PLANNING WITH GOAL-CONDITIONED POLICIES

Soroush Nasiriany, Vitchyr Pong & Sergey Levine *
Department of Electrical Engineering and Computer Science
University of California, Berkeley
Berkeley, CA 94703, USA

ABSTRACT

An open challenge in deep reinforcement learning (RL) is solving a variety of long-horizon tasks in environments using high-dimensional sensory inputs such as images. Existing model-based and model-free RL algorithms have difficulty solving such tasks, due to the accumulation of model bias and reduced capacity to solve new tasks, respectively. In this paper, we propose an algorithm that combines the advantages of flexible model-based planners with the asymptotic performance of goal-conditioned model-free policies. Our planner decomposes long-horizon tasks into shorter ones by optimizing for intermediate "sub-goals" that our goal-conditioned policy can easily reach. However, to avoid planning directly in image space, where the set of valid sub-goals is unknown, we first learn a latent embedding, where we can optimize over the set of known, valid latent states. We demonstrate that on long-horizon, simulated robotic tasks, our method outperforms prior work, including pure goal-conditioned methods and pure model-based methods.

.

1 INTRODUCTION

Deep reinforcement learning is a promising approach to allow agents to autonomously acquire complex skills, but often requires balancing sample efficiency for final performance. In particular, deep RL methods are generally divided into model-based and model-free methods, each with their own benefits. Model-free methods have been shown to achieve remarkable and even super-human performance on a variety of high dimensional tasks (Mnih et al., 2015; Silver et al., 2016; Lillicrap et al., 2016). However, these methods often require large amounts of experience, making them difficult to apply to real-world tasks where sample-efficiency is of great concern, such as robotics. On the other hand, planning-based methods learn extremely quickly by using a rich source of supervision to train forward or inverse models (Deisenroth & Rasmussen, 2011; Lenz et al., 2015; Agrawal et al., 2016; Nagabandi et al., 2018) which can be used across various tasks, but are often outperformed by model-free methods (Pong et al., 2018; Haarnoja et al., 2018). Can we develop algorithms that have both the sample-efficiency of planning-based methods and asymptotic performance of model-free methods?

One promising avenue to combine the advantages of these methods is through temporal difference models (TDMs) (Pong et al., 2018). Temporal difference models (TDMs) are a class of finite-horizon goal-conditioned value functions trained to predict how close a policy will get to a goal given a finite time budget. For short horizons, TDMs can be used to plan intermediate "goal states" analogous to planning with an inverse model. For long horizons, TDMs correspond to conventional model-free goal-conditioned policies. By choosing intermediate horizons, TDMs gain the benefits of both planing-based and model-free approaches, and have been shown to outperform both on a variety of robot locomotion and manipulation tasks.

In contrast to planning methods that use forward dynamics models, which optimize over actions, TDMs require optimizing over the state space to choose intermediate sub-goals. While such an optimization is straightforward in simple state spaces, this optimization presents a particular challenge

*snasiriany@berkeley.edu

when the set of valid states is an unknown subset of $\mathbb{R}^n$. For example, consider a locomotion task where the state consists of a robot's XY-location. If one uses a standard optimization technique such as the cross entropy method (De Boer et al., 2005), the outputted plan may require the robot to move inside a wall. More generally, one needs to know the set of valid states to prevent the optimizer from outputting invalid states. However, we would like to deploy these methods in unknown environments where the set of valid states cannot be known ahead of time. How then can we constrain our optimizer to an unknown state space?

We propose to address these issues by learning a transformation of the state space that is amenable to planning. This transformation should have two properties. First, the transformed state space should be dense, so that standard optimization techniques such as gradient descent or the cross entropy method are computationally feasible. Second, it should have low dimensionality, to reduce the computational burden. We achieve these desiderata by using a generative latent model to transform the original state space into a low-dimensional, dense state space.

