Efficient Cross-Embodiment Learning with Object-Centric Planner

Sungjae Park

2022.12.22

Seoul National University

Thesis Advisor:

Joseph J. Lim, Korea Advanced Institute of Science and Technology (KAIST)

Frank C. Park, Seoul National University (SNU)

1 Introduction

Learning from data has been a key component towards the goal of general-purpose robots, as recent advances in other fields have shown remarkable success with machine learning. However, data is expensive in robotics due to multiple factors, such as the difficulty of teleoperation, safety issues when deploying robots in the real world for automatic data collection, the price of the robot, etc.

To tackle such data scarcity, learning from indirect data, data that does not come from the same setup, has been a promising approach. Such indirect data includes data from different environments[1], tasks[2, 3], and embodiment[4, 5, 6]. In this paper, we focus on learning from different agents: **Cross-Embodiment Learning**, in the domain of robotic manipulation. From an offline dataset that consists of a source robot demonstration performing a specific task, the goal of our approach is to efficiently learn a target robot's policy for the same task. Additionally, we do not assume access to target robot action labels, making our problem setup similar to learning from observation.

When it comes to learning from indirect data, a natural but important question that should be answered is: **"What is the information that can be shared across different agents when they are trying to solve the same task?"**. To answer this question, it is worth thinking about the fundamental purpose of manipulation tasks. Following the intuition of manipulation task primitives[7] which defines primitive manipulation as a relative motion between two rigid bodies, we claim that the fundamental purpose and specification of a majority of manipulation tasks is to move objects along particular trajectories. We further claim that such task specification is sharable between different agents, and is rich with diverse utility.

Our problem setup has two phases. First, we learn an object-centric trajectory planner(**OCP**) with object-centric representation that plans the object's movement to achieve the task given an image observation. This is learned from the task demonstration data from a source robot. Second, we simply utilize this model with a control loop along with a grasping primitive, to achieve the planned object movement for solving the target task with the target robot in a zero-shot manner. Additionally, we show that the learned OCP can be used in various ways other than a simple control loop. While prior works[4, 5] tackling cross-embodiment learning is limited in that the learned model only serves as a reward function, we show that it can provide learning signals by providing dense rewards or simplifying action space. We experiment with such utilities of our model in a source robot for simplicity.

In short, our contributions are: we propose a novel method of leveraging different robot's demonstration, and show its diverse utilities compared to previous methods.

2 Preliminaries and Related Work

Markov Decision Process. A Markov Decision Process (MDP) is denoted as a tuple $< S, A, P, R, \gamma >$, where each component means state space, action space, transition probability, reward, and discounting factor respectively. When a model (transition probability p and reward r) is fully known, the MDP problem becomes a stochastic control problem. When a model is unknown, the MDP problem becomes an RL problem. The goal of reinforcement learning (RL) algorithms is to find an optimal policy $\pi(a|s)$ that makes the agent select actions over time to maximize the expected cumulative reward $\max_{\pi} E_{a\sim\pi,P}[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t, s_{t+1})]$.

