Inherently Robust Control through Maximum-Entropy Learning-Based Rollout

Anonymous authors
Paper under double-blind review

Abstract

Reinforcement Learning has recently proven extremely successful in the context of robot control. One of the major reasons is massively parallel simulation in conjunction with controlling for the so-called "sim to real" gap: training on a distribution of environments, which is assumed to contain the real one, is sufficient for finding neural policies that successfully transfer from computer simulations to real robots. Often, this is accompanied by a layer of system identification during deployment to close the gap further. Still, the efficacy of these approaches hinges on reasonable simulation capabilities with an adequately rich task distribution containing the real environment. This work aims to provide a complementary solution in cases where the aforementioned criteria may prove challenging to satisfy. We combine two approaches, *maximum-entropy reinforcement learning* (MaxEntRL) and *rollout*, into an inherently robust control method called **Maximum-Entropy Learning-Based Rollout (MELRO)**. Both promise increased robustness and adaptability on their own. While MaxEntRL has been shown to be an adversarially-robust approach in disguise, rollout greatly improves over parametric models through an implicit Newton step on a model of the environment. We find that our approach works excellently in the vast majority of cases on both the Real World Reinforcement Learning (RWRL) benchmark and on our own environment perturbations of the popular DeepMind Control (DMC) suite, which move beyond simple parametric noise. We also show its success in "sim to real" transfer with the Franka Panda robot arm.

1 Introduction

Reinforcement Learning (RL) identifies optimal behavior strategies through an automated process of trial and error and constitutes one of today's standard approaches for addressing complex sequential decision-making problems. In particular, the application of RL to humanoid robots has achieved extremely impressive results. For example, they can now successfully navigate through complex terrain (Sun et al., 2025), dexterously manipulate real-world objects purely from vision (Lin et al., 2025), and even perform complex dynamic movements (Zhuang et al., 2024). In most cases, learning policies for such tasks rely on massively parallel simulations, given that training directly on hardware is impractical in nearly all instances: often, millions of interaction steps are required. However, training control policies exclusively in simulation frequently uncovers a phenomenon referred to as the "sim to real" gap: a policy that has been extensively optimized in simulation demonstrates degraded performance, or even complete failure, during real-world deployment.

One successful and commonly used approach is domain randomization (Tobin et al., 2017), where the policy is exposed to a wide range of simulated environments. Key simulation parameters are randomized during policy training, creating more diverse experiences and forcing policies to become robust to environmental variations. The real world is then assumed to be just another variation within that distribution. Fulfilling this assumption rests on two key desiderata: (i) all critical environment parameters and their ranges need to be carefully identified such that the real world is close or contained within; (ii) the space needs to be sufficiently tight to avoid overly conservative policies that need to compromise performance over too many different instances. Whereas the last point can be addressed in parts by domain identification, this process requires precise domain knowledge, extra engineering effort, and additional computing power to work well (Josifovski et al., 2022; 2024; Dulac-Arnold et al., 2021).

In this paper, we present an orthogonal approach to domain randomization tailored to instances where it is challenging to specify all key simulation parameters or too costly to exhaust all. To that end, we integrate *maximum-entropy reinforcement learning* (MaxEntRL) (Ziebart et al., 2008) and *rollout* (Tesauro, 1994; Bertsekas, 2024) into an inherently robust method called **M**aximum-**E**ntropy **L**earning-Based **Ro**llout (**MELRO**). We demonstrate that MELRO features and extends the strengths of both frameworks. While MaxEntRL enhances policy robustness by promoting exploratory behaviors, rollout compensates for residual uncertainties through real-time re-planning. We conduct extensive evaluations in simulation and in a “sim to real” transfer to demonstrate the adaptability and the robustness of our proposed approach. We benchmark its performance on the diverse Real World Reinforcement Learning (RWRL) suite tasks. Furthermore, we assess its capabilities on a custom set of challenging environment perturbations applied to the popular DeepMind Control (DMC) suite, which are designed to probe robustness beyond simple parametric noise. Finally, we present a successful “sim to real” transfer of MELRO on the Franka Panda robot arm.

2 Preliminaries

2.1 Problem

We assume that the problem consists of an unknown number of Markov Decision Processes (MDPs) (Bellmann, 1957) sharing the same action and state space. This can be formalized as a partially observable Markov Decision Process (POMDP) (Kaelbling et al., 1998), which is characterized by the tuple $(\mathcal{S}, \mathcal{A}, \Omega, p, r, o, p_0, \gamma)$, where $\mathcal{S}$, $\mathcal{A}$ and Ω are the state, action and observation spaces, $p : \mathcal{S} \times \mathcal{A} \mapsto \mathcal{S}$ is the transition function, $r : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$ is the reward function, $o : \mathcal{S} \mapsto \Omega$ is the observation function, $\gamma \in [0, 1]$ is a discount factor, and p_0 is the initial state distribution (Humplik et al., 2019). The state space $\mathcal{S}$ then consists of the state space of the MDPs, denoted as $S^{\mathcal{M}}$, and a set of task distributions $\mathcal{T}$. Initially, the task specification τ is sampled from $\mathcal{T}$ and maintained through time, assuming it is the only unobservable part of the state. This results in $S = (S^{\mathcal{M}} \times \mathcal{T})$, $\Omega = S^{\mathcal{M}}$ and $o((s, \tau)) \mapsto s$, where $s \in S^{\mathcal{M}}$. The transition and reward functions and the initial state distribution are also conditioned on τ. The goal is then to find a policy $\pi : \mathcal{S}^{\mathcal{M}} \mapsto \mathcal{A}$ that maximizes the expected discounted return under the sampled but unknown task specification τ:

$$\pi\left(a_k|s_k\right) = \arg\max_{a_k} \max_{a_{k+1:T}} \mathbb{E}_{\Gamma^\tau}\left[\sum_{t=k}^{T} \gamma^t r(s_t, \tau, a_t)\right], \tag{1}$$

where $\Gamma^\tau = s_{k:T}$ is a state trajectory where the states are sampled from the transition function of the MDP τ, and $k \leq T \in \mathbb{N}$.

2.2 Maximum-Entropy Reinforcement Learning

Maximum-Entropy Reinforcement Learning (MaxEntRL) extends traditional RL by augmenting Equation 1 with a conditional regularization term, the policy entropy $\mathcal{H}_\pi$:

$$\begin{aligned} \pi_{\text{MaxEnt}}(a_k|s_k) &= \arg\max_{a_k} \max_{a_{k+1:T}} \mathbb{E}_{\Gamma^\tau}\left[\sum_{t=k}^{T} \gamma^t r(s_t, \tau, a_t)\right] + \eta \mathcal{H}_\pi(a_k|s_k), \\ \mathcal{H}_\pi(a_k|s_k) &= -\int \pi(a_k|s_k) \log \pi(a_k|s_k)\, da_k, \end{aligned} \tag{2}$$

where $\eta > 0$ is a temperature parameter controlling the entropy trade-off and the entropy $\mathcal{H}_\pi(a_k|s_k)$ is approximated with Monte Carlo integration. Solving for Equation 2 results in stochastic policies characterized by non-zero probability for every action at each state. The entropy is higher when more actions lead to similar rewards and lower when one action is substantially better.

