Accurate Dynamics Models for Agile Drone Flight: Zero-Shot Sim2Real-Transfer of Neural Controllers

Leonard Bauersfeld, Davide Scaramuzza

***Abstract*— Quadrotors are extremely agile, so much in fact, that classic first-principle-models come to their limits. Aerodynamic effects, while insignificant at low speeds, become the dominant model defect during high speeds or agile maneuvers. Battery non-idealities, such as voltage drops under high loads significantly deteriorate the performance of the robot. Accurate modeling is needed to design robust high-performance control systems that enable flying close to the platform's physical limits. We propose a hybrid approach fusing first-principle models and datadriven methods to model quadrotors accurately such that we can (i) use the simulation to train neural controllers via reinforcement learning (RL) and zero-shot transfer them to the real-world and (ii) verify a controller's performance in simulation. While advanced first-principle models such as computational fluid dynamics or molecular dynamics simulations are able to precisely calculate the aerodynamic forces and the battery dynamics of a quadrotor, their computational demand makes them unusable for RL training requiring millions of samples. We show that by augmenting simpler models with data-driven residual models, we can develop a simulation that is very fast to run and yet accurately captures the vehicle's dynamics. The approach is validated in the real-world by comparing our simulation with the experiments where autonomous quadrotor flies at speeds up to 70 km/h in one the world's largest motion capture systems.**

I. Introduction

Recently, learned controllers have become extremely popular in the mobile robotics community due to their success in a variety of complex real-world tasks, such as legged locomotion in challenging environments [1], underground exploration [2] and autonomous drone racing [3]–[5]. In all the aforementioned works, neural-network controllers outperform their classical model-based counterparts both in terms of performance and success rate but they require a simulation environment to be trained in. The performance of the trained neural controllers typically depends on the fidelity of the training environment and a smaller sim2real gap means that the controller can push the robot closer to its limits. The sim2real gap can be narrowed by employing strategies like domain randomization ensuring that the neural controller learns to cope with a distribution of vehicle dynamics. If this distribution is chosen sufficiently wide, the dynamics of the physical robot will be included in the training distribution. While recent publications show that training using a very simple dynamics model and applying domain randomization is viable [1], [2], [5] it suffers from two drawbacks as (i) the learned controller is only optimal w.r.t the distribution of dynamics and not w.r.t the actual physics of the robot and (ii) it makes validation of the controller in simulation more difficult since we have no guarantee that the training distribution indeed captures the true vehicle dynamics. To overcome these difficulties we propose to increase the simulation fidelity when simulating agile quadcopters by augmenting the simple models used in prior works with data-driven residuals that capture hard-to-model aerodynamic effects and non-idealities in the propulsion system.

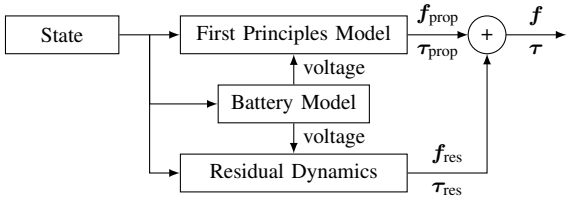

Fig. 1. Overview over the proposed hybrid model architecture to predict the dynamics of an agile quadcopter. A computationally lightweight first-principles propeller model is augmented with a data-driven residual model. To further improve the accuracy a battery model is added to account for the time-varying battery voltage.

II. Related Work

Traditionally, a rotor is assumed to produce thrust and axial torque proportional to the square of its angular rate with a constant coefficient [6], which is referred to hereinafter as the simple quadratic model. Due to its simplicity and computational performance, this model is used in well-known aerial robotics simulators such as AirSim [7], Flightmare [8], RotorS [9] and others [10]. To increase model fidelity, [11] proposes to identify the quadrotor platform dynamics using a gray-box model that uses a library of polynomials as basis functions and is able to model both aerodynamic forces and torques. This method relies on the predefined function library and also contains discontinuities in the learned model. To further increase the fidelity and reduce the risk of large discontinuities, [12] uses a first-principles model and only augments that with a learned component for the residual force and torque terms. Inspired by these prior works, in this short paper we examine a setting where we use a simple first-principle model similar to [12] but augment it with a data-driven residual component that relies on polynomials as basis functions [11]. We follow a similar approach to battery modeling, using a graybox battery model based on a Thevenin *equivalent circuit*. Depending on the fidelity of the model, it includes one resistor combined with zero, one (one time constant, OTC) or two (two time constants, TTC) capacitive networks. A review of the common OTC and TTC models is presented in [13]. The OTC model is widely used because of its well-established accuracy [14].

