LoHoRavens: A Long-Horizon Language-Conditioned Benchmark for Robotic Tabletop Manipulation

Shengqiang Zhang [1,3], **Philipp Wicke** [1,3], **Lütfi Kerem Şenel** [1,3],
Luis Figueredo [2], **Abdeldjallil Naceri** [2], **Sami Haddadin** [2], **Barbara Plank** [1,3], **Hinrich Schütze** [1,3]
[1] CIS, LMU Munich [2] RSI, MIRMI, TUM
[3] Munich Center for Machine Learning (MCML)
shengqiang@cis.lmu.de, pwicke@cis.lmu.de

Abstract

The convergence of embodied agents and large language models (LLMs) has brought significant advancements to embodied instruction following. Particularly, the strong reasoning capabilities of LLMs make it possible for robots to perform long-horizon tasks without expensive annotated demonstrations. However, public benchmarks for testing the long-horizon reasoning capabilities of language-conditioned robots in various scenarios are still missing. To fill this gap, this work focuses on the tabletop manipulation task and releases a simulation benchmark, *LoHoRavens*, which covers various long-horizon reasoning aspects spanning color, size, space, arithmetics and reference. Furthermore, there is a key modality bridging problem for long-horizon manipulation tasks with LLMs: how to incorporate the observation feedback during robot execution for the LLM's closed-loop planning, which is however less studied by prior work. We investigate two methods of bridging the modality gap: caption generation and learnable interface for incorporating explicit and implicit observation feedback to the LLM, respectively. These methods serve as the two baselines for our proposed benchmark. Experiments show that both methods struggle to solve most tasks, indicating long-horizon manipulation tasks are still challenging for current popular models. We expect the proposed public benchmark and baselines can help the community develop better models for long-horizon tabletop manipulation tasks.[1]

1 Introduction

In embodied instruction following, an embodied agent such as a robot is given a language based instruction and expected to follow the instruction to complete the designated task. Of particular interest is long-horizon instruction following: how to endow embodied agents with long-horizon instruction following capabilities attracts more and more attention. The long-horizon task usually includes a quite high-level instruction and cannot be completed in just a few steps. Thus, the embodied agent must understand the language instruction well and perform long-horizon memorizing and complex reasoning.

This work focuses on the long-horizon language-conditioned robotic tabletop manipulation task. To develop better robots for long-horizon manipulation tasks, good benchmarks are essential to test their capabilities. However, most current benchmarks either do not focus on long-horizon tasks (e.g., Meta-World [25], RLBench [15], Ravens [26, 23], Robo-

[1]The video and code of LoHoRavens are available at https://cisnlp.github.io/lohoravens-webpage/.

NeurIPS 2023 Workshop on Robot Learning: Pretraining, Fine-Tuning, and Generalization with Large Scale Models, New Orleans, United States of America

suite [28], VIMA-Bench [16]), or are not language-conditioned (e.g., FurnitureBench [13], Causal-World [1]), or are not open-sourced (e.g., Inner Monologue [14], CoP [18], Language-Table [20]). CALVIN [21] is a long-horizon language-conditioned public benchmark, but step-by-step instructions are provided to complete each long-horizon task, without the need for any long-horizon reasoning by the robot.

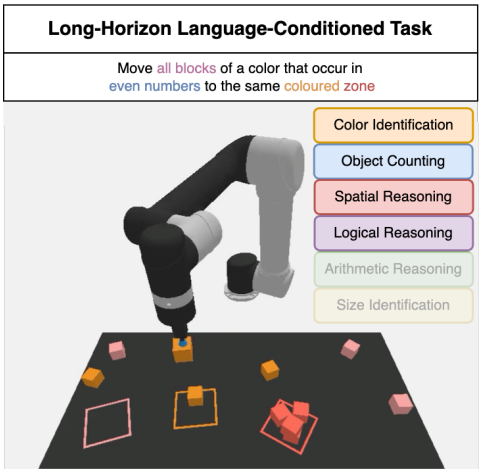

Figure 1: A long-horizon language-conditioned task example.

To fill this gap, we develop a long-horizon language-conditioned simulated benchmark, called **LoHoRavens**, for robotic tabletop manipulation tasks and open-source it. LoHoRavens is built based on the Ravens robot simulator and contains ten long-horizon language-conditioned tasks in total. LoHoRavens first requires the robot agent to understand the deep semantics of each high-level instruction well. Then LoHoRavens covers various long-horizon reasoning aspects including color, size, space, arithmetics and reference. To solve each task, the robot must combine several of the reasoning capabilities and develop its long-horizon plan accordingly. Fig. 1 gives an example of a long-horizon task that requires reasoning capabilities that go beyond a simple pick-and-place task. LoHoRavens also further boosts the complexity of each task by perturbing the environment to increase the probability of execution failure, such that the robot has to incorporate real-time observation feedback for the long-horizon planning.

