Dream to Fly: Model-Based Reinforcement Learning for Vision-Based Drone Flight

Angel Romero*, Ashwin Shenai*, Ismail Geles, Elie Aljalbout, Davide Scaramuzza

***Abstract*— Autonomous drone racing has risen as a challenging robotic benchmark for testing the limits of learning, perception, planning, and control. Expert human pilots are able to fly a drone through a race track by mapping pixels from a single camera directly to control commands. Recent works in autonomous drone racing attempting direct pixel-to-commands control policies have relied on either intermediate representations that simplify the observation space or performed extensive bootstrapping using Imitation Learning (IL). This paper leverages DreamerV3 to train visuomotor policies capable of agile flight through a racetrack using only pixels as observations. In contrast to model-free methods like PPO or SAC, which are sample-inefficient and struggle in this setting, our approach acquires drone racing skills from pixels. Notably, a perception-aware behavior of actively steering the camera toward texture-rich gate regions emerges without the need for handcrafted reward terms for the viewing direction. Our experiments show, in both simulation and real-world flight using a *hardware-in-the-loop* setup with rendered image observations, how the proposed approach can be deployed on real quadrotors at speeds of up to 9 m/s. These results advance the state of pixel-based autonomous flight and demonstrate that MBRL offers a promising path for real-world robotics research. Video: https://www.youtube.com/watch?v=nctQ2rxZnIc**

I. INTRODUCTION

In recent years, quadrotors have become a central focus of robotics research, emerging as versatile platforms with untapped potential across multiple domains, including search and rescue, inspection, agriculture, cinematography, delivery, passenger air vehicles, space exploration [1] and drone racing [2]. The drone racing domain not only benefits from cutting-edge robotics research [3] but also pushes the limits of what is possible by challenging these flying machines to outperform the most skilled human pilots, as shown by recent successes against the world's best pilots [4], [5]. Closing the loop between perception and control, learning directly from pixels to actions without the need for explicit state estimation, remains largely unfulfilled. One of the most recent works in this domain [6] manages to train a Proximal Policy Optimization (PPO) [7] policy to fly a drone through a race track from binary image representations. However, this work relies on a simplified observation space: the raw visual input is preprocessed into a binary mask highlighting only the gates. This simplification, while reducing the complexity of the observation space and facilitating learning, also discards valuable information. In [8] the authors present another recent work that tackles quadrotor flight from pixel observations. However, it relies on IL bootstrapped from a pre-trained expert policy, which had access to the full state and introduces a strong dependency on privileged information. Prior MBRL work in real-world robotics has largely relied on state estimation rather than pixel observations [9], [10], or has been limited to fixed-viewpoint [11] or low-dynamics settings [12].

Fig. 1. Real-world deployment of our DreamerV3 policy in the Figure 8 track. During training, the agent learns a world model from interactions with the environment. At the same time, the actor-critic policy is trained by sampling the predictions of the world model, also called *imagination*. The rendered images consumed by the network are marked in red.

We build on DreamerV3 [13] to demonstrate its effectiveness in vision-based robotics tasks that require egocentric perception and agile control, such as drone racing. Our contributions are as follows:

- **Pixel-to-command MBRL system for quadrotor flight:** We train quadrotor policies from scratch, without relying on intermediate representations or bootstrapping from imitation learning, which were necessary in previous works.
- **Emergent perception-aware behaviour:** the drone's camera view is naturally guided towards feature-rich areas such as the next gates. This behavior arises directly from our end-to-end optimization from pixels to commands, without requiring handcrafted reward terms that were common in previous methods.
- **Real-world policy transfer:** We validate our learned policies by deploying them on a physical quadrotor with a *hardware-in-the-loop* (HIL) setup.

*These authors contributed equally. The authors are with the Robotics and Perception Group, Department of Informatics, University of Zurich, Switzerland (http://rpg.ifi.uzh.ch). This work was supported by the European Union's Horizon Europe Research and Innovation Programme under grant agreement No. 101120732 (AUTOASSESS) and the European Research Council (ERC) under grant agreement No. 864042 (AGILE-FLIGHT).

