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 can still cause long training times, slowing 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 states 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
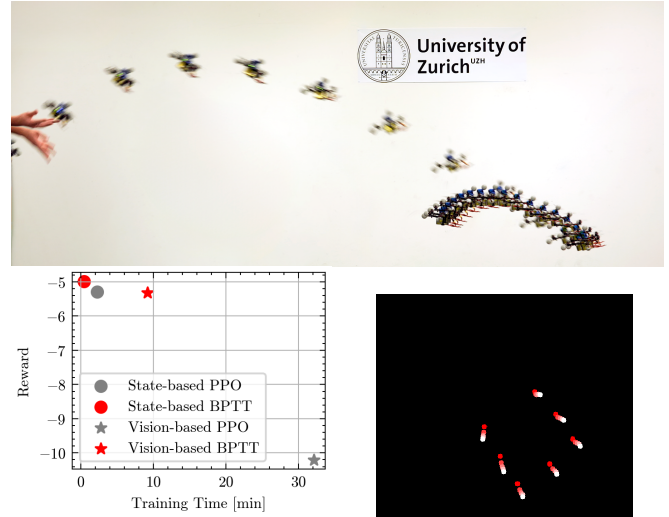**Code:** https://github.com/uzh-rpg/rpg_flightning

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

Reinforcement Learning (RL) has become a cornerstone in robotics research for designing powerful controllers. By employing large-scale simulation, neural network controllers trained with RL enabled impressive performance in many robotic tasks [1] like quadruped locomotion [2], robot soccer [3], or quadrotor control [4], [5]. The learning paradigm allows for the end-to-end design of control policies that take various input modalities and optimize task-level objectives [6]. However, the flexibility comes at the cost of low sample efficiency, which slows down the development of new controllers. Successful implementations require massively parallelized simulators and large compute resources. Yet, training times can go up to hours or even days. This issue makes iterating on the control architecture, environment settings, and hyperparameter tuning tedious and delays new results and progress.

The sample efficiency problem becomes even more pronounced in vision-based control tasks [7]. Not only is the

simulation more expensive due to the additional requirement of simulating the visual observations, a direct mapping from visual input signals to control commands is more difficult to learn. First, the dimensionality of the inputs increases, requiring more samples to optimize the policy. Second, the neural network must simultaneously build an implicit state representation and synthesize the control law.

Differentiable simulation offers an approach to improve the sample efficiency in robot learning. By incorporating the derivatives of the dynamics, policy gradients can be computed analytically. This method reduces the variance in the policy parameter updates, which potentially decreases the number of samples needed to learn a new control task.

We investigate using differentiable simulation to learn vision-based quadrotor control. In our case study, the objective is to stabilize a quadrotor from a manual throw using only visual features extracted from camera observations (see Figure 1). The task is challenging for several reasons. First, quadrotors are inherently underactuated and unstable systems; without continuous and precise feedback control, they quickly lose stability and crash. Second, state information is only indirectly available through pixel coordinates in the
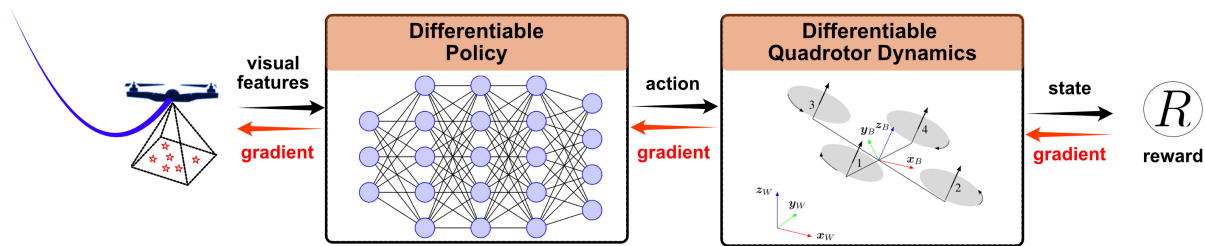
Fig. 2: Overview of policy training using differentiable simulation. A neural network policy takes actions based on visual feature observations. The simulation state is then updated using differentiable quadrotor dynamics. Based on the next state, new observations are computed with a differentiable camera 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.

image plane, making the environment partially observable. Finally, the control policy has to handle a wide initial state distribution to recover from a manual throw.

