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

Reinforcement learning and Imitation Learning approaches utilize policy learning strategies that are difficult to generalize well with just a few examples of a task. 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. Project webpage: https://j-sheikh.github.io/behavioral-search-policy

1 Introduction

In recent years, the field of robotics has significantly evolved, with robots becoming more powerful, versatile, and interactive, due to progress in the field of natural language processing, computer vision (Rana et al. 2023), reinforcement learning (Schilling & Melnik 2019, Nguyen & La 2019, Bach et al. 2020), and imitation learning (Hussein et al. 2017).

Motivation The ability of any agent to interact within an environment seamlessly depends largely on its capacity to collect, process, and understand data that are largely unstructured. This data forms the agent's perception and guides its decisions, actions, and reactions. Instead of the traditional approach of complex training of a 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).

2 Data

The CALVIN benchmark (Mees, Hermann, Rosete-Beas & Burgard 2022) contains four different tabletop environments (A, B, C, and D) as seen in Figure 1. Those environments always contain a desk with stationary and movable objects to interact with, whose initial positions vary over the

NeurIPS 2023 Workshop on Foundation Models for Decision Making, New Orleans

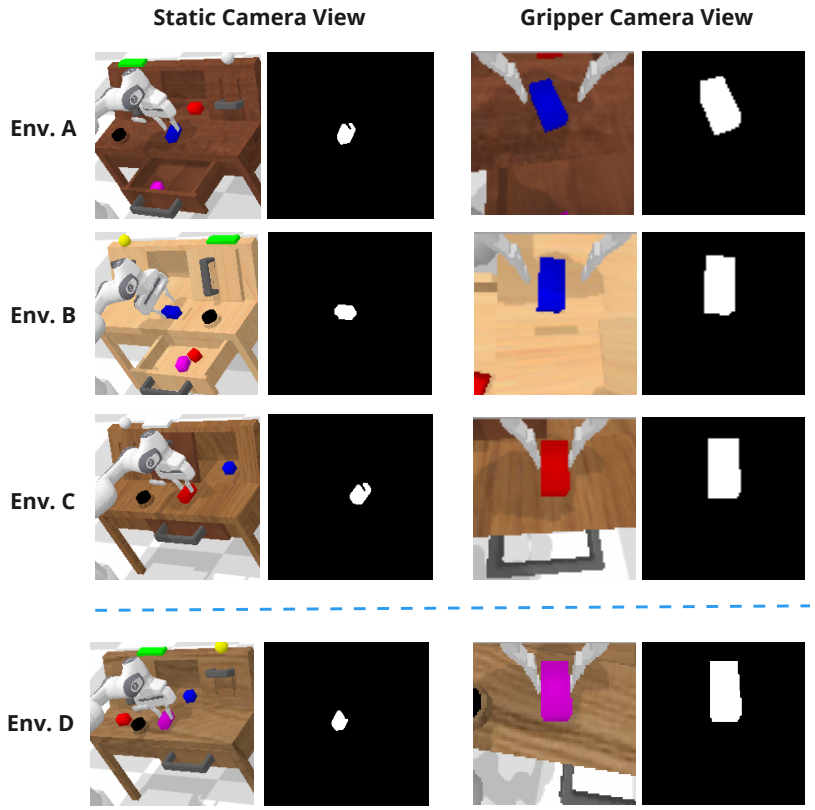

Figure 1: Overview of all four different environments in CALVIN. During inference, the *Search-Based Policy* searches for the most similar state in environments A, B and C with respect to the current state from environment D.

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.

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 $0 \leq n \leq 64\}$.

The authors labeled less than 1% of the collected data, making it possible to identify trajectories that correspond to specific tasks. By annotating these trajectories, they made it possible to describe any of the trajectories, and thus the corresponding tasks, by natural language instructions.

3 Method

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 identify similar states in a demonstration dataset, and after finding similar representations for a given scene and text task, we clone the corresponding actions to solve the given tasks until the divergence threshold is exceeded between the current state and the selected trajectory.

