This paper has been accepted for publication at the
IEEE International Conference on Robotics and Automation (ICRA), Vienna 2026. ©IEEE

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 behaviour of actively steering the camera toward texture-rich gate regions emerges without the need of 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].

Traditionally, most autonomous drone racing systems have relied on explicit state estimation integrating data from inertial measurement units (IMUs) and other onboard sensors to maintain stability and optimize performance [4], [6]–[9]. However, professional human pilots rely solely on visual feedback from a single onboard camera, showcasing a remarkable ability to navigate complex environments purely from visual inputs. Emulating this human ability to fly based solely on visual information remains a significant challenge for autonomous systems.

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.

Closing the loop between perception and control – learning directly from pixels to actions, without the need for explicit state estimation – remains largely unfulfilled. While reinforcement learning (RL) has shown promise in various robotic applications, applying it to vision-based tasks introduces unique difficulties in robotics.

One of the most recent works in this domain [10] manages to train a Proximal Policy Optimization (PPO) [11] policy to fly a drone through a race track from binary image representations. In this case, the visual inputs are first preprocessed and distilled into a simpler intermediate representation in the form of a binary mask where only the racing gates are visible. This intermediate representation reduces the overall observation space complexity and allows the PPO policy to learn the behavior efficiently. 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. For instance, excluding background information can hinder the agent's ability to navigate when no gates are immediately visible, or to obtain the gravity direction from the horizon line, for example. In [12] 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

*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 (AGILEFLIGHT).

access to the full state and introduces a strong dependency on privileged information. Despite their success, one major challenge in robot learning is the need for large amounts of physical interaction data, which can be very expensive and challenging to obtain. This challenge is compounded in vision-based control tasks, where the high dimensionality of image observations further increases data requirements.

MBRL presents a promising solution to these challenges because it is generally more sample efficient than its model-free counterpart, reducing the need for extensive environment interactions. MBRL learns the transition dynamics of the system, known as *world model*, and uses it to predict and optimize future actions. It also generalizes well to different tasks as the transition model does not vary vastly across different tasks in the same system description. However, despite the sample efficiency, this approach typically comes at the cost of longer training times, as the system must simultaneously learn a world model and optimize a control policy. Additionally, the inherent complexity of model-based RL architectures, with their multiple interdependent components, often makes them more difficult to tune and optimize.

Recent works in MBRL, such as the DreamerV3 architecture [13], have helped mitigate these issues, offering an approach that is more accessible in terms of tuning and optimization. Although DreamerV3 has shown strong results in controlled settings, its adoption in real-world robotic systems, particularly for pixel-based tasks, remains limited. We build on DreamerV3 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.

Our approach demonstrates that the learned policy effectively controls real-world dynamics directly from rendered pixel observations, and further validates the applicability and potential of MBRL for real-world mobile robotic tasks.

II. RELATED WORK

A. Reinforcement Learning from Pixels

In recent years, RL algorithms have achieved incredible feats in simple environments learning directly from pixels, such as arcade games, used as sample efficient simulators to benchmark the capabilities of AI agents [14]. This success in RL has been repeatedly shown in environments in growing complexity, such as beating human performance in Atari games [15] or in more complex games such as Go [16]. However, these successes have been primarily in environments where the action space is discrete, and not in continuous action spaces. This gap is addressed by the DeepMind Control Suite [17], which presents a diverse set of environments with different observation spaces and different action modalities — both continuous and discrete. However, learning directly from pixels in these scenarios has been shown to be less sample efficient, yielding suboptimal performance when compared to state-based learning, even in simulation environments. Recent efforts have tried to improve this by building on top of intermediate visual representations [10], [18]–[21]. For example, previous work explored different kinds of state representations via various supervised and self-supervised learning methods, such as auto-encoders [18], future predictions [20], contrastive unsupervised representations [21] or semantic segmentation [10], with the objective of reducing the performance difference between state-based and pixel-based RL.

In order to address the challenge of learning complex behaviors efficiently directly from pixels, some methods have focused on data augmentation [22], [23]. Policies that map camera observations directly to commands — also called sensorimotor or visuomotor policies — are mostly learned through extensive data simulation, making use of domain randomization techniques or incorporating additional privileged information such as joint angles [24]–[26] For more complicated tasks, the robotics community has resorted to the usage of expert demonstrations, marking a shift towards imitation learning approaches, contrary to the learning from scratch using RL paradigm. This is showcased by recent work, most notably [27].