The authors are with the Robotics and Perception Group, Department of Informatics, University of Zurich, and Department of Neuroinformatics, University of Zurich and ETH Zurich, Switzerland (http://rpg.ifi.uzh.ch). This work was supported by the Swiss National Science Foundation (SNSF) through the National Centre of Competence in Research (NCCR) Robotics, the European Union's Horizon 2020 Research and Innovation Programme under grant agreement No. 871479 (AERIAL-CORE), and the European Research Council (ERC) under grant agreement No. 864042 (AGILEFLIGHT).

III. Drone Dynamics Simulator

A. First-principles Model

The quadrotor is assumed to be a 6 degree-of-freedom rigid body of mass m and diagonal moment of inertia matrix $\boldsymbol{J} = \mathrm{diag}(J_x, J_y, J_z)$. Furthermore, the battery voltage U dependent rotational speeds of the four propellers Ω_i are modeled as a first-order system with time constant k_{mot} where the commanded motor speeds $\boldsymbol{\Omega}_{\text{cmd}}$ are the input. The state space is 17-dimensional and its dynamics can be written as:

$$\dot{\boldsymbol{x}} = \begin{bmatrix} \dot{\boldsymbol{p}}_{\mathcal{WB}} \\ \dot{\boldsymbol{q}}_{\mathcal{WB}} \\ \dot{\boldsymbol{v}}_{\mathcal{W}} \\ \dot{\boldsymbol{\omega}}_{\mathcal{B}} \\ \dot{\boldsymbol{\Omega}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{v}_{\mathcal{W}} \\ \boldsymbol{q}_{\mathcal{WB}} \cdot \begin{bmatrix} 0 & \boldsymbol{\omega}_{\mathcal{B}}/2 \end{bmatrix}^\top \\ \frac{1}{m}\left(\boldsymbol{q}_{\mathcal{WB}} \odot \left(\boldsymbol{f}_{\text{prop}} + \boldsymbol{f}_{\text{res}}\right)\right) + \boldsymbol{g}_{\mathcal{W}} \\ \boldsymbol{J}^{-1}\left(\boldsymbol{\tau}_{\text{prop}} + \boldsymbol{\tau}_{\text{res}} - \boldsymbol{\omega}_{\mathcal{B}} \times \boldsymbol{J}\boldsymbol{\omega}_{\mathcal{B}}\right) \\ \frac{1}{k_{\text{mot}}}\left(\boldsymbol{\Omega}_{\text{cmd}}(U) - \boldsymbol{\Omega}(U)\right) \end{bmatrix}, \quad (1)$$

where $\boldsymbol{g}_{\mathcal{W}} = [0, 0, -9.81\,\mathrm{ms}^{-2}]^\top$ denotes earth's gravity, $\boldsymbol{f}_{\text{prop}}$, $\boldsymbol{\tau}_{\text{prop}}$ are the collective force and the torque produced by the propellers. Model-free reinforcement learning suffers from a low sample-efficiency during training which necessitates an efficient simulator that can run fast. Hence, to model the thrust and torque produced by the i-th propeller, the commonly used and computationally lightweight quadratic model [7], [9], [15] is employed:

$$\boldsymbol{f}_i(\Omega) = \begin{bmatrix} 0 & 0 & c_{\text{l}}\,\Omega^2 \end{bmatrix}^\top \quad \boldsymbol{\tau}_i(\Omega) = \begin{bmatrix} 0 & 0 & c_{\text{d}}\,\Omega^2 \end{bmatrix}^\top \quad (2)$$

$$\boldsymbol{f}_{\text{prop}} = \sum \boldsymbol{f}_i \qquad \boldsymbol{\tau}_{\text{prop}} = \sum \boldsymbol{\tau}_i + \boldsymbol{r}_{\text{P},i} \times \boldsymbol{f}_i \quad (3)$$

where c_{l}, c_{d} denote the lift and drag coefficient of propeller respectively and $\boldsymbol{r}_{\text{P}}$.

