Language-Conditioned Semantic Search-Based Policy for Robotic Manipulation Tasks

Jannik Sheikh
Bielefeld University
Bielefeld, Germany
jsheikh@techfak.uni-bielefeld.de

Andrew Melnik
Bielefeld University
Bielefeld, Germany
andrew.melnik.papers@gmail.com

Gora Chand Nandi
Indian Institute of Information Technology
Allahabad, India
gcnandi@iiita.ac.in

Robert Haschke
Bielefeld University
Bielefeld, Germany
rhaschke@techfak.uni-bielefeld.de

Abstract

Solving various robotic manipulation tasks intelligently is a topic of great interest. Traditional reinforcement learning and imitation learning approaches require policy learning utilizing complex strategies that are difficult to generalize well. In this work, we propose a language-conditioned semantic search-based method to produce an *online search-based policy* from the available demonstration dataset of state-action trajectories. Here we directly acquire actions from the most similar manipulation trajectories found in the dataset. Our approach surpasses the performance of the baselines on the CALVIN benchmark and exhibits strong zero-shot adaptation capabilities. This holds great potential for expanding the use of our online search-based policy approach to tasks typically addressed by Imitation Learning or Reinforcement Learning-based policies.

1 Introduction

In recent years, the field of robotics has significantly evolved, with robots becoming more powerful, versatile, and interactive. Agents are no longer limited to performing repetitive industry-specific tasks. This change is driven and supported in large parts by research and successes in the areas of reinforcement learning Nguyen & La (2019), imitation learning Hussein et al. (2017), and recently by the immense progress in the field of natural language processing and computer vision Rana et al. (2023).

For any agent to be able to interact within an environment seamlessly depends largely on its ability to collect, process, and understand data that are largely unstructured. This data forms the agent's perception and guides its decisions, actions, and reactions. Among the innumerable sensory inputs an agent can use, visual data is an invaluable resource. What objects are in the environment, what objects are important to a task? Trivial for humans, but complex and challenging for robots, as it involves processes such as object recognition, object detection, or object localization using unstructured data.

Motivation Instead of the traditional approach of training a complex policy to solve specific tasks, our work explores a framework for solving various robot manipulation tasks by using a semantic search-based approach to generate an *online search-based policy*, inspired by the work of Malato et al. (2023), Beohar & Melnik (2022), Beohar et al. (2022), Rana et al. (2023).

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

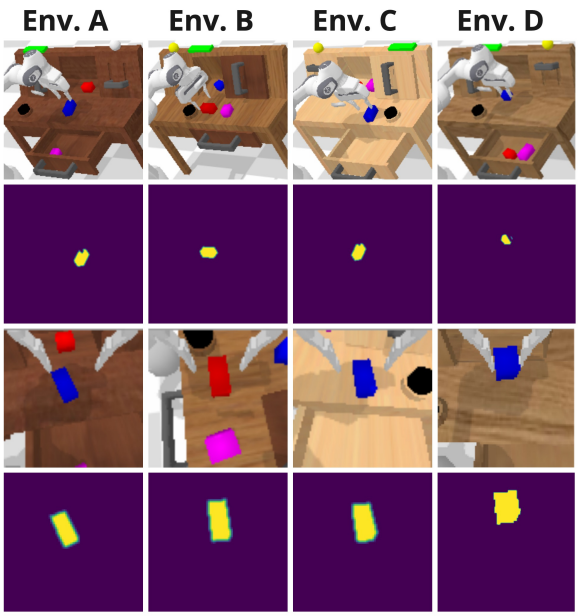

Figure 1: Overview of all four different environments in CALVIN. First row: RGB images of the static camera view. Second row: Latent representation of the static camera view of a specific object. Third row: RGB images of the gripper camera view. Fourth row: Latent representation of the gripper camera view of a specific object.