To solve the challenging LoHoRavens benchmark tasks, a key modality bridging problem arises: although using LLMs as planners has been a popular method in robotics [2, 10, 6, 7], how to incorporate the observation feedback during the robot's execution for the LLM's closed-loop long-horizon planning is still an under-explored problem. In this work, we investigate two methods for modality bridging: the *explicit method* of caption generation (see Fig. 2) and the *implicit method* of learnable interface (see Fig. 3). Explicit/implicit here refers to whether the observation feedback is given in the form of explicit (human-readable) natural language or in the form of an implicit (non-human-readable) representation of the observation feedback.

The experiments on LoHoRavens benchmark show that the proposed two baselines have a strong positive impact on long-horizon manipulation task performance. But both methods still struggle to solve most of the long-horizon tasks. The experimental results indicate that the long-horizon language-conditioned manipulation tasks are still challenging for current popular models. We hope our LoHoRavens benchmark and the two baselines can help with developing more advanced robots.

2 LoHoRavens Benchmark

As far as we know, LoHoRavens is the first public benchmark for long-horizon language-conditioned robotic tabletop manipulation tasks without giving step-by-step instructions for the high-level goal of each task. **LoHoRavens** is built based on the **Ravens** robot simulator by extending it to **Lo**ng-**Ho**rizon tasks. In the LoHoRavens simulation environment, there are a UR5e robot arm with a suction gripper and some objects on the table. Given a high-level language based instruction (e.g., "stack all the blocks of the same size"), the robot is supposed to rearrange these objects to a desired state.

Currently, LoHoRavens contains ten long-horizon tasks in total (see Table 1). In addition to the pick-and-place primitive, we borrow two interesting tasks (tasks B and F) from Inner Monologue and CoP, and design another eight long-horizon tasks by ourselves. LoHoRavens only contains three kinds of objects: block, bowl, and zone (see Fig. 1) because we do not want to test the robot's generalization capability to new or unseen objects in this work. Instead, we focus on the long-horizon reasoning capabilities which are related to the general attributes of objects like size, color and spatial position. Such reasoning capabilities can be generalized to other objects as well. In addition to these general object attributes, we are also interested in the reasoning capabilities related to attributes of

Table 1: LoHoRavens benchmark tasks and the experimental results of the two baselines.

	LoHoRavens Tasks		Explicit feedback			Implicit feedback
			CLIPort (oracle)	+Llama 2	+Open Flamingo	LLaVA
Seen tasks	A.	Pick-and-Place primitives	61.2	67.3	67.3	67.3
	B.	"Put the blocks in the bowls with matching colors"	19.7	27.9	31.4	37.0
	C.	"Stack smaller blocks over bigger blocks of the same color"	12.1	17.5	18.0	22.1
	D.	"Stack all the blocks in the [ABS_POS] area"	22.5	28.8	30.4	35.8
	E.	"Move all blocks of a color that occur in even numbers to the same colored zone"	13.4	9.1	9.6	8.2
Unseen tasks	F.	"Put the blocks in the bowls with mismatching colors"	17.3	24.8	28.5	21.1
	G.	"Stack blocks of the same size"	2.1	15.8	21.9	14.7
	H.	"Stack blocks in alternate colors"	1.8	8.7	13.2	5.2
	I.	"Stack blocks of the same color in the zone with same color, with the bigger blocks underneath"	8.5	13.6	12.8	11.7
	J.	"Move all the blocks in the [ABS_POS] area to the [ABS_POS] area"	15.1	19.7	27.4	27.2
	K.	"Stack blocks of the same color, given there are multiple blocks with the same color"	6.7	3.5	4.0	6.8

multiple objects. So we include several tasks to test arithmetic and reference reasoning capabilities (e.g., tasks E and K). More details about each task can be found in Appendix D.

To simulate the disturbance in the real world, we add noises and perturbations to the robot's environment at test time. Following Inner Monologue [14], we add Gaussian noise $\mathcal{N}(0, 3)$ for pixel observations and $\mathcal{N}(0, 2.5)$ for policy primitives. Moreover, we add a dropping probability p for the end-effector to drop the picked block every second following DoReMI [11].

More details about the including generated dataset and evaluation method can be found in Appendix A.

3 Baselines

It has been a mainstream method to use LLMs as the planner for a robot's execution. However, how to incorporate real-time visual observation feedback into the LLM's input is still an under-explored problem. This modality gap is especially severe for long-horizon robotic tasks because an execution error in each of the robot's steps can affect all the following steps. To solve this modality bridging problem, we propose two baseline methods to translate the visual observation into feedback that the LLM can understand for its closed-loop planning. We use the Planner-Actor-Reporter paradigm introduced by [9] to unify our two baselines. The feedback generation models of the two baselines are working as the Reporter module.

