Environment as Policy: Learning to Race in Unseen Tracks

Jiaxu Xing*, Hongze Wang*, Nico Messikommer, and Davide Scaramuzza

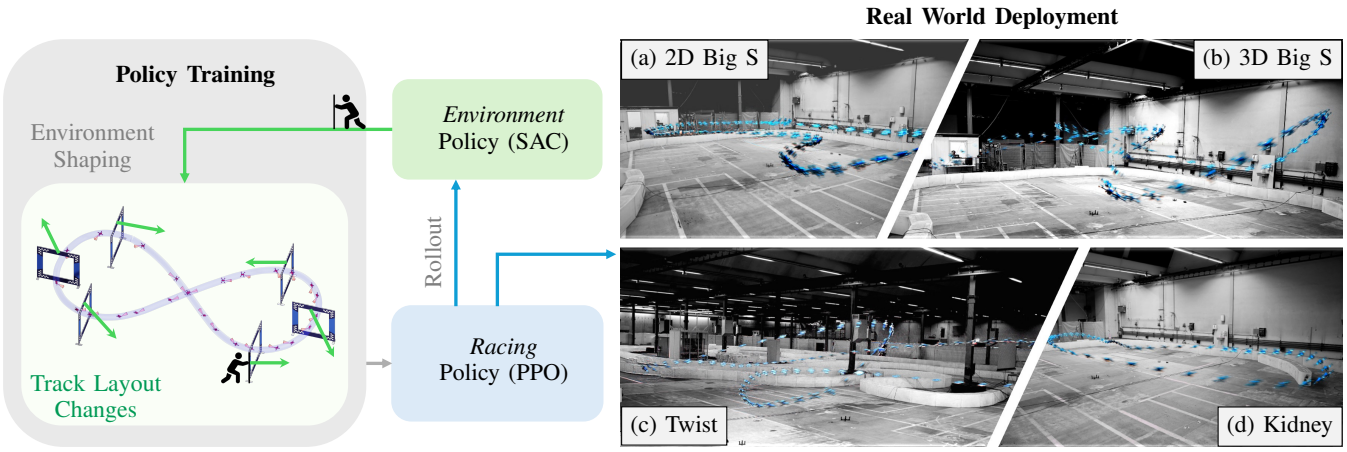


Fig. 1: In this work, we introduce an environment-shaping framework to improve the generalization of RL drone racing agents to diverse and unseen tracks without retraining. We use a Soft Actor-Critic (SAC) policy to adaptively shape the track layouts by generating difficult but achievable race tracks based on the racing agent's actual performance. The resulting *single* racing policy can fly in various unseen and challenging race tracks in the real world with competitive lap times.

***Abstract*— Reinforcement learning (RL) has achieved outstanding success in complex robot control tasks, such as drone racing, where the RL agents have outperformed human champions in a known racing track. However, these agents fail in unseen track configurations, always requiring complete retraining when presented with new track layouts. This work aims to develop RL agents that generalize effectively to novel track configurations without retraining. To enhance the generalizability of the RL agent, we propose an adaptive environment-shaping framework that dynamically adjusts the training environment based on the agent's performance. We achieve this by leveraging a secondary RL policy to design environments that strike a balance between being challenging and achievable, allowing the agent to adapt and improve progressively. Using our adaptive environment shaping, one single racing policy efficiently learns to race in diverse challenging tracks.**

Website: `http://rpg.ifi.uzh.ch/env_as_policy`.

I. Introduction

Reinforcement learning (RL) involves agents learning through trial and error by interacting with a pre-defined environment and maximizing the rewards based on these interactions. It has proven highly effective in various robotic control applications, demonstrating remarkable task performance across scenarios like dexterous manipulation [1], [2], [3], [4], quadrupedal locomotion [5], [6], and agile quadrotor flight [7], [8], [9], [10], [11]. However, while RL agents excel within the specific distributions they are trained on, they struggle with out-of-distribution configurations and may require retraining from scratch for even minor configuration changes [12]. To improve adaptability and generalization, extending the RL framework to train and perform across a broader range of distributions is essential, enabling agents to handle more diverse and dynamic environments effectively.

