Problem Space Transformations for Generalisation in Behavioural Cloning

Kiran Doshi[1] **Marco Bagatella**[1, 2] **Stelian Coros**[1]

[1] ETH Zürich [2] MPI for Intelligent Systems, Tübingen
kiran.doshi@inf.ethz.ch

Abstract: The combination of behavioural cloning and neural networks has driven significant progress in robotic manipulation. As these algorithms may require a large number of demonstrations for each task of interest, they remain fundamentally inefficient in complex scenarios. This issue is aggravated when the system is treated as a black-box, ignoring its physical properties. This work characterises widespread properties of robotic manipulation, such as pose equivariance and locality. We empirically demonstrate that transformations arising from each of these properties allow neural policies trained with behavioural cloning to better generalise to out-of-distribution problem instances.

Keywords: behavioural/behavioral cloning, out-of-distribution generalisation

1 Introduction

The behavioural cloning (BC) paradigm has been the foundation of recent advances in robotic manipulation [1, 2]. BC is particularly promising for robot manipulation, as humans are very proficient in general manipulation, and can quickly learn to collect demonstrations when given a well-designed interface [3]. An important benefit of using this data to train a robot policy is that it can be collected on the real system, thus avoiding the sim-to-real gap. However, as a supervised learning method, BC requires the collected data to cover the workspace with relatively high density [4, 5, 6]. Neural networks trained with BC, and more generally functions estimated through supervised learning, hardly generalise outside the support of the training data, i.e. "out-of-distribution" (OOD) [7, 8]. Policy prediction for OOD states can be arbitrary, which poses a safety risk. Avoiding OOD states by providing sufficient data coverage can quickly become infeasible. This is particularly aggravating for robotic manipulation, as collection of human demonstrations remains time intensive and thus expensive [2].

This work highlights and leverages practical assumptions on object-centric manipulation tasks, and explores a family of problem space transformations that enable OOD generalisation with respect to the original problem space. We observe that these transformations are a crucial design component for learning-based control of manipulators, and enable policies learned through simple BC to perform well on OOD states. We present three main contributions: (i) we determine properties underlying practical manipulation problems; (ii) we describe several transformations of the problem space that embed these properties; (iii) we provide experimental results demonstrating that the choice of problem space transformation drastically impacts the ability of OOD generalisation for three robotic manipulation tasks.

2 Preliminaries

Behavioural Cloning We assume that the data is collected in a finite-horizon Markov Decision Process (MDP) modelled as tuple $\mathcal{M} = (\mathcal{X}, \mathcal{A}, P, \mathcal{R}, \mu_0, H)$, where $\mathcal{X}$ is the state space, $\mathcal{A}$ is the

Workshop on Safe and Robust Robot Learning for Operation in the Real World (SAFE-ROL) at CoRL 2024.

action space, $\mathcal{R} : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$ is the reward function, $P : \mathcal{X} \times \mathcal{A} \to \Delta(\mathcal{X})$ is the dynamics transition probability function, $\mu_0 \in \Delta(\mathcal{X})$ is the initial state distribution and H is the horizon. BC learns a parameterised policy $\pi_\theta : \mathcal{X} \to \mathcal{A}$ from a dataset of K task rollouts $\mathcal{D} = \{\tau_1, ..., \tau_K\}$ with $\tau_k = \{(x_0, a_0), ..., (x_H, a_H)\}$, $x_0 \sim \mu_0$, $x_{t+1} \sim P(x_t, a_t)$ and $a_t \sim \pi_d(s_t)$. In this case π_d represents the demonstrator's policy. The policy π_θ is trained by regressing actions with a loss function $\mathcal{L} = \sum_{(x,a)\in\mathcal{D}} D(a, \pi_\theta(x))$, where D is an appropriate distance metric. This constitutes a *proxy* objective, as the actual goal is to maximise the policy's returns: $\mathbb{E}_{\mu_0,\pi_\theta,P} \sum_{t=0}^{H-1} R(x_t, a_t)$.