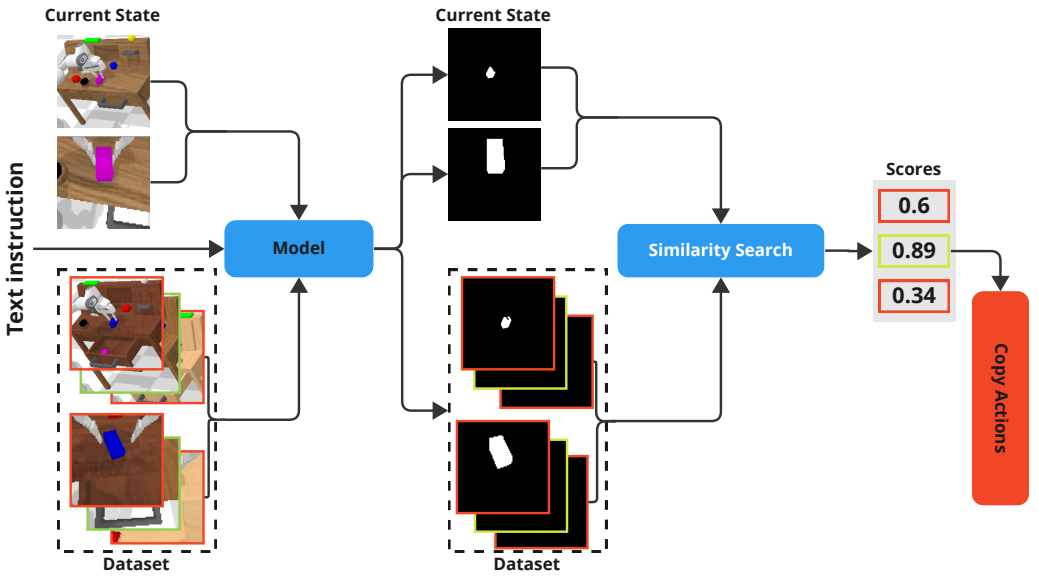

Figure 2: Overview of our framework. Given x_t, we obtain a binary mask of the object of interest in the static and gripper camera views and then compute sim_{zs} to find the most similar state in dataset trajectories and start cloning the corresponding actions.

Masking Since all objects over all the environments are of color, we first experimented 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. Concretely, we use the *HSV (Hue, Saturation, and Value)* color space to detect and segment objects in our environments. We further enhanced our segmentation by adding positional filtering. Concretely, since there exist three different gray color handles we consider the position of the object in the image to help identify the correct one. This is easily applicable to the static camera view. The gripper camera added additional complexity because this camera is constantly in motion. Here we considered the surrounding pixels as well as the area and orientation of the object to make better and more accurate decisions. To be able to find the object of interest in our dataset, we apply a mapping to recognize the objects of interest in different environments based on the task description.

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}. 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 contains at most 64 state-action pairs 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 pursued 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. Once z_{tg} contains non-zero values and our similarity score sim_{zs} exceeds a certain threshold, the agent switches to a_{rel}.

4 Results

CALVIN benchmark offers different evaluation environments. The search-based policy uses the dataset that contains only three of the four CALVIN environments (A, B, and C), and is evaluated on the unseen environment D (see Figure 1).

Evaluation Settings We evaluated the agent's performance in two different ways (see Table 1):

1. The agent performs each of the 34 tasks for 10 rollouts. Every evaluation starts by resetting the environment and agent to the initial state x_0 of an unseen demonstration.
2. The agent is evaluated over 1000 individual task instructions. At the beginning of each task evaluation, the robot arm is placed in a neutral position and the environment is initialized. The agent's goal is to successfully perform the given task within a maximum of 360 steps.

There are two baseline models, MCIL (Lynch & Sermanet 2021) and HULC (Mees, Hermann & Burgard 2022), although HULC is evaluated only in the second setting.

