Learning Quadrotor Control From Visual Features Using Differentiable Simulation

Johannes Heeg, Yunlong Song, Davide Scaramuzza

Abstract— The sample inefficiency of reinforcement learning (RL) remains a significant challenge in robotics. RL requires large-scale simulation and, still, can cause long training times, slowing down research and innovation. This issue is particularly pronounced in vision-based control tasks where reliable state estimates are not accessible. Differentiable simulation offers an alternative by enabling gradient back-propagation through the dynamics model, providing low-variance analytical policy gradients and, hence, higher sample efficiency. However, its usage for real-world robotic tasks has yet been limited. This work demonstrates the great potential of differentiable simulation for learning quadrotor control. We show that training in differentiable simulation significantly outperforms model-free RL in terms of both sample efficiency and training time, allowing a policy to learn to recover a quadrotor in seconds when providing vehicle state and in minutes when relying solely on visual features. The key to our success is two-fold. First, the use of a simple surrogate model for gradient computation greatly accelerates training without sacrificing control performance. Second, combining state representation learning with policy learning enhances convergence speed in tasks where only visual features are observable. These findings highlight the potential of differentiable simulation for real-world robotics and offer a compelling alternative to conventional RL approaches.

Video: https://youtu.be/LdgvGCLB9do
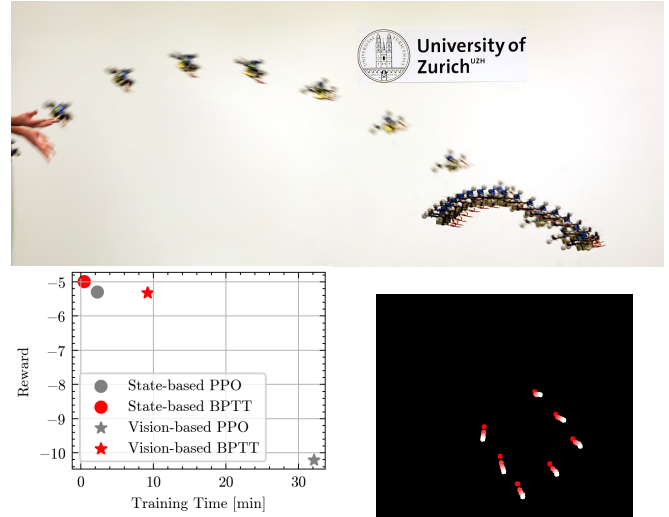**Code:** https://github.com/uzh-rpg/rpg_flightning

Fig. 1: **Learning quadrotor control from visual features.** *Top*: Stabilizing a quadrotor from a hand throw. *Bottom Left:* Backpropagation through time (BPTT) via differentiable simulation outperforms PPO, enabling state-based control in seconds and vision-based control in minutes. *Bottom Right:* A visualization of observed visual features using hardware-in-the-loop simulation.

I. INTRODUCTION

Designing effective control systems for robotic tasks remains a fundamental challenge in robotics, especially in vision-based control tasks where robots are required to operate with partial observations. Traditionally, control systems have relied on conventional cascaded controllers that require explicit and precise state estimation to function properly. Alternatively, learning-based end-to-end neural network policies have been proposed, mapping observations directly to control commands without explicit state estimation. While both paradigms have demonstrated impressive performance across various applications and real-world tasks [1], [2], [3], [4], [5], [6], they also suffer from significant limitations [7].

Conventional control algorithms depend heavily on accurate state estimation and can become unreliable when these estimates are noisy or inaccurate. On the other hand, learning-based methods, such as model-free reinforcement learning (RL), often exhibit high sample complexity and

unstable training processes. As a result, RL approaches typically require large-scale simulations and extensive data collection, even for relatively simple state-based control tasks. Moreover, in many real-world robotic scenarios where the state is not directly observable, the sample complexity increases further as the neural network must simultaneously learn both state representations from feature observations and the control policy itself.

