Domain Randomization for Robust, Affordable and Effective Closed-loop Control of Soft Robots

Domain Randomization for Robust, Affordable and Effective Closed-loop Control of Soft Robots

Gabriele Tiboni[1,†], Andrea Protopapa[1,†], Tatiana Tommasi[1], and Giuseppe Averta[1]

***Abstract*— Soft robots are gaining popularity thanks to their intrinsic safety to contacts and adaptability. However, the potentially infinite number of Degrees of Freedom makes their modeling a daunting task, and in many cases only an approximated description is available. This challenge makes reinforcement learning (RL) based approaches inefficient when deployed on a realistic scenario, due to the large domain gap between models and the real platform. In this work, we demonstrate, for the first time, how Domain Randomization (DR) can solve this problem by enhancing RL policies for soft robots with: i) robustness w.r.t. unknown dynamics parameters; ii) reduced training times by exploiting drastically simpler dynamic models for learning; iii) better environment exploration, which can lead to exploitation of environmental constraints for optimal performance. Moreover, we introduce a novel algorithmic extension to previous adaptive domain randomization methods for the automatic inference of dynamics parameters for deformable objects. We provide an extensive evaluation in simulation on four different tasks and two soft robot designs, opening interesting perspectives for future research on Reinforcement Learning for closed-loop soft robot control.**

I. INTRODUCTION

Soft robotics is a rapidly developing field that has the potential to revolutionize how robots interact with their environment [1]. Unlike their rigid counterparts, soft robots are made from materials that can deform and adapt to their surroundings, enabling them to perform novel and unprecedented tasks in fields such as healthcare [2] and exploration [3]. However, controlling continuous soft robots is a challenging task: an accurate kinematic model would require infinite degrees of freedom (DoF) [4], and their highly nonlinear dynamics are difficult to model realistically [5]. In addition, novel actuation mechanisms have introduced further challenges to approach optimal closed-loop control of soft robots. Popular designs include (1) cable-driven soft robots, which employ cables that can be extended or retracted to control the robot's shape and motion, or (2) pneumatic-based models, which rely on the pressurization of air chambers within the soft robot.

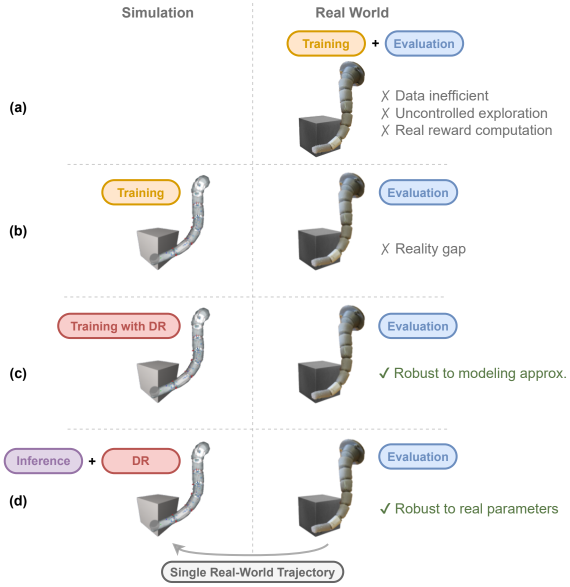


Fig. 1: Paradigms in RL-based robot learning: a) training directly on the real world; b) naïve Sim-to-Real transfer suffers from the reality gap; c) Training with domain randomization increases robustness to modelling approximations and errors; d) distributions over simulator dynamics parameters may be automatically inferred from real-world data for use with DR.

Many attempts have been made to control soft devices through model-based techniques, also pushed by the advancement of modelling techniques [6]. Yet, most complex tasks appear to be still unfeasible without the use of data-driven learning methods [7]. In particular, Reinforcement Learning has been showing promising results in recent years to learn effective closed-loop control policies for soft robots [8], [9], avoiding the need to know the exact dynamics of the system. Due to the notoriously low sample efficiency of current RL algorithms, these methods often rely on learned forward dynamics models [9], [10], or on simulated models to train a policy that can later be transferred to the real world for evaluation [11], [12] (See Fig. 1.b). The latter approach has been gaining more traction thanks to recent advances in accurate and efficient simulators for deformable objects [13], [14]. This line of research is commonly denoted as *Sim-to-Real Transfer*, which already demonstrated remarkable results for rigid robots in recent years [15], [16]. However, the performance of learned policies in the real-world often falls short of expectations due to the differences between the simulation and the real-world, namely policies are affected by the *reality gap*. This is further exacerbated by the

*This work was supported by Politecnico di Torino, CINECA, HPC@POLITO. This study was carried out within the project FAIR - Future Artificial Intelligence Research - and received funding from the European Union Next-GenerationEU (PIANO NAZIONALE DI RIPRESA E RESILIENZA (PNRR) – MISSIONE 4 COMPONENTE 2, INVESTIMENTO 1.3 – D.D. 1555 11/10/2022, PE00000013). This manuscript reflects only the authors' views and opinions, neither the European Union nor the European Commission can be considered responsible for them. †Gabriele Tiboni and Andrea Protopapa contributed equally to this work (corresponding author: Gabriele Tiboni, e-mail: gabriele.tiboni@polito.it)

[1] Politecnico di Torino, Turin, Italy first.last@polito.it

complexity of measuring deformable object parameters and designing accurate modeling, limiting current applications in soft robotics to overly simplified models for locomotion environments [17], [11] or simple trajectory tracking tasks [12]. Therefore, we identify the need to design novel sim-to-real transfer methods to scale the training of soft robots to more complex tasks, such as manipulation and contact-rich settings, and to overcome modeling approximations.

To bridge the reality gap, Domain Randomization (DR) has been proposed as a promising technique to learn transferable policies for rigid robotic systems [15], [16], [18]. DR involves training policies on simulated environments with randomized dynamics parameters sampled from a predefined distribution, promoting a policy that is robust to variations (see Fig. 1-c). Recent studies have also attempted to automatically estimate DR distributions [19], [20], [21], a method referred to as Adaptive Domain Randomization (ADR) (see Fig. 1-d). While DR and ADR have been successful in improving the transferability of policies for rigid robots, their effectiveness in soft robotics remains largely unexplored.

