2024 IEEE International Conference on Robotics and Automation (ICRA)
May 13-17, 2024. Yokohama, Japan

End-to-end Reinforcement Learning for Time-Optimal Quadcopter Flight

Robin Ferede[1], Christophe De Wagter[1], Dario Izzo[2], Guido C.H.E. de Croon[1]

2024 IEEE International Conference on Robotics and Automation (ICRA) | 979-8-3503-8457-4/24/$31.00 ©2024 IEEE | DOI: 10.1109/ICRA57147.2024.10611665

***Abstract*— Aggressive time-optimal control of quadcopters poses a significant challenge in the field of robotics. The state-of-the-art approach leverages reinforcement learning (RL) to train optimal neural policies. However, a critical hurdle is the sim-to-real gap, often addressed by employing a robust inner loop controller —an abstraction that, in theory, constrains the optimality of the trained controller, necessitating margins to counter potential disturbances. In contrast, our novel approach introduces high-speed quadcopter control using end-to-end RL (E2E) that gives direct motor commands. To bridge the reality gap, we incorporate a learned residual model and an adaptive method that can compensate for modeling errors in thrust and moments. We compare our E2E approach against a state-of-the-art network that commands thrust and body rates to an INDI inner loop controller, both in simulated and real-world flight. E2E showcases a significant 1.39-second advantage in simulation and a 0.17-second edge in real-world testing, highlighting end-to-end reinforcement learning's potential. The performance drop observed from simulation to reality shows potential for further improvement, including refining strategies to address the reality gap or exploring offline reinforcement learning with real flight data.**

***Index Terms*— Time optimal control, Reinforcement Learning, end-to-end control, reality gap, sim-to-real transfer, abstraction**

I. INTRODUCTION

The demand for autonomous quadcopters capable of high-speed flight has been steadily increasing, driven by the need for covering larger distances in various applications [1]. However, achieving efficient and agile high-speed flight remains a significant challenge, requiring the development of computationally efficient time-optimal control algorithms.

Traditionally, quadcopter control relied on established methods, including Differential-Flatness-Based Controllers (DFBC) [2]–[5] and Nonlinear Model Predictive Control (NMPC) [6]–[9], [9]–[13]. While these methods represented important steps forward, they faced difficulties with unmodeled effects [14].

Recent advancements in quadcopter control have brought Reinforcement Learning (RL) to the forefront of research [15]. While simulation studies such as [16], [17] have demonstrated RL's capability to achieve high-speed flight, real-world applications such as [18]–[21] have had to devise clever solutions to address the issues posed by the reality gap. This is because RL heavily relies on accurate simulation environments, which become increasingly challenging at high quadcopter speeds due to complex dynamics. Consequently, RL methods in actual flight often adopt an abstraction approach in which the network outputs high-level control commands that are executed by manually pre-tuned low-level controllers.

For instance, in [18], an end-to-end network is trained in simulation for racing, which was then used in a real flight to generate higher level commands in the form of guidance trajectories that were tracked with an MPC controller. In [19], RL provides thrust and body rate commands, which are tracked with an inner loop controller. In a comparison study [20], it is argued that providing thrust and body rate commands strikes the optimal balance, as direct motor commands, while effective in simulation, fail to bridge the reality gap for successful real-life flight. Particularly noteworthy is the achievement in [21], where an RL controller outperformed a human racing pilot. This success was attributed to a combination of abstraction (thrust and body rate commands) and accurate modeling of the closed-loop system using a learned residual model.

However, leaning on abstractions, in theory, places a limitation on the platform's maneuverability. This is because, in the end, it is the lowest-level controller that decides which actuator to saturate. On the contrary, end-to-end approaches, which operate without such abstractions, hold promise in enabling exceptionally agile maneuvers and pushing performance to its absolute limits.

There are already some noteworthy instances of neural networks directly governing motor control. For instance, in [22], direct motor control is achieved by a neural network, focusing on low-level controls in waypoints tracking and vehicle recovery from harsh initialization. Although this work was concerned with end-to-end control, it did not focus on time optimality or high-speed flight. In more recent developments, methods using supervised learning from optimal trajectory datasets have shown remarkable success in E2E control in the context of high-speed quadcopter flight [23], [24]. The strategies used here effectively address the reality gap by using an adaptive control approach that both measures and compensates for modeling errors.

In this article, we introduce a novel method for achieving high-speed quadcopter control through end-to-end reinforcement learning, where direct motor commands are generated.

