$\mathcal{L}_1$-RL: Robustifying Reinforcement Learning Policies with $\mathcal{L}_1$ Adaptive Control

Yikun Cheng*, Pan Zhao*, Manan Gandhi, Bo Li, Evangelos Theodorou, Naira Hovakimyan

***Abstract*— A reinforcement learning (RL) policy trained in a nominal environment could fail in a new/perturbed environment due to the existence of dynamic variations. Existing robust methods try to obtain a fixed policy for all envisioned dynamic variation scenarios through robust or adversarial training. These methods could lead to conservative performance due to emphasis on the worst case, and often involve tedious modifications to the training environment. We propose an approach to robustifying a pre-trained non-robust RL policy with $\mathcal{L}_1$ adaptive control ($\mathcal{L}_1$AC) [1]. Leveraging the capability of an $\mathcal{L}_1$AC law in fast estimation of and active compensation for dynamic variations, our approach can significantly improve the robustness of a RL policy trained in a standard (i.e., non-robust) way, either in a simulator or in the real world. Numerical experiments are provided to validate the efficacy of the proposed approach.**

I. Introduction

Reinforcement learning (RL) is a promising way to solve sequential decision-making problems [2]. In the recent years, RL has shown impressive or superhuman performance in control of complex robotic systems [3]–[5]. An RL policy is often trained in a simulator and deployed in the real world. However, the discrepancy between the simulated and the real environment, known as the sim-to-real (S2R) gap, often causes the RL policy to fail in the real world. An RL policy may also be directly trained in a real-world environment; however, the environment perturbation resulting from parameter variations, actuator failures and external disturbances can still cause the well-trained policy to fail. Take a delivery drone for example (Fig. 1). We could train an RL policy to control the drone in a nominal environment (e.g, nominal load, mild wind disturbances, healthy propellers, etc.); however, this policy could fail and lead to a crash when the drone operates in a new environment (e.g., heavier loads, stronger wind disturbances, loss of propeller efficiency, etc.). To a certain extent, the S2R gap issue can be considered as a special case of environment perturbation by treating the simulated and real environments as the old/nominal and new/perturbed environments, respectively.

This work is supported by AFOSR and NSF under the NRI grant 1830639 and CPS grant 1932529.

Y. Cheng, P. Zhao, and N. Hovakimyan are with the Mechanical Science and Engineering Department, University of Illinois at Urbana-Champaign, IL 61801, USA. Email: {yikun2, panzhao2, nhovakim}@illinois.edu. Corresponding author: P. Zhao.

*Y. Cheng and P. Zhao contributed equally to this work as first authors.

B. Li is with the Computer Science Department, University of Illinois at Urbana-Champaign, IL 61801, USA. Email: lbo@illinois.edu.

M. Gandhi and E. Theodorou are with the Institute for Robotics and Intelligent Machines, Georgia Institute of Technology, Atlanta, GA 30332, USA. Email: {mgandhi,evangelos.theodorou}@ae.gatech.edu.

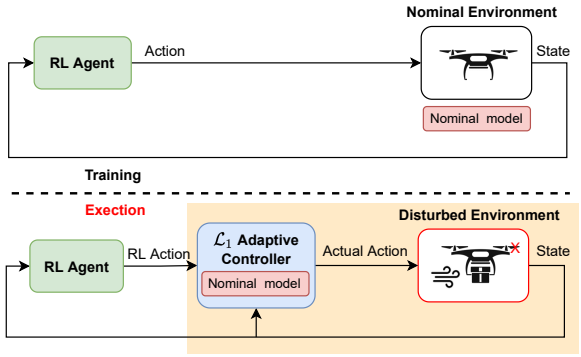

Fig. 1: Proposed $\mathcal{L}_1$-RL framework for policy robustification

To deal with the S2R gap issue, existing *robust* methods aims to improve the policy robustness through domain randomization [6]–[8] or adversarial training [9]. However, due to reliance on a *fixed* policy for all the scenarios, these methods often lead to conservative performance, and can only handle a small range of dynamic variation. Moreover, these methods often need tedious modifications to the training environment, which are most suitable for RL training in a *simulated* environment.

