Scaling Robot Policy Learning via Zero-Shot Labeling with Foundation Models

Nils Blank[1], Moritz Reuss[1], Fabian Wenzel[1], Oier Mees[2], Rudolf Lioutikov[1]

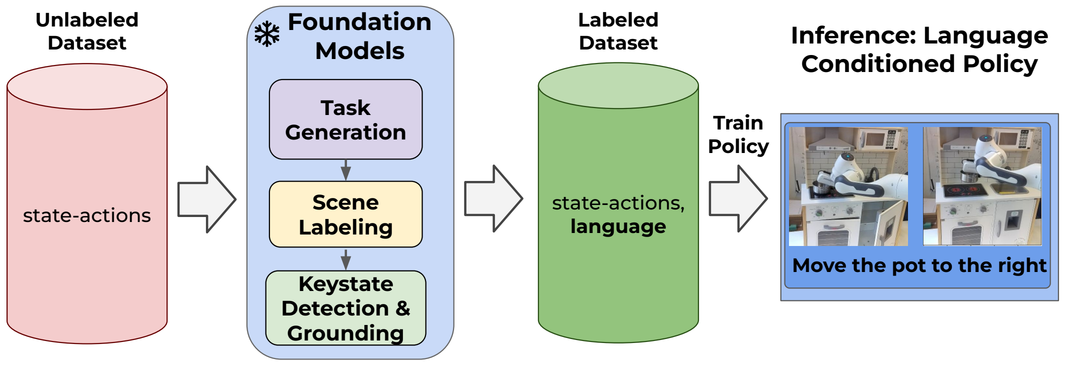

Fig. 1: LUPUS leverages an ensemble of frozen pre-trained models to segment and annotate uncurated, long-horizon robot demonstrations. The resulting fully-labeled and segmented dataset can be used to train language-conditioned policies zero-shot without any human annotation.

***Abstract*— A central challenge towards developing robots that can relate human language to their perception and actions is the scarcity of natural language annotations in diverse robot datasets. Moreover, robot policies that follow natural language instructions are typically trained on either templated language or expensive human-labeled instructions, hindering their scalability. To this end, we introduce a novel approach to automatically label uncurated, long-horizon robot teleoperation data at scale in a zero-shot manner without any human intervention. We utilize a combination of pre-trained vision-language foundation models to detect objects in a scene, propose possible tasks, segment tasks from large datasets of unlabelled interaction data and then train language-conditioned policies on the relabeled datasets. Our initial experiments show that our method enables training language-conditioned policies on unlabeled and unstructured datasets that match ones trained with oracle human annotations.**

I. Introduction

Language-conditioned policies offer an intuitive and user-friendly method to instruct robots [1], [2]. Training such policies to perform diverse skills requires large amounts of text-labeled robot trajectories for the policy to ground instructions to behavior.

In response to these obstacles, Learning from Play (LfP) has been introduced as a method of using play-like data for fast and diverse data collection. This approach acquires extended, varied demonstrations and makes the data collection process more efficient and cost-effective. However, generating language labels for play demonstrations usually includes human labeling, which is costly and inefficient.

To address this, we introduce Labeling Unstructed Play Data utilizing Specialist Language Models (LUPUS), a novel method for zero-shot labeling of long-horizon videos capturing robot play without necessitating human intervention or additional model training. LUPUS employs an ensemble of pre-trained expert models to identify relevant objects within the environment. Our method introduces an expert ensemble method to identify object-centric keystates to segment long videos consisting of multiple tasks into smaller windows with single actions. Subsequently, a Large Language Model (LLM) generates language instructions matching the scene changes tracked by various pre-trained models. As a result, LUPUS converts long-horizon play data into segmented and annotated datasets for training language-conditioned policies without any manual labeling.

We evaluate LUPUS on a challenging self-collected play dataset in a toy kitchen. Our findings indicate that LUPUS not only efficiently annotates play data with appropriate task descriptions but also surpasses state-of-the-art closed-source Vision-Language Models (VLMs), such as Gemini-Pro. Furthermore, regardless of grounding, LUPUS reliably finds important keystates in long-horizon demonstrations better than prior zero-shot methods such as UVD [3]. Finally, we demonstrate the efficacy of LUPUS in zero-shot policy learning by training a language-guided diffusion policy using our synthetically labeled data.

[1]Intuitive Robots Lab, Karlsruhe Institute of Technology, Germany
[2] Berkeley Artificial Intelligence Research Lab, University of California, Berkeley, USA

II. Method

In this section, LUPUS is introduced. We start by giving a high-level overview of the method, followed by detailed explanations of its three sub-methods for zero-shot play labeling and key-state detection.

A. *LUPUS Method Overview*

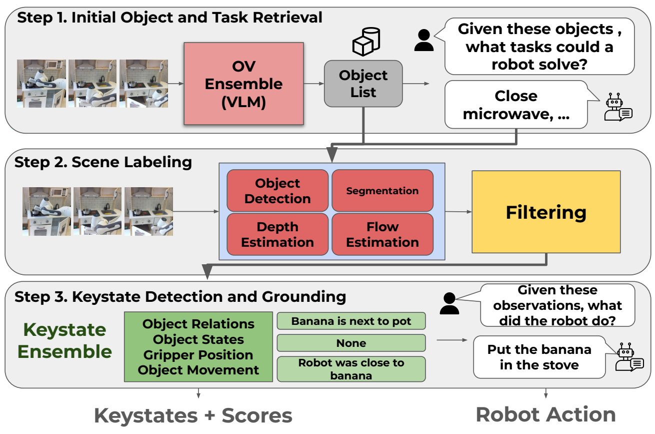

Fig. 2: Overview of the proposed LUPUS framework for labeling long-horizon robot play sequences in a zero-shot manner using an ensemble of experts.

LUPUS is divided into three primary steps: (I) Potential Task Identification and Initial Object Retrieval, (II) Scene Labeling, and (III) Keystate Detection with Grounding. Step (I) focuses on identifying all objects within the scene and querying a Large Language Model (LLM) to generate a list of all potential tasks involving these objects. Step (II) involves labeling all scene objects and monitoring their changes throughout a frame sequence. Step (III) is dedicated to identifying object-centric key states and prompting an LLM to annotate the segmented interactions with appropriate natural language descriptions. Figure 2 depicts a comprehensive overview of these steps. The subsequent sections elaborate on each component of LUPUS.

B. *Step 1: Identifying Potential Tasks from RGB Images*

To effectively annotate play data with LUPUS, LUPUS starts by determining all objects in the scene and subsequently compiling a set of potential tasks executable within the given environment.

LUPUS first generates n class-agnostic bounding boxes for 16 uniformly sampled frames with OWLv2 [4]. These boxes are then aligned with a predefined list of objects commonly appearing in robotic environments. LUPUS further calculates similarity scores with SigLIP [5] for each box proposal against all object text representations for more robust grounding. Scores from both grounding predictions are then combined and filtered based on objectness scores, grounding accuracy, and object temporal presence to finalize the scene's object set. With a finalized object list, an LLM is prompted to generate a potential task list by considering the objects' interactions. Figure 13 of the Appendix gives an example prompt to generate this task list.

C. *Step 2: Scene Labeling*

The perception module annotates all observable object changes throughout the video, encompassing bounding boxes, segmentation masks, object positions, frame-to-frame displacements, and object relationships. LUPUS integrates several state-of-the-art models for scene labeling.
For object detection, LUPUS uses an ensemble of open-vocabulary detectors, namely OWLv2 [4] and CLIPSeg [6]. For segmentation, LUPUS employs Efficient-SAM [7]. To ensure robust scene representations, LUPUS performs several filtering steps, including temporal aggregation with DEVA [8]. Details of our perception pipeline are presented in section D.1 of the Appendix.

