Crossing the Gap: A Deep Dive into Zero-Shot Sim-to-Real Transfer for Dynamics
Supplementary Material

Eugene Valassakis, Zihan Ding, and Edward Johns

1 Overview

In this document we detail supplementary information about our implementations, as well as provide the full range of results from our experiments. In section 2, we present details about our policy implementations, including neural network architectures and hyperparameters for the RL algorithms used. In section 5, we provide further details about our simulation environments, enumerating all the relevant simulator parameters, which methods randomise these parameters during training, and their sampling distributions. In section 6, we present a very detailed breakdown of all our experimental results, including trajectory plots and performance tables for each method and goal considered in our experiments.

2 Transfer Policy Implementation Details

2.1 Core architecture

Table 1 shows details for our implementation of TD3, our core RL algorithm, including details about the core neural network architecture that we use as the basis for all our methods.

Table 1: Hyperparameter details for the TD3 implementation used as our core RL algorithm.

Parameter	Value
Exploration noise decayed factor	0.9999
Exploration noise initial scale	0.3
Evaluation noise scale	0.5
Delayed update interval	3
Policy learning rate	3×10^{-4}
Q-network learning rate	3×10^{-4}
Replay buffer size	10^6
Policy size	5 layers, 512 units for each hidden layer
Q-network size	4 layers, 512 units for each hidden layer
Action range	1.
Batch size	640
Hidden activation	ReLU
Action output activation	Tanh

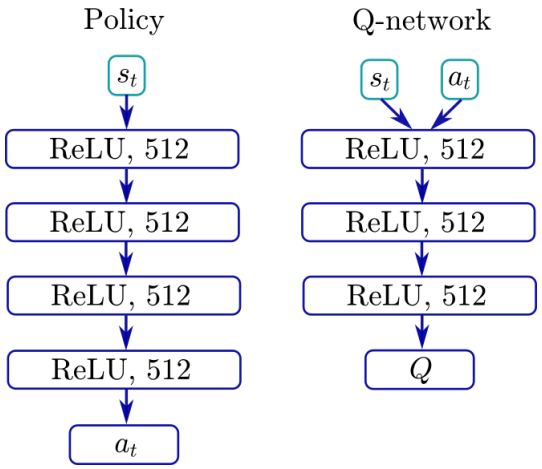

Figure 1: Illustration of our core network architecture, trained using TD3. All depicted layers are fully connected, and the numbers represent the number of units in each layer.

2.2 Adaptive Policy

The key difference between our core architecture networks and those used to train our adaptive policy is that the adaptive networks have two branches, as illustrated in Fig. 2. As in [1], one branch contains an LSTM layer which processes state-action sequences, and one branch is fully connected, similarly to our core architecture. The outputs of the LSTM branch are concatenated with the outputs of the first hidden layer of the fully connected branch before being processed by additional fully connected layers.

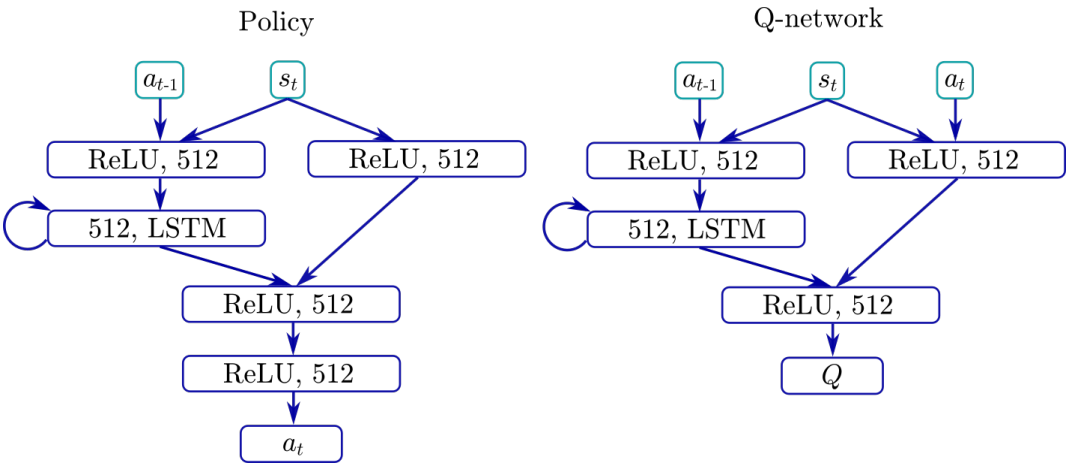

Figure 2: Illustration of the neural networks used to train our adaptive policy. A loop arrow represent the recurrence of the LSTM layer, and the numbers represent the number of units.

2.3 UPOSI

In UPOSI, there are two key models: the OSI network for predicting dynamics parameters from motion state sequences and the UP network for solving the task, conditioned on those dynamics parameters [3].

In our case, the UP network is based on our core architecture, with the dynamics parameters as an additional input, as shown in Fig. 3. Our OSI network that predicts those parameters has 3 hidden layers of 512, 256, 128 units with Tanh activations and dropout layers with a rate set to 0.1 during training. The input to our OSI network is made by stacking $m = 4$ motion states $x_{t-m} \dots x_t$, where $m = 4$ in our experiments. x_t is a motion state containing both the environment observation o_t, and the internal state s_t^{in}

of of the robot, which contains joint positions and velocities. The network architecture of OSI is shown in Fig. 4.

Our OSI network is tasked to both predict the internal dynamics parameters of the robot arm and dynamics parameters of external objects. The internal dynamics parameters are the robot link masses, the joint damping, armature and friction values, and the PID controller gains. We ignore simulator parameters such as action and observation noise ranges, timestep parameters and time delays as the inputs to the OSI network do not contain any time information, and white noise cannot be inferred from a history of states and actions. The external dynamics parameters are density, size and friction values of the puck in the pushing and sliding tasks.

As described in original paper of UPOSI [3], the overall learning process is separated into two stages: the training of UP and the training of OSI. In our experiments, we use TD3 in order to train our networks, and train the UP for a longer time than the conservative and adaptive policies (to ensure convergence). The training of OSI is further broken down into two steps: (i) the training of the OSI with UP as the exploration policy conditioned on oracle dynamics parameters, and (ii) the training of the the OSI with UP as the exploration policy, but conditioned on the OSI predicted parameters. The first step takes 100 episodes of samples for all three tasks, and the second step takes around 30000 episodes of samples. The OSI model is trained for 10 iterations on samples collected every 32 episodes, each with a different set of dynamics.

