

DROID: Minimizing the Reality Gap using Single-Shot Human Demonstration

Ya-Yen Tsai¹, Hui Xu², Zihan Ding³, Chong Zhang³, Edward Johns¹, and Bidan Huang^{3†}

Abstract—Reinforcement learning (RL) has demonstrated great success in the past several years. However, most of the scenarios focus on simulated environments. One of the main challenges of transferring the policy learned in a simulated environment to real world, is the discrepancy between the dynamics of the two environments. In prior works, Domain Randomization (DR) has been used to address the reality gap for both robotic locomotion and manipulation tasks. In this paper, we propose Domain Randomization Optimization Identification (DROID), a novel framework to exploit single-shot human demonstration for identifying the simulator’s distribution of dynamics parameters, and apply it to training a policy on a door opening task. Our results show that the proposed framework can identify the difference in dynamics between the simulated and the real worlds, and thus improve policy transfer by optimizing the simulator’s randomization ranges. We further illustrate that based on these same identified parameters, our method can generalize the learned policy to different but related tasks.

Index Terms—Transfer Learning, Learning from Demonstration, Manipulation Planning

I. INTRODUCTION

Reinforcement Learning (RL) has been widely applied to decision making, control, and planning. In the field of robot learning, many works have adopted RL as the robot controlling policy, to improve its learning efficiency and performance [1]–[3]. Recent works have demonstrated that RL can be used to control a dexterous robotic hand or a robotic arm to solve tasks that require complicated manipulation skill, such as solving a Rubik’s cube [4], [5] or opening a door [6].

RL uses self-exploration to find the optimal policy. Typically, this requires a very large amount of trial-and-error, which is time-consuming and can easily result in hardware damage if executed on the physical robot. A less costly and safer approach is learning the policy via simulation. However, the discrepancy between the real world and the simulation models could hinder the policy from directly being deployed in real world, especially when the task is contact-rich. In the literature, this *sim-to-real* problem is referred to as the reality gap, which remains an open issue to date.

System Identification (SI) and *Domain Randomization* (DR) are the two common approaches to cross the dynamics reality

† denotes the corresponding author.

¹Y.-Y. Tsai is with the Hamlyn Centre for Robotic Surgery and E. Johns is with the Robot Learning Lab, Department of Computing, Imperial College London, SW7 2AZ, London, UK {y.tsai17, e.johns}@imperial.ac.uk

²H. Xu is with School of Computer Science and Engineering, University of Electronic Science and Technology of China hui_xu@std.uestc.edu.cn

³Z. Ding, C. Zhang, and B. Huang, are with Tencent Robotics X, China {zihan.ding18, chongzhang, bidanhuang}@tencent.com

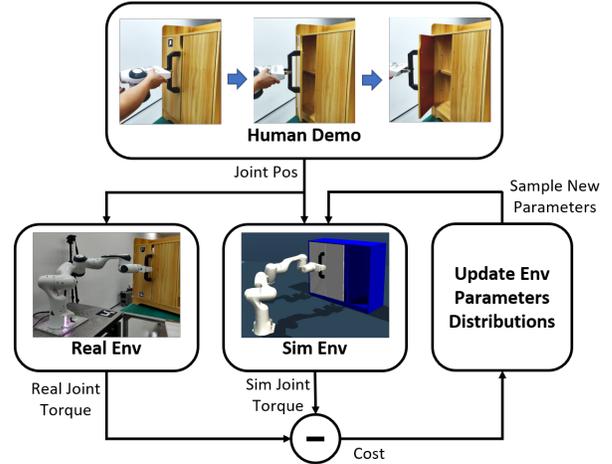


Fig. 1: This figure presents the overview of the proposed framework, DROID, used to minimize the reality gap.

gap. While SI tries to reduce the reality gap by identification of the real-world parameters, DR tries to increase robustness to the reality gap by training on randomized simulated environments. However, these methods still often struggle to accurately obtain real-world parameters or choose randomization ranges without any real data [7]. This can result in learning a biased policy, or a policy which fails to converge.

To address this, we propose a novel framework, Domain Randomization Optimization Identification (DROID), to automatically optimize the environment parameter distribution, with a combination of DR and SI approaches. Rather than determining the DR range using intuition or tedious tuning, we exploit human demonstrations and attempt to align simulated trajectories with these real-world trajectories. Through this, we identify the simulator’s optimal parameter range as a statistical model, which can then be sampled from during training with RL. An overview of this method is shown in Fig. 1.

With DROID, the learned policy can be transferred to the real world directly, thus learning efficiency is significantly improved and unsafe interactions with the environment is avoided. Our experimental results show that after parameter optimization and identification, a much higher success rate can be achieved with DROID on a real-world door opening task, compared with standard DR or SI. We also show how the learned dynamics can be directly used to train policies for different but related tasks.

II. RELATED WORK

RL’s applications in robotics suffer from expensive training data collection in real robots [8]. By offering cheaper and

safer data collection environment, simulation has gained huge popularity for RL training [4], [9]–[11]. However, the issue of the reality gap still remains one of the main problems hindering the policy learned in simulation from transferring well to the real world. To bridge the reality gap, two common strategies have been studied in prior works: SI, and DR [7].