Explicit feedback: Caption generation Inner Monologue [14] demonstrated that human-provided language feedback can significantly improve high-level instruction completion on robotic manipulation tasks. But human-written language feedback is too expensive to scale. We therefore explore a caption generation based model as an automatic way to generate language feedback without training.

As shown in Fig. 2, we use Llama 2 13B [24] and the trained pick-and-place CLIPort primitive as the Planner and Actor, respectively. For the Reporter, we use the VLM OpenFlamingo [4, 5, 3] with few-shot prompting to generate the following two types of feedback: *Observation state feedback* which is the information about the objects on the table and their potential changes, and *Action success state feedback* which is the description whether the last instruction is executed successfully or not.

When a step's action has executed, there will be a top-down RGB image rendered by the simulator. The VLM as the Reporter module will generate the caption feedback based on the current image or the whole image history. This caption feedback is sent to the LLM for its next-step planning. The Planner-Actor-Reporter closed-loop process will be iteratively executed until the high-level goal is achieved or the maximum number of trial steps has been exceeded.

Implicit feedback: Learnable interface Explicitly converting an image to language captions is straightforward and simple. However, it typically causes information loss [27, 17] and exaggerates bias present in training data [12]. On the other hand, training an end-to-end multimodal LLM would be too expensive. Thus another common solution used in many VLMs is to use a learnable interface such as a projection-based interface [19] or a group of learnable query tokens [8] to connect vision and language modalities while freezing parameters of the LLM and the visual encoder.

We use LLaVA [19] for this second baseline. LLaVA uses the simple projection-based scheme as the learnable interface between the vision model and the pretrained LLM. As shown in Fig. 3, the pretrained CLIP visual encoder ViT-L/14 [22] encodes the observation image to visual embeddings. A single-layer MLP as the learnable interface then translates the visual embeddings to the LLM's token embedding space. The LLM will generate the next-step plan conditioned on the language instruction prompts and the translated visual embeddings. LLaVA uses LLaMA as the LLM.

To fine-tune LLaVA, for each step of the task instances in the train set, we use the oracle program of the simulator to generate the image before the step and the language instruction for the step as the pair of train data. For the inference process, LLaVA receives the generated images after each step's execution (just as the caption generation based model does). LLaVA then outputs the next-step language instruction to CLIPort for execution.

4 Experiments

In this section, we aim to answer the following two questions:

(1) Is our proposed LoHoRavens benchmark a challenging benchmark for current popular models?

(2) Which method of incorporating the visual observation feedback to LLMs is better for long-horizon robotic manipulation tasks: implicit or explicit?

Table 1 gives experimental results. The results show that the performance of all models is quite poor on most tasks, which indicates LoHoRavens is a quite challenging benchmark for current popular LLMs and VLMs. We find that all models perform better on tasks requiring reasoning about only one aspect/attribute (e.g., tasks B and D) than on tasks involving several (e.g., size and color in task C, arithmetics and color in task E). Combining several types of reasoning capabilities is apparently challenging for the models.

Comparing the results of `CLIPort (oracle)`, `CLIPort + Llama 2`, and `CLIPort + Llama 2 + OpenFlamingo`, we find both LLM and VLM usually improve the single CLIPort model. The VLM is especially helpful when execution errors are likely to occur, such as the stacking tasks G and H where an error in one step such as dropping a block may easily affect previously stacked blocks. However, we also notice the LLM brings negative effects in some tasks requiring reference capability (like tasks E and K). We conjecture it is because the LLM cannot give the precise description to indicate which block should be manipulated when there are several objects of the same size and color.

We also see that the learnable interface based model can outperform the caption based model in tasks where the observations are complex in the sense that they are difficult to describe in language. For example, in task B, there are too many objects of the same size and similar color to recognize. In task D, some objects are unobservable if other objects are stacked on them. This may be the reason that the VLM fails to give an accurate description for the image in these situations. But `LLaVA` has been trained on the images of the LoHoRavens environment, so it would be more competent to deal with these complicated images than the caption model without training.

Furthermore, when transferring to the unseen tasks, both the performance of the caption-based model and the learnable interface-based model drops noticeably. However, we find the caption-based model is more robust to the unseen tasks than the learnable interface model. We think the reason is that the training-free caption-based model is less affected by whether the task is new or not.

Discussions Our findings suggest that LoHoRavens can guide research on several of the main challenges in this area: (i) how to design models with reasoning abilities, (ii) how best to provide feedback for planner/actor, (iii) how to represent information that is difficult to describe in language, (iv) how best to achieve good generalization for unseen tasks.

