Active Learning for Tuning Controller Parameters

MSc Thesis Topic

Mentors: Khaled Wahba, Sayantan Auddy
Examiners: Wolfgang Hönig

January 2026

1 Introduction

The goal of this project is to develop a systematic process for tuning controller gains to control robots in the real world. The typical workflow to do so begins in simulation: an initial guess for the controller gains is chosen and then optimized via a gradient-based [11] or black-box optimization approach. Once the controller performs satisfactorily in simulation, the gains are transferred to the real-world controller and manually tweaked until the desired level of control accuracy is obtained. This approach relies heavily on expert knowledge and trial-and-error, and a more systematic method for tuning gains is highly desirable.

We hypothesize that if we can obtain a simulator (i.e., a dynamics model) that closely resembles real-world dynamics, then optimizing the controller gains in simulation will produce gains that function effectively on the real-world controller, requiring only minor adjustments. For this, we propose an iterative process that begins by optimizing controller gains in simulation, uses active learning to select informative trajectories, collects real-world data along those trajectories and updates the simulation parameters using this data, and then repeats the cycle until convergence (see Algo. 1).

Many approaches exist for simulator parameter system identification (SI), and with the rise of reinforcement learning and simulation-based training, emphasis has shifted toward building accurate simulators to reduce the sim-to-real gap and enable deployment of simulation-trained policies. While methods such as Domain Randomization (DR) and Continual DR [14, 10, 2] focus on policy robustness and transfer rather than accurate system identification, approaches such as Active DR, Bayesian Adaptive DR, and SimOpt [7, 12, 4, 1] exploit real-world data for SI. A comprehensive overview of system identification in simulators can be found in [8]. More recently, active learning has also been utilized for SI [9]. However, in these works the focus was on training RL policies, whereas our work targets tuning controller gains, which is a much lower-dimensional optimization problem. Potential baselines we can consider in experiments include hand-tuned controllers (achieved through trial and error), RL policies, and controllers tuned using fixed trajectories rather than active learning.

The novelty of this work lies in combining controller gain optimization with system identification of simulator parameters, and particularly in developing an active learning strategy that generates unbiased trajectories to test the controller. A particular focus of this work can also be on efficiency, when compared to reinforcement learning-based workflows with a similar objective. Additionally, this approach can lead to a system that is interpretable and has provable safety guarantees.

2 Milestones and Approach

2.1 Milestones

0.a Use the differentiable simulator and a provided simulated trajectory and run the pipeline without the active learning part with fixed gains and only identify base diameter and wheel radius (initially, we can assume perfect state estimation, and in later milestones, work with imperfect estimation).

0.b Literature review: read key papers, understand the state-of-the-art, frame concrete research questions.

1. Become familiar with the codebase of typical SI workflows in sim-to-real (e.g. [7, 12, 4, 1]).
2. Take key decisions: (i) exact nature of the functions (ii) metrics for active learning data acquisition (iii) baselines (random noise, hand-designed trajectories, RL policies) (iv) metrics for comparing results (v) experimental setup and robot type (e.g., wheeled mobile robot, polulu).
3. Set up `sim1-sim2-sim1` experiments and run initial tests.
4. Set up `sim-real-sim` and run real experiments.
5. Repeat experiment for second robot type (optional).

2.2 Pseudocode

A rough outline of the proposed approach is provided below:

Algorithm 1 Active Learning for Controller Tuning

Given: Simulation with parameters θ, parameterized function $g_\gamma(\cdot)$ that outputs controller gains
Start with an initial guess for γ
while not converged **do**
  Use active learning to select a sequence of control inputs and desired states (possibly conditioned on γ):
  $u_{t_{1:T}}, s_{t_{1:T}} \leftarrow$ `ActiveLearningSelectTraj(`γ`)`
  Collect real-world data
  $\tau_{\text{real}} = (u^{\text{real}}_{t_{1:T}}, s^{\text{real}}_{t_{1:T}}) \leftarrow$ `ExecuteTrajReal(`$u_{t_{1:T}}, s_{t_{1:T}}, \gamma$`)` # $u^{\text{real}}_{t_{1:T}} = u_{t_{1:T}}$?
  while sim not converged **do**
    Collect simulation data
    $\tau_{\text{sim}} = (u^{\text{sim}}_{t_{1:T}}, s^{\text{real}}_{t_{1:T}}) \leftarrow$ `ExecuteTrajSim(`$u_{t_{1:T}}$`)` # $u^{\text{sim}}_{t_{1:T}} = u_{t_{1:T}}$?
    and update sim parameters θ using collected data $\mathcal{D} = \{\tau_{\text{real}}, \tau_{\text{sim}}\}$
    $\theta \leftarrow$ `UpdateSimParam(`$\mathcal{D}, \gamma, \theta$`)`
  end while
  Optimize γ in simulation with blackbox optimization or gradient-based method:
  $\gamma \leftarrow$ `OptGains(`$\mathcal{D}, \gamma, \theta$`)` `# Instead of a coordinate-descent-like approach, we can also consider optimizing` g `and` θ `simultaneously.`
end while

2.3 Related works

Below is a rough characterization of some important papers, and the bibliography here is just a starting point and is by no means exhaustive.