SI has been a popular approach for sim-to-real transfer [12]–[17]. It focuses on finding the exact model of the real world, so the physical behaviors match between the real and simulated system. This could be done by constructing the simulation through direct measurement of the environment parameters in the real world [18] or by collecting real world data for optimizing the simulated model parameters [19]–[21]. However, correctly identifying the system’s parameters is challenging. Many parameters cannot be explicitly measured or can involve presence of noises, especially for dynamics related ones, like friction, stiffness, and damping. In addition, the system parameters could be of high-dimensional and entangled. This further increases the difficulties in achieving accurate and precise SI results [7], [22], [23]. As a consequence, SI usually requires expertise of the system to handcraft the model.

Rather than identifying the environment parameters, DR creates multiple simulated environments by randomizing the system parameters within given ranges during policy training. This improves the policy’s generalizability and robustness against the reality gap. Recently, it has achieved significant progress for sim-to-real transfer in robotics [4]–[7], [21], [24]–[28]. Unlike SI, DR achieves better sim-to-real performance by covering a greater range of parameters distribution in the simulation containing the real values during RL training. Prior works optimize system parameters with simulated and real trajectories collected using hand-designed policies [4]; or automatically adjust the boundaries of uniform randomization distributions according to model performances [5]. While effective, these works suffer from a common drawback. Existing works [7] have demonstrated its demanding engineering efforts in adjusting the randomization ranges, which is difficult and not intuitive. Hand tuning the parameter ranges could easily cause overestimated values and lead to training RL in an invalid environment and learning a suboptimal policy. How to quickly choose the randomization ranges for different parameters and achieve effective policy generalization still remains a challenge.

Besides the two main approaches, a variant of DR, Adaptive DR, has also been studied. It was proposed to optimize the parameter distributions and minimize the chance of training RL in invalid environments. Prior works use techniques such as approximate Bayesian computation [29], Bayesian Optimization, or a relative entropy policy search [30] to estimate or optimize distributions of system parameters [26], [27], [31]. The proposed framework draws some similarities to these works in avoiding overestimation of parameter distribution, but focuses on more contact-rich task which involves determining complicated and entangled dynamics related parameters and optimizing the distribution through a more efficient and safer approach, i.e. human demonstration. In addition, we optimize the randomization distributions with respect to trajectories containing not only the observed positions and/or velocities,

but also the proprioceptive torques on joints of the robot arm. Experiments were conducted on a contact-rich task, door opening, with DROID due to its complex dynamics of the robot joints, the door hinge and the contacts between the gripper and the handle.

The rest of the paper is organized as follows: The methodologies is presented in Section III, followed by the experimental setup, results and the discussion section in IV. Finally, the conclusions and the future works are presented in Section V.

III. METHODOLOGY

Learning in a simulated environment is convenient, but transferring the learning results to the real environment requires an accurate model of that environment. Simulation typically uses mathematical models to compute the interaction force and torque between objects. These models rely on pre-defined dynamics-related parameters such as friction, stiffness and damping. Unlike kinematics-related parameters, many of them are not easily accessible and hence are often difficult to measure and identify. Therefore, tasks that involve these parameters experience difficulty in sim-to-real transfer. To this end, we propose a framework DROID that evaluates these parameters through human demonstration. Building on the concept of interaction force, we implicitly perceive dynamics information of the real-world system from the feedback of the robot and use this information to determine the distribution of the parameters in the simulation. This gives us a reasonable set of parameters for the domain randomization in RL and hence results in a successful policy transfer.

DROID is composed of three phases: the human demonstration (Section III-A), the parameter identification and optimization (Section III-B), and the policy learning with optimized DR (Section III-C). In the first phase, the human demonstrates a contact task in the real world. The robot clones the human behaviors multiple times and records the data. In the second phase, robot in the simulator repeats the same behaviors and records a same set of data. The data from the real system and the simulator is then used for identifying the distribution of the task relevant parameters. Through an iterative approach, we can gradually update the parameter distribution to minimize the differences between the two perceived feedback until the obtained simulated environments can better reflect to the real world. In the final phase, policy is trained with DR based on the parameter distribution optimized. The resulting policy can be transferred to the real with good performance for the previous optimization steps. Note that this study focuses on minimizing the reality gap and the human demonstrations only serve for the purpose of identifying the task relevant parameter distribution. Learning the task from human demonstration is out of the scope of this paper and in the third phase we learn the policy from scratch without human demonstrations. In the following sections, we will go into more details on how each part is implemented.

A. Single-Shot Human Demonstration

In this first phase, our aim is to collect the data reflecting dynamics relevant to the task. To this end, human demonstrates

the task once in the real world to provide a robot motion trajectory, q_d , through kinesthetic guidance. This demonstration is safe to be executed by the robot and allows the robot to repeat it multiple times automatically. By repeating the q_d to interact with the environment, the dynamics information can be perceived by the torque sensor feedback τ_r of the robot. Such feedback is later used as the reference to identify and update the parameter distribution in the simulation. Different from the approaches that makes the robot to randomly interact with the environment, in DROID we rely on the human to provide a trajectory that is safe for the robot to identify the system dynamics. This only requires a single-shot demonstration. We focus on the task relevant parameter distributions and limit the random exploration of the robot in the real world. This minimizes the risk and save the time in the real robot experiment.

B. Parameter Optimization and Identification

The goal in this phase is to correctly identify the parameter distribution that minimizes the discrepancy of the simulated and the real system. Rather than finding the specific value for each parameter, we determine the distributions of them for the presence of noises and uncertainties in the real world. We can then train the RL to find a policy that works within this distribution, which is the key problem solved in DROID. Falsely defined distribution will lead to failure in sim-to-real transfer, and the policy trained under an unreasonable DR can suffer bad performance. We model this distribution as a multivariate normal distribution $\Phi(\mu, \Sigma)$, where μ and the Σ is the mean and covariance, and system parameters ϕ is randomly sampled from Φ .