In this work, we present a thorough investigation of Domain Randomization in the context of closed-loop soft robot control. We include the examination of existing DR techniques for learning robust control policies on recently proposed soft robot benchmark environments [17], [22]—namely a reaching task for a trunk robot and walking for a MultiGait soft design. Moreover, we design a novel extension to state-of-the-art ADR methods and evaluate its capabilities to infer complex deformable object parameters. Finally, we propose two novel challenging manipulation setups in simulation for the cable-driven trunk robot (*pushing* and *lifting*) which we release to the public.

Our findings demonstrate that our method may accurately infer complex dynamics parameters such as Poisson's Ratios and Friction coefficients, and lead to policies that are robust to parameter discrepancies among domains. Furthermore, we discover that such policies can even be learned using simpler modeling approximations in simulation, yielding affordable and drastically reduced training time complexity. Finally, we test the capabilities of Domain Randomization to improve the task efficiency altogether, by acting as a regularization effect on the exploration of the environment. Interestingly, we notice that randomizing the surrounding environment allows the agent to find more effective strategies to solve the same task, exploiting task-specific environmental constraints.

II. RELATED WORK

A. Soft Robotics: modeling, actuation and simulation

As previously anticipated, describing continuous soft robots is a challenging task, because their modeling lays in the domain of continuum mechanics. Soft robots are actuated devices usually composed by viscoelastic material (such as silicone). Their dynamics, therefore, is regulated by infinite-dimensional Partial Differential Equations. Interestingly, recent works have demonstrated that finite-dimensional approximations of the robot's dynamics provide a reasonable trade-off between model tractability and accuracy [6]. The most popular designs implement either pneumatic or cable-driven actuation mechanisms [23]. The first consist of inflatable chambers which, when filled with air or fluids, may change their length, curvature and shape. The latter, instead, present cables or other extensive elements attached to specific points of the robot body. Cables are then actuated through external bodies, producing pushing or pulling forces on the insertion points. A proper combination of multiple actuation elements (i.e. different chambers or cables actuation schemes) can provide complex open loop robot behaviors.

Based on different modeling strategies (reviewed e.g. in [6], [23] and here omitted for the sake of space), several engines have been proposed so far to provide tools for an efficient and effective simulation of soft robotic devices. Among the others, it is worth mentioning SOFA [13], ChainQueen [24], Abaqus [25], which use volumetric FEM techniques, and Elastica [14], SimSOFT [26], and SoRoSim [27] that, instead, leverage on discretization of rod models.

B. Sim-to-Real Transfer with Domain Randomization

Domain randomization has become a popular approach for transferring learned policies from simulation to real-world hardware for *rigid* robotic systems [18]. Such approach has been widely investigated both for randomizing the visual appearance of the simulator [28], [29], [30], and its dynamics parameters [15], [16]. In these cases, the goal is to learn a policy that is invariant to changes in state space or transition dynamics, respectively. However, training a single policy to perform well on overly large variations of the environment may not always be possible, opening the challenge to design sensible posterior distributions over dynamics parameters [31], [32]. In [15], the authors proposed to use memory-based policies to allow the agent to readily adapt its behavior through implicit dynamics inference, while increasing complexity at training time. Alternatively, Adaptive Domain Randomization (ADR) methods attempt to automatically estimate DR distributions over dynamics parameters of interest, e.g. object masses and friction coefficients. Methods such as DROID [19], DROPO [21], and BayesSim [20] fall in the latter category, and have been recently shown to produce effective inference and promising policy transfer results for rigid robotic systems. These ADR methods move from the iterative-based online counterparts [33], as they don't interact with real hardware at optimization time and can make use of off-policy collected data—see [34] for reference on online vs. offline ADR approaches. Critically, DROID is confined to rigid robotics due to its reliability on joint torques measurements for position-controlled systems. Similarly, DROPO makes strict assumptions that prevent it from being applied to partially-observable environments—i.e. configurations of deformable bodies.

Despite recent advances, Domain Randomization has yet to be thoroughly investigated in the context of soft robotic systems. Two initial attempts are conducted by injecting random noise in the state observations [10], and more recently to both observations and policy actions [12]. However, although encouraged by the community [11], [17], no randomization of dynamics parameters for parametric physics engines currently exists. The same statement applies to the investigation of ADR methods for parameters inference of deformable bodies. We aim to fill this gap by sheding light on the novel challenges concerning the application of DR on infinite-DoF systems, providing evidence for its effectiveness and designing a novel algorithmic modification to DROPO to cope with partially-observable environments.

III. BACKGROUND

A. Reinforcement Learning

Consider a discrete-time dynamical system described by a Markov Decision Process (MDP) $\mathcal{M}$, with state space $\mathcal{S}$, action space $\mathcal{A}$, initial state distribution $\mu(s_0)$, transition dynamics probability distribution $\mathcal{P}(s_{t+1}|s_t, a_t)$ and reward function $r(s_t, a_t)$. At each time t, the *environment* $\mathcal{M}$ evolves according to the current state $s_t \in \mathcal{S}$ and action $a_t \in \mathcal{A}$ taken by an *agent*, i.e. the decision maker, with initial state drawn according to $\mu(s_0)$. Denote as $\pi_\theta(a_t|s_t)$ the stochastic *policy* used by the agent to interact with the environment, parameterized by θ. Under this formulation, Reinforcement Learning (RL) addresses the problem of finding an optimal policy $\pi^*_\theta(a|s)$ such as to maximize the expected (discounted) cumulative reward:

$$\pi^*_\theta = \arg\max_{\pi_\theta} \mathbb{E}_{\pi_\theta, \mathcal{P}, \mu} \left[\sum_{t=0}^{T} \gamma^t r(s_t, a_t) \right] \tag{1}$$

with discount factor $\gamma \in (0, 1]$. For complex problems with a continuous state space $\mathcal{S}$, the policy π_θ is in practice parameterized by a neural network θ, learned, e.g., through policy gradient RL algorithms such as Proximal Policy Optimization (PPO) [35].