*This work was supported by ESA

[1]The authors are with the Micro Air Vehicle Lab of the Faculty of Aerospace Engineering, Delft University of Technology, 2629 HS Delft, The Netherlands R.Ferede@tudelft.nl, G.C.H.E.deCroon@tudelft.nl, C.deWagter@tudelft.nl

[2]The authors are with the Advanced Concepts Team, European Space Agency, Keplerlaan 1, 2201 AZ, Noordwijk, The Netherlands. Dario.Izzo@esa.int

Code available at: https://github.com/tudelft/optimal_quad_control_RL

Video summary availabe at: https://youtu.be/jwQ33_vJVos

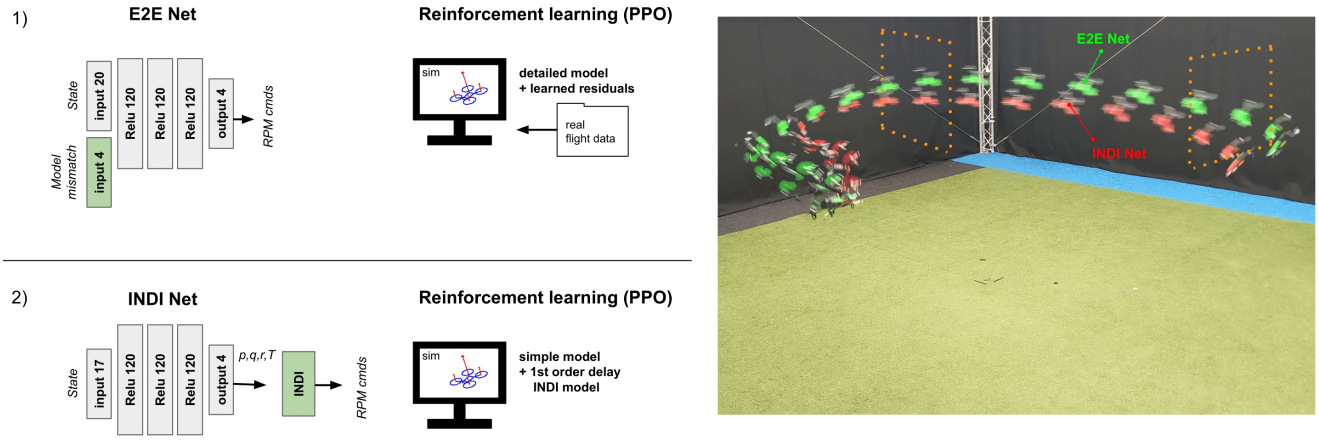

Fig. 1. Drone racing with 2 RL Control Approaches: 1) E2E Network: Computes direct motor commands, learns compensation for unmodeled moments and thrust, trained with a learned residual model. 2) INDI Network: Computes thrust and body rate commands, uses an INDI inner loop controller, trained with first order INDI model.

To effectively bridge the reality gap, our approach combines a residual moment and thrust model, acquired from flight data (akin to the approach in [21]), combined with the adaptive techniques described in [23], [24], applied to both thrust and moment model. Our study involves a comparative analysis, pitting the performance of our E2E approach against that of a state-of-the-art network generating thrust and rate commands for an inner loop INDI controller [25].

II. METHODOLOGY

A. Problem Definition

The drone racing problem is modeled as a Markov Decision Process (MDP) defined by $(\mathcal{S}, \mathcal{A}, \mathcal{P}, r, \rho_0, \gamma)$. The RL agent starts in state $s_t \in \mathcal{S}$ from distribution ρ_0. It uses stochastic policy $\pi(a_t|s_t)$ to select a continuous action $a_t \in \mathcal{A}$, transitioning to s_{t+1} with probability $\mathcal{P}^{a_t}_{s_t,s_{t+1}}$ and receiving reward r_t. The objective is to optimize the parameters θ of a stochastic neural network policy $\pi_\theta(a_t|s_t)$ to maximize expected return over an infinite horizon $\max_\theta \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{\infty} \gamma^t r_t\right]$.

The racing setup consists of four square gates (1x1m) placed in a 4x3m rectangle as seen in Fig. 3. We quantify the racing task using a reward function consisting of a progress reward that measures advancement toward the target gate, a gate reward that encourages gate passage through the center, and a collision penalty:

$$r(k) = \begin{cases} +10 - 10||\mathbf{p}_k - \mathbf{p}_{g_k}||, & \text{if gate passed} \\ -10, & \text{if collision} \\ ||\mathbf{p}_k - \mathbf{p}_{g_k}|| - ||\mathbf{p}_{k-1} - \mathbf{p}_{g_k}|| & \text{otherwise} \end{cases}$$

Here, $\mathbf{p}_{g_k}$ represents the position of the center of the current target gate, and $\mathbf{p}_k$, $\mathbf{p}_{k-1}$ are the drone's current and previous positions. A gate is deemed successfully passed when the drone intersects the gate plane inside the 1x1m boundary. A collision is registered when the drone either makes contact with the ground or passes through the gate plane outside the gate boundary. This reward function, in conjunction with a state transition model (elaborated in next section), will be employed to create a virtual environment and conduct training using the PPO algorithm [26] in the Python library Stable-Baselines3 [27]. During training, we will simulate 100 drones in parallel, using a discount factor $\gamma = 0.999$ and a maximum episode duration of 1200 time steps (12 seconds). We terminate training after 100,000,000 time steps.

