Simulator Predictive Control: Using Learned Task Representations and MPC for Zero-Shot Generalization and Sequencing

Zhanpeng He*
University of Southern California
zhanpenh@usc.edu

Ryan Julian*
University of Southern California
rjulian@usc.edu

Eric Heiden
University of Southern California
heiden@usc.edu

Hejia Zhang
University of Southern California
hejiazha@usc.edu

Joseph J. Lim
University of Southern California
limjj@usc.edu

Stefan Schaal
University of Southern California
sschaal@usc.edu

Gaurav Sukhatme
University of Southern California
gaurav@usc.edu

Karol Hausman
Google AI
karolhausman@google.com

Abstract

Simulation-to-real transfer is an important strategy for making reinforcement learning practical with real robots. Successful sim-to-real transfer systems have difficulty producing policies which generalize across tasks, despite training for thousands of hours equivalent real robot time. To address this challenge, we present a novel approach to efficiently performing new robotic tasks directly on a real robot, based on model-predictive control (MPC) and learned task representations. Rather than end-to-end learning policies for single tasks in simulation and attempting to transfer them, we use simulation to learn (1) an embedding function encoding a latent representation of task components (skills), and (2) a single latent-conditioned policy for all tasks, and directly transfer the frozen policy to the real robot. We then use MPC to perform new tasks without any exploration in the real environment, by choosing latent skill vectors to feed to the frozen policy, controlling the real system in skill latent space. Our MPC model is the frozen skill latent-conditioned policy, executed in the simulation environment, run in parallel with the real robot. In short, we show how to reuse the simulation from the pre-training step of sim-to-real methods as a tool for foresight, allowing the sim-to-real policy adapt to unseen tasks. We discuss the background and principles of our method, detail its practical implementation, and evaluate its performance by using our method to train a real Sawyer Robot to achieve motion tasks such as drawing and block pushing.

*Equal contribution

32nd Conference on Neural Information Processing Systems (NIPS 2018), Montréal, Canada.

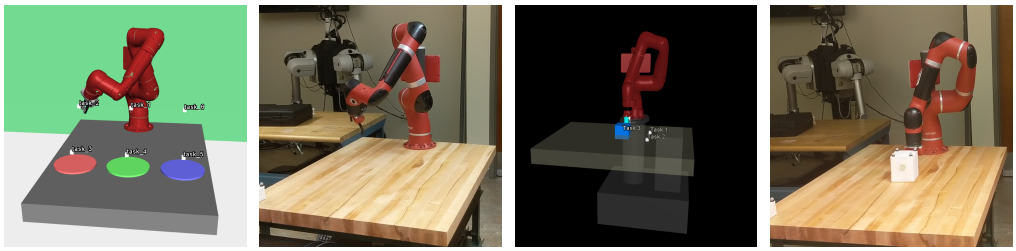

Figure 1: The Sawyer robot performing the reaching and task in simulation and real world

1 INTRODUCTION

Reinforcement learning algorithms have been proven to be effective to learn complex skills in simulation environments in [1], [2] and [3]. However, practical robotic reinforcement learning for complex motion skills remains a challenging and unsolved problem, due to the high number of samples needed to train most algorithms and the expense of obtaining those samples from real robots. Most existing approaches to robotic reinforcement learning either fail to generalize between different tasks and among variations of single tasks, or only generalize by requiring collecting impractical amounts of real robot experience. With recent advancements in robotic simulation, and the widespread availability of large computational resources, a popular family of methods seeking to address this challenge has emerged, known as "sim-to-real" methods. These methods seek to offload most training time from real robots to offline simulations, which are trivially parallelizable and much cheaper to operate. Our method combines this "sim-to-real" schema with representation learning and model-predictive control (MPC) to make transfer more robust, and to significantly decrease the number of simulation samples needed to train policies which achieve families of related tasks.

The key insight behind our method is that the simulation used in the pre-training step of a simulation-to-real method can also be used online as a tool for foresight. It allows us to predict the behavior of a known policy on an unseen task. When combined with a latent-conditioned policy, where the latent actuates variations of useful policy behavior (i.e. skills), this simulation-as-foresight tool allows our method to use what the robot has already learned to do (e.g. the pre-trained policy) to bootstrap online policies for tasks it has never seen before. That is, given a latent space of useful skills, and a simulation which predicts the rewards for those skills on a new task, we can reduce the adaptation problem to intelligently choosing a sequence of latent skills which maximize rewards for the new task.