Despite significant efforts to enhance state estimation methods, there has been relatively less focus on developing control algorithms capable of effectively handling both state-based control tasks—where the robot's state is known—and vision-based control tasks that rely solely on feature observations. While many control algorithms are effective in state-based control, they often struggle with controlling a robot directly without state estimation. This gap motivates our work and leads us to address the following research question: *How can we design a control system that effectively solves robot control tasks in both state-based and vision-based scenarios?*

We investigate this question by focusing on quadrotor control, where the objective is to control a quadrotor using only

visual features extracted from camera observations. This is a particularly difficult task for several reasons. First, quadrotors are inherently underactuated and unstable systems; without continuous and precise feedback control, they quickly lose stability and crash. Second, visual features only provide information about pixel values and their positions within the image plane. These features cannot be directly applied in classical control algorithms, which typically require full-state information such as position, orientation, and velocity for solving a control task. Finally, the system must simultaneously infer both the state representation in an implicit manner and the control law from visual inputs alone, significantly complicating the optimization of the control task.

We explore differentiable simulation, an emerging learning-based strategy, for quadrotor control. Differentiable simulation provides low-variance, first-order gradients by backpropagating gradients through the dynamics model. This capability opens up significant potential for effectively optimizing neural network control policies, especially for more complex, vision-based applications and beyond.

We begin with a classical state-based control task and then transition to vision-based control directly from visual features (i.e., without state estimation). Remarkably, after just 23 seconds of training, our state-based control policy trained in differentiable simulation successfully learns to stabilize the quadrotor from randomly initial configurations. In comparison, PPO (Proximal Policy Optimization) [8] requires over five times more samples and training time to achieve similar results. For feature-based control, our network trained with differentiable simulation learns to stabilize the quadrotor based on observations from visual features only. In contrast, PPO, even after extensive training, converges to a sub-optimal reward that is significantly lower than that of differentiable simulation, highlighting its limitations in vision-based tasks.

Contribution: The key contribution of our work is leveraging differentiable simulation for quadrotor control, focusing on controlling the vehicle from visual features only. We demonstrate that differentiable simulation offers a significant advantage over reinforcement learning for both state-based and vision-based control tasks. Importantly, we provide detailed descriptions of our design choices and several important insights that are useful for researchers to tackle more challenging tasks. First, incorporating a simplified surrogate model in backward pass speeds up gradient computation without compromising on sample efficiency and performance. Second, pretraining the control policy on an auxiliary representation task accelerates and stabilizes the overall training process. This paper may also serve as a tutorial for researchers to apply differentiable simulation for other robot control tasks.

II. RELATED WORK

One of the most common control strategies for quadrotors is the cascaded PID controller, where the inner loop controls the attitude and the outer loop controls the position or velocity [9], [10], [3], [11], [6]. Optimal control methods are also commonly used for quadrotor control, such as Linear Quadratic Regulator (LQR) and Model Predictive Control (MPC). In [12], the authors presented body-rate control using LQR to consider both the body rate and the single motor dynamics. MPC handles nonlinear systems, constraints, and disturbances more effectively than LQR. Additionally, MPC formulations are more flexible since they can incorporate different loss terms. For example, PAMPC [13] optimizes perception objectives for robust and reliable sensing by maximizing the visibility of a point of interest and minimizing its velocity in the image plane. The conventional control methods require explicit state estimation and are very sensitive to the accuracy of a given estimated state.

Neural network-based controllers offer a promising alternative to conventional methods due to their universal approximation capability, allowing the simultaneous learning of both state representation and control policies.

Model-free reinforcement learning (RL) has been used for optimizing neural network control policies for quadrotor control, including stabilizing a quadrotor [2], [14] and high-speed drone racing [15], [16]. However, the progress in applying RL to quadrotor control is largely driven by the enhanced computational capabilities provided by modern hardware rather than substantial breakthroughs in the underlying algorithms. Consequently, RL struggles with tasks where data collection is difficult to be accelerated through computational means, such as vision-based control. In general, vision-based control presents a significant challenge for RL because it requires the system to learn a state representation alongside the control policy from high-dimensional observations.

Differentiable simulation is an emerging technique in robot learning that enables the propagation of gradients through a system's dynamics model, allowing for more efficient policy optimization via analytical policy gradients. Most studies on differentiable simulation for policy optimization have been limited to simulations [17], [18], [19], with only a few successes in real-world applications, such as learning quadruped locomotion [20] and agile, vision-based drone swarm navigation [21]. In particular, [21] demonstrates how a simple point-mass model can be used to learn vision-based navigation policies for agile flight in complex real-world environments. Despite its success, this approach does not account for the vehicle's rotational dynamics. There remains a gap in research exploring low-level quadrotor control that fully incorporates the complete vehicle dynamics, which is important for leveraging the full potential of a quadrotor for super agile maneuvers, such as acrobatic flight or time-optimal flight.

