2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 23-27, 2022, Kyoto, Japan

Using Simulation Optimization to Improve Zero-shot Policy Transfer of Quadrotors

Sven Gronauer, Matthias Kissel, Luca Sacchetto, Mathias Korte and Klaus Diepold

Abstract— **In this work, we propose a data-driven approach to optimize the parameters of a simulation such that control policies can be directly transferred from simulation to a real-world quadrotor. Our neural network-based policies take only onboard sensor data as input and run entirely on the embedded hardware. In real-world experiments, we compare low-level Pulse-Width Modulated control with higher-level control structures such as Attitude Rate and Attitude, which utilize Proportional-Integral-Derivative controllers to output motor commands. Our experiments show that low-level controllers trained with Reinforcement Learning require a more accurate simulation than higher-level control policies at the expense of being less robust towards parameter uncertainties.**

I. INTRODUCTION

Programming intelligent control strategies for complex robot systems is a challenging task. Reinforcement learning (RL) promises the automated generation of control strategies through a data-driven approach instead of explicitly designing hand-crafted solutions through expert knowledge. In recent years, the field of RL has witnessed outstanding successes and raised a surge of interest in the control of dynamical systems through such a trial-and-error paradigm. The combination of RL and deep learning methods excel at problems that can be quickly simulated like robotics [13] and video games [19] or in domains where the exact model is known but long-horizon planning is not computationally tractable, e.g. board games like Go and Chess [26].

Despite the significant advances in recent years, the applicability of RL algorithms is still limited when the data at test time differ from those seen during training [10]. Since many real-world systems cannot afford to learn policies from scratch due to the expense of data, simulations are the preferred approach to build data-driven control policies in the RL community. Naturally, a gap between the simulation and the real world exists because an accurate model either requires in-depth expert knowledge or is simply not desirable, e.g. calculating all aero-dynamical effects can significantly increase simulation time. A successful policy transfer thus requires the reality gap to be small.

In this paper, we address the sim-to-real gap in the domain of quadrotor control and investigate three different structures for quadrotor control. Our contributions are:

1) We apply simulation optimization as a data-driven approach to narrow the reality gap and demonstrate that zero-shot policy transfers become feasible with minimal tuning effort. The data used for optimization were collected for approximately one hour on the real-world quadrotor with Proportional-Integral-Derivative (PID) controllers.
2) We deploy low-level control policies based on Pulse-Width Modulated (PWM) thrust commands trained entirely in simulation on the quadrotor without fine-tuning the policy. To the best of our knowledge, this is the first work using RL as framework for low-level decision making that accomplishes successful zero-shot transfers to a real-world quadrotor using only onboard sensing and computation.
3) We study three control structures that differ in their level of abstraction: PWM, Attitude Rate and Attitude control. Through real-world experiments, we investigate the required fidelity of the simulation with respect to the deployed control structure.

Throughout this work, we focus on the context of zero-shot policy transfer, i.e. we train the agent entirely in simulation and then deploy the controller on a quadrotor robot without using real-world data to fine-tune the policy.

II. RELATED WORK

A. Quadrotor Control

For attitude control and set-point tracking of quadrotors, a common approach is a hierarchical control that consists of nested PID controllers [17]. Approaches like the linear quadratic regulator are also suitable methods for the stabilization around the hover conditions under reasonably small roll and pitch angles. However, when more dynamic flight behaviors are desired, more complex controllers are required [18]. Due to their highly dynamic movement capabilities, quadrotors depict an interesting platform to test maneuvers such as landing [11] and perform acrobatic maneuvers like loopings and rolls [9], multi-flips [16] or flying through narrow vertical gaps [18]. While most of the aforementioned works rely on highly accurate state estimation, only a few papers have considered quadrotor control based on pure onboard sensing. In [12], model-based RL was used to train a policy from real-world data while relying on onboard sensor measurements to run control with direct motor PWM signals. The authors in [15] showed that a quadrotor can be navigated through a variety of cluttered real-world scenarios like forests or buildings with onboard sensing and computation.

B. Bridging the Reality Gap

To reduce the gap between simulation and real-world, *domain randomization* has been proposed to augment the diversity of data by randomization. One way is to randomize

The authors are with Department of Electrical and Computer Engineering, Technical University of Munich, Arcisstr. 21, 80333 Munich, Germany. Contact author: sven.gronauer@tum.de

2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) | 978-1-6654-7927-1/22/$31.00 ©2022 IEEE | DOI: 10.1109/IROS47612.2022.9981229

the dynamics where simulation parameters responsible for the realization of the system are re-sampled at the beginning of every trajectory [22]. Another approach is to randomize the rendering of image-based observations in the simulation [27]. In [14] it was shown that agile drone racing is possible with convolutional neural networks when trained on an abundance of image-based data. Besides domain randomization, another method to overcome the reality gap is *simulation optimization* [1] which is a data-driven approach to identify simulation parameters based on real-world data. In [2] a simulation parameter distribution was learned based on collected data from a physical manipulator robot.