2 BACKGROUND

Most simulation-to-real approaches so far have focused on addressing the "reality gap" problem. The reality gap problem is the domain shift performance loss induced by differences in dynamics and perception between the simulation (policy training) and real (policy execution) environments. Training a policy only in a flawed simulation generally yields control behavior which is not adaptable to even small variations in the environment dynamics. Furthermore, simulating the physics behind many practical robotic problems (e.g. sliding friction and contact forces) is an open problem in applied mathematics, meaning it is not possible to solve a completely accurate simulation for many important robotic tasks [4]. Rather than attempt to create an explicit alignment between simulation and real [5], or randomize our simulation training to a sufficient degree to learn a policy which generalizes to nearby dynamics [6], our method seeks to learn a sufficient policy in simulation, and adapt it quickly to the real world online during real robot execution.

Our proposed approach is based on four key components: reinforcement learning with policy gradients (RL) [7], variational inference [8], model-predictive control (MPC), and physics simulation. We use variational inference to learn a low-dimensional latent space of skills which are useful for tasks, and RL to simultaneously learn single policy which is conditioned on these latent skills. The precise long-horizon behavior of the policy for a given latent skill is difficult to predict, so we use MPC and an online simulation to evaluate latent skill plans in simulation before executing them on the real robot.

3 RELATED WORK

Learning skill representations to aid in generalization has been proposed in works old and new. Previous works proposed frameworks such as Associative Skill Memories [9] and probabilistic movement primitives [10] to acquire a set of reusable skills. Our approach is built upon [11], which learns a embedding space of skills with reinforcement learning and variational inference, and [12] which shows that these learned skills are transferable and composable on real robots. While [12] noted that predicting the behavior of latent skills is an obstacle to using this method, our approach addresses the problem by using model-predictive control to successfully complete unseen tasks with no fine-tuning on the real robot. Exploration is a key problem in robot learning, and our method addresses this problem by drastically reducing the dimensionality of the exploration space, from actions or parameters to a low-dimensional latent space of known-useful state-action sequences. Using learned latent spaces to make exploration more tractable is also studied in [13] and [14]. Our method exploits a latent space for task-oriented exploration: it uses model-predictive control and simulation to choose latent skills which are locally-optimal for completing unseen tasks, then executes those latent skills on the real robot.

Using reinforcement learning with model-predictive control has been explored previously. Kamthe *et al.* [15] proposed using MPC to increase the data efficiency of reinforcement algorithms by training probabilistic transition models for planning. In our work, we take a different approach by exploiting our learned latent space and simulation directly to find policies for novel tasks online, rather than learning and then solving a model.

Simulation-to-real transfer learning approaches include randomizing the dynamic parameters of the simulation [6], and varying the visual appearance of the environment [16], both of which scale linearly or quadratically the amount of computation needed to learn a transfer policy. Other strategies, such as that of Barrett *et al.* [17] reuse models trained in simulation to make sim-to-real transfer more efficient, similar to our method, however this work requires an explicit pre-defined mapping between seen and unseen tasks. Saemundson *et al.* [18] use meta-learning and learned representations to generalize from pre-trained seen tasks to unseen tasks, however their approach requires that the unseen tasks be very similar to the pre-trained tasks, and is few-shot rather than zero-shot. Our method is zero-shot with respect to real environment samples, and can be used to learn unseen tasks which are significantly out-of-distribution, as well as for composing learned skills in the time domain to achieve unseen tasks which are more complex than the underlying pre-trained task set.

Our work is closely related to simultaneous work performed by Co-Reyes *et al.* [19]. Whereas our method learns an explicit skill representations using pre-chosen skills identified by a known ID, [19] learn an implicit skill representation by clustering trajectories of states and rewards in a latent space. Furthermore, we focus on MPC-based planning in the latent space to achieve robotic tasks online with a real robot, while their analysis focuses on the machine learning behind this family of methods and uses simulation experiments.

4 METHOD

4.1 Task Embedding Algorithm

In our multi-task RL setting, we pre-define a set of training tasks with IDs $\mathcal{T} = \{1, \ldots, N\}$, and accompanying, per-task reward functions $r^t(s, a)$.