B. E2E Network

1) Quadcopter model: We adopt a quadcopter model similar to the one outlined in [23], [24]. The quadcopter's state and control inputs are defined as follows:

$$\mathbf{x} = [\mathbf{p}, \mathbf{v}, \boldsymbol{\lambda}, \boldsymbol{\Omega}, \boldsymbol{\omega}, \mathbf{M}_{ext}, \mathbf{F}_{ext}]^T \quad \mathbf{u} = [u_1, u_2, u_3, u_4]^T$$

Here $\mathbf{p}$ is the position of the drone, $\mathbf{v}$ is the velocity, $\boldsymbol{\lambda}$ are the Euler angles, $\boldsymbol{\Omega}$ are the body rates and $\boldsymbol{\omega}$ are the propeller speed in RPM with a range between $\omega_{min} = 3000$ and $\omega_{max} = 11000$. $\mathbf{u}$ represents the motor RPM commands. Similar to the approach in [23], [24], we include external (specific) forces and moments as part of the state representation. These external influences will be measured onboard to mitigate modeling inaccuracies. The equations of motion are expressed as follows:

$$\begin{aligned} \dot{\mathbf{p}} &= \mathbf{v} & \dot{\mathbf{v}} &= g\mathbf{e_3} + R(\lambda)[\mathbf{F}_{mod} + \mathbf{F}_{ext}] \quad (1) \\ \dot{\lambda} &= Q(\lambda)\boldsymbol{\Omega} & I\dot{\boldsymbol{\Omega}} &= -\boldsymbol{\Omega} \times I\boldsymbol{\Omega} + \mathbf{M}_{mod} + \mathbf{M}_{ext} \\ \dot{\boldsymbol{\omega}} &= (\mathbf{u} - \boldsymbol{\omega})/\tau & \dot{\mathbf{F}}_{ext} &= 0 \quad \dot{\mathbf{M}}_{ext} = 0 \end{aligned}$$

In contrast to [23], [24], $\mathbf{M}_{mod}$ and $\mathbf{F}_{mod}$ are formulated as a hybrid model, comprising a nominal model rooted in physical principles [28] and a residual black-box model fine-tuned using empirical data: $\mathbf{F}_{mod} = \mathbf{F}_{nom} + \mathbf{F}_{res}$ and $\mathbf{M}_{mod} = \mathbf{M}_{nom} + \mathbf{M}_{res}$. The nominal model, identical to the one identified for the Bebop drone in [23], is defined as:

$$\mathbf{F}_{nom} = \begin{bmatrix} -k_x v_x^B \sum_{i=1}^4 \omega_i \\ -k_y v_y^B \sum_{i=1}^4 \omega_i \\ -k_\omega \sum_{i=1}^4 \omega_i^2 - k_z v_z^B \sum_{i=1}^4 \omega_i - k_h (v_x^{B2} + v_y^{B2}) \end{bmatrix}$$

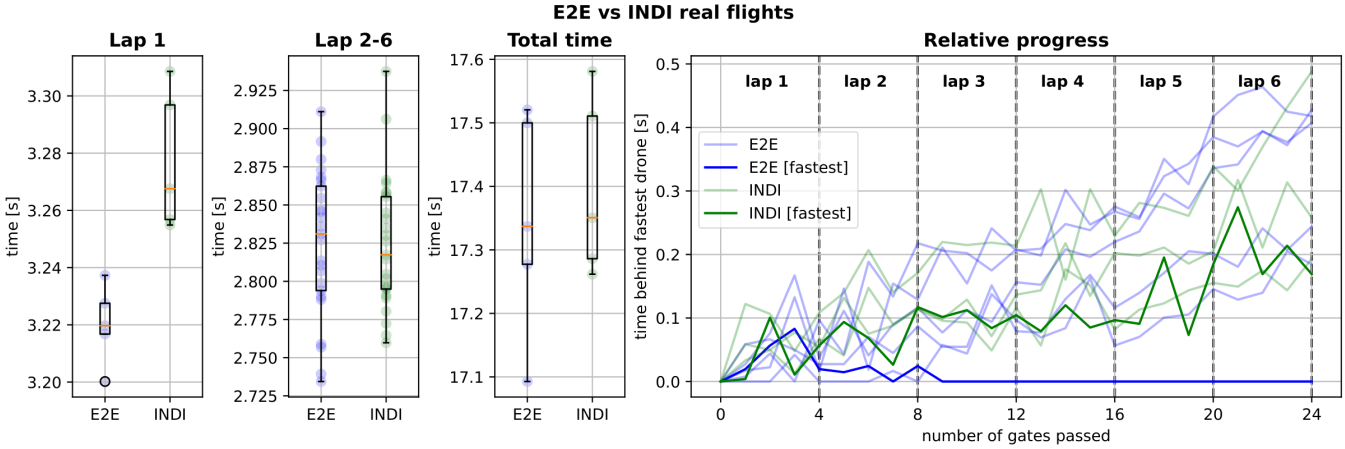

Fig. 2. Flight test comparison E2E vs. INDI Net (5 Repetitions): 1) E2E outpaces INDI in the first lap (starting in hover). 2) Laps 2-6 show comparable performance. 3) E2E has a slight advantage over INDI in total track completion time. 4) Relative progress of all flights E2E leads by 0.17 seconds.