B. Learning from randomized simulators

In the sim-to-real transfer paradigm a simulator is used as training environment, referred to as the *source domain* and denoted with $\mathcal{M}_s$. We assume $\mathcal{M}_s$ to share the same state space and action space of the real-world environment $\mathcal{M}_r$, noted as the *target domain*. In contrast, the source environment is further parameterized by its dynamics parameters $\xi \in \mathbb{R}^{n_\xi}$, such as masses, friction coefficients, and Poisson's ratio of deformable objects, which ultimately affect the source transition dynamics $\mathcal{P}_\xi$. Dynamics parameters can be generally assumed to be random variables that obey a parametric distribution $p_\phi(\xi)$, parameterized by ϕ. In a domain randomization setting, the agent's goal is therefore to learn a policy that maximizes (1) while acting in a randomized environment according to $p_\phi(\xi)$:

$$\pi^*_\theta = \arg\max_{\pi_\theta} \mathbb{E}_{\xi \sim p_\phi(\xi)} \left[\mathbb{E}_{\pi_\theta, \mathcal{P}_\xi, \mu} \left[\sum_{t=0}^{T} \gamma^t r(s_t, a_t) \right] \right] \tag{2}$$

In practice, DR may be easily integrated into existing RL algorithms by randomly sampling new dynamics parameters $\xi \sim p_\phi(\xi)$ at the beginning of each training episode.

C. Adaptive Domain Randomization

We consider here the *offline* Adaptive Domain Randomization paradigm, i.e. the more general case of ADR where no assumptions are made on how target domain data is collected for the inference phase—hence suitable to work with off-policy data or human demonstrations. In particular, let $\mathcal{D}$ be a dataset of state-action transitions $\mathcal{D} = \{(s_0, a_0, s_1), \dots, (s_T, a_T, s_{T+1})\}$, previously collected on the target domain. Under this formulation, ADR methods introduce an *inference phase* prior to policy optimization, that aims to find the optimal $p^*_\phi(\xi)$ domain randomization distribution given $\mathcal{D}$. Later, $p^*_\phi(\xi)$ is used to train a policy using DR with the objective in (2), which can ultimately be transferred to the target domain for evaluation.

IV. RESET-FREE DROPO

We describe our novel algorithmic modification to the current state-of-the-art offline ADR method DROPO [21], to cope with partial observability in soft robotic environments.

A. Method overview

By design, Domain Randomization Off-Policy Optimization (DROPO) relies on replaying real-world data in simulation, by resetting the simulator to each visited real-world state and executing the same corresponding action. On one hand, this allows the method to avoid the rising of compounding errors when replaying real trajectories [34]. On the other hand, this approach assumes that target observations encode full information on the configuration of the scene, such that the internal state of the simulator may be reset to each state. While this may be a reasonable assumption in rigid robotics, where robot state is typically observable at all times, it prevents the algorithm from being applied to partially-observable environments. In other words, a soft robot manipulator may only be tracked by a discrete number of points, but its exact infinite-DoF configuration is too complex to encode, hence impossible to recover in simulation. In this work, we relax this algorithmic assumption and develop an extension named *Reset-Free* (RF) *DROPO*. Our novel implementation draws inspiration from DROID [19], where only the initial full configuration of the environment is assumed to be known, and actions are replayed in open-loop during inference. In contrast to DROID, however, we retain the likelihood-based objective of DROPO which does not suffer from converging to point-estimates—as in [19]—due to inherent convergence properties of the evolutionary search algorithm CMA-ES [36]—see claims in [21], [34]. Finally, we introduce a novel regularization technique to allow our method to effectively deal with compounding errors and obtain informative likelihood estimates. Following the original pipeline, RF-DROPO ultimately consists of (1) a data collection phase where the dataset $\mathcal{D}$ is made available, (2) an inference phase where $p_\phi(\xi)$ is optimized to maximize the likelihood of real-world data to occur in simulation, and (3) policy training with the converged domain randomization distribution.

Algorithm 1: Reset-Free DROPO

```
input : Initialize $\phi$, sequence $NonDecrSeq$, $\tau_0 = 1$
output: Parameters $\phi^*$ of $p_{\phi^*}(\xi)$
Collect a dataset of transitions
  $D = \{(s_t, a_t, s_{t+1})\}_{t=0}^{T}$ from the target domain;
for $i = 0, \dots, M$ opt. iterations do
    Sample a set $\Lambda$ of $\min(\tau_i, L)$ transitions from $\mathcal{D}$;
    forall $(s_t, a_t, s_{t+1}) \in \Lambda$ do
        Estimate $S_{t+1}^{\phi} \sim \mathcal{P}_\phi(\cdot|s_0, a_{0,\dots,t}; \phi)$ through
          repeated sampling;
        Compute likelihood estimate of $s_{t+1}$ under
          $S_{t+1}^{\phi}$: $\mathcal{L}_t = \mathcal{P}_\phi(S_{t+1}^{\phi} = s_{t+1}|s_0, a_{0,\dots,t}; \phi)$
    end
    $\mathcal{L} = \sum_t \log \mathcal{L}_t$;
    Update $\phi$ towards maximizing $\mathcal{L}$;
    $\tau_{i+1} \leftarrow \min(T,\ NonDecrSeq(\tau_i))$;
end
Train a policy with DR using the converged $p_\phi^*(\xi)$ ;
```

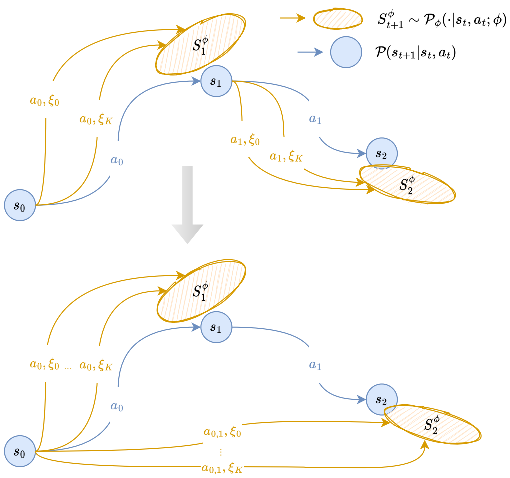


Fig. 2: Overview of (bottom) Reset-Free DROPO algorithm vs. (top) its original counterpart with intermediate state-resetting.