D. *Step 3: Keystate Detection by Heuristic Consensus*

Given the scene annotations from Step 2, LUPUS utilizes these representations to detect keystates and perform grounding. LUPUS uses multiple heuristics to detect keystates in the long-horizon trajectory. By combining multiple heuristics, we can filter out keystates induced by noise in the observations and control the quality of the keystates with the resulting score. Each heuristic monitors keystate changes for individual objects. This approach minimizes overlapping agreements in different parts of the scene caused by noise.

An object-centric keystate o_i is valid if its score exceeds a user-specified threshold. LUPUS considers keystates of different heuristics within a certain range to be referring to the same keystate. The keystates are averaged across the heuristics for a final keystate.

LUPUS uses gripper position, object states, object relations, object movement, and gripper close signals as heuristics. The heuristics are detailed in Appendix D.2.

E. *Action Retrieval and Grounding*

Each keystate heuristic outputs a natural description of why a keystate was detected. LUPUS uses this information to construct a prompt to query a large language model. The LLM is tasked to reason about detected object movement and relation changes to determine the possible actions performed by the robot. Sometimes, the observations are insufficient to reason about the performed task. In such cases, we instruct the LLM to output all possible tasks that could result in the observations. Despite the instruction being ambiguous, it could still be useful for downstream policy learning. We evaluate this hypothesis in our experiments. We provide a list of prompts in Section I of the Appendix.

III. Evaluation

In this section, we study LUPUS as an effective tool for labeling uncurated play datasets. We want to answer the following key questions: (I) Is LUPUS able to label uncurated, long-horizon robot data in different environments with a high accuracy? (II) How good are the generated keystates? (III) How does LUPUS perform in grounding against recent state-of-the-art vision-language foundation models? (IV) How well does a policy trained on automatic annotations understand language instructions compared to a policy trained with human annotations?

A. *Experiment Description*

To test the capabilities of LUPUS, we collect a long-horizon play dataset in our own play kitchen using teleoperation. The dataset and evaluation setup details can be found in Sec. B.

B. *Keystate Evaluation*

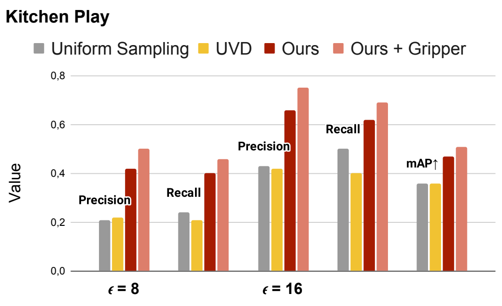

Fig. 3: Keystate accuracy for different frame distance tolerances. When additionally incorporating gripper-close signals, the keystate quality further increases.

We perform a quantitative evaluation of the keystates produced by our method on the play dataset recorded in a toy kitchen and BridgeV2 [9] for different tolerance thresholds. If a predicted keystates distance to an actual keystate is smaller than the threshold, it is labeled as correct. Fig. 3 shows the result of our method on our Kitchen Play dataset, compared against UVD [3] and uniform sampling with interval 64.

Notably, LUPUS outperforms UVD by a large margin in precision and recall for both tolerances and mAP. Additionally, incorporating gripper-close signals further improves the performance. We provide ablations of the performance of our method with different heuristics in Table IV of the Appendix. Results on CALVIN and BridgeV2 can be seen in Sec. F of the Appendix.

C. *Grounding Evaluation*

Method	LLM	Accuracy ($\epsilon = 8$) Amb.	Accuracy ($\epsilon = 8$) Single	Accuracy ($\epsilon = 16$) Amb.	Accuracy ($\epsilon = 16$) Single
S3D	-	0.04		0.03	
XCLIP	-	0.07		0.09	
Gemini		0.13		0.13	
LUPUS	GPT-3.5	0.70	0.55	0.67	0.53
	GPT-4	**0.84**	**0.77**	**0.79**	**0.71**
	Gemini (Lang)	0.80	0.61	0.75	0.57
	Mixtral8x7b	0.66	0.52	0.62	0.49

TABLE I: Grounding accuracy of our framework. For Amb., the prediction is labeled correct if the list of answers contains the ground truth. In Single, ambiguous predictions are considered wrong.

To address Question (III), we compare the annotations produced by our framework with ground truth language annotations obtained through human labeling on the Kitchen Play environment. Given the natural language observations made by our heuristics, we prompt four different LLMs to select up to two tasks from a task list. We compare against Gemini Pro Vision [10], which we prompt with eight evenly spaced frames, S3D [11], [12] and XCLIP [13].

LUPUS outperforms all baselines by a large margin, as seen in Table I.

D. *Language-conditioned Policy Training*

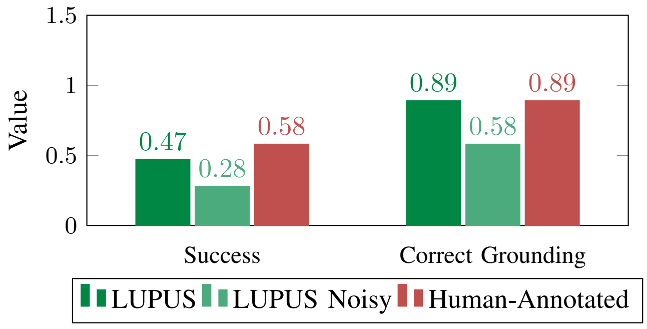

Fig. 4: Comparison of policies trained with different data labels in our real world setting.

Next, we train a language-conditioned policy [14] on our automatically labeled dataset and compare it against a policy trained with a human-annotated dataset. For evaluation, we task the policies to solve tasks specified in natural language. We evaluate two quantities: instruction understanding and success rate. Detailed descriptions of our policy, environment setup, and evaluation are summarized in B of the Appendix. Fig. 4 shows the grounding and success ratio across 12 tasks. The policy trained with automatically labeled data performs on par with the human-annotated dataset regarding grounding, while the success rate is slightly lower. This can be attributed to the overall lower number of training samples produced by our method. Including noisy samples seems to hurt the performance significantly.

IV. Conclusion

In this work, we introduced LUPUS, the first method, which is able to fully label long-horizon play datasets without needing any human interventions or model training. The framework leverages a set of vision-language foundation models and an LLM to detect key-frames and ground actions in the demonstrations. We show that LUPUS' keystate detection heuristics can be used to extract informative keyframes from long horizon data with RGB images only. Furthermore, LUPUS is able to generate natural language labels for long-horizon videos. Our experiments demonstrate that language-guided policies trained on the artificially labeled dataset perform competitively with those trained on fully human-labeled data.

V. Acknowledgements

The work presented here was funded by the German Research Foundation (DFG) – 448648559. The authors acknowledge support by the state of Baden-Württemberg through HoreKa supercomputer funded by the Ministry of Science, Research and the Arts Baden-Württemberg and by the German Federal Ministry of Education and Research.

References

[1] O. Mees, L. Hermann, and W. Burgard, "What Matters in Language Conditioned Robotic Imitation Learning over Unstructured Data," Aug. 2022, arXiv:2204.06252 [cs]. [Online]. Available: http://arxiv.org/abs/2204.06252

[2] C. Lynch and P. Sermanet, "Language Conditioned Imitation Learning over Unstructured Data," Jul. 2021, arXiv:2005.07648 [cs]. [Online]. Available: http://arxiv.org/abs/2005.07648

[3] Z. Zhang, Y. Li, O. Bastani, A. Gupta, D. Jayaraman, Y. J. Ma, and L. Weihs, "Universal visual decomposer: Long-horizon manipulation made easy," 2023.