In parallel with learning the joint task policy π_θ as in conventional RL, we learn an embedding function p_ϕ which encodes these tasks to a latent variable z. Note that the true task identity t is hidden from the policy behind the embedding function p_ϕ. Rather than reveal the task ID to the policy, once per rollout we feed the task ID t, encoded as a one-hot vector, through the stochastic embedding function p_ϕ to produce a latent vector z. We feed this same value of z to the policy for the entire rollout, so that all steps in a trajectory are correlated with the same value of z.

$$\mathfrak{L}(\theta, \phi, \psi) = \max_{\pi} \mathbb{E}_{\substack{\pi(a,z|s,t) \\ t \in T}} \left[\sum_{i=0}^{\infty} \gamma^i \hat{r}(s_i, a_i, z, t) \middle| s_{i+1} \right] + \alpha_1 \mathbb{E}_{t \in T} \left[\mathbb{H} \left(p_\phi(z|t) \right) \right] \tag{1}$$

where

$$\hat{r}(s_i, a_i, z, t) = r^t(s_i, a_i) + \alpha_2 \log q_\psi(z|(s_i, a_i)^H) + \alpha_3 \mathbb{H} \left(\pi_\theta(a_i|s_i, z) \right) \tag{2}$$

In order to perform variational inference on the embedding function, we learn an inference function q_ψ which, given a state-action trajectory window $(s_i, a_i)^H$ of length H, predicts the latent vector z which was fed to the task policy when it produced that trajectory. This allows us to define an augmented reward which encourages the policy to produce distinct trajectories for different latent vectors. We learn q_ψ in parallel with the policy and embedding functions, as shown in Eq. 1.

We add a policy entropy bonus $\mathbb{H}\left(\pi_\theta(a_i|s_i, z)\right)$, which ensures that the policy does not collapse to a single solution for each skill. For a detailed derivation, refer to [11].

4.2 Latent Skill Space Criterion

In order for the learned latent space to be useful for completing unseen tasks, we seek to constrain the embedding distribution to satisfy two important properties:

1. **High entropy:** Each task should induce a distribution over latent vectors which is wide as possible, corresponding to many variations of a single skill.
2. **Identifiability:** Given an arbitrary trajectory window, the inference network should be able to predict with high confidence the latent vector fed to the policy to produce that trajectory.

When applied together, these properties ensure that during training the policy is trained to encode high-reward controllers for many parameterizations of a skill (high-entropy), while simultaneously ensuring that each of these latent parameterizations corresponds to a distinct variation of that skill. This dual constraint is the key for using model predictive control or other composing methods in the latent space, as we discuss in Sec. 4.3.

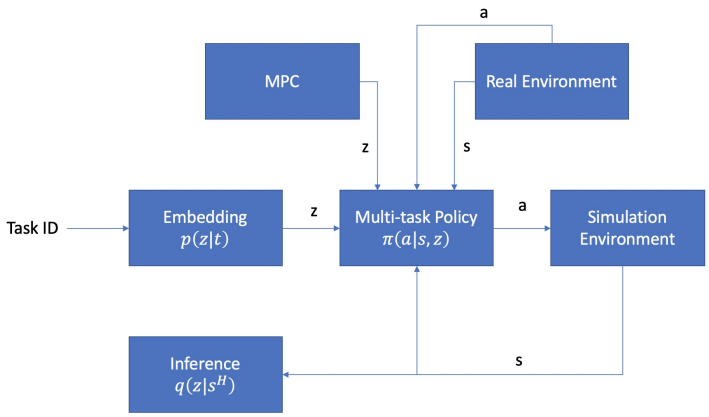

Figure 2: Skill Embedding Algorithm and MPC

We train the policy and embedding networks using Proximal Policy Optimization [20], though our method may be used by any parametric reinforcement learning algorithm. We use the MuJoCo physics engine [21] to implement our Sawyer robot simulation environments. We represent the policy, embedding, and inference functions using multivariate Gaussian distributions whose mean and diagonal covariance are parameterized by the output of a multi-layer perceptron. The policy and embedding distributions are jointly optimized by the reinforcement learning algorithm, while we train the inference distribution using supervised learning and a simple cross-entropy loss. In practice, to increase training stability, we omit actions from the trajectory snippets and use only states s_i^H for inference.

4.3 Using Model Predictive Control for Zero-Shot Adaptation

To achieve unseen tasks on a real robot with no additional training, we freeze the latent multi-task policy learned in Sec. 4.1, and use a new algorithm which we refer to as a "composer." The composer achieves unseen tasks by choosing new sequences of latent skill vectors to feed to the frozen latent-conditioned policy. Exploring in this smaller space is faster and more sample-efficient, because it encodes high-level properties of tasks and their relations. Each skill latent induces a different

pre-learned skill, and our method reduces the adaptation problem to choosing sequences of these pre-learned skills–continuously parameterized by the task embedding–to achieve new tasks.

