Delft University of Technology

End-to-end neural network based optimal quadcopter control

Ferede, Robin; de Croon, Guido; De Wagter, Christophe; Izzo, Dario

Important note
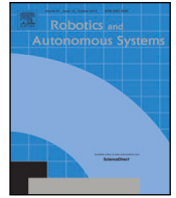To cite this publication, please use the final published version (if applicable).
Please check the document version above.

End-to-end neural network based optimal quadcopter control ☆

Robin Ferede [a],[*], Guido de Croon [a], Christophe De Wagter [a], Dario Izzo [b]

[a] *Micro Air Vehicle lab, Control and Simulation, Delft University of Technology, Kluyverweg 1, Delft, 2629 HS, The Netherlands*
[b] *Advanced Concepts Team, European Space Agency, Keplerlaan 1, Noordwijk, 2201 AZ, The Netherlands*

ARTICLE INFO

ABSTRACT

Developing optimal controllers for aggressive high-speed quadcopter flight poses significant challenges in robotics. Recent trends in the field involve utilizing neural network controllers trained through supervised or reinforcement learning. However, the sim-to-real transfer introduces a reality gap, requiring the use of robust inner loop controllers during real flights, which limits the network's control authority and flight performance. In this paper, we investigate for the first time, an end-to-end neural network controller, addressing the reality gap issue without being restricted by an inner-loop controller. The networks, referred to as G&CNets, are trained to learn an energy-optimal policy mapping the quadcopter's state to rpm commands using an optimal trajectory dataset. In hover-to-hover flights, we identified the unmodeled moments as a significant contributor to the reality gap. To mitigate this, we propose an adaptive control strategy that works by learning from optimal trajectories of a system affected by constant external pitch, roll and yaw moments. In real test flights, this model mismatch is estimated onboard and fed to the network to obtain the optimal rpm command. We demonstrate the effectiveness of our method by performing energy-optimal hover-to-hover flights with and without moment feedback. Finally, we compare the adaptive controller to a state-of-the-art differential-flatness-based controller in a consecutive waypoint flight and demonstrate the advantages of our method in terms of energy optimality and robustness.

1. Introduction

Nowadays there is an increasing demand for autonomous quadcopters for various military and civilian applications [1]. For many applications such as emergency response, inspection, delivery or racing the drone must fly as fast, and as energy efficient as possible [2]. However, developing autonomous systems for aggressive high-speed flight still poses many challenges. One of these challenges is developing computationally efficient optimal control algorithms that take into account non-linear dynamics and actuator limits.

Current state-of-the-art research on optimal quadcopter control focuses on making controllers track a reference guidance trajectory. Popular tracking methods include the differential-flatness-based controller (DFBC) [3–6] and the traditional nonlinear-model-predictive controller (NMPC) [7–12]. While the DFBC is more computationally efficient, traditional NMPC has gained a lot of popularity in quadcopter control due to advances in hardware. The advantages of NMPC over DFBC are improved tracking accuracy for dynamically infeasible trajectories as well as improved robustness to model mismatch [13] (especially by means of adaptive algorithms [10,14]). Furthermore,

in recent work, a traditional NMPC method was shown to outperform human pilots in a drone-racing task by tracking offline-generated time-optimal trajectories [15].

Both NMPC and DFBC suffer from a fundamental limitation: they dissect the control problem into multiple layers of abstraction. These layers typically encompass the following components:

1. The reference trajectory, which is designed to solve the primary optimization objective. This process is often computationally intensive, requiring either offline calculations or online sub-optimal simplifications, such as polynomial guidance [4,6,16], point mass trajectories [12,17], or numerical approximation methods [18–20]
2. A controller for trajectory tracking, which serves a secondary role in minimizing trajectory tracking errors
3. A low-level rate controller, responsible for determining motor commands. It carries a third objective of minimizing and prioritizing thrust and rate tracking errors.

When applied in a real-world context, these layers of abstraction inherently introduce sub-optimal behaviors. For instance, when the drone
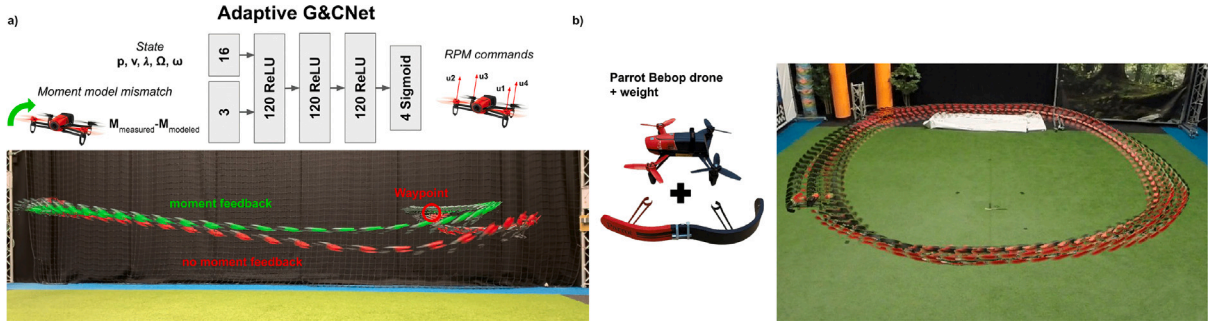

Fig. 1. (a) The reality gap issue is resolved by estimating the moment model mismatch and feeding it to the adaptive G&CNet. b) We perturb the dynamics by adding a weight on one side, the Bebop drone successfully flies through the 3×4 m track by adapting its rpm command based on the observed state and moment model mismatch.

deviates from the optimal planned trajectory due to external disturbances, there may exist an alternative trajectory that becomes more optimal.