In this paper, we propose a framework to robustify an RL policy leveraging $\mathcal{L}_1$ adaptive control ($\mathcal{L}_1$AC) [1], which we term as $\mathcal{L}_1$-RL, and is illustrated in Fig 1. The essential idea of $\mathcal{L}_1$AC is to actively and quickly estimate the dynamic uncertainties and use the estimated value to compute the control input to compensate for these uncertainties – within the bandwidth of the control channel – so that the actual uncertain or perturbed system behaves like a nominal model. A unique feature of $\mathcal{L}_1$AC is the decoupling of the estimation loop from the control loop. This decoupling allows the use of fast adaptation, desired for quickly estimating the (potentially fast-varying) dynamic uncertainties, without sacrificing robustness. Different from most of existing policy robustification methods based on domain randomization or adversarial training [6]–[9], our $\mathcal{L}_1$-RL framework can be used to robustify an RL policy, trained in the standard (i.e., non-robust) way, either in a simulator or in the real world. The core of $\mathcal{L}_1$-RL is the built-in $\mathcal{L}_1$AC scheme which quickly estimates and compensates for the dynamic variations such that *the perturbed environment is close to the nominal environment*, where the RL policy is expected to function well.

A. Related work

Robust/adversarial training. Domain/dynamics randomization was proposed to close the sim-to-real S2R) gap [6]–

[8] when transferring a policy from a simulator to the real world. Robust adversarial training addresses the S2R gap and environment perturbations by formulating a two-player zero-sum game between the agent and the disturbance [9]. These methods involve tedious modifications to the training environment, which can mostly happen in a simulator. More importantly, the resulting policies trained in this way could overfit to the worst-case scenarios, and thus lead to conservative or degraded performance in other cases [10].

Active dynamic variations compensation. Kim et al. [11] proposed to use an disturbance-observer (DOB) to improve the robustness of an RL policy, in which the mismatch between the simulated training environment and the testing environment is estimated as disturbance and compensated for. A similar idea was pursed in [12], which used a model reference adaptive control (MRAC) scheme to estimate and compensate for parameteric uncertainties. Our objectives are similar to the ones in [11] and [12], but our approach is fundamentally different, as we consider a broader class of dynamic uncertainties (e.g., unknown input gain that cannot be handled by [11], and time-dependent disturbances that cannot be handled by [12]), and – with the help of $\mathcal{L}_1$ adaptive controller – we ensure guaranteed, scalable and predictable transient performance.

Learning to adapt. Meta-RL has recently been proposed to achieve fast adaptation of a pre-trained policy in the presence of dynamic variations [13]–[17]. Despite impressive performance mainly in terms of fast adaptation demonstrated by these methods, the intermediate policies learned during the adaptation phase will most likely still fail. This is because a certain amount of information-rich data needs to be collected in order to *learn* a good model and/or policy. On the other hand, rooted in the theory of adaptive control and disturbance estimation, [1], [18]–[20], our proposed method can *quickly estimate* the discrepancy between a nominal model and the actual dynamics, and *actively compensate* for it in a timely manner. We envision that our proposed method can be combined with these methods to achieve robust and fast adaptation.

II. Problem Setting

We assume that we have access to the dynamics of an RL agent in the *nominal* environment, either simulated or in the real world, and it is described by a nonlinear control-affine model:

$$\dot{x}(t) = f(x(t)) + g(x(t))u(t) \triangleq F_{\text{nom}}(x(t), u(t)), \quad (1)$$

where $x(t) \in \mathbb{R}^n$ and $u(t) \in \mathbb{R}^m$ are the state and input vectors, respectively, $f : \mathbb{R}^n \to \mathbb{R}^n$ and $g : \mathbb{R}^n \to \mathbb{R}^m$ are known functions; moreover, $g(x)$ has full column rank.