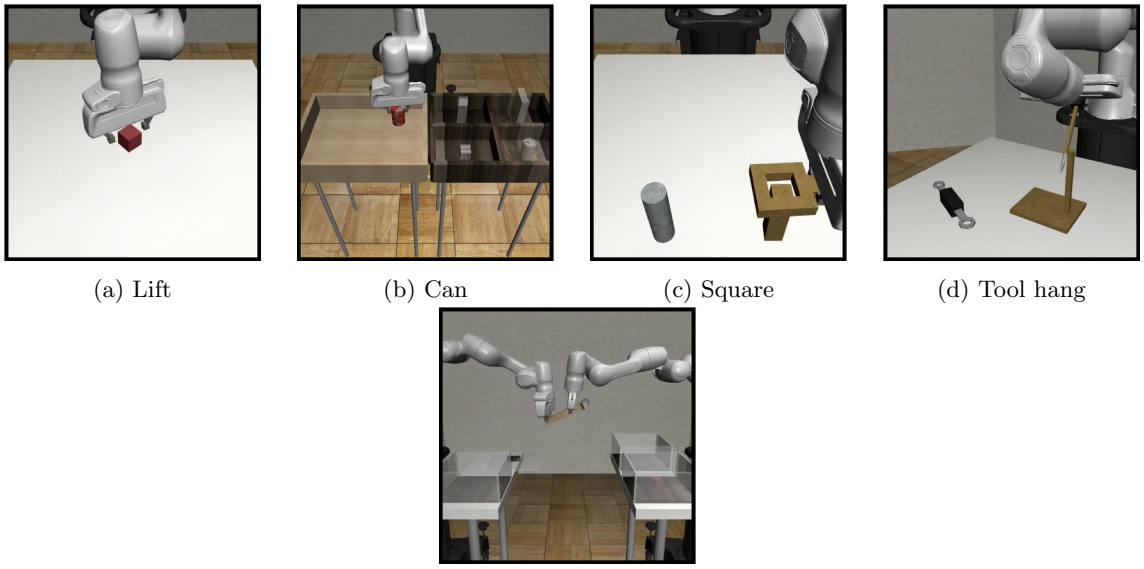
Manipulation Task Primitives. To identify a sufficient set of manipulation tasks to span manipulation tasks we care about, manipulation task primitive[7] is defined and classified by the relative motion between two (rigid) parts. Such primitives are classified into a small number of categories, depending on whether the individual frames are free or constrained. For instance, circular peg insertion is a motion between a peg and hole, with 2 motions possible (rotational, transitional) along the axis of insertion. Door opening is usually a motion with single degrees of freedom, where the doorknob can be rotated in a desired way. Manipulation task primitives have shown their effectiveness in representing manipulation tasks while being efficient, as only twenty different relative motions are possible, a surprisingly small number of classes.

Inspired by such task specification, we claim that a majority of manipulation tasks have an identical purpose: moving a target object relative to another object or world in a desired way. For instance, putting cabbage in a pot, placing a bottle, and picking up a spoon can be defined as Figure 1. The task is naturally specified by: a target object to manipulate, and a specific target object motion to follow. Such task specification is naturally agent-agnostic, and we hope to extract such information from the source robot's demonstration in our work.

Figure 1: **Object-Centric Trajectories as Task Specification.**

Cross Embodiment Learning. Cross-embodiment learning has gained interest as it has the potential to leverage data collected from diverse agents, including humans. Previous works[5, 4, 6] tackle such problem, but are limited in several aspects. First, the learned model from source robot data is limited in its use. For instance, learned models are only limited to be used as a reward model[5, 4]. Additionally, some works make radical assumptions[6], such as frame-to-frame correspondences between expert demonstration videos and learned embodiments. Compared to previous works, our work is different in that it can be used in diverse ways other than just the reward model, and it does not assume frame-to-frame correspondence.

Pose Estimation. 6D pose estimation has been explored for a long time, with a focus on deep learning for the last few years. Xiang et al[8] introduce PoseCNN using CNN network for semantic labeling and object 6D pose estimation, and Park et al[9] suggest predicting 3D coordinates of each object pixel for object pose estimation by leveraging the colored coordinate model of the object. He et al[10] proposes a framework for few-shot pose estimation of novel objects. While our method leverages object pose labels directly acquired from simulation, the recent advances in object pose estimation supports that it is feasible to get object pose labels for learning object trajectory planner from source robot demonstrations.



(a) Lift (b) Can (c) Square (d) Tool hang

(e) Transport

Figure 2: **5 Tasks in Robomimic Dataset.**

3 Learning an Object-Centric Trajectory Planner from source robot demonstration data

In this section, we introduce how to utilize the demonstration data of the source robot. As aforementioned, we assume that the object trajectory a source robot achieves can be transferred to the target robot as a rich task specification. For brevity, we also use the term object-centric planner and OCP to refer to object-centric trajectory planner.

Environments. To start with, we used the Square task from Robomimic[11], a simple peg insertion task. Robomimic is a framework for evaluating imitation learning and offline RL algorithms for robot learning, consisting of 5 representative tasks (Lift, Can, Square, Tool Hang, Transport) with different difficulties and their demonstration data. We utilize a proficient human dataset, which consists of 200 demonstrations collected by a human expert. Figure 2 shows examples of tasks and environments in the Robomimic dataset. In the Square(peg insertion) task, an agent's goal is to insert the square ring in the square rod. The source robot is the Franka Panda robot and 200 expert demonstration trajectories from the source robot for the peg-insertion task are provided in the Robomimic dataset. We only assume access to image observation and pose information of the object to manipulate from the provided dataset. OCP takes image observation as an input and predicts the target object's trajectory for a fixed horizon to solve the target task. For the target robot, we chose Sawyer.