[4] M. Minderer, A. Gritsenko, and N. Houlsby, "Scaling Open-Vocabulary Object Detection," Jul. 2023, arXiv:2306.09683 [cs]. [Online]. Available: http://arxiv.org/abs/2306.09683

[5] X. Zhai, B. Mustafa, A. Kolesnikov, and L. Beyer, "Sigmoid loss for language image pre-training," 2023.

[6] T. Lüddecke and A. S. Ecker, "Image Segmentation Using Text and Image Prompts," Mar. 2022, arXiv:2112.10003 [cs]. [Online]. Available: http://arxiv.org/abs/2112.10003

[7] Y. Xiong, B. Varadarajan, L. Wu, X. Xiang, F. Xiao, C. Zhu, X. Dai, D. Wang, F. Sun, F. Iandola, R. Krishnamoorthi, and V. Chandra, "Efficientsam: Leveraged masked image pretraining for efficient segment anything," 2023.

[8] H. K. Cheng, S. W. Oh, B. Price, A. Schwing, and J.-Y. Lee, "Tracking anything with decoupled video segmentation," 2023.

[9] H. Walke, K. Black, A. Lee, M. J. Kim, M. Du, C. Zheng, T. Zhao, P. Hansen-Estruch, Q. Vuong, A. He, V. Myers, K. Fang, C. Finn, and S. Levine, "Bridgedata v2: A dataset for robot learning at scale," 2024.

[10] G. Team, "Gemini: A family of highly capable multimodal models," 2023.

[11] S. A. Sontakke, J. Zhang, S. Arnold, K. Pertsch, E. Biyik, D. Sadigh, C. Finn, and L. Itti, "RoboCLIP: One Demonstration is Enough to Learn Robot Policies," Nov. 2023. [Online]. Available: https://openreview.net/forum?id=DVlawv2rSI

[12] A. Miech, D. Zhukov, J.-B. Alayrac, M. Tapaswi, I. Laptev, and J. Sivic, "Howto100m: Learning a text-video embedding by watching hundred million narrated video clips," 2019.

[13] Y. Ma, G. Xu, X. Sun, M. Yan, J. Zhang, and R. Ji, "X-clip: End-to-end multi-grained contrastive learning for video-text retrieval," 2022.

[14] M. Reuss and R. Lioutikov, "Multimodal diffusion transformer for learning from play," in *2nd Workshop on Language and Robot Learning: Language as Grounding*, 2023. [Online]. Available: https://openreview.net/forum?id=nvtxqMGpn1

[15] C. Lynch, M. Khansari, T. Xiao, V. Kumar, J. Tompson, S. Levine, and P. Sermanet, "Learning Latent Plans from Play," Dec. 2019, arXiv:1903.01973 [cs]. [Online]. Available: http://arxiv.org/abs/1903.01973

[16] E. Rosete-Beas, O. Mees, G. Kalweit, J. Boedecker, and W. Burgard, "Latent Plans for Task-Agnostic Offline Reinforcement Learning," Sep. 2022, arXiv:2209.08959 [cs]. [Online]. Available: http://arxiv.org/abs/2209.08959

[17] A. Gupta, V. Kumar, C. Lynch, S. Levine, and K. Hausman, "Relay policy learning: Solving long-horizon tasks via imitation and reinforcement learning," 2019.

[18] S. Nair and C. Finn, "Hierarchical foresight: Self-supervised learning of long-horizon tasks via visual subgoal generation," 2019.

[19] K. Shiarlis, M. Wulfmeier, S. Salter, S. Whiteson, and I. Posner, "Taco: Learning task decomposition via temporal alignment for control," 2018.

[20] O. Mees, L. Hermann, and W. Burgard, "What Matters in Language Conditioned Robotic Imitation Learning over Unstructured Data," Aug. 2022, arXiv:2204.06252 [cs]. [Online]. Available: http://arxiv.org/abs/2204.06252

[21] O. Mees, J. Borja-Diaz, and W. Burgard, "Grounding Language with Visual Affordances over Unstructured Data," Mar. 2023, arXiv:2210.01911 [cs]. [Online]. Available: http://arxiv.org/abs/2210.01911

[22] O. Mees, L. Hermann, E. Rosete-Beas, and W. Burgard, "Calvin: A benchmark for language-conditioned policy learning for long-horizon robot manipulation tasks," 2022.

[23] V. Myers, A. He, K. Fang, H. Walke, P. Hansen-Estruch, C.-A. Cheng, M. Jalobeanu, A. Kolobov, A. Dragan, and S. Levine, "Goal Representations for Instruction Following: A Semi-Supervised Language Interface to Control," Aug. 2023, arXiv:2307.00117 [cs]. [Online]. Available: http://arxiv.org/abs/2307.00117

[24] O. Mees, L. Hermann, and W. Burgard, "What Matters in Language Conditioned Robotic Imitation Learning over Unstructured Data," Aug. 2022, arXiv:2204.06252 [cs]. [Online]. Available: http://arxiv.org/abs/2204.06252

[25] M. Reuss, M. Li, X. Jia, and R. Lioutikov, "Goal-Conditioned Imitation Learning using Score-based Diffusion Policies," Jun. 2023, arXiv:2304.02532 [cs]. [Online]. Available: http://arxiv.org/abs/2304.02532

[26] J. Borja-Diaz, O. Mees, G. Kalweit, L. Hermann, J. Boedecker, and W. Burgard, "Affordance learning from play for sample-efficient policy learning," 2022.

[27] S. James and A. J. Davison, "Q-attention: Enabling efficient learning for vision-based robotic manipulation," 2022.

[28] M. Shridhar, L. Manuelli, and D. Fox, "Perceiver-actor: A multi-task transformer for robotic manipulation," 2022.

[29] Z. Liu, A. Bahety, and S. Song, "REFLECT: Summarizing Robot Experiences for Failure Explanation and Correction," Oct. 2023, arXiv:2306.15724 [cs]. [Online]. Available: http://arxiv.org/abs/2306.15724

[30] A. Yang, A. Nagrani, P. H. Seo, A. Miech, J. Pont-Tuset, I. Laptev, J. Sivic, and C. Schmid, "Vid2seq: Large-scale pretraining of a visual language model for dense video captioning," 2023.

[31] S. Yan, T. Zhu, Z. Wang, Y. Cao, M. Zhang, S. Ghosh, Y. Wu, and J. Yu, "VideoCoCa: Video-Text Modeling with Zero-Shot Transfer from Contrastive Captioners," Mar. 2023, arXiv:2212.04979 [cs]. [Online]. Available: http://arxiv.org/abs/2212.04979

[32] Z. Wang, M. Li, R. Xu, L. Zhou, J. Lei, X. Lin, S. Wang, Z. Yang, C. Zhu, D. Hoiem, S.-F. Chang, M. Bansal, and H. Ji, "Language models with image descriptors are strong few-shot video-language learners," 2022.