Robotic Manipulation and Problem Space In this work, we consider robotic manipulation tasks. We define the space of state-action tuples as the problem space $\mathcal{P} = \mathcal{X} \times \mathcal{A}$. As multiple MDPs can model the same environment, the problem space is in general chosen by the designer of the system. A common choice of state and action space by practitioners is as follows [9]. The state space will include proprioceptive information $\mathbf{x}_r$, e.g. in the form of joint positions or the end-effector (EE) pose. Furthermore, the poses of the entities in the scene would be included, resulting in $\mathbf{x} = [\mathbf{x}_r, (\mathbf{x}_{o,i})_1^{N_0}]$, where N_O is the number of entities. The action space is usually a set point for the low level robot control, either at joint or at EE level $\mathbf{a} = [\mathbf{a}_r]$. As forward and inverse kinematics (calculation of EE pose from joint position values and its inverse) are usually accessible, we assume that both action and proprioception are expressed as EE poses without loss of generality. We also assume that the action is given as the next target EE position, though it is also common to include it as an offset to the current EE position or as a velocity. We leave out the state and change of the gripper in state and action space respectively for the sake of brevity.

Out of Distribution Generalisation and BC Out-of-distribution (OOD) generalisation is the desirable capability of a model to return reasonable predictions for unseen data points. We provide a practical, more specific definition in the context of this work. As the learned model π_θ operates over states, we introduce a state occupancy $\Omega \in \Delta(\mathcal{S})$ such that samples in the dataset $\mathcal{D}$ can be considered to be drawn independently and identically distributed (iid) from it. For a given choice of Ω (e.g. the solution of MLE in a class of smooth densities) and a threshold ϵ, in-distribution generalisation occurs when the learned policy π_θ returns the unseen demonstrator's action for a state $\mathbf{x} \notin \mathcal{D}$, but with $\Omega(\mathbf{x}) \geq \epsilon$. We can thus introduce an in-distribution manifold $\hat{\mathcal{X}} = \{\mathbf{x} \in \mathcal{X} \mid \Omega(\mathbf{x}) \geq \epsilon\}$. Similarly, we define

Definition 1 (OOD generalisation, informal) *A policy π is capable of OOD generalisation if its error is low for arbitrary states $\mathbf{x} \notin \mathcal{D}$ such that $\Omega(\mathbf{x}) \leq \epsilon$.*

Let us consider a desired manifold of states $\mathcal{X}^\star \supset \hat{\mathcal{X}}$ including OOD data points. In general, a policy trained through BC on $\mathcal{D}$ will not generalise to $\mathcal{X}^\star$, as supervised learning assumes that the training and test data to be iid. Further assumptions on the problem space are thus needed to enable OOD generalisation (see Appendix A for a broader discussion).

3 Problem Space Transformation

In order to enable OOD generalisation over $\mathcal{X}^\star$, we propose to apply a transformation $\mathcal{T} : \mathcal{P} \to \mathcal{Q}$, and thus introduce a transformed problem space $\mathcal{Q}$ over which the policy is learned [1]. Let $\hat{\pi}_\theta$ be the minimiser of the empirical BC loss over the transformed dataset $\mathcal{T}(\mathcal{D}) = \{(\mathcal{T}(\mathbf{x}, \mathbf{a}) \mid (\mathbf{x}, \mathbf{a}) \in \mathcal{D}\}$. The goal of this transformation is to maximise data coverage over the desired manifold [2]: $\min_{\mathcal{T}} \mid \mathcal{T}(\mathcal{X}^\star) \setminus \mathcal{T}(\hat{\mathcal{X}}) \mid$, while ensuring that the BC solution can recover the demonstrator's actions from the transformed state space: $\pi_d(\mathbf{x}) \approx \mathcal{T}^{-1}(\hat{\pi}_\theta(\mathcal{T}(\mathbf{x}))) \ \forall \mathbf{x} \in \hat{\mathcal{X}}$. Intuitively, while the objective can be optimised by "removing information" (e.g. via a low-rank linear projection), the constraint ensures that any task-relevant information is retained. If the demonstrator π_d fulfils certain assumptions, then the problem space may contain irrelevant information, and the objective can be optimised.

[1] To ease notation, we will overload $\mathcal{T}$ to also operate over states and actions separately.

[2] $|\cdot|$ represents an n-dimensional volume or Lebesgue measure.