During the human demonstration phase, data has been collected from the real system. Taking the identical steps, we can program the robot to repeat the same task in the simulation and hence obtain the torque sensor feedback τ_s . As the real world parameter value ϕ' is not easily accessible, we align the simulation and the real environment by aligning the robot behaviors in them. Here, we define the ‘‘behavior’’ as the robot action and perception pairs, i.e. the motion trajectory q_d and the torque sensing τ_s . Changing ϕ changes the dynamics of the simulation and hence changes the robot behavior. Sampling different ϕ from its distribution Φ , we observe the different behaviors under different environments in the simulator and identify the ones that are most similar to the real world behavior. We hence update Φ based on the robot behaviors.

We formulate this as a distribution optimization problem and update Φ iteratively based on the Covariance Matrix Adaptation Evolution Strategy (CMA-ES) approach [32] with the following objective function:

$$\mathcal{J}(\phi) = \frac{1}{N} \sum_{n=1}^N (\|\tau_s(\phi) - \tau_r^n(\phi')\| + c\beta), \phi \sim \Phi \quad (1)$$

where N , β , c is the total number of trajectories from the real robot, a penalty for failure of the task and the factor for the penalty, respectively.

The process is iterated from the initial guess, $\Phi_{init} = \mathcal{N}(\mu_{init}, \Sigma_{init})$, until convergence. Note that we append $c\beta$ at

the end of the fitness function to penalize the situation where the robot fails to grip the door knob during door opening process in the simulation. This occasionally happens when for example the friction coefficients of the fingers become to low or when the joint damping is set to be invalid. The overview of the described algorithm is summarized in Algorithm 1. M is the total number of samples from the current distribution Φ . For each iteration, the robot performs the task in the simulator under the environment parameter ϕ sampled from the distributions Φ . The cost $\mathcal{J}(\phi)$ is therefore evaluated with the resulting τ_s in Eqn. 1. Note that in total N real robot trajectories are used to evaluate the cost and $\mathcal{J}(\phi)$ is the average value. After M iterations, the CMA-ES updates Φ from the x best candidates which associated with the lowest costs $\mathcal{J}(\phi)$. The update process of Φ (i.e., μ and Σ) and hyper-parameters of CMA-ES following the standard procedure. This repeats until Φ converges and we achieve the optimized Φ^*

Algorithm 1 Optimizing parameter distribution

- 1: Initialize hyper-parameters of CMA-ES
 - 2: Initialize Φ with $\mathcal{N}(\mu_{init}, \Sigma_{init})$
 - 3: **while** not converged **do**
 - 4: **for** $m = 1 : M$ **do**
 - 5: Sample ϕ_m from Φ
 - 6: Robot perform task in simulation with ϕ_m
 - 7: Collect $\tau_s(\phi_m)$
 - 8: Calculate $\mathcal{J}(\phi_m)$ in Eqn.1 with N real trajectories $\{\tau_r^1(\phi'), \dots, \tau_r^N(\phi')\}$
 - 9: **end for**
 - 10: Select x best ϕ from $\{\phi_1, \dots, \phi_M\}$ by $\min \mathcal{J}(\phi)$
 - 11: Update $\Phi = \mathcal{N}(\mu, \Sigma)$ with the selected ϕ set
 - 12: Update hyper-parameters of CMA-ES
 - 13: **end while**
 - 14: **return** optimized Φ^*
-

C. Policy Learning

In this paper, we adopt a reinforcement learning framework to learn an optimal control policy π_θ , parameterized by θ , through policy gradient. This is formulated as a Markov decision process (MDP) which is defined by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, r, \rho_0, \gamma)$, where \mathcal{S} is the state space, \mathcal{A} is the action space, $\mathcal{P} : p(s_{t+1}|s_t, a_t)$ is a distribution of state transition, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ is the reward function, ρ_0 the initial state distribution and $\gamma \in (0, 1)$ is the discount factor. The optimal policy π_θ^* aims to maximize the cumulative reward over episodes:

$$\pi_\theta^* = \arg \max_{\pi_\theta} \mathbb{E}_{\pi_\theta} \left[\sum_t r(s_t) \right]. \quad (2)$$

Proximal Policy Optimization (PPO) [33] is deployed for the robot learning purpose. It updates the policy by using the surrogate objective:

$$L(\theta) = E_t[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1 - \epsilon, 1 + \epsilon)\hat{A}_t)], \quad (3)$$

where, \hat{A}_t is the estimate of the advantage function at timestep t , and $r_t(\theta)$ denotes the ratio between the current policy and



Fig. 2: (a) and (b) shows the hardware setup in the real world and simulation. The Aruco markers attached to the table and the cabinet are used for the tracking purpose. (c) shows four different locations of the door knob, each separated by 5cm along the lever arm of the door. (d) shows three variants of the door. From left to right are the original door, door with one spring and door with two springs attached to the hinge. The springs were used to emulate doors with different dynamics.

the previous policy. The clipped term keeps the ratio inside the interval $[1 - \epsilon, 1 + \epsilon]$, the minimum of the clipped and unclipped term is used for the expectation. This provides a lower bound, or a pessimistic bound, on the unclipped term.