2 Method

Benchmark The CALVIN benchmark Mees, Hermann, Rosete-Beas & Burgard (2022) contains four different tabletop environments (A, B, C, and D) as seen in Figure 1. We evaluated our approach on 34 robotic manipulation tasks of the benchmark. The demonstration dataset of the benchmark was obtained from teleoperated "play" data, thus consisting of state x_i and action a_i pair trajectories τ. Therefore τ contains the exact information of how the agent, controlled by a human, got from some initial state x_0 to a goal state x_g, for completing a task. This guarantees that the goal state is reachable from the initial state under the performed actions. This results in a dataset $D_{\text{play}} = \{\tau | \tau = \{(x_i, a_i)\}_{i=0}^{n}$ and $0 \leq n \leq 64\}$

Those environments always contain a desk with stationary and movable objects to interact with, whose initial positions vary over the environments. A drawer and sliding door can be opened and closed. A button toggles an LED light and a switch operates a light bulb. Further three different-sized, colored, and shaped blocks are somewhere located on the desk. A 7-DoF robot arm with a parallel gripper is used to interact with the environment.

Instead of training a policy to solve tasks, we used search in the latent space of object shapes Melnik et al. (2021), Rothgaenger et al. (2023) to find similar states in the demonstration dataset (see Figure 2) and clone the actions from the most similar trajectory.

Masking In our approach, we experimented first with transforming the current state x_t obtained by the static and the gripper camera views into latent spaces z_{ts} and z_{tg} by color-based and low-level feature-based segmentation methods. To ensure that our framework achieves better generalization and scalability we further investigated the use of *Large Scale Models*, like *GLIP* Li et al. (2022) and FastSAM Zhao et al. (2023). We fine-tune GLIP on a small subset of examples based on the color-based and low-level feature-based segmentation method. GLIP leverages the pre-trained BERT text encoder Devlin et al. (2019) to condition the object detection on text. Since we have natural language instructions, these instructions can be used to identify the relevant objects that we want to encapsulate in our latent space. Subsequently, we use the result of GLIP to condition FastSAM to generate the mask of the object of interest.

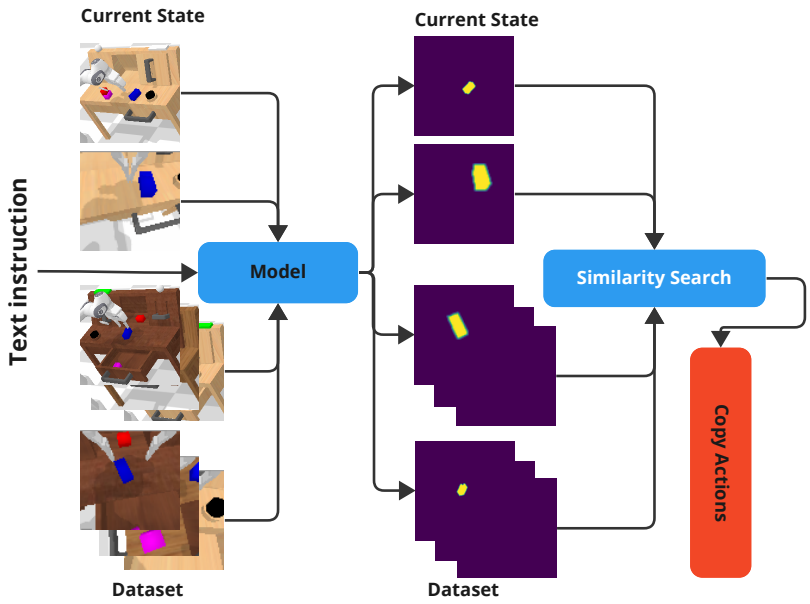

Figure 2: Overview of our approach. Given x_t, we obtain a binary mask of the object of interest in the static and gripper camera views and then compute sim_{zs} to find the most similar state in dataset trajectories and start cloning the corresponding actions.

Search We obtain reference images img_{ts} and img_{tg} from the static and gripper camera view, capturing the current state t in the environment. By passing img_{ts} and img_{tg} through our segmentation pipeline conditioned on text l we obtain two latent representations: z_{ts} and z_{tg}.