C. Zero-shot Policy Transfer

In the area of RL, there have been only a few works that study the zero-shot policy transfer to real-world quadrotors. The work of [9] showed that Attitude Rate control based on image-based and onboard data could perform highly agile maneuvers such as rolls and loopings. Another work [15] performed zero-shot transfers into unknown and cluttered real-world environments while using onboard sensing and computation only. The policy computed collision-free trajectories based on cameras which are then tracked by a model-predictive controller. Most similar to our paper is [20] where low-level PWM-based controllers were able to stabilize and generalize to multiple sizes of quadrotors. The authors used an external tracking system to accurately estimate the drone state and showed that a zero-shot transfer is possbile. In contrast to the aforementioned works, we study the zero-shot policy transfer with onboard sensing and computation while using RL as the only framework for low-level decision making.

III. PRELIMINARIES

A. Reinforcement Learning

A Markov Decision Process (MDP) is formalized by a tuple $(\mathbb{S}, \mathbb{A}, \mathcal{P}, r, \mu)$, where $\mathbb{S}$ and $\mathbb{A}$ denote the state and action spaces respectively. $\mathcal{P} : \mathbb{S} \times \mathbb{A} \to P(\mathbb{S})$ describes the system transition probability, μ denotes the initial state distribution and $r : \mathbb{S} \times \mathbb{A} \to \mathbb{R}$ is the reward function. Let $\tau = (\boldsymbol{s}_0, \boldsymbol{a}_0, \boldsymbol{s}_1, \boldsymbol{a}_1, \dots)$ be a trajectory generated under policy π with $\boldsymbol{s}_{t+1} \sim \mathcal{P}(\cdot|\boldsymbol{s}_t, \boldsymbol{a}_t), \boldsymbol{a}_t \sim \pi(\cdot|\boldsymbol{s}_t)$ and $\boldsymbol{s}_0 \sim \mu$. We apply the shortcut $\tau \sim \pi$ when trajectories are generated under π and denote the trajectory return by $R(\tau) = \sum_{t=0}^{\infty} \gamma^t r(\boldsymbol{s}_t, \boldsymbol{a}_t)$ with the discount factor $\gamma \in (0, 1)$. In RL, the goal of the agent is to learn a control policy $\pi : \mathbb{S} \to \mathcal{P}(\mathbb{A})$ that maximizes the expected return $J(\pi) = \mathbb{E}_{\tau\sim\pi}[R(\tau)]$. We use π_θ to denote that the policy is parametrized by a neural network with weights $\boldsymbol{\theta}$.

In this work, we sample system parameters $\boldsymbol{\xi}$ from a distribution Ξ at the beginning of each trajectory. This results in the system transition probability being dependent on the system parameters $s_{t+1} \sim \mathcal{P}(\cdot|s_t, a_t, \xi)$ which is known as domain randomization. Thus, we seek policy parameters that maximize the expected return

$$\max_{\boldsymbol{\theta}} J(\pi_\theta) = \mathbb{E}_{\boldsymbol{\xi}\sim\Xi}\left[\mathbb{E}_{\tau\sim\pi_\theta}[R(\tau)]\right] \tag{1}$$

over the dynamics induced by the distribution of simulation parameters.

B. Quadrotor

1) Dynamics Model: The dynamics of the quadrotor are modeled by the differential equation

$$\dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u}) \tag{2}$$

where the drone state $\boldsymbol{x} = [\boldsymbol{p}^T, \dot{\boldsymbol{p}}^T, \boldsymbol{\varphi}^T, \boldsymbol{\omega}^T]^T \in \mathbb{R}^{13}$ encompasses the position $\boldsymbol{p}$, the linear velocity $\dot{\boldsymbol{p}}$, the body angle $\boldsymbol{\varphi}$ in quaternions and the angular speed $\boldsymbol{\omega}$. The acceleration of the drone's center of mass is described by Newton's equation

$$m\ddot{\boldsymbol{p}} = \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} + R \begin{bmatrix} 0 \\ 0 \\ \sum F_i \end{bmatrix} \tag{3}$$

in the inertial frame $\mathcal{O}$ with gravity g. The quadrotor mass is m and $\sum F_i$ is the sum of the vertical forces acting on the rotors. R is the rotation matrix from the body frame $\mathcal{B}$ to the inertial frame. The angular acceleration governed by Euler's rotation equations in $\mathcal{B}$ is

$$I\dot{\boldsymbol{\omega}} = \boldsymbol{\eta} - \boldsymbol{\omega} \times (I\boldsymbol{\omega}) \tag{4}$$

with the inertia matrix I. The torques $\boldsymbol{\eta}$ acting in the body frame are determined by

$$\boldsymbol{\eta} = \begin{bmatrix} \frac{1}{\sqrt{2}} L(-F_1 - F_2 + F_3 + F_4) \\ \frac{1}{\sqrt{2}} L(-F_1 + F_2 + F_3 - F_4) \\ -M_1 + M_2 - M_3 + M_4 \end{bmatrix} \tag{5}$$

with the rotor forces F_i, the corresponding motor torques M_i and the drone's arm length L. An overview of the quadrotor setup and the coordinate frames is depicted in Fig. 1 (Left).