As the model of the real world environment is described by Φ^* , sufficient simulation environments are hence required to be sampled from the distribution in order to reflect to the reality. Therefore, the RL agent is trained on multiple simulated environments sampled from Φ^* . By doing so, we hope to maximize the similarity in state transition between the simulated and the real environments and enhance the transferability of the learned policy to the real world application.

The structure of the reward function follows closely to the one defined in [6] and is presented as follows:

$$r = \begin{cases} \omega_1 \cdot r_{door} + \omega_2 \cdot r_{ori} \\ \quad + \omega_3 \cdot r_{dist} + \omega_4 \cdot r_{log_dist} + \omega_5 \cdot r_{slip}, & \text{if } \lambda < 30^\circ, \\ \omega_1 \cdot r_{door} + \omega_2 \cdot r_{ori} + \omega_5 \cdot r_{slip}, & \text{otherwise} \end{cases} \quad (4)$$

where λ is the hinge angle of the door, ranging from 0° (closed door) to 90° (completely opened door). The reward function has a total five terms. The r_{door} rewards the action that results in the increase to the door hinge. The r_{ori} rewards the relative orientation between the door knob and the gripper. The higher reward is given if the relative orientation is closer to orthogonal. The r_{dist} and r_{log_dist} are associated with the relative displacement between the door knob and the gripper. The higher reward is given to the short displacement. One major difference between our door opening strategy and the DoorGym's, lies in that they use hooks to open the door while our robot is trained to firmly grip knob handle during the entire task. This significantly increases the complexity of dynamics involved. To this end, we added a penalty term r_{slip} in the reward function to prevent the fingers from slipping. $\omega_1, \omega_2, \omega_3, \omega_4$, and ω_5 are five coefficients for normalizing each reward terms.

IV. EXPERIMENTS AND RESULTS

We evaluate our work through a door opening task for its complexity in the dynamics involves. The main focus of our experiments is to access how well the RL policy learned with

DROID can be transferred to the robot in the real world. This evaluation consists of three experiments.

The first experiment validated that DROID can identify the parameter distributions for environments and with different dynamics. In the second part, we applied one of the optimized distributions to DR and train a policy for door opening. The evaluation on the performance of this policy was done by comparing its robustness and effectiveness in sim-to-real transfer with other approaches (standard DR and without DR). Finally, we tested the generalizability of the RL policy learned from the optimized parameter distributions.

A. Experimental Setup

The experiment consisted of a real and a simulated system. In the real system, a cabinet door, a camera and a 7-DoF Franka Emika robot arm were used as illustrated in Fig. 2(a)(b). The Franka robot was equipped with 7-DoF joint torque sensors allowing it to record the torque feedback while interacting with the environment. A two fingers gripper was mounted at the end effector for grasping and manipulation purpose. The cabinet was the target of interest in which we attempted to identify its dynamics parameter distributions. The fixed camera provided the relative pose information between the robot and the door as well as the door angle information via tracking the visual markers attached to the door and the optical table during the experiment.

For the simulation part, we deployed the MuJoCo platform [34] to perform the RL training. The simulated environment was defined by a set of parameters including kinematics tree, and many dynamics such as mass, damping, friction etc. Many kinematics such as relative poses and the geometric dimensions and the robot related dynamics such as mass and inertia were either provided officially or can be directly measured and hence were not within our focus. Inertia of the door was also not considered as it can be estimated once we obtained the CAD model. Our main interest was to identify the parameter distributions of the dynamics of the robot and the door that were not provided and difficult to measured in the real world. These parameters included mass, joint friction loss, and joint damping and the sliding and torsional frictions.

B. Parameter distribution identification

In the first experiment, we investigate the feasibility of DROID in the distributions identification. This verifies whether DROID can find the distribution of parameters that correctly reflects the interaction behavior encountered in the real-world door opening task.

For the given cabinet door (Fig. 2), the real robot followed the human demonstration to obtain ten sets of τ_r that reflected the dynamics of the interactions with the door. We first made an initial guess of the Φ_{init} for the parameters of interest. The estimation was made by referencing Franka’s officially provided values and DoorGym’s parameters [6] with the exceptions being the door and knob masses which were directly measured. At the first iteration, 30 simulated environments were sampled from the initial distributions. Parameters sampled with negative values were omitted and resampled. With each environment, the robot cloned the human demonstration by following q_d to obtain τ_s . The associated cost for each trail was calculated using Eqn. 1. The higher the discrepancy among the torques, the higher the cost would be gained. For the simulation that failed to successfully open the door due to factors like grasp slipping, an extra penalty of 10 was added to the cost. CMA-ES algorithm took the top five best candidates of ϕ to update the means and the covariances and hence the Φ . In a new iteration, the updated Φ were used to generate another 30 new simulations and this process was iterated until convergence.

With the above steps, we have estimated the parameter distribution for our cabinet door (Tab. I). These parameters reflect the dynamics of the robot arm, of the door hinge and the contacts. Fig. 5 illustrates the optimization process and results. Fig. 5(a) shows the joint torque trajectories of the robot obtained in the simulation before and after the optimization, and the joint torque trajectory obtained in the real world. This only displays one of the joint for illustration. As it can be seen, differences between the red (after optimization) and the black (real robot) lines are much smaller than the differences between the blue (before optimization) and the black. This suggests that the proposed approach can indeed minimize the reality gap. The same conclusion can be drawn from Fig. 5(c) plotting the cost against the iteration. Among the 30 simulations the average cost gradually converged to a lower value. This indicates that the simulations sampled from the optimized distribution lead to smaller reality gap than those sampled from the initialed distributions. The means and the variances of three parameters at each iteration are shown in Fig. 5(c) and these show the distribution of these parameters converged to fixed ranges. The quantitative results comparing the unoptimized and optimized distributions are summarized in Tab. I. We have applied this result to the sim-to-real transfer experiment, which is detailed in the next section.