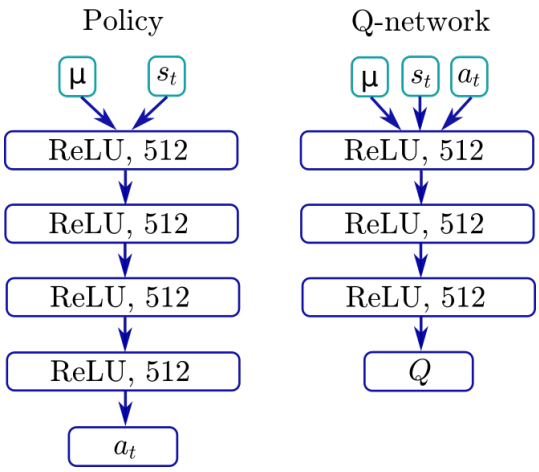

Figure 3: The policy and Q-network architectures used in training the UP with TD3. μ represents the dynamics parameters of the environment, which we get either from OSI or by querying them directly from the simulator, and the numbers represent the number of units in each layer.

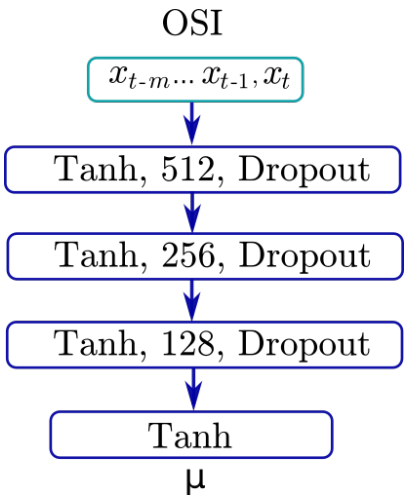

Figure 4: The network architecture of the OSI model, where x_t is the motion state formed by concatenating the policy inputs s_t with the internal state of the robot s_t^{in}. The numbers represent the nymber of units in each layer.

2.4 EPI

For training the models using EPI, we mostly follow the original paper [4], with our particular design decisions described in this section.

There are three key stages to the training procedure: (i) to collect a transition dataset with a pre-trained policy, (ii) to train the EPI policy together with the embedding network and prediction networks using this dataset, and (iii) to train the task-specific policy conditioned on the embedding vectors from the trained embedding network.

For collecting the transition dataset in (i), although in [4] the exploration policy is trained on a fixed dynamics environment, we simply used our pre-trained conservative policies for each task. Using it, we collected 100 episodes of samples, also adding $\epsilon-$greedy exploration ($\epsilon = 0.2$), where each episode has a randomised set of parameters and 500 timesteps. Similarly to the original paper, the Vine method and separation loss were applied in our data collection process [4]. In order to apply the separation loss, continuous randomisation ranges are discretised into 5 bins, while categorically distributed randomisation parameters are kept.

For state (ii), the training of the embedding and prediction networks, a total number of 1000 iterations of training are taken. For each iteration, a batch size of 20 sampled environments are picked from the transition dataset, and applied for EPI policy to roll out a total of 30 timesteps of trajectories for embedding. Each input sequence to the embedding network has a length of 10, so the 30 timesteps are split evenly into 3 pieces of input samples. An EPI reward is generated for the 10 timesteps of probing policy rollout as an additional reward to the task-oriented reward, with a multipliplicative factor of 5×10^4 for normalisation. The embedding, prediction networks and probing policy networks are all updated 10 times for each batch of samples in an iteration, and the probing policy has an inner loop of 10 updates.

For stage (iii), the embedding and probing policy networks are loaded, and after an initial 10 timesteps at the beginning of each episode the dynamics embedding z is predicted. Then the task-specific policy takes over without resetting the environment at this stage, and receives as an input the predicted dynamics embedding z. The training process of task-specific policy is done in the same way as the UP training described in the previous section.

In Table 2, we show the hyperparameters used in our implementation of PPO (used to train the probing policy), including the architecture details of the networks.

Table 2: Hyperparameter details for the PPO implementation used to train the probing policy in the EPI method.

Parameter	Value
Policy learning rate	1×10^{-4}
Value network learning rate	2×10^{-4}
Policy update epochs	10
Value network update epochs	10
ϵ	0.2
Policy size	5 layers, 512 units for each hidden layer
Value network size	4 layers, 512 units for each hidden layer
Action range	1.
Batch size	128
Hidden activation	ReLU
Action output activation	Tanh

The architectures of the embedding and prediction networks are shown in Fig. 5. The embedding network has 3 layers and the prediction network has 4 layers. Both of them have 512 nodes and ReLU activations for each hidden layer, and Tanh activations for the outputs. The state sequence $s_{t-n}...s_{t-1}, s_t$ input to the embedding network has a length of 10, and the learned embedding z has a dimension of 10.

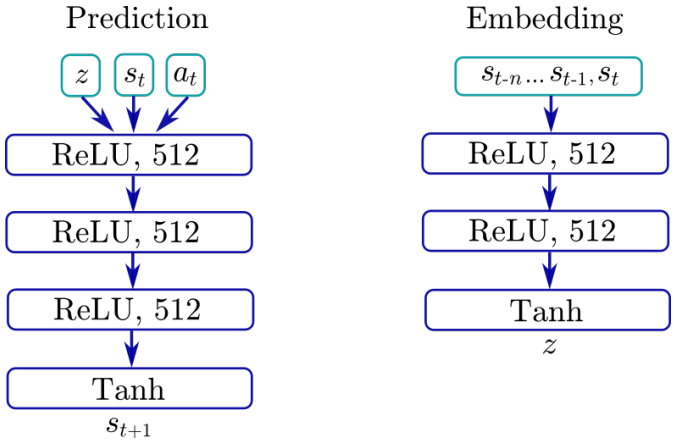

Figure 5: The network architecture of prediction and embedding models in EPI, where z is the learned embedding vector.The numbers represent the number of units in each layer.