II. METHODOLOGY

A. Problem Statement

Formally, we seek a policy $\pi(x)$ that maps raw visual observations x directly to control commands a, minimizing the time required to complete the course. We frame the problem within the Partially Observable Markov Decision Process (POMDP) framework, defined by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{X}, P, R, \gamma, \Omega)$, where $\mathcal{S}$ is the set of latent states, $\mathcal{A}$ is the set of actions, $\mathcal{X}$ is the set of observations, $P(s_{k+1} \mid s_k, a_k)$ is the state transition probability, $R(s_k, a_k)$ is the reward function, $\gamma \in [0, 1)$ is the discount factor, and $\Omega(x_k \mid s_k)$ is the observation model. In our vision-based drone racing task, the latent state s_k is not observed directly; instead, the agent receives the RGB image observation $x_k \in \mathcal{X}$ rendered at time step k. The action a_k consists of the collective thrust and body rates applied to the quadrotor. The reward function is designed to incentivize fast and collision-free navigation through the gates (detailed in Section II-C). The objective is to find an optimal policy $\pi_\theta^* : \mathcal{X} \to \mathcal{A}$ that maximizes the expected cumulative discounted reward by optimizing the parameters θ of a neural network,

$$\pi_\theta^* = \arg\max_{\pi} \mathbf{E}\left[\sum_{k=0}^{\infty} \gamma^k r_k\right].$$

B. Observation and Action Spaces

1) Observation Space: The observation space consists of RGB images captured at each time step, denoted as $\mathbf{x}_k \in \mathbb{R}^{H \times W \times 3}$, where H and W represent the image height and width, respectively, and 3 corresponds to the RGB color channels. These images are normalized to the range $[0, 1]$ by dividing pixel values by 255.

2) Action Space: At each time step k, the policy outputs a four-dimensional action vector $\mathbf{a}_k = [c, \omega_x, \omega_y, \omega_z]$, where c represents the mass-normalized collective thrust, and $\omega_x, \omega_y, \omega_z$ are the body rate setpoints in the drone's body frame. These actions are expressed in the Collective Thrust and Body Rates (CTBR) format, a control interface commonly used by professional drone pilots [4], [14]. To ensure bounded and physically realizable control actions, the action space is constrained to $\mathcal{A} = [-1, 1]^4$, which are then mapped to the actual collective thrust and body rates limits. This constraint is enforced by applying a hyperbolic tangent (tanh) activation function to the output of the actor policy network.

C. Reward Function

The racing track is defined as a sequence of linearly connected waypoints placed at the center of the gates. The reward function is designed to incentivize progress along this track while penalizing undesirable behaviors. The reward at time step k, $r(k)$, is defined as

$$r_k = \begin{cases} r_{collision}, & \text{if collision;} \\ r_{passed}, & \text{if gate passed;} \\ b_1(\|\mathbf{g}_k - \mathbf{p}_{k-1}\| - \|\mathbf{g}_k - \mathbf{p}_k\|) & \\ \quad - b_2\|\boldsymbol{\omega}_k\|, & \text{otherwise,} \end{cases} \tag{1}$$

where $\mathbf{g}_k$ is the center of the target gate, $\mathbf{p}_k$ and $\mathbf{p}_{k-1}$ are the drone's positions at the current and previous time steps, and $\boldsymbol{\omega}_k$ is the body rate vector. The primary component of the reward, $b_1(\|\mathbf{g}_k - \mathbf{p}_{k-1}\| - \|\mathbf{g}_k - \mathbf{p}_k\|)$, is called the progress term, as it encourages the drone to move closer to the target gate. The term $b_2\|\boldsymbol{\omega}_k\|$ penalizes excessive body rates. We use the coefficients $b_1 = 1.0, b_2 = 0.01$. In addition to the continuous reward component, discrete rewards are provided for specific events, e.g., $r_{collision} = -4.0$ and $r_{passed} = +10.0$.