B. Model-based RL for Real-World Robotics

As mentioned above, there are many approaches that have attempted to train policies using pixels as observations, relying in simplifying assumptions. However, the landscape of model-based RL (MBRL) methods applied to real-world robotics remains relatively limited, and existing research primarily focuses on simulation environments. Within the context of real-world robotic applications, prior MBRL work often relies on state estimations rather than raw pixel observations. Specifically, there are some works tackling the problem of flying a quadcopter using model-based RL, for high-level control [28] or low-level controller learning [29]. However, these works have been using state estimation as observations, and not directly pixels as observations.

When speaking about works that use MBRL from pixels and that deploy in real-world robots, the number of works is even lower. One of the early works in this direction is [30], where a learned world model of the dynamics is built from pixels, and is then used by an MPC planner to get the optimal policy, which outputs torque commands for a real-world manipulator. Another interesting work that applies DreamerV3 to a real-world robotic system is [31], where the table top labyrinth game is solved in the real world

by using top-view fixed camera in conjunction with the measurement of the table angle. However, this setup does not fall under mobile robotics, as the observations are not egocentric but rather taken from a fixed viewpoint with most of the environment remaining unchanged in the field of view. One of the most salient works that applies DreamerV3 [13] to real-world robotics is [32], where different policies are trained from images, depth and state, for different tasks. These include pick-and-place manipulation utilizing a fusion of RGB images, depth data, and proprioceptive sensing, as well as quadruped locomotion trained solely on proprioceptive sensor data. Notably, the only instance of direct learning from pixels to commands involves a 2D navigation task with a Sphero robot, a relatively simple scenario in terms of dynamics and environmental complexity.

III. 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 address this challenge using reinforcement learning (RL), framing the problem within the Partially Observable Markov Decision Process (POMDP) framework. A POMDP is 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 III-C) The objective in a POMDP 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,

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

This optimization considers both immediate and future rewards, with future rewards discounted by γ.

B. Observation and Action Spaces

The RL framework learns a visuomotor, end-to-end policy that directly maps raw RGB images as observations to control inputs. These images provide a rich and high-dimensional representation of the environment, enabling the agent to infer its state and surroundings without relying on explicitly estimated states.

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. Unlike approaches that rely on explicit state estimation (e.g., position, velocity, or attitude), this image-based approach directly leverages rich visual information to guide the policy.

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], [33]. This format directly commands the low-level actuation of the drone, bypassing the need for intermediate, high-level abstractions such as position or velocity commands. 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 r_k is the reward at timestep k, $\mathbf{g}_k$ 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, respectively, 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 $b_1 = 1.0, b_2 = 0.01$. This formulation directly rewards progress toward the gate center. Importantly, deviations from the exact path defined by the waypoints are not penalized, allowing the agent to discover potentially more efficient trajectories.

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

In this section, we briefly describe the key aspects of the DreamerV3 algorithm. For more details, we refer the reader to the original paper [13]. DreamerV3 is an off-policy, model-based reinforcement learning algorithm. The algorithm is 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

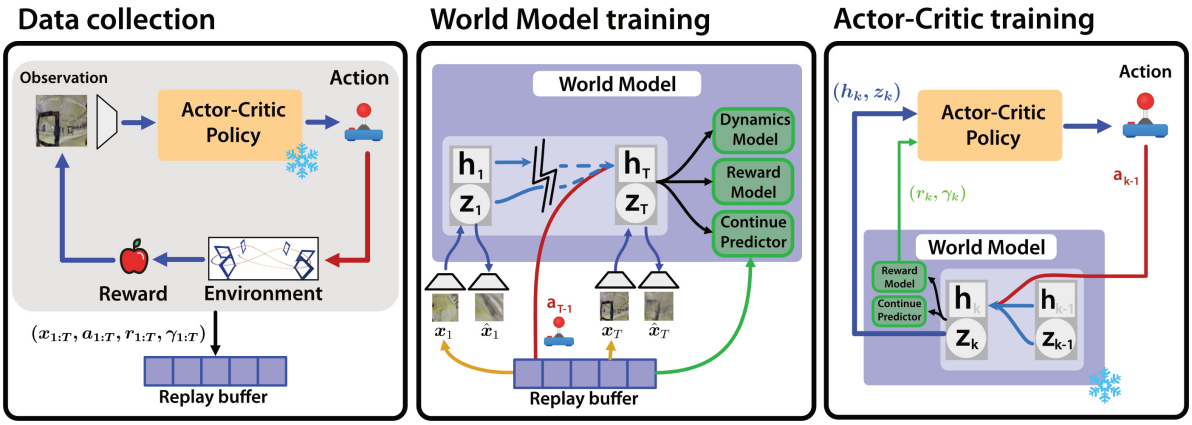

Fig. 2. The process begins with data collection in the simulation environment using the current policy, storing experiences in a replay buffer. This buffer is used to train the world model components: the encoder, decoder, RSSM, dynamics model, reward model, and continue predictor (Section III-D). Subsequently, an actor-critic policy is trained within the learned world model to maximize expected (imagined) returns. This updated policy is then used to collect new data, restarting the loop.