The architecture of the probing policy is shown in Fig. 6. The stochastic

policy has 5 layers with 512 nodes and a ReLU activation for each hidden layer. The output layer for the action mean applies a Tanh activation, and the standard deviation is a single layer of learnable variables. The actions are sampled from the diagonal Gaussian distribution with those mean and standard deviation. The value network has 4 layers with 512 hidden nodes each and ReLU activations on the hidden layers.

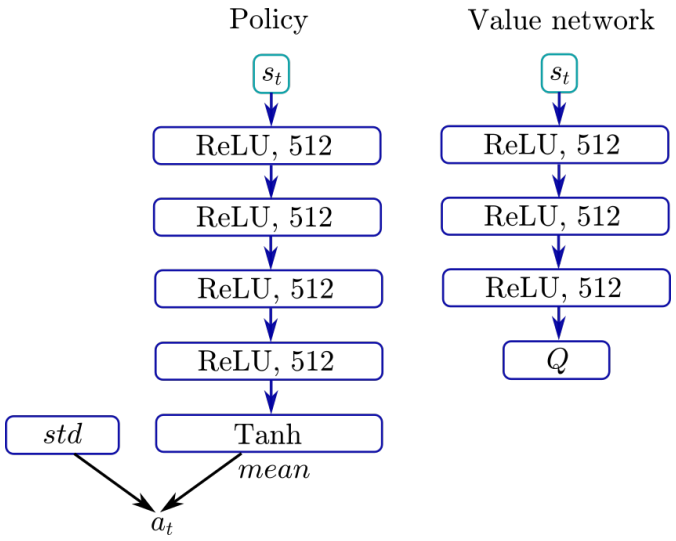

Figure 6: The network architecture of the probing policy in the EPI method. The numbers represent the number of units in each layer.

In order to train our task-specific policy, we use the TD3 algorithm as per the conservative, adaptive and UPOSI policies, but with the policy and Q-network additionally conditioned on the embedding vector z, as shown in Fig. 7.

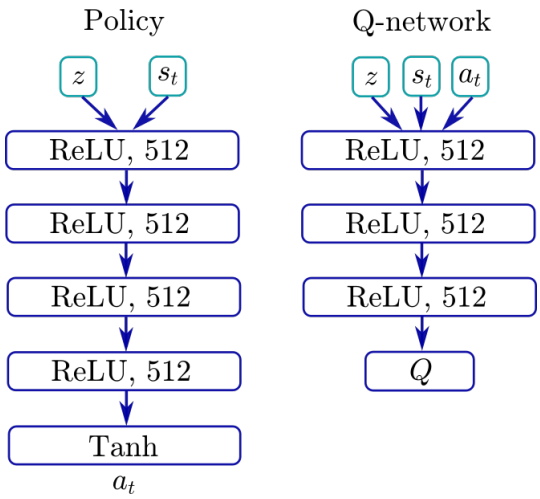

Figure 7: The network architecture of the EPI task-specific policy trained using TD3, where z is the learned embedding vector. The numbers represent the number of units in each layer.

3 Reward Functions

Our reward functions can be written as $r = r_{common} + r_{task}$. For all tasks, $r_{common} = 0.1 * \mathbb{1}_{success} - \sum_{joints} \mathbb{1}_{lim}$, with $\mathbb{1}_{success}$ the indicator function of whether the goal region is reached, and $\mathbb{1}_{lim}$ an indicator of whether a certain joint hits its limit. For reaching, the goal region has a 1 cm radius, and $r_{task} = -d_{eg} - \mathbb{1}_{table}$, with d_{eg} the end-effector-to-goal distance and $\mathbb{1}_{table}$ an indicator of whether the robot hits the table. For pushing, the goal region has a 3 cm radius, and $r_{task} = -d_{eo} - d_{og}$, with d_{eo} and d_{og} the end-effector-to-object and object-to-goal distances, respectively. For sliding, the goal region has a 2.3cm radius, and $r_{task} = -d_{og} - \mathbb{1}_{fall}$, with d_{og} the object-to-goal distance, and $\mathbb{1}_{fall}$ an indicator on whether the object has fallen off the sliding platform.

4 Differential evolution Parameters Used

For our differential evolution optimisation we used an out-of-the-box implementation [2] , with the following parameters:

Table 3: Differential evolution parameters used

Parameter	**Value**
Population size	15 x the search space dimension
Crossover Probability	0.7
Differential weight	0.5
Strategy	best1bin

5 Simulation Environments

In this section we describe in detail all the environment parameters that were considered for each of our three tasks. Because there is a significant overlap between them, we organise Tables 4, 5 and 6 as follows. Table 4 shows all the parameters relevant to the reaching simulator. Table 5, shows all the parameters that are considered in the pushing simulator but not in the reaching simulator. Together, Tables 4 and 5 list all the parameters that are considered in the pushing simulator. Table 6 lists the parameters that are relevant to the sliding simulator, but to not to the reaching or pushing simulators. When Table 6 lists a parameter that appears in either Table 4 or 5, then the distribution range of this parameter is different for sliding. Finally, the sliding simulator does not utilise any parameters that have to do with the end effector, and all the joint parameters only have dimension 2 (only the last two joints of the robot are actuated in this task). All in all, the total set of parameters relevant to sliding can be retrieved by combining Tables 4, 5 and 6, ignoring all entries that correspond to "End effector" parameters, and considering all the joint parameters as having dimension 2 (only for the last two joints).

Table 4: Table of parameters defining **reaching** environments, their sampling distributions, and the methods which utilise them.