1. [15, 3, 6] - Learn to predict physical parameters (or latent representations thereof) of objects via exploratory interactions with the objects (pushing, etc., and observing the effects). The physics parameters can be used to update the simulator in which a final downstream policy is learned.
2. [1] Learns the distribution over randomization parameters, iterates between training policy in sim and data collection rollouts in real. Minimizes discrepancy between sim and real rollouts.
3. [4] Exploration policy generates real-world trajectories that are used to optimize dynamics parameters, which in turn are used to find a task policy.
4. [9] Uses different exploration and task policies, exploration policy is trained so that Fisher information is maximized.
5. [13] Survey paper - discusses measures used in active learning.
6. [5] introduces a real-sim-real approach comprising a real-sim training phase and a sim-real inference phase. In the real-sim phase, the sim env is constructed based on the real world and a DRL policy is trained; in the sim-real phase, the sim-trained policy controls the real robot.
7. [11] - automatic gradient-based gain tuning.
8. [8] is a study on tuning simulation parameters, and discusses the pros and cons of different approaches of doing so. This paper compares online system identification (SI) (collecting trajectories using an iteratively-trained policy) with SI using trajectories from a fixed (or trained) policy.

3 Bibliography

[1] Y. Chebotar, A. Handa, V. Makoviychuk, M. Macklin, J. Issac, N. Ratliff, and D. Fox. "Closing the sim-to-real loop: Adapting simulation randomization with real world experience". In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE. 2019, pp. 8973–8979.

[2] J. Josifovski, S. Auddy, M. Malmir, J. Piater, A. Knoll, and N. Navarro-Guerrero. "Continual Domain Randomization". In: *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2024, pp. 4965–4972.

[3] K. N. Kumar, I. Essa, S. Ha, and C. K. Liu. "Estimating mass distribution of articulated objects using non-prehensile manipulation". In: *arXiv preprint arXiv:1907.03964* (2019).

[4] J. Liang, S. Saxena, and O. Kroemer. "Learning active task-oriented exploration policies for bridging the sim-to-real gap". In: *arXiv preprint arXiv:2006.01952* (2020).

[5] N. Liu, Y. Cai, T. Lu, R. Wang, and S. Wang. "Real–sim–real transfer for real-world robot control policy learning with deep reinforcement learning". In: *Applied Sciences* 10.5 (2020), p. 1555.

[6] N. Mavrakis, R. Stolkin, et al. "Estimating an object's inertial parameters by robotic pushing: a data-driven approach". In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2020, pp. 9537–9544.

[7] B. Mehta, M. Diaz, F. Golemo, C. J. Pal, and L. Paull. "Active domain randomization". In: *Conference on Robot Learning*. PMLR. 2020, pp. 1162–1176.

[8] B. Mehta, A. Handa, D. Fox, and F. Ramos. "A user's guide to calibrating robotic simulators". In: *Conference on Robot Learning*. PMLR. 2021, pp. 1326–1340.

[9] M. Memmel, A. Wagenmaker, C. Zhu, P. Yin, D. Fox, and A. Gupta. "Asid: Active exploration for system identification in robotic manipulation". In: *arXiv preprint arXiv:2404.12308* (2024).

[10] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel. "Sim-to-real transfer of robotic control with dynamics randomization". In: *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE. 2018, pp. 3803–3810.

[11] A. von Prittwitz, K. Wahba, and W. Hönig. "Automatic Gain Tuning for Multirotors Using Differentiable Optimization". In: ().

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

[13] A. T. Taylor, T. A. Berrueta, and T. D. Murphey. "Active learning in robotics: A review of control principles". In: *Mechatronics* 77 (2021), p. 102576.

[14] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel. "Domain randomization for transferring deep neural networks from simulation to the real world". In: *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE. 2017, pp. 23–30.

[15] Z. Xu, J. Wu, A. Zeng, J. B. Tenenbaum, and S. Song. "Densephysnet: Learning dense physical object representations via multi-step dynamic interactions". In: *arXiv preprint arXiv:1906.03853* (2019).