Furthermore, a validation was carried to determine the possibility variations in estimating the parameter distribution using different single human demonstration. Different human demonstration was provided through an alternative robot pose as illustrated in Fig. 3 for validation. Note that due to robot workspace limitation, only the presented two poses could be



Fig. 3: Two different initial robot poses used in the human demonstrations. (a) is the initial robot pose used to estimate parameter distribution in Tab. I. (b) is another initial robot pose used to validate the possibility of a bias in parameter estimation.

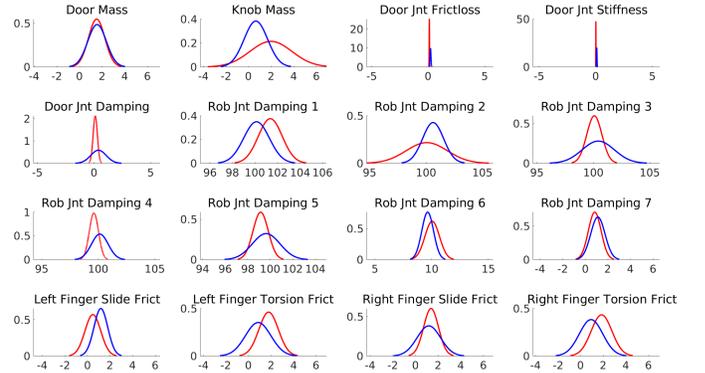


Fig. 4: This compares the parameter distributions estimated using two different initial poses and human demonstrations. Red and blue curves represent the parameter distribution estimated using robot pose (a) and (b) shown in Fig. 3.

applied to interact with and open the door. The parameter distribution identification process was repeated using this pose. The experimental result comparing the estimated parameter distribution using the two human demonstrations is shown in Fig. 4. We verified that despite for the minor variations, the majority of parameter distributions estimated using the alternative pose still converged to similar distributions as stated in Tab. I and single human demonstration was adequate for parameter estimation.

In order to further verify the approach, we have conducted two sets of controlled experiments by changing the dynamics of the real door.

The original door was modified to include additional springs to change its dynamics. Fig. 2(d) illustrates the three variants of the door, without any spring, with one spring and with two springs attached to the hinge. These springs were identical and share similar dynamics.

The optimization results for different door dynamics are also summarized in Tab. I. It can be seen from the parameter values that the robot dynamics is not much affected but the environment dynamics varies a lots. This is due to the fact that we only changed the door dynamics and the algorithm is able to identify this change correctly. Doors equipped with different amount of springs would expect to have different distributions for the door joint frictionloss, stiffness and damping. For the frictionloss, door with 2 springs seem to have higher friction loss than the other two scenarios. The joint stiffness

increases as the number of spring equipped increase, which is reasonable. Doors with springs tend to have higher joint damping values than the door without the spring as expected. The joint dampings associated to the robot falls into the similar values. This shows the proposed framework have consistant results. The masses and the frictions vary across different scenarios, and this may be attribute to the robot which needs higher friction in the gripper in order to sustain the sufficient force to grip the door knob in the simulation when more spring is loaded to the door.

Based on the above evaluation results, the proposed DROID framework has demonstrated to be feasible in determining the parameter distribution of the real world dynamics. In theory, the reality gap is smaller after optimization. The framework has also shown to be realizable to identify task with different dynamics. Most outcomes have appeared to be reasonable with a few exceptions which may be attribute to factors such as imperfect demonstration resulting in loss of grip when door becomes stiffer or noises present in joint torque sensors in the real world. We have used the optimized result in sim-to-real practice and detail it in the Section IV-C.

C. Sim-to-real transfer with optimized DR

While in many prior works, domain randomization methods have shown to be effective in addressing the reality gap, selecting the randomization ranges still remain challenging, and training the RL agent on overestimated ranges may be inevitable and could lead to poor learning performance. In the last experiment, we have successfully identified the parameter distributions, but we have yet demonstrated it to be effective to learn adequate policy and transfer to the real world. Therefore, in this experiment, we aimed at comparing the learning performance of policies learned from three methods, namely normal DR, DROID without DR and DROID with DR. These policies were trained to open the door without springs in the simulation and then directly transferred to the real world to open the same door. Normal DR approach was trained on the estimated parameter distributions which correspond to the left most columns in Tab. I. DROID without DR, similar to SI, was trained using only fixed parameters. In this case, the parameters used were the optimized μ listed in column four of the Tab. I. Finally, DROID with DR was trained based on the optimized distributions, i.e. column four and five of Tab. I.