Parameter	Symbol	Range	Distribution	Dimension	Methods	Further comments
Joint Torques	J_T	$[0., 1.5]$	Uniform	7	RFI, RFI+	One dimension per joint. At each timestep, a torque for each joint is sampled from U[-J_T, J_T], and applied to that joint.
Timestep	t_{loop}	$[0.0, 0.01]$	Uniform	1	DR	At each timestep, the policy control loop iteration time will be $0.1s + \lambda$, with $lamda \sim exp(t_{loop})$, and exp the exponential distribution.
PID timestep	t_{pid}	$[0.0, 0.04]$	Uniform	1	DR	The PID feedback loop timestep.
Action additive noise	α_{add}	$[0.01, 0.1]$	Uniform	1	DR	At each timestep, the policy action a is set to $a = a * \epsilon_m + \epsilon_{add} + \epsilon_s$, with ϵ_{add} $U[-\alpha_{add}, \alpha_{add}]$.
Action multiplicative noise	α_m	$[0.005, 0.02]$	Uniform	1	DR	At each timestep, the policy action a is set to $a = a * \epsilon_m + \epsilon_{add} + \epsilon_s$, with $\epsilon_m = 1.0 + u, u$ $U[-\alpha_m, \alpha_m]$.
Action systematic noise	α_s	$[-0.05, 0.05]$	Uniform	1	DR	At each timestep, the policy action a is set to $a = a * \epsilon_m + \epsilon_{add} + \epsilon_s$, with ϵ_s sampled once per episode from $U[-\alpha_s, \alpha_s]$.
End effector position noise	σ_{p-eef}	$[0.0005, 0.001]$	Uniform	1	DR, RFI+	At each timestep, the end effector position observation fed into the policy is perturbed by random normal noise with mean 0 and standard deviation σ_{p-eef}.
End effector velocity noise	σ_{v-eef}	$[0.0005, 0.001]$	Uniform	1	DR, RFI+	At each timestep, end effector velocity observation fed into the policy is perturbed by random normal noise with mean 0 and standard deviation σ_{v-eef}.
Link masses	M_l	$[0.98, 1.02]$	Uniform	7	DR	One per robot link. Defines multiplicative factors to the baseline robot link masses.
Joint damping	J_D	$[0.5, 2.]$	LogUniform	7	DR	One per robot joint. Defines multiplicative factors to the baseline joint damping values.
Armature	J_A	$[0.66, 1.5]$	LogUniform	7	DR	One per robot joint. Defines multiplicative factors to the baseline joint armature parameter values.
Joint Friction	J_F	$[0.66, 1.5]$	LogUniform	7	DR	One per robot joint. Defines multiplicative factors to the baseline joint friction loss parameter values.
Proportional Gains	K_p	$[0.66, 1.5]$	LogUniform	7	DR	One per robot joint. Defines multiplicative factors to the baseline proportional gains of the velocity PID controller.
Integral Gains	K_i	$[0.66, 1.5]$	LogUniform	7	DR	One per robot joint. Defines multiplicative factors to the baseline integral gains of the velocity PID controller.
Derivative Gains	K_d	$[0.66, 1.5]$	LogUniform	7	DR	One per robot joint. Defines multiplicative factors to the baseline derivative gains of the velocity PID controller.

Table 5: Table of parameters defining **pushing** environments, their sampling distributions, and the methods which utilise them.

Parameter	Symbol	Range	Distribution	Dimension	Methods	Further Comments
Object forces	F_o	[0.0, 0.0011]	Uniform	3	RFI, RFI+	At each timestep, a force is sampled from $U[-F_o, F_o]$ in each dimension, and applied to the puck.
Object torques	T_o	[0.0, 0.0005]	Uniform	3	RFI, RFI+	At each timestep, a torque is sampled from $U[-F_o, F_o]$ in each dimension, and applied to the puck.
End effector time delay	Td_{eef}	[0,1]	Categorical	1	DR, RFI+	The end-effector position the policy observes is delayed by 0 or 1 control loops, sampled at the beginning of an episode.
Object time delay	Td_o	[0,2]	Categorical	1	DR, RFI+	The puck position the policy observes is delayed by 0 or 1 control loops, sampled at the beginning of an episode.
Object position noise	σ_{p-o}	[0.0005, 0.001]	Uniform	1	DR, RFI+	At each timestep, the puck position observation fed into the policy is perturbed by random normal noise with mean 0 and standard deviation σ_{p-o}.
Object velocity noise	σ_{v-o}	[0.0005, 0.0015]	Uniform	1	DR, RFI+	At each timestep, the puck velocity observation fed into the policy is perturbed by random normal noise with mean 0 and standard deviation σ_{v-o}.
Object angle noise	σ_{a-o}	[0.005, 0.05]	Uniform	1	DR, RFI+	At each timestep, the puck angular position observation fed into the policy is perturbed by random normal noise with mean 0 and standard deviation σ_{a-o}.
Object density	ρ_o	$[100, 800]$	Uniform	1	DR	The puck's mass density is sampled every episode.
Object size	sz_o	$[0.995, 1.005]$	Uniform	1	DR	The puck's size is scaled at each episode by sz_o.
Object sliding friction	fr_s	$[0.01, 0.8]$	Uniform	1	DR	Sampled every episode.
Object torsional friction	fr_t	$[0.001, 0.3]$	Uniform	1	DR	Sampled every episode.

Table 6: Table of parameters defining **sliding** environments, their sampling distributions, and the methods which utilise them. Only parameters additional to, or differing from, Tables 4 and 5 are shown.

Parameter	Symbol	Range	Distribution	Dimension	Methods	Further Comments
Joint position noise	σ_{jp}	[0.0005, 0.005]	Uniform	2	DR, RFI+	At each timestep, the end, joint positions observations fed into the policy are perturbed by random normal noise with mean 0 and standard deviationσ_{jp}
Joint velocity noise	σ_{jv}	[0.005, 0.005]	Uniform	2	DR, RFI+	At each timestep, the end, joint velocities observations fed into the policy are perturbed by random normal noise with mean 0 and standard deviationσ_{jv}
Object density	ρ_o	[100, 900]	Uniform	1	DR	The puck's mass density is sampled every episode.
Object sliding friction	fr_s	[0.1, 0.85]	Uniform	1	DR	Sampled every episode.

6 Results

In this section we showcase our full range of experimental results, both in simulation and reality. Fig. 8 and 9 show illustrations of simulation and real world trajectories for the pushing and reaching tasks, respectively. Tables 7-12 show, for each task, the average performances over several trajectories for each method and each goal. Reaching results are shown in Table 7 (real world) and Table 8 (simulation). Pushing results are shown in Table 9 (real world) and Table 10 (simulation). Sliding results are shown Table 11 (real world) and Table 12 (simulation). We note that in the following tables RL performance is shown in terms of costs, with the rewards obtained during RL training being the negative of the costs shown.