while the agent interacts with the environment. A high level depiction of the training process is described in Fig. 2.

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 $P(\boldsymbol{s}_{k+1} \mid \boldsymbol{s}_k, \boldsymbol{a}_k)$. By encoding high-dimensional sensory inputs into low-dimensional representations, the world model enables predicting future latent states and rewards based on the agent's actions. The world model is implemented as a Recurrent State Space Model (RSSM) [20], and consists of the following components:

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

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

- **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 $c_k \in \{0, 1\}$, indicating whether the episode terminates or continues.

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

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

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

- **Decoder:** A decoder reconstructs the original observations from the latent representations. This reconstruction loss ensures that the latent variables $\boldsymbol{z}_k$ retain essential information from the environment.

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

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 different losses: the prediction loss $\mathcal{L}_{pred}(\phi)$, which trains the decoder, the reward and the continue flags; the dynamics loss $\mathcal{L}_{dyn}(\phi)$, which trains the sequence model; and the representation loss $\mathcal{L}_{rep}(\phi)$, aims to make the representations more predictable. For more details about these losses, we refer the reader to [13].

The overall world model objective is a linear combination of the above defined losses:

$$\mathcal{L}(\phi) = \mathbf{E}_{q_\phi}\left[\sum_{k=1}^{T} \beta_{pred}\mathcal{L}_{pred}(\phi) + \beta_{dyn}\mathcal{L}_{dyn}(\phi) + \beta_{rep}\mathcal{L}_{rep}(\phi)\right],$$

where $\beta_{pred} = 1, \beta_{dyn} = 1, \beta_{rep} = 0.1$ and T is the batch sequence length. The world model is trained by randomly sampling T-length snippets of inputs $\boldsymbol{x}_{1:T}$, actions $\boldsymbol{a}_{1:T}$, rewards $r_{1:T}$ and continuation flags $c_{1:T}$ from episodes in the replay buffer.

By jointly training these components, the RSSM-based world model learns a compact, predictive representation of the environment, and a latent dynamics model, supporting more efficient decision-making and planning in latent space.

2) Actor-Critic Training: Our approach employs an actor-critic architecture trained using imagined trajectories generated by a learned world model (see Fig. 2). This allows the agent to learn complex behaviors without requiring extensive real-world interactions. 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. This bootstrapping allows the critic to estimate long-term returns even within the limited prediction horizon of the world model.

The actor learns to maximize these λ-returns while exploring through an entropy regularizer. To ensure robust exploration across diverse reward scales and frequencies, the returns are normalized to approximately lie in $[0, 1]$. To optimize the policy, DreamerV3 uses the REINFORCE estimator combining unbiased but high-variance policy gradients with a stop-gradient operation on the value targets.

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 $T = 16$. This horizon balances computational cost and the ability to capture longer-term dependencies.

IV. Simulation Experiments

We implemented DreamerV3 in PyTorch, leveraging the dreamerv3-torch open-source implementation and building on top of the stable-baselines3 library [34]. Our experiments were conducted within a high-fidelity simulation environment combining Flightmare [35] and Agilicious [36] 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 [37], [38], enabling training with the renderer in the loop at several thousand frames per second. This same training environment has been successfully employed in prior research [12].

Our method's performance is benchmarked against two model-free baselines: Proximal Policy Optimization (PPO) [4], [5], [10], [39] and Soft Actor Critic (SAC) [40]. 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 [10].

For DreamerV3, we adopted the hyperparameters detailed in [13] and used the *Large (L)* network configuration described in Appendix B of [41]. 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. These input images are visualized in Figure 4.

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 three challenging drone racing tracks: Circle, Kidney, and Figure 8. The results of these experiments are visualized in Figure 3. The top row of the figure presents the overall reward evolution for each track, comparing the performance of both DreamerV3, PPO and SAC. 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. This enables the agent to acquire policies that are capable of racing through each track. 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. This explains the oscillatory reward values observed in Fig. 3, as the system adjusts to the changing rewards.