III. METHOD

The key benefit of differentiable simulators is the possibility of back-propagating gradients from objective functions through the dynamics to optimization parameters. This feature can be used to train control policies with analytically derived gradients. Our fully differentiable simulation pipeline for vision-based control is summarized in Figure 2. In the following, we introduce how differentiable simulation can
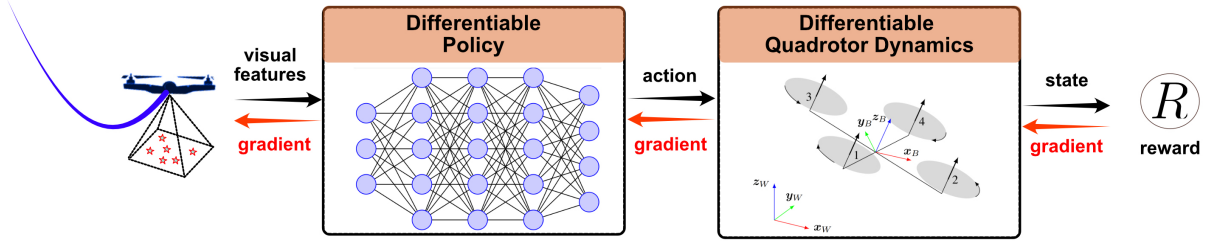
Fig. 2: Overview of policy training using differentiable simulation. A neural network policy takes actions based on visual feature observations. The simulation state is then updated using differentiable quadrotor dynamics. Based on the next state, new observations are computed with a differentiable camera model. For each state and action pair, the actor receives a reward. The differentiability of the whole pipeline allows back-propagating the gradients from the rewards to the policy parameters.

be used for policy optimization and describe the pipeline in more detail.

A. Policy Optimization via Differentiable Simulation

The quadrotor is modeled as a discrete-time dynamical system, characterized by continuous state and control input spaces, denoted as $\mathcal{X}$ and $\mathcal{U}$, respectively. At each time step t, the system state is $x_t \in \mathcal{X}$, and the corresponding control input is $u_t \in \mathcal{U}$. An observation $o_t \in \mathcal{O}$ is generated at each time step based on the current state x_t through an observation model $h : \mathcal{X} \to \mathcal{O}$, such that $o_t = h(x_t)$. The system dynamics are governed by the function $f : \mathcal{X} \times \mathcal{U} \to \mathcal{X}$, which describes the time-discretized evolution of the system $x_{t+1} = f(x_t, u_t)$. At each time step t, the robot receives a reward signal $r_t = r(x_t, u_t)$. The control policy is a deterministic, differentiable function, such as a neural network, $u_t = \pi_\theta(o_t)$. The neural network takes the observation o_t as input and outputs the control input u_t.

The objective function, the cumulative reward given the policy paramters $R(\theta)$, is to be maximized to find the optimal policy parameters θ^* by means of gradient ascent

$$\theta^* = \arg\max_\theta R(\theta)$$

$$R(\theta) = \sum_{t=0}^{N-1} r(x_t, u_t) = \sum_{t=0}^{N-1} r(x_t, \pi_\theta(o_t))$$

$$= \sum_{t=0}^{N-1} r(x_t, \pi_\theta(h(x_t)))$$

$$\theta_{k+1} \leftarrow \theta_k + \alpha \nabla_\theta R(\theta_k),$$

where α is the learning rate.

Back-propagation Through Time (BPTT): Based on the structure of $R(\theta)$, the policy gradient is obtained through BPTT and reads

$$\nabla_\theta R(\theta) =$$

$$\frac{1}{N} \sum_{t=0}^{N-1} \left(\sum_{i=1}^{t} \frac{\partial r_t}{\partial x_t} \prod_{j=i}^{t} \left(\frac{dx_j}{dx_{j-1}} \right) \frac{\partial x_i}{\partial \theta} + \frac{\partial r_k}{\partial u_k} \frac{\partial u_k}{\partial \theta} \right),$$

where the matrix of derivatives dx_j/dx_{j-1} is the Jacobian of the close-loop dynamical system $f(x, \pi_\theta(h(x)))$.