3.1 Practical Assumptions for Robotic Manipulation

This works leverages two assumptions. First, a well-known property of many manipulation demonstrations is equivariance to transformations in $\mathrm{SE}(n)$ (where n is equal to 2 or 3, depending on the problem dimension) with respect to (w.r.t.) to a fixed world frame $\mathcal{W}$. For example, in picking up an object with a parallel gripper, what is relevant is the relative location of the gripper to the object, not the absolute location in $\mathcal{W}$. The second assumption is that object manipulation often affects objects *locally*. As a consequence, it is often sufficient to have complete information about the surroundings of the EE. For example, the exact position and orientation of an entity are not decisive when the EE is further away from the object than a distance $\lambda \in \mathbb{R}^+$. This distance λ is task specific, and excessively low values might make the demonstrator's policy irrecoverable in the transformed problem space. Nonetheless, for most tasks, we hypothesise that an appropriate value can be determined with low effort.

3.2 Applying Assumptions in Problem Space

Transformation $\mathcal{T}_1$ The first transformation we consider encodes $\mathrm{SE}(n)$ equivariance to changes of the entities' poses w.r.t. to a fixed world frame $\mathcal{W}$, where $\mathcal{W}$ measures the state values in a Cartesian coordinate system with fixed origin, denoted as W (e.g., at the base of the robot arm). A state $\mathbf{x} \in \mathcal{X}$ can be expressed as $\mathbf{x} = [\,{}_{\mathcal{W}}\mathbf{x}_r, {}_{\mathcal{W}}\mathbf{x}_{o,1}, ..., {}_{\mathcal{W}}\mathbf{x}_{o,N_O}\,]$, where the prescript denotes the frame in which the state is measured. Actions are expressed analogously $\mathbf{a} = [{}_{\mathcal{W}}\mathbf{a}_r]$. We propose to transform $\mathcal{X}$ to a frame $\mathcal{E}$ matching the position and the orientation of the end-effector. Additionally to the change of the frame of reference, the EE pose can be removed from the state space as it has a constant (zero) value. This induces a transformed state $\mathcal{T}_1(\mathbf{x}) : [\,{}_{\mathcal{E}}\mathbf{x}_{o,1}, ..., {}_{\mathcal{E}}\mathbf{x}_{o,N_O}\,]$. Any action[3] $\mathbf{a} \in \mathcal{A}$ is also transformed accordingly: $\mathcal{T}_1(\mathbf{a}) = [{}_{\mathcal{E}}\mathbf{a}_r]$. This ensures that interactions between end-effector and entities may have the same representation, regardless of poses in the fixed coordinate frame $\mathcal{W}$. In turn, this can increase the density of the occupancy over the transformed state space, effectively enlarging the in-distribution manifold. An important aside to $\mathcal{T}_1$: For some MDP modelling choices, an additional entity corresponding to a fixed point (usually the target position) in $\mathcal{W}$ needs to be added to $\mathcal{X}$ when transforming the problem space to frame $\mathcal{E}$ such that the demonstrator's policy remains recoverable.

Transformation $\mathcal{T}_2$ The second transformation we consider encodes the assumption that manipulation largely occurs locally. Starting from the output of $\mathcal{T}_1$, we introduce a parameter $\lambda \in \mathbb{R}^+$ and project the position of each entity $\mathbf{x}_{o,i}$ with $i \in [1, \ldots, N_O]$ to the surface of the λ-ball centred in the origin:

$$\mathrm{proj}(\mathbf{x}_{o,i}) = \begin{cases} (\mathrm{pos}(\mathbf{x}_{o,i}), \mathrm{rot}(\mathbf{x}_{o,i})), & \text{if } \|\mathrm{pos}(\mathbf{x}_{o,i})\|_2 < \lambda \\ (\frac{\lambda \mathrm{pos}(\mathbf{x}_{o,i})}{\|\mathrm{pos}(\mathbf{x}_{o,i})\|_2}, \mathrm{rot}(\mathbf{x}_{o,i})), & \text{otherwise} \end{cases} \tag{1}$$

where $\mathrm{pos}(\cdot)$ and $\mathrm{rot}(\cdot)$ denote the positional and rotational parts of the object pose vectors, respectively. For $(\mathbf{x}, \mathbf{a}) \in \mathcal{X} \times \mathcal{A}$, the transformation $\mathcal{T}_2$ can thus be written as

$$\mathcal{T}_2(\mathbf{x}) = [\,\mathrm{proj}({}_{\mathcal{E}}\mathbf{x}_{o,1}), ..., \mathrm{proj}({}_{\mathcal{E}}\mathbf{x}_{o,N_O})\,] \quad \text{and} \quad \mathcal{T}_2(a) = [{}_{\mathcal{E}}\mathbf{a}_r]. \tag{2}$$

Small values of λ can greatly reduce the size of the transformed desired manifold $\mathcal{T}_2(\mathcal{X}^\star)$ (i.e., by "clipping" it), and thus maximise data coverage over it. As long as the demonstrator is concerned with local information (i.e., it is invariant to entities that are further away than λ), its policy can be recovered after the transformation. On the other hand, if λ is too small, an arbitrary policy might be irrecoverable as $\mathcal{T}_2$ effectively loses information (i.e., on the exact position of distant entities).