In our study, 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). We compare learning in differentiable simulation with PPO (Proximal Policy Optimization) [8], a popular model-free RL algorithm. Remarkably, after just 23 seconds of training, our state-based control policy trained in differentiable simulation successfully learns to stabilize the quadrotor from random initial configurations. In comparison, PPO requires over five times more samples and training time to achieve similar results. For feature-based control, our network trained in 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, first, differentiable simulation offers a significant advantage over model-free RL for both state-based and vision-based control tasks. Second, incorporating a simplified surrogate model in backward pass speeds up gradient computation without compromising on sample efficiency and performance. Third, pretraining the control policy on an auxiliary representation task accelerates and stabilizes the overall training process.

II. RELATED WORK

Model-free RL is popular method to learn quadrotor control policies, including stabilization [4], [9] and high-speed drone racing [6], [10]. For vision-based control, it is common to split the design process into state estimator and controller design. In [10], the quadrotor state is estimated using a Kalman filter that receives measurements from a visual-inertial odometry module and additional visual features. The state estimate is then fed into a state-based policy that was trained using RL. Splitting state estimation and control gives rise to a modular structure. However, the composition can

lead to new failure modes caused by compounding errors. Alternatively, an end-to-end learning approach is studied in [7] for drone racing. The authors demonstrate robust flight performance solely relying on visual inputs. Their experiments show that learning from visual inputs requires much more samples and longer training times of around 24 hours.

Most studies on differentiable simulation for policy optimization in robotics have been limited to simulated experiments [11]–[13], with only a few successes in real-world applications, such as learning quadruped locomotion [14] and vision-based drone swarm navigation [15]. In particular, [15] demonstrates how a simple point-mass model can be used to learn vision-based navigation policies for agile flight in complex real-world environments. Their approach requires state estimation and a low-level controller that can stabilize the quadrotor and track the acceleration commands from the policy. Hence, there remains a research gap in exploring low-level quadrotor control that fully incorporates the complete vehicle dynamics, which is important for leveraging the full potential of a quadrotor for agile maneuvers, such as acrobatic flight or time-optimal flight.

Using a surrogate model for policy gradient computations was first studied in [16]. The authors propose using an approximate simulation model to compute gradients along real-world trajectories. They prove that the resulting policy parameters lie close to a local optimum with respect to the true but unknown model. Another work [17] even considers the case in which the surrogate model only predicts the sign of the derivatives. Recently, [14] proposed a surrogate model for gradient computations in the context of quadruped locomotion.

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

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, a neural network, $u_t = \pi_\theta(o_t)$. The policy takes the observation o_t as the input and outputs the control input u_t.

The objective function $R(\theta)$, the cumulative reward given the policy parameters, 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): By taking advantage of the structure of $R(\theta)$ and applying the chain rule along the temporal axis, the policy gradient is obtained through BPTT and reads

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

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

where the matrix of derivatives dx_j/dx_{j-1} is the Jacobian of the close-loop dynamical system $f(x, \pi_\theta(h(x)))$. For a derivation of the BPTT gradient, the reader is referred to [18].

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, referred to as body rates. Additionally, let c represent the mass-normalized collective thrust produced by all motors, and let $c = \begin{bmatrix} 0 & 0 & c \end{bmatrix}^\mathsf{T}$ denote the collective thrust vector and g the gravity vector. The simple quadrotor dynamics read

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

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: Simulation 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.

where $[\cdot]_\times$ is the skew symmetric matrix operator and $\mathrm{vec}(\cdot)$ indicates vectorization. For quadrotor control, the body rates ω and the collective thrust c are the inputs to the system.

C. JAX-based Differentiable Simulator

To enable back-propagating gradients through simulated rollouts and exploit GPU-accelerated computing, the differentiable simulation framework is entirely written using JAX [19]. During 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 [20], 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 $\hat{f}(x, u)$ (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

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

This technique combines the realistic forward model with a computationally cheap backward model and keeps the overhead of BPTT low (Table I).
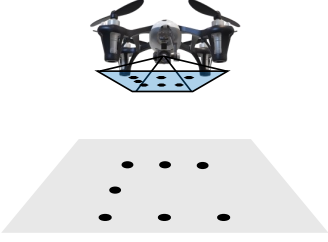
Fig. 3: Vision-based control. The actor observers only a history of pixel coordinates of seven points on the ground and the last three actions taken.