Preprocessing. When the robot is not manipulating the object, the object stays still if no perturbations are coming from the environment. In such cases, the ground truth of the object trajectory is static, and the learned trajectory planner would also predict the object to be static. This is problematic as we use the planned object trajectory for controlling the target robot to achieve the plan, and it would result in the target robot not moving at all. Hence, we remove observations from the source robot demonstrations when the source robot is not manipulating the object and use the remaining observations as training data. Furthermore, we normalize the position of the object.

Object-Centric Trajectory Planner (OCP). Our model, OCP, plans the object's trajectory (position/pose w.r.t. time), using an LSTM[12] and Variational Encoder[13]. The overall model architecture for the planner is in Figure 3. We have learned an object's position trajectory or a pose trajectory.

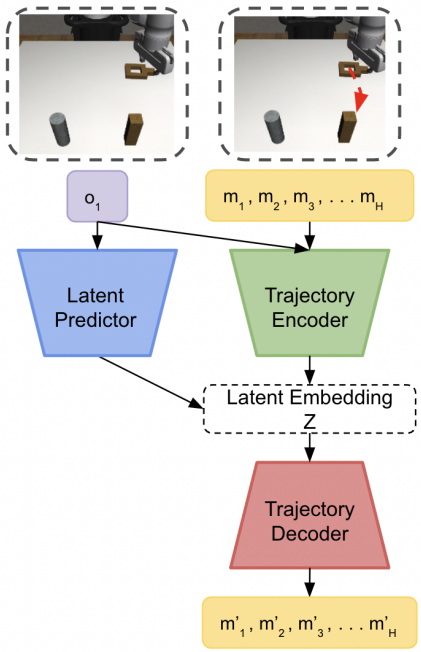


Figure 3: **Overall Architecture of OCP.**

In the model, the trajectory encoder and trajectory decoder learns how to encode/decode the object's trajectory $m = \{m_1, m_2 \dots m_H\}$, The encoder and decoder has an LSTM architecture, where it outputs the hidden state of the trajectory/outputs the reconstructed object positions given the hidden state, respectively. The loss of LSTM-VAE is sum of L2-norm between $m = \{m_1, m_2 \dots m_H\}$ and $m' = \{m'_1, m'_2 \dots m'_H\}$ and the KL-divergence between posterior distribution and prior distribution.

$$L_{LSTM} = |m - m'|^2 + KL[N(z_\mu, z_\sigma|N(0, I)]$$

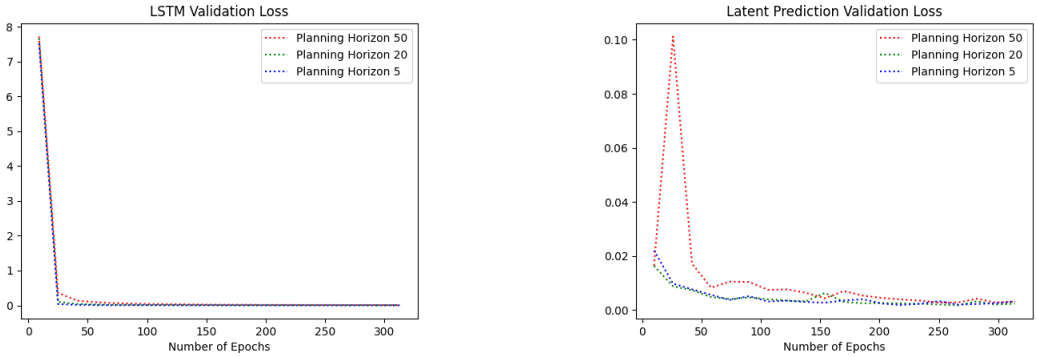
For better performance, we represent the rotation of the object with 2 axes of the rotation matrix, as proposed in Zhou et al[14]. Additionally, we adopt different weights for position and rotation, resulting in the following loss.