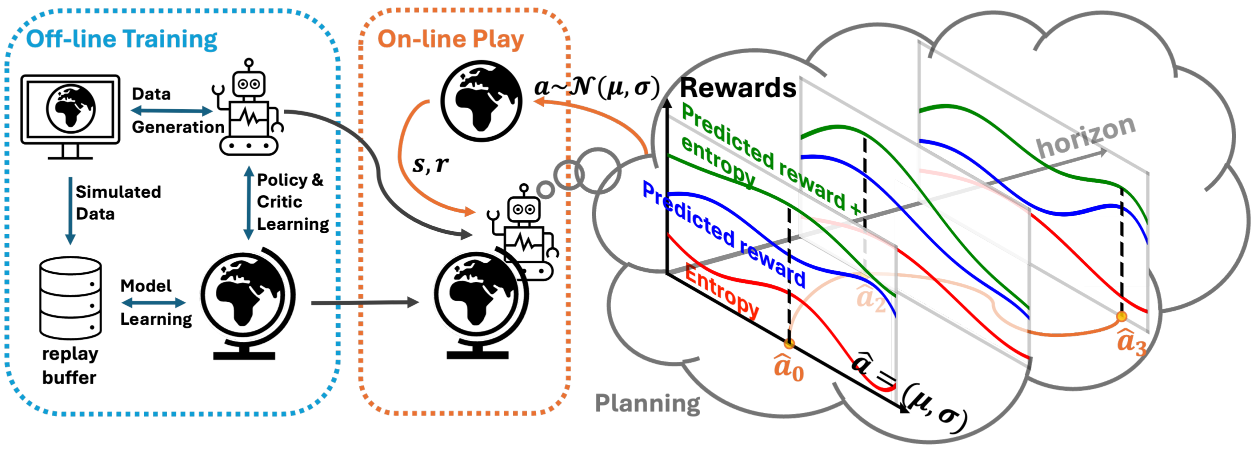


Figure 1: **Approach.** We propose a framework that uses both maximum-entropy reinforcement learning and rollout to achieve inherently robust control complementary to domain randomization. Our method incorporates maximum-entropy regularization during off-line training and on-line planning. To achieve this, the world model is trained on observations and action distribution parameters. Planning then occurs directly within the action distribution parameter space.

2.3 Learning-based Rollout

Rollout is a control method employing a dynamic system model to predict future behavior and optimize controls online. An elegant theoretical interpretation of rollout positions it as implementing a Newton step in value space, a perspective described by Bertsekas (2024). At each time step, rollout optimizes the controls for a given lookahead horizon $L \in \mathbb{N}$ to minimize the predicted future costs and applies the first optimized control in a receding-horizon fashion:

$$\pi_{\mathrm{r}}(a_k|s_k) = \arg\max_{a_k} \max_{a_{k+1:k+L-1}} \mathbb{E}_{\Gamma^\tau}\left[r(s_k,\tau,a_k) + \sum_{t=k+1}^{k+L-1} \gamma^{t-k} r(s_t,\tau,a_t) + \gamma^L R(s_{k+L}) \right], \tag{3}$$

where $R : \mathcal{S}^{\mathcal{M}} \mapsto \mathbb{R}^+$ is the terminal reward function estimating all future rewards and $\Gamma^\tau = s_{k:k+L}$. The rollout components, the dynamics model, reward, and terminal cost function are typically hand-engineered. Instead, we use multi-layer perceptrons (MLPs) whose parameters are learned from environmental interactions.

3 Maximum-Entropy Learning-based Rollout

MELRO uses the control framework introduced by Bertsekas (2022). It consists of two stages: *off-line training* and *on-line play*. The approach is visualized in Figure 1.

3.1 Off-line Training

In *off-line training*, we use a model-based reinforcement learning (MBRL) framework to train a world model, a base policy, and a critic.

Model Learning The model architecture and training are based on the variational state-space model used in Bayer et al. (2021), with the following modifications. Given that each individual task considered here is fully observable, the emission model directly propagates the states forward as the mean of a Gaussian with fixed variance:

$$p(x_t|s_t) = \mathcal{N}\left(x_t|\mu = s_t, \sigma_x^2 = 0.01\right). \tag{4}$$

Moreover, a cost-function head is introduced, which maps states to rewards:

$$r_{\theta_r}(r_t|s_t, \hat{a}_t) = \text{MLP}_{\theta_r}(s_t, \hat{a}_t), \tag{5}$$

where $\hat{a} = (\mu, \sigma)$ is the mean and the standard deviation of a Gaussian distribution. The transition is modeled as a Gaussian residual component where the output is a convex combination of the previous state s_{t-1} and an update $\hat{s}_{t-1}$. The mixing coefficient ρ and the update are outputs of a MLP with parameters θ_s:

$$p_{\theta_s}(s_t|s_{t-1}, \hat{a}_{t-1}) = \mathcal{N}\left(\mu = \text{RC}_{\theta_s}(s_{t-1}, \hat{a}_{t-1}), \sigma_s^2 = 0.001\right) \tag{6}$$

$$\text{RC}_{\theta_s}(s_{t-1}, \hat{a}_{t-1}) = \rho * s_{t-1} + (1-\rho) * \hat{s}_{t-1} \tag{7}$$

$$(\hat{\rho}, \hat{s}_{t-1}) = \text{MLP}_{\theta_s}(s_{t-1}, \hat{a}_{t-1}), \quad \rho = \frac{1}{2}(\texttt{softsign}(\hat{\rho}) + 1). \tag{8}$$

The inference model, parametrized by θ_q, is given by a backward-oriented GRU (Cho et al., 2014) model:

$$q(s_t|x_{1:T}, \hat{a}_{1:T}) = \text{GRU}_{\theta_q}(x_{1:T}, r_{1:T}, \hat{a}_{1:T})_t. \tag{9}$$

All model components are trained on a replay buffer containing all policy-environment interactions. This is achieved by optimizing the model parameters θ_r, θ_s, and θ_q such that the evidence lower bound (ELBO) is maximized.

Policy Learning The policy is modeled as a normal distribution, parametrized by ϕ, where a two-headed MLP estimates the mean and standard deviation:

$$\pi_\phi(a|s) = \mathcal{N}\left(a|(\mu, \sigma) = \text{MLP}_\phi(s)\right). \tag{10}$$

The critic $\hat{v}_\chi$ is modeled as a deterministic MLP with parameters χ. Both the policy and the critic use layer normalization (Ba et al., 2016). The critic loss $\mathcal{L}_{\hat{v}_\chi}$ is given by a supervised learning objective on TD(λ)-targets which are computed on imagined model and policy rollouts:

$$\mathcal{L}_{\hat{v}_\chi} = \mathbb{E}_{a_t \sim \pi_\phi}\left[\frac{1}{H_v}\sum_{t=0}^{H_v}\frac{1}{2}\left\|\hat{v}_\chi(s_t, a_t) - R(s_t)\right\|_2^2\right], \tag{11}$$

with TD(λ)-targets:

$$R(s_t) = r_{\theta_r}(s_t, a_t) + \eta\mathcal{H}_{\pi_\phi}(a_t|s_t) + \gamma\left((1-\lambda)\texttt{sg}\left(\hat{v}_\chi(s_t, a_t)\right) + \lambda R(s_{t+1})\right), \tag{12}$$

$$R(s_{t+H_v}) = \texttt{sg}\left(\hat{v}_\chi(s_{t+H_v}, a_{t+H_v})\right), \tag{13}$$

where $\texttt{sg}$ is the `stop-gradient` operator. The policy loss $\mathcal{L}_{\hat{\pi}_\phi}$ is given by the expected values of the TD(λ)-targets:

$$\mathcal{L}_{\pi_\phi} = \mathbb{E}_{a_t \sim \pi_\phi}\left[\sum_{t=0}^{H_\pi} R(s_t)\right]. \tag{14}$$

The optimal parameters for the policy and critic are obtained by maximizing the weighted sum of the policy and critic loss on model rollouts. Additionally, a regularization term consisting of the norm of the gradients of the policy and critic loss is introduced to improve the convergence of the optimization problem by increasing the conservativity (Mescheder et al., 2017). The policy-critic loss is then given by:

$$\mathcal{L}_{\pi_\phi, \hat{v}_\chi} = \zeta\mathcal{L}_{\pi_\phi} + \mathcal{L}_{\hat{v}_\chi} + \eta * \left(\frac{\left\|\nabla\mathcal{L}_{\pi_\phi}\right\|_2^2}{|\nabla\mathcal{L}_{\pi_\phi}|} + \frac{\left\|\nabla\mathcal{L}_{\hat{v}_\chi}\right\|_2^2}{|\nabla\mathcal{L}_{\hat{v}_\chi}|}\right), \tag{15}$$

where the loss is scaled by a weighting hyperparameter ζ and the conservativity regularization term by η. During training, the gradients of all components are clipped by the Euclidean norm, with a bound specified for each component, and zero-centered Gaussian noise is added to the actions to enhance exploration.