Remark 1. Control-affine models are commonly used for control design and can represent a broad class of mechanical and robotic systems. In addition, a control non-affine model can be converted into a control-affine model by introducing extra state variables (see e.g., [21]). Therefore, the control-affine assumption is not very restrictive.

Remark 2. The nominal model (1) can be from physics-based modeling, data-driven modeling or a combination of both. Methods exist for maintaining the control affine structure in data-driven modeling (see e.g., [22]).

We further assume that the dynamics of the agent in the *perturbed* environment can be represented by

$$\dot{x} = f(x) + g(x)\Lambda(x)u + d(t, x), \quad (2)$$

where $\Lambda(x)$ is an unknown input gain matrix, which is non-singular for any x, $d(t, x)$ is an unknown function that can capture parameter perturbations, unmodeled dynamics and external disturbances.

Remark 3. Uncertain input gain is very common in real-world systems under environment perturbations. For instance, actuator failures, and variations in mass or inertia for force- or torque-controlled robotic systems, normally induce such input gain uncertainty. Our representation of such uncertainty in (2) is broad enough to capture a large class of scenarios, while still allowing for effective compensation of such input gain uncertainty using $\mathcal{L}_1$AC (detailed in Section III).

Assumption 1. We have access to a *nominal* policy, $\pi_o(x)$, which functions well for the *nominal* dynamics (1).

The policy $\pi_o(x)$ can be trained either in a simulator or in the real world in the standard (i.e., non-robust) way. The nominal policy π_0 could fail in the perturbed environment due to the dynamic variations. In this paper, we propose a method to robustify this nominal policy so that it could function in the presence of such dynamic variations, by leveraging $\mathcal{L}_1$AC [1].

III. $\mathcal{L}_1$-RL framework for Policy Robustification

A. Overview of the $\mathcal{L}_1$-RL framework

The idea of our proposed $\mathcal{L}_1$-RL framework is depicted in Fig. 1. Within $\mathcal{L}_1$-RL, the *training* phase is standard: the nominal policy can be trained using standard methods in a nominal environment, which does not need domain randomization or adversarial training. During policy *execution*, an $\mathcal{L}_1$ controller uses the nominal dynamics (1) as an internal nominal model, estimates the discrepancy between the nominal model and the actual dynamics and compensates for this discrepancy so that *the actual dynamics with the $\mathcal{L}_1$ controller* (illustrated by the shaded area of Fig. 1) *behaves like the nominal dynamics*. Since the RL policy works well under the nominal dynamics, it is expected to work well in the presence of dynamic variations *and* the $\mathcal{L}_1$ augmentation.

B. RL training for the nominal policy

As mentioned before, the policy can be trained in the standard way, using a large amount of RL methods including both model-free and model-based ones. The only requirement is that one has nominal dynamics of the training environment in the form of (1).

As an illustration of the idea, in the numerical experiments, we choose PILCO [23], a model-based policy search method,

and a trajectory optimization method based differential dynamic programming (DDP) [24], [25] to obtain the nominal policy.

C. $\mathcal{L}_1$ augmentation for policy robustification

In this section, we explain how an $\mathcal{L}_1$AC law can be designed to augment the nominal policy to improve its robustness. An $\mathcal{L}_1$ controller mainly consists of three components: a state predictor, an adaptive law, and a control law. The state predictor is used to predict the system's state evolution, and the prediction error is subsequently used in the adaptive law to update the disturbance estimates. The control law aims to compensate for the estimated disturbance. For the perturbed system (2) with the nominal dynamics (1), these components are detailed as follows. The **state predictor** is defined as:

$$\dot{\hat{x}} = F_{\text{nom}}(x,u) + g(x)\hat{\sigma}_m(t)) + g^{\perp}(x)\hat{\sigma}_{um}(t) - a\tilde{x}, \quad (3)$$