$$\mathbf{M}_{nom} = \begin{bmatrix} k_p(\omega_1^2 - \omega_2^2 - \omega_3^2 + \omega_4^2) + k_{pv} v_y^B \\ k_q(\omega_1^2 + \omega_2^2 - \omega_3^2 - \omega_4^2) + k_{qv} v_x^B \\ k_{r1}(-\omega_1 + \omega_2 - \omega_3 + \omega_4) + \\ \dots k_{r2}(-\dot{\omega}_1 + \dot{\omega}_2 - \dot{\omega}_3 + \dot{\omega}_4) - k_{rr} r \end{bmatrix}$$

The residual model consists of two small neural networks, one for residual (mass normalized) thrust and the other for residual moment:

$$\mathbf{F}_{res} = [0, 0, -\mathcal{N}_T(\boldsymbol{\omega}, \mathbf{v}^B)]^T \quad \mathbf{M}_{res} = \mathcal{N}_M(\boldsymbol{\omega}, \mathbf{v}^B, \boldsymbol{\Omega})$$

Both $\mathcal{N}_T$ (7 inputs 1 output) and $\mathcal{N}_M$ (10 inputs 3 outputs) utilize a neural network architecture featuring a single hidden layer comprising 32 neurons with a ReLU activation function. The networks are trained through supervised learning with a dataset of high speed flight data.

2) Policy: The policy is implemented as a three-layered, fully connected neural network with a ReLU activation function, as visually depicted in Fig. 1. The network takes 24 inputs containing the quadcopter's state, as well as the modeling mismatches represented by $\mathbf{M}_{ext}$ and $\mathbf{F}_{ext}$, along with information current and future gates.

$$\mathbf{x}_{in} = [\mathbf{p}^{g_i}, \mathbf{v}^{g_i}, \boldsymbol{\lambda}^{g_i}, \boldsymbol{\Omega}, \boldsymbol{\omega}, \mathbf{M}_{ext}, \mathbf{F}_{ext}, \mathbf{p}^{g_i}_{g_{i+1}}, \psi^{g_i}_{g_{i+1}}]^T$$

Here, the superscript ^{g_i} denotes that the variables are represented in the reference frame aligned with the target gate g_i. The symbols $\mathbf{p}_{gi+1}$ and $\psi_{g_{i+1}}$ respectively indicate the position and orientation of the succeeding gate (i.e., the one following the target gate). The network has 4 outputs corresponding to rpm commands $\mathbf{u}$. Since our approach employs a stochastic policy, the network's outputs serve as the means of a normal distribution governing the rpm commands. The standard deviation of this distribution is an extra trainable parameter of the network. For the purpose of generating a control command, whether during training or testing, we draw samples from this distribution and constrain the result within the minimum and maximum rpm limits.

3) Training: To train within the RL framework, we convert our continuous-time dynamical model into a discrete-time MDP using a time step of 0.01s via the Forward Euler method. The episode is initialized by uniformly sampling initial states from the following intervals:

$$\begin{aligned} &x \in [-\tfrac{1}{2}, \tfrac{1}{2}] + x_s && y \in [-\tfrac{1}{2}, \tfrac{1}{2}] + y_s && z \in [-\tfrac{1}{2}, \tfrac{1}{2}] + z_s \\ &v_x \in [-\tfrac{1}{2}, \tfrac{1}{2}] && v_y \in [-\tfrac{1}{2}, \tfrac{1}{2}] && v_z \in [-\tfrac{1}{2}, \tfrac{1}{2}] \\ &\phi \in [-\tfrac{2\pi}{9}, \tfrac{2\pi}{9}] && \theta \in [-\tfrac{2\pi}{9}, \tfrac{2\pi}{9}] && \psi \in [-\pi, \pi] \\ &p \in [-1, 1] && q \in [-1, 1] && r \in [-1, 1] \\ &\boldsymbol{\omega} \in [\omega_{min}, \omega_{max}]^4 \end{aligned} \tag{2}$$

Here x_s, y_s, z_s is the starting point of the race track. Also $\mathbf{M}_{ext}$ and $\mathbf{F}_{ext}$ are sampled from

$$\begin{aligned} &M_{x,ext}, M_{y,ext} \in [-0.03, 0.03] && M_{z,ext} \in [-0.01, 0.01] \\ &F_{x,ext}, F_{y,ext} = 0 && F_{z,ext} \in [-0.5, 0.5] \end{aligned}$$

C. INDI Network

1) Quadcopter model: Integrating the INDI controller into the loop streamlines the quadcopter modeling process considerably. With the INDI Network providing rate and thrust commands, our RL environment now primarily focuses on simulating the INDI controller's response, specifically how actual thrust and rates relate to the commanded values. As illustrated in [25], the INDI controller's command response can be expressed as a first-order delay model. The remaining quadcopter modeling tasks only involve the kinematic equations and a basic drag model:

$$\mathbf{x} = [\mathbf{p}, \mathbf{v}, \boldsymbol{\lambda}, \boldsymbol{\Omega}, T]^T \quad \mathbf{u} = [\boldsymbol{\Omega}_{cmd}, T_{cmd}]^T$$