Hyperparamters: For both evaluation settings we assign a value of 0.9 to α (Eq. 1), emphasizing the latent representation of the gripper camera in our search. This choice is influenced by our intention to prioritize the gripper camera as soon as an object is detected in its view to allow precise maneuvers and finer adjustments. The static camera primarily helps in approaching the object and provides slight guidance in the further course. In order to fasten the processing time of finding the most similar representation in the training data, we uniformly take only 8% of the annotated data from each of the three environments A, B, and C, and search within this subset. Finally, we only consider every fourth image in a trajectory. We trigger a new search if the standard deviation of sim_{zs} is greater than 0.03. If z_g is empty, we set the threshold to 0.003 based on the Equation 1.

Method	Input	Success Rate First Setting	Success Rate Second Setting
Baseline	Static RGB & Gripper RGB	38%	30.4%
HULC	Static RGB & Gripper RGB		41.8%
Ours	Static RGB & Gripper RGB	**61.4**%	**57.2**%

Table 1: Combined results for the Zero-Shot Multi Environment in different evaluation settings.

First evaluation setting): As seen in Table 1 our proposed method outperforms the current baseline by more than 23% in the first setting.

A detailed breakdown across all tasks can be found in Table 2.

Task	Success Rate	Task	Success Rate
push pink block left	100%	rotate pink block left	80%
push red block left	100%	rotate red block left	90%
push blue block left	70%	rotate blue block left	30%
push pink block right	90%	rotate pink block right	40%
push red block right	20%	rotate red block right	70%
push blue block right	80%	rotate blue block right	40%
push into drawer	0%	unstack block	70%
lift pink block drawer	90%	stack block	0%
lift red block drawer	70%	turn on led	90%
lift blue block drawer	90%	turn off led	50%
lift pink block slider	50%	turn on lightbulb	70%
lift red block slider	20%	turn off lightbulb	80%
lift blue block slider	10%	place in drawer	100%
lift pink block table	40%	place in slider	30%
lift red block table	30%	move slider right	80%
lift blue block table	50%	move slider left	70%
open drawer	100%	close drawer	90%

Table 2: Our results over all tasks in the first evaluation setting.

Second evaluation setting: The results for the second evaluation setting are also shown in Table 1.

The agent completes 75% of the tasks in 125 steps or fewer. It is important to emphasize that we set the maximum step size of our evaluation to 180. This decision arises from the observations of prior experiments, which indicate that the probability of failure is high if our strategy does not solve the task within this step range, and also to further reduce the computation time for each task.

Large-Scale Models Since we have natural language instructions that can serve as targets x_g, these instructions can be used to identify the relevant objects that we want to encapsulate in our latent space. With large language models, we can encode our task instructions and use the resulting latent representation to drive the segmentation process, which can also be replaced with an additional foundation model to identify the objects needed to solve the task.

We leverage the large pre-trained text embedding model GTE_{base} (Li et al. 2023) which is based on the BERT framework (Devlin et al. 2019) and can be used for various downstream tasks. We further fine-tuned the model for three epochs using an A100-GPU using the task instructions of our training data. Afterward, we use the fine-tuned model to generate embeddings of size $\mathbb{R}^{768}$ for our train and test instructions. Our results are visualized in Figure 3 for the train instructions and in Figure 4 for the test instructions. In order to objectively measure the quality and separability of the clusters generated from our embeddings, we further report the Silhouette Score, Adjusted Rand Index (ARI), and Normalized Mutual Information (NMI) from scikit-learn (Pedregosa et al. 2011). The silhouette score ranges from -1 to 1, with a high value indicating that the embedding fits well with its own cluster and poorly with neighboring clusters. ARI measures the similarity between true labels and predicted labels and NMI measures the similarity between true labels and predicted labels. The clustering results of our embeddings are shown in Table 3. These scores indicate an almost perfect separation.

