Closing the Sim-to-Real Loop: Adapting Simulation Randomization with Real World Experience

Yevgen Chebotar, NVIDIA, University of Southern California

Ankur Handa, NVIDIA

Viktor Makoviychuk, NVIDIA

Miles Macklin, NVIDIA, University of Copenhagen

Jan Issac, NVIDIA

Nathan Ratliff, NVIDIA

Dieter Fox, NVIDIA, University of Washington

The Reality Gap

The difficulty of transferring simulated experience into the real world

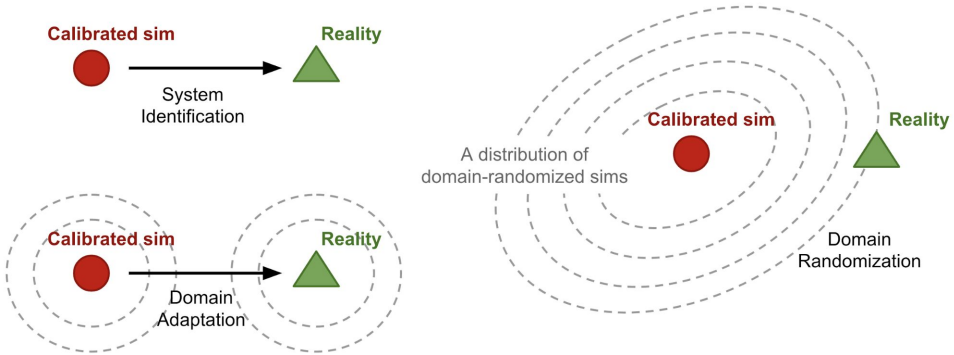
- Imprecise simulation modules
- Lack of high fidelity replication of real-world scenes

Proposal of the Paper: closing the reality gap by learning policies on distributions of simulated scenarios that are optimized for a better policy transfer.

https://ai.googleblog.com/2017/10/closing-simulation-to-reality-gap-for.html

Closing the Reality Gap

- System identification
  - Building mathematical model for the physical system
  - Calibrating simulator
- Domain adaptation
  - Matching simulator data distribution to real one
  - Adversarial loss or GAN
- Domain randomization
  - Train model across a variety of simulated environments
  - Real system is expected to be one sample in that rich distribution of training variations



Weng L., Domain Randomization for Sim2Real Transfer, lilianweng.github.io/lil-log

Preliminaries

- ➔ M = (S, A, P, R, p_0, γ, T) is a finite-horizon Markov Decision Process (MDP)
- ➔ **Aim of RL:** Maximize $E_{\pi\,\theta}$ [R(τ)]
- ➔ Distribution of simulation parameters ξ ~ p_{ϕ}(ξ) parameterized by ϕ
- ➔ Resulting system dynamics $P_{\xi \sim p\phi}$ = P (s_{t+1} |s_t, a_t, ξ)

$$\max_{\theta} \mathbb{E}_{P_{\xi \sim p_\phi}} \left[\mathbb{E}_{\pi_\theta} [R(\tau)] \right] \qquad (1)$$

Problem with Domain Randomization

- Requires significant expertise
- Tedious manual fine-tuning to design simulation parameters
- Also, using overly wide distributions is disadvantageous

Solution: Automating the learning of $p_{\phi}(\xi)$

Learning simulation Randomization

GOAL: *"find a distribution of simulation parameters that brings observations or partial observations induced by the policy trained under this distribution closer to the observations of the real world."*

π_{θ,p_ϕ} policy trained under the simulated dynamics distribution $P_{\xi\sim p_\phi}$

$D(\tau_\xi^{ob}, \tau_{real}^{ob})$ discrepancy between real world and simulation observation

$$\min_{\phi} \mathbb{E}_{P_{\xi\sim p_\phi}} \left[\mathbb{E}_{\pi_{\theta,p_\phi}} \left[D(\tau_\xi^{ob}, \tau_{real}^{ob}) \right] \right] \qquad (2)$$

Iterating for each Φ

$$\min_{\phi_{i+1}} \mathbb{E}_{P_{\xi_{i+1}\sim p_{\phi_{i+1}}}} \left[\mathbb{E}_{\pi_{\theta,p_{\phi_i}}} \left[D(\tau_{\xi_{i+1}}^{ob}, \tau_{real}^{ob}) \right] \right] \qquad (3)$$

$$\text{s.t.} \quad D_{KL}\left(p_{\phi_{i+1}} \| p_{\phi_i}\right) \leq \epsilon,$$

