angles would yield a close-
    up picture that can
best answer the question?

(a) Top of the object
(b) Right side of the object
(c) Left side of the object
(d) Front of the object
(e) Behind the object

Response: The stack would best be viewed from its
    side.

(d) Front of the object
```

```
"""
```

Listing 6. Prompt for Generating Informative Close-Up Angles

3) Prompt for Choosing an Action Plan: As the ultimate step of our framework, we prompt the LLM to answer our benchmark questions by choosing the most socially appropriate action to tidy each object. When prompting the LLM, we first include the context accumulated so far: the follow-up questions and their VLM-generated answers (see Listing 7 for an example).

```
"""
Here is some information about the `scrunchie` in
question-answer format.

Is the `scrunchie` neatly placed on the desk? Yes
Does the `scrunchie` have any stains? Yes
Does the `scrunchie` have any loose threads? No
"""
```

Listing 7. Example of Context for Action Plan Generation

We append the benchmark question to the prompt and have the LLM generate an appropriate tidying action:

```
"""
Based on the information above, what is the most
    appropriate
way to tidy the `scrunchie`?

Choose the best option.
(a) The scrunchie is neatly coiled and placed on
    the desk.
  -> Leave the neatly coiled scrunchie as is in a
    designated area.
(b) The scrunchie is stretched out and tangled with
     other
  items on the desk.
  -> Untangle, coil neatly, and place in a
    designated area.
(c) The scrunchie is dirty or stained and needs to
    be cleaned.
  -> Clean, dry, and place in a designated area.
(d) The scrunchie is partially unraveled or damaged
    .
  -> Repair or replace, and place in a designated
    area.
(e) The scrunchie is being used to hold together a
    bundle
  of cables or cords on the desk.
  -> Remove from cables, coil neatly, and place in
    a designated area.

The best option is:
"""
```

Listing 8. Prompt For Generating Answers to Benchmark Questions

4) Prompt for Generating MESSYSURFACES *Benchmark Questions:* As described in Section 3 of the main paper, we prompt an LLM to generate multiple choice options for the question *"What is the most appropriate way to tidy the object?"* for each object in the MESSYSURFACES dataset. To generate each set of multiple choice options, we first prompt the LLM to list five possible states each object could be in:

```
"""
These are the objects on the desk:
    `scrunchie`, `lotion`, `vaseline`, `brush`.

Your goal is to tidy each `object` up, but there is
     not
enough information about each object. For each `
    object`,
list 5 possible states the object could be in that
    would
affect how you tidy it up.

Label the 5 states (a)-(e). Make sure each state is
significantly different from each other. Remember
    that
all the objects are placed on the desk.
"""
```

Listing 9. Example Prompt For Generating Benchmark Questions (1/2)

After receiving the LLM's response, we ask it to generate a cleaning action for each state. The purpose of first asking it to generate object states is so that the LLM can generate diverse cleaning actions:

```
"""
For each state (a)-(e), tell me how you would tidy
    the `object`.
Make sure each answer choice is significantly
    different from each
other. Include an option to 'leave the object as is
    '.
Each object should be in apostrophes like so: `
    object`.
"""
```

Listing 10. Example Prompt For Generating Benchmark Questions (2/2)

5) Prompt for Real-Robot Code Generation from Language: Following §C, we use the LLM to generate valid API calls for a given natural language action (e.g., "dispose of the coffee cup"). To do this, we use the following instruction prompt for GPT-3 that defines the interface formally:

```
INITIAL_INSTRUCTION = (
    """
    Translate each of the following language
    instructions to a
    sequence of predefined API calls that will be
    executed by
    a robot manipulator to help "clean up" a
    workspace.
    When generating code, make sure to use the API
    provided below:
    """
)

ROBOT_API = (
    """
    interface RobotManipulationInterface {
        // Leaves the <object> alone
        func leave_alone(object: str) -> None;

        // Sets the "designated receptacle" for
        // following actions --> *stateful*
        func set_designated(receptacle: str) ->
    None;

        // Relocates / gathers the <object> and
    moves it to the
        // designated receptacle
        func relocate(object: str) -> None;

        // Discards the <object> by placing it in
    the
        // designated receptacle
        func cleanup(object: str) -> None;

        // Signals end of execution
        func done() -> None;
    }
```

```
    // Create a `robot` (callable instance of interface)
    robot = RobotManipulationInterface();
    """
)

API_DOCS = (
    """
    You can invoke a given function on the robot by calling
    `robot.<func>("object name"). For example:
    `robot.set_designated_area("recycling bin")`.