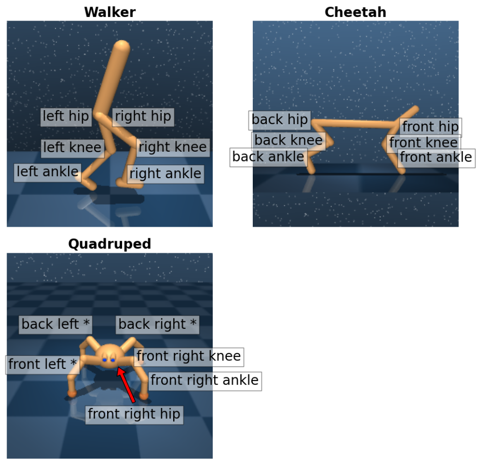


Figure 2: **Disabled joints tasks.** Visualization of the Walker, Cheetah, and Quadruped joints.

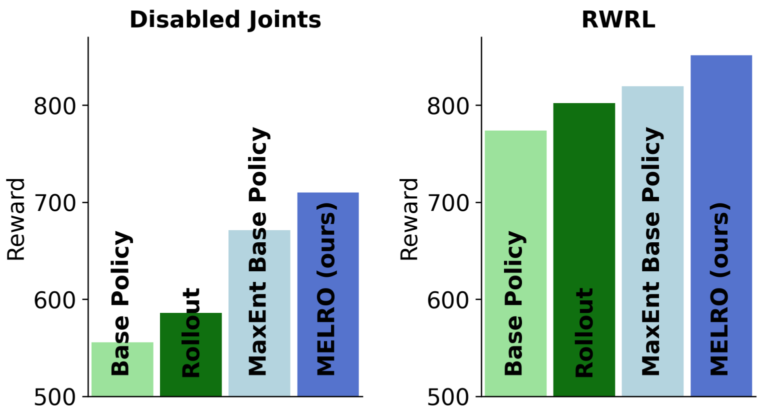


Figure 3: **Overview.** Average reward of all evaluated methods over all simulated environments and perturbations.

3.2 On-line Play

In *on-line play*, we combine the trained components into the rollout algorithm. We thereby improve standard rollout in the following ways. First, online planning happens in the space of action means and standard deviations of a Gaussian policy. Secondly, we generate imagined rollouts for the planning horizon based on the learned world model and the base policy. The resulting base policy action distribution parameters, i.e. means and standard deviations over time, are utilized as initial values for gradient descent. Thirdly, in an analogous manner to standard MaxEntRL, the rollout cost function, as defined in Equation 3, is regularized by the entropy of the action distribution. The idea behind this regularization is that it will lead to more robust actions being selected during planning. Furthermore, immediately following the model rollouts and before the application of the critic, a fixed number of base policy steps are performed as suggested by Bertsekas (2024). This truncated rollout with the base policy improves the stability property of the rollout policy at low additional computational costs.

The resulting rollout problem is then given by the following equations:

$$\begin{aligned}\pi_{\text{MELRO}}(\hat{a}_k|s_k) &= \arg\max_{\hat{a}_k} \max_{\hat{a}_{k+1},\dots,\hat{a}_{k+L},\hat{a}_{k+L+m}} \mathcal{L}_{\text{MELRO}},\\ \mathcal{L}_{\text{MELRO}} &= \mathbb{E}\left[r_{\theta_r}(s_k,a_k) + \sum_{i=k+1}^{k+L-1}\gamma^{i-k} r_{\theta_r}(s_i,a_i) + \tilde{R}_{k+L}\right] + \eta \sum_{j=k}^{k+L-1}\mathcal{H}_\pi(\hat{a}_j|s_j),\\ \tilde{R}_{k+L} &= \sum_{j=k+L}^{k+L+m-1} r_{\theta_r}(s_j,a_j) + \gamma^{k+L+m}\hat{v}_\chi(s_{k+L+m},a_{k+L+m}),\end{aligned} \tag{16}$$

where $\hat{a}_i = (\mu_i, \sigma_i)$, $a_i \sim \mathcal{N}(\mu_i, \sigma_i)\ \forall i \in \{k, k+1, \dots, k+L, a_k+L+m\}$, $a_{k+L:k+L+m-1}$ is sampled from the base policy π_ϕ, $s_{k:k+L+m}$ is sampled from the transition function p_{θ_s} and $\tilde{R}_{k+L}$ is the approximation of the future rewards starting from the time step $k+L$. The final controls are then computed in a receding-horizon fashion, optimizing Equation 16 using Adam (Kingma & Ba, 2015), where the search direction in the space of action means and standard deviations is generated using Augmented Random Search (ARS) (Mania et al., 2018).

4 Experiments

To evaluate the robustness and transfer capabilities, experiments were conducted both in simulation and on a real Franka Panda robot arm.

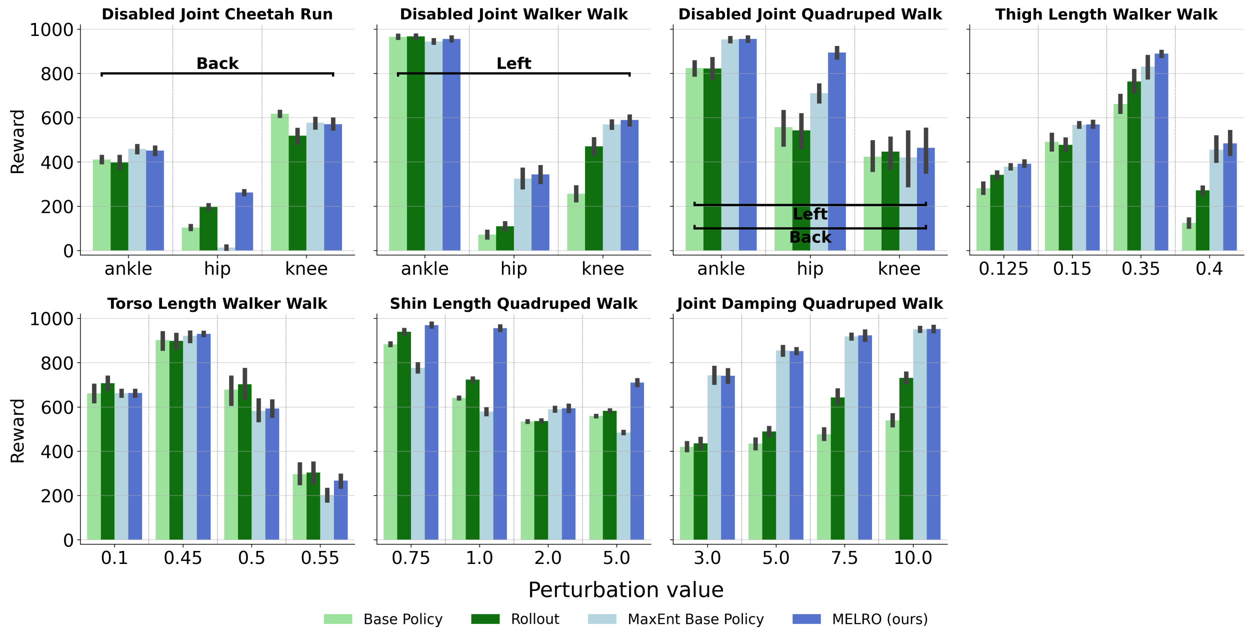


Figure 4: **Individual simulation results.** Average reward and 95%-CI for a subset of perturbations. The horizontal brackets indicate the positioning of the joints.

4.1 Simulation

We evaluated MELRO across four simulated environments from the DMC suite (Tassa et al., 2018), paired with five distinct perturbations. Four perturbation types - torso and thigh length for "Walker Walk", and shin length and joint damping for "Quadruped Walk" - have been adapted from the RWRL suite (Dulac-Arnold et al., 2020). The fifth perturbation type - disabled joints for "Walker Walk", "Quadruped Walk", and "Cheetah Run" - has been extended on Nagabandi et al. (2019). Even though the joints repeat for each extremity, we still noticed differences in the performance of all evaluated methods, so we disabled all the joints individually. This perturbation type introduces strong non-parametric disturbances, which move beyond simple noise. The joints for each environment are visualized in Figure 2.

We ablate MELRO against its MaxEnt base policy, the base policy trained without MaxEntRL and standard rollout over the latter. Both baseline policies and their corresponding dynamics models were initially trained on standard DMC environments. We then picked the best-performing model, policy, and critic combination and optimized the rollout parameters for each perturbation through hyperparameter search. Final performance metrics were obtained by assessing each methodology across all perturbation tasks through 10 independent random seed evaluations containing 10 independent rollouts.