Our contributions are as follows: First, we demonstrate that in environments with simple state spaces, we can use a TDM to automatically generate multiple sub-goals to decompose a long-horizon problem into multiple, short-horizon problems that can be solved by a goal-conditioned policy. While this potential approach is discussed in the original TDM paper, we provide a concrete instantiation and show that this results in a much more sample-efficient algorithm, as compared to pure model-free methods and vanilla TDMs. Second, we propose a new method for extending TDMs to state spaces that reside in an unknown manifold of a higher dimension, where a standard TDM method may be computationally feasible. Third, we demonstrate that this new method, which we call projected temporal difference model planning (PTP), outperforms existing approaches to solve image-based environments, suggesting that PTP is a promising approach for combining the advantages of model-based and model-free methods for solving long-horizon MDPs with high-dimensional state spaces.

2 Background

We consider a finite-horizon Markov decision process (MDP) defined by a tuple $(\mathcal{S}, \mathcal{A}, p, \mathbb{R}, T_{\max}, \rho_0)$, where $\mathcal{S}$ is the set of states, $\mathcal{A}$ is the set of actions, $p(\mathbf{s}_{t+1} \mid \mathbf{s}_t, \mathbf{a}_t)$ is the (possibly unknown) dynamics function, $\mathbb{R}$ is the time-varying reward function, $T_{\max}$ is the maximum horizon, and ρ_0 is the initial state distribution. The objective is to obtain a policy $\pi(\mathbf{a}_t \mid \mathbf{s}_t)$ to maximize the expected sum of rewards $\mathbb{E}[\sum_{t=0}^{T_{\max}} R(\mathbf{s}_t, t)]$, where states are sampled according to $\mathbf{s}_0 \sim \rho_0$, $\mathbf{a}_t \sim \pi(\mathbf{a}_t \mid \mathbf{s}_t)$, and $\mathbf{s}_{t+1} \sim p(\mathbf{s}_{t+1} \mid \mathbf{s}_t, \mathbf{a}_t)$. We consider the special case of goal-conditioned MDPs (Schaul et al., 2015), where the state includes a goal vector and the policy is rewarded based on how close it gets to this goal at the final time step.

2.1 Temporal Difference Models

Since the reward function depends only on the final state reached, a powerful framework for optimizing these MDPs is to use temporal difference models (TDMs) (Pong et al., 2018). TDMs are goal-conditioned value functions (Schaul et al., 2015), defined by[1]

$$V^\pi(\mathbf{s}, \mathbf{g}, t) = \mathbb{E}\left[\sum_{t'=T_{\max}-t}^{T_{\max}} R_{\text{TDM}}(\mathbf{s}_{t'}, \mathbf{g}, t') \mid \mathbf{s}_{T_{\max}-t} = \mathbf{s}, \pi \text{ is conditioned on } \mathbf{g}\right] \tag{1}$$

where the TDM reward R_{TDM} is given by

$$R_{\text{TDM}}(\mathbf{s}, \mathbf{g}, t) = -\mathbf{1}(t = T_{\max}) d(\mathbf{s}, \mathbf{g}) \tag{2}$$

where $\mathbf{1}$ is the indicator function, and the distance function d is defined by the user. This particular choice of reward function gives a TDM the following interpretation: given a state $\mathbf{s}$, how close will the goal-conditioned policy π get to $\mathbf{g}$ after t time steps of attempting to reach $\mathbf{g}$? TDMs can thus be used as a measure of reachability by quantifying how close to another state the policy can get in t time steps. In practice, this value function cannot be computed exactly, and we rely on Q-learning (Watkins & Dayan, 1992) to approximate V.

[1] While the authors originally defined TDMs in terms of Q functions, we use the analogous V function.

3 PLANNING WITH TEMPORAL DIFFERENCE MODELS

We now describe how planning with TDMs can leverage the advantages of model-based and model-free methods. We henceforth omit the dependence of V on π to keep the notation uncluttered.

Suppose that we are solving a long-horizon problem, where the maximum horizon $T_{\max}$ is large. At the beginning of each episode, we could employ our TDM starting with the horizon $T_{\max}$, and decrement the horizon at each step. Since $T_{\max}$ is large, the TDM may suffer from high uncertainty for the initial stages of the episode, leading to suboptimal policies. This is due to the fact that the reward signal from the last timestep must propagate to the beginning of the episode, leading to the accumulation of model uncertainty.