The equations of motion are expressed as:

$$\begin{aligned} &\dot{\mathbf{p}} = \mathbf{v} && \dot{\mathbf{v}} = g\mathbf{e_3} + R(\lambda)\mathbf{F} \\ &\dot{\lambda} = Q(\lambda)\boldsymbol{\Omega} && \dot{\boldsymbol{\Omega}} = (\boldsymbol{\Omega}_{cmd} - \boldsymbol{\Omega})/\tau_\Omega \\ &\dot{T} = (T_{cmd} - T)/\tau_T \end{aligned} \tag{3}$$

Where $\mathbf{F} = [-d_x v_x^B, -d_y v_y^B, -T]^T$ with $d_x = 0.34$, $d_y = 0.43$, and $\tau_T = \tau_\Omega = 0.03$ derived from flight data.

An inherent limitation of relying on an INDI controller is that it requires some margins to reject disturbances and track the desired command. This means that a trade-off

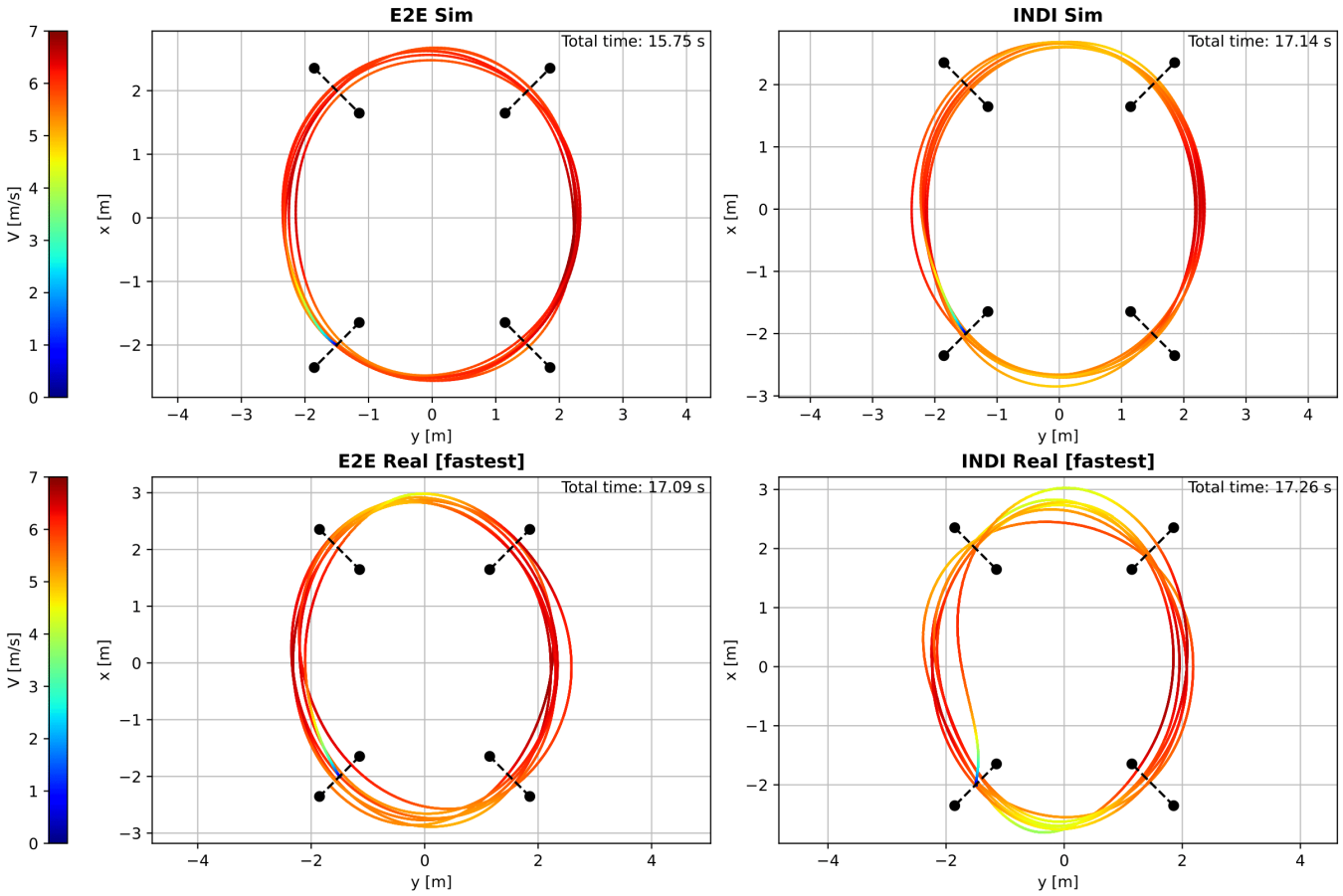

Fig. 3. Trajectory comparison E2E vs INDI: simulated runs and fastest achieved real flights. The performance gap is most significant in simulation with E2E leading by 1.39 seconds. In the real flights, this difference is reduced to 0.17 seconds.