For the sake of fairness, all of them were trained using model-free on-policy approach, *i.e.* PPO, and shared identical RL hyperparameters, architectures of networks, observations, and reward functions, with the parameter ranges being the only differences. In the reinforcement learning setting, the observation included the joint positions and velocities of the arm, the relative positions between the robot gripper and the knob. These combined to form the 23 DoF state space. The action space consisted of 9 DoF, specifying the 7 robot joint positions and 2 gripper widths. The critic and actor models were represented by a neural network with two hidden layers of size 64, following tanh activation functions. We performed our simulations on MuJoCo physics engine with timestep of 0.001s. During the training, each episode took a maximum of

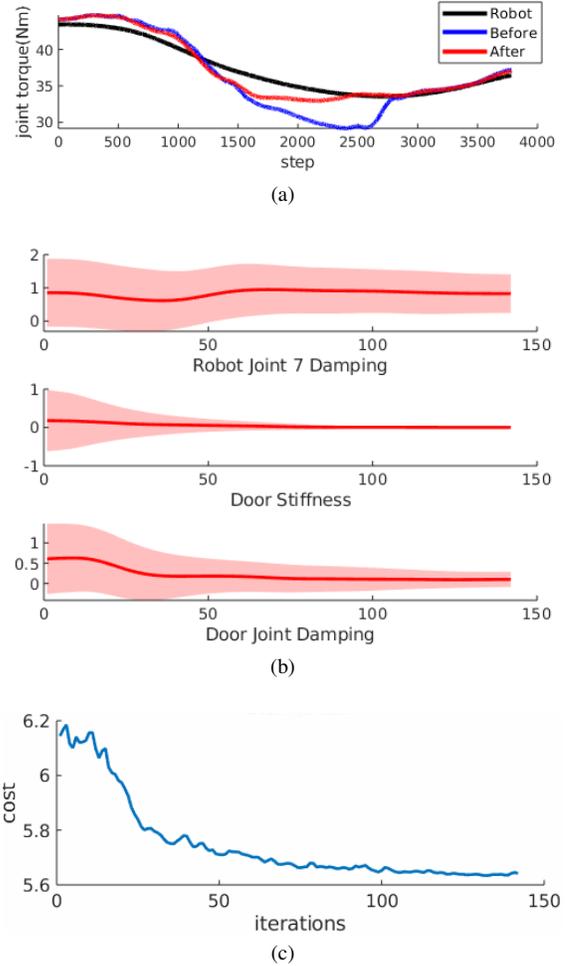


Fig. 5: (a) shows the comparison between a robot joint torque trajectory in real world and the joint torque trajectories before and after the optimization and identification for parameter distribution in simulation. (b) shows the changes in means (the solid lines) and the variances (the shaded areas) of the three parameters over iterations during optimization. The red line represent the mean and the shaded region represent the variance. (c) shows the associated cost computed using Eqn. 1 over iterations.

512 steps. We employed the ADAM optimizer with a stepsize of 0.001 and mini-batch of 64 episodes to update the policy and value networks.

We have trained three policies with each approach, and have tested each of them in the real world for 10 times. The results for sim-to-real comparisons are displayed in Tab. II. The door opening angles have been recorded each time and the histogram is shown in Fig. 6. We define a successful door opening case as the one with the hinge angle of the door larger than 30° in real-world trials. From these results, we can see the later two approaches (DROID with DR and without DR) outperform the standard DR. In the DROID with DR setting, the optimized means and variances for DR (80% success rate) can achieve significant advantages in real-world tests over the normal DR (20% success rate) with initialized means and variances. We also notice that the policies trained

TABLE I: This compares the initial and final parameters distribution after optimization and identification. The distribution is defined by μ and σ . Additionally, this also compares the optimized distributions for doors equipped with one or two springs.

	DoorGym		Door without spring		Door with 1 spring		Door with 2 springs	
	μ_{init}	$diag(\Sigma_{init})$	μ_{opt}	$diag(\Sigma_{opt})$	μ_{opt}	$diag(\Sigma_{opt})$	μ_{opt}	$diag(\Sigma_{opt})$
Door Properties								
Door Mass(kg)	1.144	0.5	1.50	0.73	0.92	0.52	2.54	1.23
Knob Mass(kg)	0.199	0.1	1.98	1.86	2.31	0.42	3.99	1.29
Friction Loss	0.05	0.025	0.10	0.02	0.0	0.0	0.56	0.05
Joint Stiffness	0.01	0.005	0.002	0.008	1.06	0.0	1.24	0.06
Joint Damping	2.0	1.0	0.10	0.19	0.71	0.01	0.6	0.18
Robot Properties								
Joint Damping (7DoF)	[100, 100, 100, 10, 0.4]	[2.0, 2.0, 2.0, 2.0, 2.0, 1.0, 0.2]	[101.35, 100.06, 100.08, 99.61, 99.14, 10.05, 0.83]	[1.06, 1.83, 0.67, 0.41, 0.68, 0.64, 0.57]	[98.11, 98.69, 100.38, 99.64, 101.01, 9.72, 1.17]	[0.41, 0.63, 0.65, 0.15, 0.96, 0.04, 1.19]	[98.63, 99.28, 95.83, 100.12, 95.71, 11.55, 1.54]	[1.23, 0.71, 1.42, 0.81, 0.77, 0.46, 0.67]
Gripper Properties (Left Right)								
Sliding Friction	[0.5, 0.5]	[0.25, 0.25]	[0.47, 1.78]	[0.70, 0.86]	[1.37, 0.76]	[0.35, 0.39]	[3.04, 2.68]	[0.46, 0.54]
Torsional Friction	[0.5, 0.5]	[0.25, 0.25]	[1.38, 1.84]	[0.66, 0.92]	[0.79, 2.29]	[0.42, 0.80]	[1.04, 2.0]	[0.72, 0.83]