Domain randomization is a commonly used technique to improve the learning capability of RL agents across a broader range of tasks [12], [13], [14], [15]. This approach varies the parameters of the training environment to expose the agent to a wide variety of potential deployment scenarios. By learning in diverse conditions, the agent develops robust policies less likely to overfit the specific characteristics of a single environment. Domain randomization is often combined with curriculum learning, where the complexity and variability of training scenarios are incrementally increased [15]. However, curriculum learning often depends on manual design, introducing human biases. This reliance on fixed, non-adaptive progressions can limit training diversity and restrict the agent's ability to generalize to new, real-world situations [16]. To address this problem, we propose an automatic adaptive environment-shaping framework to enhance the agent's generalization ability to fly in unseen race tracks. This framework dynamically adjusts the environment curriculum based on the agent's learning progress (Fig. 1). The key idea is to create consistently challenging yet attainable environments, avoiding tracks that are too easy or overly difficult, which could hinder learning. By automating the environment shaping, the system can more precisely tailor the learning environment to the agent's performance, enhancing the agent's ability to generalize across various unseen track configurations. We show that our approach outperforms the existing environment-shaping approaches using the same number of actions and enables the racing policy to generalize to a diverse set of unseen and complicated tracks.

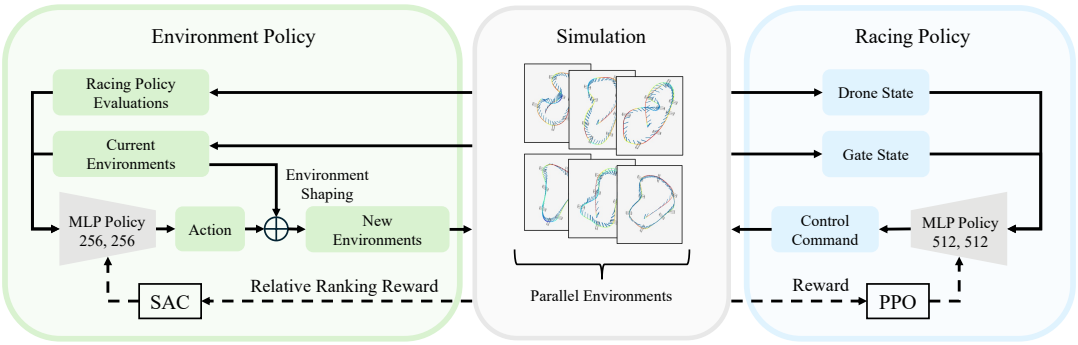


Fig. 2: Overview of the proposed method.

II. Related Works

Environment shaping has been widely applied in reinforcement learning by designing a distribution of environments that progressively improves the agent's performance. Previous works like [17], [18], [19], [20] on environment shaping have been successful in simulation tasks like Bipedal Walker and maze games, where environments are easily parameterized for manual curriculum design. However, this process becomes more challenging for more complex real-world robotic tasks. Domain randomization is a common solution, where environments are sampled from a predefined range to enable robust task execution. For instance, [15], [14] used domain randomization to achieve dexterous manipulation in the real world. Additionally, [5] proposed an adaptive terrain curriculum using a particle filter to sample environment parameters, enabling quadrupedal locomotion across various terrains in real-world scenarios.

III. Methodology

Racing Policy Training. The autonomous racing task can be framed as an optimization problem, where the objective is to minimize the time it takes for an agile quadrotor to pass through a predefined sequence of gates [21], as illustrated in Fig. 3. In this task, we define the observations as $\boldsymbol{o}_{\text{racing}} = \left[\tilde{\boldsymbol{R}}, \boldsymbol{v}, \boldsymbol{\omega}, a_{\text{prev}}, \delta\boldsymbol{p}_1, \delta\boldsymbol{p}_2\right]$, where $\tilde{\boldsymbol{R}} \in \mathbb{R}^6$ is a vector comprising the first two columns of $\boldsymbol{R}_{WB}$ [22], $\boldsymbol{v} \in \mathbb{R}^3$ and $\boldsymbol{\omega} \in \mathbb{R}^3$ denote the linear and angular velocity of the drone, a_{prev} represents the previous action from the actor policy, and $\delta\boldsymbol{p}_1, \delta\boldsymbol{p}_2 \in \mathbb{R}^{12}$ represent the relative difference in position of the four next gate corners (4×3) in the world frame. The total reward at time t, denoted as r_t, consists of several components:

$$r_t^{\text{racing}} = r_t^{\text{prog}} + r_t^{\text{act}} + r_t^{\text{br}} + r_t^{\text{pass}} + r_t^{\text{crash}}, \tag{1}$$