$$L_{LSTM} = c_{pos}|m_{pos} - m'_{pos}|^2 + c_{rot}|m_{rot} - m'_{rot}|^2 + KL[N(z_\mu, z_\sigma|N(0, I)]$$

The latent predictor tries to predict the hidden latent state of the latent embedding given the environment's image observation. It uses the L2 norm between the predicted latent value (from the latent predictor) and real latent value (from LSTM-VAE) as its loss. We also trained with KL divergence between predicted latent distribution and actual latent distribution but figured out that the performance was better with simple L2 norm. Once trained, the latent predictor combined with the trajectory decoder can plan the object's movement for the given task.

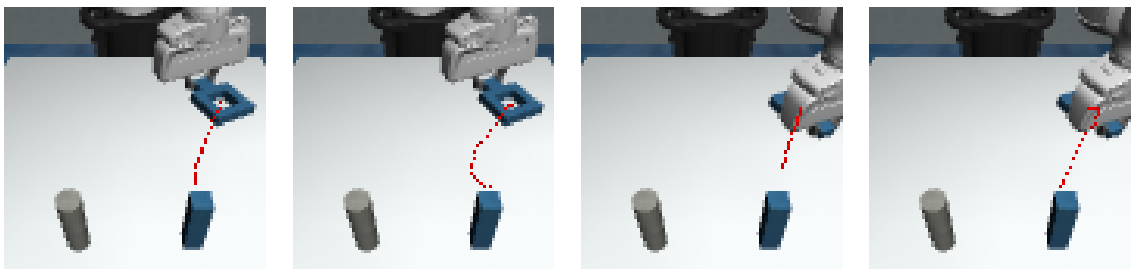
$$L_{Latent\ Predictor} = |\{z'_\mu, z'_\sigma\} - \{z_\mu, z_\sigma\}|^2$$

Results - Loss Curve. The training loss curves of our model are provided in Figure 4. Since it is hard to plan the object's long-horizon motion, we tried learning 3 models that plan the object's motion for the next 5, 20, and 50 timesteps respectively. We have found out that up to planning horizon length 50, training results are of high quality.



(a) Validation Loss of LSTM-VAE (b) Validation Loss of Latent Predictor

Figure 4: **Validation Loss of Training**

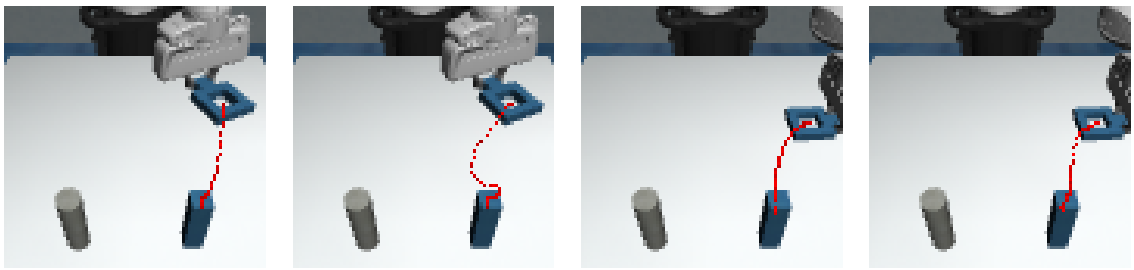


(a) Plan (b) Ground truth (c) Plan (d) Ground truth

Figure 5: **Planned and Ground Truth Object Trajectories. (Horizon: 20 timesteps)**

Results - Latent Plan Visualization. Figure 5, 6 shows some of the planned results and ground truth object trajectory with red points, with respect to different planning horizons. For brevity, we only visualized the position of the plan and ground truth. Although they are not exactly the same, we can observe that the learned model is planning the square ring's movement in a reasonable way, so that an agent can succeed in the task as long as it follows the plan accurately.

We additionally visualize the learned latent space of the OCP. Specifically, we visualize the predicted latent which describes the planned object-centric trajectory with T-SNE, along 10 demonstration trajectories. Note that the object-centric plan has a relatively shorter horizon compared to the demonstration length, as aforementioned. The results are shown in Figure 7. As expected, the latent values are temporally well aligned across different demonstration trajectories, indicating that the learned latent plan can predict meaningful signals for solving the Square task.