D. Model-based Reinforcement Learning: DreamerV3

DreamerV3 [13] is an off-policy, model-based reinforcement learning algorithm structured in two main blocks: the **world model**, and the **actor-critic policy**. These two blocks are trained in an alternating fashion using an experience replay buffer while the agent interacts with the environment.

1) World Model training: Our approach employs a world model that captures the state transition dynamics in a compact latent space. Specifically, we consider a latent state $\boldsymbol{s}_k = (\boldsymbol{h}_k, \boldsymbol{z}_k)$ and model the state transition probability $P(\boldsymbol{s}_{k+1} \mid \boldsymbol{s}_k, \boldsymbol{a}_k)$. The world model is implemented as a Recurrent State Space Model (RSSM) [15], and consists of the following components:

- **Encoder:** An encoder network maps raw observations $\boldsymbol{x}_k$ into a stochastic latent representation $\boldsymbol{z}_k$.

$$\boldsymbol{z}_k \sim q_\phi(\boldsymbol{z}_k \mid \boldsymbol{h}_k, \boldsymbol{x}_k). \tag{2}$$

- **Recurrent Sequence Model:** A recurrent sequence model, parameterized by the recurrent state $\boldsymbol{h}_k$, predicts the evolution of the latent representation. Given the previous latent state $[h_{k-1}, z_{k-1}]$ and the action $\boldsymbol{a}_{k-1}$, it predicts the current recurrent state $\boldsymbol{h}_k$.

$$\boldsymbol{h}_k = f_\phi(\boldsymbol{h}_{k-1}, \boldsymbol{z}_{k-1}, \boldsymbol{a}_{k-1}). \tag{3}$$

- **Dynamics, Reward and Continue Prediction:** The dynamics predictor predicts the stochastic state $\hat{\boldsymbol{z}}_k$ given the recurrent state $\boldsymbol{h}_k$. The reward and continue predictors are conditioned on the latent state $\boldsymbol{s}_k = (\boldsymbol{h}_k, \boldsymbol{z}_k)$, and predict the immediate reward r_k and the episode continuation flag $c_k \in \{0, 1\}$.

$$\text{Dynamics predictor:} \quad \hat{\boldsymbol{z}}_k \sim p_\phi(\hat{\boldsymbol{z}}_k \mid \boldsymbol{h}_k) \tag{4}$$

$$\text{Reward predictor:} \quad \hat{\boldsymbol{r}}_k \sim p_\phi(\hat{\boldsymbol{r}}_k \mid \boldsymbol{h}_k, \boldsymbol{z}_k) \tag{5}$$

$$\text{Continue predictor:} \quad \hat{\boldsymbol{c}}_k \sim p_\phi(\hat{\boldsymbol{c}}_k \mid \boldsymbol{h}_k, \boldsymbol{z}_k). \tag{6}$$

- **Decoder:** A decoder reconstructs the original observations from the latent representations.

$$\hat{\boldsymbol{x}}_k \sim p_\phi(\hat{\boldsymbol{x}}_k \mid \boldsymbol{h}_k, \boldsymbol{z}_k). \tag{7}$$

The encoder and decoder use convolutional neural networks (CNNs) for image inputs and multi-layer perceptrons (MLPs) for vector inputs. These models are trained by minimizing the prediction loss $\mathcal{L}_{pred}(\phi)$, the dynamics loss $\mathcal{L}_{dyn}(\phi)$, and the representation loss $\mathcal{L}_{rep}(\phi)$. For more details about these losses, we refer the reader to [13].

2) Actor-Critic Training: Our approach employs an actor-critic architecture trained using imagined trajectories generated by a learned world model. Given a starting state representation, the actor generates an imagined trajectory consisting of model states $(\mathbf{h}_{1:T}, \mathbf{z}_{1:T})$, actions $\mathbf{a}_{1:T}$, rewards $r_{1:T}$, and continuation flags $c_{1:T}$. The critic then learns to evaluate the quality of these imagined trajectories by predicting the distribution of bootstrapped λ-returns. For further details on the actor-critic training and loss functions, we refer the reader to [13]. In our case, the prediction horizon for the imagined trajectories is set to $T = 16$.