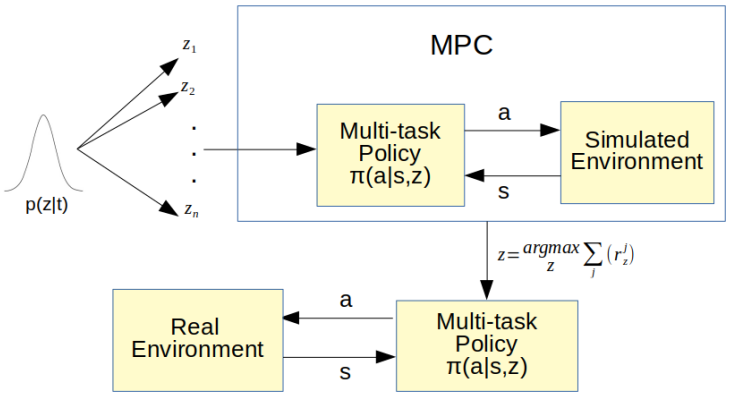

Figure 3: Using model-predictive control with embedding functions and multi-task policy

Note that we use the simulation itself to evaluate the future outcome of the next action. For each step, we set the state of the simulation environment to the observed state of the real environment. This equips our robot with with the ability to predict the behavior of different skill latents. Since our robot is trained in a simulation-to-real framework, we can reuse the simulation from the pre-training step as a tool for foresight when adapting to unseen tasks. This allow us to select a latent skill online which is locally-optimal for a task, even if that task was seen not during training. We show that this scheme allows us to perform zero-shot task generalization and composition for families of related tasks. This is in contrast to existing methods, which have mostly focused on direct alignment between simulation and real, or data augmentation to generalize the policy using brute force. Despite much work on simulation-to-real methods, neither of these approaches has demonstrated the ability to provide the adaptation ability needed for general-purpose robots in the real world. We believe our method provides a third path towards simulation-to-real adaptation that warrants exploration, as a higher-level complement to these effective-but-limited existing approaches.

We denote the new task t^{new} corresponding to reward function r^{new}, the real environment in which we attempt this task $\mathcal{R}(s'|s,a)$, and the RL discount factor γ. We use the simulation environment $\mathcal{S}(s'|s,a)$, frozen task embedding $p_\phi(z|t)$, and skill latent-conditioned policy $\pi_\theta(a|s,z)$, all trained in Sec. 4.1, to apply model-predictive control in the latent space as follows (Algorithm 1).

We first sample k candidate latents $\mathcal{Z} = \{z_1, \ldots, z_k\}$ according to $p(z) = \mathbb{E}_{t\sim p(t)} p_\phi(z|t)$, and observe the state s_{real} of real environment $\mathcal{R}$.

For each candidate latent z_i, we set the initial state of the simulation $\mathcal{S}$ to s_{real}. For a horizon of T time steps, we sample the frozen policy π_θ, conditioned on the candidate latent $a_{j\in T} \sim \pi_\theta(a_j|s_j, z_i)$, and execute the actions a_j in the simulation environment $\mathcal{S}$, yielding a total discounted reward $R_i^{\text{new}} = \sum_{j=0}^{T} \gamma^j r^{\text{new}}(s_j, a_j)$ for each candidate latent. We then choose the candidate latent acquiring the highest reward $z^* = \text{argmax}_i R_i^{\text{new}}$, and use it to condition and sample the frozen policy $a_{l\in N} \sim \pi_\theta(a_j|s_j, z^*)$ to control the real environment $\mathcal{R}$ for a horizon of $N < T$ time steps. We repeat this MPC process to choose and execute new latents in sequence, until the task has been achieved.

The choice of MPC horizon T has a significant effect on the performance of our approach. Since our latent variable encodes a skill which only partially completes the task, executing a single skill for too long unnecessarily penalizes a locally-useful skill for not being globally optimal. Hence, we set the MPC horizon T to not more than twice the number of steps that a latent is actuated in the real environment N.