The overall aggregated metrics are visualized in Figure 3, and the results for a subset of perturbations are shown in Figure 4 (Figures showing all the perturbations are in the appendix). In general, the MaxEnt policies produce superior results to the non-MaxEnt policies, and rollout enhances the performance of the respective base policy.

4.1.1 Improvement of Rollout over Neural Base Policies

The MaxEnt base policy already demonstrated robust performance across a broad spectrum of perturbations, achieving consistently high returns relative to the perturbation. Nevertheless, there were still some perturbations where the MaxEnt base policy exhibited catastrophic degradation in performance, highlighting inherent limitations in its generalization capacity (compare to Figure 4). In contrast, MELRO outperformed the MaxEnt base policy in almost all tested perturbation scenarios. Notably, for perturbations where the performance of the MaxEnt base policy deteriorated significantly, the MELRO algorithm demonstrated gen-

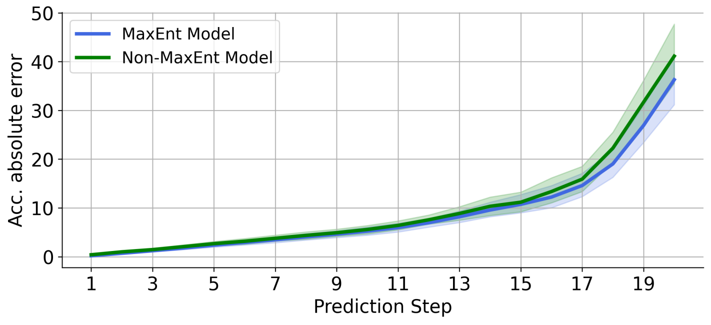


Figure 5: **Model Predictions.** Comparison of the absolute, accumulated error between the world model trained in a MaxEnt framework and the one without for all perturbations.

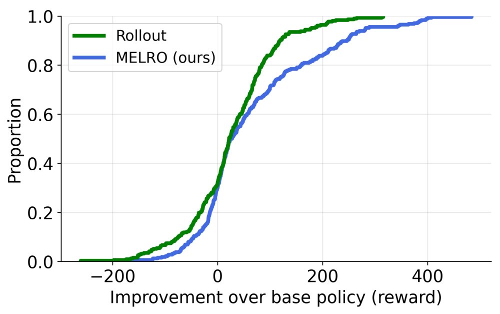


Figure 6: **Improvements.** ECDFs of reward improvements of MELRO and standard rollout relative to the respective base policy.

erally a considerable enhancement in its effectiveness in adapting to the new situation (compare, for example, to the disabled back hip joint of the Cheetah in Figure 4). Also, the standard rollout demonstrates a clear enhancement in performance when compared with the non-MaxEnt base policy.

4.1.2 Effects of Maximum-Entropy on World Models and Rollout

The training of a world model using MaxEntRL has already been demonstrated to enhance exploration, consequently leading to a more accurate world model (Ma et al., 2022). As demonstrated in Figure 5, this approach also results in a more robust model and lower prediction errors. In addition, the superior model produces superior control results. This is apparent from the more pronounced improvement of MELRO over its base policy, compared to the improvements shown by standard rollout (refer to Figure 6).

4.2 Franka Panda Robot Arm

We evaluated MELRO's "sim to real" capabilities by testing it on a simple reach task using the Franka Emika Panda robot arm. The objective was to maneuver the end-effector robot arm to a randomly selected position in front of the robot. To train the components in simulation, we modeled the task in MuJoCo (Todorov et al., 2012) using the Franka Panda MuJoCo Menagerie model (Zakka et al., 2022). The reward function has been selected as the negative sum of squares of the difference between the current end-effector position and the goal, minus a scaled penalty for rotating the end-effector out of a neutral orientation. It is important to note that no further reward shaping has been applied. Here, all policies - the base policy, rollout, MaxEnt base policy, and MELRO - have been trained to generate joint velocity actions given the current joint angles, joint velocities, and the current goal. Detailed information about the simulation is given in the appendix. The same training process as explained in Chapter 4.1 has been used. During deployment, both policies generated new joint velocity actions at a frequency of 10 Hz. These were then converted into joint torque commands using a PD-controller running at 1 kHz.

Although we evaluated multiple sets of hyperparameters for both the base policy and rollout, none of them produced stable and safe actions on the real robot. Since we did not apply extensive reward shaping, we believe that exploration for the non-MaxEnt components during the MBRL stage was insufficient. Consequently, MELRO was only evaluated against the MaxEnt base policy. The mean distance for each method over the 20 test runs on the real Franka Panda is shown in Figure 8. Additionally, Figure 9 visualizes the individual differences between MELRO and the MaxEnt base policy for each test trajectory. On average, MELRO was 11.7% closer to the goal than the MaxEnt base policy. This was achieved through faster convergence and generally bringing the end-effector closer to the goal.

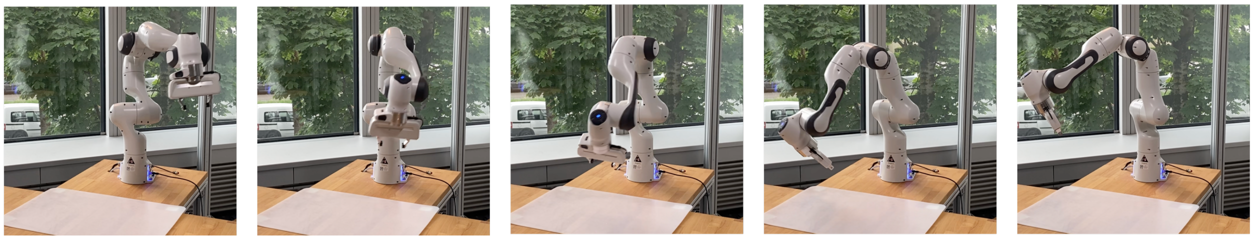


Figure 7: **Franka Trajectory.** Example of a real-world trajectory of the Franka Panda robot arm, controlled by MELRO. The movement took approximately five seconds.

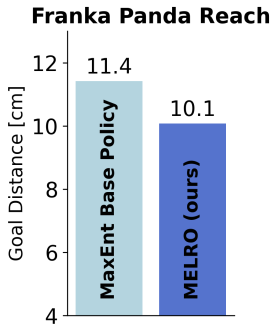


Figure 8: **Franka Panda.** Mean goal distances in cm of the MaxEnt base policy and MELRO on the real-world Franka Panda.

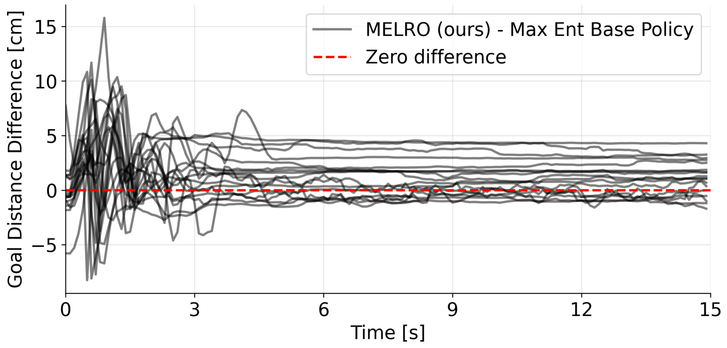


Figure 9: **Test trajectories.** Difference between MELRO and the MaxEnt base policy for all test runs in cm.

5 Related Work

5.1 Maximum-Entropy RL

It is well-known that MaxEntRL improves exploration (Haarnoja et al., 2017; Huang et al., 2020). This approach is also widely adopted, underpinning prominent model-free algorithms like SAC (Haarnoja et al., 2018), influencing implementations of methods like PPO (Schulman et al., 2017), and featuring in model-based approaches such as DreamerV3 (Hafner et al., 2023) and TD-MPC2 (Hansen et al., 2024). The primary distinction between the usage of MaxEntRL in MELRO and the aforementioned algorithms is that MELRO additionally incorporates maximum entropy regularization in the online planning objective.