III. Simulation Experiments

We implemented DreamerV3 in PyTorch, leveraging the dreamerv3-torch open-source implementation and building on top of the stable-baselines3 library [16]. Our experiments were conducted within a high-fidelity simulation environment combining Flightmare [17] and Agilicious [18] for realistic quadrotor dynamics and track generation. To enable fast, real-time rendering and direct access to the image feed during training, we integrated the Habitat simulator [19], [20], enabling training with the renderer in the loop at several thousand frames per second.

Our method's performance is benchmarked against two model-free baselines: Proximal Policy Optimization (PPO) [4]–[6], [21] and Soft Actor Critic (SAC) [22]. Both the PPO and SAC baselines used a CNN followed by four-layer Multi-Layer Perceptrons (MLPs) with 768 neurons per layer for both the actor and critic networks. The physical parameters of the quadrotor used were consistent across all experiments: mass $m = 0.6$ kg, diagonal inertia matrix $J = \text{diag}([0.002410, 0.001800, 0.003759])$ kg m^2, rotor torque constant $\kappa = 0.022$, and arm length 0.14 m. The maximum rotor thrust was limited to 4.0 N, resulting in a thrust-to-weight ratio of 2.7. The dynamics, platform and parameters are the same one as in [6].

For DreamerV3, we adopted the hyperparameters detailed in [13] and used the *Large (L)* network configuration described in Appendix B of [23]. This configuration consists of four-layer MLPs with 768 units for the decoder, predictors, actor, and critic networks, and 2048 recurrent units for the world model's recurrent component.

All experiments were performed on a single Quadro RTX 8000. Input images from the simulated camera were resized to 64×64 RGB pixels before being fed into the DreamerV3 agent.

A. Results

We conducted training experiments on our agent using three distinct reinforcement learning algorithms: DreamerV3, PPO and SAC. These experiments were carried out across different challenging drone racing tracks: Kidney, and Figure 8. The results of these experiments are visualized in Figure 2. Each run was conducted with 5 different random seeds. The resulting plots depict the average reward across these 5 runs, represented by a solid center line, and the shaded area surrounding the line indicates the standard deviation. Our findings indicate that DreamerV3, by leveraging a learned world model, effectively learns directly from pixel observations. In contrast, PPO, and similarly SAC, which rely solely on direct interaction with the environment, struggle to converge to high-reward solutions. Furthermore, the model-free policies fail to execute any meaningful flight maneuvers, resulting in consistently low reward values. An important observation is that our system is trained using curriculum learning, where the parameter b_2 in Eq. (1) is initially set to 0.0 and gradually increases once the total reward surpasses 50.0. We note, however, that this comes at the cost of a significantly longer training time than the baselines, reaching 240 hours.

Figure 3 presents a comparative analysis of real observations and their corresponding reconstructions for the Figure 8 track at two distinct time steps, t_1 and t_2. The reconstructions are sampled at three different training stages: early training (0.4 million steps), mid-training (1 million steps), and late-stage training (10 million steps). A visual inspection of the figure reveals a progressive improvement in reconstruction quality as training progresses.

B. Perception-aware emergent behaviour

An additional interesting observation is that after training, the agent exhibits a consistent tendency to orient the camera towards the gates, a behavior that naturally emerged during training rather than being explicitly incentivized by the reward function, as it is generally done in previous works [6], [8]. This emergent behavior can be attributed to the fact that we are optimizing end-to-end from pixels to commands, therefore allowing for closing the action-perception loop. As depicted in Figure 3, the gates remain visually clear throughout the entire track, while the background details become increasingly blurred due to the downsampling of the input image to 64×64 pixels. This suggests that the policy strategically prioritizes focusing on the information-rich gates, which are essential for successful navigation. To reinforce this hypothesis, we have conducted an additional ablation experiment where we place two additional gates in the periphery of the Figure 8 track. These gates are there only in the rendering, but do not need to be passed through. Fig. 2B shows the same policy trained with and without these additional gates. In the supplementary video we show the first person rendered view of the two policies depicted in Fig. 2B deployed in a simulated environment.