B. Implementation

1) Data collection: A fixed dataset $\mathcal{D}$ with target domain state-action transition shall be made available to run the inference phase. Such data may be collected with any desirable strategy, such as kinesthetic teaching or hard coded policies. Note that no reward computation in the real-world is needed, as the inference phase relies on state-action transition pairs only.

2) Dynamics parameter inference: Our objective follows the original DROPO implementation, as we seek to maximize the likelihood of real-world state transitions $s_{0,\dots,T}$ according to the simulation transition dynamics $\mathcal{P}_\xi$, under randomized parameters $\xi \sim p_\phi(\xi)$. Therefore, let S_{t+1}^{ϕ} be the random state variable distributed according to the new transition dynamics $\mathcal{P}_\phi(\cdot|s_t, a_t; \phi)$, with stochasticity induced by the domain randomization distribution $p_\phi(\xi)$. Our modification implies the avoidance of resetting the simulator state to each $s_t \in \mathcal{D}$. Instead, the probability of observing a target state s_t is estimated through the execution of all preceding actions $a_{0,\dots,t-1}$, starting from the initial known configuration s_0—assumed to be the same as in the real-world. Note how, for $t = 1$, RF-DROPO and DROPO coincide, whereas longer-horizon likelihood computations differ. An illustration of the proposed algorithmic modification is depicted in Fig. 2. Overall, RF-DROPO maximizes the following objective function, acting upon the DR distribution ϕ:

$$\phi^* = \arg\max_{\phi} \sum_{t=0}^{T} \log \mathcal{P}_\phi(S_{t+1}^{\phi} = s_{t+1}|s_0, a_{0,\dots,t}; \phi) \quad (3)$$

where the *log*-likelihood is considered for better numerical stability. In practice, the likelihood function—which models the relationship between dynamics parameters and state transition dynamics—is assumed to be unknown, as physics simulators act as black-box non-differentiable systems. The quantity in (3) is then estimated independently for each time step by sample estimates, i.e. repeatedly observing and inferring the sim dynamics distribution $\mathcal{P}_\phi$ for different values $\xi_k \sim p_\phi(\xi)$, as in the original implementation. It is worth noting that the likelihood computation for different timesteps t is still independent, assuming i.i.d. sampling of $\xi_k \sim p_\phi(\xi)$ for different time steps and noting that a function—simulator dynamics— of i.i.d. random variables still leads to statistically independent random variables. For the sake of simplicity, we stick to uncorrelated multivariate Gaussian distributions as parameterization for $p_\phi(\xi)$, with additive homoschedastic variance ϵ to all dimensions—See Sec. 3.4 in [21].

In the attempt to stabilize likelihood estimation for long-horizon time steps—which could suffer from compounding errors when executing multiple consecutive actions—we introduce a regularization temperature parameter $0 \leq \tau \leq T$. We then propose to limit the time-horizon of our objective function (3) to τ—instead of T—and set τ to gradually increase during each optimization iteration until finally reaching the full trajectory length T. Such addition allows the algorithm to focus on a smaller number of state transitions in the near-future at the beginning of the inference phase, guiding the process to optimize dynamics parameters that capture gradually further state-transitions. In principle, any non-decreasing sequence of τ may be adopted; throughout our experiments, we use an exponential schedule $\tau(i) = T\left(1 - e^{-i\ \cdot\ (2\log 10/M)}\right)$ for each opt. iteration $i = 0, \dots, M$, which reaches 90% of the total number of transitions T half-way through the process.

Finally, without loss of generality, we allow the algorithm to consider only a random sub-sample of L transitions at each CMA-ES iteration, among the total number of τ transitions considered within the current time horizon. Note how such extension draws inspiration from Stochastic Gradient Descent (GD) vs. vanilla GD, allowing to reduce inference time. The overall modifications of RF-DROPO to the original implementation are summarized in Algorithm 1.

3) Policy Training: The converged distribution $p_\phi^*(\xi)$ obtained through the inference phase may be finally used to

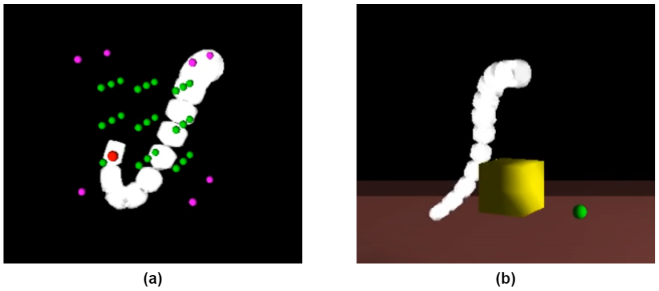


Fig. 3: TrunkReach and TrunkPush setups. a) Purple dots define the box of possible goal locations sampled at training time, green dots are 27 fixed target locations for evaluation, with the red dot being the current goal. b) Green dot is the desired target location for the box center of mass.

train a policy with Domain Randomization. We train our policies through Proximal Policy Optimization (PPO) [35] across all our experiments.

V. EXPERIMENTS

We carry out a thorough experimental evaluation in simulation on the effectiveness of Domain Randomization in the context of soft robotics. In particular, we aim at answering the following research questions:

- can RF-DROPO infer accurate posterior distributions over dynamics parameters for deformable objects?
- is Domain Randomization capable of efficiently transferring policies learned from simplified dynamics models, hence reducing training time and modeling complexity?
- can Domain Randomization lead to more effective strategies through better exploration, e.g. by exploiting environmental constraints?

A. Tasks

We test our method by building on recently introduced benchmark tasks in the domain of soft robotics for manipulation and locomotion [17]. In particular, we consider four evaluation domains for the purpose of our analysis: *TrunkReach*, *TrunkPush*, *TrunkLift*, *MultiGait*. The Trunk robot [22] is a cable-driven continuum deformable manipulator, depicted in Fig. 3. Its configuration is encoded in a 63-dimensional vector of positional keypoints along the robot, used as input observation vector for RL agents. The action space consists of a discrete set of 16 actions, encoding the extension or retraction of one of the 8 cables.