Acknowledgments

We would like to thank helpful discussions from Ruotong Liao and Gengyuan Zhang at LMU Munich. This work was partially funded by the European Research Council (grant #740516). This work was also supported by the DAAD programme Konrad Zuse Schools of Excellence in Artificial Intelligence, sponsored by the Federal Ministry of Education and Research.

References

[1] Ossama Ahmed, Frederik Träuble, Anirudh Goyal, Alexander Neitz, Yoshua Bengio, Bernhard Schölkopf, Manuel Wüthrich, and Stefan Bauer. Causalworld: A robotic manipulation benchmark for causal structure and transfer learning. *arXiv preprint arXiv:2010.04296*, 2020.

[2] Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Daniel Ho, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Eric Jang, Rosario Jauregui Ruano, Kyle Jeffrey, Sally Jesmonth, Nikhil Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Kuang-Huei Lee, Sergey Levine, Yao Lu, Linda Luu, Carolina Parada, Peter Pastor, Jornell Quiambao, Kanishka Rao, Jarek Rettinghouse, Diego Reyes, Pierre Sermanet, Nicolas Sievers, Clayton Tan, Alexander Toshev, Vincent Vanhoucke, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Mengyuan Yan, and Andy Zeng. Do as i can and not as i say: Grounding language in robotic affordances. In *arXiv preprint arXiv:2204.01691*, 2022.

[3] Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katie Millican, Malcolm Reynolds, Roman Ring, Eliza Rutherford, Serkan Cabi, Tengda Han, Zhitao Gong, Sina Samangooei, Marianne Monteiro, Jacob Menick, Sebastian Borgeaud, Andy Brock, Aida Nematzadeh, Sahand Sharifzadeh, Mikolaj Binkowski, Ricardo Barreira, Oriol Vinyals, Andrew Zisserman, and Karen Simonyan. Flamingo: a visual language model for few-shot learning. *ArXiv*, abs/2204.14198, 2022.

[4] Anas Awadalla, Irena Gao, Josh Gardner, Jack Hessel, Yusuf Hanafy, Wanrong Zhu, Kalyani Marathe, Yonatan Bitton, Samir Gadre, Shiori Sagawa, Jenia Jitsev, Simon Kornblith, Pang Wei Koh, Gabriel Ilharco, Mitchell Wortsman, and Ludwig Schmidt. Openflamingo: An open-source framework for training large autoregressive vision-language models. *arXiv preprint arXiv:2308.01390*, 2023.

[5] Anas Awadalla, Irena Gao, Joshua Gardner, Jack Hessel, Yusuf Hanafy, Wanrong Zhu, Kalyani Marathe, Yonatan Bitton, Samir Gadre, Jenia Jitsev, Simon Kornblith, Pang Wei Koh, Gabriel Ilharco, Mitchell Wortsman, and Ludwig Schmidt. Openflamingo, March 2023. URL `https://doi.org/10.5281/zenodo.7733589`.

[6] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022.

[7] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023.

[8] Wenliang Dai, Junnan Li, Dongxu Li, Anthony Meng Huat Tiong, Junqi Zhao, Weisheng Wang, Boyang Li, Pascale Fung, and Steven Hoi. Instructblip: Towards general-purpose vision-language models with instruction tuning, 2023.

[9] Ishita Dasgupta, Christine Kaeser-Chen, Kenneth Marino, Arun Ahuja, Sheila Babayan, Felix Hill, and Rob Fergus. Collaborating with language models for embodied reasoning. In *Second Workshop on Language and Reinforcement Learning*, 2022. URL `https://openreview.net/forum?id=YoS-abmWjJc`.

[10] Danny Driess, Fei Xia, Mehdi S. M. Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, Wenlong Huang, Yevgen Chebotar, Pierre Sermanet, Daniel Duckworth, Sergey Levine, Vincent Vanhoucke, Karol Hausman, Marc Toussaint, Klaus Greff, Andy Zeng, Igor Mordatch, and Pete Florence. Palm-e: An embodied multimodal language model. In *arXiv preprint arXiv:2303.03378*, 2023.

[11] Yanjiang Guo, Yen-Jen Wang, Lihan Zha, Zheyuan Jiang, and Jianyu Chen. Doremi: Grounding language model by detecting and recovering from plan-execution misalignment. *arXiv preprint arXiv:2307.00329*, 2023.

[12] Lisa Anne Hendricks, Kaylee Burns, Kate Saenko, Trevor Darrell, and Anna Rohrbach. Women also snowboard: Overcoming bias in captioning models. In *Proceedings of the European conference on computer vision (ECCV)*, pages 771–787, 2018.

[13] Minho Heo, Youngwoon Lee, Doohyun Lee, and Joseph J. Lim. Furniturebench: Reproducible real-world benchmark for long-horizon complex manipulation. In *Robotics: Science and Systems*, 2023.