(a) Plan (b) Ground truth (c) Plan (d) Ground truth

Figure 6: **Planned and Ground Truth Object Trajectories. (Horizon: 50 timesteps)**

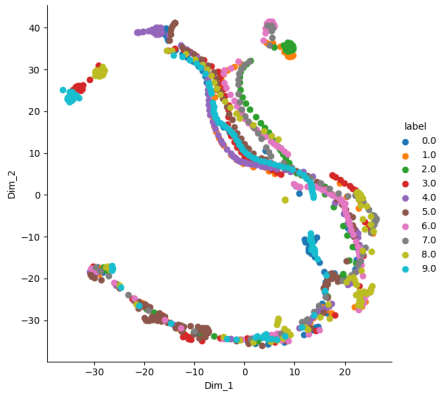


Figure 7: **Visualization of Latent Plans within 10 Trajectories**

4 Utilizing Object-Centric Planner in the Target Robot

Zero-Shot Generalization. In this section, we check the power of a learned model in the target robot Sawyer. For simplicity, we assume grasping skill to be given. Under such an assumption, once the object is grasped by the robot, simply computing pose differences between two consecutive frames in an object-centric plan can be directly used to provide control signals for the end effector.

Figure 8 shows the result of the Sawyer robot trying to insert the square ring. Even though we have only learned object-centric plan with position and we learned it from different robot demonstrations, we can see that the Sawyer robot is reasonably trying to achieve the task, without any further learning. However, as the target robot is failing by a small margin, further training of the policy is required.

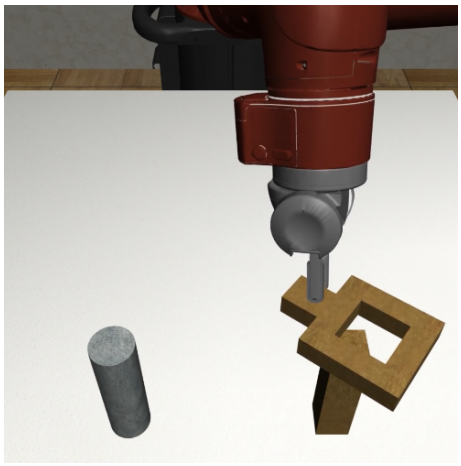


Figure 8: **Zero-Shot Transfer.**

5 Diverse Utilities of Object-Centric Trajectory Planner

In this section, we show that the learned OCP can provide rich, diverse supervision, either by providing dense rewards or hints for the actor, applicable for further learning. Specifically, we propose two ways of its usage, as follows.

Object-Centric Planner for Actor Learning. This refers to utilizing the planned object trajectory to provide hints for learning an actor. If the learned object-centric plan is ideal and

when the agent grasps the square ring, we can assume that the policy's action output, which includes the desired delta pose of the end effector, should have a linear relation to the planned delta position/pose, hence making learning easier. For this, we compute the difference between the current object pose and the planned object pose at the next time step with OCP. Then, we make the policy predict the ratio it has to multiply to the planned delta position and rotation, and the residual action in order to compensate for the inaccuracy of the planned delta object pose. The overall structure for this is in Figure 9. The action can be formulated as follows, where m' refers to the planned object pose at each timestep, x/y being predicted position/rotation weight correspondingly, and A being predicted residual action.

$$a_t = (x, y) * (m'_{t+1} - m'_t) + A_t$$

The benefit of such usage is that it can be used for behavior cloning, while prior works[4, 5] are limited to RL setup as a learned model can only be used as a reward model.