B. Residual Model

Compared to state-of-the-art methods that leverage blade-element-momentum theory [12], this quadratic model does not account for aerodynamic effects, such as rotor drag or blade flapping. This deficiency widens the sim-to-real gap when using RL to train a controller and then deploy in the real-world. We account for such residual aerodynamic effects and introduce a lumped residual term $\boldsymbol{f}_{\text{res}}$, $\boldsymbol{\tau}_{\text{res}}$ on the forces and torques respectively. We use a polynomial graybox model [12], [16] for the residual terms as this is computationally lightweight. The polynomial basis functions are a linear and quadratic terms of the velocity components in bodyframe, the squared average motor speed and all interaction terms between the terms. The coefficients of the polynomial model are fitted from measurements from real-world flights in a motion-capture system similar to [12].

C. Battery Model

Battery voltage models leverage Thevenin equivalent circuits to predict the battery voltage. Fig. 2 shows the equivalent circuit diagram for the used OTC (one time constant) battery model. The voltage of the voltage source U_0 corresponds to the open-circuit voltage of the battery. When a possibly time-varying load is connected to the circuit and a current $I_{\text{load}}(t)$ flows, the voltage $U_{\text{bat}}(t)$ at the output terminals can be calculated as [17]

$$\dot{U}_{\text{cap}}(t) = \frac{-U_{\text{cap}}(t)}{R_1 \cdot C_1} + \frac{I_{\text{load}}(t)}{C_1}, \quad (4)$$

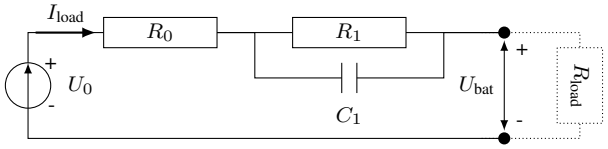

Fig. 2. Thevenin equivalent circuit for the one-time-constant (OTC) battery model. The load does not need to be static but could, for example, be a multirotor aerial vehicle.

$$U_{\text{bat}}(t) = U_0(t) - U_{\text{cap}}(t) - R_0(t) I_{\text{load}}(t), \quad (5)$$

where $R_0(t)$, R_1, C_1 are defined as shown in Fig 2. The reader is reffered to our prior work [18] for further detail.

IV. Neural Controller

In this work, the task of fast and agile quadrotor flight is defined as navigating through a sequence of drone racing gates as fast as possible. Or, to rephrase this using broader terms: Navigate through a sequence of predefined waypoints g_i (see Fig. 3) in minimum time and pass each waypoint within an l_∞ distance less then the dimension of a racing gate. To accomplish this, the control policy directly maps an observation $\boldsymbol{o}_t$ and a conditioning input ζ_t to an action (control command) $\boldsymbol{a}_t$. The control policies are trained using model-free reinforcement learning (PPO [19]) purely in simulation.

1) Observation and Action Space: At each timestep t the policy has access to an observation $\boldsymbol{o}_t$ from the environment which contains (i) the current robot state, (ii) the relative position to the next waypoint to be passed, Specifically, the state consists of the vehicle position $\boldsymbol{p}_{\mathcal{WB}}$, its velocity in body-frame $\boldsymbol{v}_{\mathcal{B}}$ and its attitude. To avoid discontinuities the latter is represented by a rotation matrix instead of directly using the quaternion $\boldsymbol{q}_{\mathcal{WB}}$ [20]. The control command $\boldsymbol{a}_t$ consists of a desired collective mass-normalized thrust c and a bodyrate setpoint $\boldsymbol{\omega}_{\mathcal{B},\text{ref}}$. Those commands are then tracked by a low-level controller, which finally controls the motors. In contrast to more abstract control modalities such as linear velocity references, operating on collective thrust and bodyrates has been shown to be well suited for agile learned quadrotor control [21].