[14] Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, Pierre Sermanet, Tomas Jackson, Noah Brown, Linda Luu, Sergey Levine, Karol Hausman, and Brian Ichter. Inner monologue: Embodied reasoning through planning with language models. In Karen Liu, Dana Kulic, and Jeff Ichnowski, editors, *Proceedings of The 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pages 1769–1782. PMLR, 14–18 Dec 2023. URL `https://proceedings.mlr.press/v205/huang23c.html`.

[15] Stephen James, Zicong Ma, David Rovick Arrojo, and Andrew J Davison. Rlbench: The robot learning benchmark & learning environment. *IEEE Robotics and Automation Letters*, 5(2): 3019–3026, 2020.

[16] Yunfan Jiang, Agrim Gupta, Zichen Zhang, Guanzhi Wang, Yongqiang Dou, Yanjun Chen, Li Fei-Fei, Anima Anandkumar, Yuke Zhu, and Linxi Fan. Vima: General robot manipulation with multimodal prompts. In *Fortieth International Conference on Machine Learning*, 2023.

[17] KunChang Li, Yinan He, Yi Wang, Yizhuo Li, Wenhai Wang, Ping Luo, Yali Wang, Limin Wang, and Yu Qiao. Videochat: Chat-centric video understanding. *arXiv preprint arXiv:2305.06355*, 2023.

[18] Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. Code as policies: Language model programs for embodied control. In *CoRL 2022 Workshop LangRob, Workshop on Language and Robot Learning*, 2022. URL `https://openreview.net/pdf?id=fmtvpopfLC6`.

[19] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. In *arXiv:2304.08485*, 2023.

[20] Corey Lynch, Ayzaan Wahid, Jonathan Tompson, Tianli Ding, James Betker, Robert Baruch, Travis Armstrong, and Pete Florence. Interactive language: Talking to robots in real time. *arXiv preprint arXiv:2210.06407*, 2022.

[21] Oier Mees, Lukas Hermann, Erick Rosete-Beas, and Wolfram Burgard. Calvin: A benchmark for language-conditioned policy learning for long-horizon robot manipulation tasks. *IEEE Robotics and Automation Letters*, 2022.

[22] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.

[23] Mohit Shridhar, Lucas Manuelli, and Dieter Fox. Cliport: What and where pathways for robotic manipulation. In *Conference on Robot Learning*, pages 894–906. PMLR, 2022.

[24] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

[25] Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on robot learning*, pages 1094–1100. PMLR, 2020.

[26] Andy Zeng, Pete Florence, Jonathan Tompson, Stefan Welker, Jonathan Chien, Maria Attarian, Travis Armstrong, Ivan Krasin, Dan Duong, Vikas Sindhwani, et al. Transporter networks: Rearranging the visual world for robotic manipulation. In *Conference on Robot Learning*, pages 726–747. PMLR, 2021.

[27] Zhuosheng Zhang, Aston Zhang, Mu Li, Hai Zhao, George Karypis, and Alex Smola. Multimodal chain-of-thought reasoning in language models. *arXiv preprint arXiv:2302.00923*, 2023.

[28] Yuke Zhu, Josiah Wong, Ajay Mandlekar, Roberto Martín-Martín, Abhishek Joshi, Soroush Nasiriany, and Yifeng Zhu. robosuite: A modular simulation framework and benchmark for robot learning. In *arXiv preprint arXiv:2009.12293*, 2020.

A LoHoRavens environment details

A.1 Simulation environment

LoHoRavens is built based on the **Ravens** robot simulator by extending it to **Lo**ng-**Ho**rizon tasks. In the LoHoRavens simulation environment, there are a UR5e robot arm with a suction gripper and some objects on the table. Given a high-level language based instruction (e.g., "stack all the blocks of the same size"), the robot is supposed to rearrange these objects to a desired state. Based on Ravens, the observation space includes an RGB-D reconstruction from three camera views (front, left and right view). Besides, we also provide an RGB image rendered from the top-down view to the observation space. The action space of LoHoRavens consists of a language-conditioned pick-and-place motion primitive which is parameterized by object names.

Currently, LoHoRavens contains ten long-horizon tasks in total (see Table 1). To support more complex long-horizon reasoning, besides the vanilla pick-and-place primitive (e.g., "pick up the red block and place it on the yellow block"), we add two other pick-and-place primitives.[2] One is related to size reasoning (e.g., "pick up the smaller red block and place it on the bigger yellow block"), the other is related to spatial reasoning (e.g., "pick up the red block and place it in the top right area"). In addition to the pick-and-place primitive, we borrow two interesting tasks (tasks B and F) from Inner Monologue and CoP, and design another eight long-horizon tasks by ourselves.