2) Motor Model: The angular speed ν_i of rotor i produces a vertical force

$$F_i = \frac{mg}{4} k_F \nu_i^2 \tag{6}$$

that lifts the quadrotor. We use $k_F \in \mathbb{R}$ to denote the thrust-to-weight ratio. The rotors also produce a moment according to

$$M_i = k_{M_1} F_i + k_{M_2} \tag{7}$$

with k_{M_1} and k_{M_2} being scalars. The motor speeds are normalized $\nu_i \in [0, 1]$ and are modeled $\boldsymbol{\nu} = \sqrt{\boldsymbol{u}}$ as the square root of normalized commanded thrusts $u_i \in [0, 1]$. For PWM control, the relationship between the action $\boldsymbol{a}$ taken by the agent and the normalized commanded thrust is $u_i = \frac{1}{2}[\min(\max(a_i, -1), 1) + 1]$. We model the motor dynamics with a differential equation of first order

$$T_m \dot{\nu}_i = -\nu_i + \sqrt{u_i}, \tag{8}$$

where $T_m \in \mathbb{R}$ is the motor time constant. A common approach in related work is to neglect the motor dynamics and assume an instantaneous thrust acting on the rotors. However, as we observed in our experiments, an accurate actuator model is crucial for a successful sim-to-real transfer. Additionally, we model the overall latency Δ of the system

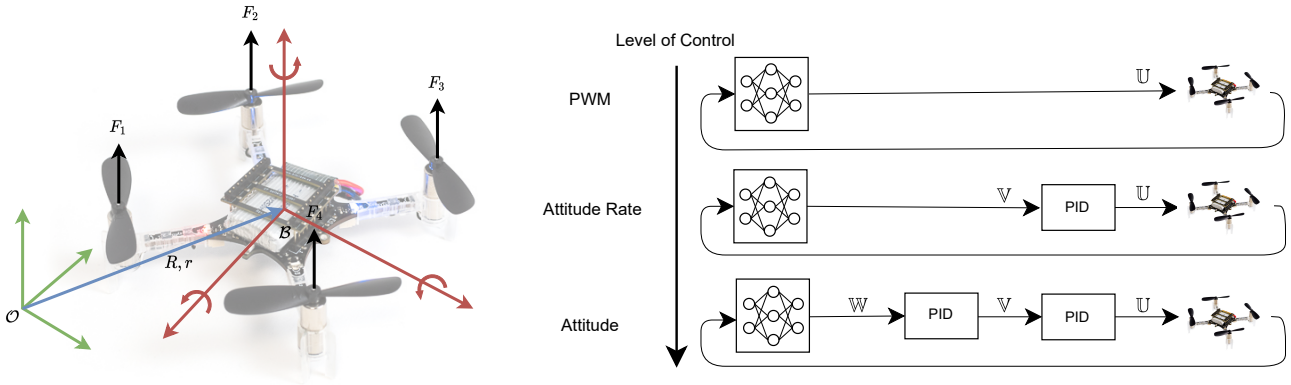


Fig. 1: The CrazyFlie Quadrotor and the investigated control structures. (Left) Overview and coordinates frames of the *CrazyFlie* quadrotor. (Right) Applied control structures from low-level PWM to high-level Attitude.

which should capture all delays emerging in the hardware. Typically, latency arises in the state estimator, in the actuation of the motors and by propagating data through the neural network policy.

IV. METHODS

We aim to find a policy that reliably transfers from the simulation to the quadrotor robot without using real-world data to fine-tune the policy. To render a zero-shot transfer feasible, we optimize the simulation parameters based on collected real-world measurements.

Besides simulation optimization, we want to find an answer to our hypothesis that *low-level control structures demand a higher simulation fidelity.* The intuition behind this assumption is that low-level control requires the policy to capture the mapping from drone state to motor commands and thus to understand to underlying quadrotor dynamics. High-level control, on the other hand, encourages the agent to learn an abstract understanding of the task since only the mapping from the drone state to an intermediate space is required. The mapping from the policy output space to the motor commands is subsequently handled by PID controllers.

The remainder of this section describes the details of our simulation environment including the observation and action space, the procedure of the simulation optimization using Bayesian optimization and an introduction to the three control structures that were deployed and evaluated on the real-world quadrotor.

A. Simulation Environment

We use the physics simulator PyBullet [3] to calculate the dynamics of the quadrotor akin to the work of [21]. We run the simulator and the motor dynamics with 200Hz while the agent receives noisy observations with $\leq$ 100Hz depending on the selected control structure. The implementation is publicly available at: `https://github.com/SvenGronauer/phoenix-drone-simulation`.

a) Observations: In every time step t, the agent receives the observation $\boldsymbol{s}_t = [\boldsymbol{x}_t^T, \boldsymbol{e}_t^T, \boldsymbol{a}_{t-1}^T]^T \in \mathbb{R}^{20}$ containing the drone state vector $\boldsymbol{x} \in \mathbb{R}^{13}$, the difference vector $\boldsymbol{e} = \boldsymbol{p} - \boldsymbol{p}_{ref} \in \mathbb{R}^3$ between the drone's current position and the set-point position, and the action $\boldsymbol{a}_{t-1}^T \in \mathbb{R}^4$ taken previously. Note that the agent has not access to all information of the drone state because the rotor speeds cannot be directly measured. To cope with partial observability, the agent is equipped with a history of observations with size $H \geq 1$, i.e. $(\boldsymbol{s}_{t-H+1}, \dots, \boldsymbol{s}_t) \in \mathbb{R}^{20H}$. We conducted a study about the optimal history size $H \in [1, 2, 4, 6, 8]$ in simulation and found that $H = 2$ performed best.