TABLE I: Posterior parameter estimation for the TrunkReach task. Search space is reported in $[min, max]$ format. Numerical values RF-DROPO and BayesSim (MDNN) are reported in mean and standard deviation ($\mu \pm \sigma$).

	Trunk Mass $[kg]$	**Poisson's Ratio**	**Young's Modulus** $[kg/(mm \cdot s)]$
Target	0.42	0.45	4500
Search space	[0.005, 1]	[0.4, 0.5]	[2000, 7000]
BayesSim	0.32 ± 1.28E-01	0.49 ± 2.30E-03	1914.25 ± 7.31E+06
RF-DROPO	0.64 ± 1.00E-05	0.45 ± 1.00E-05	6878.01 ± 2.77E-02

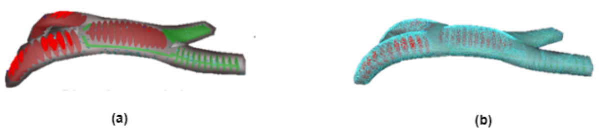


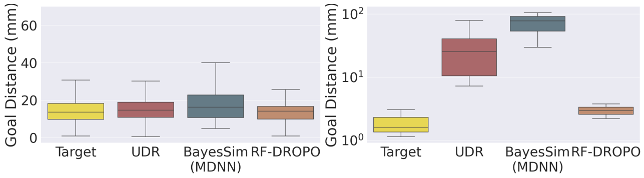
Fig. 4: Multigait locomotion environment: (a) simplified vs. (b) accurate simulation model.

First, we test the trunk's capabilities in a reaching task, where the agent's goal is to reach a randomly located point in space with the robot's endpoint (*TrunkReach*). Then, we design two novel manipulation settings to analyse the effects of DR in contact-rich settings: pushing a cube to a desired target location (*TrunkPush*), and lifting a flat object in the presence of a nearby wall (*TrunkLift*)—see Fig. 8. Finally, we consider the *multi-gait* design [22], a complex pneumatic soft robot tasked with walking forward (see Fig. 4). The latter setting introduces critical challenges by means of modeling approximations, which can drastically reduce the training time at the cost of higher modeling errors. In this context, we investigate the ability of DR to overcome modeling approximations and allow more affordable models to be used at training time. To do this, we instantiate two multi-gait models, the first (see 4-b) is the original detailed model, while the second (see 4-a) is a simplified version obtained through Model Order Reduction [22].

All DR-compatible benchmark tasks, together with an implementation of our method, are publicly available at `https://andreaprotopapa.github.io/dr-soro/`.

B. DR for robustness: parameter discrepancies

We compare the capabilities of RF-DROPO and the more established method BayesSim [20] to infer complex deformable object parameters in simulation. We consider the TrunkReach and TrunkPush tasks, and feed the respective methods with a single trajectory—the equivalent of 5 seconds in wall time—collected on each environment, by rolling out a semi-converged policy. In principle, collecting more data may increase the accuracy of the inference phase, but we found one trajectory to be sufficient while limiting the computational time for running RF-DROPO to roughly the same as policy optimization (≈48 hours on 20 CPU cores). It's also worth noting that target data in $\mathcal{D}$ does not need to contain high-reward state-transition pairs, but simply informative data to estimate the desired parameters. We present our inference results in Tab. I and II, respectively for the TrunkReach and TrunkPush tasks. In particular, we experiment with both MDNN and MDRFF features of BayesSim and report the best results for brevity—as uncorrelated approximation of the full Gaussian distributions. Noteworthy, although a Gaussian mixture model is returned by BayesSim, a single modality was often found to be dominant. In general, MDRFF provided less stable results with occasional unfeasible parameter values. Interestingly, RF-DROPO demonstrated accurate and reliable estimation of the Poisson's Ratio and Friction coefficient to impressive precision, whereas BayesSim fell shorter. On the other hand,



(a) TrunkReach (b) TrunkPush

Fig. 5: Vanilla parameter estimation: policy evaluation in terms of distance from the goal position (lower is better).

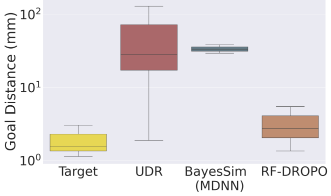


Fig. 6: Unmodeled parameter estimation setting for the TrunkPush task. Evaluation on the nominal target domain in terms of distance from the goal position (lower is better).

accurate inference of the Young's Modulus and respective masses proves to be more challenging in general, likely due to parameter correlations. Later, we investigate whether policies trained with Domain Randomization on the estimated posterior distributions may transfer well to the target domain with nominal parameter values. For each method and task, three policies are trained with three different repetitions of the inference phase, and finally evaluated (see Fig. 5). In addition, we compare the results with a Uniform Domain Randomization (UDR) baseline, which reflects the performance of 10 policies trained on uniform DR distributions randomly sampled from the search space in Tab. I and II. Notably, although the estimated parameters do not perfectly match, the performance of RF-DROPO on the target domain are still comparable to that of an oracle policy directly trained on it. On the other hand, BayesSim fails to transfer effectively in the TrunkPush task, likely due to a poorer posterior estimation—which is pivotal for contact-rich tasks. In particular, note how a random search over sensible uniform distributions (*i.e.* UDR) may produce well-performing policies for a simple reaching task, yet fails on average when contacts and more complex dynamics are involved. Finally, we design a more challenging *unmodelled setting* for the TrunkPush task where the Young's Modulus is misspecified by 80% and excluded from the estimation process of ADR methods. In this context, DR distributions should then be inferred to compensate for such misidentified parameter. The final performance of compared ADR methods is reported in Fig. 6, with their respective estimated posteriors in Tab. II. Similarly to the vanilla setting, we notice the performance gap between RF-DROPO and BayesSim, with the former consistently pushing the box about $8mm$ away from the target location on average over three repetitions.