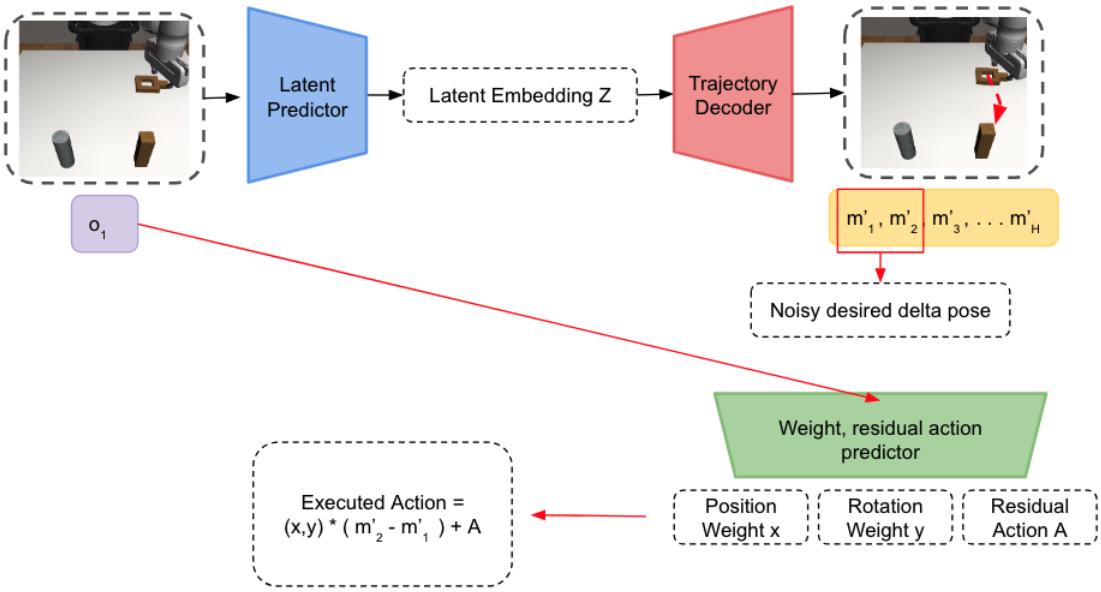


Figure 9: **Utilizing OCP for Learning an Actor.**

Object-Centric Planner as Dense Reward Generator. This refers to using planned object trajectory as a dense reward supervisor. Since we have the object's planned trajectory, we can generate dense reward by computing the difference between the object's planned trajectory and its' real trajectory. In the experiment, we either add the generated dense reward on top of the ground truth sparse reward, or only use our reward for learning the policy. When doing so, we only augment/use OCP when the robot is grasping the object, as our model may predict wrong signals when the object is not interacting with the robot arm. Additionally, we disentangle reward for position and reward for rotation, and multiply weight a and b to each of them for computing the reward. The overall structure for this is in Figure 10. The reward term is as follows, m being position and rotation in next timestep and m' being planned position and rotation at next timestep. We use 0.003 for both a and b.

$$r_{t,OCP} = a|m'_{t+1,position} - m_{t+1,position}|^2 + b|m'_{t+1,rotation} - m_{t+1,rotation}|^2$$

As the Robomimic uses only sparse reward based on task completion, we hope the provided dense reward boost Q values to propagate faster, enhancing RL algorithm's ability to learn. Both method requires the current pose of the object to manipulate, which is not a strong assumption considering

the recent advance in object pose estimation. In our setup, we get the ground truth pose of the square ring from the simulation.