2) Reward Function: We use a dense shaped reward to encode the task of high-speed flight through a set of predefined waypoints. The reward r_t at time step t is given by

$$r_t = r_t^{\text{prog}} + r_t^{\text{perc}}(\zeta) - r_t^{\text{twr}}(\zeta) - r_t^{\text{crash}}, \quad (6)$$

where r^{prog} rewards progress towards the next gate to be passed [5], $r^{\text{perc}}(\zeta)$ encodes perception awareness by adjusting the vehicle's attitude such that the optical axis of its camera points towards the next gate's center with an optional

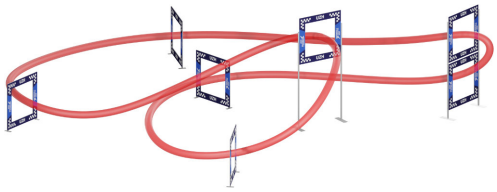

Fig. 3. We evaluate neural policy on the task of fast waypoint flight on a challenging track spanning an area of 12 m × 16 m.

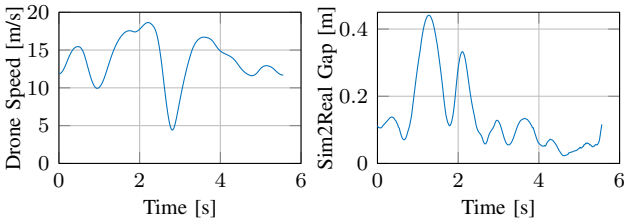

Fig. 4. The left plot shows the velocity of the quadcopter along the track. The right plot shows the positional difference between a simulated rollout and the real-world experiment.

user-specified offset, $r^{\text{twr}}(\zeta)$ is a penalty for violating the user-specified maximum thrust-to-weight ratio, and r^{crash} is a binary penalty that is only active when colliding with a gate or when the platform leaves a pre-defined bounding box, which also ends the episode.

Progress, perception, thrust-to-weight, and collision reward components are formulated as follows:

$$\begin{aligned} r_t^{\text{prog}} &= \lambda_1 \left(d_{\text{Gate}}(t-1) - d_{\text{Gate}}(t)\right) \\ r_t^{\text{perc}}(\zeta) &= \lambda_2 \exp\left(\lambda_3 \cdot \delta_{\text{cam}}(\zeta)^4\right) \\ r_t^{\text{twr}}(\zeta) &= \max(\lambda_4 \exp\left(\lambda_5 (c_{\text{cmd}} - c_{\text{twr}}(\zeta)) \;/\; c_{\text{max}}\right) - 1, 0) \\ r_t^{\text{crash}} &= \begin{cases} -5.0, & \text{if } p_z < 0 \text{ or in collision with gate.} \\ 0, & \text{otherwise} \end{cases}, \end{aligned} \tag{7}$$

where $d_{\text{Gate}}(t)$ denotes the distance from the quadrotor's center of mass to the center of the next gate, $\delta_{\text{cam}}(\zeta)$ is the angle between the optical axis of the camera and the direction towards the center of the next gate. The parameters c_{cmd}, $c_{\text{twr}}(\zeta)$ and c_{max} are the commanded mass normalized thrust, the current user-specified maximum allowable mass normalized thrust and the maximum mass normalized thrust physically available for the quadrotor, respectively. The hyperparameters $\lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda_5$ trade-off objectives regarding perception awareness and thrust-to-weight ratio constraints against progress objectives.

3) Policy Training: All control policies are trained using Proximal Policy Optimization (PPO) [19]. PPO has been shown to achieve state-of-the-art performance on a set of continuous control tasks and is well suited for learning problems where interaction with the environment is fast. Data collection is performed by simulating 100 agents in parallel using TensorFlow Agents [22]. At each environment reset, every agent is initialized in a random gate on the track layout with bounded perturbation around a state previously observed when passing the respective gate.

V. Experimental Results

The quadcopter flies at speeds up to 19 m/s (see Fig. 4) and experiences accelerations up to 4.5 g. To validate the accuracy of our model, we deploy the policy zero-shot in the real world in one of the worlds largest motion capture systems. In Fig. 4 (right) we show the distance between a simulated rollout and a real-world rollout. On average the two trajectories are 0.13 m apart, highlighting how accurate the dynamics model is. In our prior work [12] we achieved a similar accuracy only at lower speeds and had to rely on a neural network to capture the residual dynamics. The high simulation accuracy is enabled by the inclusion of a residual model and an accurate battery model.

References

[1] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning quadrupedal locomotion over challenging terrain," *Science robotics*, vol. 5, no. 47, 2020.