C. DR for affordability: simplified vs. complex modeling

Due to their modeling complexities, directly training on accurate models of soft robots is not always feasible. This is the case for the MultiGait robot, where a full training procedure would take an estimated time of $\sim$ 55days (see Tab. IV) on 24 parallel CPU threads. For this reason, we investigate the effect of DR to overcome modeling approximations by training with a substantially simpler model and transferring the policy for evaluation to the more accurate MultiGait model. In particular, we vary the Poisson's Ratio and Young's Modulus of both its constituent materials, and its overall mass at training time. Note how no parameters discrepancy in principle exists between the two models, hence no inference is performed in this analysis. Therefore, domain randomization is applied by means of a random Gaussian noise applied to the nominal values, namely with a fixed $\sigma = \{0.005, 0.005, 500, 0.01, 10\}$ respectively for the mass, PDMS-PoissonRatio, PDMS-YoungModulus, EcoFlex-PoissonRatio, and EcoFlex-YoungModulus. We train policies on the simple model only, with both 6-timestep long episodes (i.e. the original version [17]) and 12-timestep long episodes, for 500k timesteps. The former version is heavily limited by the low cardinality of possible strategies[1] and could be simply solved by a brute-force search, failing to motivate the use of RL. We illustrate the results in Fig. 7, depicting the final displacement achieved by the robot when rolling out the policies in both the simplified (training) and complex (test) models—3 seeds per training setting. We notice that policies trained with DR converged to the same strategy as non-DR policies in the training environment (vanilla setting). More importantly, we observe that DR led to significantly more robust policy behaviors on the complex model, achieving either on par or better average return. The performance gap is exacerbated when testing under noisy observations (5 rollouts per seed), as non-DR policies would sometimes even lead to a backward motion. Overall, we conclude that Domain Randomization for the MultiGait soft robot design may reduce the training time by 7.9x by leveraging approximated models for training, yet transfer the learned behavior well to more accurate models at evaluation time.

D. DR for effectiveness: environment exploitation

Motivated by the intrinsic compliance of soft robot bodies, we finally investigate the ability of RL policies to adapt to an unknown environment and explore it in an effective manner. In particular, our analysis revolves around the effects of training on randomized properties of the simulator which, besides allowing robustness to parameter discrepancies and modeling errors, may bias soft agents towards learning more effective strategies. Indeed, note how soft robots offer a much broader spectrum of potential solutions for completing a task, due to their infinite-DoF nature. For the sake of this analysis, we consider the TrunkLift setup as a representative example of a task which may be solved through different strategies—e.g. exerting a force on the object's longitudinal side or leverage the nearby wall to lift it up from the shorter side. Arguably, the latter strategy would allow more effective execution under different circumstances—such as different

[1] possible strategies $= 6^6 = 46656$, as only 6 discrete actions are available at each timestep.

TABLE II: Parameter estimation in the vanilla and unmodelled settings for the TrunkPush task.

			Cube Mass $[kg]$	**Friction Coefficient**	**Trunk Mass** $[kg]$	**Poisson's Ratio**	**Young's Modulus** $[kg/(mm \cdot s^2)]$
	Target		0.05	0.3	0.42	0.45	4500
	Search space	[min, max]	[0.005, 1]	[0.01, 1]	[0.005, 1]	[0.4, 0.5]	[2000, 7000]
Vanilla	**BayesSim**	$(\mu \pm \sigma)$	0.37 ± 9.43E-02	0.37 ± 1.07E-01	0.69 ± 1.29E-01	0.42 ± 3.50E-03	1855.10 ± 8.17E+06
	RF-DROPO	$(\mu \pm \sigma)$	0.06 ± 4.40E-03	0.30 ± 1.51E-03	0.52 ± 3.24E-03	0.45 ± 4.50E-04	5557.07 ± 2.45E+00
Unmodelled	**BayesSim**	$(\mu \pm \sigma)$	0.52 ± 8.96E-02	0.51 ± 8.29E-02	0.48 ± 7.47E-02	0.44 ± 9.00E-04	3600 (fixed -20%)
	RF-DROPO	$(\mu \pm \sigma)$	0.07 ± 1.01E-02	0.30 ± 1.70E-03	0.50 ± 1.10E-03	0.45 ± 8.00E-04	3600 (fixed -20%)

TABLE III: TrunkLift: policies trained with a randomized location of the wall learn to exploit environmental constraints and lift up the object's center of mass to a higher elevation.

Object Mass $[kg]$	**DR Wall**	**Environmental Exploitation**	**Final Object Pose**	**Elevation** $[mm]$ $(\mu \pm \sigma)$
0.1	**No**	No	Unchanged	2.34 ± 4.03
	Yes	Yes	Horizontal	7.96 ± 1.67
0.05	**No**	No	Horizontal	8.55 ± 1.27
	Yes	Yes	Vertical	12.07 ± 4.72

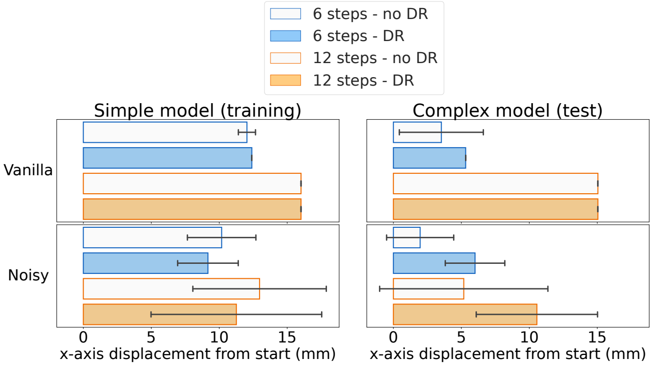


Fig. 7: Mean and stdev. displacement achieved by policies trained on the simplified MultiGait model. A mild Gaussian noise with $\sigma = \{0.5mm, 100Pa\}$ is optionally applied at test time respectively to the center of mass and the air pressure observations of the MultiGait, compared to the initial state values of $83mm$ and $3500Pa$.