where $\tilde{x} \triangleq \hat{x} - x$ is the prediction error, a is a positive scalar, $\hat{\sigma}_m(t)$ and $\hat{\sigma}_{um}(t)$ are the *matched* and *unmatched* disturbance estimates[1], respectively, $g^{\perp}(x) \in \mathbb{R}^{n-m}$ satisfies $g(x)^{\top} g^{\perp}(x) = 0$, and $\text{rank}[G(x)] = n$ for any x with $G(x) \triangleq \left[g(x)\ g^{\perp}(x)\right]$. Note that unmatched disturbances (or mismatched disturbances used in the disturbance-observer based control literature [20]) cannot be directly canceled by control signals and are generally challenging to deal with. Following the piecewise-constant (PWC) **adaptive law** (which connects with the CPU sampling time) [1, Section 3.3], the disturbance estimates are updated as

$$\begin{aligned} \begin{bmatrix} \hat{\sigma}_m(t) \\ \hat{\sigma}_{um}(t) \end{bmatrix} &= \begin{bmatrix} \hat{\sigma}_m(iT) \\ \hat{\sigma}_{um}(iT) \end{bmatrix}, \quad t \in [iT, (i+1)T), \\ \begin{bmatrix} \hat{\sigma}_m(iT) \\ \hat{\sigma}_{um}(iT) \end{bmatrix} &= -G^{-1}(x(iT)) \frac{a}{e^{aT}-1} \tilde{x}(iT), \end{aligned} \quad (4)$$

where T is the sampling time. The **control law** (applied to the actual system) is defined as

$$u(s) = -\frac{1}{s} K \hat{\eta}(s), \quad (5)$$

where $K \in \mathbb{R}^{m\times m}$ is a feedback gain matrix, $\hat{\eta}(s)$ is the Laplace transform of $\hat{\eta}(t) = \omega_0 u(t) + \hat{\sigma}_m(t) - u_{\text{RL}}(t)$ with $\omega_0 \in \mathbb{R}^{m\times m}$ being a nominal gain matrix (which is often selected to be an identity matrix) and $u_{\text{RL}}(t) = \pi_0(x(t))$ being the control command from the nominal RL policy $\pi_0(t)$. For details on deriving the estimation and control laws, readers can refer to [26], [27]. It is worth emphasizing that the control law only compensates for the matched estimated disturbance ($\hat{\sigma}_m$) by directly canceling it, and a feedback structure is introduced in (5) to compensate for the effect of unknown input gain $\Lambda(x)$, which computes the ultimate control command using $\hat{\sigma}_m$ and u_{RL}.

[1]In a $\mathcal{L}_1$AC scheme with a piecewise constant adaptive law [1, section 3.3], all the dynamic uncertainties (parametric uncertainties, unmodeled dynamics, external disturbances, etc) are *lumped* together and estimated as disturbances.

IV. Numerical Experiments

We now present the numerical experiments on a simple cart-pole benchmark problem, and a complex 12-state quadrotor control example.

A. Cart-pole

The dynamics of the cart-pole system is taken from [28]. The system states include cart position (x_c) and velocity ($\dot{x}_c$), and pole angle (θ) and angular velocity ($\dot{\theta}$). The input is the force applied to the cart. The nominal value of the key parameters in the dynamics are $m_1 = 0.5\ kg$ (cart mass), $m_2 = 0.5\ kg$ (pole mass), $l_{\text{pole}} = 0.6\ m$ (pole length). The pole is roughly hanging straight down ($\theta = 0$) with small random perturbations at the beginning. The goal is to search for a policy that can swing up the pole and balance it at the straight up position (corresponding to $x_c = 0$ and $\theta = 180°$).