[33] H. Fang, P. Xiong, L. Xu, and Y. Chen, "CLIP2Video: Mastering Video-Text Retrieval via Image CLIP," Jun. 2021, arXiv:2106.11097 [cs]. [Online]. Available: http://arxiv.org/abs/2106.11097

[34] D. Ko, J. S. Lee, W. Kang, B. Roh, and H. J. Kim, "Large Language Models are Temporal and Causal Reasoners for Video Question Answering," Nov. 2023, arXiv:2310.15747 [cs]. [Online]. Available: http://arxiv.org/abs/2310.15747

[35] P. Sermanet, T. Ding, J. Zhao, F. Xia, D. Dwibedi, K. Gopalakrishnan, C. Chan, G. Dulac-Arnold, S. Maddineni, N. J. Joshi, P. Florence, W. Han, R. Baruch, Y. Lu, S. Mirchandani, P. Xu, P. Sanketi, K. Hausman, I. Shafran, B. Ichter, and Y. Cao, "RoboVQA: Multimodal Long-Horizon Reasoning for Robotics," Nov. 2023, arXiv:2311.00899 [cs]. [Online]. Available: http://arxiv.org/abs/2311.00899

[36] J. Pan, Z. Lin, X. Zhu, J. Shao, and H. Li, "St-adapter: Parameter-efficient image-to-video transfer learning," 2022.

[37] T. Xiao, H. Chan, P. Sermanet, A. Wahid, A. Brohan, K. Hausman, S. Levine, and J. Tompson, "Robotic Skill Acquisition via Instruction Augmentation with Vision-Language Models," Jul. 2023, arXiv:2211.11736 [cs]. [Online]. Available: http://arxiv.org/abs/2211.11736

[38] S. Karamcheti, S. Nair, A. S. Chen, T. Kollar, C. Finn, D. Sadigh, and P. Liang, "Language-Driven Representation Learning for Robotics," Feb. 2023, arXiv:2302.12766 [cs]. [Online]. Available: http://arxiv.org/abs/2302.12766

[39] Y. J. Ma, W. Liang, V. Som, V. Kumar, A. Zhang, O. Bastani, and D. Jayaraman, "LIV: Language-Image Representations and Rewards for Robotic Control," Jun. 2023, arXiv:2306.00958 [cs]. [Online]. Available: http://arxiv.org/abs/2306.00958

[40] Y. J. Ma, S. Sodhani, D. Jayaraman, O. Bastani, V. Kumar, and A. Zhang, "Vip: Towards universal visual reward and representation via value-implicit pre-training," 2023.

[41] A. Adeniji, A. Xie, C. Sferrazza, Y. Seo, S. James, and P. Abbeel, “Language Reward Modulation for Pretraining Reinforcement Learning,” Aug. 2023, arXiv:2308.12270 [cs]. [Online]. Available: http://arxiv.org/abs/2308.12270

[42] R. Goyal, S. E. Kahou, V. Michalski, J. Materzyńska, S. Westphal, H. Kim, V. Haenel, I. Fruend, P. Yianilos, M. Mueller-Freitag, F. Hoppe, C. Thurau, I. Bax, and R. Memisevic, “The ”something something” video database for learning and evaluating visual common sense,” 2017.

[43] D. Damen, H. Doughty, G. M. Farinella, S. Fidler, A. Furnari, E. Kazakos, D. Moltisanti, J. Munro, T. Perrett, W. Price, and M. Wray, “Scaling egocentric vision: The epic-kitchens dataset,” 2018.

[44] T. Yu, T. Xiao, A. Stone, J. Tompson, A. Brohan, S. Wang, J. Singh, C. Tan, D. M, J. Peralta, B. Ichter, K. Hausman, and F. Xia, “Scaling Robot Learning with Semantically Imagined Experience,” Feb. 2023, arXiv:2302.11550 [cs]. [Online]. Available: http://arxiv.org/abs/2302.11550

[45] H. Xu, J. Zhang, J. Cai, H. Rezatofighi, and D. Tao, “Gmflow: Learning optical flow via global matching,” 2022.

[46] L. Yang, B. Kang, Z. Huang, X. Xu, J. Feng, and H. Zhao, “Depth anything: Unleashing the power of large-scale unlabeled data,” 2024.

[47] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, “A density-based algorithm for discovering clusters in large spatial databases with noise,” in *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, ser. KDD’96. AAAI Press, 1996, p. 226–231.

[48] B. Chen, Z. Xu, S. Kirmani, B. Ichter, D. Driess, P. Florence, D. Sadigh, L. Guibas, and F. Xia, “SpatialVLM: Endowing Vision-Language Models with Spatial Reasoning Capabilities,” Jan. 2024, arXiv:2401.12168 [cs]. [Online]. Available: http://arxiv.org/abs/2401.12168

[49] H. Xu, J. Zhang, J. Cai, H. Rezatofighi, F. Yu, D. Tao, and A. Geiger, “Unifying flow, stereo and depth estimation,” 2023.

[50] X. Wang, Y. Zhou, X. Liu, H. Lu, Y. Xu, F. He, J. Yoon, T. Lu, G. Bertasius, M. Bansal, H. Yao, and F. Huang, “Mementos: A comprehensive benchmark for multimodal large language model reasoning over image sequences,” 2024.

[51] J. Gao, B. Sarkar, F. Xia, T. Xiao, J. Wu, B. Ichter, A. Majumdar, and D. Sadigh, “Physically Grounded Vision-Language Models for Robotic Manipulation,” Sep. 2023, arXiv:2309.02561 [cs]. [Online]. Available: http://arxiv.org/abs/2309.02561

APPENDIX

A. Related Work

Learning From Play. LfP is a training data paradigm for imitation learning, that leverages play-like data for imitation learning. Instead of collecting a fixed set of expert demonstrations, an operator interacts in the given environment without constraints. Collecting demonstrations this way results in more diverse and wide state space and easier data collection. Consequently, a policy trained from such play data usually shows a better generalization and performance in unseen state goal spaces. To apply this paradigm to goal-conditioned imitation learning, goal states in these long demonstrations must be available. When working with image goals, goal states can be extracted by sampling random windows from the long-horizon play sequence or with robot proprietary information [15], [16], [14]. Hierarchical methods use multiple policies to extract and learn subskills from play data [17], [18], [19], [20], [21].

Several recent work try to learn a multimodal goal space to embed goal states of text instructions and goal images [22], [2], [23], [24], [25], [11]. Some methods require as little as 1% language annotated data combined with efficiently collected image goals to train policies [2], [22], [14]. Despite these efforts, learning language-conditioned policies from play still requires some language annotations.

Key State Identification. Long-horizon tasks often consist of multiple subtasks. To efficiently learn goal-conditioned policies from long-horizon tasks, one has to identify important keystates from long-horizon tasks. From these keystates, various goal modalities, such as image or language goals, can be derived. Several methods can be used to extract keystates from long-horizon demonstrations. These methods can range from very simple to complex, depending on the complexity of the task at hand. For instance, some approaches use the robot’s proprioceptive observation [26], [21], [27], [28] to detect points of interest. While proprioceptive observations are strong indicators for keystates, they cannot universally detect keystates for all actions.

Waypoint reconstruction is used to obtain important keyframes from a long-horizon trajectory [26]. All these methods require knowledge about robot or environment states to extract keystates. UVD proposes to leverage foundation models trained on large-scale robotic datasets to identify keyframes by detecting phase shifts in the latent space of these models [3]. This allows for keystate identification based solely based on RGB observations. While this approach presents a strong baseline, incorporating additional information can significantly improve the keystate quality. We argue that access to robot proprietary information is a viable assumption, as this data is required to train a policy.