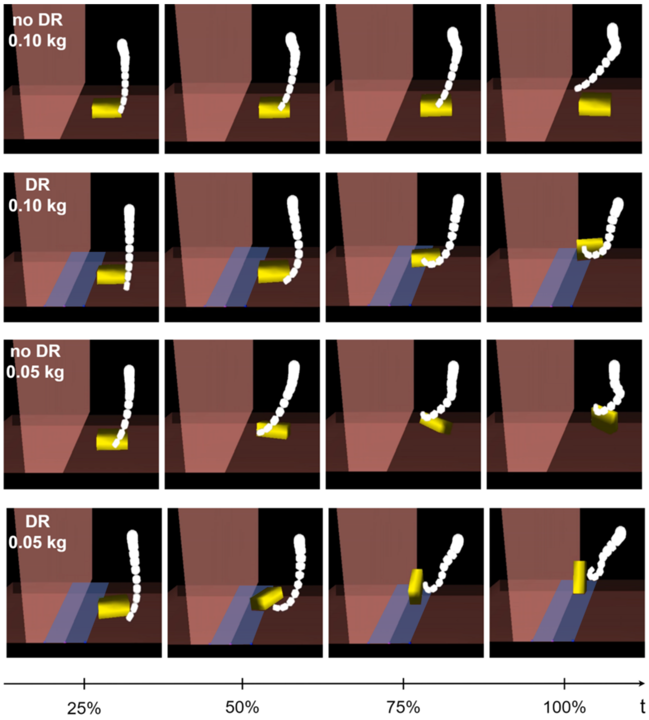


Fig. 8: TrunkLift: policies learned with a randomized position of the wall (blue shaded area) learn to exploit it to lift up the object reliably and reach higher average reward.

object masses and friction coefficients—and would require inferior effort. One could inject prior knowledge through reward shaping and bias policy search towards specified goals. Instead, we simply randomize the position of the wall at training time, *without* including it among the agent observations, to encourage a more effective exploration of the environment. Note how, in accordance with Sec. III-B, this effectively corresponds to an instance of Domain Randomization, forcing the agent to find a *robust* strategy in response to varying wall positions at the start of each episode—which are unknown to the agent. Notably, the uniform randomization range used (see blue shaded area in Fig. 8) does not affect the object at its initial location, even in the right-most case. A tabular description of the experimental analysis is reported in Tab. III, highlighting how the lift-up learned strategy is affected by the introduction of DR. In particular, we observe that for a heavier object mass ($0.1kg$), a feasible solution that does not exploit the wall may not be found, and only policies trained with DR reliably converged to strategies that leveraged the available environmental constraint—3 training repetitions. Note how all policies are tested with the wall location fixed to the nominal location, making it a fair comparison with non-DR policies. Intuitively, decreasing the mass of the object makes it easier to solve the task, and resulted in both settings (with vs. without DR) finding a way to lift up the object. However, the DR-trained policies converged to the most favorable strategy, as shown in Fig. 8.

TABLE IV: Training and test time complexity on the Multigait simple vs. complex model. Trainings are run on CPU-only, with 24 parallelized environments.

	Simple model	**Complex model**
Training (500k timesteps)	7 days	∼55 days
Test (12 timesteps)	1 min	8 minutes

VI. CONCLUSIONS

In this work, we provide evidence on how Domain Randomization can improve the robustness, affordability and effectiveness of closed-loop RL policies for soft robot control.

In particular, we propose a novel method to automatically infer posteriors over simulator dynamics parameters given a single trajectory collected on the target domain. We demonstrate that such distributions can be used to train control policies robust to parameter discrepancies and unmodeled phenomena (TrunkReach, TrunkPush). Additionally, we show that DR with a fixed Gaussian distribution can be used to learn on simplified dynamics models of deformable objects, with a drastic reduction in training time (MultiGait). Finally, we report evidence that randomizing the agent's surrounding environment may implicitly induce better exploration, and, e.g. lead to exploitation of environmental constraints for optimal performance (TrunkLift). Future directions of work should thoroughly analyse the deployment of DR-trained policies on real-world setups, closing the sim-to-real loop for soft robot control in contact-rich tasks.

References

[1] S. Kim, C. Laschi, and B. Trimmer, "Soft robotics: a bioinspired evolution in robotics," *Trends in biotechnology*, vol. 31, no. 5, pp. 287–294, 2013.

[2] M. Runciman, A. Darzi, and G. P. Mylonas, "Soft robotics in minimally invasive surgery," *Soft robotics*, vol. 6, no. 4, pp. 423–443, 2019.

[3] S. Aracri, F. Giorgio-Serchi, G. Suaria, M. E. Sayed, M. P. Nemitz, S. Mahon, and A. A. Stokes, "Soft robots for ocean exploration and offshore operations: A perspective," *Soft Robotics*, vol. 8, no. 6, pp. 625–639, 2021.

[4] M. Dubied, M. Y. Michelis, A. Spielberg, and R. K. Katzschmann, "Sim-to-Real for Soft Robots Using Differentiable FEM: Recipes for Meshing, Damping, and Actuation," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 5015–5022, 2022.

[5] F. Renda, M. Giorelli, M. Calisti, M. Cianchetti, and C. Laschi, "Dynamic model of a multibending soft robot arm driven by cables," *IEEE Transactions on Robotics*, vol. 30, no. 5, pp. 1109–1122, 2014.

[6] C. Della Santina, C. Duriez, and D. Rus, "Model based control of soft robots: A survey of the state of the art and open challenges," *arXiv preprint arXiv:2110.01358*, 2021.

[7] D. Kim, S.-H. Kim, T. Kim, B. B. Kang, M. Lee, W. Park, S. Ku, D. Kim, J. Kwon, H. Lee, *et al.*, "Review of machine learning methods in soft robotics," *Plos one*, vol. 16, no. 2, p. e0246102, 2021.

[8] S. Bhagat, H. Banerjee, Z. T. Ho Tse, and H. Ren, "Deep Reinforcement Learning for Soft, Flexible Robots: Brief Review with Impending Challenges," *Robotics*, vol. 8, no. 1, p. 4, Mar. 2019.