B. Quadrotor Dynamics

Let p, R, and v represent the position, orientation matrix, and linear velocity of the quadrotor, respectively, expressed in the world frame. Let ω denote the angular velocity of the quadrotor expressed in the body frame. Additionally, let c represent the mass-normalized collective thrust produced by all motors, and let $c = \begin{bmatrix} 0 & 0 & c \end{bmatrix}^\top$ denote the collective thrust vector and g is the gravity vector. Finally, J is the diagonal moment of inertia. The simple quadrotor dynamics read

$$\dot{x} = \begin{bmatrix} p \\ \mathrm{vec}(R) \\ v \end{bmatrix} = \begin{bmatrix} v \\ \mathrm{vec}(R[\omega]_\times) \\ Rc + g \end{bmatrix} \tag{1}$$

where $[\cdot]_\times$ is the skew symmetric matrix operator and $\mathrm{vec}(\cdot)$ indicates the vectorization of the input matrix.

C. JAX-based Differentiable Simulator

To enable back-propagating gradients through simulator rollouts and exploit GPU-accelerated computing, the differentiable simulation framework is entirely written using JAX [22]. During the runtime, the code is just-in-time compiled and lowered to GPU, resulting in fast execution times of up to million of steps per second (Table I). Besides high learning speeds, JAX supports automatic differentiation which can be used to compute analytical policy gradients.

Inspired by the quadrotor simulator presented in [23], our differentiable simulator models air-drag, the low-level control architecture, the motor speeds, and the transmission delays. This fidelity allows the transfer of policies trained in simulation to the real world zero-shot.

D. Fast Surrogate Gradients

Even though our simulator is fully differentiable, automatic differentiation through a full low-level control stack is not desirable. The high simulation frequency of the dynamics model, 1000 Hz, and the different sub-models involved result in an expansive computational graph for gradient computation, leading to slower execution and larger memory demand.

To circumvent this problem while keeping the high fidelity of our simulator, we use the simple dynamics model $\hat{f}(x, u)$

Parallelized Environments	Steps per Second		
	Rollout	BPTT Simple	BPTT Full
1	1,539	835	234
10	12,972	7,318	2,221
100	98,919	59,767	19,432
1,000	875,082	489,927	165,425
10,000	4,907,277	2,132,523	–
100,000	7,227,766	–	–

TABLE I: Simulations speeds with and without gradient computation (BPTT) on an Nvidia Titan RTX (24 GB VRAM). Using a simple model for the backward path (BPTT simple) keeps computation and memory costs low. For empty entries, the computational graph required more memory than accessible.

(see equation (1)), to compute the gradients. While during the forward path, the full model

$$x_{t+1} = f(x_t, u_t)$$

is used, during backpropagation, we set

$$\frac{\partial x_{t+1}}{\partial x_t} := \left.\frac{\partial \hat{f}}{\partial x}\right|_{(x_t, u_t)}, \quad \frac{\partial x_{t+1}}{\partial u_t} := \left.\frac{\partial \hat{f}}{\partial u}\right|_{(x_t, u_t)}.$$

This technique combines the realistic forward model with a computationally cheap backward model and keeps the overhead of BPTT low (Table I).

E. Pretraining on State Representation Learning

Learning from visual features comes with additional challenges for policy optimization using BPTT. First, the function mapping from features to commands is more difficult for a neural network to approximate. Second, the observation model causes indirect gradient flow from the actions back to the states. Both challenges presumably decrease the sample efficiency. We propose to pretrain the neural network parameters on the state representation task, which is known to improve the convergence of reinforcement learning [24].

The pretraining procedure comprises three steps. First, a randomly initialized policy network is used to collect data. Then, a neural network is trained to predict the underlying simulation state (the quadrotor state) from the visual observations. Lastly, the resulting weights are copied to the policy network.

F. Training Details

1) Policy: We parameterize the policy as a multi-layer perceptron with two hidden layers. The size of the hidden layers is 512 state-based and 1024 for vision-based control.