Iterative approach to approximate the optimization

Learning simulation Randomization

GOAL: *"find a distribution of simulation parameters that brings observations or partial observations induced by the policy trained under this distribution closer to the observations of the real world."*

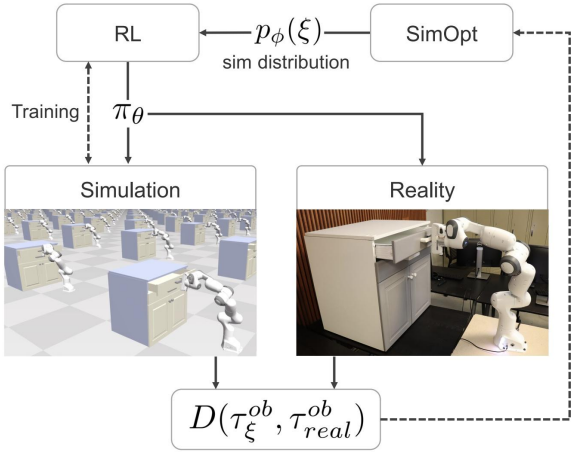


Fig. 3. The pipeline for optimizing the simulation parameter distribution. After training a policy on current distribution, we sample the policy both in the real world and for a range of parameters in simulation. The discrepancy between the simulated and real observations is used to update the simulation parameter distribution in SimOpt.

Learning simulation Randomization

GOAL: *"find a distribution of simulation parameters that brings observations or partial observations induced by the policy trained under this distribution closer to the observations of the real world."*

Algorithm 1 SimOpt framework

$p_{\phi_0} \leftarrow$ Initial simulation parameter distribution
$\epsilon \leftarrow$ KL-divergence step for updating p_ϕ
for iteration $i \in \{0, \ldots, N\}$ **do**
  env $\leftarrow$ Simulation(p_{ϕ_i})
  $\pi_{\theta, p_{\phi_i}} \leftarrow$ RL(env)
  $\tau^{ob}_{real} \sim$ RealRollout$(\pi_{\theta, p_{\phi_i}})$
  $\xi \sim$ Sample(p_{ϕ_i})
  $\tau^{ob}_{\xi} \sim$ SimRollout$(\pi_{\theta, p_{\phi_i}}, \xi)$
  $c(\xi) \leftarrow D(\tau^{ob}_{\xi}, \tau^{ob}_{real})$
  $p_{\phi_{i+1}} \leftarrow$ UpdateDistribution$(p_{\phi_i}, \xi, c(\xi), \epsilon)$

A Particular Implementation

- **RL Training** on a GPU based non-differentiable simulator using a parallelized version of PPO
- **Simulator parameter distribution** Gaussian
- **Discrepancy function:**

$$D(\tau_{\xi}^{ob}, \tau_{real}^{ob}) = \tag{4}$$
$$w_{\ell_1} \sum_{i=0}^{T} |W(o_{i,\xi} - o_{i,real})| + w_{\ell_2} \sum_{i=0}^{T} \|W(o_{i,\xi} - o_{i,real})\|_2^2,$$

- **To optimize the objective (3):** Sampling based gradient-free algorithm based on relative entropy policy search (REPS) which is able to perform updates of p_ϕ
  - Optimize using only samples $\xi \sim p_\phi$ and the costs from discrepancy function.

REPS

- A policy search method
- Finding the optimal policy that maximizes the expected return based on all observed series of states, actions and reward
- Intends to bound the loss of information measured using relative entropy between the **observed data** distribution and the **data distribution** generated by the new policy π
- As a result, it prevents the **new data distribution** to move outside the **old data distribution** by using KL divergence

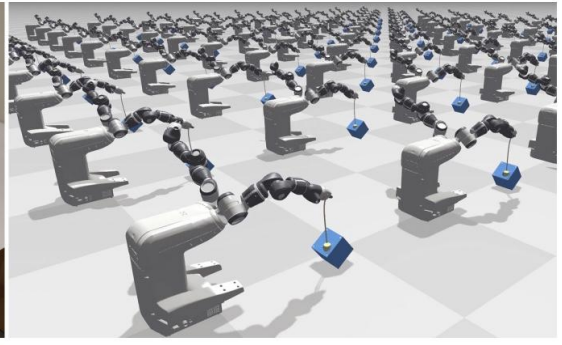
J. Peters, K. Mlling, and Y. Altun. Relative entropy policy search. In AAAI, 2010

Experiments

1. How does our method compare to standard domain randomization?
2. How learning a simulation parameter distribution compares to training on a very wide parameter distribution?
3. How many SimOpt iterations and real world trials are required for a successful transfer of robotic manipulation policies?
4. Does our method work for different real world tasks and robots?