    The API also enables multiple function invocations (separated
    by newlines).

    Note that each call to `relocate` and `cleanup` *must* be preceded
    by a call to `set_designated` to be valid!

    To terminate execution for a given action, call `robot.done()`.
    """
)
```

Listing 11. Prompt for Generating Real-Robot API Calls from Natural Language Actions

In addition to this API definition, we provide three in-context examples in the prompt, as follows:

```
ICL_INSTRUCTION = (
    """
    Here are some examples of translating language instructions to
    API calls. Each instruction defines two variables:

    1) a list of interactable `Objects: ["obj1", "obj2", ...]`
    --> these should be the only "object" arguments to the
    `relocate` and `cleanup` API calls!

    2) a mapping of objects to receptacles `Receptacles:
       {"obj": {"relocate": "<receptacle>", "cleanup": "<receptacle>"}}`
       --> these should be the only "receptacle" arguments for the
       `set_designated` API calls!

    Note that if there is *not* a good API call that reflects the
    desired behavior, it is ok to skip!
    """
)

EXAMPLE_ONE = (
    """
    Instruction: "Retrieve all the crayons and organize them
    tidily in the designated container."
    Objects: ["crayons", "colored pencils", "notebook", "eraser",
              "crumpled up napkin"]
    Receptacles: {
      "crayons": {"relocate": "art box", "cleanup": "trash"},
      "notebook": {"relocate": "desk", "cleanup": "recycling"},
      "eraser": {"relocate": "art box", "cleanup": "trash"},
      "crumpled up napkin": {"relocate": "desk", "cleanup": "trash"}
    }
    Program:
    ```
    robot.set_designated("art box");
    robot.relocate("crayons");
    robot.done();
    ```
    """
)

EXAMPLE_TWO = (
    """
    Instruction: "Throw away the half-eaten apple."
    Objects: ["apple", "orange", "half-eaten peach",
              "coffee cup", "pink plate"]
    Receptacles: {
      "apple": {"relocate": "counter", "cleanup": "trash"},
      "orange": {"relocate": "counter", "cleanup": "trash"},
      "half-eaten peach": {"relocate": "counter", "cleanup": "trash"},
      "coffee cup": {"relocate": "counter", "cleanup": "recycling"},
      "pink plate": {"relocate": "counter", "cleanup": "sink"}
    }
    Program:
    ```
    robot.set_designated("trash can");
    robot.cleanup("apple");
    robot.done();
    ```
    """
)

EXAMPLE_THREE = (
    """
    Instruction: "Leave the castle as is in a designated area, then
                  put away the removeable parts in a continer."
    Objects: ["toy castle", "castle parts", "figurine", "cheerios"]
    Receptacles: {
      "toy castle": {"relocate": "shelf", "cleanup": "toy box"},
      "castle parts": {"relocate": "play mat", "cleanup": "toy box"},
      "figurine": {"relocate": "shelf", "cleanup": "toy box"},
      "cheerios": {"relocate": "play mat", "cleanup": "trash"}
    }
    Program:
    ```
    robot.leave_alone("toy castle");
    robot.set_designated("toy box");
    robot.cleanup("castle parts");
    robot.done();
    ```
    """
)
```

Listing 12. In-Context Examples for Real-Robot Code Generation

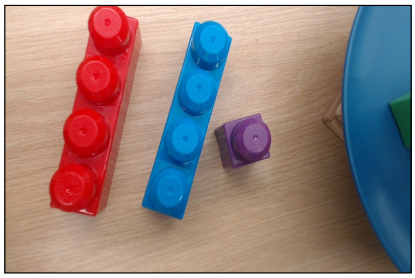
Finally, we produce the following continuation string that we use to seed the LLM completion; the $\{,\}$ denote variables that are filled in programmatically at runtime:

```
CONTINUATION = (
    "
    Instruction: {instruction}
    Objects: {objects}
    Receptacles: {receptacles}
```

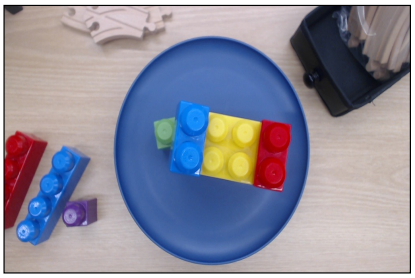
```
    Program:
    ```
    " +
    """< LLM CONTINUATION >"""
)
```

Listing 13. Continuation Prompt for Code Generation

### G. *Examples*

We list several examples our dataset and framework below.
```