Changes in object relations and object states can also be a strong indicator for subtask completion. REFLECT constructs a scene graph containing object relations and states [29]. If a scene graph changes for two consecutive frames the frame is marked as a keyframe. This method works well with access to ground truth state information and object positions, usually only available in simulated environments. While we use a similar approach to detect keystates, our studies focus on the general applicability in real-world environments without these restrictions.

Action Recognition. Video action recognition is the task of retrieving an action performed over multiple frames. Video action recognition consists of two subtasks. Dense video action recognition aims to extract multiple actions and their corresponding time frames from a video. Video action classification assumes that the provided video only contains a single action. Recent advances in generalist VLMs enable them to solve these tasks in a zero-shot, open-vocabulary manner [30], [31], [32], [33], [34].

RoboVQA collects a dataset of long-horizon demonstrations [35]. Human annotators then divide the tasks into short-horizon tasks and label the sequences accordingly. The resulting data is used to finetune a video VLM on a VQA dataset derived from the labeled robot demonstrations. The model can then answer several questions regarding a video demonstration, including the action performed by the robot. Several recent works fine-tune CLIP foundation models using in-domain robotics data [36], [37], [23]. The finetuned models are used for goal-conditioned behavior learning or action retrieval for a larger, unlabeled dataset. These approaches assume known key states and require labeled in-domain finetuning data. LUPUS does not require any finetuning and is thus environment agnostic. Several studies [38], [39], [40], [41] train generalist visual representation models on large-scale egocentric datasets [42], [43]. Although the main purpose of these models is to provide a general representation for downstream policy learning, they can also be used for action retrieval, given their alignment of language and images during pretraining. Often, finetuning of the models is required to align language and images for new unseen environments.

B. Real World Experiments Description

The following section describes our real-world play kitchen environment in detail.

We collect play data through teleoperation in our robot kitchen environment. The setup is illustrated in Fig 5. The robot can solve 12 tasks in the environment, as shown in Fig. 6.

The dataset consists of 1 hour of pure play trajectories. Each trajectory consists of at least 10 different tasks that are completed randomly. The demonstration data contains 439 short-horizon demonstrations of 12 different tasks. In our evaluation, we distinguish between keystate and grounding evaluation. Additionally, we investigate the capability of a policy trained with automatically labeled data to understand language instructions. We compare the performance of this policy against one trained on the same dataset but with human annotations. We evaluate the performance of the trained policies based on two metrics:

Success Rate We perform each task three times and calculate the average number of successful task completions. We then compute the average success rate over all tasks.

Correct Grounding We evaluate whether the policy correctly understands language instructions. The task does not need to be completed successfully. The robot only has to show that it correctly understood the task. For instance, if the robot approaches the oven and tries to open it but fails, we label the task as correctly grounded. We again compute the average over all possible tasks.

We train three policies:

Human. A policy trained on ground truth, human-annotated data.

LUPUS. A policy trained on data labeled by LUPUS. We discard ambiguous labels.

LUPUS Noisy. If the LLM outputs multiple language instructions, we incorporate the demonstration in the training dataset multiple times with each generated instruction.

The grounding accuracies and success rates for each task are shown in Table II.

Task	Human	Lupus	Lupus Noisy
banana in sink	3-2	3-3	3-3
pot right	3-3	3-3	3-2
pot left	3-2	3-2	3-1
open microwave	3-3	2-2	0-0
open oven	3-0	3-0	3-0
open fridge	3-0	3-0	3-1
close microwave	3-3	3-3	2-2
close oven	3-3	3-2	0-0
close fridge	2-1	3-2	1-1
banana on stove	3-1	1-0	0-0
banana oven	0-0	0-0	0-0
pot in sink	3-2	3-0	3-0

TABLE II: Number of correct task groundings and successful task completions. The first number depicts the number of correctly grounded tasks, and the second the number of successful completions. We evaluate each task three times.

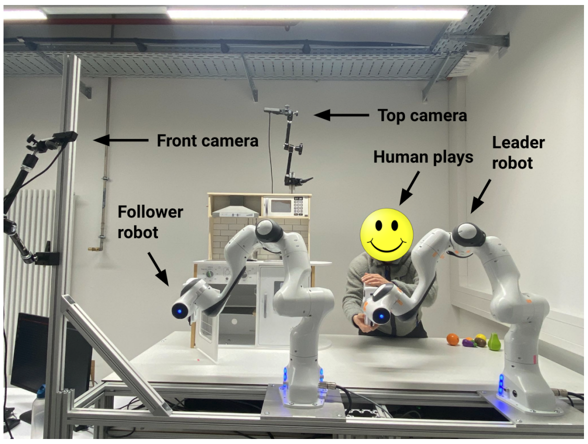

Fig. 5: Overview of the teleoperation setup on the real kitchen environment. The human operates on the leader robot. The follower robot imitates the actions of the leader. The top and front cameras record the play trajectory at 30Hz.

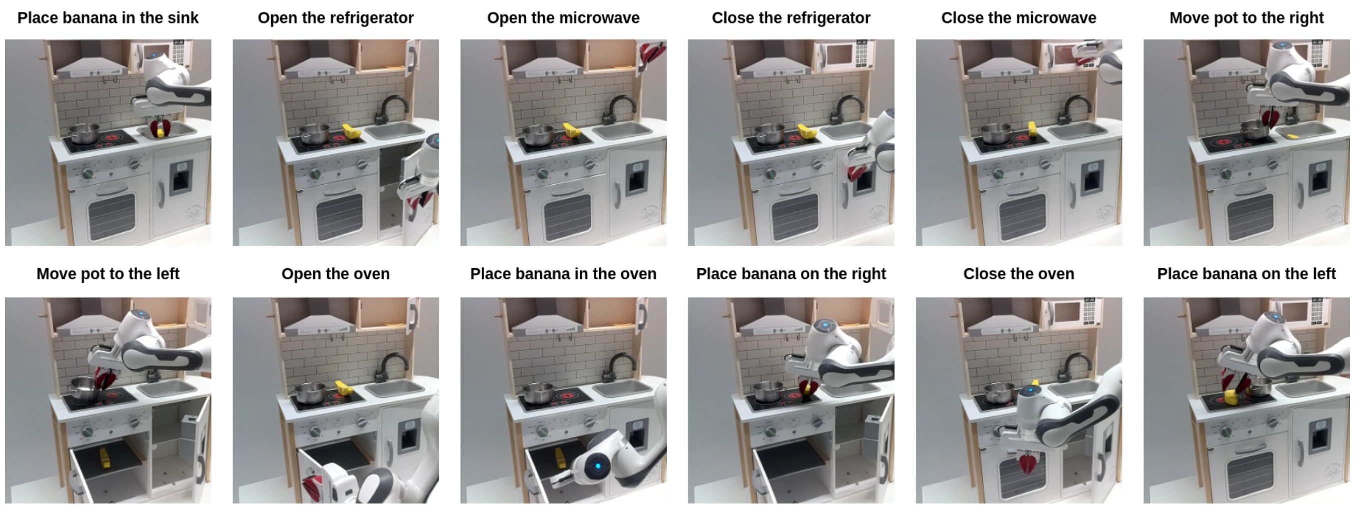

Fig. 6: Overview of the 12 tasks recorded during play from the preprocessed front camera perspective.

C. Real Robot Policy

For our experiments, we use the Multimodal Diffusion Transformer (MDT) policy architecture [14]. The model consists of a transformer encoder-decoder architecture and uses a continuous-time diffusion generative model to generate a sequence of 20 future actions. To encode the text instructions, a pre-trained CLIP text encoder is used, while images are encoded with FiLM-conditioned ResNets-18. We follow all hyperparameter recommendations from the paper for our own implementation and train the resulting policy on our real-robot dataset for approx. 400 epochs with a batch size of 512. Our policy learns to predict a sequence of joint state positions.