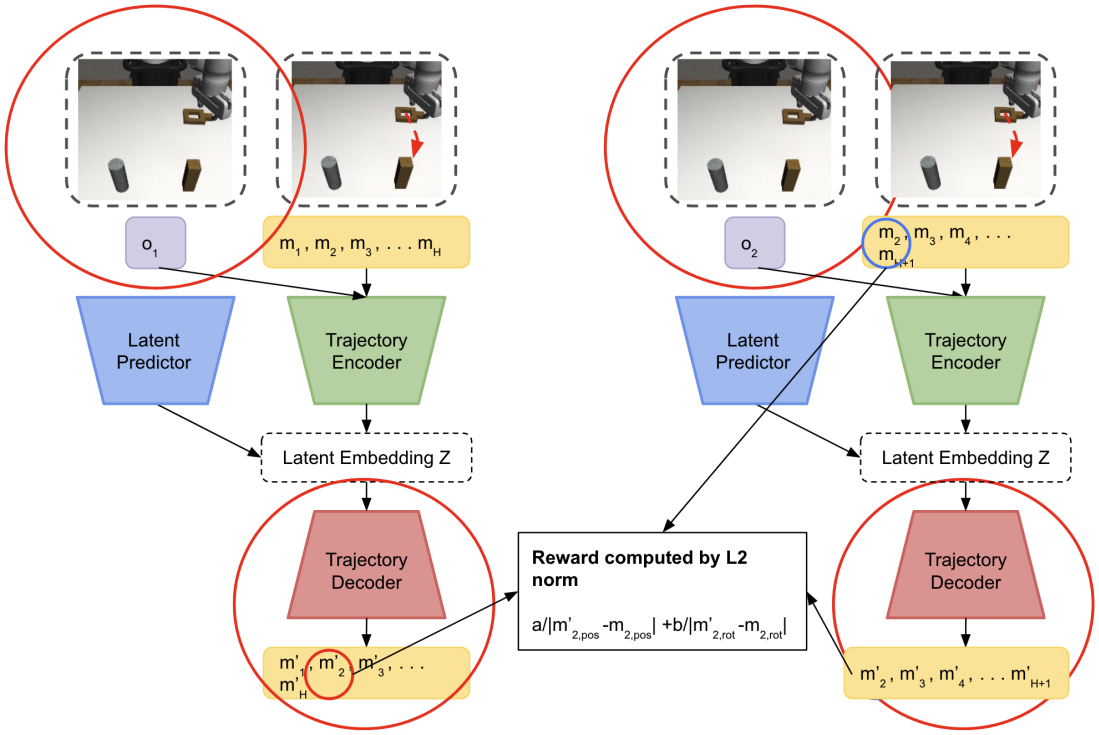


Figure 10: **Utilizing OCP as a Dense Reward Generator.**

We test the utility as hints for learning an actor with BC, BC-RNN, with gaussian-mixture-model as the action parametrization. We use our method to predict the mean of each gaussian modes. For the utility of a dense reward generator, we test our method with Batch-Constrained deep Q-learning (BCQ) [15], a commonly used offline reinforcement learning algorithm. For all experiments, we use the default hyparameters provided by Robomimic without further training. For simplicity of experiments, we show our method's benefit only on the source robot policy training with the source robot demonstration data. One can easily extend our experiment to target robot policy offline/online learning setup, which we leave for future work.

Offline Policy Learning Results on a Source Robot. The results are in Table 1, 2. Note that the original refers to training without leveraging our model, and the results are from the Robomimic paper. For Table 2, augment refers to adding OCP generated reward on top of ground truth sparse reward, and only OCP indicates only using OCP generated reward without access to ground truth sparse reward.

Compared to the original results from the Robomimic paper, leveraging our model on BC-RNN achieves similar performance. This is possible because the performance difference between state-based experiments and image observation-based experiments is not significant for BC-RNN. Additionally, we only leverage OCP for single-step action prediction. As the learned planner can plan in a temporarily abstracted manner, leveraging such temporal abstraction would be a meaningful direction to explore further. In the meanwhile, BC and BCQ achieve superior performance compared to image observation-based results, while being worse to the state-based original results. Surprisingly, only using OCP reward is much better than the ground truth reward, indicating the OCP can provide a strong learning signal. Additionally, it is even better than augmenting it on top of the ground truth reward. As we didn't tune the coefficient when augmenting OCP reward, this remains room for further improvement. Although it is not a fair comparison to compare directly

Table 1: **Offline Policy Learning Results on Source Robot. - Object-Centric Planner for Actor Learning** We present success rates (mean and standard deviation) over 3 seeds for each method.

	BC(state)	BC(image)	BC-RNN(state)	BC-RNN(image)
Original	78.7 ± 1.9	62.0 ± 4.9	84 ± 0.0	82.0 ± 0.0
Ours	-	69.3 ± 3.1	-	80.7 ± 1.2

Table 2: **Offline Policy Learning Results on Source Robot. - Object-Centric Planner as Dense Reward Generator** We present success rates (mean and standard deviation) over 3 seeds for each method. Original results do not leverage our method, using only ground truth sparse reward.