4 Experimental Results

In this section we want to understand how BC policies perform in-distribution and OOD, in the baseline problem space $\mathcal{P}$, and in those defined by the proposed transformations. We consider

[3] This transformation can be applied no matter if the action is supplied as a next EE position, an offset to the current position or as a velocity, though the exact transformation varies.

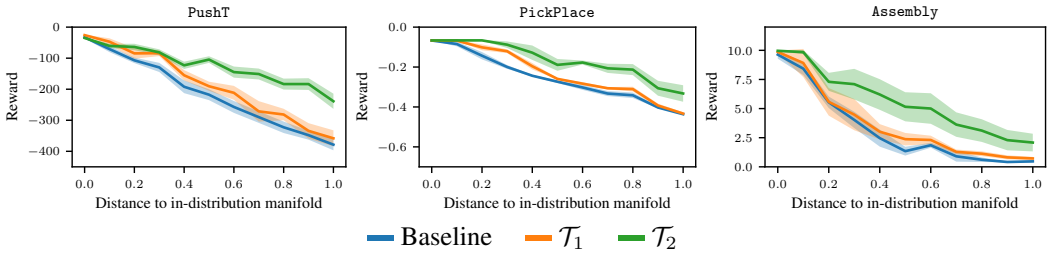

Figure 1: Comparison of BC policies trained in the original problem space $\mathcal{P}$, in $\mathcal{T}_1(\mathcal{P})$ and $\mathcal{T}_2(\mathcal{P})$ on in-distribution and OOD initial states. The x-axis shows the normalised distance to the in-distribution manifold, where the value at 0 represents the in-distribution performance. The y-axis shows the mean and standard deviation of final rewards across seeds (higher reward is better).

three environments increasing in difficulty. `PushT` is a simulated 2D environment, where the demonstrator controls a 2D point mass EE, adapted from Chi et al. [1]. The task is to move the T shape from its initial position to the goal position at the centre of the screen. In `PickPlace` and `Assembly`, the demonstrator controls the position of the EE of a 7-DOF robot arm in a 3D environment. In the former, the task is to move a block from an arbitrary initial position to a goal position fixed above the table. In the latter, the task is to pick up a tool with a handle and a loop and then to place the tool with the loop around the peg. We treat `Assembly` as a 2D problem when applying $\mathcal{T}_2$ (see Appendix C for more details).

All policies are trained by minimising MSE of policies parametrised with standard deterministic MLPs with respect to demonstrations. In-distribution performance is measured by initialising the environments as during data collection, while OOD performance is evaluated by sampling OOD initial object positions (i.e. are outside of the support of μ_0). Further details are in Appendix B.

Figure 1 reports the final rewards for all tasks and problem spaces, as a function of the distance of the initial configuration with respect to those for which data was collected. From left to right, the position of entities (e.g. T, cube or ring with handle) is initialised in bins, which expand concentrically from the in-distribution manifold. As expected, all problem spaces display performance degradation OOD. However, $\mathcal{T}_2$ performs significantly better in all tasks, which also highlights the relative importance of locality to $\mathrm{SE}(n)$ equivariance.

In Figure 2, the heat map of final rewards per initial cube position in `PickPlace` is shown, providing a detailed comparison between the three problem spaces. The policy in $\mathcal{P}$ is able to solve the task for initial states of the object which lie on the fringes of the in-distribution manifold, and not far beyond. The performance improves only somewhat for $\mathcal{T}_1$, while $\mathcal{T}_2$ is able to successfully execute tasks far outside the in-distribution manifold, where the others fail to pick up the object, let alone bring it to the target position. Heat maps for `PushT` and `Assembly` can be found in Appendix D.

5 Conclusion

Problem space transformations have the potential to enable broader OOD generalisation for BC manipulation policies by leveraging practical assumptions about manipulation problems. As our experimental validation is restricted to vanilla BC settings, one direction of future investigation is to understand if problem space transformations would also benefit more advanced BC architectures such as Diffusion Policies [1]. A second question is to understand if these transformations can be recovered automatically when an appropriate regulariser is added to the BC loss.