D. Additional Implementation Details

1) Scene Labeling: **Object Annotations and Segmentations.** LUPUS combines a state-of-the-art open vocabulary object detector, OWL-v2, [44] and CLIPSeg, an open-vocabulary semantic segmentation model [6]. The open vocabulary detector struggles with some classes, and its grounding confidence scores are not properly aligned for direct usage, resulting in incomplete and wrong annotations. To tackle this issue, LUPUS ensembles the segmentation model and detector: First, we extract bounding boxes for all objects extracted in Step 1 with a low detection threshold. LUPUS then computes the agreement between the object detector and dense predictor by summing the logits inside each proposed bounding box. This results in complete bounding boxes and more robust predictions, as illustrated in Figure **??**.
LUPUS further extracts optical flow with GMFlow [45] and metric depth estimates with DepthAnything [46] for all frames.

Increasing Detection Robustness for Static Objects. For non-moving objects, LUPUS employs a temporal consensus approach to enhance detection robustness and accuracy, especially in scenarios prone to occlusion. A two-step filtering process identifies the most representative bounding box for static objects over time, first by eliminating statistical outliers and then by clustering the remaining boxes using DB-SCAN [47]. The final bounding box for each static object is derived from the cluster with the highest overall confidence, representing the object consistently across frames.

Object Filtering and Mask Refinement through Temporal Aggregation. Initial object detections may suffer from temporal misalignments, such as missing detections for certain frames or an object being classified with a synonym for different frames. To address this challenge, LUPUS utilizes DEVA [8], a mask-tracking model, to capture temporal correlations between objects. LUPUS extends DEVA to incorporate a class score for each propagated mask. The resulting final mask belonging to each object has multiple class scores of possibly different associated classes. LUPUS obtains the most confident class and labels the object as the determined class. DEVA sometimes continues tracking a portion of the overlapping object as the occluded object. To mitigate this, LUPUS analyzes each mask's components and keeps the component with the highest intersection-over-union relative to the overall mask.

After applying DEVA and filtering, the masks are temporally more consistent and have consistent class labels.

2) Keystate Heuristics: **Gripper Position.** The gripper position over time can indicate robot-object interactions. Specifically, if the gripper is close to an object for a time span of n frames, the robot likely interacted with that object. To compute gripper-object proximity, we utilize the object segmentation mask of the robot and objects. This heuristic first estimates the end-effector position from a predicted depth map and then calculates end-effector object distances in pixel space.

State Prediction. LUPUS predicts the state of all objects over all timeframes and outputs a keystate if a state change is detected. LUPUS detects object states with SigLIP [5], a contrastive foundation model. We first obtain an image of the object at frame t by cropping the original frame with the object's bounding box. Then, we compare the CLIP similarities of state text embeddings defined by a large language model and the cropped images for non-occluded frames.

Object Relations LUPUS additionally analyzes object relations and determines keystates based on object-relation changes. LUPUS constructs an object-relation graph where nodes represent scene objects and edges denote their spatial relations, as inspired by [29]. Some spatial relations (e.g. *inside, behind*) require depth information. To address this, we project scene objects onto a point cloud using a predicted depth map [46], followed by canonicalization to reason about directions in natural language [48].

Object Movement LUPUS tracks object movement based on a predicted flow-map [49] and bounding-box displacement. We select keyframes based on object movement if the movement is above an object-specific adaptive threshold and occurs for at least three frames.

Gripper Close Signals. LUPUS can also consider gripper close signals as possible keystates, if available. Similar to prior work [26], [21], [27], [28], [25], our gripper close heuristic identifies a keystate when the gripper was previously closed for several frames and subsequently opens. Typically, this indicates that the robot has completed an interaction.

The employed keystate heuristics do not apply to all robot-object interactions. For instance, the state heuristic only applies to objects with states, the relation heuristic only applies to movable objects, and the gripper position heuristic often fails for small objects. Nevertheless, the heuristics complement each other, depending on the interacted object.

E. Ablations

		($\epsilon = 8$)		($\epsilon = 16$)		
		Amb.	Single	Amb.	Single	
Grounding	Naive	0.59	0.34	0.60	0.33	
	Naive - SG	0.62	0.27	0.59	0.26	
	- Temporal Alignment	0.76	0.53	0.68	0.51	
	- Detection ensembling	0.59	0.50	0.59	0.50	
	F.F.	0.80	0.61	0.75	0.57	
		Precision	Recall	Precision	Recall	mAP ↑
Keystates	Naive	0.46	0.36	0.67	0.53	0.36
	- Temporal alignment	0.41	0.45	0.70	0.77	0.45
	- Detection ensembling	0.46	0.48	0.69	0.73	0.48
	F.F.	0.50	0.46	0.75	0.69	0.51

TABLE III: Ablation for the effectiveness of our perception filtering. For Naive, we simply use OWL-v2 and SAM to extract masks and bounding boxes without additional filtering or temporal aggregation. In Naive-SG, we provide a full object-relation prompt to the LLM when retrieving the action. F.F. depicts full filtering.

In Table III, we provide ablations of our perception module. To assess the effectiveness of our perception module, we compare against simple box generation with OWLv2 and object segmentation with Efficient-SAM [7]. We perform ablations by disabling several components: ensembling with a dense open vocabulary predictor, statistical mask outlier filtering, temporal aggregation, state prediction without occlusion, and static object box aggregation. We perform all experiments with a keystate threshold of 0.3 and Gemini as the LLM.

When we omit our heavy postprocessing steps, we observe a significant decline in keystate quality and grounding accuracy. Although the drop in keystate precision is not substantial, the recall shows a notable decrease. Additionally, the grounding accuracy drops significantly, especially when only unambiguous prompts are considered valid. We observed that constraining the prompt information to a specific object and its relations helps to reduce hallucination and results in more precise predictions.

These findings underscore the necessity of robust postprocessing techniques to effectively leverage current state-of-the-art perception models in novel and challenging domains.

		$\epsilon = 8$		$\epsilon = 16$	
LUPUS Gripper	gripper close	0.35	0.32	0.73	0.65
	gripper close + object state	0.38	0.36	0.73	0.69
	all	0.50	0.46	0.75	0.69
LUPUS RGB	all	0.42	0.40	0.66	0.62
	object relations	0.21	0.13	0.56	0.35
	+ gripper pos.	0.29	0.40	0.52	0.71
	+ gripper pos. + state	0.39	0.32	0.62	0.51
	object movement	0.31	0.40	0.52	0.68
	+ gripper pos.	0.34	0.42	0.56	0.70
	+ gripper pos. + state	0.42	0.38	0.65	0.59
	gripper pos.	0.31	0.43	0.50	0.69

TABLE IV: Keystate precision and recall when using different keystate heuristics. For all experiments, the keystate detection threshold is set to 0.3, if applicable.

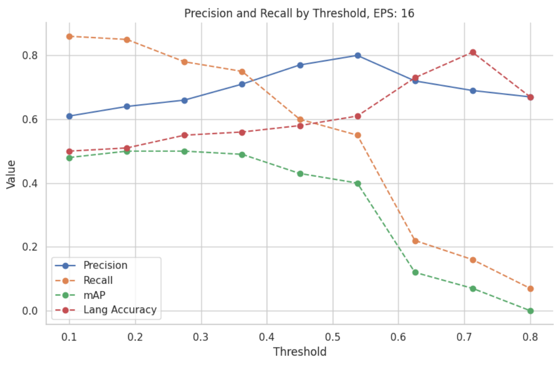

Fig. 7: Keystate detection precision and recall for different threshold values.