5 EXPERIMENTS

We evaluate our approach by completing two sequencing tasks on a Sawyer robot: drawing a sequence of points and pushing a box along a sequential path. For each of the experiments, the robot

Algorithm 1 MPC in Skill Latent Space

Require: A latent-conditioned policy $\pi_\theta(a|s,z)$, a skill embedding distribution $p_\phi(z|t)$, a skill distribution prior $p(t)$, a simulation environment $\mathcal{S}(s'|s,a)$, a real environment $\mathcal{R}(s'|s,a)$, a new task t^{new} with associated reward function $r^{\text{new}}(s,a)$, an RL discount factor γ, an MPC horizon T, and a real environment horizon N.

while t^{new} is not complete **do**
  Sample $\mathcal{Z} = \{z_1, \ldots, z_k\} \sim p(z)$ where $p(z) = \mathbb{E}_{t\sim p(t)} p_\phi(z|t)$
  Observe s_{real} from $\mathcal{R}$
  for $z_i \in \mathcal{Z}$ **do**
    Set inital state of $\mathcal{S}$ to s_{real}
    for $j \in \{1, \ldots, T\}$ **do**
      Sample $a_j \sim \pi_\theta(a_j|s_j, z_i)$
      Execute simulation $s_{j+1} = \mathcal{S}(s_j, a_j)$
    end for
    Calculate $R_i^{\text{new}} = \sum_{j=0}^{T} \gamma^j r^{\text{new}}(s_j, a_j)$
  end for
  Choose $z^* = \text{argmax}_{z_i} R_i^{\text{new}}$
  for $l \in \{1, \ldots, N\}$ **do**
    Sample $a_l \sim \pi_\theta(a_l|s_l, z^*)$
    Execute real environment $s_{l+1} = \mathcal{R}(s_l, a_j)$
  end for
end while

must complete an overall task by sequencing skills learned during the embedding learning process. Sequencing skills poses a challenge to conventional RL algorithms due to the sparsity of rewards in sequencing tasks [22]. Because the agent only receives a reward for completing several correct complex actions in a row, exploration under these sequencing tasks is very difficult for conventional RL algorithms. By reusing the skills we have consolidated in the embedding space, we show a high-level controller can effectively compose these skills in order to achieve such difficult sequencing tasks.

5.1 Sawyer: Drawing a Sequence of Points

In this experiment, we ask the Sawyer Robot to move its end-effector to a sequence of points in 3D space. We first learn the low level policy that receives an observation with the robot's seven joint angles as well as the Cartesian position of the robot's gripper, and output incremental joint positions (up to 0.04 rads) as actions. We use the Euclidean distance between the gripper position and the current target is used as the cost function. We trained the policy and the embedding network on eight goal positions in simulation, forming a 3D rectoid enclosing the workspace. Then, we use the model-predictive control to choose a sequence latent vector which allows the robot to draw an unseen shape. For both simulation and real robot experiments, we attempted two unseen tasks: drawing a rectangle in 3D space (Figs. 4 and 6) and drawing a triangle in 3D space (Figs. 5 and 7).

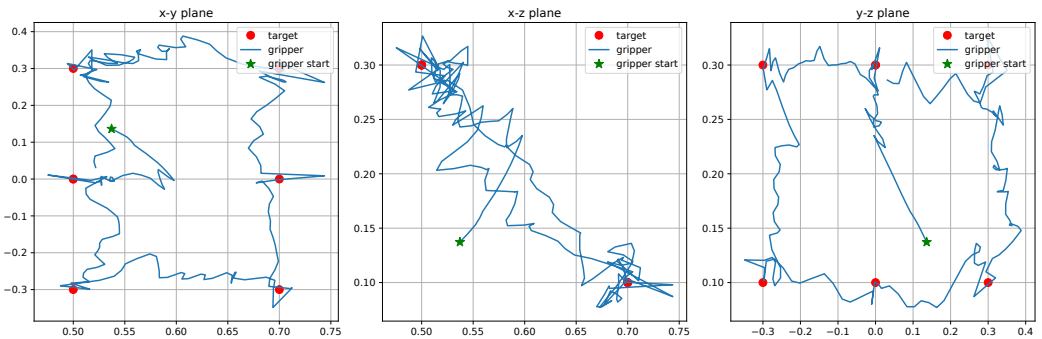

Figure 4: Gripper position plots for the unseen rectangle-drawing experiment in simulation. In this experiment, the unseen task is drawing a rectangle in 3D space.