2) Action Space: The policy network provides actions at 50 Hz to the on-board flight controller. The actions are 4-dimensional and consist of desired mass-normalized collective thrust and body rates $u = [c, \omega_x, \omega_y, \omega_z]^\top$. This control modality keeps full control authority to the policy network and does not need state estimation on the flight controller. On the other hand, it requires the policy to not only steer the quadrotor but rather stabilize it at all times which is not-trivial given the non-linear, unstable dynamics (1).
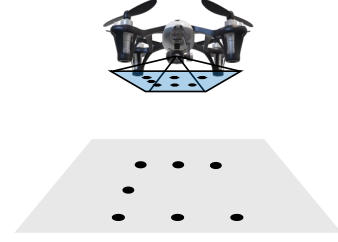

Fig. 3: Vision-based control. The actor observers only a history of pixel coordinates of seven points on the ground and the last three actions taken.

3) Observation Space: In our first experiment, we assume the quadrotor state to be known resulting in an observation $o = [\mathbf{p}^\top, \text{vec}(\mathbf{R})^\top, \mathbf{v}]^\top$. In our second experiment, learning to fly from visual features, the policy observes the pixel coordinates of 7 features on the ground as indicated in Figure 3. Since it is impossible to estimate velocities and accelerations from only one frame, we include the pixel coordinates from the past 5 frames and the last 3 actions taken by the policy. The pixel coordinates are obtained through a realistic camera model [25]. We implemented the camera model in a differentiable fashion, allowing gradients to be propagated through the observations.

4) Reward Function: Stabilizing the quadrotor means regulating its state toward a hovering condition. The desired behavior is encoded into the reward function. At time t

$$r_t = r_t^{\text{pos}} + r_t^{\text{vel}} + r_t^{\text{act}}$$

where the individual terms are

$$r_t^{\text{pos}} = -0.2 \cdot L_{\text{H}}(5 \cdot (\boldsymbol{p}_t - \boldsymbol{g}))$$
$$r_t^{\text{vel}} = -0.1 \cdot L_{\text{H}}(\boldsymbol{v}_t) - 0.1 \cdot L_{\text{H}}(\boldsymbol{\omega}_t)$$
$$r_t^{\text{act}} = -0.5 \cdot L_{\text{H}}(\boldsymbol{u}_t - \boldsymbol{u}_{\text{hover}}) - 0.01 \cdot L_{\text{H}}(\boldsymbol{u}_t - \boldsymbol{u}_{t-1}).$$
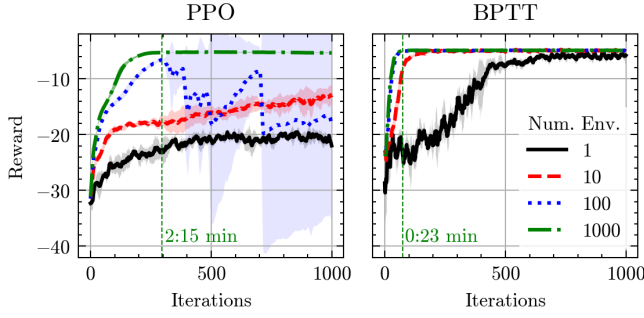
Here, L_{H} is the Huber loss and $\boldsymbol{u}_{\text{hover}} = [9.81, 0, 0, 0]^\top$, the action required to withstand the gravity.

IV. EXPERIMENTS

To assess the effectiveness of our approach, we aim to address the following questions:

- How does learning in differentiable simulation compare to model-free RL in terms of sample efficiency and training time?
- Can flight control policies trained in differentiable simulation be deployed in the real world?
- How does using a simple dynamics model for BPTT affect the learning?
- Does pretraining state representations help improve policy learning in differentiable simulation?

The task is to recover the quadrotor from a random initial condition and stabilize it. In the first experiment, the actor has access to the quadrotor state. In the second one, only visual features and past actions are observed (see III-F.3). We compare training in differentiable simulation using BPTT with PPO [8], a widely used model-free algorithms that is

Fig. 4: Comparisons of learning state-based control using PPO and BPTT.

Reward Target	Samples [Mio.]		Time [min]	
	PPO	BPTT	PPO	BPTT
-15	8.4	2.55	0:25	0:05
-11	14.7	3.45	0:44	0:07
-7	22.8	5.4	1:09	0:11
-5.3	44.4	8.25	2:14	0:17

Fig. 5: Comparisons of learning feature-based control using PPO and BPTT.

Reward Target	Samples [Mio.]		Time [min]	
	PPO	BPTT	PPO	BPTT
-15	32.1	2.25	4:10	0:07
-11	118.2	3.9	15:21	0:13
-7	—	11.25	—	0:37
-5.3	—	179.85	—	10:00

well-known to work well when large-scale simulation can be exploited. In the experiments, we include runs with different numbers of parallel environments to examine the gradient variance. All training procedures were run on a workstation with an Nvidia Titan RTX (24 GB VRAM).