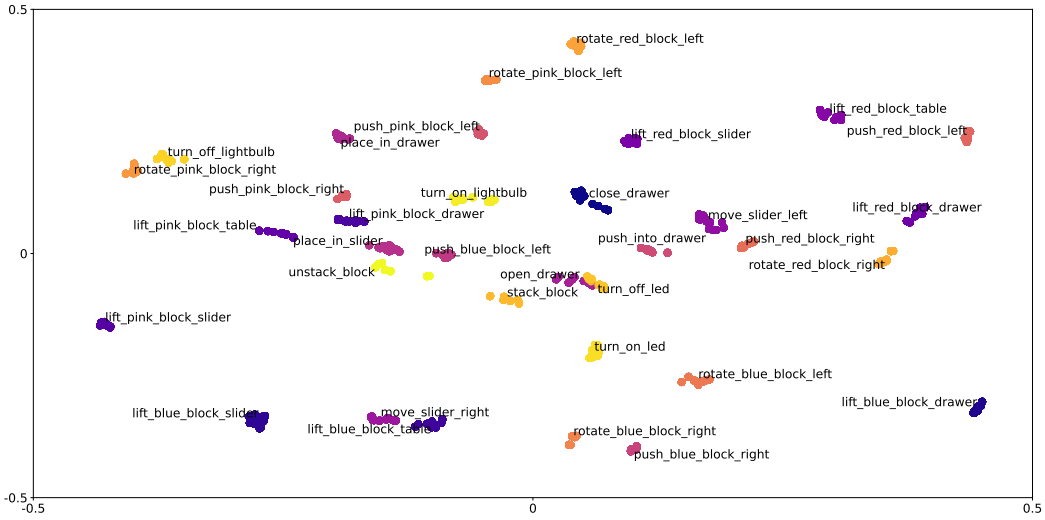

Figure 3: Visualization of the clustered natural language instructions with PCA in a 2D space. For clustering, we fit K-Means to the train embeddings of size $\mathbb{R}^{768}$ generated by the fine-tuned GTE model (Li et al. 2023) and set the number of clusters to k, where k represents the total number of tasks, 34. Each data point represents a unique natural language instruction corresponding to a task, and the cluster labels denote the respective tasks. The plot shows the embeddings from our training data.

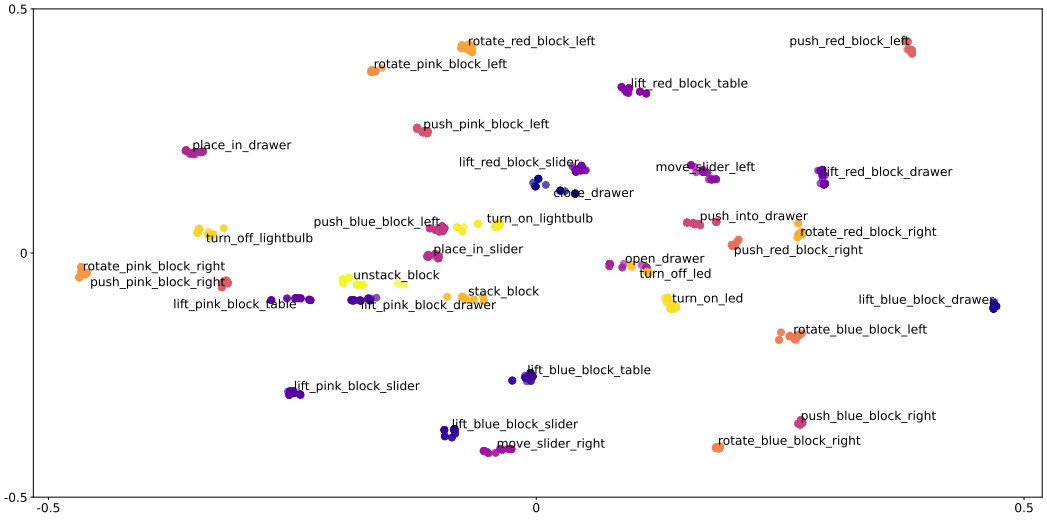

Figure 4: Visualization of clustered natural language test instructions with PCA in a 2D space. We generate the embeddings from the fine-tuned GTE model (Li et al. 2023) and use the fitted K-Means algorithm to predict the clusters of the test embeddings of size $\mathbb{R}^{768}$.