b) Actions: Actions represent abstract, possibly high-level commands which are transformed by the control structure to the corresponding motor command $\boldsymbol{u} \in \mathbb{U}$. The implementation differs for each control structure and is explained in Section IV-C.

c) Domain Randomization: Throughout training, we randomize the following system parameters uniformly in the range $\pm 10\%$ of the default value: Thrust-to-weight ratio k_F, physics time-step, quadrotor mass m, diagonal of inertia matrix I, motor time constant T_m and yaw-torque factors k_{m_1} and k_{m_2}.

d) Noise: As sensor noise, we add Gaussian and uniform noise for positions $\boldsymbol{p}$, velocities $\dot{\boldsymbol{p}}$ and angles $\boldsymbol{\varphi}$. For the angle rates, we apply the sensor model proposed in [5] which adds a Gaussian noise and a time-varying bias to $\boldsymbol{\omega}$. In a similar vain to [20], we use a discretized Ornstein-Uhlenbeck process to model actuator noise which we add to $\boldsymbol{u}$.

B. Simulation Optimization

Simulation optimization is an approach to tune the parameters of a simulation through data [1]. The aim is to bring the simulation as close as possible to the real-world and ideally close the reality gap. The objective function which we want to minimize is

$$\min_{\boldsymbol{\xi} \in \Xi} F(\boldsymbol{\xi}, \mathcal{D}) = \sum_t^T \delta^t \| W \left(\boldsymbol{x}_t^{sim}(\boldsymbol{\xi}) - \boldsymbol{x}_t^{real} \right) \|_1 + \sum_t^T \delta^t \| W \left(\boldsymbol{x}_t^{sim}(\boldsymbol{\xi}) - \boldsymbol{x}_t^{real} \right) \|_2 \quad (9)$$

where T is the mini-trajectory length, $\mathcal{D}$ is the data set and $\delta \in (0, 1)$ is a factor to discount later stages of the trajectory where the divergence increases due to discrepancies between simulation and real-world. We weight the drone state difference $\boldsymbol{x}_t^{sim}(\boldsymbol{\xi}) - \boldsymbol{x}_t^{real}$ according to the matrix W with the purpose to scale angle errors and angle rate errors differently. Similar to [2], we use the sum of L1 and L2 norm.

Prior sim-to-real work reported that the accurate modeling of the actuators including dynamics, noise and delays is crucial for a successful transfer. In [22] it was pointed out that randomizing the latency of the controller and injecting sensor noise to the simulation are key components for a high success rate. Similarly, it was demonstrated in [8] that learned actuator dynamics significantly helped to reduce the reality gap. Guided by these results, we decided to optimize those simulation parameters $\boldsymbol{\xi} = [k_F, T_m, \Delta]^T$ that describe the motor behavior as well as the system latency.

Bayesian Optimization (BO) is our method of choice, as it is regarded as a good option for the optimization in continuous domains with less than 20 dimensions and is robust against stochastic function evaluations [4]. BO is particularly suited for objective functions that have long evaluation times and can cope with low sample complexity. In contrast to gradient-based methods, BO is not prone to converge to local optima.

C. Control Structures

In this work, we investigate the following three control structures, ordered from low to high-level:

1) *PWM Control:* The agent is supposed to learn a mapping $\pi : \mathbb{S} \rightarrow \mathbb{U}$ from observation space to thrust commands. The agent receives noisy observations from the Kalman state estimator with 100Hz.
2) *Attitude Rate Control:* The policy outputs lie in the space $[c, \boldsymbol{\omega}_d^T]^T \in \mathbb{V} \subseteq \mathbb{R}^4$ which consists of the mass-normalized collective thrust c and the desired body angle rate $\boldsymbol{\omega}_d$. The PID controller calculates the thrust commands $\boldsymbol{u} \in \mathbb{U}$ based on c and the error between the actual $\boldsymbol{\omega}$ and the desired $\boldsymbol{\omega}_d$. The agent receives noisy observations with 50Hz and is supposed to learn the mapping $\pi : \mathbb{S} \rightarrow \mathbb{V}$.
3) *Attitude Control:* The agent is incentivized to learn the mapping $\pi : \mathbb{S} \rightarrow \mathbb{W}$ where the policy output space $[c, \boldsymbol{\varphi}_d^T] \in \mathbb{W} \subseteq \mathbb{R}^4$ encompasses the mass-normalized collective thrust c and the desired body angle $\boldsymbol{\varphi}_d$. Subsequently, the mapping $\mathbb{W} \rightarrow \mathbb{V} \rightarrow \mathbb{U}$ is handled by two cascaded PID controllers. Noisy observations are fed into the policy with 25Hz.

These three control structures are illustrated in Fig. 1. We test our hypothesis about the fidelity of the simulator based on the deployed control method in the next section.