Unlike Ravens and VIMA-Bench's complicated and various objects, LoHoRavens only contains three kinds of objects: block, bowl, and zone (see Fig. 1) because we do not want to test the robot's generalization capability to new or unseen objects in this work. Instead, we focus on the long-horizon reasoning capabilities which are related to the general attributes of objects like size, color and spatial position. Such reasoning capabilities can be generalized to other objects as well. In addition to these general object attributes, we are also interested in the reasoning capabilities related to attributes of multiple objects. So we include several tasks to test arithmetic and reference reasoning capabilities (e.g., tasks E and K).

To simulate the disturbance in the real world, we add noises and perturbations to the robot's environment at test time. Following Inner Monologue [14], we add Gaussian noise $\mathcal{N}(0, 3)$ for pixel observations and $\mathcal{N}(0, 2.5)$ for policy primitives. Moreover, we add a dropping probability p for the end-effector to drop the picked block every second following DoReMI [11].

A.2 Dataset

Like Ravens and VIMA-Bench, our simulator can also generate expert demonstrations automatically with the scripted oracle program. The oracle agent has access to the ground-truth pick and place poses and uses pre-specified heuristics to complete the tasks. All the tasks can be instantiated into thousands of task instances with different random seeds. To train the pick-and-place primitives, we generate 20,000 demonstrations for each primitive. To build the benchmark, we generate 1,000 demonstrations as the train set, 200 demonstrations as the validation set, and 200 demonstrations as the test set for each long-horizon task. Note that the colors of objects are chosen randomly, so they are generally different in training, validation and test sets. We split all the tasks into seen tasks and unseen tasks. The seen tasks are used for training and writing prompts. The unseen tasks are used for evaluating the model's generalization abilities to new tasks. Most of the task instances need five or more steps to complete. However, due to the attributes of some tasks, it is difficult to design a high-level goal that needs many steps. Taking stacking blocks as an example, it is difficult to stack more than five blocks in the same position because the blocks will easily fall if they are stacked too high.

A.3 Evaluation

Depending on the task, there are two different match methods for evaluating whether the states of the objects are correct compared to the ground-truth states. One is based on "pose match", which means an object's position and rotation should be the same as the ground-truth one. Another one is based on "zone match", which means the overlap area of two objects should be larger than a threshold. Following Ravens and CLIPort, LoHoRavens adopts a score from 0 (fail) to 100 (success) to evaluate the final state for each task instance. The score assigns the partial rewards according to the total

[2]We use the expression "pick-and-place primitives" to refer to all three primitives in the table.

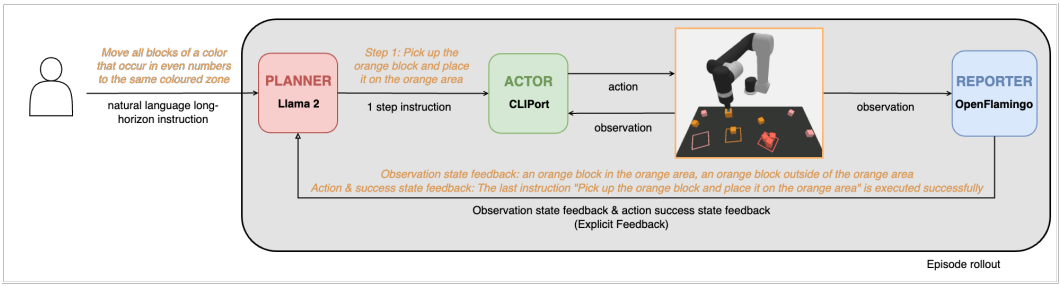

Figure 2: **Explicit feedback: Caption generation.** This baseline takes the human input ("Move all blocks of a color that occur in even numbers to the same coloured zone") and asks an LLM to create the next step that needs to be done in order to achieve the task. The LLM acts as a planner (red box) that provides a single step instruction to the actor (green box). In both baselines, the planner and actor are the same, namely Llama 2 and CLIPort respectively. The actor provides action policies, i.e., the actions of the robot. The results of those actions are observed by both actor and reporter. In this baseline, the reporter (blue box) is the vision-language model OpenFlamingo. The reporter provides **captions** that report on the observation state ("an orange block in the orange area, an orange block outside of the orange area") and an action & success state ("The last instruction "Pick up the orange block and place it on the orange area" is executed successfully"), which are both sent back to the planner as explicit language-based feedback to produce the next step.