Each $\tau \in D_{\text{play}|l}$ consists of a series of x_{is} and x_{ig}, where x_{is} refers to the RGB images of x_i the static camera and x_{ig} of the gripper camera. Both x_{is} and x_{ig} give a visual representation of the agent's progress toward the target object. Analogous to our approach described for the reference images, these sequences can be processed to obtain the corresponding latent representations s_{is} and s_{ig}. Rather than processing every image-pairs within a given trajectory, we strategically select frames at regular intervals. This is motivated by the observation that successive steps often encapsulate similar information. By selecting images at every i^{th} step from τ, we further optimize for computational efficiency.

Similarity Measurement Given the reference latent representations z_{ts} and z_{tg} for state t, and the latent representations s_{is} and s_{ig} from a trajectory $\tau \in D_{\text{play}|l}$, we derive a weighted similarity coefficient as follows:

$$sim_{zs} = \alpha \cdot \text{score}(z_{tg}, s_{ig}) + (1 - \alpha) \cdot \text{score}(z_{ts}, s_{is}) \tag{1}$$

Here, the *score* is defined by the Dice coefficient:

$$\frac{2|A \cap B|}{|A| + |B|} = \frac{2\text{TP}}{2\text{TP} + \text{FP} + \text{FN}}$$

In addition, we scale the dice coefficient by a size coefficient. This coefficient serves to help identify latent representations containing objects of similar size, thus capturing the physical proximity of the robot arm to the objects. This primarily influences the relationship between z_{tg} and s_{ig}, since z_{ts} and s_{is} are always captured from the same distance to the table. When objects in the binary masks m_{tg} and m_{ig} have nearly equivalent sizes, it indicates that the robot arm is at a similar distance from the objects in both scenarios. The size coefficient, $size\_coef$, is then defined as the ratio between z_{tg} and s_{ig}.

Finally, the weighted dice coefficient is calculated as:

$$\text{weighted\_dice\_coef } (\textit{score}) = \text{dice\_coef} \times \text{size\_coef} \tag{2}$$

Initiating the Search Process Given that the length of each trajectory is finite and over time the observed state will differ from the initial search and, consequently, from the trajectory we copy, we keep track of the standard deviation of sim_{zs}. After each execution of $a_i \in \tau$, we collect the next state x_{i+1} from τ and generate s_{i+1s} and s_{i+1g}. Concurrently, we obtain the newly observable state in the environment x_t, from which we derive z_{ts} and z_{tg}. We then compute sim_{zs} and store the results. If the change in the standard deviation over the last two steps exceeds a certain threshold T or if there are no actions left in the current trajectory, we trigger a new search with the current state x_t.

Switching Trajectories If the similarity score sim_{zs} between z_{ts},z_{tg} and s_{is}, s_{ig} is greater than the similarity value of the currently pursued trajectory, we switch to the i^{th} step of that new trajectory. Moreover, if there are no actions left in the current trajectory we pursue, we switch to τ corresponding to the highest sim_{zs} computed in the given step.

Executing Actions Our action set has both absolute (a_{abs}) and relative (a_{rel}) actions. a_{abs} enable long-range movements that allow the agent to quickly reduce the distance to the target object. In contrast, a_{rel}, allows finer movements that are important for local control and adjustment. On this basis, we execute a_{abs} primarily at the beginning of an evaluation or when z_{tg} is a zero vector. This indicates that the object of interest is not visible in the gripper camera. However, once z_{tg} contains non-zero values and our similarity score sim_{zs} exceeds a certain threshold, we switch to a_{rel}.

3 Results

CALVIN offers different evaluation environments. We are particularly interested in getting correct output in the zero-shot multi-environment scenario, where the agent is trained on three of the four environments, A, B, and C, and evaluated on the unseen environment D (see Figure 1). Each evaluation starts by resetting the simulator to the initial state of an unseen demonstration. As seen in Table 1, our method outperforms the previous best methods MCIL Lynch & Sermanet (2021) and HULC Mees, Hermann & Burgard (2022).