V. RESULTS

In this section, we describe our experiments and discuss the results. First, we elaborate on the setup of the hardware and the learning task. Second, we explain our simulation optimization experiments with BO and present our found simulation parameters. Third and last, we describe our zero-shot transfer experiments that were conducted on the real-world quadrotor. After training control policies in simulation, we studied the three control structures PWM, Attitude Rate and Attitude in order to test our control level hypothesis of Section IV.

A. Experimental Setup

As quadrotor robot, we used the *CrazyFlie 2.1*. Due to its small size with a motor-to-motor diameter of 13cm and light weight of 30g, the *CrazyFlie* is very agile and demands high control frequencies. Furthermore, due to its low-cost design, the *CrazyFlie* exhibits high parameter uncertainties in its building parts which additionally requires the controller to be robust over a large system parameter range. The drone is equipped with a 3-axis gyroscope, a 3-axis accelerometer and a z-axis LIDAR. The sensor data were fed into an extended Kalman filter that runs with 100Hz for drone state estimation. For our experiments, we only used onboard sensor measurements and did not use an external tracking system. For evaluation purposes, we transmitted the flight information via *CrazyRadio* to a host computer where we analyzed the logged data.

As learning task, we want the quadrotor to fly a circle figure with a diameter of 0.5m and with a period of 3s. Each trajectory starts on a random point of the reference circle at the height of 1m going in clock-wise direction. The trajectory length in simulation is 500 time steps. To incentivize the drone following the set-point trajectory $\boldsymbol{p}_{ref}$, the reward function is designed as

$$\begin{aligned} r(\boldsymbol{s}, \boldsymbol{a}, t) = &- \|\boldsymbol{e}\|_2 - 0.0001\|0.5(\boldsymbol{a}_t + 1)\|_2 \\ &- 0.001\|\boldsymbol{a}_{t-1} - \boldsymbol{a}_t\|_2 - 0.001\|\boldsymbol{\omega}\|_2 + r_f \end{aligned} \quad (10)$$

with $r_f = -100$ being the terminal reward if $\|\boldsymbol{e}\|_2 > 0.25$ else $r_f = 0$. The actions are clipped into $[-1, 1]$ to regard actuator limits.

B. Simulation Optimization

We used the pre-implemented cascaded PID position controller from the *CrazyFlie* platform to collect data tuples $(\boldsymbol{x}, \boldsymbol{u})_t$ while the drone was flying the circle figure as described in Section V-A. Data were logged with 100Hz and we collected approximately one hour of real-world data. We think that less data would be also sufficient but we did not consider the amount of samples in this work. For building the data set $\mathcal{D}$, we used every tenth data-point from the collected real-world data as starting state for a mini-trajectory of length T. For instance, for $T = 50$ we obtained a data set of size $\mathcal{D} \in \mathbb{R}^{37673 \times 50 \times 13}$.

With BO, we opted to find the global optimum over the bounded set of system parameters $k_F \in [1.5, 2.5]$, $T_m \in [0.01, 0.50]$ and $\Delta \in [0.00, 0.05]$ with respect to $\mathcal{D}$. We ran the experiment for 250 evaluations on the objective from (9) and averaged the results over three trials. We tested different mini-trajectory lengths $T \in \{10, 20, 30, 40, 50\}$. As hyper-parameters, we used $\delta = 0.95$ as the discount factor and as function approximator we used a Gaussian process

TABLE I: Simulation optimization results found by BO. The mean and standard deviation was calculated over three trials.

Parameter	$T = 10$	$T = 20$	$T = 30$	$T = 40$	$T = 50$
Thrust-weight k_F	1.746 ± 0.0050	1.733 ± 0.0093	1.725 ± 0.0018	1.725 ± 0.0081	1.722 ± 0.0027
Time Constant T_m	0.102 ± 0.0110	0.087 ± 0.0087	0.088 ± 0.0006	0.096 ± 0.0035	0.104 ± 0.0012
Latency Δ	0.006 ± 0.0039	0.014 ± 0.0038	0.018 ± 0.0003	0.018 ± 0.0007	0.018 ± 0.0004

with the acquisition function uniformly chosen from lower confidence bound, expected improvement and probability of improvement.

The results obtained by the simulation optimization are shown in Table I. Contrary to our expectation, the mini-trajectory length T had only a minor impact on the obtained results. Except for the results for $T = 10$, the found parameters $\boldsymbol{\xi}$ lie in a similar range.

C. Zero-Shot Experiments

We conducted zero-shot experiments on the real quadrotor where we first trained policy networks with different simulation parameters and thereafter measured the transfer performance. We decided on such experimental setup for three reasons. First, we wanted to evaluate the performance of the parameters found by simulation optimization for an induced reality gap. Second, we intended to study the robustness of the policies trained in different scenarios. Third and last, we wanted to test our control level hypothesis as stated in Section IV.