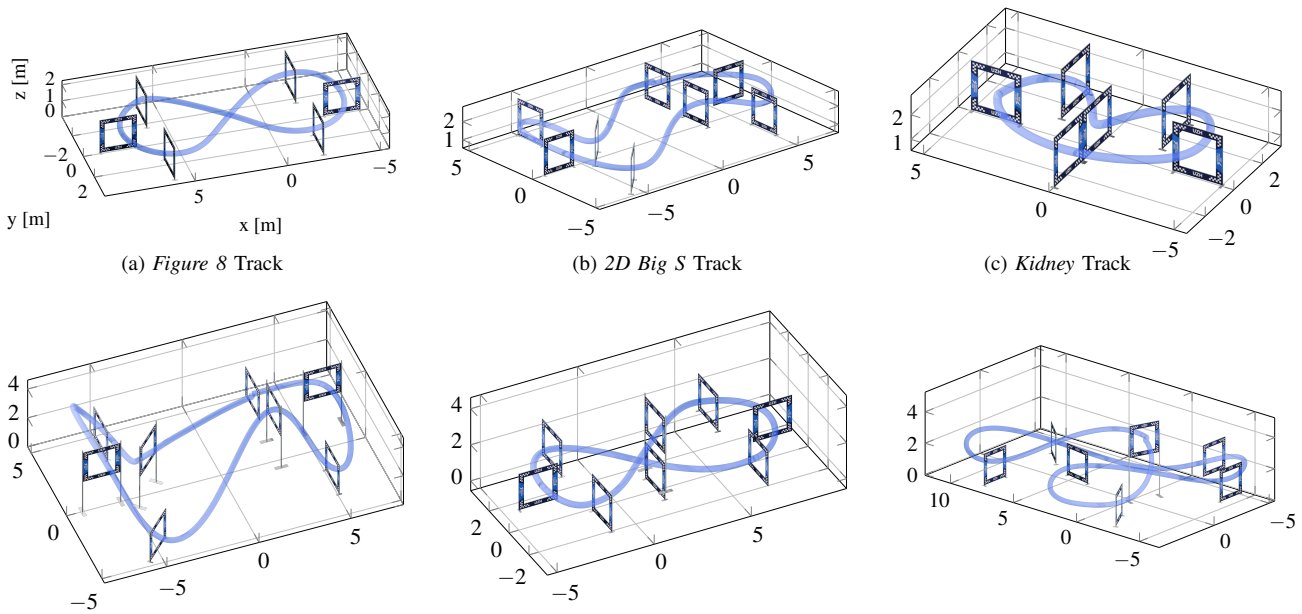
where r_t^{prog} represents progress toward passing the next gate [23], r_t^{act} penalizes changes in actions from the previous time step, r_t^{br} discourages high body rates to ensure a stable flying behavior, r_t^{pass} is a binary reward for successfully passing the next gate, and r_t^{crash} is a binary penalty applied when a collision occurs, which also terminates the episode. The reward components are formulated as follows

$$\begin{aligned} r_t^{\text{prog}} &= \alpha_1(d_{\text{Gate}}(t-1) - d_{\text{Gate}}(t)), \\ r_t^{\text{act}} &= \alpha_2\|\boldsymbol{u}_t - \boldsymbol{u}_{t-1}\|, \\ r_t^{\text{br}} &= \alpha_3\|\boldsymbol{\omega}_{\mathcal{B},t}\|, \\ r_t^{\text{pass}} &= \alpha_4 \quad \text{if robot passes the next gate,} \\ r_t^{\text{crash}} &= \alpha_5 \quad \text{if robot crashes (gates, ground).} \end{aligned} \tag{2}$$

Environment Policy Training. The track layout used for training the racing policy plays a critical role in shaping its flying capabilities while also impacting the overall training stability. If the tracks are too difficult, the agent will struggle to extract meaningful learning signals. Conversely, simple tracks fail to challenge the agent, limiting its ability to generalize to more complex environments. Thus, to ensure effective learning, the track difficulty must be *continuously adapted to the current capabilities of the agent*. To achieve this, we introduce a learned environment policy π_{env} that dynamically adjusts the race tracks. This adaptive approach allows the agent to consistently gather relevant and progressive learning experiences, optimizing its training stability and performance in diverse race tracks.