without DR using optimized means μ_{opt} as system parameters in simulation can achieve the highest success rate of 86.7% in reality, even higher than the DROID with DR method. On one hand, this implies that the optimized μ_{opt} indeed achieves a good sim-to-real transfer. On the other hand, the fact that DROID without DR produces a higher success rate but lower door opening angles than DROID with DR is interesting. We attribute this to the relatively weak dependency on system dynamics when the door was opened with small angles (*e.g.*, 30°), but as the hinge angle increased, the systems dynamics became more important for the door opening process due to the larger contact forces between the gripper and the door knob. Hence, the policies cannot further open the door well without a proper DR in simulated training process, which was also testified in Fig. 6. From the distributions of maximal door-opening angle, we show that DROID with DR can achieve the door opening with more concentration on larger maximal hinge angles, *e.g.* around 60° , compared against normal DR and DROID without DR. Tab. II also demonstrates that DROID with DR can achieve the highest average door-opening angle among all three methods, even though it does not perform as great as the normal DR method in simulation. As for the number of steps taken to open the door, our method have advantageous performances consistently in both simulation and reality, which indicates a faster door-opening process.

D. Generalization of the Learned Policy

As mentioned in the previous section, we only used one human demonstration to determine the parameter distribution. This demonstration was, however, not used during RL because it can only serve for a door with the same dimensions. In this experiment, we tested the policy with three additional door knob locations in the simulation and in the real system as shown in the Fig 2(c) to emulate doors with different dimensions. These locations were set to be $5cm$ apart along the lever arm of the door. With DROID, the trained policy

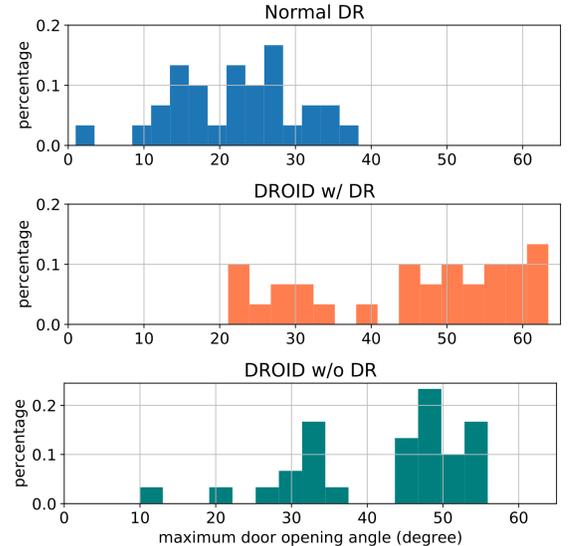


Fig. 6: Comparison of the door opening performance in reality for three methods (from top to bottom): normal DR, DROID with DR and DROID without DR. The horizontal axis is the maximal opening angle of the door in degrees. The vertical axis is the percentile value for each bin. It compares the distributions of maximal angles over 30 runs for each method.

TABLE II: Comparisons of Sim-To-Real results

		DR ($\mu_{init}, \Sigma_{init}$)	μ_{opt}	DR (μ_{opt}, Σ_{opt})
Success Rate (angle $> 30^\circ$)	sim	100%	100%	100%
	real	20%	86.7%	80%
Open Angle (mean \pm std)	sim	91.3 \pm 0.4	70.8 \pm 16.0	91.2 \pm 0.2
	real	22.4 \pm 8.2	42.2 \pm 11.1	45.4 \pm 13.6
Open Steps (mean \pm std)	sim	96.3 \pm 107.2	60.3 \pm 2.6	38.7 \pm 14.1
	real	68.7 \pm 8.4	73.6 \pm 7.0	68.3 \pm 11.0

was able to pull the door knob, at different locations, and open the door successfully. As such, we have verified that once the parameter distribution is identified using a human demonstration, different RL policies can be trained within the optimized distribution to accomplish tasks other than the one demonstrated.

V. CONCLUSION

In this paper, we proposed a novel and generic framework, Domain Randomization Optimization IDentification (DROID) to minimize the reality gap between simulated and real environments. The approach is designed to be applicable to a range of contact-rich manipulation tasks, such as door opening. By executing a human demonstration trajectory in both simulation and reality, the differences in their dynamics can be minimized by iteratively updating and identify the distributions of the real dynamics parameters. Using a door opening task as an example, we have verified its capability to identify reasonable parameter distributions and thus reduce the reality gap. A successful RL policy can then be obtained by training in this distribution and directly transferring to the real world. The sim-to-real performance has shown to be superior than training with typical DR and SI approaches. Finally, we also demonstrated that a generalized RL policy can be trained to accomplish different tasks, given that the system dynamics remains unchanged.