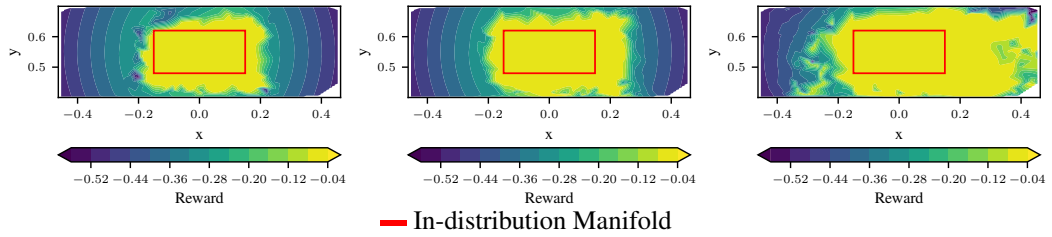

Figure 2: Comparison of Baseline $\mathcal{P}$ (left), $\mathcal{T}_1(\mathcal{P})$ (middle) and $\mathcal{T}_2(\mathcal{P})$ (right) for `PickPlace`. Plot colour indicates (interpolated) reward per initial box position. The in-distribution manifold lies within the red rectangle, the OOD manifold outside of it.

Acknowledgments

K.D. extends his gratitude to Núria Armengol, Yijiang Huang and Miguel Zamora for valuable and insightful discussions. M.B. is supported by the Max Planck ETH Center for Learning Systems.

References

[1] C. Chi, S. Feng, Y. Du, Z. Xu, E. Cousineau, B. Burchfiel, and S. Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Proceedings of Robotics: Science and Systems (RSS)*, 2023.

[2] T. Z. Zhao, J. Tompson, D. Driess, P. Florence, S. K. S. Ghasemipour, C. Finn, and A. Wahid. ALOHA unleashed: A simple recipe for robot dexterity. In *8th Annual Conference on Robot Learning*, 2024. URL https://openreview.net/forum?id=gvdXE7ikHI.

[3] T. Z. Zhao, V. Kumar, S. Levine, and C. Finn. Learning fine-grained bimanual manipulation with low-cost hardware. *arXiv preprint arXiv:2304.13705*, 2023.

[4] S. Ross, G. Gordon, and D. Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In G. Gordon, D. Dunson, and M. Dudík, editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pages 627–635, Fort Lauderdale, FL, USA, 11–13 Apr 2011. PMLR. URL https://proceedings.mlr.press/v15/ross11a.html.

[5] M. Laskey, J. Lee, R. Fox, A. Dragan, and K. Goldberg. Dart: Noise injection for robust imitation learning. In *Conference on robot learning*, pages 143–156. PMLR, 2017.

[6] S. Belkhale, Y. Cui, and D. Sadigh. Data quality in imitation learning. *Advances in Neural Information Processing Systems*, 36, 2024.

[7] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals. Understanding deep learning (still) requires rethinking generalization. *Communications of the ACM*, 64(3):107–115, 2021.

[8] J. Liu, Z. Shen, Y. He, X. Zhang, R. Xu, H. Yu, and P. Cui. Towards out-of-distribution generalization: A survey. *arXiv preprint arXiv:2108.13624*, 2021.

[9] O. Kroemer, S. Niekum, and G. Konidaris. A review of robot learning for manipulation: Challenges, representations, and algorithms. *Journal of Machine Learning Research*, 22(30): 1–82, 2021. URL http://jmlr.org/papers/v22/19-804.html.

[10] A. Simeonov, Y. Du, A. Tagliasacchi, J. B. Tenenbaum, A. Rodriguez, P. Agrawal, and V. Sitzmann. Neural descriptor fields: Se (3)-equivariant object representations for manipulation. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 6394–6400. IEEE, 2022.

[11] J. Yang, C. Deng, J. Wu, R. Antonova, L. Guibas, and J. Bohg. Equivact: Sim(3)-equivariant visuomotor policies beyond rigid object manipulation. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9249–9255, 2024. doi:10.1109/ICRA57147.2024.10611491.