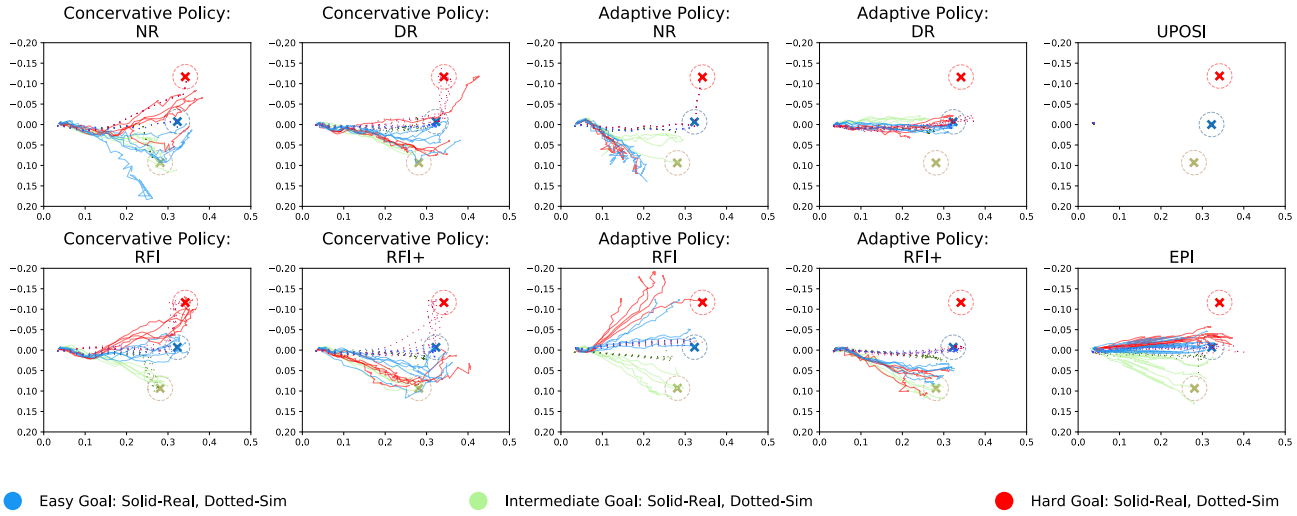

Figure 8: Trajectory of the puck over 5 trials for each method on the sliding task. The axes distances are in meters.

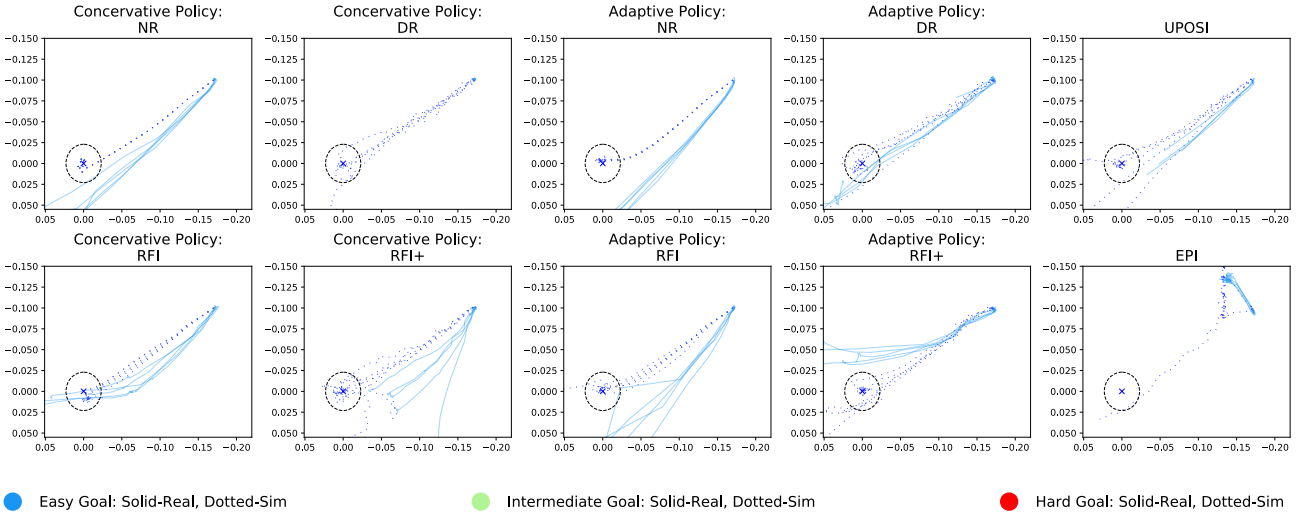

Figure 9: Trajectory of the puck over 5 trials for each method on the pushing task. The axes distances are in meters.

Table 7: Full table of results for the Reaching task over the real world experiments. For each goal, we present the mean and standard deviation of each performance score over 7 trajectories. The cost at each timestep corresponds to the distance between the end effector and the goal. The cumulative cost is the sum of the costs over all timesteps in a trajectory and the final cost corresponds to the maximum cost that occured in the last 0.5s of execution.