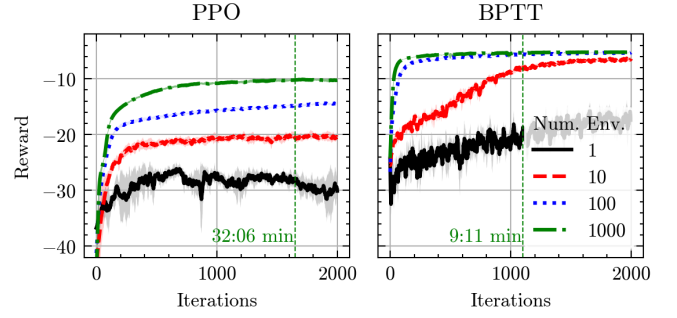### A. Learning State-based Control

We train the control policy for 1000 iterations using BPTT and PPO with 1 to 1000 parallel environments as shown in Figure 4. We observe that using PPO, only the experiment with 1000 parallel environments stably converges. On the other hand, all BPTT runs with more than 1 environment converge to very similar high rewards, and even with 1 single environment, the learning curve almost reaches the same performance. This result confirms the practical advantage of a low-variance gradient estimate.

Regarding the training time, using 1000 parallel environments gives the best results for both methods. Here, PPO converges after 2:15 minutes, while BPTT needs only 23 seconds.

The table in Figure 4 compares how many samples and how much training time is needed to reach certain reward targets. We find that BPTT reaches all targets significantly faster and requires only a fraction of samples. In particular, BPTT demonstrates an almost 8-times speed-up to reach the highest reward target -5.3 and requires less than 20% of samples.

B. Feature-based Control without State Estimation

For the more challenging task of vision-based quadrotor control, we pretrain the networks using state representation learning for about 1:30 min and subsequently run policy optimization for 2000 iterations. Unsurprisingly, the time to convergence is higher in all cases. Only BPTT with 100 or 1000 parallel environments manages to converge to a performance close to the state-based case. Even disregarding the significantly lower reward of PPO policies at convergence,

Fig. 6: Feature-based control experiment (HITL). Visualization of flight trajectories with randomly initialized drone configurations.

the convergence time of BPTT, 9:11 min, is more than 3 times lower compared to 32:06 min, the convergence time of PPO. Furthermore, The table in Figure 5 displays the wall-clock training time and simulation samples required to reach the reward targets. Again, we observe much lower sample and time requirements. For the last target hit by PPO, a reward of -11, BPTT was over 70 times faster and needed less than 4% of samples.

C. Sim-to-real Transfer

For real-world experiments, we use a lightweight drone built with off-the-shelf components. Using hardware-in-the-loop (HITL) simulation, the pose of the physical quadrotor is monitored by a motion capture system, and the actor receives virtual observations, illustrated in Figure 1. The HITL experiments allow us to focus on testing the control performance without the need for a feature detection module. Note that even though the simulated observation model causes no sim-
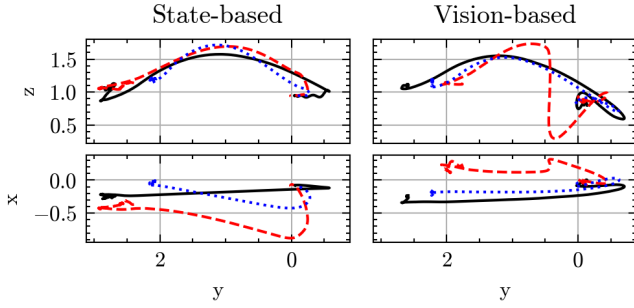
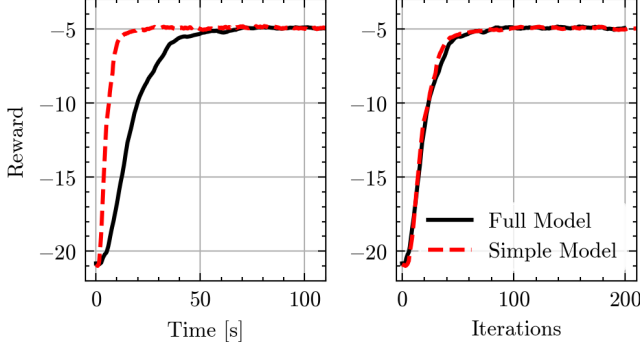Fig. 7: Real-world trajectories showing stabilization from a manual throw.