We used PILCO [23] to search for a policy for the nominal environment defined by the nominal values mentioned above. PILCO uses Gaussian processes (GPs) [29] to learn the systems dynamics, uses the learned dynamics together with uncertainty propagation (e.g., based on moment matching or linearization) to compute the cost function, and then applies gradient descent to search for the optimal policy. PILCO achieved unprecedented records in terms of data-efficiency in RL.

For $\mathcal{L}_1$AC law design, the parameters in (3)–(5) were chosen to be $a = 10$, $T = 0.002$ second, $w_0 = 1$ and $K = 200$.

We next perturbed the environment to test the robustness of the nominal policy with and without $\mathcal{L}_1$AC augmentation. For design of the $\mathcal{L}_1$AC law we used the physics-based model with the nominal parameter values as the nominal model, instead of the GP model learned during policy training, for simplicity. Fig. 2 shows the results in the presence of perturbations in the cart mass and pole length. One can see that the $\mathcal{L}_1$ augmentation significantly improves the robustness of the PILCO policy. For instance, PILCO plus $\mathcal{L}_1$ augmentation was able to consistently achieve the goal even when the cart pass was perturbed to 3 kg (six times of its nominal value) or when the pole length was reduced to 0.2 m (one third of its nominal value).

We further performed testing under ten scenarios from random joint perturbations of the cart mass, pole mass and length parameters, in the range of $m_1 \in [0.1, 5]$, $m_2 \in [0.1, 5]$, $l_{\text{pole}} \in [0.6, 1]$. The sampled parameters and the success/failure results for each scenario are shown in Fig. 3. Once again, the $\mathcal{L}_1$ augmentation significantly improved the policy robustness, as validated by the much higher mission success rate.

B. 3-D Quadrotor

We next do experiments on a 12-state 3-D quadrotor example. The equations of dynamics are taken from [12] which uses Euler angles. The states include quadrotor position (x, y, z) in an inertia frame and the roll, pitch, and yaw angles of the quadrotor body frame with respect to the inertial frame, as well as their derivatives. Motor mixing is

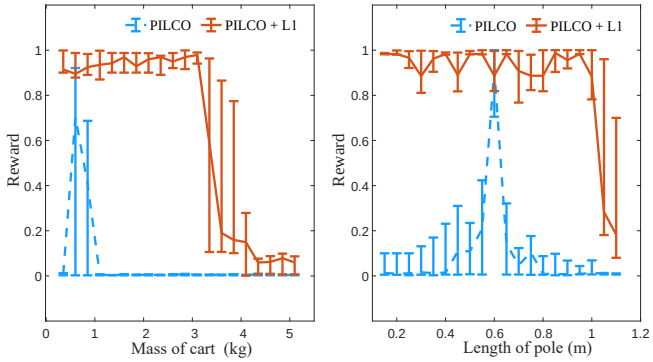

Fig. 2: Results in the presence of perturbations in cart mass and pole length. Ten trials were performed and average results with variances are shown for each perturbation case. Reward is normalized.

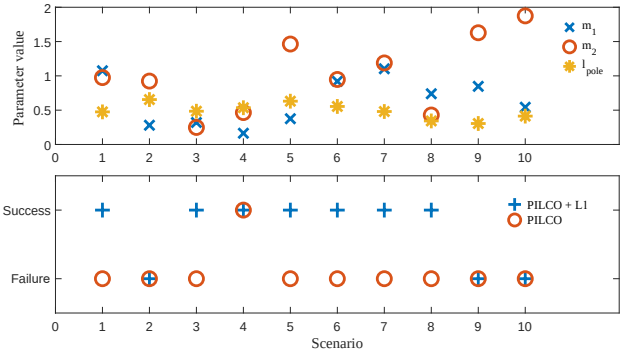

Fig. 3: Results (bottom) under ten scenarios from random perturbations of the cart mass, pole mass and length (sampled value shown at the top)

also included in the dynamics. The inputs are the four thrusts output of the four propellers.