Alternatively, we could leverage the fact that the TDM provides a measure of "reachability" to segment long-horizon tasks into a sequence of short-horizon tasks that each encapsulate significantly less uncertainty. Suppose that we can express $T_{\max} = KT$, where T represents the horizon for each short-horizon problem. Our goal is to effectively solve for $K-1$ intermediate sub-goals $\mathbf{s}^*_T, \ldots, \mathbf{s}^*_{(K-1)T}$ that serve as way-points from the current state to the goal. The sub-goals are optimized such that the sum of the reachability measures between every consecutive pair of sub-goals is maximized:

$$\mathbf{s}^*_T, \ldots, \mathbf{s}^*_{(K-1)T} = \underset{\mathbf{s}_{2T}, \ldots, \mathbf{s}_{(K-1)T}}{\arg\max} V(\mathbf{s}, \mathbf{s}_T, T) + \sum_{k=1}^{K-2} V(\mathbf{s}_{kT}, \mathbf{s}_{(k+1)T}, T) + V(\mathbf{s}_{(K-1)T}, \mathbf{g}, T) \tag{3}$$

In order optimize the sub-goals, we employ the cross entropy method (CEM). Our objective is no longer to reach $\mathbf{g}$ in $T_{\max}$ timesteps, but rather to reach $\mathbf{s}^*_T$ in T timesteps. After T steps, the objective is to reach $\mathbf{s}^*_{2T}$ and so forth. We can interpret this method as a hybrid model-based and model-free method, where we use a model-free policy at the low level to optimize for short-horizon problems and a model-based (planning) approach at the high level to optimize for long-horizon problems. While this planning-based use of TDMs was described in (Pong et al., 2018), the experiments focused on the model-free component of TDMs, i.e. directly using the goal-conditioned policy trained with Equation 2; we label this method "model-free TDMs" (MF TDM). In this paper we provide a concrete instantiation of the planning version of TDMs, which we call "Planning TDMs."

4 PLANNING IN HIGH-DIMENSIONAL SPACES

Ideally, we want to solve MDPs with high-dimensional states, such as image-based environments. One possible approach to this problem is using the raw sensory image observations as direct inputs to the policy and value function networks and basing the reward function on raw pixel distance between the current and goal image. However, this formulation may be suboptimal for TDMs and other methods, because raw pixel distance is often not a semantically meaningful metric for complex tasks. Rather, we employ variational autoencoders (VAEs) (Kingma & Welling, 2014; Rezende et al., 2014), to provide a latent space that captures the underlying characteristics of the environment, and also to leverage their role as generative models for sampling meaningful sub-goals. The input to the VAE is the raw image observation $\mathbf{o}$ from the agent, and the output $\mathbf{z}_o$ is the mean of the VAE encoding distribution $q(\mathbf{z}|\mathbf{o})$. The reward function between the current image $\mathbf{o}$ and the goal image $\mathbf{g}$ is defined as

$$R_{\text{TDM}}(\mathbf{z}_o, \mathbf{z}_g, t) = -\mathbf{1}(t = T_{\max}) d(\mathbf{z}_o, \mathbf{z}_g) \tag{4}$$

where $\mathbf{z}_o$ and $\mathbf{z}_g$ are the mean of the encoding distribution for $\mathbf{o}$ and $\mathbf{g}$, respectively. If we have access to the true states $\mathbf{s}$ and $\mathbf{g}$ during training time, we can alternatively define our reward as Equation 2. In either case, the policy and value function networks directly operate on the latent embeddings, so we only require access to the raw image observations during testing time.