Fig. 8: Using a simple model for gradient computations reduces training time without sacrificing on performance.
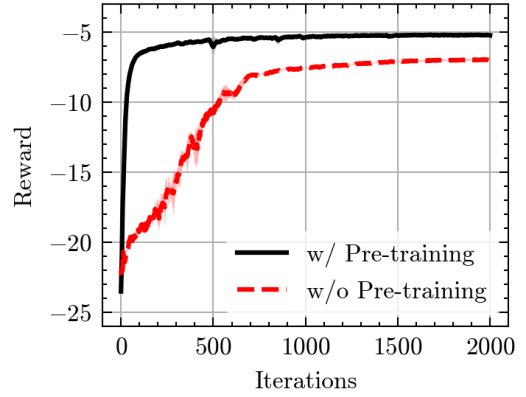

Fig. 9: Pre-training the neural network parameters on state representation learning improves convergence and performance during policy optimization.

improves both convergence and asymptotic performance significantly. Note, however, that even without pretraining, the performance surpasses the one of PPO on the same task.

We hypothesize that state representation pretraining is particularly useful when combined with differentiable simulation because it not initializes the weights to provide a more meaningful internal representation but also improves the quality of the gradient. Possibly, the pretrained weights make the first-order approximation of the policy optimization objective more accurate in a larger vicinity of the current parameters.

V. CONCLUSION

This paper investigated using differentiable simulation to learn quadrotor control policies from state and visual observations. We found that differentiable simulation can accelerate policy training drastically, leading to more than 7 times faster training speeds and requiring less than 20% of samples compared to model-free reinforcement learning. We show that by using a simple dynamics model, computational costs can be significantly reduced without compromising on sample efficiency. Moreover, leveraging pretraining on a state representation task helps convergence and improves performance substantially.

Our work demonstrates the first application of differentiable simulation to low-level quadrotor control. We deploy the trained policies successfully in the real world on the challenging task of stabilizing the quadrotor after a manual throw without state estimation and only observing visual features.

Overall, the results suggest that differentiable simulation has the potential of introducing a paradigm shift for learning in simulation. We believe that despite the increasing efforts on developing high-fidelity differentiable simulators, using simple backward models will be faster and sufficient in many cases and requires only limited engineering effort. Therefore, we hope to inspire the community to try differentiable simulation and accelerate their research and innovation.

to-real gap, the measurement noise of the motion capture systems results in out-of-distribution observations.

Figure 6 illustrates the deployment of a vision-based control policy trained in differentiable simulation. The policy successfully navigates and stabilizes the quadrotor at the desired goal position. Figure 1 shows a second experiment where the quadrotor is thrown manually resulting in diverse and highly unstable initial states. However, as indicated by Figure 7, both state-based and vision-based control policies successfully perform the task in successive trials without failure.

D. Ablation Studies

1) Policy Optimization Using Surrogate Gradients: Using a simpler dynamics model for the backward path trivially increases the number of simulation steps per second, as shown in Table I. However, using only a surrogate gradient might weaken the quality of the gradient and hence, decrease sample efficiency. Figure 8 shows two runs with BPTT over 200 iterations (100 environments) on the state-based control task. Both runs use the same initial parameters and random seeds. The only difference is the quadrotor model. It clearly shows that the simple model speeds up training but does not sacrifice sample efficiency.

2) State Representation Learning as Pretraining: Figure 9 compares the BPTT training curves on the vision-based control task with and without state representation pretraining using 1000 environments. We observe that pretraining

REFERENCES

[1] P. Foehn, D. Brescianini, E. Kaufmann, T. Cieslewski, M. Gehrig, M. Muglikar, and D. Scaramuzza, "Alphapilot: Autonomous drone racing," *RSS: Robotics, Science, and Systems*, 2020.

[2] J. Hwangbo, I. Sa, R. Siegwart, and M. Hutter, "Control of a quadrotor with reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 2, no. 4, pp. 2096–2103, 2017.

[3] M. Hehn and R. D'Andrea, "Quadrocopter trajectory generation and control," *IFAC proceedings Volumes*, vol. 44, no. 1, pp. 1485–1491, 2011.