The nominal value of the key parameters are set to be $[I_x, I_y, I_z] = [0.082, 0.0845, 0.1377]$ kgm^2 (moment of inertia), $m = 4.34\ kg$ (qaudrotor mass), and $c_{pi} = 1$ $(i = 1, 2, 3, 4)$ (propeller control coefficients). The mission in this example is to control the quadrotor to fly from the origin to the target point $(4, 4, 2)$. To obtain a policy for achieving the mission, we chose to use trajectory optimization, which together with model learning is commonly used for model-based RL [30], [31]. We further selected to use differential dynamic programming (DDP) [25] a specific trajectory optimization method. Since our focus is not on the training but on robustifying a pre-trained policy, we once again use the physics-based dynamic model with the nominal parameter values as the model "learned" in the nominal environment. This model is used for computing the DDP policy, and for designing the $\mathcal{L}_1$AC law.

For $\mathcal{L}_1$AC law design, the parameters in (3)–(5) were chosen to be $a = 10$, $T = 0.001$ second, $\omega_0 = I_4$ and $K = 200$.

We tested the performance of the DDP policy with and without $\mathcal{L}_1$ augmentation under three types of dynamic perturbations. The first one is **loss of propeller efficiency**, to mimic the effect of propeller failures, which are simulated by adjusting the control coefficients c_{pi} $(i = 1, 2, 3, 4)$. The resulting trajectories under ten scenarios are shown in Fig. 4. One can see that $\mathcal{L}_1$ augmentation significantly improved the robustness of the DDP policy, leading to consistent trajectories that are close to the ideal trajectory obtained by applying the policy to the nominal dynamics. The second type of dynamic perturbations are the **mass and**

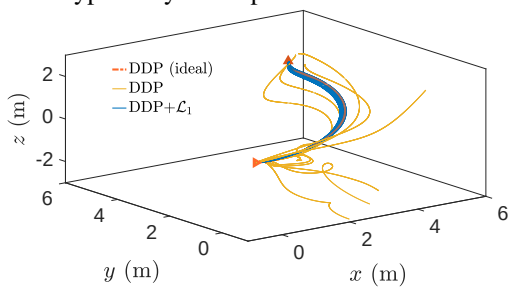

Fig. 4: Results under loss of propeller efficiency. In each of ten scenarios, the control coefficients of two propellers were randomly selected to be in $[0.5, 1]$. DDP (ideal) denotes the trajectory obtained by applying the policy to the nominal dynamics.

inertia change, e.g., to mimic the effect of carrying different packages for a delivery drone. The results under ten scenarios with randomly increased mass and inertia through a scale of $[2, 5]$ are shown in Fig. 5. Once again, $\mathcal{L}_1$ augmentation significantly improved the policy robustness, leading to close-to-ideal trajectories. The third type of dynamic variations

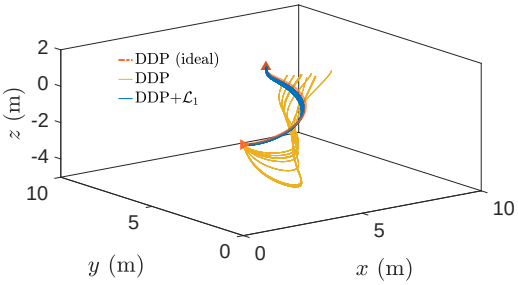

Fig. 5: Results under perturbations in quadrotor mass and inertia. In each of the ten scenarios, the mass and inertia are scaled by a random number in $[2, 5]$.

is related to **wind disturbances** in the horizontal plane, which causes disturbance forces to the x and y directions. In each of the ten scenarios, the forces were simulated by stochastic variables with the mean values randomly sampled from $[10, 25]$. The results are shown in Fig. 6. $\mathcal{L}_1$ augmentation improved the robustness, but was not able to yield close-to-ideal performance. This is mainly because the wind disturbances will cause unmatched disturbances (σ_{um} in (3) and (4)), which are not compensated for in the control law (5).