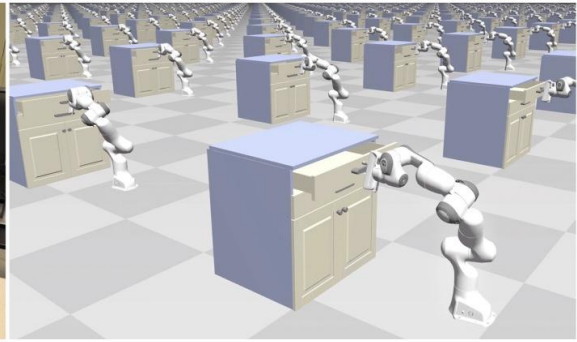
Tasks - Swing-peg-in-hole

- 7-DoF Yumi robot
- **Observation Space:** 7-DoF arm joint configurations, 3D position of the peg.
- **Reward in simulation:** Distance of the peg from the hole, angle alignment with the hole, a binary reward for solving the task.



Tasks - Drawer opening

- 7-DoF Panda arm
- **Observation Space:** 7D robot joint angles, 3D position of the cabinet drawer handle.
- **Reward in simulation:** the distance penalty between the handle and end-effector positions, the angle alignment of the end-effector and the drawer handle, the opening distance of the drawer, an indicator function ensuring that both robot fingers are on the handle.



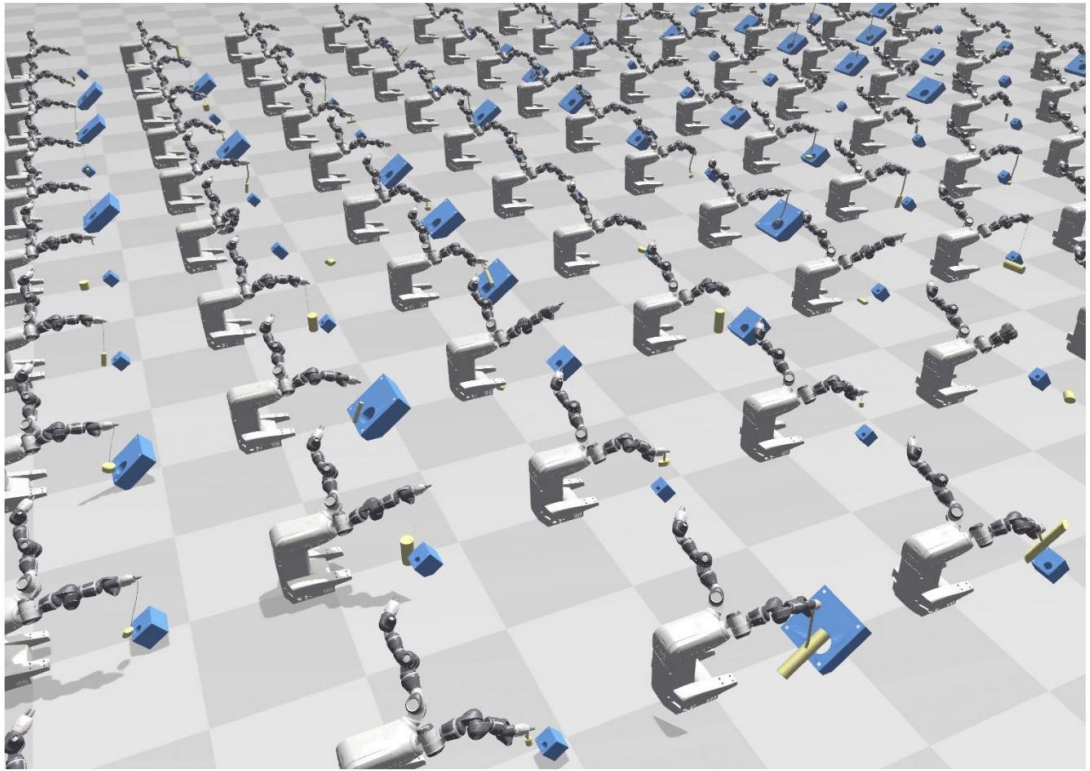


Fig. 4. An example of a wide distribution of simulation parameters in the swing-peg-in-hole task where it is not possible to find a solution for many of the task instances.