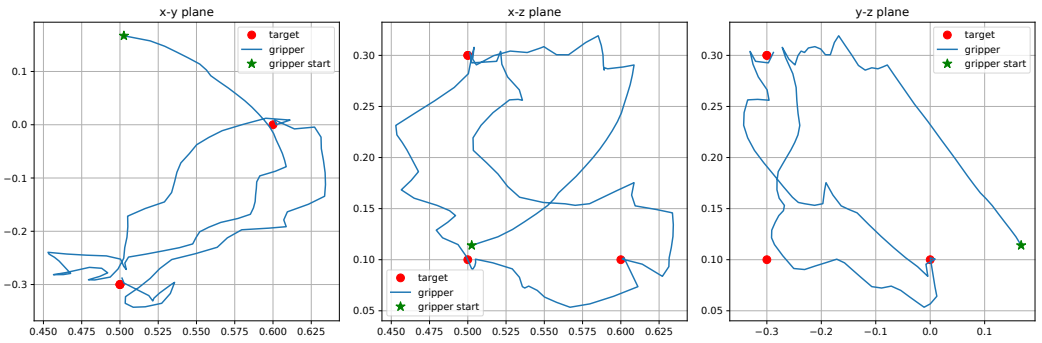

Figure 5: Gripper position plots in unseen triangle-drawing experiment in simulation. In this experiment, the unseen task is to move the gripper to draw a triangle.

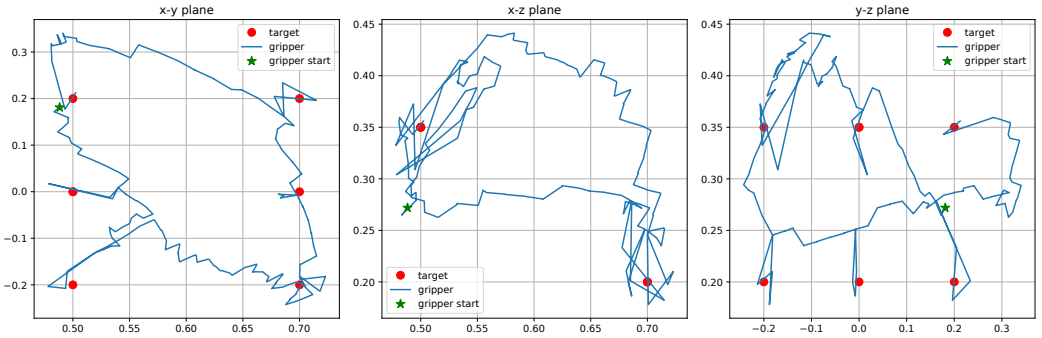

Figure 6: Gripper position plots for the rectangle-drawing experiment on the real robot. In this experiment, the unseen task is to draw a triangle.

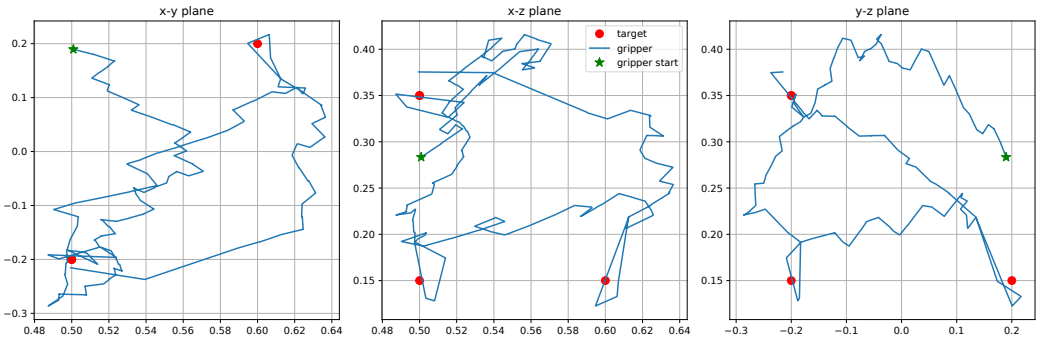

Figure 7: Gripper position plots in unseen triangle-drawing experiment on the real robot. In this experiment, the unseen task it to move the gripper to draw a triangle.

5.2 Sawyer: Pushing the Box through a Sequence of Waypoints

In this experiment, we test our approach with a task that requires contact between the Sawyer Robot and an object. We ask the robot to push a box along a sequence of points in the table plane. We choose the Euclidean distance between the position of the box and the current target position as the reward function. The policy receives a state observation with the relative position vector between the robot's gripper and the box's centroid and outputs incremental gripper movements (up to $\pm 0.03\,\mathrm{cm}$) as actions.