Before training our RL algorithm, we train the VAE using a dataset of images (which ideally captures the entire range of environment settings). In practice, while the standard VAE formulation

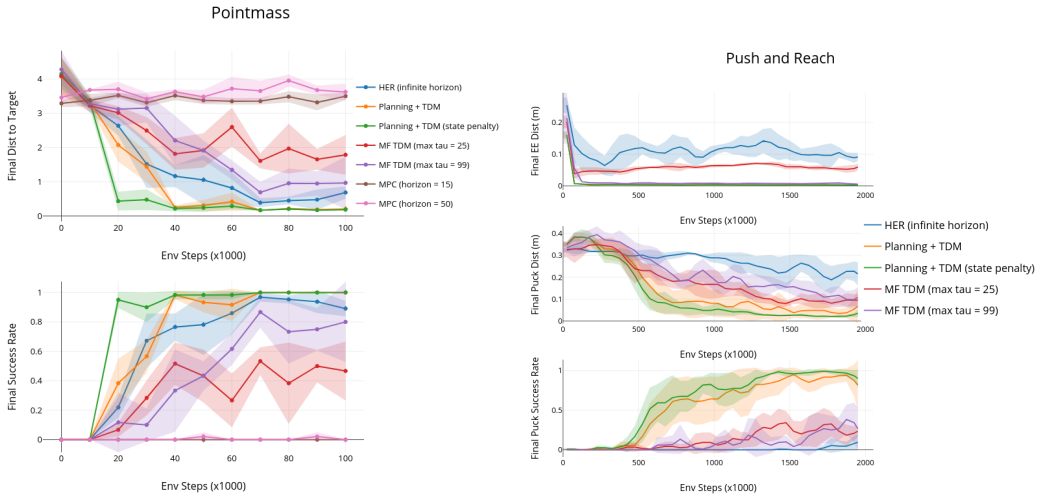

Figure 1: Full-state results

works sufficiently well in reconstructing images and producing meaningful samples, the distances between images in latent space (ie. the distances between the latent embeddings of images) are not semantically meaningful, which may lead to suboptimal RL performance. To address the issue, we append an additional linear latent dynamics penalty to the standard VAE objective. The linear dynamics penalty is defined as

$$d(f(\mathbf{z}_t, \mathbf{a}_t), \mathbf{z}_{t+1}) \tag{5}$$

where $\mathbf{z}_t$ and $\mathbf{z}_{t+1}$ are the VAE embeddings for consecutive images $\mathbf{o}_t$ and $\mathbf{o}_{t+1}$ in the dataset, and f is a linear model. In practice, this penalty encourages points that are close in state space to also be close in latent space, inducing a more semantically meaningful latent space.

Once we train the VAE, we fix it as the basis for our representation and proceed to training the TDM as usual. With this low-dimensional representation, it is feasible to run MF TDMs; however this representation may still not be suitable for Planning TDMs. The linear dynamics penalty that we add to the VAE objective imposes strong constraints, causing the induced prior of the model to no longer be Gaussian. In many instances, there are "holes" in the middle of the latent space that do not correspond to any images, thus making the space non-dense. This presents a challenge for planning, as the optimizer may select "invalid" sub-goals that the TDM was never trained on.

To address this issue, we employ a projection network that projects any point in the latent space back onto the manifold of valid latents. We call this method projected temporal difference model planning (PTP). Our projection network takes as input a latent $\mathbf{z}$, decodes it to an image $\hat{\mathbf{o}}$. and re-encodes the decoded image back to a latent $\hat{\mathbf{z}}$. Effectively, the decoded image is somewhat close to the underlying training distribution for the VAE, so once it is re-encoded the resulting latent is in the manifold of valid latents. At each iteration of CEM in which we select latents to optimize for, we transform those latents using our projection network before evaluating them.

5 Experiments

We are interested in solving long-horizon tasks, with sample efficiency and asymptotic performance in mind. We evaluate our method and competing baselines on (1) a 2D pointmass task with a u-shaped wall and (2) a Sawyer push and reach task in the MuJoCo simulator (Todorov et al., 2012). Further details for these tasks are provided in the appendix section. For each task, we train our goal-conditioned policy for any general configuration of the task (ie. the agent starts in any setting and the goal is any setting). To examine the performance of our method, we test in a restricted setting in which we pick specific reset and goal settings that correspond to long-horizon tasks. We report results in the testing phase. We compare our method to model-free TDMs, for short and long-term