[12] J. Yang, Z.-a. Cao, C. Deng, R. Antonova, S. Song, and J. Bohg. Equibot: Sim (3)-equivariant diffusion policy for generalizable and data efficient learning. *arXiv preprint arXiv:2407.01479*, 2024.

[13] T. Zhang, Y. Hu, J. You, and Y. Gao. Leveraging locality to boost sample efficiency in robotic manipulation. *arXiv preprint arXiv:2406.10615*, 2024.

[14] D. Wang, R. Walters, and R. Platt. $\mathrm{SO}(2)$-equivariant reinforcement learning, 2022. URL https://arxiv.org/abs/2203.04439.

[15] A. Tangri, O. Biza, D. Wang, D. Klee, O. Howell, and R. Platt. Equivariant offline reinforcement learning. *arXiv preprint arXiv:2406.13961*, 2024.

[16] T. Yu, D. Quillen, Z. He, R. Julian, K. Hausman, C. Finn, and S. Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on Robot Learning (CoRL)*, 2019. URL https://arxiv.org/abs/1910.10897.

A Related Works

Using Robotic Manipulation Properties in Policy Learning Past work has explored exploiting different invariances and equivariances in robot manipulation learning. In [10], learnt $\mathrm{SE}(3)$-equivariant object representations, which have an object point cloud as input, are deployed to enable a pick and place system that requires only a few demonstrations. In [11], $\mathrm{SIM}(3)$ equivariance (which additionally to $\mathrm{SE}(3)$ includes scale equivariance) is embedded in the object representation learning module (again from point clouds) as well as in the BC policy network module. The work in [12] takes a similar approach, though adding a $\mathrm{SIM}(3)$-equivariant diffusion policy as the BC policy module. Concurrent work additionally explores the usefulness of the *locality* of manipulation problems to increase sample efficiency by predicting actions as displacements to points in the scene point cloud [13]. All of the above works assume that the scene entities are sensed as point clouds. Some past work also looks at the benefits of introducing $\mathrm{SO}(2)$-equivariance to online [14] and offline RL [15] where the scene entity poses are assumed to be available.

B Implementation Details

We train our models using PyTorch. For all three tasks we use the same basic training setup. The policy is implemented as a deterministic MLP with ReLU activation functions. We use dropout and L_2 regularisaition. The models are trained using mini random batches and all data is standardised to zero mean and one standard deviation before being passed to the neural network. The loss function is the 2-norm between policy prediction and dataset sample. The Adam optimizer is used to update the weights. Training time for all tasks is in the order of minutes, while evaluation in simulation is on the order of 10 minutes.

The `PushT` MLP has 5 hidden layers with a hidden dimension of 512; dropout probability is 0.05 and the regularisation weight is 1e-5; batch size is 1024 and it is trained for 1200 epochs. The projection uses $\lambda = 150$ pixels.

The `PickPlace` MLP has 5 hidden layers with a hidden dimension of 512; dropout probability is 0.05 and the regularisation weight is 1e-5; batch size is 512 and it is trained for 600 epochs. The projection uses $\lambda = 0.1$ meters.

The `Assembly` MLP has 5 hidden layers with a hidden dimension of 512; dropout probability is 0.001 and the regularisation weight is 1e-5, batch size is 512 and it is trained for 600 epochs. The projection uses $\lambda = 0.2$ meters.

C Task Details

Push T The shape can take an arbitrary initial orientation, and the goal is a position only, thus any orientation is allowed as long as the position is correct. The shape is moved using the contact friction between point mass and shape to push the shape in the desired direction, pulling is not possible. 200 training episodes are collected by a human demonstrator. The in-distribution performance is measured with 100 initial position samples, the OOD performance with 400 initial position sampled uniformly on the respective manifolds. The in-distribution manifold and the OOD manifold are tori. The in-distribution manifold has an inner radius of 32 pixels and an outer radius of 150 pixels. In cylindrical coordinates, the in-distribution initial position is sampled with the radius $r \in [32, 150]$ (pixel) and $\theta \in [0, 2\pi]$. The OOD initial object positions are sampled with $r \in [150, 406]$ (pixels) and $\theta \in [0, 2\pi]$. The results are averaged over 10 seeds. The reward is the distance of the object position to the target.