[2] M. Tranzatto, T. Miki, M. Dharmadhikari, L. Bernreiter, M. Kulkarni, F. Mascarich, O. Andersson, S. Khattak, M. Hutter, R. Siegwart, and K. Alexis, "Cerberus in the darpa subterranean challenge," *Science Robotics*, vol. 7, no. 66, p. eabp9742, 2022.

[3] P. Foehn, D. Brescianini, E. Kaufmann, T. Cieslewski, M. Gehrig, M. Muglikar, and D. Scaramuzza, "Alphapilot: Autonomous drone racing," *Auton. Robots*, vol. 46, no. 1, p. 307–320, 2022.

[4] E. Ackerman, "Autonomous Drones Challenge Human Champions in First "Fair" Race," *IEEE Spectrum*.

[5] Y. Song, M. Steinweg, E. Kaufmann, and D. Scaramuzza, "Autonomous drone racing with deep reinforcement learning," *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*, 2021.

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

[7] S. Shah, D. Dey, C. Lovett, and A. Kapoor, "Airsim: High-fidelity visual and physical simulation for autonomous vehicles," in *Field and Service Robot.*, Springer, 2018.

[8] Y. Song, S. Naji, E. Kaufmann, A. Loquercio, and D. Scaramuzza, "Flightmare: A flexible quadrotor simulator," 2020.

[9] F. Furrer, M. Burri, M. Achtelik, and R. Siegwart, "Rotors—a modular gazebo mav simulator framework," in *Robot Operating System (ROS)*, Springer, 2016.

[10] J. Meyer, A. Sendobry, S. Kohlbrecher, U. Klingauf, and O. Von Stryk, "Comprehensive simulation of quadrotor uavs using ros and gazebo," in *International conference on simulation, modeling, and programming for autonomous robots*, pp. 400–411, Springer, 2012.

[11] S. Sun, C. C. de Visser, and Q. Chu, "Quadrotor gray-box model identification from high-speed flight data," *Journal of Aircraft*, vol. 56, no. 2, pp. 645–661, 2019.

[12] L. Bauersfeld, E. Kaufmann, P. Foehn, S. Sun, and D. Scaramuzza, "Neurobem: Hybrid aerodynamic quadrotor model," in *Proceedings of Robotics: Science and Systems*, 2021.

[13] X. Zhang, W. Zhang, and G. Lei, "A review of li-ion battery equivalent circuit models," *Transactions on Electrical and Electronic Materials*, 2016.

[14] L. Zhang, S. Wang, D.-I. Stroe, C. Zou, C. Fernandez, and C. Yu, "An accurate time constant parameter determination method for the varying condition equivalent circuit model of lithium batteries," *Energies*, 2020.

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

[16] S. Sun, C. C. de Visser, and Q. Chu, "Quadrotor gray-box model identification from high-speed flight data," *Journal of Aircraft*, vol. 56, no. 2, pp. 645–661, 2019.

[17] A. Rahmoun and H. Biechl, "Modelling of li-ion batteries using equivalent circuit diagrams," *Przeglad Elektrotechniczny*, 2012.

[18] L. Bauersfeld and D. Scaramuzza, "Range, endurance, and optimal speed estimates for multicopters," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 2953–2960, 2022.

[19] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv e-prints*, 2017.

[20] Y. Zhou, C. Barnes, J. Lu, J. Yang, and H. Li, "On the continuity of rotation representations in neural networks," in *IEEE Int. Conf. Comput. Vis. Pattern Recog. (CVPR)*, 2019.

[21] E. Kaufmann, L. Bauersfeld, and D. Scaramuzza, "A benchmark comparison of learned control policies for agile quadrotor flight," in *2022 International Conference on Robotics and Automation (ICRA)*, IEEE, 2022.

[22] S. Guadarrama, A. Korattikara, O. Ramirez, P. Castro, E. Holly, S. Fishman, K. Wang, E. Gonina, N. Wu, E. Kokiopoulou, L. Sbaiz, J. Smith, G. Bartók, J. Berent, C. Harris, V. Vanhoucke, and E. Brevdo, "TF-Agents: A library for reinforcement learning in tensorflow." `https://github.com/tensorflow/agents`, 2018. [Online; accessed 25-June-2019].