We aligned the hyper-parameters to the ones suggested in [6] and [7]. We applied a distributed learner setup where the policy gradients were computed and averaged across 64 workers with a total batch size of 64000. We trained with the Proximal Policy Optimization [24] algorithm over 500 iterations and applied as discount factor $\gamma = 0.99$. For the neural network architecture, we used multi-layer perceptrons with two hidden layers. The critic network used 64 neurons each followed by tanh non-linearities whereas the actor had 50 neurons in each hidden layer with ReLU as activations. Since the policy network is supposed to run on the drone micro-controller, we reduced the number of hidden neurons to achieve an inference time of $<$ 1ms for one forward pass. The weights were set according to Kaiming Uniform and biases were initialized as zero vectors. We did not apply parameter sharing between the actor and the critic. Both networks were optimized with Adam where the learning rate of the value network was 0.001 and 0.0003 for the policy. To reduce the variance of critic estimates, we applied Generalized Advantage Estimation (GAE) [23] with the weighting factor $\lambda = 0.95$. Over the training, we used stochastic policies where the policy output is the mean of a multi-variate Gaussian distribution $\boldsymbol{a} \sim \mathcal{N}(\pi(\boldsymbol{s}), \epsilon I)$ with I being the identity matrix and $\epsilon \in \mathbb{R}$ the exploration noise that was linearly annealed from 0.5 to 0.01. Fig. 2 shows in the top row the learning curves of the RL training. Note that the return generated by PWM is not comparable to the returns of Attitude and Attitude Rate since the reward function prefers negative actions over zero actions. Attitude and Attitude Rate control typically produce zero outputs for angle stabilization. Returns larger than the terminal reward r_f indicate that the quadrotor does not crash and tracks the set point. For the three control structures, all trained policies were capable of tracking the set-point in simulation.

After training three policies for each combination of motor time constant $T_m \in \{0.04, 0.08, 0.12\}$s and latency $\Delta \in \{0, 0.015, 0.02\}$s in simulation, we tested the policies on the real quadrotor robot. We applied this evaluation procedure for each of the three control structures and determined the flight time of policies over three real-world trials. Overall, this resulted in $3^4 = 81$ trained neural networks and thus 243 flights. If a policy was able to fly longer than 20s, we manually stopped the execution of the neural network. If the policy was not able to stabilize the quadrotor, the trial was terminated by a stabilizing backup PID controller, which intervened when the roll or pitch angle was greater than $30°$ or when roll and pitch rates exceeded $800°$/s. We observed that the rollouts also failed when the distance between the drone and the set-point exceeded the episode termination criterion of $\|\boldsymbol{e}\|_2 > 0.25$m. The results are displayed in the bottom row of Fig. 2.

The PWM control structure showed good zero-shot performance when trained on the motor time constant $T_m = 120$ms with latency $\Delta \geq 15$ms. The parameter setting $T_m = 80$ms, $\Delta = 20$ms closest to the parameters suggested by the simulation optimization resulted in a median flight time of 20s and two outlier flights. However, outside this parameter setting, PWM fastly dropped in performance and often failed. When the latency was set to $\Delta = 0$ms, PWM yielded the worst performance compared to the other two control approaches.

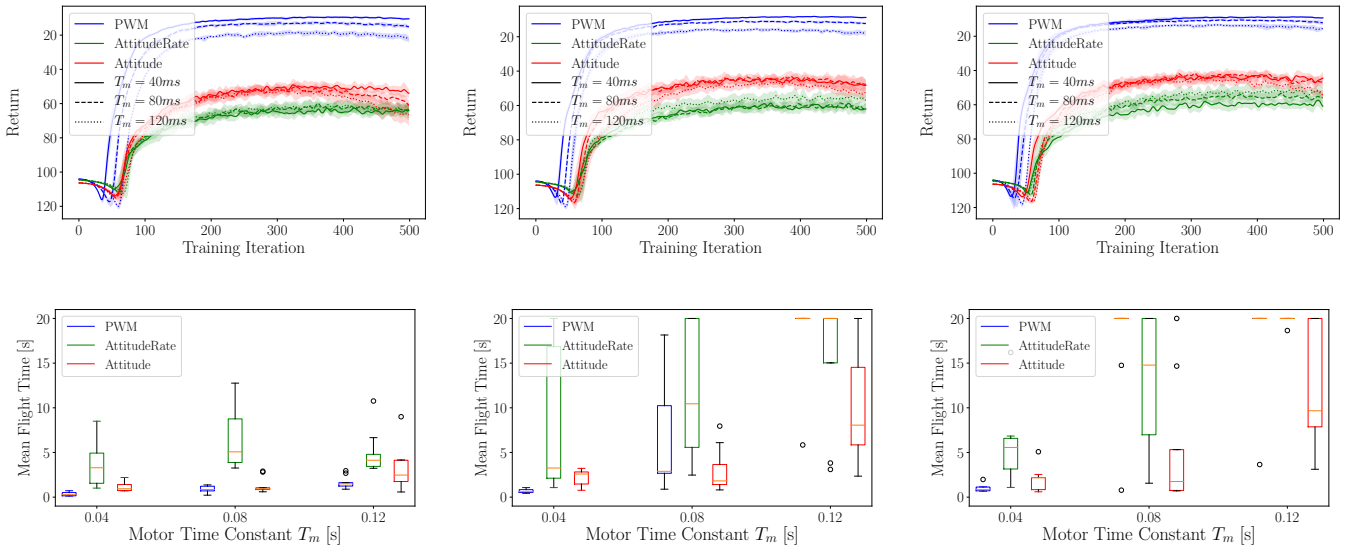
Overall, the Attitude Rate control structure showed the most robust results over the tested system parameters. The best performance was observed for the setting $T_m = 120$ms and $\Delta = 20$ms. In contrast to the other control structures, Attitude Rate was also able to stabilize the quadrotor when trained on latency $\Delta = 0$ms, although most flights terminated early due to the $\|\boldsymbol{e}\|_2 > 0.25$m termination criterion.