	NR							
Mean over 7 trajectories	Conservative Policy				Adaptive Policy			
	Easy Goal	Intermediate Goal	Hard Goal	Average	Easy Goal	Intermediate Goal	Hard Goal	Average
Mean Final Cost	0.009	0.010	0.008	0.009	0.013	0.012	0.011	0.012
Mean Final Cost Std	0.001	0.002	0.001	0.002	0.003	0.002	0.001	0.002
Mean Cumulative Cost	3.864	3.942	3.925	3.910	3.955	4.135	4.009	4.033
Mean Cumulative Cost Std	0.028	0.019	0.014	0.021	0.046	0.249	0.106	0.134
Success Rate	0.143	0.143	0.286	0.190	0.000	0.000	0.000	0.000
	DR							
	Conservative Policy				Adaptive Policy			
	Easy Goal	Intermediate Goal	Hard goal	Average	Easy Goal	Intermediate Goal	Hard Goal	Average
Mean Final Cost	0.006	0.009	0.005	0.007	0.010	0.008	0.009	0.009
Mean Final Cost Std	0.002	0.004	0.001	0.002	0.001	0.001	0.001	0.001
Mean Cumulative Cost	3.780	3.860	3.809	3.816	3.919	3.953	3.914	3.928
Mean Cumulative Cost Std	0.035	0.048	0.011	0.031	0.026	0.012	0.017	0.018
Success Rate	0.714	0.286	0.857	0.619	0.000	0.143	0.286	0.143
	RFI							
	Conservative Policy				Adaptive Policy			
	Easy Goal	Intermediate Goal	Hard Goal	Average	Easy Goal	Intermediate Goal	Hard Goal	Average
Mean Final Cost	0.005	0.005	0.003	0.004	0.008	0.007	0.009	0.008
Mean Final Cost Std	0.004	0.003	0.000	0.002	0.001	0.001	0.002	0.001
Mean Cumulative Cost	3.897	3.890	3.929	3.905	3.845	3.909	3.944	3.900
Mean Cumulative Cost Std	0.072	0.045	0.053	0.057	0.012	0.022	0.027	0.021
Success Rate	0.857	0.714	1.000	0.857	0.000	0.143	0.143	0.095
	RFI+							
	Conservative Policy				Adaptive Policy			
	Easy Goal	Intermediate Goal	Hard Goal	Average	Easy Goal	Intermediate Goal	Hard Goal	Average
Mean Final Cost	0.004	0.006	0.004	0.004	0.007	0.007	0.007	0.007
Mean Final Cost Std	0.002	0.004	0.002	0.003	0.001	0.002	0.001	0.001
Mean Cumulative Cost	3.777	3.824	3.876	3.826	3.844	3.873	3.875	3.864
Mean Cumulative Cost Std	0.024	0.033	0.022	0.026	0.024	0.018	0.009	0.017
Success Rate	0.571	0.714	0.857	0.714	0.143	0.286	0.429	0.286
	UPOSI				EPI			
	Easy Goal	Intermediate Goal	Hard Goal	Average	Easy Goal	Intermediate Goal	Hard Goal	Average
Mean Final Cost	0.029	0.023	0.026	0.026	0.010	0.010	0.014	0.012
Mean Final Cost Std	0.004	0.004	0.003	0.004	0.002	0.003	0.002	0.002
Mean Cumulative Cost	4.460	4.395	4.462	4.439	3.913	4.069	5.245	4.409
Mean Cumulative Cost Std	0.115	0.028	0.019	0.054	0.039	0.136	0.059	0.078
Success Rate	0.000	0.000	0.000	0.000	0.000	0.143	0.000	0.048

Table 8: Full table of results for the Reaching task over the simulation world experiments. For each goal, we present the mean and standard deviation of each performance score over 50 trajectories. The cost at each timestep corresponds to the distance between the end effector and the goal. The cumulative cost is the sum of the costs over all timesteps in a trajectory and the final cost corresponds to the maximum cost that occurred in the last 0.5s of execution.

	NR							
Mean over 50 trajectories	Conservative Policy				Adaptive Policy			
	Easy Goal	Intermediate Goal	Hard Goal	Average	Easy Goal	Intermediate Goal	Hard Goal	Average
Mean Final Cost	0.008	0.008	0.008	0.008	0.004	0.003	0.005	0.004
Mean Final Cost Std	0.000	0.000	0.000	0.000	0.001	0.001	0.000	0.001
Mean Cumulative Cost	4.036	4.072	4.199	4.103	3.974	3.992	4.361	4.109
Mean Cumulative Cost Std	0.000	0.000	0.000	0.000	0.033	0.015	0.003	0.017
Success Rate	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
	DR							
	Conservative Policy				Adaptive Policy			
	Easy Goal	Intermediate Goal	Hard Goal	Average	Easy Goal	Intermediate Goal	Hard Goal	Average
Mean Final Cost	0.004	0.004	0.004	0.004	0.003	0.004	0.004	0.004
Mean Final Cost Std	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0.001
Mean Cumulative Cost	4.004	4.110	4.158	4.090	4.057	4.127	4.217	4.134
Mean Cumulative Cost Std	0.186	0.181	0.167	0.178	0.169	0.150	0.200	0.173
Success Rate	0.980	1.000	1.000	0.993	1.000	1.000	1.000	1.000
	RFI							
	Conservative Policy				Adaptive Policy			
	Easy Goal	Intermediate Goal	Hard Goal	Average	Easy Goal	Intermediate Goal	Hard Goal	Average
Mean Final Cost	0.003	0.003	0.003	0.003	0.003	0.003	0.003	0.003
Mean Final Cost Std	0.000	0.000	0.001	0.001	0.001	0.001	0.001	0.001
Mean Cumulative Cost	4.033	4.034	4.150	4.073	4.048	4.079	4.140	4.088
Mean Cumulative Cost Std	0.099	0.082	0.160	0.114	0.120	0.106	0.094	0.107
Success Rate	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
	RFI+							
	Conservative Policy				Adaptive Policy			
	Easy Goal	Intermediate Goal	Hard Goal	Average	Easy Goal	Intermediate Goal	Hard Goal	Average
Mean Final Cost	0.003	0.003	0.003	0.003	0.003	0.003	0.003	0.003
Mean Final Cost Std	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0.001
Mean Cumulative Cost	3.975	4.016	4.155	4.049	3.976	4.005	4.184	4.055
Mean Cumulative Cost Std	0.063	0.088	0.095	0.082	0.104	0.105	0.128	0.113
Success Rate	1.000	1.000	0.980	0.993	1.000	1.000	1.000	1.000
	UPOSI				EPI			
	Easy Goal	Intermediate Goal	Hard Goal	Average	Easy Goal	Intermediate Goal	Hard Goal	Average
Mean Final Cost	0.027	0.023	0.031	0.027	0.005	0.006	0.005	0.005
Mean Final Cost Std	0.005	0.005	0.004	0.005	0.001	0.001	0.001	0.001
Mean Cumulative Cost	4.738	4.614	4.949	4.767	4.580	4.412	5.292	4.761
Mean Cumulative Cost Std	0.128	0.178	0.155	0.153	0.255	0.158	0.201	0.205
Success Rate	0.000	0.000	0.000	0.000	0.980	0.920	1.000	0.967

Table 9: Full table of results for the Pushing task over the real world experiments. For each goal, we present the mean and standard deviation of each performance score over 7 trajectories. The cost at each timestep corresponds to the distance between the puck.The cumulative cost is the sum of the costs over all timesteps in a trajectory and the goal, and the final cost corresponds to the maximum cost that occured in the last 0.5s of execution. The '-' indicates that the policy was too dangerous to run full experiments.