Table IV shows the performance of our method when incorporating different keystate heuristics. Gripper close signals present a very strong baseline. However, as mentioned before, gripper close signals are not always available and can not represent all different kinds of tasks. This is shown by the increased precision and recall when incorporating additional heuristics. Especially for a smaller threshold, we observe a significantly increased performance when incorporating additional heuristics.

Incorporating additional heuristics usually results in an increase in precision and a decrease in recall. Always using all heuristics is desired, as the precision-recall tradeoff can then be best controlled by setting an appropriate threshold.

Fig. 7 depicts the relation between threshold, keystate precision and recall, and grounding accuracy. With increasing threshold, the grounding accuracy and keystate precision increase. This indicates that with our scoring method, the quality of samples can be controlled effectively. In the future, we plan to evaluate the impact of different quality samples on policy training more thoroughly in a simulated environment.

F. Additional Experiments

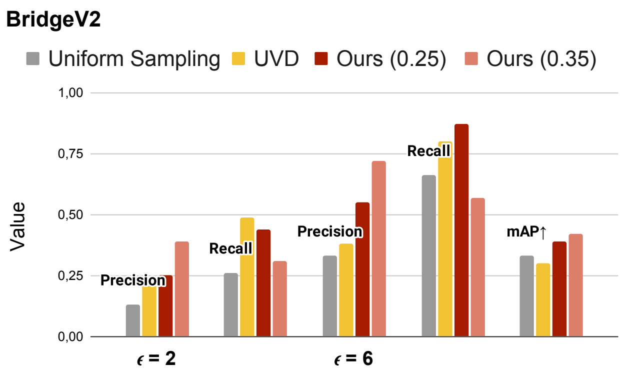

Fig. 8: Keystate accuracy for different frame distance tolerances on BridgeV2. We report the precision and recall of our method at two different keystate thresholds.

Method		$\epsilon = 8$ Precision	$\epsilon = 8$ Recall	$\epsilon = 16$ Precision	$\epsilon = 16$ Recall
UVD	VIP	0.16	0.06	0.28	0.10
LUPUS		0.37	0.21	0.53	0.31

TABLE V: Keystate accuracy for different frame distance tolerances on CALVIN with RGB Image-Data Only.

In Table V we show the keystate detection precision and recall on the CALVIN [22] benchmark. CALVIN is a challenging benchmark and is especially hard given the large domain shift of the low-resolution simulation. LUPUS utilizes off-the-shelf models trained on real-world data. Thus, these models struggle significantly in simulated environments that contain abstract objects. We had to perform prompt engineering to make the detection and OV-segmentation models detect any objects in the scene. Nevertheless, our method performs reasonably well. Fig. 8 shows the performance of our framework on BridgeV2 compared against the same baselines. Given the shorter average task length in BridgeV2, we opted for lower evaluation tolerance thresholds. We also assess the quality of keystates produced by our framework at two keystate thresholds, $\theta_o = 0.25$ and $\theta_o = 0.35$. Our approach surpasses both UVD and Uniform Sampling in terms of precision and mAP. A notable increase in the precision of our generated keystates is observed when the keystate threshold is raised to 0.35, suggesting that our method's keystate score can effectively manage keystate quality. One challenge with the BridgeV2 dataset is the frequent introduction of new objects by humans in between short-horizon demonstrations. Our current framework does not accommodate this, leading to some objects going undetected and consequently lowering our method's overall recall in this environment. We aim to enhance our framework in the future by incorporating a feature to verify the detection of all objects in the current frame.

We do not perform quantitative grounding evaluation on CALVIN and BridgeV2 due to the difficulties that arise when evaluating language commands in high-dimensional natural language task spaces. Multiple instructions can be considered valid, and evaluating the correctness of an instruction is not trivial.

G. Open-Ended Language Annotation

We show qualitative examples of our framework's produced natural language instructions on BridgeV2 [9] in Figs. 9–12.

We do not restrict the LLM to choose a task from a predefined list for this task. Instead, we task it to generate possible actions given the natural language descriptions generated by our method. As shown in the figures, LUPUS often generates useful tasks that could later be used for downstream policy learning. The examples show that lupus can produce clear language instructions, while Gemini Vision Pro suffers from heavy hallucinations. Evaluating these tasks automatically poses challenges due to the absence of a definitive ground truth. Multiple instructions can be deemed valid, and assessing predicted instructions necessitates an understanding of the scene or accurate knowledge of object locations. While a potential evaluation methodology is outlined in [50], the most dependable source of validation in such scenarios likely lies in human intervention.

H. Limitations

Perception. Our work shows that it is possible to leverage off-the-shelf specialist models to annotate challenging long-horizon data. The major limitations of our framework are induced by these off-the-shelf models. Commonly used robotic environments and their contained objects are still very challenging for state-of-the-art models. For instance, common evaluation environments in robotics are toy kitchens. Open-vocabulary detectors often struggle with grounding in such environments. For instance, our framework frequently detects the banana as a sponge in our toy kitchen setup. This hinders the applicability of our framework in challenging scenarios, such as BridgeV2. While there are models specifically applicable to the robotic domain, such as Spatial-VLM[48], RoboVQA [35] or PGBlip [51], these models are either not easily accessible or too specific for broader grounding applications.

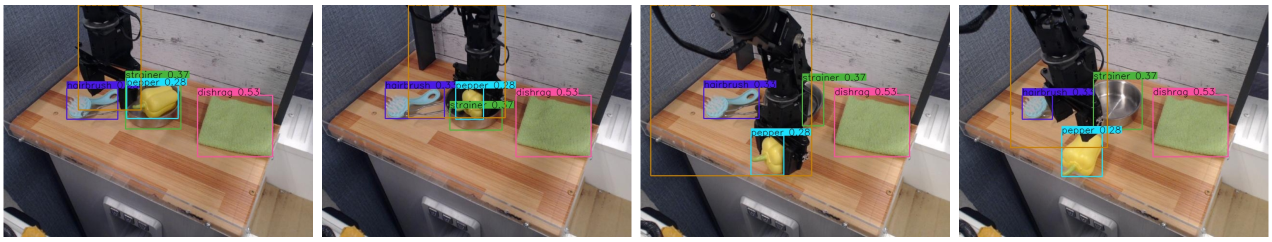

Fig. 9: Instructions generated by LUPUS: Move the pepper from inside the strainer to in front of the strainer, Take the pepper out of the strainer and place it forward, Relocate the pepper to a position in front of the strainer
Instructions generated by Gemini Pro Vision: Move the yellow bell pepper to the left, Place the yellow bell pepper in the pot, Move the pot to the right.