Our experimental results align with the findings presented in [12], where the authors encountered similar challenges in training end-to-end PPO policies directly from pixel observations. To overcome these difficulties, they adopted a hybrid approach, combining imitation learning with subsequent reinforcement learning fine-tuning. In contrast, using a model-based approach, we are able to completely avoid such additional efforts. We note, however, that this comes at the cost of a significantly longer training time than the baselines, reaching 240 hours.

As illustrated in Figure 2, the DreamerV3 architecture incorporates an observation reconstruction module. This component ensures that the latent state captures sufficient information to accurately represent the observed sensory input. Figure 4 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. Notably, the fine details of the environment, such as the yellow floor lines at time step t_1, become identifiable only in the late-stage reconstructions. Similarly, the shape of the gate at time step t_2 is accurately reconstructed by the model only at late-stage training.

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 [10], [12]. 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 4, the gates remain visually clear throughout the entire track, while the background details

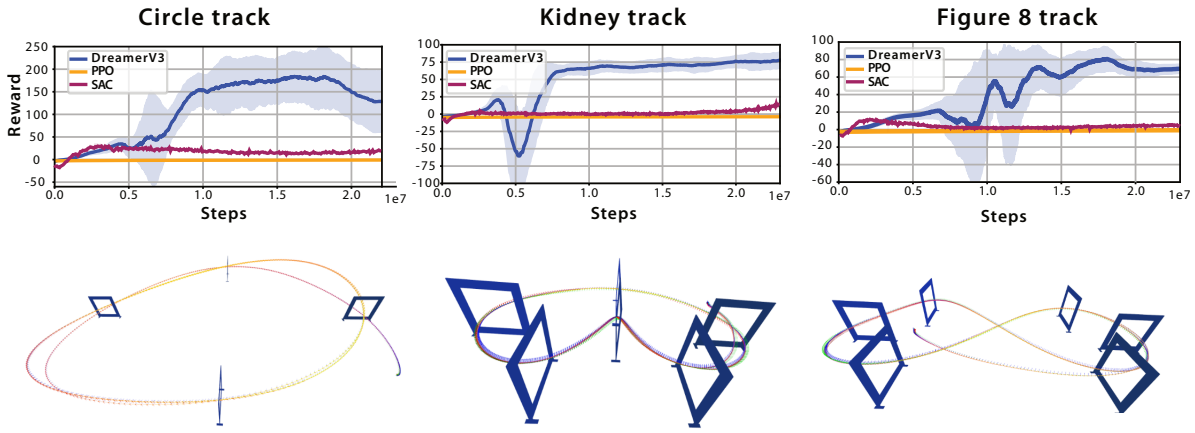

Fig. 3. Reward evolution by number of steps for three different tracks: Circle track, Kidney Track and Figure 8 track. The training performance of DreamerV3 is shown in blue, for PPO in orange and for SAC in red. We show that none PPO nor SAC are able to achieve any considerable training in 20 million environment interactions, while DreamerV3 is able to train to convergence for the three tracks.

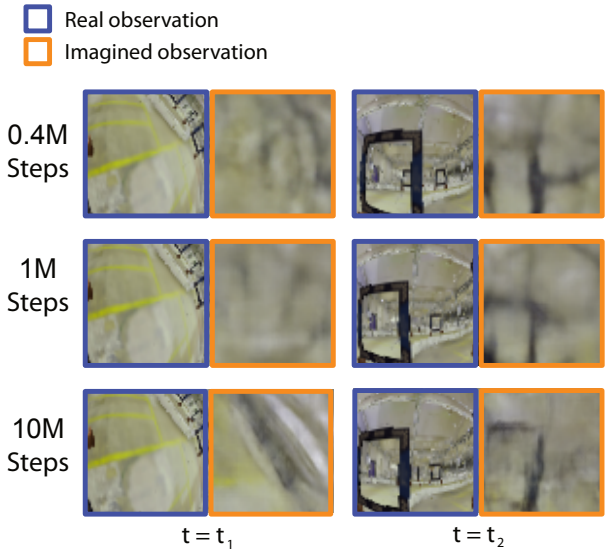

Fig. 4. Comparison of real observations and imagined observations for the Figure 8 track. Imagined observations are observations that are reconstructed by the world model. The figure shows the reconstructed observations after 0.4M steps (early stage training), 1M timesteps (mid stage training), and after 10M steps (training convergence). One can observe how the reconstruction gets better and better as the training evolves.