IV. Real-world Experiments

The software setup is identical to the one used in the simulation experiments, explained in Section III. Regarding the hardware, we use a modification of the *Agilicious* platform [18] for the real-world deployment. We have replaced the

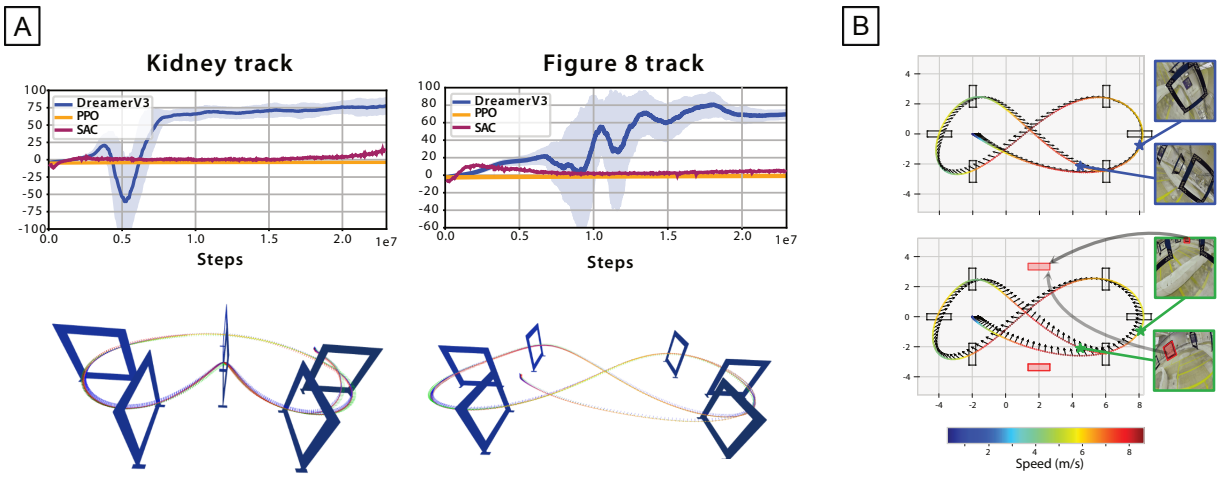

Fig. 2. **A:** Reward evolution over training steps for the Kidney and Figure 8 tracks. DreamerV3 (blue) converges to high-reward policies on both tracks, while PPO (orange) and SAC (red) fail to achieve any considerable training within 20 million environment interactions. **B:** Ablation study on the perception-aware behaviour of our policies. **B**-Top: DreamerV3 policy trained on pixel observations in an environment where the only rendered gates are the actual gates. As indicated by the black arrows (representing camera direction), the platform predominantly focuses on the next gate. **B**-Bottom: We introduce two additional gates to the rendering engine (marked in red). These gates are not required to be traversed, and are not part of the objective, but serve as a valuable source of information for platform localization. Consequently, the policy's behavior shifts, and the platform now distributes its camera view more evenly across both the actual and the extra gates.