[4] C. Richter, A. Bry, and N. Roy, "Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments," in *Robotics research*. Springer, 2016, pp. 649–666.

[5] M. Kamel, M. Burri, and R. Siegwart, "Linear vs nonlinear mpc for trajectory tracking applied to rotary wing micro aerial vehicles," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 3463–3469, 2017.

[6] S. Sun, A. Romero, P. Foehn, E. Kaufmann, and D. Scaramuzza, "A comparative study of nonlinear mpc and differential-flatness-based control for quadrotor agile flight," *IEEE Transactions on Robotics*, vol. 38, no. 6, pp. 3357–3373, 2022.

[7] C. Tang, B. Abbatematteo, J. Hu, R. Chandra, R. Martín-Martín, and P. Stone, "Deep Reinforcement Learning for Robotics: A Survey of Real-World Successes," Aug. 2024, arXiv:2408.03539 [cs]. [Online]. Available: http://arxiv.org/abs/2408.03539

[8] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms," Aug. 2017, 9994 citations (Semantic Scholar/arXiv) [2023-11-26] arXiv:1707.06347 [cs]. [Online]. Available: http://arxiv.org/abs/1707.06347

[9] S. Bouabdallah, A. Noth, and R. Siegwart, "Pid vs lq control techniques applied to an indoor micro quadrotor," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, vol. 3. IEEE, 2004, pp. 2451–2456.

[10] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *IEEE Int. Conf. Robot. Autom. (ICRA)*, 2011, pp. 2520–2525.

[11] M. Faessler, F. Fontana, C. Forster, and D. Scaramuzza, "Automatic re-initialization and failure recovery for aggressive flight with a monocular vision-based quadrotor," in *IEEE Int. Conf. Robot. Autom. (ICRA)*, 2015, pp. 1722–1729.

[12] M. Faessler, D. Falanga, and D. Scaramuzza, "Thrust mixing, saturation, and body-rate control for accurate aggressive quadrotor flight," *IEEE Robot. Autom. Lett.*, vol. 2, no. 2, pp. 476–482, 2017.

[13] D. Falanga, P. Foehn, P. Lu, and D. Scaramuzza, "Pampc: Perception-aware model predictive control for quadrotors," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct 2018, pp. 1–8.

[14] J. Eschmann, D. Albani, and G. Loianno, "Learning to fly in seconds," *IEEE Robotics and Automation Letters*, 2024.

[15] 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.

[16] 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.

[17] C. D. Freeman, E. Frey, A. Raichuk, S. Girgin, I. Mordatch, and O. Bachem, "Brax–a differentiable physics engine for large scale rigid body simulation," *arXiv preprint arXiv:2106.13281*, 2021.

[18] J. Xu, V. Makoviychuk, Y. Narang, F. Ramos, W. Matusik, A. Garg, and M. Macklin, "Accelerated policy learning with parallel differentiable simulation," in *International Conference on Learning Representations*, 2021.

[19] T. Howell, S. Le Cleac'h, J. Bruedigam, Z. Kolter, M. Schwager, and Z. Manchester, "Dojo: A differentiable simulator for robotics," *arXiv preprint arXiv:2203.00806*, 2022.

[20] Y. Song, S. Kim, and D. Scaramuzza, "Learning quadruped locomotion using differentiable simulation," *arXiv preprint arXiv:2403.14864*, 2024.

[21] Y. Zhang, Y. Hu, Y. Song, D. Zou, and W. Lin, "Back to newton's laws: Learning vision-based agile flight via differentiable physics," *arXiv preprint arXiv:2407.10648*, 2024.

[22] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang, "JAX: composable transformations of Python+NumPy programs," 2018. [Online]. Available: http://github.com/google/jax

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

[24] T. de Bruin, J. Kober, K. Tuyls, and R. Babuška, "Integrating State Representation Learning Into Deep Reinforcement Learning," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1394–1401, Jul. 2018, conference Name: IEEE Robotics and Automation Letters. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8276247

[25] V. Usenko, N. Demmel, and D. Cremers, "The double sphere camera model," in *2018 International Conference on 3D Vision (3DV)*, Sep. 2018, pp. 552–560, arXiv:1807.08957 [cs]. [Online]. Available: http://arxiv.org/abs/1807.08957