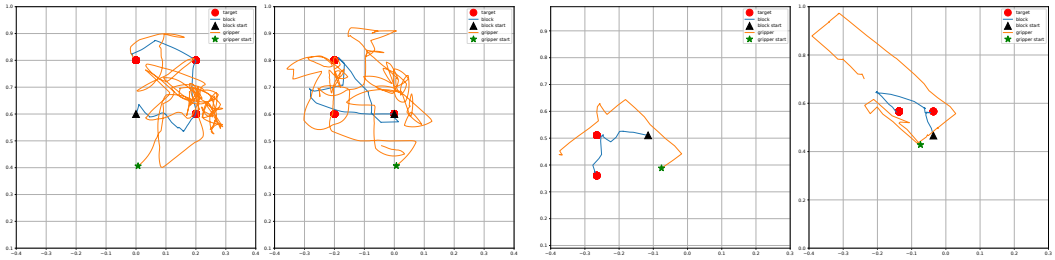

Figure 8: Plot of block positions and gripper positions in simulation (left two) and real robot experiments (right two). In experiment I, the robot pushes the box to the right, up and then left. In experiment II, the robot pushes the box to the left, then up, and then back to its starting position. In experiment III, the robot pushes the box to the left, and then down. In experiment IV, the robot push the box to the up, and then left.

We first pre-train a policy to push the box to four goal locations relative to its starting position in simulation. We trained the low-level multi-task policy with four tasks in simulation: $20\,\mathrm{cm}$ up, down, left, and right of the box starting position. We then use the model-predictive control to choose a latent vectors and feed it with the state observation to frozen multi-task policy which controls the robot.

For both simulation and real robot experiments, we use the simulation as a model of the environments. In the simulation experiments, we use model-predictive controller to push the box to three points. In the real robot experiments, we ask the Sawyer Robot to complete two unseen tasks: pushing up-then-left and pushing left-then-down.

6 RESULTS

6.1 Sawyer Drawing

In the unseen drawing experiments, we sampled $k = 15$ vectors from the skill latent distribution, and for each of them performed an MPC optimization with a horizon of $T = 4$ steps. We then execute the latent with highest reward for $N = 2$ steps on the target robot. In simulation experiments, the Sawyer Robot successfully draws a rectangle by sequencing 54 skill latents (Fig. 2) and drew by sequencing a triangle with 56 skill latents (Fig. 3). In the real robot experiments, the Sawyer Robot successfully completed the unseen rectangle-drawing task by choosing 62 skill latents (Fig. 4) in 2 minutes of wall clock time and completed the unseen triangle-drawing task by choosing 53 skill latents (Fig. 5) in less than 2 minutes of wall clock time.

6.2 Sawyer Pusher Sequencing

In the pusher sequencing experiments, we sample $k = 50$ vectors from the latent distribution. We use an MPC optimization with a simulation horizon of $T = 30$ steps, and execute each chosen latent in the environment for $N = 10$ steps. In simulation experiments, the robot completed the unseen up-left task less than 30 seconds of equivalent wall clock time and the unseen right-up-left task less than 40 seconds of equivalent wall clock time. In the real robot experiments, the robot successfully completed the unseen left-down task by choosing 3 skill latents over approximately 1 minute of wall clock time, and the unseen push up-left task by choosing 8 skill latents in about 1.5 minutes of wall clock time.

6.3 Analysis

These experiment results show that our learned skills are composable to complete the new task. In comparison with performing a search as done in [12], our approach is faster in wall clock time because we perform the model prediction in simulation instead of on the real robot. Note that our approach can utilize the continuous space of latents, whereas previous search methods only use an artificial discretization of the continuous latent space. In the unseen box-pushing real robot experiment (Fig. 7, *Right*), the Sawyer robot pushes the box towards the bottom-right right of the workspace to fix an error it made earlier in the task. This intelligent reactive behavior was never explicitly trained during the pre-training in simulation process. This shows that by sampling from our latent space,

the model-predictive controller exhibits adaptive behavior which was not seen during the training process.