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 [21].

The pretraining procedure comprises three steps. First, a randomly initialized policy network π_θ is used to collect a dataset $\mathcal{D}$ of pairs of simulation states and observations (x_t, o_t). Then, a neural network is trained to predict the underlying simulation state (the quadrotor state) from the visual observations. I.e., the neural network ψ_θ is trained to minimize the mean squared error

$$\frac{1}{N} \sum_{(x_i, o_i) \in \mathcal{D}} \|x_i - \psi_\theta(o_i)\|^2.$$

Lastly, the resulting weights θ are copied to the policy network π_θ. Note that the last layer is changed do to the different output modalities.

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 for 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).

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 [22]. 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 at a desired position $\boldsymbol{p}_{\text{des}}$. 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

$$r_t^{\text{pos}} = -0.2 \cdot L_{\text{H}}(5 \cdot (\boldsymbol{p}_t - \boldsymbol{p}_{\text{des}}))$$
$$r_t^{\text{vel}} = -0.1 \cdot L_{\text{H}}(\boldsymbol{v}_t) - 0.1 \cdot L_{\text{H}}(\boldsymbol{\omega}_t)$$
$$r_t^{\text{act}} = -0.5 \cdot L_{\text{H}}(\boldsymbol{u}_t - \boldsymbol{u}_{\text{hover}}) - 0.01 \cdot L_{\text{H}}(\boldsymbol{u}_t - \boldsymbol{u}_{t-1}).$$

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. We use the Huber loss to limit the magnitude of the gradient even for large deviations to improve training stability.
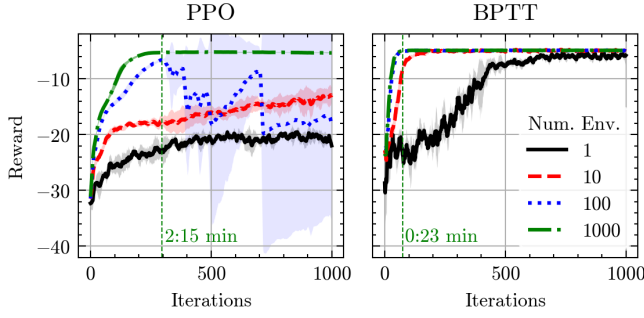
IV. EXPERIMENTS

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 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.

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. 4: Comparisons of learning state-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

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

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.
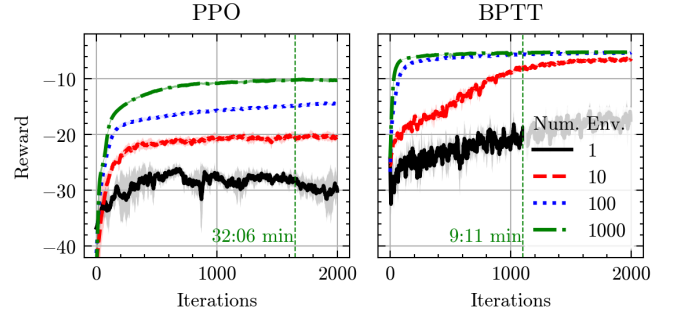
The table in Figure 4 compares how many samples and how much training time is needed to reach certain reward targets (with 1000 parallel environments). We find that BPTT reaches all targets significantly faster and requires only a fraction of samples. In particular, BPTT demonstrates an almost 8-times 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 policy networks using state representation learning on 100,000 samples for 500 epochs (pretraining time is around 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, 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