Recent theoretical work established that MaxEntRL implicitly optimizes a lower bound on a robust RL objective, conferring inherent resilience against perturbations in dynamics and rewards without specialized robust optimization techniques (Eysenbach & Levine, 2022; Brekelmans et al., 2022). Despite this advantage, the resulting policies can sometimes be overly conservative, potentially sacrificing peak performance (Josifovski et al., 2022). Furthermore, standard MaxEntRL lacks explicit mechanisms for targeted adaptation, potentially hindering swift responses to rapidly evolving environments. Our work aims to address these limitations, harnessing MaxEntRL benefits while enabling more controlled robustness and faster adaptation, by adding online planning.

5.2 Robustness in Model-Based Reinforcement Learning

In model-based reinforcement learning (MBRL), model inaccuracies are commonly addressed by incorporating model uncertainty into the learning objective (Janner et al., 2019; Wang et al., 2024) or directly into the decision-making process (Chua et al., 2018; Wu et al., 2022). Epistemic uncertainty is typically estimated by ensemble disagreement but can also be predicted directly by the model. Although this method encourages the selection of actions that are more predictable and therefore likely to be more robust, it also leads to less exploration and, consequently, to suboptimal policies. In addition, all methods depend on accurate estimates of the uncertainty. Particularly for longer planning horizons, reduced performance and sampling efficiency have been observed as the inevitable model inaccuracies are interpreted as model uncertainty (Wang et al., 2024). Given that MELRO employs MaxEnt regularization, it is not subject to these limitations.

6 Conclusion

We introduced MELRO, a novel, inherently robust control method that combines maximum-entropy reinforcement learning (Haarnoja et al., 2018) with rollout, an online planning method (Tesauro, 1994). Instead of relying on extensive domain randomization to bridge the “sim to real” gap, we explored methods that are inherently more robust to such deviations from the model at hand. It is hence able to cope with variations that are unknown a priori. Experimentally, this approach significantly enhances system robustness against diverse and unforeseen dynamical perturbations and the success of “sim to real” transfers. Our results clearly confirm that rollout consistently improves performance and robustness across various scenarios, substantiating and extending the findings presented by Bertsekas (2024). Another appeal of this approach is its computational efficiency; it allows for the direct reuse of pre-trained world models, policies, and critics from standard model-based RL frameworks, without the necessity of retraining them from scratch. We believe that our work constitutes an important contribution to the research and deployment of autonomous agents that are robust to the unexpected.

References

Lei Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. *CoRR*, abs/1607.06450, 2016. URL http://arxiv.org/abs/1607.06450.

Justin Bayer, Maximilian Soelch, Atanas Mirchev, Baris Kayalibay, and Patrick van der Smagt. Mind the gap when conditioning amortised inference in sequential latent-variable models. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=a2gqxKDvYys.

Richard Bellmann. A markovian decision process. *Journal of Mathematics and Mechanics*, 6(5):679–684, 1957. ISSN 00959057, 19435274.

Dimitri Bertsekas. Newton's method for reinforcement learning and model predictive control. *Results in Control and Optimization*, 7:100121, 2022. ISSN 2666-7207. doi: https://doi.org/10.1016/j.rico.2022.100121. URL https://www.sciencedirect.com/science/article/pii/S2666720722000157.

Dimitri P. Bertsekas. Model predictive control and reinforcement learning: A unified framework based on dynamic programming. *CoRR*, abs/2406.00592, 2024. doi: 10.48550/ARXIV.2406.00592. URL https://doi.org/10.48550/arXiv.2406.00592.

Rob Brekelmans, Tim Genewein, Jordi Grau-Moya, Grégoire Delétang, Markus Kunesch, Shane Legg, and Pedro A. Ortega. Your policy regularizer is secretly an adversary. *Trans. Mach. Learn. Res.*, 2022, 2022. URL https://openreview.net/forum?id=berNQMTYWZ.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In Alessandro Moschitti, Bo Pang, and Walter Daelemans (eds.), *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pp. 1724–1734. ACL, 2014. doi: 10.3115/V1/D14-1179. URL https://doi.org/10.3115/v1/d14-1179.

Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pp. 4759–4770, 2018. URL https://proceedings.neurips.cc/paper/2018/hash/3de568f8597b94bda53149c7d7f5958c-Abstract.html.

Gabriel Dulac-Arnold, Nir Levine, Daniel J. Mankowitz, Jerry Li, Cosmin Paduraru, Sven Gowal, and Todd Hester. An empirical investigation of the challenges of real-world reinforcement learning. *CoRR*, abs/2003.11881, 2020. URL https://arxiv.org/abs/2003.11881.

Gabriel Dulac-Arnold, Nir Levine, Daniel J. Mankowitz, Jerry Li, Cosmin Paduraru, Sven Gowal, and Todd Hester. Challenges of real-world reinforcement learning: definitions, benchmarks and analysis. *Mach. Learn.*, 110(9):2419–2468, 2021. doi: 10.1007/S10994-021-05961-4. URL https://doi.org/10.1007/s10994-021-05961-4.

Benjamin Eysenbach and Sergey Levine. Maximum entropy RL (provably) solves some robust RL problems. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL https://openreview.net/forum?id=PtSAD3caaA2.

Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. Reinforcement learning with deep energy-based policies. In Doina Precup and Yee Whye Teh (eds.), *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1352–1361. PMLR, 2017. URL http://proceedings.mlr.press/v70/haarnoja17a.html.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In Jennifer G. Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1856–1865. PMLR, 2018. URL `http://proceedings.mlr.press/v80/haarnoja18b.html`.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy P. Lillicrap. Mastering diverse domains through world models. *CoRR*, abs/2301.04104, 2023. doi: 10.48550/ARXIV.2301.04104. URL `https://doi.org/10.48550/arXiv.2301.04104`.

Nicklas Hansen, Hao Su, and Xiaolong Wang. TD-MPC2: scalable, robust world models for continuous control. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL `https://openreview.net/forum?id=Oxh5CstDJU`.

Shiyu Huang, Hang Su, Jun Zhu, and Ting Chen. SVQN: sequential variational soft q-learning networks. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL `https://openreview.net/forum?id=r1xPh2VtPB`.

Jan Humplik, Alexandre Galashov, Leonard Hasenclever, Pedro A. Ortega, Yee Whye Teh, and Nicolas Heess. Meta reinforcement learning as task inference. *CoRR*, abs/1905.06424, 2019. URL `http://arxiv.org/abs/1905.06424`.

Michael Janner, Justin Fu andx Marvin Zhang, and Sergey Levine. When to trust your model: Model-based policy optimization. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 12498–12509, 2019. URL `https://proceedings.neurips.cc/paper/2019/hash/5faf461eff3099671ad63c6f3f094f7f-Abstract.html`.

Josip Josifovski, Mohammadhossein Malmir, Noah Klarmann, Bare Luka Zagar, Nicolás Navarro-Guerrero, and Alois C. Knoll. Analysis of randomization effects on sim2real transfer in reinforcement learning for robotic manipulation tasks. In *IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2022, Kyoto, Japan, October 23-27, 2022*, pp. 10193–10200. IEEE, 2022. doi: 10.1109/IROS47612.2022.9981951. URL `https://doi.org/10.1109/IROS47612.2022.9981951`.

Josip Josifovski, Sayantan Auddy, Mohammadhossein Malmir, Justus H. Piater, Alois Knoll, and Nicolás Navarro-Guerrero. Continual domain randomization. In *IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2024, Abu Dhabi, United Arab Emirates, October 14-18, 2024*, pp. 4965–4972. IEEE, 2024. doi: 10.1109/IROS58592.2024.10802060. URL `https://doi.org/10.1109/IROS58592.2024.10802060`.

Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Artif. Intell.*, 101(1-2):99–134, 1998. doi: 10.1016/S0004-3702(98)00023-X. URL `https://doi.org/10.1016/S0004-3702(98)00023-X`.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun (eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL `http://arxiv.org/abs/1412.6980`.

Toru Lin, Kartik Sachdev, Linxi Fan, Jitendra Malik, and Yuke Zhu. Sim-to-real reinforcement learning for vision-based dexterous manipulation on humanoids. *CoRR*, abs/2502.20396, 2025. doi: 10.48550/ARXIV.2502.20396. URL `https://doi.org/10.48550/arXiv.2502.20396`.