Metric	Score
Silhouette Score	0.943
Adjusted Rand Index	1.0
Normalized Mutual Information	1.0

Table 3: Clustering Evaluation Metrics

5 Discussion

Our work shows that using an *online search-based policy* that exploits latent representations achieves notable success in solving a variety of robot manipulation tasks. By searching for similar latent representations in a demonstration dataset and mirroring the associated actions, our proposed method outperforms the current baseline models in both evaluation settings and generalizes for multi-environments. This highlights the compelling effectiveness of using a search-based policy within the latent space, a result consistent with the research of Malato et al. (2022, 2023) within the dynamic world of Minecraft.

The first evaluation setting provided insight into the overall effectiveness of our approach across all tasks and highlighted its consistent performance over numerous rollouts. The second evaluation setting demonstrates the robustness and efficiency of our method, as the agent navigated from a neutral position to the target object and then completed the task reasonably fast. This highlights the potential to address robot manipulation challenges without relying on complex reinforcement or imitation learning policies.

The performance differences between similar tasks (see Table 2) with differently sized blocks indicate that the decision process for starting a new search and transitioning to an alternative trajectory requires more research to enable better precision and adaptive interactions. When interacting with the large (pink) block, the success rate across all corresponding tasks is almost 20% higher than when interacting with the small (blue) block. Possible improvements could include the use of specific cost, exponential function, or other non-linear functions. Such approaches could offer advantages in effectively modeling the relationships between the latent representations of the static and gripper cameras and improve decision-making.

We further had to reduce the dataset due to the computing time when performing our search. The search process itself to find the most similar state in $D_{\text{play}|l}$ is performed on the CPU and takes between 1.4 and 1.9 ms. Future research could investigate the effectiveness of initially generating a dataset of latent representations and then using clustering and indexing techniques to improve search speed. This approach provides flexibility to easily incorporate new latent representations and thus allows for new tasks to be performed.

Finally, foundation models seamlessly align with our suggested architecture (see Figure 2). Models such as GLIP (Li et al. 2022) and FastSAM (Zhao et al. 2023) show promising capabilities in generating latent representations of the object of interest conditioned by natural language. Further research is needed to evaluate such models in combination with our *online search-based policy*.

6 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 trajectory. We show that the proposed method generalizes for multi-environments.

References

Bach, N., Melnik, A., Schilling, M., Korthals, T. & Ritter, H. (2020), Learn to move through a combination of policy gradient algorithms: Ddpg, d4pg, and td3, *in* 'Machine Learning, Optimization, and Data Science: 6th International Conference, LOD 2020, Siena, Italy, July 19–23, 2020, Revised Selected Papers, Part II 6', Springer, pp. 631–644.

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'.

Li, Z., Zhang, X., Zhang, Y., Long, D., Xie, P. & Zhang, M. (2023), 'Towards general text embeddings with multi-stage contrastive learning'.

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* .

Malato, F., Leopold, F., Raut, A., Hautamäki, V. & Melnik, A. (2022), 'Behavioral cloning via search in video pretraining latent space', *arXiv preprint arXiv:2212.13326* .

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.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M. & Duchesnay, E. (2011), 'Scikit-learn: Machine learning in Python', *Journal of Machine Learning Research* **12**, 2825–2830.

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* .

Schilling, M. & Melnik, A. (2019), An approach to hierarchical deep reinforcement learning for a decentralized walking control architecture, *in* 'Biologically Inspired Cognitive Architectures 2018: Proceedings of the Ninth Annual Meeting of the BICA Society', Springer, pp. 272–282.

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