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

virtual observations, visualized 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-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. As indicated by Figure 7, both state-based and vision-based control policies successfully perform the task in successive trials without failure. The state-based policy stabilizes the quadrotor faster and with less oscillations around the desired position. In particular, in the third vision-based trial (denoted in red), the quadrotor came close to the floor before approaching the goal position. This
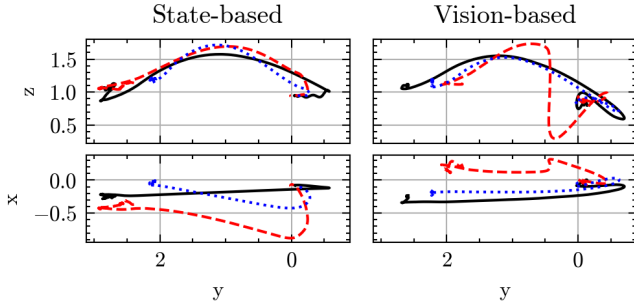
Fig. 7: Real-world trajectories showing stabilization from a manual throw. Each line represents an individual trial.
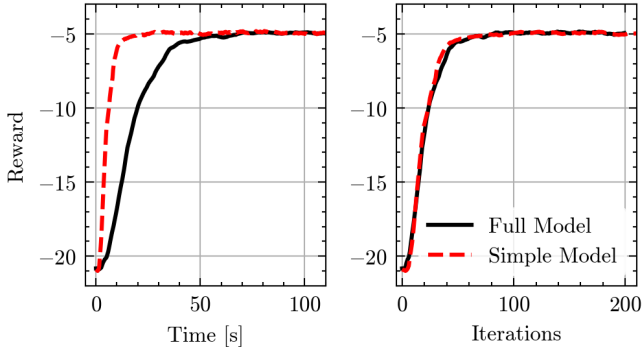

Fig. 8: Using a simple model for gradient computations reduces training time without sacrificing on performance.
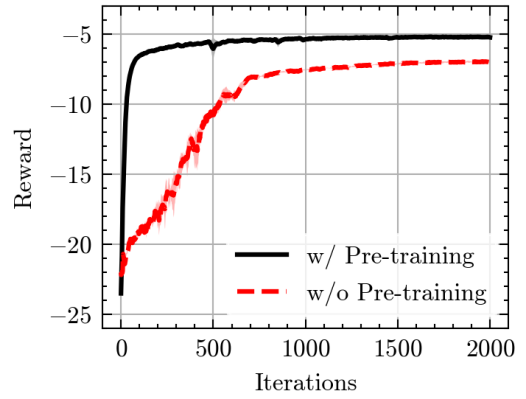

Fig. 9: Pre-training the neural network parameters on state representation learning improves convergence and performance during policy optimization.

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.

behavior indicates that the vision-based performance could benefit from an improved observation noise model to reduce the sim-to-real gap.

D. Ablation Studies

1) Policy Optimization Using Surrogate Gradients: Using a simpler dynamics model for the backward path directly 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 or performance.

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 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 only initializes the weights to provide a more meaningful internal representation but also improves the quality of the gradient. Possibly, the pretrained weights

REFERENCES

[1] 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

[2] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter, "Learning agile and dynamic motor skills for legged robots," *Science Robotics*, vol. 4, no. 26, p. eaau5872, 2019.

[3] T. Haarnoja, B. Moran, G. Lever, S. H. Huang, D. Tirumala, J. Humplik, M. Wulfmeier, S. Tunyasuvunakool, N. Y. Siegel, R. Hafner, M. Bloesch, K. Hartikainen, A. Byravan, L. Hasenclever, Y. Tassa, F. Sadeghi, N. Batchelor, F. Casarini, S. Saliceti, C. Game, N. Sreendra, K. Patel, M. Gwira, A. Huber, N. Hurley, F. Nori, R. Hadsell, and N. Heess, "Learning agile soccer skills for a bipedal robot with deep reinforcement learning," *Science Robotics*, vol. 9, no. 89, p. eadi8022, Apr. 2024.

[4] 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.

[5] 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.

[6] 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.

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

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

[10] 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.

[11] 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.

[12] 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.

[13] 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.

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

[15] 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.

[16] P. Abbeel, M. Quigley, and A. Y. Ng, "Using inaccurate models in reinforcement learning," in *Proceedings of the 23rd International Conference on Machine Learning - ICML '06*. Pittsburgh, Pennsylvania: ACM Press, 2006, pp. 1–8.

[17] J. Z. Kolter and A. Y. Ng, "Policy search via the signed derivative," in *Robotics: Science and Systems V*. Robotics: Science and Systems Foundation, Jun. 2009.

[18] L. Metz, C. D. Freeman, S. S. Schoenholz, and T. Kachman, "Gradients are not all you need," *arXiv preprint arXiv:2111.05803*, 2021.

[19] 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

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

[21] 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

[22] 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