Hongying Ma, Wuyang Xue, Rendong Ying, and Peilin Liu. Maxent dreamer: Maximum entropy reinforcement learning with world model. In *International Joint Conference on Neural Networks, IJCNN 2022, Padua, Italy, July 18-23, 2022*, pp. 1–9. IEEE, 2022. doi: 10.1109/IJCNN55064.2022.9892381. URL `https://doi.org/10.1109/IJCNN55064.2022.9892381`.

Horia Mania, Aurelia Guy, and Benjamin Recht. Simple random search of static linear policies is competitive for reinforcement learning. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pp. 1805–1814, 2018. URL `https://proceedings.neurips.cc/paper/2018/hash/7634ea65a4e6d9041cfd3f7de18e334a-Abstract.html`.

Lars M. Mescheder, Sebastian Nowozin, and Andreas Geiger. The numerics of gans. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 1825–1835, 2017. URL `https://proceedings.neurips.cc/paper/2017/hash/4588e674d3f0faf985047d4c3f13ed0d-Abstract.html`.

Anusha Nagabandi, Ignasi Clavera, Simin Liu, Ronald S. Fearing, Pieter Abbeel, Sergey Levine, and Chelsea Finn. Learning to adapt in dynamic, real-world environments through meta-reinforcement learning. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL `https://openreview.net/forum?id=HyztsoC5Y7`.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017. URL `http://arxiv.org/abs/1707.06347`.

Wandong Sun, Baoshi Cao, Long Chen, Yongbo Su, Yang Liu, Zongwu Xie, and Hong Liu. Learning perceptive humanoid locomotion over challenging terrain. *CoRR*, abs/2503.00692, 2025. doi: 10.48550/ARXIV.2503.00692. URL `https://doi.org/10.48550/arXiv.2503.00692`.

Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, Timothy P. Lillicrap, and Martin A. Riedmiller. Deepmind control suite. *CoRR*, abs/1801.00690, 2018. URL `http://arxiv.org/abs/1801.00690`.

Gerald Tesauro. Td-gammon, a self-teaching backgammon program, achieves master-level play. *Neural Comput.*, 6(2):215–219, 1994. doi: 10.1162/NECO.1994.6.2.215. URL `https://doi.org/10.1162/neco.1994.6.2.215`.

Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter 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, Vancouver, BC, Canada, September 24-28, 2017*, pp. 23–30. IEEE, 2017. doi: 10.1109/IROS.2017.8202133. URL `https://doi.org/10.1109/IROS.2017.8202133`.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033. IEEE, 2012. doi: 10.1109/IROS.2012.6386109.

Xiyao Wang, Ruijie Zheng, Yanchao Sun, Ruonan Jia, Wichayaporn Wongkamjan, Huazhe Xu, and Furong Huang. Coplanner: Plan to roll out conservatively but to explore optimistically for model-based RL. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL `https://openreview.net/forum?id=jnFcKjtUPN`.

Zifan Wu, Chao Yu, Chen Chen, Jianye Hao, and Hankz Hankui Zhuo. Plan to predict: Learning an uncertainty-foreseeing model for model-based reinforcement learning. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022. URL `http://papers.nips.cc/paper_files/paper/2022/hash/65beb73449888fabcf601b3a3ef4b3a7-Abstract-Conference.html`.

Kevin Zakka, Yuval Tassa, and MuJoCo Menagerie Contributors. MuJoCo Menagerie: A collection of high-quality simulation models for MuJoCo, 2022. URL `http://github.com/google-deepmind/mujoco_menagerie`.

Ziwen Zhuang, Shenzhe Yao, and Hang Zhao. Humanoid parkour learning. In Pulkit Agrawal, Oliver Kroemer, and Wolfram Burgard (eds.), *Conference on Robot Learning, 6-9 November 2024, Munich, Germany*, volume 270 of *Proceedings of Machine Learning Research*, pp. 1975–1991. PMLR, 2024. URL `https://proceedings.mlr.press/v270/zhuang25a.html`.

Brian D. Ziebart, Andrew L. Maas, J. Andrew Bagnell, and Anind K. Dey. Maximum entropy inverse reinforcement learning. In Dieter Fox and Carla P. Gomes (eds.), *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, AAAI 2008, Chicago, Illinois, USA, July 13-17, 2008*, pp. 1433–1438. AAAI Press, 2008. URL `http://www.aaai.org/Library/AAAI/2008/aaai08-227.php`.

A MELRO in Detail

In Algorithm 1 and in Algorithm 2, we provide a detailed description of the off-line training and on-line play of MELRO.

Algorithm 1: MELRO (off-line training)

```
input : $\theta_s, \theta_r, \theta_q, \psi, \chi$: initial transition, reward, inference, policy and critic parameters
        $\omega, \gamma$: expected return weighting factor and discount factor
        $n_{init}, T_{init}$: number of samples, and time steps for model pre-training
        $n_t, T_{train}$: number of samples, and time steps for model training
        $D, \bar{D}$: optimization steps for model and for policy and critic
        $H_{policy}, T_{policy}$: number of samples and time steps for policy and critic training
        $\mathcal{B}$: replay buffer
        $env_{train}$: unperturbed training environment.
output: $p_{\theta_s}, r_{\theta_r}$ and $q_{\theta_q}$: learned transition, reward and inference functions
        $\pi_\psi, v_\chi$: learned neural base policy and critic
1  begin
2      // Pre-train model:
3      Sample $n_{init}$ trajectories $\{\Gamma_{init}\}_{1:n_{init}}$ each with length $T_{init}$ from $env_{train}$ and a random policy
4      $\mathcal{B} = \mathcal{B} \cup \{\Gamma_{init}\}_{1:n_{init}}$ // Add initial trajectories to buffer
5      while not converged do
6          // Train model:
7          $\{\Gamma\}_{1:H_{\text{model}}} \sim \mathcal{B}$ // Sample H trajectories from buffer
8          for $d = 1, \ldots, D$ do
9              Update model parameters using the ELBO and Adam
10         end
11         // Update policy and critic:
12         for $\bar{d} = 1, \ldots, \bar{D}$ do
13             Sample $H_{policy}$ trajectories $\{\Gamma\}_{1:H_{policy}}$ each with length $T_{policy}$ from $\hat{p}_\theta$ and $\hat{r}_\theta$ using $\pi_\theta$
14             Compute TD($\lambda$) returns $R$ using Equation 12 $\forall$t $\in \{1 : T_{policy}\}$ and $\forall$h $\in \{1 : H_{policy}\}$
15             Update policy and critic parameters using Equation 15 and Adam
16         end
17         Sample $n_t$ trajectories $\{\Gamma\}_{1:n_t}$ each with length $T_{train}$ from $env_{train}$ and $\pi_\psi$
18         $\mathcal{B} = \mathcal{B} \cup \{\Gamma\}_{1:n_t}$ // Add trajectories to buffer
19     end
20 end
```

Algorithm 2: MELRO (on-line play)

```
input  : $p_{\theta_s}, r_{\theta_r}$ and $q_{\theta_q}$: learned transition, reward and inference functions
         $\pi_\psi, v_\chi$: learned neural base policy and critic
         $\beta, D$: learning rate and optimization steps for MPC
         $env_{deploy}$: perturbed deploy environment.
output: $(a_1, ..., a_T)$: actions
1  begin
2      $s_0 = env_{deploy}.reset()$
3      for $t = 1, \dots, T$ do
4          Sample initial action mean and standard deviation trajectory $\Gamma_t^0 = (\hat{a}_t^0, \hat{a}_{t+1}^0, ..., \hat{a}_{t+H}^0)$ from $\pi_\psi$
            and $p_{\theta_s}$
5          Add small uniform noise to $\Gamma_t^0$.
6          for $d = 1, \dots, D$ do
7              Compute search direction $\Delta_{\Gamma_t^d}$ based on Equation 16 starting from $\Gamma_t^{d-1}$ using ARS, $r_{\theta_s}$ and
                $v_\chi$
8              Update action distribution parameter trajectory $\Gamma_t^d$ in the direction $\Delta_{\Gamma_t^d}$ using Adam
9          end
10         $a_t \sim \mathcal{N}((\mu, \sigma) = \hat{a}_t)$  // Sample action
11         $s_{t+1} = env_{deploy}.step(a_t)$
12     end
13 end
```

B Hyperparameter

In the following tables, we list all hyperparameters of all the methods used. Note that the entropy_weight for the base policy in Table 2 as well as for rollout in Table 5 was fixed to zero.