	NR							
Mean over 7 trajectories	Conservative				Adaptive			
	Easy Goal	Intermediate Goal	Hard Goal	Average	Easy Goal	Intermediate Goal	Hard Goal	Average
Mean Final Cost	0.0677	0.0287	0.0813	0.0592	0.1952	0.0571	0.2788	0.1770
Mean Final Cost Std	0.0513	0.0135	0.0520	0.0389	0.0171	0.0546	0.0079	0.0265
Mean Return	9.1713	5.8160	11.5122	8.8332	15.7753	7.2288	22.3730	15.1257
Mean Return Std	2.6381	0.6884	3.0323	2.1196	1.0398	3.2626	0.4186	1.5737
Success Rate	0.1429	0.4286	0.0000	0.1905	0.0000	0.2857	0.0000	0.0952
	DR							
	Conservative				Adaptive			
	Easy Goal	Intermediate Goal	Hard Goal	Average	Easy Goal	Intermediate Goal	Hard Goal	Average
Mean Final Cost	0.0276	0.0397	0.1502	0.0725	0.0153	0.1124	0.1224	0.0834
Mean Final Cost Std	0.0206	0.0102	0.0547	0.0238	0.0070	0.0094	0.0306	0.0157
Mean Return	6.6423	5.9991	15.0656	9.2357	6.3655	11.0234	13.4090	10.2660
Mean Return Std	1.2085	0.5692	2.5541	1.4440	0.4162	0.5469	1.7337	0.8989
Success Rate	0.7143	0.2857	0.0000	0.3333	1.0000	0.0000	0.0000	0.3333
	RFI							
	Conservative				Adaptive			
	Easy Goal	Intermediate Goal	Hard Goal	Average	Easy Goal	Intermediate Goal	Hard Goal	Average
Mean Final Cost	0.0191	0.0207	0.0271	0.0223	0.0797	0.0228	0.1463	0.0829
Mean Final Cost Std	0.0058	0.0081	0.0111	0.0084	0.0661	0.0180	0.0647	0.0496
Mean Return	5.7900	5.1578	8.3925	6.4468	9.2487	5.1180	13.8788	9.4152
Mean Return Std	0.3404	0.4038	1.1638	0.6360	3.8034	1.1098	3.7167	2.8766
Success Rate	1.0000	0.7143	0.7143	0.8095	0.2857	0.7143	0.1429	0.3810
	RFI+							
	Conservative				Adaptive			
	Easy Goal	Intermediate Goal	Hard Goal	Average	Easy Goal	Intermediate Goal	Hard Goal	Average
Mean Final Cost	0.0302	0.0280	0.1539	0.0707	0.0678	0.0286	0.2058	0.1007
Mean Final Cost Std	0.0215	0.0256	0.0492	0.0321	0.0165	0.0218	0.0272	0.0219
Mean Return	7.6319	5.6627	17.0818	10.1255	8.4912	5.5342	17.8657	10.6304
Mean Return Std	0.8576	1.4164	1.7859	1.3533	0.9906	1.2206	1.5436	1.2516
Success Rate	0.5714	0.5714	0.0000	0.3810	0.0000	0.4286	0.0000	0.1429
	UPOSI				EPI			
	Easy Goal	Intermediate Goal	Hard Goal	Average	Easy Goal	Intermediate Goal	Hard Goal	Average
Mean Final Cost	-	-	-	-	0.0211	0.0448	0.1120	0.0593
Mean Final Cost Std	-	-	-	-	0.0099	0.0278	0.0331	0.0236
Mean Return	-	-	-	-	11.5938	10.3513	15.9171	12.6207
Mean Return Std	-	-	-	-	0.5196	1.0282	1.0024	0.8501
Success Rate	-	-	-	-	0.8571	0.2857	0.0000	0.3810

Table 10: Full table of results for the Pushing task over the simulation experiments. For each goal, we present the mean and standard deviation of each performance score over 50 trajectories. The cost at each timestep corresponds to the distance between the puck and the goal.The cumulative cost is the sum of the costs over all timesteps in a trajectory, and the final cost corresponds to the maximum cost that occured in the last 0.5s of execution.