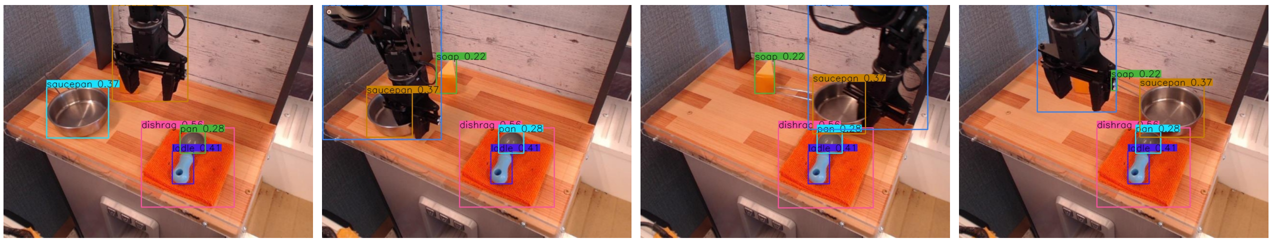

Fig. 10: Place the saucepan on top of the dishrag, Move the saucepan to the right of the soap, Position the saucepan behind the ladle.
Instructions generated by Gemini Pro Vision: Move the cheese to the right, Move the bowl to the right, Move the spoon to the right, Move the dishrag to the right

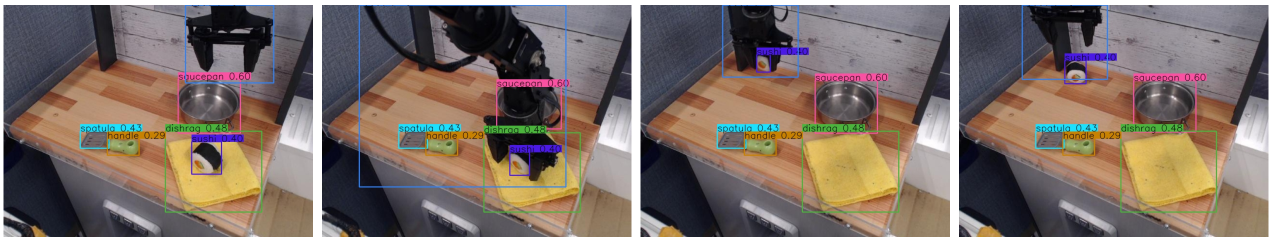

Fig. 11: Insturctions generated by LUPUS:
Move the sushi from on top of the dishrag to a new location away from the saucepan,Relocate the sushi to clear the area on top of the dishrag, Shift the sushi to organize the workspace, ensuring it is no longer next to the saucepan.

Instructions generated by Gemini Pro Vision: The robot moved the green spatula from the left of the cutting board to the right of the cutting board, The robot moved the yellow cloth from the right of the cutting board to the left of the cutting board, The robot moved the pot from the right of the cutting board to the left of the cutting board.

Furthermore, our initial object detection currently assumes that all objects are visible within 16 frames uniformly sampled over the long horizon trajectory. However, this assumption does not hold in some cases. We plan to extend our framework to be more robust in such cases.

Runtime. Using multiple different models to generate scene representations introduces substantial computational cost. The inference time of our framework is significantly higher compared to our baselines. However, the increase in performance justifies this overhead. Furthermore, the framework is designed to be applied offline to prerecorded play data, so computation time should not be much of an issue.

Objectness Assumptions LUPUS relies heavily on objectness assumptions and properties trackable with current foundation models. As such, the framework has issues with granular objects, which can not be detected reliably with current object detectors. Furthermore, the framework can not detect tasks that involve small movements, such as flipping a cup or pot upright. While these changes could be detected by monitoring the object state, these kinds of states are uncommon for such objects and, thus, are not generated by the LLM. However, with prior knowledge that such tasks can frequently appear in the dataset, the LLM prompt few shot examples for initial possible object state retrieval can be slightly modified to account for such cases.

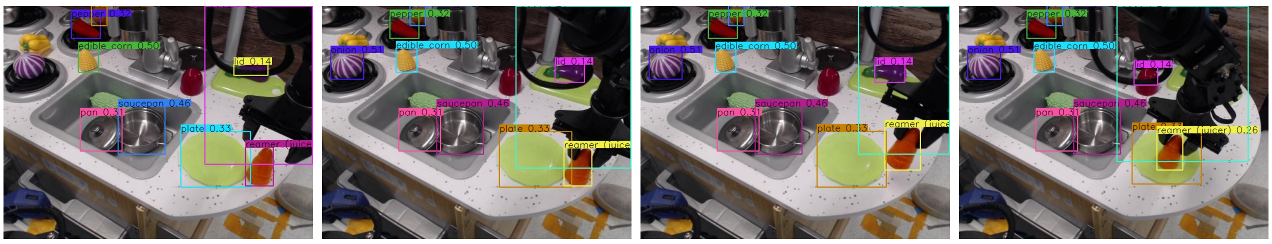

Fig. 12: Put the reamer (juicer) inside the plate, Move the reamer (juicer) from next to the plate to inside it, Place the reamer (juicer) into the plate for storage or preparation

Grounding Error. The initial object detections are wrong. Although the action is correctly predicted, the referenced object is not correct.

Instructions generated by Gemini Pro Vision: The robot picked up a carrot that was resting on a green plate and placed it in the sink. The robot moved a carrot from a green plate to the sink. The robot picked up a carrot and put it in the sink.

I. Example Prompts

We give example prompts used to generate the list of potential tasks given a list of objects in Fig. 13. In Fig. 14, an example prompt used to label the task a robot solved in between two keystates is given.

```
You will be provided with a list of objects
observed by a robot. Based on the objects, give
possible instructions to the robot. Infer the
type of environment from the provided objects.
Follow these guidelines:

- Keep the instructions simple. Focus on
tasks that only require a single step.
- Include tasks like placing an object inside
another object or moving the object. Only for
movable objects.
- Dont assume the presence of any objects not
listed.
Output at least 20 possible instructions
delimited by comma.

Here are a few examples: "Place the tin
can to the left of the pot.", "Move the dishrag
to the bottom of the table next to the towel","Put
the pot to the right of the fruit","Turn on
stove", "Open the microwave"

The following objects are in the environment:
[OBJECT_LIST]
```

Fig. 13: Task generation prompt

```
You will be provided with observations of a robot
interaction with an environment, delimited by
triple quotes.
Select a task from this list that best describes
the robots actions:
‘‘‘ [TASK_LIST] ‘‘‘
Follow these guidelines:
Step 1: Determine the object the robot interacted
with and then determine tasks that include that
object. Output the possible tasks after this step
delimited by commas.
Step 2: Determine the object movement and the
resulting object relations. Think about where the
object and its relational objects are located
in the scene on a global scale. Think step by
step and list the locations and relations of
all objects. Pay special attention to the object
relations from Step 1.
Step 3: Determine what tasks result in the object
relations from Step 2. Explain why the task
accomplishes the object relations. If the task
is not clear, output None. If multiple tasks are
possible, output multiple tasks with a low score.
Step 4: Some tasks do not have specific object
relations, but instead require moving objects
in some direction. Also consider these tasks by
examining the object movements.
Follow the steps above. Explain your reasoning.
Output the reasoning delimited by ***.

After, produce your output as JSON. The
format should be: ‘‘‘{ "task candidates":
"Possible tasks from the list after Step 1,
delimited by commas.", "tasks": "The tasks
that can be considered valid, delimited by
comma. Make sure to output all tasks that
match the description. Output up to 2 tasks.",
"confidence": "A confidence score for each
task between 0 and 10, delimited by commas. Be
pessimistic." }‘‘‘

Observations: ‘‘‘[OBSERVATIONS]‘‘‘
```

Fig. 14: Main action retrieval prompt

```
<Frame 1>...<Frame 8>
Given the video frames, what task did the robot
perform? Choose matching tasks from the list:
‘‘‘ [TASK_LIST] ‘‘‘
Sometimes multiple tasks are possible. Output all
possible tasks delimited by commas.
```

Fig. 15: Gemini Vision Pro Baseline Prompt