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. 5 shows the same policy trained with and without these additional gates. In the top part of Fig. 5, we show the policy trained without the extra gates. As indicated by the black arrows (representing camera view direction), the platform predominantly focuses on the following gates. In the bottom part of the figure, we show the policy trained

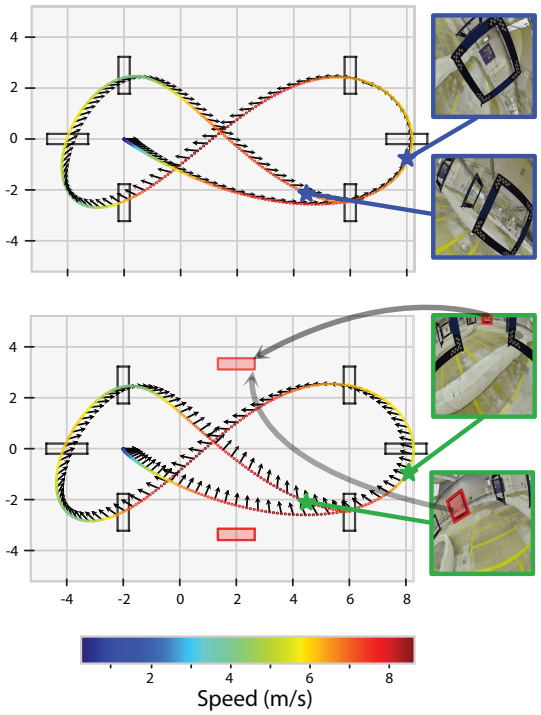

Fig. 5. **Ablation study on the perception aware behaviour of our policies. Top:** DreamerV3 policy trained on pixel observations in an environment where the only rendered gates are the actual gates. As indicated by the black arrows (representing camera direction), the platform predominantly focuses on the next gate. **Bottom:** We introduce two additional gates to the rendering engine (marked in red color). 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.

with the two additional gates to the rendering engine (marked in red color). As it can be seen, the policy's behavior changes, and the platform now distributes its camera view more evenly across both the actual and the extra gates. In the supplementary video we show the first person rendered view of the two policies depicted in Fig. 5 deployed in a simulated environment.

V. Real-world Experiments

A. Setup

The software setup is identical to the one used in the simulation experiments, explained in Section IV. Regarding the hardware, we use a modification of the *Agilicious* platform [36] for the real-world deployment. We have replaced the 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 [10], 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.

B. Results

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. Figure 6 presents a comparative analysis of the simulated and real-world trajectories and speed profiles for this track. A strong similarity is observed between the two, indicating a minimal sim-to-real gap in our dynamic model. Moreover, the camera axis, visualized by black arrows in Figure 6, aligns closely between the simulated and real-world scenarios. As already mentioned in Section IV-B, both the simulated and deployed policies show the same perception-aware behavior: they keep the camera view aligned with the next gate, even if there are no reward terms guiding or incentivizing it. 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. The supplementary video shows the deployment of our system in the real world.

VI. 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.

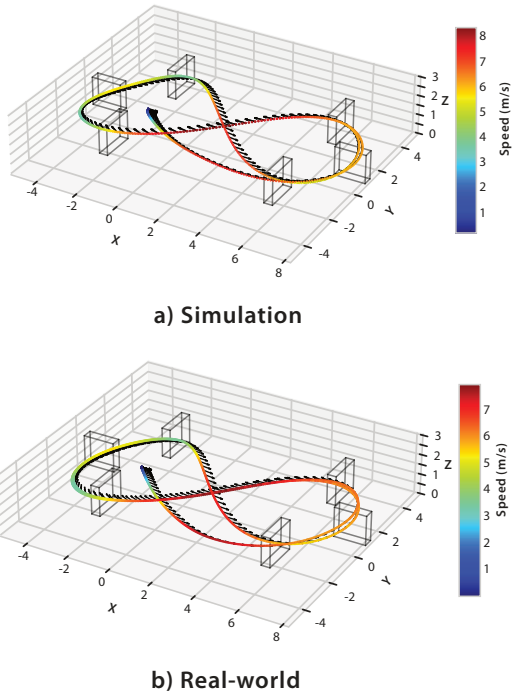

Fig. 6. Real-world deployment of the trained policy for the Figure 8 track. We show the deployment in simulation (top) and in the real-world (bottom). By looking at the 3D trajectories and the speed profile, we note that our policies transfer and result in a small sim-to-real gap.