1) MDP Formulation: In our racing scenario, the states of the environment are represented by the position $\boldsymbol{p}$ and orientation $\boldsymbol{R}$ of each individual gate, such that the full gate state vector is given by $\boldsymbol{s}_{\text{gates}} = [\boldsymbol{p}_1, \boldsymbol{R}_1, \ldots, \boldsymbol{p}_N, \boldsymbol{R}_N]$, where N is the total number of gates in the environment. The environment policy leverages the observation $\boldsymbol{o}_{\text{env}} = [\boldsymbol{s}_{\text{gates}}, \boldsymbol{e}_{\text{racing}}]$, where s_{gates} represents the current states of all gates, and $\boldsymbol{e}_{\text{racing}}$ indicates the performance of the drone agent achieved in the track corresponding to the environment state s_{gates}. The evaluation performance vector $\boldsymbol{e}_{\text{racing}} = [\boldsymbol{e}_1, \ldots, \boldsymbol{e}_N]$ contains the gate-passing error for each of the N gates, where the error $\boldsymbol{e}_i$ is computed as the distance between the drone's position and the center of gate i at the moment the drone passes through the gate. The environment policy outputs actions $a = [\Delta\boldsymbol{p}_{\text{gates}}, \Delta\text{yaw}_{\text{gates}}]$, where $\Delta\boldsymbol{p}_{\text{gates}}$ specifies changes in gate positions, and $\Delta\text{yaw}_{\text{gates}}$ indicates changes in the yaw angles of the gates, both in the world frame coordinates. Hence, the transition dynamics $\mathbb{P}$ is naturally $s^t_{\text{gates}} = s^{t-1}_{\text{gates}} + a^{t-1}$. As training progresses, the training track layout reflects accumulated changes from the initial track, naturally producing various tracks.

Another key component of our framework is the development of a metric that precisely measures the effectiveness



(a) *Figure 8* Track (b) *2D Big S* Track (c) *Kidney* Track

(d) *3D Big S* Track (e) *3D Figure 8* Track (f) *Twist* Track

Fig. 3: Visualization of the drone racing tracks used for the experiments, each characterized by varying levels of complexity. All the tracks maintain a consistent size scale, spanning widths from 8 meters to 16 meters.

of the environment policy's actions on the performance of the racing agent. In our work, we propose a reward for the environment policy grounded in the *relative ranking* of the performance of the racing policy in different environments. Specifically, after each training phase, we rank all the parallel training environments according to the number of gates the agent successfully passed. The higher the ranking number, the worse the policy performs. The environment policy is then penalized for generating track layouts at the extremes of the ranking—those that are either too easy or too difficult. Additionally, we reward race tracks that fall within an intermediate range, where the agent demonstrates a degree of success but has not fully mastered the track. The reward is defined as follows:

$$r_t^{env} = \begin{cases} R\frac{N_{\text{env}} - \text{rank}}{N_{\text{env}} - r_{\text{upper}}} & \text{if } rank > r_{\text{upper}} \\ R & \text{if } rank \in [r_{\text{lower}}, r_{\text{upper}}] \\ R\frac{\text{rank}}{r_{\text{lower}}} & \text{if } rank < r_{\text{lower}} \end{cases} \quad (3)$$

where R represents a positive constant, r_{lower} and r_{upper} are the fixed thresholds determining the mid-range representing the tracks that are not too easy and not too hard, N_{env} represents the total number of parallel training environments. This formulation represents a smooth reward function that provides fine-grained feedback on the environment policy for each generated training track.

IV. Experiments

Our experiments aim to address the following key research questions: (i) How does our approach generalize to unseen race tracks? (ii) How does our environment policy perform on different training configurations? (iv) Does our policy transfer to the real world?