[9] T. G. Thuruthel, E. Falotico, F. Renda, and C. Laschi, "Model-based reinforcement learning for closed-loop dynamic control of soft robotic manipulators," *IEEE Transactions on Robotics*, vol. 35, no. 1, pp. 124–134, 2019.

[10] A. Centurelli, L. Arleo, A. Rizzo, S. Tolu, C. Laschi, and E. Falotico, "Closed-loop dynamic control of a soft manipulator using deep reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 4741–4748, 2022.

[11] C. Schaff, A. Sedal, and M. R. Walter, "Soft robots learn to crawl: Jointly optimizing design and control with sim-to-real transfer," *arXiv preprint arXiv:2202.04575*, 2022.

[12] Y. Li, X. Wang, and K.-W. Kwok, "Towards Adaptive Continuous Control of Soft Robotic Manipulator using Reinforcement Learning," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2022, pp. 7074–7081, iSSN: 2153-0866.

[13] E. Coevoet, T. Morales-Bieze, F. Largilliere, Z. Zhang, M. Thieffry, M. Sanz-Lopez, B. Carrez, D. Marchal, O. Goury, J. Dequidt, *et al.*, "Software toolkit for modeling, simulation, and control of soft robots," *Advanced Robotics*, vol. 31, no. 22, pp. 1208–1224, 2017.

[14] N. Naughton, J. Sun, A. Tekinalp, T. Parthasarathy, G. Chowdhary, and M. Gazzola, "Elastica: A compliant mechanics environment for soft robotic control," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3389–3396, 2021.

[15] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Sim-to-real transfer of robotic control with dynamics randomization," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 3803–3810.

[16] J. Tan, T. Zhang, E. Coumans, A. Iscen, Y. Bai, D. Hafner, S. Bohez, and V. Vanhoucke, "Sim-to-real: Learning agile locomotion for quadruped robots," *arXiv preprint arXiv:1804.10332*, 2018.

[17] P. Schegg, E. Ménager, E. Khairallah, D. Marchal, J. Dequidt, P. Preux, and C. Duriez, "Sofagym: An open platform for reinforcement learning based on soft robot simulations," *Soft Robotics*, 2022.

[18] W. Zhao, J. P. Queralta, and T. Westerlund, "Sim-to-real transfer in deep reinforcement learning for robotics: a survey," in *2020 IEEE symposium series on computational intelligence (SSCI)*. IEEE, 2020, pp. 737–744.

[19] Y.-Y. Tsai, H. Xu, Z. Ding, C. Zhang, E. Johns, and B. Huang, "DROID: Minimizing the Reality Gap Using Single-Shot Human Demonstration," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3168–3175, 2021.

[20] F. Ramos, R. C. Possas, and D. Fox, "Bayessim: adaptive domain randomization via probabilistic inference for robotics simulators," *arXiv preprint arXiv:1906.01728*, 2019.

[21] G. Tiboni, K. Arndt, and V. Kyrki, "Dropo: Sim-to-real transfer with offline domain randomization," *Robotics and Autonomous Systems*, p. 104432, 2023.

[22] O. Goury and C. Duriez, "Fast, Generic, and Reliable Control and Simulation of Soft Robots Using Model Order Reduction," *IEEE Transactions on Robotics*, vol. 34, no. 6, pp. 1565–1576, Dec. 2018.

[23] P. Schegg and C. Duriez, "Review on generic methods for mechanical modeling, simulation and control of soft robots," *Plos one*, vol. 17, no. 1, p. e0251059, 2022.

[24] Y. Hu, J. Liu, A. Spielberg, J. B. Tenenbaum, W. T. Freeman, J. Wu, D. Rus, and W. Matusik, "Chainqueen: A real-time differentiable physical simulator for soft robotics," in *International Conference on Robotics and Automation (ICRA)*, 2019, pp. 6265–6271.

[25] Y. S. Narang, J. J. Vlassak, and R. D. Howe, "Mechanically versatile soft machines through laminar jamming," *Advanced Functional Materials*, vol. 28, no. 17, p. 1707136, 2018.

[26] S. Grazioso, G. Di Gironimo, and B. Siciliano, "A Geometrically Exact Model for Soft Continuum Robots: The Finite Element Deformation Space Formulation," *Soft robotics*, vol. 6, Nov. 2018.

[27] A. T. Mathew, I. M. B. Hmida, C. Armanini, F. Boyer, and F. Renda, "Sorosim: A matlab toolbox for hybrid rigid-soft robots based on the geometric variable-strain approach," *IEEE Robotics & Automation Magazine*, pp. 2–18, 2022.

[28] 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)*, 2017, pp. 23–30.

[29] S. James, A. J. Davison, and E. Johns, "Transferring end-to-end visuomotor control from simulation to real world for a multi-stage task," in *Conference on Robot Learning*. PMLR, 2017, pp. 334–343.

[30] F. Sadeghi and S. Levine, "Cad2rl: Real single-image flight without a single real image," *arXiv preprint arXiv:1611.04201*, 2016.

[31] F. Muratore, F. Ramos, G. Turk, W. Yu, M. Gienger, and J. Peters, "Robot learning from randomized simulations: A review," *Frontiers in Robotics and AI*, p. 31, 2022.

[32] Q. Vuong, S. Vikram, H. Su, S. Gao, and H. I. Christensen, "How to pick the domain randomization parameters for sim-to-real transfer of reinforcement learning policies?" *arXiv preprint arXiv:1903.11774*, 2019.

[33] 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 *International Conference on Robotics and Automation (ICRA)*, 2019, pp. 8973–8979.

[34] G. Tiboni, K. Arndt, G. Averta, V. Kyrki, and T. Tommasi, "Online vs. offline adaptive domain randomization benchmark," in *Human-Friendly Robotics 2022*. Springer International Publishing, 2023, pp. 158–173.

[35] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[36] N. Hansen, "The cma evolution strategy: a comparing review," *Towards a new evolutionary computation: Advances in the estimation of distribution algorithms*, pp. 75–102, 2006.