Table 1: Critic Parameters

Parameter	**Values**
max_iter	1000
activation	[`softsign`, `elu`]
n_hidden	[128, 256, 512, 1024]
n_layers	2
use_layer_norm	true

Table 2: Policy Parameters

Parameter	**Values**
exploration_noise	[0.0, 0.05, 0.1]
n_cells	8
step_size	[3e-4, 1e-4]
max_iter	[20, 40, 60, 100, 120]
activation	`softsign`
n_hidden	[64, 128, 256, 512]
n_layers	[2, 3]
use_layer_norm	true
discount	[0.95, 0.975, 0.99, 0.995]
entropy_weight	[0.01, 0.05, 0.1]
n_samples	[64, 128, 256]
n_time_steps	[4, 6, 8, 16, 24, 32]
td_lambda	[0.9, 0.95, 0.99]
max_grad_norm	[1, 10, 100]

Table 3: VRSSM Parameters

Parameter	**Values**
batch_size	[32, 64, 128, 256]
emission_scale	[0.01, 0.05, 0.1, 0.2]
max_grad_norm	[10, 100, 1000]
max_iter	[10, 20, 40, 100]
n_time_steps	4
step_size	[3e-3, 1e-3, 3e-4, 1e-4]
initial_iter	[100, 250, 500, 1000]
rnn_activation	`softsign`
rnn_n_hidden	[24, 64, 128]
mlp_activation	`softsign`
mlp_n_hidden	[128, 256, 512]
mlp_n_layers	[2, 3]

Table 4: MBRL Parameters

Parameter	**Values**
action__repeat	2
conservativity	[0.01, 0.05, 0.1]
critic__weight	[0.05, 0.1, 0.25, 0.5, 1.0]
n__iter__per__eval	1e6
n__steps__per__eval	10000
max__env__steps	3e6
n__initial__rollouts	1
n__rollouts__per__iter	1
n__time__steps	998
n__collect__steps	[10, 20, 40, 100]
n__initial__time__steps	[100, 500, 1000, 2500]

Table 5: Rollout Parameters

Parameter	**Values**
entropy__weight	[0.0001, 0.005, 0.01, 0.05, 0.1, 0.5]
n__base__policy__steps	[0, 1, 2, 4, 6, 8]
ars__n__perturbations	[8, 12, 16, 24]
ars__std	[0.05, 0.1, 0.15, 0.2, 0.25, 0.3]
ars__top__k	[2, 4, 8]
discount	[0.8, 0.9, 0.95, 0.999]
n__optimize__steps	[1, 2, 5, 10, 15, 20, 35, 45]
n__parallel__rollouts	1
n__planning__steps	[1, 2, 4, 6, 8, 12, 20]
optimizer	Adam
step__size	[0.001, 0.003, 0.01, 0.03, 0.1, 0.125, 0.15, 0.175, 0.2, 0.25]

C Franka Panda Robot Simulation Details

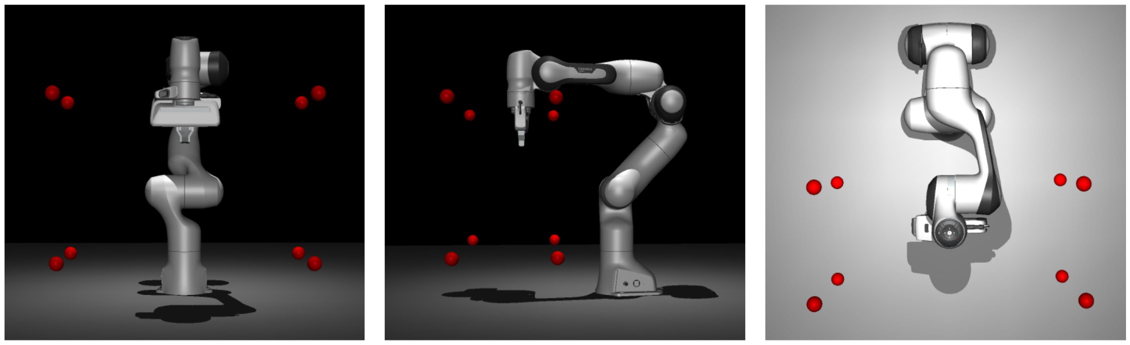


Figure 10: **Franka Reach Goal Region.** Visualization of the goal range in the MuJoCo simulation. The red dots mark the corners of the cube where the goals are sampled from.

As mentioned in the main text, the Franka Panda MuJoCo Menagerie model (Zakka et al., 2022) has been used to train the methods in simulation. The desired goal position g has been sampled for each trajectory uniformly from a cube relative to the base of the robot arm:

$$g \sim \mathcal{U}\left[\begin{pmatrix}0.2\\-0.35\\0.15\end{pmatrix}, \begin{pmatrix}0.5\\0.35\\0.6\end{pmatrix}\right]. \tag{17}$$

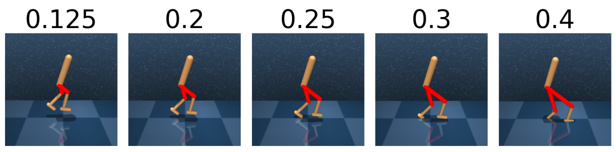
The borders of the box are defined in meters. 10 visualizes the goal cube in the simluation. Besides the goal position, a fixed straight downward-facing quaternion $g_q = (0, 1, 0, 0)$ has been used to penalize any other end-effector rotations. The reward for an end-effector position s and rotation s_q has then been computed as follows:

$$r(s, s_q, g, g_q) = -\sum_{i=1}^{3}(s_i - g_i)^2 - min\{|s_q - g_q|, |s_q + g_q|\}, \tag{18}$$

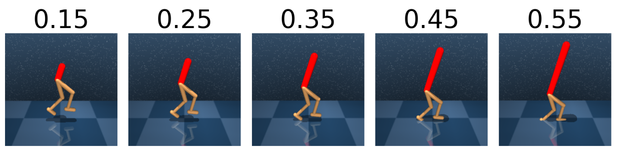
using the antipodal symmetry of quaternions.

D Additional Task Visualizations

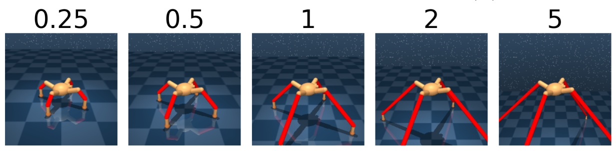
Figure 11 provides some additional task visualization of different perturbation types and values for the Real World Reinforcement Learning benchmark.



(a) Walker Walk thigh length



(b) Walker Walk torso length



(c) Quadruped Walk shin length

Figure 11: **RWRL tasks.** Visualization of some RWRL perturbations.

E Additional Experiments Results

In this chapter, we provide the complete results for all simulated experiments.