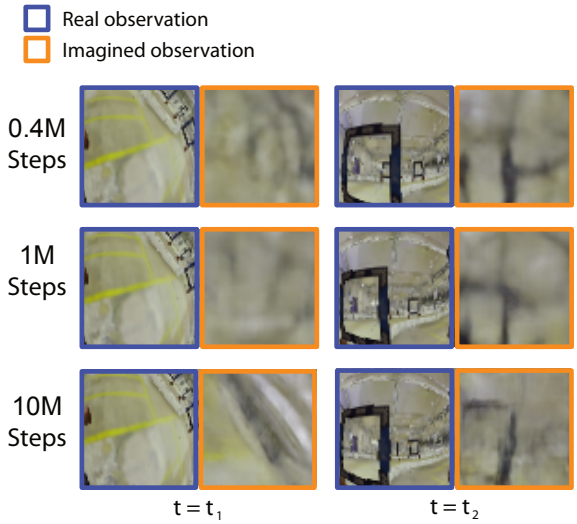

Fig. 3. Comparison of real observations and imagined observations for the Figure 8 track. Imagined observations are observations that are reconstructed by the world model. The figure shows the reconstructed observations after 0.4M steps (early stage training), 1M timesteps (mid stage training), and after 10M steps (training convergence). One can observe how the reconstruction gets better and better as the training evolves.

onboard computer with an RF receiver, which is connected directly to the *Betaflight* flight controller and takes care of parsing the collective thrust and bodyrate commands from the offboard computer. Our hardware setup is the same setup as in [6], similar to the one used by professional drone racing pilots. For the deployment in the real world, we use a hardware-in-the-loop (HIL) setup, where the images are rendered using the habitat simulator and fed into the network, and the commands produced are sent directly to the real drone platform. This way, we have the real world dynamics in the loop, which allows us to asses the sim-to-real gap and our policy performance when tested on the real system. We demonstrate the efficacy of our policies by deploying them in the real world on the Figure 8 track. This experiment is shown in the supplementary material video at https://www.youtube.com/watch?v=nctQ2rxZnIc. We observe a strong similarity between the simulated and real-world trajectories, indicating a minimal sim-to-real gap in our dynamic model, reaching speeds of up to 9 m/s. Moreover, the camera axis aligns closely between the simulated and real-world scenarios. To the best of our knowledge, our HIL demonstration marks the first RL approach to learn drone racing directly from pixel inputs to control commands, effectively closing the loop between perception and action.

V. Conclusion

This paper presented a MBRL approach using DreamerV3 to train end-to-end visuomotor policies for agile drone flight. Our method learns directly from raw pixel inputs, removing the need for intermediate representations or imitation learning bootstrapping. Our experiments demonstrated that this approach is more sample-efficient than model-free baselines like PPO and SAC and results in emergent perception-aware behaviors without explicit reward shaping.

We validated the learned policies in simulation and on a physical quadrotor using a hardware-in-the-loop (HIL) setup. While this confirms the policy's effectiveness on the real quadrotor dynamics, additional challenges remain to transfer to real camera pixel observations. Additionally, the training process is computationally intensive, requiring approximately 240 hours to converge. Future work should focus on using photorealistic simulators or exploring other observation modalities that allow for deployment directly on real pixels, and on improving the computational efficiency of the world model. Overall, our work is a promising step towards applying pixel-to-command MBRL on real-world robotic systems.

REFERENCES

[1] S. A. H. Mohsan, N. Q. H. Othman, Y. Li, M. H. Alsharif, and M. A. Khan, "Unmanned aerial vehicles (UAVs): practical aspects, applications, open challenges, security issues, and future trends," *Intelligent Service Robotics*, vol. 16, pp. 109–137, 2023.

[2] 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*, vol. 40, p. 3044–3067, 2024.

[3] H. Moon, J. Martinez-Carranza, T. Cieslewski, M. Faessler, D. Falanga, A. Simovic, D. Scaramuzza, S. Li, M. Ozo, C. De Wagter *et al.*, "Challenges and implemented technologies used in autonomous drone racing," *Intelligent Service Robotics*, 2019.

[4] E. Kaufmann, L. Bauersfeld, A. Loquercio, M. Müller, V. Koltun, and D. Scaramuzza, "Champion-level drone racing using deep reinforcement learning," *Nature*, vol. 620, no. 7976, pp. 982–987, 2023.