REFERENCES

- [1] A. A. Rusu, M. Večerik, T. Rothörl, N. Heess, R. Pascanu, and R. Hadsell, "Sim-to-real robot learning from pixels with progressive nets," in *Conference on Robot Learning*. PMLR, 2017, pp. 262–270.
- [2] L. J. Lin, "Scaling up reinforcement learning for robot control," in *ICML*, 1993.
- [3] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1238–1274, 2013.
- [4] O. M. Andrychowicz, B. Baker, M. Chociej, R. Jozefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, *et al.*, "Learning dexterous in-hand manipulation," *The International Journal of Robotics Research*, vol. 39, no. 1, pp. 3–20, 2020.
- [5] I. Akkaya, M. Andrychowicz, M. Chociej, M. Litwin, B. McGrew, A. Petron, A. Paino, M. Plappert, G. Powell, R. Ribas, *et al.*, "Solving rubik's cube with a robot hand," *arXiv preprint arXiv:1910.07113*, 2019.
- [6] Y. Urakami, A. Hodgkinson, C. Carlin, R. Leu, L. Rigazio, and P. Abbeel, "Doorgym: A scalable door opening environment and baseline agent," *arXiv preprint arXiv:1908.01887*, 2019.
- [7] E. Valassakis, Z. Ding, and E. Johns, "Crossing the gap: A deep dive into zero-shot sim-to-real transfer for dynamics," *arXiv preprint arXiv:2008.06686*, 2020.
- [8] S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, and D. Quillen, "Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection," *The International Journal of Robotics Research*, vol. 37, no. 4-5, pp. 421–436, 2018.
- [9] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," *arXiv preprint arXiv:1606.01540*, 2016.
- [10] Y.-Y. Tsai, B. Xiao, E. Johns, and G.-Z. Yang, "Constrained space optimization and reinforcement learning for complex tasks," *IEEE Robotics and Automation Letters (RA-L)*, vol. 5, no. 2, 2020.
- [11] G. Garcia-Hernando, E. Johns, and T.-K. Kim, "Physics-based dexterous manipulations with estimated hand poses and residual reinforcement learning," in *International Conference on Intelligent Robots and Systems (IROS)*, 2020.
- [12] 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.
- [13] A. Allevato, E. S. Short, M. Pryor, and A. Thomaz, "Tunenet: One-shot residual tuning for system identification and sim-to-real robot task transfer," in *Conference on Robot Learning*, 2020, pp. 445–455.
- [14] R. Jeong, J. Kay, F. Romano, T. Lampe, T. Rothörl, A. Abdolmaleki, T. Erez, Y. Tassa, and F. Nori, "Modelling generalized forces with reinforcement learning for sim-to-real transfer," *arXiv preprint arXiv:1910.09471*, 2019.
- [15] J. Liang, S. Saxena, and O. Kroemer, "Learning active task-oriented exploration policies for bridging the sim-to-real gap," *arXiv preprint arXiv:2006.01952*, 2020.
- [16] M. Kaspar, J. D. M. Osorio, and J. Bock, "Sim2real transfer for reinforcement learning without dynamics randomization," *arXiv preprint arXiv:2002.11635*, 2020.
- [17] D. Schafroth, C. Bermes, S. Bouabdallah, and R. Siegwart, "Modeling, system identification and robust control of a coaxial micro helicopter," *Control Engineering Practice*, vol. 18, no. 7, pp. 700–711, 2010.
- [18] J. Tan, Z. Xie, B. Boots, and C. K. Liu, "Simulation-based design of dynamic controllers for humanoid balancing," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016, pp. 2729–2736.
- [19] A. Farchy, S. Barrett, P. MacAlpine, and P. Stone, "Humanoid robots learning to walk faster: From the real world to simulation and back," in *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, 2013, pp. 39–46.
- [20] W. Yu, V. C. Kumar, G. Turk, and C. K. Liu, "Sim-to-real transfer for biped locomotion," *arXiv preprint arXiv:1903.01390*, 2019.
- [21] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 23–30.
- [22] S. Zhu, D. Surovik, K. Bekris, and A. Boularias, "Efficient model identification for tensegrity locomotion," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 2985–2990.
- [23] N. Fazeli, R. Tedrake, and A. Rodriguez, "Identifiability analysis of planar rigid-body frictional contact," in *Robotics Research*. Springer, 2018, pp. 665–682.
- [24] 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)*. IEEE, 2018, pp. 1–8.
- [25] S. James, A. J. Davison, and E. Johns, "Transferring end-to-end visuomotor control from simulation to real world for a multi-stage task," *arXiv preprint arXiv:1707.02267*, 2017.
- [26] Y. Chebotar, A. Handa, V. Makoviychuk, M. Macklin, J. Issac, N. Ratliff, and D. Fox, "Closing the sim-to-real loop: Adapting simulation randomization with real world experience," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 8973–8979.
- [27] F. Ramos, R. C. Possas, and D. Fox, "Bayessim: adaptive domain randomization via probabilistic inference for robotics simulators," *arXiv preprint arXiv:1906.01728*, 2019.
- [28] R. Alghonaim and E. Johns, "Benchmarking domain randomisation for visual sim-to-real transfer," *arXiv preprint arXiv:2011.07112*, 2020.
- [29] M. A. Beaumont, W. Zhang, and D. J. Balding, "Approximate bayesian computation in population genetics," *Genetics*, vol. 162, no. 4, pp. 2025–2035, 2002.
- [30] J. Peters, K. Mülling, and Y. Altun, "Relative entropy policy search," in *AAAI*, vol. 10. Atlanta, 2010, pp. 1607–1612.
- [31] F. Muratore, C. Eilers, M. Gienger, and J. Peters, "Bayesian domain randomization for sim-to-real transfer," *arXiv preprint arXiv:2003.02471*, 2020.
- [32] N. Hansen, "The cma evolution strategy: a comparing review," in *Towards a new evolutionary computation*. Springer, 2006, pp. 75–102.
- [33] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.
- [34] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 5026–5033.