	NR							
Mean over 50 trajectories	Conservative				Adaptive			
	Easy Goal	Intermediate Goal	Hard Goal	Average	Easy Goal	Intermediate Goal	Hard Goal	Average
Mean Final Cost	0.0093	0.0145	0.0100	0.0113	0.0019	0.0801	0.0018	0.0279
Mean Final Cost Std	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
Mean Return	5.3478	5.7769	7.0283	6.0510	4.6564	8.6790	8.2663	7.2006
Mean Return Std	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
Success Rate	1.0000	1.0000	1.0000	1.0000	1.0000	0.0000	1.0000	0.6667
	DR							
	Conservative				Adaptive			
	Easy Goal	Intermediate Goal	Hard Goal	Average	Easy Goal	Intermediate Goal	Hard Goal	Average
Mean Final Cost	0.0082	0.0880	0.0332	0.0431	0.0068	0.0892	0.1025	0.0662
Mean Final Cost Std	0.0038	0.0125	0.0324	0.0162	0.0035	0.0066	0.0086	0.0062
Mean Return	5.1907	9.1214	10.1621	8.1581	6.0003	9.8028	12.2566	9.3532
Mean Return Std	0.4115	0.4075	1.1208	0.6466	0.4907	0.4313	0.5241	0.4820
Success Rate	1.0000	0.0000	0.6600	0.5533	1.0000	0.0000	0.0000	0.3333
	RFI							
	Conservative				Adaptive			
	Easy Goal	Intermediate Goal	Hard Goal	Average	Easy Goal	Intermediate Goal	Hard Goal	Average
Mean Final Cost	0.0070	0.0079	0.0059	0.0069	0.0129	0.0816	0.1034	0.0660
Mean Final Cost Std	0.0036	0.0044	0.0038	0.0039	0.0035	0.0064	0.0080	0.0060
Mean Return	5.0290	5.6128	8.3518	6.3312	5.4607	8.5295	11.4460	8.4787
Mean Return Std	0.1895	0.1543	0.1848	0.1762	0.2351	0.1980	0.2930	0.2420
Success Rate	1.0000	1.0000	1.0000	1.0000	1.0000	0.0000	0.0000	0.3333
	RFI+							
	Conservative				Adaptive			
	Easy Goal	Intermediate Goal	Hard Goal	Average	Easy Goal	Intermediate Goal	Hard Goal	Average
Mean Final Cost	0.0084	0.0679	0.0216	0.0326	0.0150	0.0744	0.1038	0.0644
Mean Final Cost Std	0.0044	0.0285	0.0179	0.0169	0.0031	0.0094	0.0036	0.0054
Mean Return	5.3845	8.6796	9.0276	7.6972	5.3248	8.7919	11.7893	8.6353
Mean Return Std	0.3296	0.8403	0.6823	0.6174	0.2635	0.3152	0.2839	0.2875
Success Rate	1.0000	0.1800	0.7400	0.6400	1.0000	0.0000	0.0000	0.3333
	UPOSI				EPI			
	Easy Goal	Intermediate Goal	Hard Goal	Average	Easy Goal	Intermediate Goal	Hard Goal	Average
Mean Final Cost	0.2840	0.2605	0.3254	0.2899	0.0188	0.0730	0.1103	0.0674
Mean Final Cost Std	0.0004	0.0004	0.0002	0.0003	0.0066	0.0251	0.0104	0.0140
Mean Return	22.7155	20.8401	26.0347	23.1967	11.9658	13.0460	16.8893	13.9670
Mean Return Std	0.0082	0.0118	0.0065	0.0088	0.6002	0.4793	0.5239	0.5344
Success Rate	0.0000	0.0000	0.0000	0.0000	0.9400	0.0800	0.0000	0.3400

Table 11: Full table of results for the Sliding task over the real world experiments. For each goal, we present the mean and standard deviation of each performance score over 7 trajectories. The cost at each timestep corresponds to the distance between the puck and the goal, if the puck is still on the sliding panel, or 0.866 if the puck has fallen off the panel. The final cost corresponds to the maximum cost that occured in the last 0.5s of execution. If the puck fell before the last 0.5s, this results in a final cost of 0.866. The cumulative cost corresponds to the sum of the costs over all the trajectory.

	NR	
Mean over 7 trajectories	Conservative	Adaptive
Mean Final Cost	0.866	0.866
Mean Final Cost Std	0.000	0.000
Mean Return	39.369	37.837
Mean Return Std	4.908	0.321
Fall Rate	1.000	1.000
Success Rate	0.000	0.000
	DR	
	Conservative	Adaptive
Mean Final Cost	0.199	0.748
Mean Final Cost Std	0.001	0.289
Mean Return	11.974	33.160
Mean Return Std	0.026	11.754
Fall Rate	0.000	0.857
Success Rate	0.000	0.000
	RFI	
	Conservative	Adaptive
Mean Final Cost	0.508	0.866
Mean Final Cost Std	0.414	0.000
Mean Return	13.666	34.293
Mean Return Std	7.126	0.779
Fall Rate	0.571	1.000
Success Rate	0.143	0.000
	RFI+	
	Conservative	Adaptive
Mean Final Cost	0.194	0.399
Mean Final Cost Std	0.276	0.405
Mean Return	10.704	14.111
Mean Return Std	6.607	7.971
Fall Rate	0.143	0.429
Success Rate	0.000	0.000
	UPOSI	EPI
Mean Final Cost	0.866	0.195
Mean Final Cost Std	0.000	0.003
Mean Return	42.456	11.740
Mean Return Std	3.906	0.115
Fall Rate	1.000	0.000
Success Rate	0.000	0.000

Table 12: Full table of results for the Sliding task over the simulation experiments. For each goal, we present the mean and standard deviation of each performance score over 7 trajectories. The cost at each timestep corresponds to the distance between the puck and the goal, if the puck is still on the sliding panel, or 0.866 if the puck has fallen off the panel. The final cost corresponds to the maximum cost that occured in the last 0.5s of execution. If the puck fell before the last 0.5s, this results in a final cost of 0.866.The cumulative cost corresponds to the sum of the costs over all the trajectory.

	NR	
Mean over 50 trajectories	Conservative	Adaptive
Mean Final Cost	0.016	0.017
Mean Final Cost Std	0.000	0.002
Mean Return	2.808	2.847
Mean Return Std	0.000	0.013
Fall Rate	0.000	0.000
Success Rate	1.000	0.940
	DR	
	Conservative	Adaptive
Mean Final Cost	0.117	0.188
Mean Final Cost Std	0.228	0.339
Mean Return	7.999	10.458
Mean Return Std	6.956	12.219
Fall Rate	0.080	0.200
Success Rate	0.400	0.680
	RFI	
	Conservative	Adaptive
Mean Final Cost	0.026	0.046
Mean Final Cost Std	0.018	0.134
Mean Return	3.232	3.695
Mean Return Std	0.452	3.615
Fall Rate	0.000	0.040
Success Rate	0.780	0.800
	RFI+	
	Conservative	Adaptive
Mean Final Cost	0.110	0.372
Mean Final Cost Std	0.244	0.411
Mean Return	5.221	15.777
Mean Return Std	6.133	14.119
Fall Rate	0.100	0.420
Success Rate	0.560	0.440
	UPOSI	EPI
Mean Final Cost	0.629	0.866
Mean Final Cost Std	0.380	0.000
Mean Return	26.073	17.574
Mean Return Std	14.182	1.576
Fall Rate	0.720	1.000
Success Rate	0.220	0.000

References

[1] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 1–8. IEEE, 2018.

[2] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. Jarrod Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. Carey, İ. Polat, Y. Feng, E. W. Moore, J. Vand erPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and S. . . Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.

[3] W. Yu, J. Tan, C. K. Liu, and G. Turk. Preparing for the unknown: Learning a universal policy with online system identification. *arXiv preprint arXiv:1702.02453*, 2017.

[4] W. Zhou, L. Pinto, and A. Gupta. Environment probing interaction policies. *arXiv preprint arXiv:1907.11740*, 2019.