Comparison to standard domain randomization

- In DR, learning performance strongly depends on the variance of the parameter distribution.
- Increasing variance → Conservative Policies
- Further increase in variance:
  - Unrealistic scenarios
  - Catastrophic breakdown of the simulation
- $\sigma^2 = 0.0007$ (2cm)
- $\sigma^2 = 0.01$ (10cm)

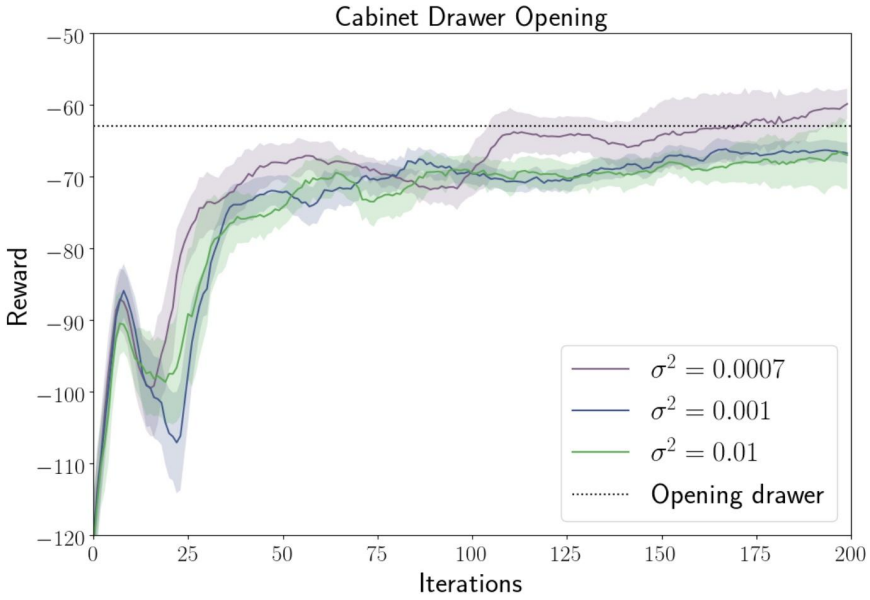


Fig. 5. Performance of the policy training with standard domain randomization for different variances of the distribution of the cabinet position along the X-axis in the drawer opening task.

Policy Transfer

- Train policy with RL on a conservative initial simulation parameters
- Run on target scene to collect roll-outs
- Use roll-outs to perform SimOpt iteration

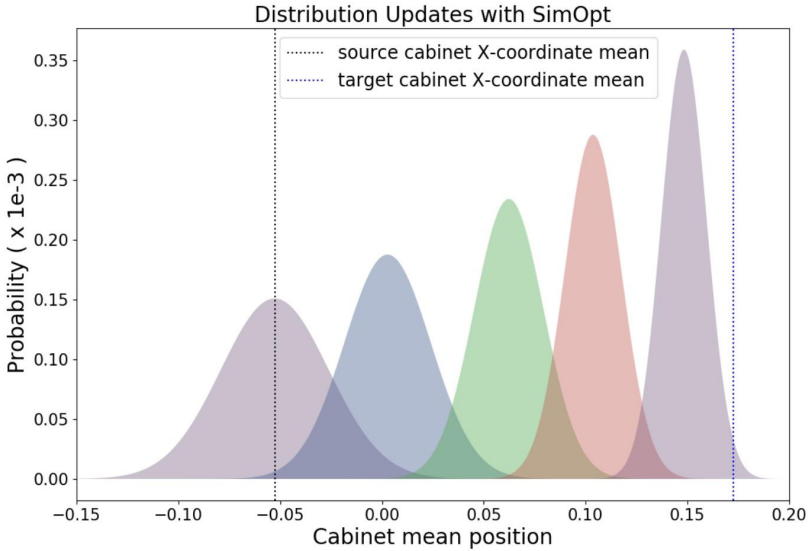


Fig. 6. Initial distribution of the cabinet position in the source environment, located at extreme left, slowly starts to change to the target environment distribution as a function of running 5 iterations of SimOpt.

Real Robot Experiments

- RL training and SimOpt parameter sampling on 64 GPUs
  - 150 simulated agents per GPU
- Real world object tracking with DART to continuously tracking 3D positions of peg and handle of the cabinet
- **Simulation parameters:** Multivariate Gaussian

Real Robot Experiments