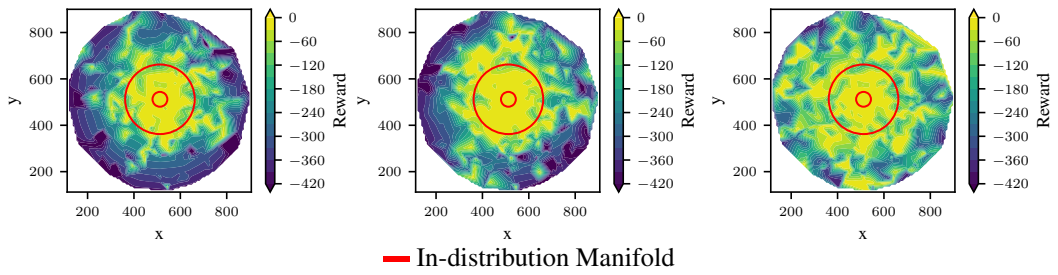

Figure 3: Comparison of Baseline $\mathcal{P}$ (left), $\mathcal{T}_1(\mathcal{P})$ (middle) and $\mathcal{T}_2(\mathcal{P})$ (right) for `PushT`. Plot colour indicates (interpolated) reward per initial box position. The in-distribution manifold lies within the two red circles, the OOD manifold outside of it.

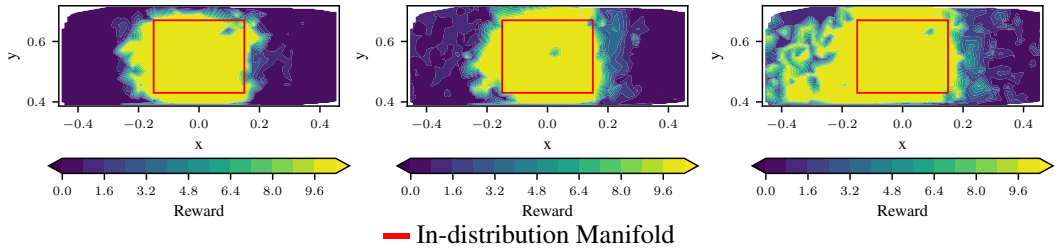

Figure 4: Comparison of Baseline $\mathcal{P}$ (left), $\mathcal{T}_1(\mathcal{P})$ (middle) and $\mathcal{T}_2(\mathcal{P})$ (right) for `Assembly`. Plot colour indicates (interpolated) reward per initial box position. The in-distribution manifold lies within the red rectangle, the OOD manifold outside of it.

Pick Place We modify the original environment by replacing a cylinder for a box. The block position is always initialised with the same orientation. To solve the task, the scripted policy moves towards the block position at an elevated height above the table top. Once above the block, the robot moves down to pick it up and then moves to the goal position. The orientation of the end-effector is fixed throughout the entire episode. 100 training episodes are collected by a scripted policy. The scripted policy can be inspected in the open implementation [16]. The in-distribution performance is measured with 100 initial position samples, the OOD performance with 450 initial positions sampled uniformly on the respective manifolds. The in-distribution manifold and the OOD manifold are rectangles. The in-distribution range for the object initial position: $x_1 \in [-0.15, 0.15], x_2 \in [0.48, 0.62]$. The OOD range for the object initial position: $x_1 \in [-0.465, 0.465], x_2 \in [0.4, 0.7]$. The results are averaged over 8 seeds. The reward is the distance of the object position to the target.

Assembly While the environment is generally a 3D environment, the vertical direction is mainly used to simply move up or down to pick and place the handle and loop object. All of the data which is characteristic to solving the task - moving to the object and after gripping it moving to the loop - happen in the plane. Therefore we treat the problem as a 2D problem for the transformation $\mathcal{T}_2$ and apply the projection in the dimensions which form the plane parallel to the table and not the perpendicular dimension. 200 training episodes are collected by a scripted policy. The scripted policy can be inspected in the open implementation [16]. The in-distribution performance is measured with 100 initial position samples, the OOD performance with 450 initial positions sampled uniformly on the respective manifolds. The in-distribution manifold and the OOD manifold are rectangles. The in-distribution range for the object initial position: $x_1 \in [-0.15, 0.15], x_2 \in [0.43, 0.67]$. The OOD range for the object initial position: $x_1 \in [-0.465, 0.465], x_2 \in [0.38, 0.72]$. The results are averaged over 8 seeds. The reward function is based on different factors, we refer to the openly available implementation of the reward function which we have not modified [16].

D Additional Results

Figure 3 shows the heat maps for `PushT` and Figure 4 shows them for `Assembly`.