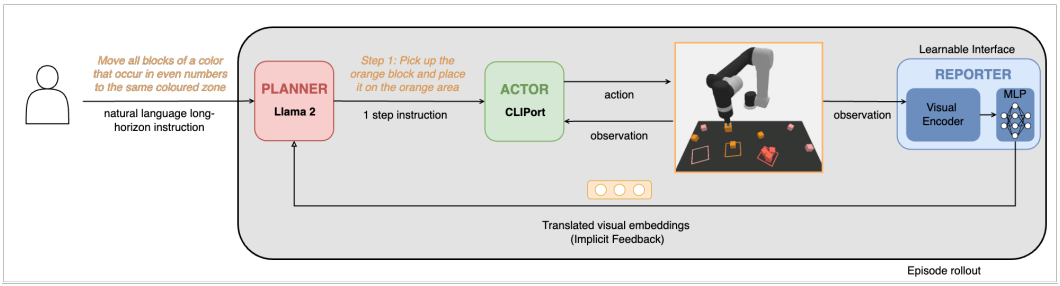

Figure 3: **Implicit feedback: Learnable interface.** This baseline has the same planner (red box) and actor (green box) architecture as the caption-based baseline in Fig. 2. The difference is that in this baseline the reporter (blue box) is a **learnable interface** (as described in Sec. 3). It provides the translated visual embedding as implicit feedback to the LLM to produce the next step.

number of pick-and-place steps for each task instance. For example, if a task needs ten pick-and-place steps to complete, and the test model finishes eight of them, the score for this instance would be $8/10 = 80\%$.

B Baselines details

The figures for the two baselines can be seen in Fig. 2 and Fig. 3.

C Experiment details

C.1 Experimental settings

There are two baselines in our experiments: explicit caption based model and implicit learnable interface based model. For the caption based model, we can further compare the effects of each module of Planner, Actor, and Reporter. Except for the `CLIPort (oracle)` model, all the other models use the same pick-and-place primitive Actor trained on three sets (one for each of the three primitives) of 20,000 demonstrations by multi-task learning.

`CLIPort (oracle)` refers to using CLIPort as the actor model (without using a planner or a reporter). It is a multi-task policy trained on all the training data of the seen tasks. Because the vanilla CLIPort

does not know when to stop execution, following Inner Monologue and CaP, we use an oracle termination variant that uses the oracle information from the simulator to detect the success state and stop the execution process. `CLIPort + Llama 2` [3] is the model combining Actor and Planner. `CLIPort + Llama 2 + OpenFlamingo` [4] is the model combining Actor, Planner, and Reporter. Both Llama 2 and OpenFlamingo use 10-shot prompts for inference. For the learnable interface model, `LLaVA` [5] serves as the Reporter and Planner modules. As mentioned before (end of Sec. 3), we fine-tune it on our generated training data consisting of pairs of simulator rendered images and corresponding next-step language instruction.

D Task details

D.1 A. Pick-and-Place primitives

This task asks the robot to pick up a block of a specific color or size and place it at a relative or absolute position.

Instruction Pick up the `[SIZE?/COLOR]` block and place it on the `[SIZE?/COLOR] [OBJECT]`; Pick up the `[SIZE?/COLOR]` block and place it on the `[POSITION]` of the table.

Variables `SIZE` $\in \{\text{smaller}, \text{bigger}\}$,

`COLOR` $\in \{\text{blue}, \text{yellow}, \text{red}, \text{green}, \text{pink}, \text{grey}, \text{white}, \text{brown}, \text{cyan}, \text{purple}\}$,

`OBJECT` $\in \{\text{bowl}, \text{block}, \text{zone}\}$,

`POSITION` $\in \{\text{top left area}, \text{top right area}, \text{bottom left area}, \text{bottom right area}\}$.

Description The task requires spatial reasoning and the ability to discern colors and shapes. The task is a simple pick up block and place block, but the color of the source and target blocks can be varied as well as their sizes. Target position can either be a object (e.g., bowl, block, zone), or an absolute position of the table (e.g. bottom right area of the table).

Capabilities Reasoning about colors, shapes, sizes and spatial relations.

Success criteria All specified blocks are placed in the specified areas.

D.2 B. Put all the blocks in the bowls with matching colors

This task requires the robot to correctly match and place blocks of specific colors into corresponding bowls.

Instruction Place all blocks into bowls of the respective color, e.g. blue block in blue bowl and so on.

Variables -

Description In this task, there are several colored blocks and matching colored bowls. The robot needs to correctly identify the color of each block and place it into the bowl with the corresponding color. For example, the blue block should go into the blue bowl, the yellow block into the yellow bowl, and so on.

Capabilities Color recognition.

Success criteria All blocks are placed in the bowls with matching colors.

[3] We use the Llama 2 13B version.

[4] We use the OpenFlamingo-9B-vitl-mpt7b version.

[5] We fine-tune the LLaVA 13B version.

D.3 C. Stack smaller blocks over bigger blocks of the same color

This task requires the robot to stack blocks of the same color with smaller blocks on top of bigger blocks.

