1 Introduction

The paper proposes to bridge the "reality gap" by randomizing the dynamics of the simulator during training. The method is demonstrated on an object-pushing task using a robotic arm.

2 Object pushing task description

Their experiments are conducted on a puck pushing task using a 7-DOF Fetch robotic arm. The goal g for each episode specifies a random target position on the table that the puck should be moved to. The reward is binary with $r_t = 0$ if the puck is within a given distance of the target, and $r_t = -1$ otherwise. At the start of each episode, the arm is initialized to a default pose (?) and the initial location of the puck is randomly placed within a fixed bound (?) on the table.

3 Legends

1. Text followed by (?) indicates that the paper does not explain clearly the point in the text.

4 Method description

4.1 State and Action representation

The state consists of the joint positions and velocities of the arm, the position of the gripper (there is no gripper?), the puck's position, orientation, linear and angular velocities (how are these measured?). The combined features result in 52D state space. Actions from the policy specify target joint angles for a position controller. Target angles are specified as relative offsets from the current joint rotations. This yields a 7D action space.

4.2 Domain Randomization

During training, rollouts are organized into episodes of a fixed length (what's the length?). At the start of each episode, a random set (?) of dynamic parameters μ are sampled according to ρ_μ and held fixed for the duration of the episode. In total, there are 95 randomized parameters. Within each episode, action timestep and the observation noise varies randomly each timestep. The rest of the parameters are sampled at the beginning of each episode and kept constant across the episode. Each subsection below discusses how each dynamic parameter is randomized:

4.2.1 Mass of each link in the robot's body

The value is logarithmically sampled from the range $[0.25, 4]\times$ default mass of each link.

4.2.2 Damping of each joint

The value is logarithmically sampled from the range $[0.2, 20]\times$ default damping of each joint.

4.2.3 Mass, friction and damping of the puck

The mass is uniformly sampled from $[0.1, 0.4]kg$. Friction is logarithmically sampled from $[0.1, 5]$. Damping is logarithmically sampled from $[0.01, 0.2]Ns/m$.

4.2.4 Table height

The value is uniformly sampled from $[0.73, 0.77]m$

4.2.5 Gain for the controller (which controller?)

The value is logarithmically sampled from $[0.5, 2.0]\times$ default gain

4.2.6 Timestep between actions

The timestep between actions specifies the amount of time an action is applied before the policy is queried again to sample a new action. This serves as a simple model of the latency exhibited by the physical controller. The action timestep varies randomly each timestep. The timestep Δt between actions varies every step according to $\Delta t \sim \Delta t_0 + Exp(\lambda)$, where $\Delta t_0 = 0.4$ is the default control timestep and $Exp(\lambda)$ is an exponential distribution with rate parameter λ. While Δt varies every timestep, λ is fixed within each episode. The action timestep λ is uniformly sampled from the range $[125, 1000]s^{-1}$ (not totally clear what -1 means here).

4.2.7 Sensor (observation) noise

The observation noise models uncertainty in the sensors and is implemented as independent Gaussian noise applied to each state feature. The observation noise varies each timestep. The noise has mean 0 and a standard deviation of 5% of the running standard deviation of each feature.

4.3 Simulation details

All simulations are performed with Mujoco with a simulation timestep of $0.002s$. 20 simulation timesteps are performed for every control timestep. Each episode consists of 100 control timestep, corresponding to approximately 4 seconds per episode, but may vary as a result of the random timesteps between action.