Surprisingly, Attitude control showed the worst performance despite being the highest control level structure. Only the policies that were trained on settings with $T_m = 120$ms and latency $\Delta \geq 15$ms could achieve a flight duration over 10s.

VI. DISCUSSION

In this section, we discuss our results from the experiments and draw connections to related work. Further, we point out important observations that we made during our experiments.

a) Simulation Optimization: The system parameters closest to the ones found by the simulation optimization showed a reasonable transfer success for the PWM and



(a) Latency $\Delta = 0$ms (b) Latency $\Delta = 15$ms (c) Latency $\Delta = 20$ms

Fig. 2: Training curves in simulation and evaluation of real-world experiments. (Top) The learning curves of the training in simulation are shown for the three different control structures and different values for (T_m, Δ). Each curve is the avarage over three independent runs while the shaded area denotes the standard deviation. The difference in return is due to the action penalties that prefer negative actions over zero actions. All trained policies were able to track the set-point trajectory in simulation. (Bottom) The flight times in the zero-shot experiments were evaluated for different system parameters and control structures. Each box represents nine real-world flights with a maximum flight time of 20s after which we manually stopped the policy execution.

Attitude Rate control. This underlines the potential of data-driven parameter estimation compared to classical methods known from system identification. A benefit is the ease of use since a handful of system parameters can be estimated simultaneously and no isolated measurements of individual physical properties must be conducted.

b) Control-level Hypothesis: For low-level PWM control, we found that good zero-shot performance can only be achieved when the dynamics of the actuator and the latency are accurately estimated. However, if the reality gap is too large, low-level policy control is prone to fail. The higher-level Attitude Rate control, in contrast, shows high robustness towards the choice of system parameters. This confirms our control level hypothesis that low-level control requires higher simulation fidelity than higher-level control structures. Merely Attitude control showed disappointing results in the zero-shot experiments, which we think is due to the small control frequency of 25Hz.

c) Actuator Model: Related work from quadrupedal robots [8] and robotic manipulators [22] suggested that accurate modeling of the actuators is a crucial component for reliable zero-shot policy transfers. Our experiments confirmed that this statement is also valid for quadrotor robots. In addition to that, we observed that adding latency to the simulation acts as a kind of regularization which helped to improve the zero-shot results.

d) Robustness: Due to crashes, we frequently changed spare hardware parts like propellers and motors. Although facing a variety of drone parameters, the zero-shot transfers worked reliably. Moreover, we tested some of the trained policies also on other *CrazyFlie* drones which showed similar flight behavior. This indicates that the trained policies are robust over a large parameter distribution and different drone systems.

e) Bang-bang behavior: Bang-bang behavior is a known issue in RL where policies prefer an action selection towards the boundaries of the action space [25]. Without adding penalties to the actions, we observed that agents produced large changes in the control outputs which caused a severe performance drop in the transfers. By adding penalties for actions $\|\boldsymbol{a}_t\|_2$ and action rates $\|\boldsymbol{a}_{t-1} - \boldsymbol{a}_t\|_2$, we were able to conduct reliable transfers to the real robot. Similar conclusions were made in sim-to-sim experiments where penalties for action and action rate improved both the transfer and the robustness towards disturbances [25]. Further, we noticed that the transfers failed when the action range was selected too high, e.g. an Attitude rate control with the range $[-360, 360]°$/s did not work whereas the range $[-60, 60]°$/s showed the desired results.

VII. CONCLUSION

In this paper, we demonstrated that low-level control policies trained with Reinforcement Learning entirely in simulation can be successfully transferred to a quadrotor robot when the reality gap is small. To render such zero-shot policy transfers feasible, we narrowed the sim-to-real gap by collecting real-world data with a pre-implemented PID controller and applyed simulation optimization. Our neural

network-based policies used onboard sensing and computation only. Finally, we conducted real-world experiments and compared three different control structures ranging from low-level PWM motor commands to high-level Attitude control based on cascaded PID controllers. The experiments confirm our hypothesis that low-level control policies require a higher simulation fidelity, which suggests the use of higher-level action structures like Attitude Rate control when the robot system parameters cannot be accurately estimated or are partially unknown.

APPENDIX

TABLE II: Used drone parameters.