7 CONCLUSION

In this work, we combine task representation learning, simulation-to-real training, and model-predictive control to efficiently adapt to unseen tasks with no additional on-robot training or exploration. Our experiments show that applying model predictive control to these learned skill representations can be a efficient method for online adaptation. The tasks the robot is able to perform using our method are more complex than the underlying pre-trained tasks used to achieve them, and the behaviors exhibited by our robot while executing unseen tasks were more adaptive than demanded by the simple reward functions used during pre-training. Our method provides a partial escape from the reality gap problem in simulation-to-real methods, by mixing simulation-based long-range foresight with locally-optimal online behavior.

For future work, we hope to apply our method as part of an algorithm for continual learning, and to explore using it to guide exploration for more general reinforcement learning algorithms such as DDPG. We are particularly excited about the potential of skill embeddings to enable for efficient online learning a policy with real robots.

References

[1] V. Mnih, K. Kavukcuoglu, D. Silver, A.A. Rusu, J. Veness, M.G. Bellemare, A. Graves, M. Riedmiller, A.K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[2] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Vedavyas Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy P. 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, 2016.

[3] Ali Yahya, Adrian Li, Mrinal Kalakrishnan, Yevgen Chebotar, and Sergey Levine. Collective robot reinforcement learning with distributed asynchronous guided policy search. *CoRR*, abs/1610.00673, 2016.

[4] Nick Jakobi, Phil Husbands, and Inman Harvey. Noise and the reality gap: The use of simulation in evolutionary robotics. In Federico Morán, Alvaro Moreno, Juan Julián Merelo, and Pablo Chacón, editors, *Advances in Artificial Life*, pages 704–720, Berlin, Heidelberg, 1995. Springer Berlin Heidelberg.

[5] Author(s A. Visser, N. Dijkshoorn, M. Van Der Veen, and R. Jurriaans. Closing the gap between simulation and reality in the sensor and motion models of an autonomous ar.drone.

[6] X.B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. *CoRR*, abs/1710.06537, 2017.

[7] John Schulman, Pieter Abbeel, and Xi Chen. Equivalence between policy gradients and soft q-learning. *CoRR*, abs/1704.06440, 2017.

[8] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. *CoRR*, abs/1312.6114, 2013.

[9] P. Pastor, M. Kalakrishnan, L. Righetti, and S. Schaal. Towards associative skill memories. In *Humanoids*, Nov 2012.

[10] E. Rueckert, J. Mundo, A. Paraschos, J. Peters, and G. Neumann. Extracting low-dimensional control variables for movement primitives. In *ICRA*, May 2015.

[11] Karol Hausman, Jost Tobias Springenberg, Ziyu Wang, Nicolas Heess, and Martin Riedmiller. Learning an embedding space for transferable robot skills. In *International Conference on Learning Representations*, 2018.

[12] Ryan C Julian, Eric Heiden, Zhanpeng He, Hejia Zhang, Stefan Schaal, Joseph Lim, Gaurav S Sukhatme, and Karol Hausman. Scaling simulation-to-real transfer by learning composable robot skills. In *International Symposium on Experimental Robotics*. Springer, 2018.

[13] S. Gu, E. Holly, T. Lillicrap, and S. Levine. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In *ICRA*. IEEE, 2017.

[14] Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is all you need: Learning skills without a reward function. February 2018.

[15] Sanket Kamthe and Marc Peter Deisenroth. Data-efficient reinforcement learning with probabilistic model predictive control. *CoRR*, abs/1706.06491, 2017.

[16] F. Sadeghi and S. Levine. CAD2RL: Real single-image flight without a single real image. In *RSS*, 2017.

[17] Samuel Barrett, Matthew E. Taylor, and Peter Stone. Transfer learning for reinforcement learning on a physical robot. In *Ninth International Conference on Autonomous Agents and Multiagent Systems - Adaptive Learning Agents Workshop (AAMAS - ALA)*, May 2010.

[18] Steindór Sæmundsson, Katja Hofmann, and Marc Peter Deisenroth. Meta reinforcement learning with latent variable gaussian processes. *CoRR*, abs/1803.07551, 2018.

[19] John D Co-Reyes, Yuxuan Liu, Abhishek Gupta, Benjamin Eysenbach, Pieter Abbeel, and Sergey Levine. Self-Consistent trajectory autoencoder: Hierarchical reinforcement learning with trajectory embeddings. June 2018.

[20] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017.

[21] MuJoCo: A physics engine for model-based control. In *IROS*, 2012.

[22] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, P. Abbeel, and W. Zaremba. Hindsight experience replay. In *NIPS*, 2017.