Experimental Setup. To evaluate the performance of our approach, we tested the racing policy on several fixed racetracks with varying difficulties and complexities. All of these tracks are significantly different than the initial training track, with most having a different number of gates than during training. For the evaluation, we primarily used two metrics: success rate (SR %) and lap time (LT [s]). The success rate is calculated as the ratio of successful runs, where the drone passes all the gates without crashing, over the total number of trials. Lap time refers to the drone's total time to successfully complete a race track. These metrics are commonly used in drone racing and are essential in evaluating whether the policy enables fast and stable flight performance [24], [25].

Baselines. To evaluate the generalization ability of our proposed methods, we compared them with three baseline methods: (i) RL policy without curriculum: This method shares the same initial environment setup as ours but does not use a curriculum for training. (ii) Domain randomization from [7]: In this approach, the initial environment is the same as in our method. However, environment policy actions are sampled randomly within the action space; we use the identical configurations from [7]. (iii) Particle Filter Curriculum from [5]: This method also uses the same initial environment as ours but relies on a particle filter to sample environments.

How does our approach generalize to unseen race tracks? We first evaluated our method on six unseen tracks to assess its generalization ability across tracks with varying numbers of gates and different layouts compared to those used during training. Table I presents the performance results. As can be observed, the reference single-track RL policies on the left side demonstrate the best performance in most experiments. This is because these policies overfit the specific track, allowing them to optimize and perform well in that particular environment. Consequently, the results

Track Type	Track Name	Methods									
		Single-track RL		*RL w/o curriculum*		*Particle Filter*		*Domain Randomization*		*Ours*	
		SR [%]	LT [s]	SR [%]	LT [s]	SR [%]	LT [s]	SR [%]	LT [s]	SR [%]	LT [s]
2D	*Figure 8*	100.00	4.263	0.00	-	**100.00**	5.128	**100.00**	6.205	**100.00**	**4.746**
	Kidney	100.00	4.260	0.00	-	0.00	-	**100.00**	4.984	**100.00**	**4.943**
	Big S	100.00	9.245	0.00	-	0.00	-	0.00	-	**100.00**	**7.513**
3D	3D *Figure 8*	100.00	5.010	0.00	-	**100.00**	**5.654**	0.00	-	**100.00**	5.856
	3D *Big S*	100.00	10.187	0.00	-	0.00	-	0.00	-	**100.00**	**9.761**
	Twist	100.00	7.444	0.00	-	0.00	-	0.00	-	**100.00**	**10.199**

TABLE I: We compare the success rate (SR) and lap time (LT) of our method against four baselines. Six different unseen racetracks are evaluated in a realistic BEM simulation [26], including three 2D tracks and three 3D tracks. Here apart from the our and baseline approaches, we also include the *Single-track RL* as a reference, which is a vanilla PPO agent trained and tested on individual track.

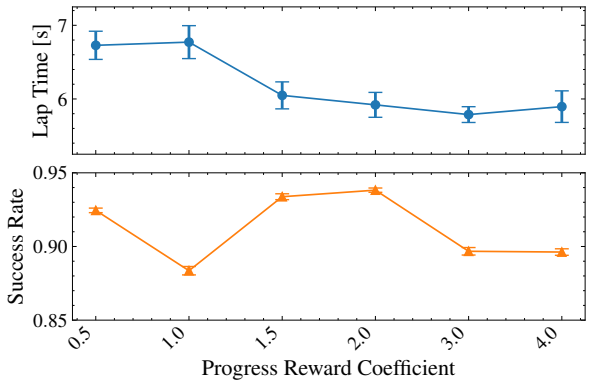


Fig. 4: Ablation study on the progress reward. Due to the fluctuations in evaluation results across different iterations, to fairly compare the performance of different coefficients, we take the average and variance of the success rate and lap time after the model has stabilized for comparison.

from the RL method without a curriculum show that directly training on one track without a curriculum and testing on different tracks is not successful.