Parameter	Value	Physical Unit
Gravitational acceleration g	9.81	m·s ^{2}
Inertial I_{xx}	$1.33 \cdot 10^{-5}$	kg·m^2
Inertial I_{yy}	$1.33 \cdot 10^{-5}$	kg·m^2
Inertial I_{zz}	$2.64 \cdot 10^{-5}$	kg·m^2
Length L	0.0396	m
Mass m	0.030	kg
Torque-to-weight ratio k_{M_1}	$5.96 \cdot 10^{-3}$	m
Torque-to-weight ratio k_{M_2}	$1.56 \cdot 10^{-5}$	N·m

ACKNOWLEDGMENT

We thank Matthias Emde for his support with the implementation of the drone firmware. The project was supported by the German Federal Ministry of Education and Research under grant number 01IS17049. The authors are responsible for the content of this publication.

REFERENCES

[1] Y. Carson and A. Maria, "Simulation optimization: Methods and applications," in *Proceedings of the 29th Conference on Winter Simulation*, ser. WSC '97. USA: IEEE Computer Society, 1997, pp. 118–126.

[2] Y. Chebotar, A. Handa, V. Makoviychuk, M. Macklin, J. Issac, N. Ratliff, and D. Fox, "Closing the sim-to-real loop: Adapting simulation randomization with real world experience," in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 8973–8979.

[3] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation for games, robotics and machine learning," http://pybullet.org, 2016–2021, accessed: 2021-12-19.

[4] P. I. Frazier, "A tutorial on bayesian optimization," *CoRR*, vol. abs/1807.02811, 2018.

[5] F. Furrer, M. Burri, M. Achtelik, and R. Siegwart, *RotorS—A Modular Gazebo MAV Simulator Framework*. Springer International Publishing, 2016, pp. 595–625.

[6] S. Gronauer, M. Gottwald, and K. Diepold, "The successful ingredients of policy gradient algorithms," in *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, 8 2021, pp. 2455–2461.

[7] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger, "Deep reinforcement learning that matters," in *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*. AAAI Press, 2018, pp. 3207–3214.

[8] 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, 2019.

[9] E. Kaufmann, A. Loquercio, R. Ranftl, M. Müller, V. Koltun, and D. Scaramuzza, "Deep drone acrobatics," in *Proceedings of Robotics: Science and Systems*, Corvalis, Oregon, USA, July 2020.

[10] R. Kirk, A. Zhang, E. Grefenstette, and T. Rocktäschel, "A survey of generalisation in deep reinforcement learning," *CoRR*, vol. abs/2111.09794, 2021.

[11] J. E. Kooi and R. Babuska, "Inclined quadrotor landing using deep reinforcement learning," in *IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2021, Prague, Czech Republic, September 27 - Oct. 1, 2021*. IEEE, 2021, pp. 2361–2368.

[12] N. O. Lambert, D. S. Drew, J. Yaconelli, S. Levine, R. Calandra, and K. S. J. 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.

[13] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," in *4th International Conference on Learning Representations*, 2016.

[14] A. Loquercio, E. Kaufmann, R. Ranftl, A. Dosovitskiy, V. Koltun, and D. Scaramuzza, "Deep drone racing: From simulation to reality with domain randomization," *IEEE Transactions on Robotics*, 2019.

[15] A. Loquercio, E. Kaufmann, R. Ranftl, M. Müller, V. Koltun, and D. Scaramuzza, "Learning high-speed flight in the wild," *Science Robotics*, vol. 6, no. 59, 2021.

[16] S. Lupashin, A. Schöllig, M. Sherback, and R. D'Andrea, "A simple learning strategy for high-speed quadrocopter multi-flips," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA) 2010*, Zurich, 2009.

[17] R. Mahony, V. Kumar, and P. Corke, "Multirotor aerial vehicles: Modeling, estimation, and control of quadrotor," *IEEE Robotics Automation Magazine*, vol. 19, no. 3, pp. 20–32, 2012.

[18] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *2011 IEEE International Conference on Robotics and Automation*, 2011, pp. 2520–2525.

[19] 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, pp. 529 EP –, 02 2015.

[20] A. Molchanov, T. Chen, W. Hönig, J. A. Preiss, N. Ayanian, and G. S. Sukhatme, "Sim-to-(multi)-real: Transfer of low-level robust control policies to multiple quadrotors," *CoRR*, vol. abs/1903.04628, 2019.

[21] J. Panerati, H. Zheng, S. Zhou, J. Xu, A. Prorok, and A. P. Schoellig, "Learning to fly—a gym environment with pybullet physics for reinforcement learning of multi-agent quadcopter control," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021.

[22] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Sim-to-real transfer of robotic control with dynamics randomization," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 3803–3810.

[23] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," in *Proceedings of the International Conference on Learning Representations*, 2016.

[24] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *CoRR*, vol. abs/1707.06347, 2017.

[25] T. Seyde, I. Gilitschenski, W. Schwarting, B. Stellato, M. Riedmiller, M. Wulfmeier, and D. Rus, "Is bang-bang control all you need? solving continuous control with bernoulli policies," in *Thirty-Fifth Conference on Neural Information Processing Systems*, 2021.

[26] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. Lillicrap, K. Simonyan, and D. Hassabis, "A general reinforcement learning algorithm that masters chess, shogi, and go through self-play," *Science*, vol. 362, no. 6419, pp. 1140–1144, 2021/12/10 2018.

[27] 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, pp. 23–30.