	BCQ(state)	BCQ(image, augment)	BCQ(image, only OCP)
Original	50.0 ± 4.9	41.3 ± 4.1	41.3 ± 4.1
Ours	-	44.7 ± 8.33	46.7 ± 8.33

against image observation-based experiments as our method requires object pose information on top of image observation to be trained, we conclude that utilizing our model can improve learning efficiency, even in the same agent setting. Such results support our claim that OCP can serve as a rich supervisor. As aforementioned, experiments on the target robot on offline/online policy learning setup would be a promising direction to explore, to further strengthen our claim.

6 Conclusion

In this work, we have learned an object-centric planner that plans the object's movement to achieve the task. This was done by utilizing the demonstration data from a source robot. With a learned object-centric planner, we observed a certain level of zero-shot generalization to the target robot. Moreover, we have shown the model's versatility as a rich supervisor for policy learning, showing a performance improvement on policy learning for the same robot when used as a dense reward generator. In the future, we plan to learn the policy offline/online on the target robot with the methods aforementioned in Section 5. Limitations of our work are that we assumed that the object trajectory for achieving the task could be shared among various agents, and considered only prehensile manipulation. To deal with these limitations, learning a multimodal object-centric planner would be a plausible method to take, so that the target robot can use the appropriate plan. For non-prehensile manipulation, our method would be easily extended as long as we trim the frames when the object is not in contact with the robot and equip a robot with reaching skill.

References

[1] Ashish Kumar, Zipeng Fu, Deepak Pathak, and Jitendra Malik. Rma: Rapid motor adaptation for legged robots. *arXiv preprint arXiv:2107.04034*, 2021.

[2] Grace Zhang, Ayush Jain, Injune Hwang, Shao-Hua Sun, and Joseph J Lim. Efficient multi-task reinforcement learning via selective behavior sharing. *arXiv preprint arXiv:2302.00671*, 2023.

[3] Dibya Ghosh, Avi Singh, Aravind Rajeswaran, Vikash Kumar, and Sergey Levine. Divide-and-conquer reinforcement learning. *arXiv preprint arXiv:1711.09874*, 2017.

[4] Kevin Zakka, Andy Zeng, Pete Florence, Jonathan Tompson, Jeannette Bohg, and Debidatta Dwibedi. Xirl: Cross-embodiment inverse reinforcement learning. In *Conference on Robot Learning*, pages 537–546. PMLR, 2022.

[5] Sateesh Kumar, Jonathan Zamora, Nicklas Hansen, Rishabh Jangir, and Xiaolong Wang. Graph inverse reinforcement learning from diverse videos. *arXiv preprint arXiv:2207.14299*, 2022.

[6] Karl Schmeckpeper, Oleh Rybkin, Kostas Daniilidis, Sergey Levine, and Chelsea Finn. Reinforcement learning with videos: Combining offline observations with interaction. *arXiv preprint arXiv:2011.06507*, 2020.

[7] J.D. Morrow and P.K. Khosla. Manipulation task primitives for composing robot skills. In *Proceedings of International Conference on Robotics and Automation*, volume 4, pages 3354–3359 vol.4, 1997.

[8] Yu Xiang, Tanner Schmidt, Venkatraman Narayanan, and Dieter Fox. Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes. *arXiv preprint arXiv:1711.00199*, 2017.

[9] Kiru Park, Timothy Patten, and Markus Vincze. Pix2pose: Pixel-wise coordinate regression of objects for 6d pose estimation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7668–7677, 2019.

[10] Yisheng He, Yao Wang, Haoqiang Fan, Jian Sun, and Qifeng Chen. Fs6d: Few-shot 6d pose estimation of novel objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6814–6824, 2022.

[11] Ajay Mandlekar, Danfei Xu, Josiah Wong, Soroush Nasiriany, Chen Wang, Rohun Kulkarni, Li Fei-Fei, Silvio Savarese, Yuke Zhu, and Roberto Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. *arXiv preprint arXiv:2108.03298*, 2021.

[12] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[13] Diederik P Kingma, Max Welling, et al. An introduction to variational autoencoders. *Foundations and Trends® in Machine Learning*, 12(4):307–392, 2019.

[14] Yi Zhou, Connelly Barnes, Jingwan Lu, Jimei Yang, and Hao Li. On the continuity of rotation representations in neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5745–5753, 2019.

[15] Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In *International conference on machine learning*, pages 2052–2062. PMLR, 2019.