Additionally, we found that a simple curriculum has a limited effect on improving the racing policy's generalization ability. Methods such as domain randomization and particle filters can enhance the agent's generalization ability, allowing it to fly on some simple unseen tracks, such as *Figure 8* or *Kidney* tracks. However, since these methods do not take into account the continuous improvement of the agent's policy during training or the evolution of the environment, they fail to perform well in rather complicated unseen tracks, e.g., in *2D Big S* or *Twist* tracks. Only our proposed method demonstrated stable and successful flight across all six unseen tracks. **How does our environment policy perform on different training configurations?** To assess the impact of different configurations on the performance of our environment policy in racing policy training, we performed an ablation study. Within the same parameter setting for the environment policy, we vary the progress reward coefficients α_1 from Equation 2 of the racing policy. As shown in Fig. 4, within a large range of parameters, increasing the progress reward coefficient helps the agent learn to fly faster, as evidenced by the decreasing lap times on specific tracks. At the same time, we observe no significant decrease in generalization, as the success rate remains fairly stable. This indicates the robustness of our framework, where a dedicated human-defined curriculum usually needs to be re-designed for different speed ranges.

Does our policy transfer in the real world? To validate the effectiveness of our proposed method, we conducted tests in real-world conditions. We used the Agilicious quadrotor platform [27] with precise state estimation provided by a VICON motion capture system, ensuring accurate inputs for the policy. The BetaFlight2 firmware was employed for low-level control to execute the collective thrusts and body rate commands. We performed nine laps on each of the six tracks (Fig. 3), demonstrating that our single racing policy can successfully navigate these previously unseen tracks in the real world with a success rate of 100%, as shown on the right side of Fig. 1. For further details, we invite readers to view the supplementary video.

V. Conclusion

In this work, we proposed an adaptive environment-shaping framework enabling, for the first time, a learned policy to race on unseen and dynamic tracks. Our method works by leveraging a secondary environment policy that shapes the training environments for a drone racing policy. Using our novel relative ranking reward, our environment policy can generate track layouts that are challenging but feasible based on the racing agents' performance. One single drone racing policy trained with our framework can race in various race tracks with different complexity with 100% success rate, whereas the state-of-art curriculum learning approach mostly cannot fly. Furthermore, we validated the policy's generalization ability by racing on a track with moving gates, where existing methods performed significantly worse at adapting to the fast-changing gate positions. We believe our work represents a significant advancement in enabling agile robots to achieve greater generalization and robustness in complex, dynamic, and open environments.

References

[1] O. M. Andrychowicz, B. Baker, M. Chociej, R. Jozefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, *et al.*, "Learning dexterous in-hand manipulation," *The International Journal of Robotics Research*, vol. 39, no. 1, pp. 3–20, 2020.

[2] D. Kalashnikov, A. Irpan, P. Pastor, J. Ibarz, A. Herzog, E. Jang, D. Quillen, E. Holly, M. Kalakrishnan, V. Vanhoucke, *et al.*, "Scalable deep reinforcement learning for vision-based robotic manipulation," in *Conference on robot learning*, pp. 651–673, PMLR, 2018.

[3] D. Kalashnikov, J. Varley, Y. Chebotar, B. Swanson, R. Jonschkowski, C. Finn, S. Levine, and K. Hausman, "Scaling up multi-task robotic reinforcement learning," in *Conference on Robot Learning*, 2022.

[4] E. Aljalbout, F. Frank, M. Karl, and P. van der Smagt, "On the role of the action space in robot manipulation learning and sim-to-real transfer," *IEEE Robotics and Automation Letters*, 2024.

[5] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning quadrupedal locomotion over challenging terrain," *Science robotics*, vol. 5, no. 47, p. eabc5986, 2020.

[6] A. Kumar, Z. Fu, D. Pathak, and J. Malik, "Rma: Rapid motor adaptation for legged robots," *Proceedings of Robotics: Science and Systems (RSS)*, 2021.

[7] Y. Song, M. Steinweg, E. Kaufmann, and D. Scaramuzza, "Autonomous drone racing with deep reinforcement learning," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1205–1212, IEEE, 2021.

[8] N. Messikommer, Y. Song, and D. Scaramuzza, "Contrastive initial state buffer for reinforcement learning," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2024.