V. Conclusion

This paper presents $\mathcal{L}_1$-RL, a framework for robustifying a pre-trained reinforcement learning (RL) policy, leveraging $\mathcal{L}_1$ adaptive control to quickly and actively estimate and

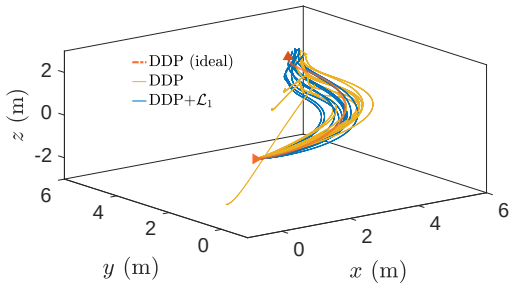

Fig. 6: Results under wind disturbance. In each of the ten scenarios, the mean value of the wind disturbance force applied to the x and y directions were sampled from the range of $[10, 25]$.

compensate for the dynamic variations which could happen during execution of this policy. Our framework allows for the policy to be trained in a standard way (i.e., without use of domain randomization or adversarial training), either in a simulator or in the real world. Numerical experiments on a simple benchmark and a 12-state quadrotor examples illustrate the efficacy of the proposed framework.

Future work includes demonstration of the proposed framework in a model-free RL setting, comparison of the framework with existing robust/adversarial training based methods [6]–[9], as well as validations on real-world experiments. We also plan to incorporate the extension of $\mathcal{L}_1$ adaptive control for unmatched uncertainties from [32] to achieve improved performance for a broader class of uncertainties.

REFERENCES

[1] N. Hovakimyan and C. Cao, *$\mathcal{L}_1$ Adaptive Control Theory: Guaranteed Robustness with Fast Adaptation*. Philadelphia, PA: Society for Industrial and Applied Mathematics, 2010.

[2] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT press, 2018.

[3] E. Kaufmann, A. Loquercio, R. Ranftl, A. Dosovitskiy, V. Koltun, and D. Scaramuzza, "Deep drone racing: Learning agile flight in dynamic environments," in *Conference on Robot Learning*, pp. 133–145, 2018.

[4] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter, "Learning agile and dynamic motor skills for legged robots," *Science Robotics*, vol. 4, no. 26, 2019.

[5] Y. Song, M. Steinweg, E. Kaufmann, and D. Scaramuzza, "Autonomous drone racing with deep reinforcement learning," *arXiv preprint arXiv:2103.08624*, 2021.

[6] 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 *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 23–30, 2017.

[7] 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)*, pp. 3803–3810, 2018.

[8] A. Loquercio, E. Kaufmann, R. Ranftl, A. Dosovitskiy, V. Koltun, and D. Scaramuzza, "Deep drone racing: From simulation to reality with domain randomization," *IEEE Transactions on Robotics*, vol. 36, no. 1, pp. 1–14, 2019.

[9] L. Pinto, J. Davidson, R. Sukthankar, and A. Gupta, "Robust adversarial reinforcement learning," in *International Conference on Machine Learning*, pp. 2817–2826, 2017.

[10] L. Rice, E. Wong, and Z. Kolter, "Overfitting in adversarially robust deep learning," in *International Conference on Machine Learning*, pp. 8093–8104, 2020.

[11] J. W. Kim, H. Shim, and I. Yang, "On improving the robustness of reinforcement learning-based controllers using disturbance observer," in *IEEE 58th Conference on Decision and Control (CDC)*, pp. 847–852, 2019.

[12] A. Guha and A. Annaswamy, "MRAC-RL: A framework for on-line policy adaptation under parametric model uncertainty," *arXiv preprint arXiv:2011.10562*, 2020.

[13] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *Proceedings of the 34th International Conference on Machine Learning*, vol. 70 of *Proceedings of Machine Learning Research*, pp. 1126–1135, 06–11 Aug 2017.