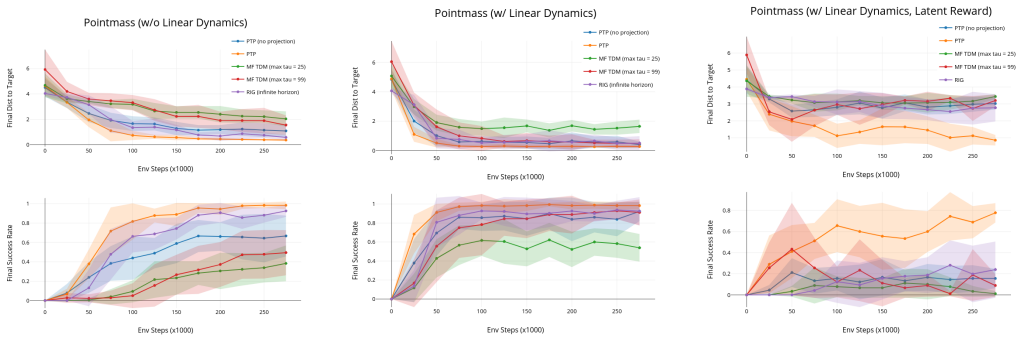

Figure 2: Image-based results

temporal capacities. In addition, we compare to an infinite horizon model-free formulation, and a random shooting model predictive control (MPC) method as the pure model-based baseline.

5.1 Full state results

For pointmass, our method learns to solve the task significantly faster than competing baselines. We note that MPC is unable to solve this simple task, so we exclude it from further consideration in other experiments. For push and reach, our method is the only method that successfully solves the task. When optimizing for sub-goals, we do not distinguish between valid and invalid sub-goals, so we also consider a variant in which we discard any invalid sub-goals (this is the "state penalty" variant). Even in the absence of the state penalty our method does well — however the state space in these tasks is relatively dense, and our method may exhibit sub-optimal performance for tasks with highly non-dense state spaces.

5.2 Image-based results

For image-based tasks, we evaluate exclusively the pointmass environment. We consider both state-based rewards (see Equation 2) and latent-based rewards (see Equation 4). For state-based rewards, our method provides marginal performance improvements over competing baselines. In addition, the linear dynamics penalty for the VAE helps boost RL performance significantly. We note that the projection contributes a significant improvement in performance for our method, even when the VAE is trained without linear dynamics. For latent-based rewards, PTP outperforms other baselines by a significant margin.

6 Discussion and Future Work

In this work, we examine the benefits of planning with goal-conditioned policies. For tasks with simple, low-dimensional state spaces, we show that planning with temporal difference models allows us to solve long-horizon tasks with greater sample efficiency and asymptotic performance than other goal-conditioned methods. We then extend this idea to tasks with high dimensional spaces where the underlying space of valid states is unknown. Using the projection network, we establish that we can outperform existing baselines on an image-based task.

An interesting avenue for future work is to scale this method for more complex image-based tasks, such as robotic tasks. Other alternatives to the projection network may be considered, such as flow models that map noise to the induced prior in latent space. This could establish a more robust planning space that can allow us to scale up easily to more challenging domains.

References

Pulkit Agrawal, Ashvin V Nair, Pieter Abbeel, Jitendra Malik, and Sergey Levine. Learning to poke by poking: Experiential learning of intuitive physics. In D. D. Lee, M. Sugiyama, U. V.

Luxburg, I. Guyon, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 29*, pp. 5074–5082. Curran Associates, Inc., 2016.

Pieter-Tjerk De Boer, Dirk P Kroese, Shie Mannor, and Reuven Y Rubinstein. A tutorial on the cross-entropy method. *Annals of operations research*, 134(1):19–67, 2005.

Marc Peter Deisenroth and Carl Edward Rasmussen. PILCO: A model-based and data-efficient approach to policy search. In *International Conference on Machine Learning (ICML)*, pp. 465–472, 2011. URL http://mlg.eng.cam.ac.uk/pub/pdf/DeiRas11.pdf.

Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, and Sergey Levine. Soft actor-critic algorithms and applications. *CoRR*, abs/1812.05905, 2018.