Method	**Input**	**Score**
Baseline	Static RGB & Gripper RGB	30.4%
HULC	Static RGB & Gripper RGB	41.8%
Ours	Static RGB & Gripper RGB	**50.3**%

Table 1: Zero-Shot Multi Environment.

4 Conclusion

In this work, we propose a method for solving various robot manipulation tasks using semantic search in the demonstration dataset and copying actions from the best matching state. We show that the proposed method generalizes for multi-environments. For better generalization and scalability, further research needs to be conducted to verify the proposed method for different domains and problems. Also, we may try evaluating the weighted similarity coefficient in the exponential domain to see if it can further improve the results. Large-scale pre-trained models seamlessly align with the suggested architecture, holding significant promise for capturing the latent representation of semantic segmentation of language-conditioned objects of interest.

5 Limitations

Due to efficiency reasons, we could not use the entire dataset for our search method. To address this issue, future research could explore options such as using pre-generated datasets and merging clustering and indexing techniques to speed up the process.

References

Beohar, S., Heinrich, F., Kala, R., Ritter, H. & Melnik, A. (2022), 'Solving learn-to-race autonomous racing challenge by planning in latent space', *arXiv preprint arXiv:2207.01275* .

Beohar, S. & Melnik, A. (2022), 'Planning with rl and episodic-memory behavioral priors', *arXiv preprint arXiv:2207.01845* .

Devlin, J., Chang, M.-W., Lee, K. & Toutanova, K. (2019), 'Bert: Pre-training of deep bidirectional transformers for language understanding'.

Hussein, A., Gaber, M. M., Elyan, E. & Jayne, C. (2017), 'Imitation learning: A survey of learning methods', *ACM Computing Surveys (CSUR)* **50**(2), 1–35.

Li, L. H., Zhang, P., Zhang, H., Yang, J., Li, C., Zhong, Y., Wang, L., Yuan, L., Zhang, L., Hwang, J.-N., Chang, K.-W. & Gao, J. (2022), 'Grounded language-image pre-training'.

Lynch, C. & Sermanet, P. (2021), 'Language conditioned imitation learning over unstructured data'.

Malato, F., Leopold, F., Hautamaki, V. & Melnik, A. (2023), 'Behavioral cloning via search in embedded demonstration dataset', *arXiv preprint arXiv:2306.09082* .

Mees, O., Hermann, L. & Burgard, W. (2022), 'What matters in language conditioned robotic imitation learning over unstructured data', *IEEE Robotics and Automation Letters* **7**(4), 11205–11212.

Mees, O., Hermann, L., Rosete-Beas, E. & Burgard, W. (2022), 'Calvin: A benchmark for language-conditioned policy learning for long-horizon robot manipulation tasks', *IEEE Robotics and Automation Letters (RA-L)* **7**(3), 7327–7334.

Melnik, A., Harter, A., Limberg, C., Rana, K., Sünderhauf, N. & Ritter, H. (2021), Critic guided segmentation of rewarding objects in first-person views, *in* 'KI 2021: Advances in Artificial Intelligence: 44th German Conference on AI, Virtual Event, September 27–October 1, 2021, Proceedings 44', Springer, pp. 338–348.

Nguyen, H. & La, H. (2019), Review of deep reinforcement learning for robot manipulation, *in* '2019 Third IEEE International Conference on Robotic Computing (IRC)', pp. 590–595.

Rana, K., Melnik, A. & Sünderhauf, N. (2023), 'Contrastive language, action, and state pre-training for robot learning', *arXiv preprint arXiv:2304.10782* .

Rothgaenger, M., Melnik, A. & Ritter, H. (2023), 'Shape complexity estimation using vae', *arXiv preprint arXiv:2304.02766* .

Zhao, X., Ding, W., An, Y., Du, Y., Yu, T., Li, M., Tang, M. & Wang, J. (2023), 'Fast segment anything'.