Instruction Stack smaller blocks over bigger blocks of the same color.

Variables -

Description In this task, there are blocks of different colors and sizes available, and the robot must identify blocks of the same color and stack them with smaller blocks on top of bigger blocks. For example, if there are two blue blocks (one smaller and one bigger), the robot should stack the smaller blue block on top of the bigger blue block. This applies to blocks of all colors.

Capabilities Color recognition, size recognition.

Success criteria All blocks of the same color are stacked correctly, with smaller blocks on top of bigger blocks.

D.4 D. Stack all the blocks in the [ABS_POS] area

Instruction Stack all the blocks in the `[POSITION]` area.

Variables `POSITION` $\in \{\text{top left area}, \text{top right area}, \text{bottom left area}, \text{bottom right area}\}$.

Description In this task, the robot is asked to stack all the blocks in an absolute area of the table.

Capabilities Spatial reasoning.

Success criteria All blocks in the specific area are stacked correctly.

D.5 E. Move all blocks of a color that occur in even numbers to the same colored zone

This task requires the robot to stack blocks with even-numbered occurrence.

Instruction Move all blocks of a color that occur in even numbers to the same colored zone.

Variables -

Description In this task, the robot must stack blocks that occur in even numbers.

Capabilities Counting, arithmetic reasoning, color recognition.

Success criteria All blocks that occur in even numbers are successfully stacked.

D.6 F. Put all the blocks in the bowls with mismatching colors

This task requires the robot to intentionally mismatch blocks with bowls of different colors.

Instruction Place all blocks into the bowls of different color.

Variables -

Description In this task, the robot must place blocks into bowls with colors that do not match. Each block has a specific color, and there are bowls with mismatched colors. For example, the robot should place a blue block into a yellow bowl, a red block into a green bowl, and so on.

Capabilities Color recognition.

Success criteria All blocks are placed in bowls with colors that do not match the blocks' colors.

D.7 G. Stack blocks of the same size

This task requires the robot to stack blocks of identical sizes on top of each other.

Instruction Stack blocks of the same size.

Variables -

Description In this task, there are blocks of different sizes available, and the robot must identify blocks of the same size and stack them on top of each other. For example, if there are two smaller blocks and three bigger blocks, the robot should create two stacks, one with the smaller blocks and one with the bigger blocks.

Capabilities Size recognition.

Success criteria All blocks are stacked according to their size, with blocks of the same size stacked together.

D.8 H. Stack blocks in alternate colors

This task requires the robot to stack blocks in an alternating color pattern.

Instruction Stack the blocks in alternate colors.

Variables -

Description In this task, there are blocks of two colors available, and the robot must stack them in a pattern where colors alternate. For example, if there are blue, yellow blocks, the robot should stack them as: blue, yellow, blue, yellow, and so on.

Capabilities Color recognition, logic planning.

Success criteria The blocks are stacked in an alternating color pattern.

D.9 I. Stack blocks of the same color in the zone with the same color, with the bigger blocks underneath

This task requires the robot to stack blocks of the same color in designated zones, with bigger blocks at the bottom.

Instruction Stack blocks of the same color in the zone with the same color, with the bigger blocks underneath

Variables -

Description In this task, there are blocks of different colors and sizes available. The robot must identify blocks of the same color and stack them in a specified zone with the bigger blocks placed at the bottom. For example, if there are two red blocks (one bigger and one smaller), the robot should stack the smaller red block on top of the bigger red block in the designated red zone.

Capabilities Color recognition, size recognition.

Success criteria Blocks of the same color are stacked correctly in their designated zones, with bigger blocks at the bottom.

D.10 J. Move all the blocks in the [ABS_POS] area to the [ABS_POS] area

This task requires the robot to relocate blocks from an area to another area on the table.

Instruction Move all the blocks in the `[POSITION]` to the `[POSITION]`.

Variables `POSITION` $\in \{\text{top left area}, \text{top right area}, \text{bottom left area}, \text{bottom right area}\}$.

Description In this task, the robot's objective is to move all the blocks from a specific area to another area on the table.

Capabilities Spatial reasoning.

Success criteria All blocks from the source area are successfully moved to the target area on the table.

D.11 K. Stack blocks of the same color, given there are multiple blocks with the same color

This task requires the robot to stack blocks of the same color when there are multiple blocks with the same color available.

Instruction Stack the blocks of the same color, given there are multiple blocks with the same color.

Variables -

Description In this task, there are blocks of different colors, and some colors have multiple blocks available. The robot must identify blocks of the same color and stack them on top of each other. For example, if there are three red blocks, the robot should stack all the red blocks on top of each other.

Capabilities Color recognition, reference reasoning.

Success criteria Blocks of the same color are stacked correctly on top of each other.