Diederik P Kingma and Max Welling. Auto-Encoding Variational Bayes. In *International Conference on Learning Representations (ICLR)*, 2014. URL https://arxiv.org/pdf/1312.6114.pdf.

Ian Lenz, Ross Knepper, and Ashutosh Saxena. DeepMPC: Learning Deep Latent Features for Model Predictive Control. In *Robotics: Science and Systems (RSS)*, 2015.

Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2016. ISBN 0-7803-3213-X. doi: 10.1613/jair.301. URL https://arxiv.org/pdf/1509.02971.pdf.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, and Others. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

Anusha Nagabandi, Gregory Kahn, Ronald S Fearing, and Sergey Levine. Neural Network Dynamics for Model-Based Deep Reinforcement Learning with Model-Free Fine-Tuning. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2018.

Vitchyr Pong, Shixiang Gu, Murtaza Dalal, and Sergey Levine. Temporal Difference Models: Model-Free Deep RL For Model-Based Control. In *International Conference on Learning Representations (ICLR)*, 2018. URL https://arxiv.org/pdf/1802.09081.pdf.

Danilo J Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic Backpropagation and Approximate Inference in Deep Generative Models. In *International Conference on Machine Learning (ICML)*, 2014. URL https://arxiv.org/pdf/1401.4082.pdf.

Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. Universal Value Function Approximators. In *International Conference on Machine Learning (ICML)*, pp. 1312–1320, 2015. ISBN 9781510810587. URL http://proceedings.mlr.press/v37/schaul15.pdfhttp://jmlr.org/proceedings/papers/v37/schaul15.html.

David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, jan 2016. ISSN 0028-0836. doi: 10.1038/nature16961.

Emanuel Todorov, Tom Erez, and Yuval Tassa. MuJoCo: A physics engine for model-based control. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5026–5033, 2012. ISBN 9781467317375. doi: 10.1109/IROS.2012.6386109. URL https://homes.cs.washington.edu/{~}todorov/papers/TodorovIROS12.pdf.

Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

A TASK DESCRIPTIONS

A.1 POINTMASS

The environment consists of a point navigating around a u-shaped wall in an 8×8 unit space. The agent is operated via 2D position control, and at each time step is restricted to moving up to 0.25 units in any direction. During the training phase, we consider any valid starting and goal position for the agent. During the testing phase, we restrict the agent to start inside the u-wall and the goal to be on the other side of the u-wall. This constitutes a long-horizon problem, as the agent must learn to navigate around the wall to reach the goal.

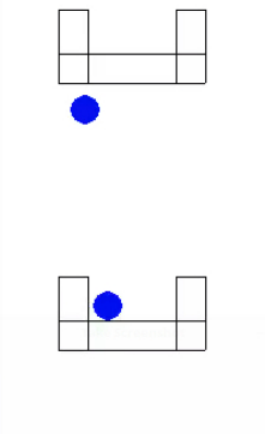

Figure 3: Pointmass

A.2 PUSH AND REACH

The environment consists of a Sawyer robot and a puck on a table. The goal is for the robot to push a puck to the goal position and also move its end effector to a separate goal position. The robot is operated via 2D position control. The robot and puck both operate in a 20 cm $\times$ 40 cm rectangular space. During the training phase, the goal for the puck and end effector is any arbitrary position within this space. At resets however, the end effector and puck are placed within close proximity of one another, to alleviate the exploration problem of the robot learning to reach the puck and push it to receive reward. During the testing phase, this restriction is removed. The puck and end effector are reset within opposite corners of the rectangular space. The goal puck position is the initial position of the end effector at reset time, and the goal end effector position is the initial position of the puck at reset time.

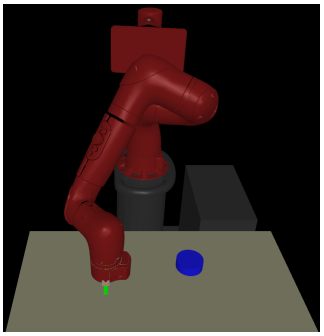

Figure 4: Push and Reach