[9] J. Xing, L. Bauersfeld, Y. Song, C. Xing, and D. Scaramuzza, "Contrastive learning for enhancing robust scene transfer in vision-based agile flight," in *2024 IEEE international conference on robotics and automation (ICRA)*, IEEE, 2024.

[10] M. Krinner, E. Aljalbout, A. Romero, and D. Scaramuzza, "Accelerating model-based reinforcement learning with state-space world models," *arXiv preprint arXiv:2502.20168*, 2025.

[11] A. Romero, E. Aljalbout, Y. Song, and D. Scaramuzza, "Actor-critic model predictive control: Differentiable optimization meets reinforcement learning," *arXiv preprint arXiv:2306.09852*, 2024.

[12] 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," 2017.

[13] J. Tobin, L. Biewald, R. Duan, M. Andrychowicz, A. Handa, V. Kumar, B. McGrew, J. Schneider, P. Welinder, W. Zaremba, and P. Abbeel, "Domain randomization and generative models for robotic grasping," 2018.

[14] OpenAI, M. Andrychowicz, B. Baker, M. Chociej, R. Jozefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, J. Schneider, S. Sidor, J. Tobin, P. Welinder, L. Weng, and W. Zaremba, "Learning dexterous in-hand manipulation," 2019.

[15] OpenAI, I. Akkaya, M. Andrychowicz, M. Chociej, M. Litwin, B. McGrew, A. Petron, A. Paino, M. Plappert, G. Powell, R. Ribas, J. Schneider, N. Tezak, J. Tworek, P. Welinder, L. Weng, Q. Yuan, W. Zaremba, and L. Zhang, "Solving rubik's cube with a robot hand," 2019.

[16] Y. Kong, L. Liu, J. Wang, and D. Tao, "Adaptive curriculum learning," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 5067–5076, 2021.

[17] R. Portelas, C. Colas, K. Hofmann, and P.-Y. Oudeyer, "Teacher algorithms for curriculum learning of deep rl in continuously parameterized environments," 2019.

[18] J. Parker-Holder, M. Jiang, M. Dennis, M. Samvelyan, J. Foerster, E. Grefenstette, and T. Rocktäschel, "Evolving curricula with regret-based environment design," 2023.

[19] M. Jiang, M. Dennis, J. Parker-Holder, J. Foerster, E. Grefenstette, and T. Rocktäschel, "Replay-guided adversarial environment design," 2022.

[20] M. Dennis, N. Jaques, E. Vinitsky, A. Bayen, S. Russell, A. Critch, and S. Levine, "Emergent complexity and zero-shot transfer via unsupervised environment design," 2021.

[21] D. Hanover, A. Loquercio, L. Bauersfeld, A. Romero, R. Penicka, Y. Song, G. Cioffi, E. Kaufmann, and D. Scaramuzza, "Autonomous drone racing: A survey," *IEEE Transactions on Robotics*, 2024.

[22] Y. Zhou, C. Barnes, J. Lu, J. Yang, and H. Li, "On the continuity of rotation representations in neural networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5745–5753, 2019.

[23] Y. Song, M. Steinweg, E. Kaufmann, and D. Scaramuzza, "Autonomous drone racing with deep reinforcement learning," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1205–1212, 2021.

[24] I. Geles, L. Bauersfeld, A. Romero, J. Xing, and D. Scaramuzza, "Demonstrating agile flight from pixels without state estimation," in *Proceedings of Robotics: Science and Systems*, 2024.

[25] J. Xing, A. Romero, L. Bauersfeld, and D. Scaramuzza, "Bootstrapping reinforcement learning with imitation for vision-based agile flight," *arXiv preprint arXiv:2403.12203*, 2024.

[26] L. Bauersfeld, E. Kaufmann, P. Foehn, S. Sun, and D. Scaramuzza, "Neurobem: Hybrid aerodynamic quadrotor model," in *Proceedings of Robotics: Science and Systems*, 2021.

[27] P. Foehn, E. Kaufmann, A. Romero, R. Penicka, S. Sun, L. Bauersfeld, T. Laengle, G. Cioffi, Y. Song, A. Loquercio, and D. Scaramuzza, "Agilicious: Open-source and open-hardware agile quadrotor for vision-based flight," *Science Robotics*, vol. 7, no. 67, 2022.