must be made between how much margin is kept to reject disturbances (which makes the drone slower) or how little margin is acceptable before actuator limits are exceeded and the INDI can not track the reference anymore. We establish boundaries for thrust and rate based on the fastest stable flight performance we could achieve in our setup:

$$p_{cmd}, q_{cmd}, r_{cmd} \in [-3, 3] \quad T_{cmd} \in [0, 15]$$

2) Policy: The policy closely resembles the E2E Network but differs only in the number of inputs due to the system simplifications. The network takes 17 inputs, consisting of the state and gate information:

$$\mathbf{x}_{in} = [\mathbf{p}^{g_i}, \mathbf{v}^{g_i}, \boldsymbol{\lambda}^{g_i}, \boldsymbol{\Omega}, T, \mathbf{p}^{g_i}_{g_{i+1}}, \psi^{g_i}_{g_{i+1}}]^T$$

The network has 4 outputs corresponding to the thrust and body rate control command. Again the network's outputs are the means of normal distributions governing thrust and body rate commands, along with a learned standard deviation.

3) Training: In the training phase, we once more convert our dynamical model into a discrete-time MDP. We uniformly sample initial states from the same intervals as described in Eq. 2, where applicable. The thrust is sampled from the range [7.4, 7.6].

III. EXPERIMENTAL SETUP

For our experiments, we use the Parrot Bebop 1 quadcopter. While it does not have the high-speed performance of a racing drone, its unique characteristics make it a compelling choice. The Bebop 1 features a notably flexible frame, posing a more intricate challenge for modeling compared to the rigid racing counterparts. Furthermore, it provides an excellent opportunity to push the boundaries of actuation within a confined space. The onboard software of the Bebop is replaced by the Paparazzi-UAV open-source autopilot [29]. Computation occurred in real-time on a Parrot P7 dual-core CPU Cortex A9 processor. The Bebop featured an MPU6050 IMU sensor for specific force and angular velocity measurements, as well as RPM measurements for each propeller, essential for our control method. Flight tests were conducted in The CyberZoo, a 10x10x7 m flight arena at TU Delft's Aerospace Engineering faculty, equipped with an OptiTrack motion capture system for real-time position and attitude data. An extended Kalman filter is used to fuse Optitrack position and attitude measurements with the accelerometer and gyro data from the IMU. This integration allows us to estimate position, velocity, attitude, and IMU biases accurately.

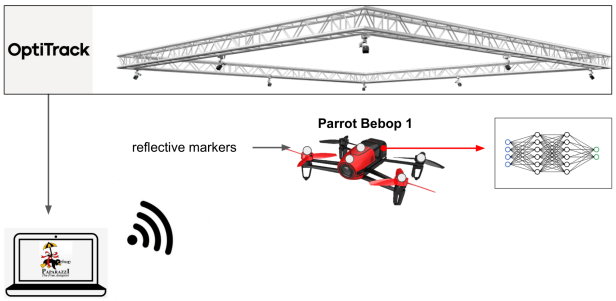

Fig. 4. Experimental setup: The drone's position and attitude are obtained from the Optitrack motion capture system and fused with IMU data by an extended Kalman filter. Image from [23]

	lap1	lap2	lap3	lap4	lap5	lap6	total
E2E Sim	2.97	2.51	2.56	2.55	2.59	2.57	15.75
INDI Sim	3.20	2.82	2.75	2.80	2.81	2.76	17.14
E2E1	3.23	2.79	2.81	2.84	2.84	2.83	17.34
E2E2	**3.22**	**2.74**	**2.79**	**2.80**	**2.76**	**2.79**	**17.09**
E2E3	3.20	2.73	2.91	2.76	2.85	2.83	17.28
E2E4	3.22	2.85	2.89	2.87	2.87	2.81	17.50
E2E5	3.24	2.81	2.85	2.86	2.87	2.88	17.52
Mean	3.22	2.79	2.85	2.82	2.84	2.83	17.35
INDI1	3.30	2.86	2.80	2.86	2.91	2.79	17.51
INDI2	**3.26**	**2.79**	**2.80**	**2.79**	**2.84**	**2.77**	**17.26**
INDI3	3.25	2.80	2.83	2.86	2.76	2.84	17.35
INDI4	3.27	2.78	2.82	2.76	2.83	2.83	17.29
INDI5	3.31	2.80	2.86	2.81	2.87	2.94	17.58
Mean	3.28	2.80	2.82	2.82	2.84	2.83	17.40

TABLE I

LAP TIMES (IN SECONDS) FASTEST FLIGHT IN BOLD.

IV. RESULTS & DISCUSSION

We conducted tests using the trained E2E and INDI networks, involving both simulation roll-outs and real flight tests covering 6 laps on the race track. The real flights were repeated 5 times for both networks. Our assessment focused on time optimality as the primary performance metric. Fig. 2 provides an overview of the real flights conducted, along with corresponding values found in Table I. Fig. 3 highlights the fastest flights achieved by both controllers compared to their simulated counterparts.