Children's Play Desk (Real Robot Experiments)

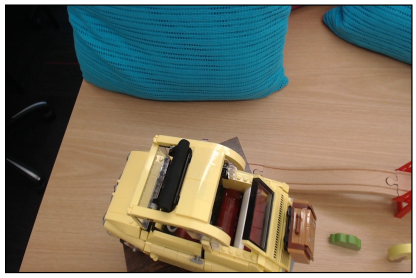
scene



blocks



car



sports car

toy train set

track pieces

Iteration 1

Social Reasoning → Question	You should store the `toy train set` in a designated area if it is not being actively played with, but leave it on the desk if it is in use.→ Are the `toy train set` pieces connected and arranged in a layout?
LLM Angle	Top
VLM Answer	Yes
Bl. Question	Is the toy train set made of plastic?

Iteration 2

Social Reasoning → Question	You should check if the `toy train set` is complete or if any pieces are missing, as it may need to be reassembled or stored properly.→ Are all the `toy train set` pieces present?
LLM Angle	Top
VLM Answer	Yes
Bl. Question	Is the toy train set battery-operated?

Iteration 3

Social Reasoning → Question	You should check if the `toy train set` has a designated storage container or area, as it may need to be stored properly when not in use.→ Does the `toy train set` have a designated storage container or area?
LLM Angle	Front
VLM Answer	No
Bl. Question	Are there any passenger cars in the toy train set?

Iteration 4

Social Reasoning → Question	You should check if the `toy train set` has any battery-operated components, as batteries may need to be removed or replaced for proper storage or maintenance.→ Does the `toy train set` have any battery-operated components?
LLM Angle	Top
VLM Answer	No
Bl. Question	Is there a caboose at the end of the toy train set?

Iteration 5

Social Reasoning → Question	You should check if the `toy train set` has any additional accessories or components that should be kept together with the set for proper organization and storage.→ Does the `toy train set` have any additional accessories or components?
LLM Angle	Top
VLM Answer	Yes
Bl. Question	Does the toy train set have any cargo cars?

Benchmark Question

(a) Leave the arranged train set as is in a designated area. ✅
(b) Connect disconnected pieces and arrange neatly in a designated area.
(c) Assemble scattered pieces and arrange neatly in a designated area.
(d) Leave the train set in its packaging in a designated area.
(e) Repair or replace damaged pieces and arrange neatly in a designated area.

Fig. 14. **Children's Play Desk (Real-Robot Experiments).** Example photos of each object (from a top-down angle). We also provide examples of our framework's outputs for the *toy train set* for all 5 iterations using InstructBLIP.

Kitchen Cleanup Desk (Real Robot Experiments)

scene

apple

banana

bottle

can

cardboard

Iteration 1

Social Reasoning → Question	You should throw away the `banana` if it is partially eaten or the peel is discarded, but not if it is intact.→ Is the `banana` partially eaten or peeled?
LLM Angle	Right
VLM Answer	Yes
Bl. Question	Is the banana ripe (yellow)?

Iteration 2

Social Reasoning → Question	You should check if the `banana` is overripe, as it may need to be thrown away or used soon.→ Is the `banana` overripe?
LLM Angle	Front
VLM Answer	Yes
Bl. Question	Is the banana's peel intact (no bruises)?

Iteration 3

Social Reasoning → Question	You should check if the `banana` is starting to brown, as it may need to be consumed soon or used in a recipe..→ Is the `banana` starting to brown?
LLM Angle	Front
VLM Answer	Yes
Bl. Question	Are there any brown spots on the banana's peel?

Iteration 4

Social Reasoning → Question	You should check if the `banana` has any mold, as it may need to be discarded immediately.→ Does the `banana` have any mold?
LLM Angle	Top
VLM Answer	No
Bl. Question	Is the banana's peel starting to split open?

Iteration 5

Social Reasoning → Question	You should check if the `banana` has any fruit flies around it, as it may need to be discarded or stored properly..→ Are there any fruit flies around the `banana`?
LLM Angle	Top
VLM Answer	No
Bl. Question	Is there any visible stringiness on the banana's exposed flesh?

Benchmark Question

(a) Throw in recycling bin.
(b) Throw in landfill bin. ✅
(c) Leave as is

Fig. 15. **Kitchen Cleanup Desk (Real-Robot Experiments).** Example photos of each object (from a top-down angle). We also provide examples of our framework's outputs for the *banana* for all 5 iterations using InstructBLIP.