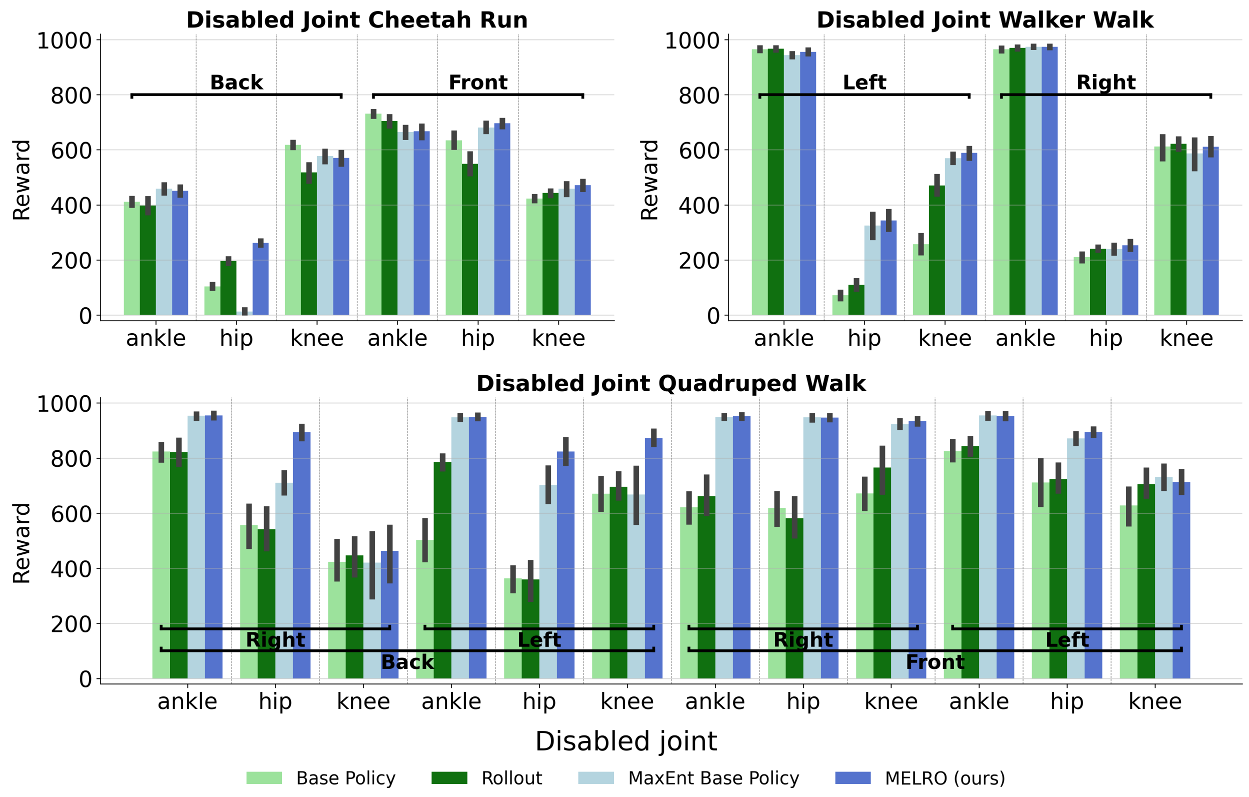


Figure 12: **Disabled joints results.** Average reward and 95%-CI for all disabled joint perturbations. The horizontal brackets indicate the positioning of the joints.

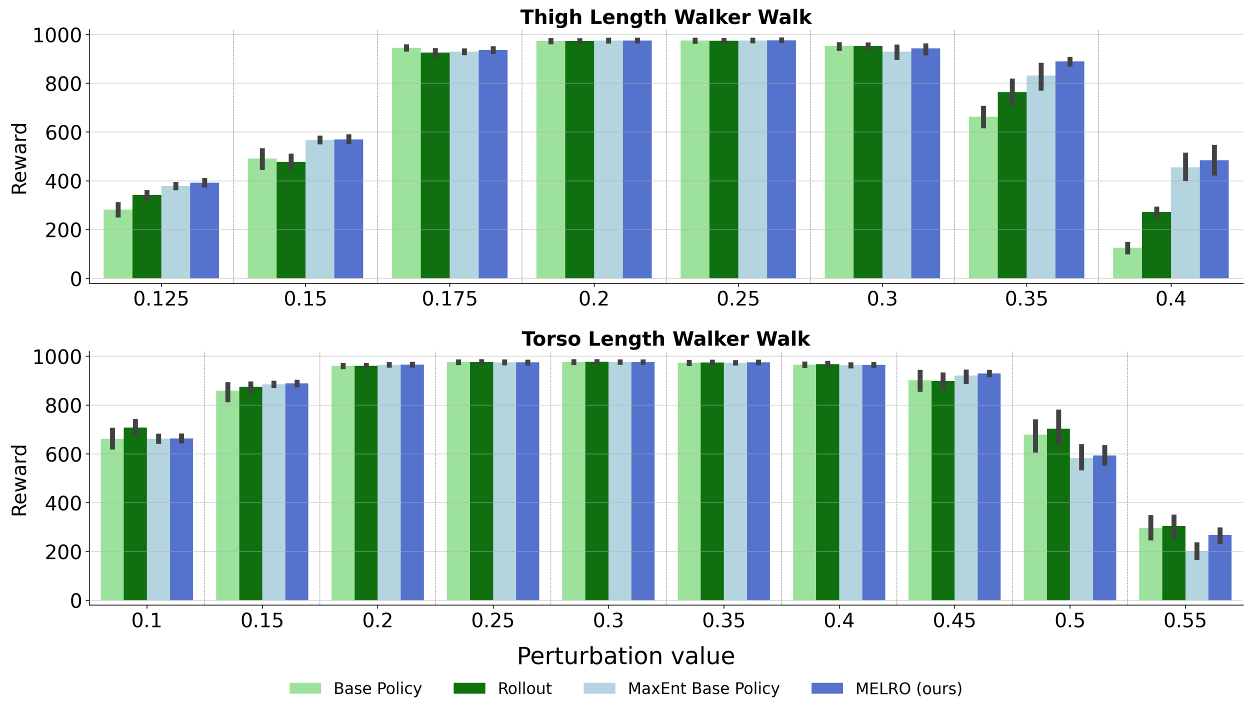


Figure 13: **Walker RWRL perturbations results.** Average reward and 95%-CI for all Walker RWRL perturbations.

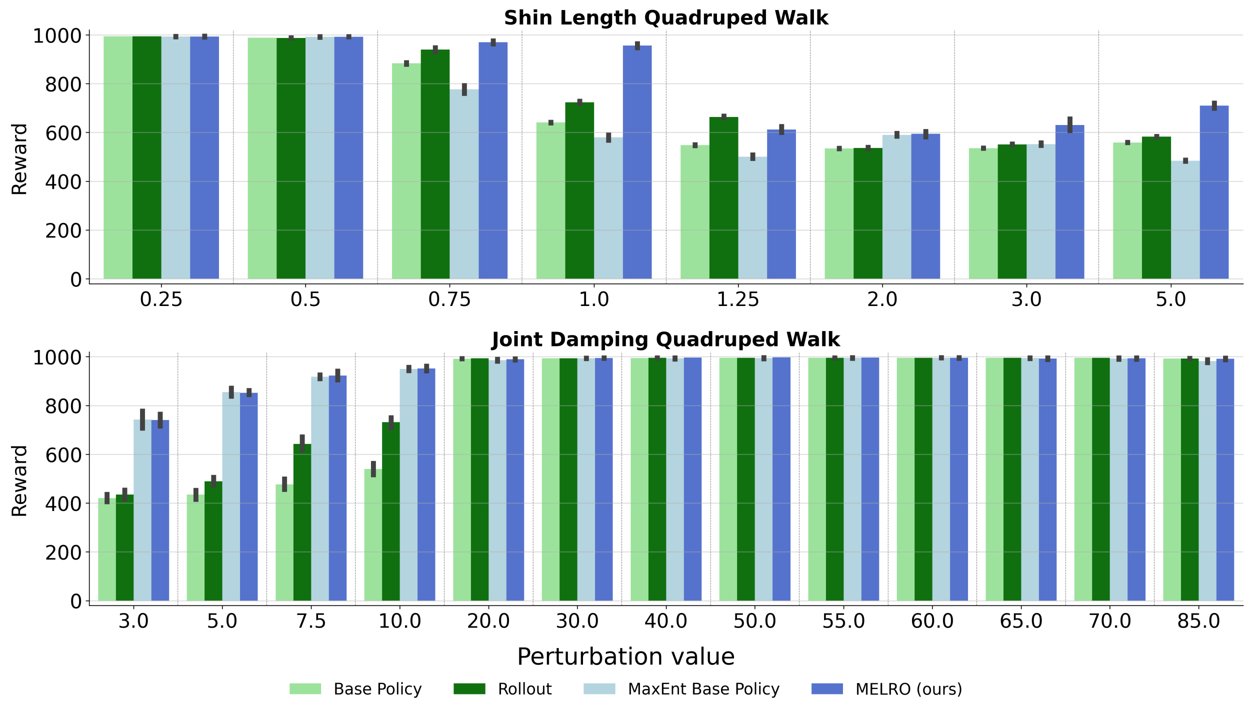


Figure 14: **Quadruped RWRL perturbations results.** Average reward and 95%-CI for all Quadruped RWRL perturbations.