Our key observation is that E2E outperforms INDI on the 6-lap racing task, both in simulation and in real-world scenarios. The most substantial performance gap is evident in simulation, with E2E completing the track 1.39 seconds faster. In the real world, the performance difference, while reduced, remains existent, with an average improvement of 0.05 seconds over the entire track. When comparing the fastest runs, E2E is in the lead by 0.17 seconds. An intriguing insight, highlighted in Fig. 2, reveals that E2E Networks primarily leverages its advantage during the first lap when the drone starts in hover. In laps 2 through 6, both methods exhibit remarkably similar performance. This observation highlights the E2E Net's capability to efficiently perform rapid accelerations.

When comparing the results between simulated runs and real-life flights, several noteworthy observations come to light. In the simulated environment, the INDI controller achieves a track completion time of 17.14, with only a minor increase to 17.40 seconds on average in the real flight tests. In contrast, the E2E method exhibits a more pronounced reality gap, with completion times increasing from 15.75 in simulation to an average of 17.35 in real-world scenarios. This points in the direction of an interesting trade-off: INDI needs margins to be able to control the drone in case of uncertainty or disturbance, which limits its speed. E2E can access 100% of the actuators but has a bigger sim-to-real gap. Despite this substantial gap between simulation and reality, the fact that E2E maintains a performance advantage over INDI, albeit modest, underscores the success of the sim-to-real transfer. Furthermore, these findings suggest that E2E may hold untapped potential for further enhancing real-world drone performance.

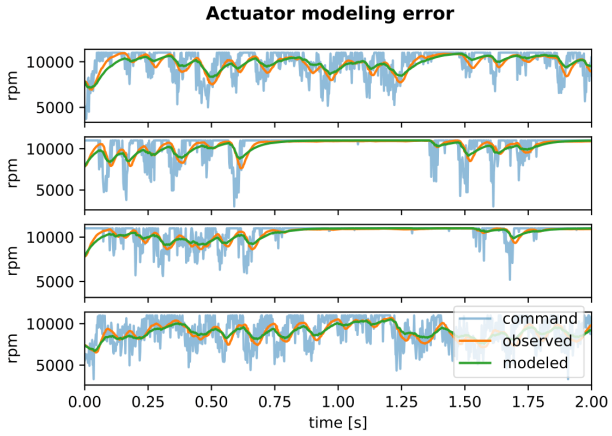

Fig. 5. Modeling errors in the first order actuator delay model used for the E2E Net

While the E2E net addresses thrust and moment modeling errors, it's important to note that certain challenges remain unaddressed. In Fig. 5, we observe deviations in motor RPM between actual and modeled values during the fastest flight, potentially explaining performance drops in the real world. The INDI Net has a smaller simulation-to-reality gap, but it also encounters modeling errors. In Fig. 6 we can see how the observed rates and thrust also deviate from the modeled first-order delay. These deviations could explain the different trajectory shapes observed in the real INDI flight in Fig. 3.

V. CONCLUSION

We have presented a novel neural network approach for achieving high-speed quadcopter flight through end-to-end reinforcement learning. We addressed the reality gap issue by using an adaptive method alongside a learned residual model. Comparing our approach to a state-of-the-art INDI network, we observed that our E2E network can outperform the state of the art both in simulation in real flights. In the

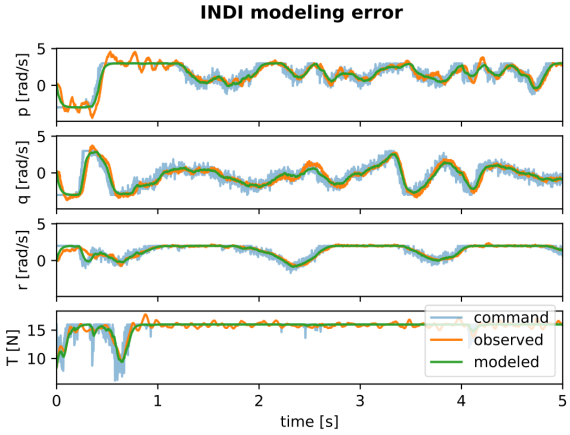

Fig. 6. Modeling errors in the first order delay model of the INDI controller used for the INDI Net.

simulated environment, E2E exhibits a notable advantage, completing tasks 1.39 seconds faster. This performance gap, while smaller in real flight scenarios, remains significant and showcases the competitiveness of E2E architectures when compared to the state of the art. The performance contrast observed between simulation and real-life scenarios implies that our E2E approach exhibits greater sensitivity to modeling errors, as opposed to the robustness of the INDI controller. Future efforts could be directed toward enhancing the E2E network's ability to adapt to modeling errors. Additionally, apart from addressing moment and thrust offsets, it's also possible to identify other discrepancies in the model, such as variations in battery voltage or maximum RPM. Alternatively, a promising avenue for future research may involve refining our network through the application of offline RL techniques using real flight data.

REFERENCES

[1] M. Hassanalian and A. Abdelkefi, "Classifications, applications, and design challenges of drones: A review," *Progress in Aerospace Sciences*, vol. 91, pp. 99–131, 2017.