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 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] S. Jung, S. Cho, D. Lee, H. Lee, and D. H. Shim, "A direct visual servoing-based framework for the 2016 IROS Autonomous Drone Racing Challenge," *Journal of Field Robotics*, vol. 35, no. 1, pp. 146–166, 2018.

[7] E. Kaufmann, A. Loquercio, R. Ranftl, A. Dosovitskiy, V. Koltun, and D. Scaramuzza, "Deep Drone Racing: Learning Agile Flight in Dynamic Environments," in *Proceedings of The 2nd Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, A. Billard, A. Dragan, J. Peters, and J. Morimoto, Eds., vol. 87. PMLR, 29–31 Oct 2018, pp. 133–145.

[8] C. De Wagter, F. Paredes-Vallés, N. Sheth, and G. C. H. E. de Croon, "The Sensing, State-Estimation, and Control Behind the Winning Entry to the 2019 Artificial Intelligence Robotic Racing Competition," *Field Robotics*, vol. 2, p. 1263–1290, 2022. [Online]. Available: http://dx.doi.org/10.55417/fr.2022042

[9] C. Qin, M. S. Michet, J. Chen, and H. H.-T. Liu, "Time-optimal gate-traversing planner for autonomous drone racing," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 8693–8699.

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

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

[12] 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.

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

[14] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling, "The Arcade Learning Environment: An Evaluation Platform for General Agents," *Journal of Artificial Intelligence Research*, vol. 47, p. 253–279, Jun. 2013.

[15] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015.

[16] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, "Mastering the game of Go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, Jan. 2016.

[17] Y. Tassa, Y. Doron, A. Muldal, T. Erez, Y. Li, D. d. L. Casas, D. Budden, A. Abdolmaleki, J. Merel, A. Lefrancq, T. Lillicrap, and M. Riedmiller, "DeepMind Control Suite," *arXiv preprint arXiv:1801.00690*, 2018.

[18] D. Yarats, A. Zhang, I. Kostrikov, B. Amos, J. Pineau, and R. Fergus, "Improving Sample Efficiency in Model-Free Reinforcement Learning from Images," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 12, p. 10674–10681, May 2021.

[19] E. Aljalbout, J. Chen, K. Ritt, M. Ulmer, and S. Haddadin, "Learning vision-based reactive policies for obstacle avoidance," in *Conference on Robot Learning*. PMLR, 2021, pp. 2040–2054.

[20] 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.

[21] M. Laskin, A. Srinivas, and P. Abbeel, "CURL: Contrastive Unsupervised Representations for Reinforcement Learning," in *Proceedings of the 37th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, H. D. III and A. Singh, Eds., vol. 119. PMLR, 13–18 Jul 2020, pp. 5639–5650.

[22] M. Laskin, K. Lee, A. Stooke, L. Pinto, P. Abbeel, and A. Srinivas, "Reinforcement Learning with Augmented Data," in *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., 2020.

[23] D. Yarats, R. Fergus, A. Lazaric, and L. Pinto, "Mastering Visual Continuous Control: Improved Data-Augmented Reinforcement Learning," in *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29*, 2022.

[24] A. A. Rusu, M. Vecerík, T. Rothörl, N. Heess, R. Pascanu, and R. Hadsell, "Sim-to-Real Robot Learning from Pixels with Progressive Nets," in *1st Annual Conference on Robot Learning, CoRL 2017, Mountain View, California, USA, November 13-15, 2017, Proceedings*, ser. Proceedings of Machine Learning Research, vol. 78. PMLR, 2017, pp. 262–270.

[25] 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 2017, Vancouver, BC, Canada, September 24-28, 2017*. IEEE, 2017, pp. 23–30.

[26] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *Journal of Machine Learning Research*, vol. 17, no. 39, pp. 1–40, 2016.

[27] Z. Fu, T. Z. Zhao, and C. Finn, "Mobile ALOHA: Learning Bimanual Mobile Manipulation with Low-Cost Whole-Body Teleoperation," *arXiv preprint arXiv:2401.02117*, 2024.

[28] 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.

[29] 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.

[30] N. Wahlström, T. B. Schön, and M. P. Deisenroth, "From pixels to torques: Policy learning with deep dynamical models," *arXiv preprint arXiv:1502.02251*, 2015.

[31] 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.

[32] 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.

[33] 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.

[34] 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.

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

[36] 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.

[37] 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.

[38] 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.

[39] 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.

[40] 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.

[41] 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.