[14] A. Nagabandi, I. Clavera, S. Liu, R. S. Fearing, P. Abbeel, S. Levine, and C. Finn, "Learning to adapt in dynamic, real-world environments through meta-reinforcement learning," *arXiv preprint arXiv:1803.11347*, 2018.

[15] A. Nagabandi, C. Finn, and S. Levine, "Deep online learning via meta-learning: Continual adaptation for model-based RL," *arXiv preprint arXiv:1812.07671*, 2018.

[16] M. Xu, W. Ding, J. Zhu, Z. LIU, B. Chen, and D. Zhao, "Task-agnostic online reinforcement learning with an infinite mixture of gaussian processes," in *Advances in Neural Information Processing Systems*, vol. 33, pp. 6429–6440, 2020.

[17] B. Chen, Z. Liu, J. Zhu, M. Xu, W. Ding, and D. Zhao, "Context-aware safe reinforcement learning for non-stationary environments," *arXiv preprint arXiv:2101.00531*, 2021.

[18] K. J. Åström and B. Wittenmark, *Adaptive Control*. New York: Dover, 2nd ed., 2008.

[19] P. A. Ioannou and J. Sun, *Robust Adaptive Control*. Mineola, NY: Dover Publications, Inc., 2012.

[20] W.-H. Chen, J. Yang, L. Guo, and S. Li, "Disturbance-observer-based control and related methods—An overview," *IEEE Transactions on Industrial Electronics*, vol. 63, no. 2, pp. 1083–1095, 2015.

[21] R. TAKANO and M. YAMAKITA, "Robust control barrier function for systems affected by a class of mismatched disturbances," *SICE Journal of Control, Measurement, and System Integration*, vol. 13, no. 4, pp. 165–172, 2020.

[22] M. J. Khojasteh, V. Dhiman, M. Franceschetti, and N. Atanasov, "Probabilistic safety constraints for learned high relative degree system dynamics," in *Annual Conference on Learning for Dynamics and Control*, pp. 781–792, 2020.

[23] M. Deisenroth and C. E. Rasmussen, "PILCO: A model-based and data-efficient approach to policy search," in *Proceedings of the 28th International Conference on machine learning*, pp. 465–472, 2011.

[24] W. Li and E. Todorov, "Iterative linear quadratic regulator design for nonlinear biological movement systems.," in *International Conference on Informatics in Control, Automation and Robotics*, pp. 222–229, 2004.

[25] Y. Tassa, T. Erez, and E. Todorov, "Synthesis and stabilization of complex behaviors through online trajectory optimization," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4906–4913, IEEE, 2012.

[26] P. Zhao, Y. Mao, C. Tao, N. Hovakimyan, and X. Wang, "Adaptive robust quadratic programs using control Lyapunov and barrier functions," in *59th IEEE Conference on Decision and Control*, pp. 3353–3358, 2020.

[27] P. Zhao, S. Snyder, N. Hovakimyan, and C. Cao, "Robust adaptive control of linear parameter-varying systems with unmatched uncertainties," Submitted to *Automatica,* arXiv:2010.04600, 2020.

[28] M. P. Deisenroth, A. McHutchon, J. Hall, and C. E. Rasmussen, "PILCO code documentation v0.9," 2013.

[29] C. K. I. Williams and C. E. Rasmussen, *Gaussian Processes for Machine Learning*, vol. 2. Cambridge, MA: MIT press, 2006.

[30] S. Levine and V. Koltun, "Learning complex neural network policies with trajectory optimization," in *International Conference on Machine Learning*, pp. 829–837, PMLR, 2014.

[31] Y. Pan and E. Theodorou, "Probabilistic differential dynamic programming," in *Advances in Neural Information Processing Systems*, pp. 1907–1915, 2014.

[32] P. Zhao, S. Snyder, N. Hovakimyana, and C. Cao, "Robust adaptive control of linear parameter-varying systems with unmatched uncertainties," Submitted to *Automatica, arXiv:2010.04600*, 2020.