[2] M. J. Van Nieuwstadt and R. M. Murray, "Real-time trajectory generation for differentially flat systems," *International Journal of Robust and Nonlinear Control: IFAC-Affiliated Journal*, vol. 8, no. 11, pp. 995–1020, 1998.

[3] 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.

[4] M. Faessler, A. Franchi, and D. Scaramuzza, "Differential flatness of quadrotor dynamics subject to rotor drag for accurate tracking of high-speed trajectories," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 620–626, 2017.

[5] E. Tal and S. Karaman, "Accurate tracking of aggressive quadrotor trajectories using incremental nonlinear dynamic inversion and differential flatness," *IEEE Transactions on Control Systems Technology*, vol. 29, no. 3, pp. 1203–1218, 2020.

[6] P. Ru and K. Subbarao, "Nonlinear model predictive control for unmanned aerial vehicles," *Aerospace*, vol. 4, no. 2, 2017. [Online]. Available: https://www.mdpi.com/2226-4310/4/2/31

[7] D. Bicego, J. Mazzetto, R. Carli, M. Farina, and A. Franchi, "Nonlinear model predictive control with enhanced actuator model for multi-rotor aerial vehicles with generic designs," *Journal of Intelligent & Robotic Systems*, vol. 100, no. 3-4, pp. 1213–1247, Sep. 2020.

[8] C. Liu, H. Lu, and W.-H. Chen, "An explicit mpc for quadrotor trajectory tracking," in *2015 34th Chinese Control Conference (CCC)*, 2015, pp. 4055–4060.

[9] G. Torrente, E. Kaufmann, P. Foehn, and D. Scaramuzza, "Data-driven MPC for quadrotors," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3769–3776, apr 2021.

[10] A. Romero, S. Sun, P. Foehn, and D. Scaramuzza, "Model predictive contouring control for near-time-optimal quadrotor flight," *CoRR*, vol. abs/2108.13205, 2021. [Online]. Available: https://arxiv.org/abs/2108.13205

[11] A. Romero, R. Penicka, and D. Scaramuzza, "Time-optimal online replanning for agile quadrotor flight," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 7730–7737, Jul. 2022.

[12] S. Sun, A. Romero, P. Foehn, E. Kaufmann, and D. Scaramuzza, "A comparative study of nonlinear mpc and differential-flatness-based control for quadrotor agile flight," *IEEE Transactions on Robotics*, 2022.

[13] D. Hanover, P. Foehn, S. Sun, E. Kaufmann, and D. Scaramuzza, "Performance, precision, and payloads: Adaptive nonlinear mpc for quadrotors," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 690–697, 2022.

[14] 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. [Online]. Available: https://www.science.org/doi/abs/10.1126/scirobotics.adg1462

[15] D. Hanover, A. Loquercio, L. Bauersfeld, A. Romero, R. Penicka, Y. Song, G. Cioffi, E. Kaufmann, and D. Scaramuzza, "Past, present, and future of autonomous drone racing: A survey," *arXiv preprint arXiv:2301.01755*, 2023.

[16] K. Nagami and M. Schwager, "Hjb-rl: Initializing reinforcement learning with optimal control policies applied to autonomous drone racing." in *Robotics: science and systems*, 2021.

[17] U. Ates, "Long-term planning with deep reinforcement learning on autonomous drones," in *2020 Innovations in Intelligent Systems and Applications Conference (ASYU)*. IEEE, 2020, pp. 1–6.

[18] 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.

[19] R. Penicka, Y. Song, E. Kaufmann, and D. Scaramuzza, "Learning minimum-time flight in cluttered environments," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 7209–7216, 2022.

[20] 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, pp. 10 504–10 510.

[21] 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, Aug 2023. [Online]. Available: https://doi.org/10.1038/s41586-023-06419-4

[22] 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.

[23] R. Ferede, G. C. H. E. de Croon, C. De Wagter, and D. Izzo, "End-to-end neural network based quadcopter control," 2023. [Online]. Available: https://arxiv.org/abs/2304.13460

[24] S. Origer, C. De Wagter, R. Ferede, G. C. H. E. de Croon, and D. Izzo, "Guidance & control networks for time-optimal quadcopter flight," 2023. [Online]. Available: https://arxiv.org/abs/2305.02705

[25] E. J. J. Smeur, Q. Chu, and G. C. H. E. de Croon, "Adaptive incremental nonlinear dynamic inversion for attitude control of micro air vehicles," *Journal of Guidance, Control, and Dynamics*, vol. 39, no. 3, pp. 450–461, mar 2016.

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

[27] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, "Stable-baselines3: Reliable reinforcement learning implementations," *Journal of Machine Learning Research*, vol. 22, no. 268, pp. 1–8, 2021. [Online]. Available: http://jmlr.org/papers/v22/20-1364.html

[28] M. Bangura, M. Melega, R. Naldi, and R. Mahony, "Aerodynamics of rotor blades for quadrotors," 2016.

[29] B. Gati, "Open source autopilot for academic research-the paparazzi system," in *American Control Conference (ACC), 2013*. Washington, DC: IEEE, Jun. 2013, pp. 1478–1481.