[5] Y. Song, A. Romero, M. Müller, V. Koltun, and D. Scaramuzza, "Reaching the Limit in Autonomous Racing: Optimal Control versus Reinforcement Learning," *Science Robotics*, vol. 8, no. 82, p. eadg1462, 2023.

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

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

[8] J. Xing, A. Romero, L. Bauersfeld, and D. Scaramuzza, "Bootstrapping Reinforcement Learning with Imitation for Vision-Based Agile Flight," *8th Conference on Robot Learning (CoRL)*, 2024.

[9] P. Becker-Ehmck, M. Karl, J. Peters, and P. van der Smagt, "Learning to fly via deep model-based reinforcement learning," *arXiv preprint arXiv:2003.08876*, 2020.

[10] N. O. Lambert, D. S. Drew, J. Yaconelli, S. Levine, R. Calandra, and K. S. Pister, "Low-level control of a quadrotor with deep model-based reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 4224–4230, 2019.

[11] T. Bi and R. D'Andrea, "Sample-Efficient Learning to Solve a Real-World Labyrinth Game Using Data-Augmented Model-Based Reinforcement Learning," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 7455–7460.

[12] P. Wu, A. Escontrela, D. Hafner, P. Abbeel, and K. Goldberg, "DayDreamer: World Models for Physical Robot Learning," in *Conference on Robot Learning (CoRL)*. PMLR, 2022.

[13] D. Hafner, J. Pasukonis, J. Ba, and T. Lillicrap, "Mastering Diverse Domains through World Models," *arXiv preprint arXiv:2301.04104*, 2023.

[14] C. Pfeiffer and D. Scaramuzza, "Human-piloted drone racing: Visual processing and control," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3467–3474, 2021.

[15] D. Hafner, T. Lillicrap, I. Fischer, R. Villegas, D. Ha, H. Lee, and J. Davidson, "Learning Latent Dynamics for Planning from Pixels," in *Proceedings of the 36th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, K. Chaudhuri and R. Salakhutdinov, Eds., vol. 97. PMLR, 09–15 Jun 2019, pp. 2555–2565.

[16] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, "Stable-Baselines3: Reliable Reinforcement Learning Implementations," *Journal of Machine Learning Research*, vol. 22, no. 268, pp. 1–8, 2021.

[17] Y. Song, S. Naji, E. Kaufmann, A. Loquercio, and D. Scaramuzza, "Flightmare: A flexible quadrotor simulator," in *Conference on Robot Learning*, 2020.

[18] P. Foehn, E. Kaufmann, A. Romero, R. Penicka, S. Sun, L. Bauersfeld, T. Laengle, G. Cioffi, Y. Song, A. Loquercio *et al.*, "Agilicious: Open-source and open-hardware agile quadrotor for vision-based flight," *Science Robotics*, vol. 7, no. 67, p. eabl6259, 2022.

[19] M. Savva, A. Kadian, O. Maksymets, Y. Zhao, E. Wijmans, B. Jain, J. Straub, J. Liu, V. Koltun, J. Malik *et al.*, "Habitat: A platform for embodied ai research," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 9339–9347.

[20] X. Puig, E. Undersander, A. Szot, M. D. Cote, T.-Y. Yang, R. Partsey, R. Desai, A. Clegg, M. Hlavac, S. Y. Min, V. Vondruš, T. Gervet, V.-P. Berges, J. M. Turner, O. Maksymets, Z. Kira, M. Kalakrishnan, J. Malik, D. S. Chaplot, U. Jain, D. Batra, A. Rai, and R. Mottaghi, "Habitat 3.0: A Co-Habitat for Humans, Avatars, and Robots," in *The Twelfth International Conference on Learning Representations*, 2024.

[21] 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)*. IEEE, 2021, pp. 1205–1212.

[22] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International Conference on Machine Learning*. PMLR, 2018, pp. 1861–1870.

[23] D. Hafner, T. P. Lillicrap, J. Ba, and M. Norouzi, "Dream to Control: Learning Behaviors by Latent Imagination," in *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*, 2020.