A recent trend in quadcopter control research is the application of machine learning techniques to guidance and control problems. Deep neural networks have been trained for trajectory generation using reinforcement learning [21] and supervised machine learning [22]. Similarly, trajectory tracking has been improved by training neural networks either from flight data [23,24] or from simulation data [25,26]. Other studies have merged guidance and control within a single neural network, efficiently removing one abstraction layer through reinforcement learning [27,28] or supervised learning [29–31]. An especially remarkable case can be seen in a recent Nature paper [27], where a deep neural network trained with reinforcement learning outperforms human drone racing champions by directly issuing thrust and rate commands. In the supervised learning methods [29–31] (also employed in space applications [32,33]), networks known as G&CNets are trained to replicate optimal state feedback from a dataset of optimal trajectories. Once trained, the G&CNets provide a computationally efficient means to compute optimal control onboard the quadcopter, eliminating the need for trajectory (re)planning. Real flight tests have successfully demonstrated this approach for longitudinal trajectories, employing a simplified 2-dimensional quadcopter model [29]. In these experiments, the G&CNets calculated thrust and pitch acceleration commands, which were tracked by an INDI controller [34].

In this article, we take the G&CNet approach a step further and investigate for the first time an end-to-end, i.e., state-to-rpm network for a high dimensional quadcopter model taking into account drag, aerodynamic effects and actuator delays. Unlike any of the previous work our network directly calculates the rpm motor commands allowing us to take advantage of the actuator's limits without being limited by any abstraction layers. In our experiments, the optimization objective is achieving energy optimality, an objective closely linked to time optimality, resulting in generally rapid and smooth trajectories. The biggest obstacle with our approach is the reality gap between the model and the real world. In this research, we identify the reality gap and propose an adaptive method to mitigate the effects of unmodeled roll, pitch and yaw moments. Furthermore, we benchmark our controller's performance against a state-of-the-art differential-flatness-based controller using an identical setup with the same hardware. We selected DFBC as our benchmark over NMPC due to its similar onboard computational demands in comparison to our approach. Through this comparison, we highlight the benefits of our method in relation to energy optimality and robustness.

2. Methodology

2.1. Quadcopter model

Referring to the quadcopter configuration and axes definition illustrated in Fig. 2, the state and control input of the quadcopter can be
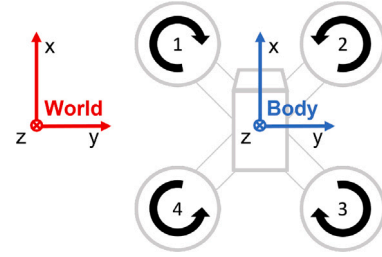
described as follows:

Fig. 2. Quadcopter configuration and axes definition (z-axis points downwards).

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

Where $\mathbf{p} = [x, y, z]$ and $\mathbf{v} = [v_x, v_y, v_z]$ are the position and velocity in the world frame, $\boldsymbol{\Omega} = [p, q, r]$ is the angular velocity in body frame, $\boldsymbol{\lambda} = [\phi, \theta, \psi]$ are the Euler angles that describe the orientation of the body frame and $\boldsymbol{\omega} = [\omega_1, \omega_2, \omega_3, \omega_4]$ are the angular velocities of each of the propellers in rpm. The control input $\mathbf{u}$ contains the normalized rpm commands $u_i \in [0, 1]$. The system dynamics are described by:

$$\dot{\mathbf{p}} = \mathbf{v} \qquad\qquad \dot{\mathbf{v}} = \mathbf{g} + R(\lambda)\mathbf{F} \qquad\qquad (1)$$

$$\dot{\lambda} = Q(\lambda)\boldsymbol{\Omega} \qquad\qquad I\dot{\boldsymbol{\Omega}} = -\boldsymbol{\Omega} \times I\boldsymbol{\Omega} + \mathbf{M}$$

$$\dot{\boldsymbol{\omega}} = ((\omega_{max} - \omega_{min})\mathbf{u} + \omega_{min} - \boldsymbol{\omega})/\tau$$

Where $\mathbf{g} = [0, 0, g]^T$ is the gravitational acceleration, I is the moment of inertia matrix given by $\text{diag}(I_x, I_y, I_z)$, ω_{min} and ω_{max} are the minimum and maximum propeller rpm limits and τ is the first order delay parameter of the actuator model. Furthermore, $R(\lambda)$ is the rotation matrix defined by:

$$R(\lambda) = \begin{bmatrix} c_\theta c_\psi & -c_\phi s_\psi + s_\phi s_\theta c_\psi & s_\phi s_\psi + c_\phi s_\theta c_\psi \\ c_\theta s_\psi & c_\phi c_\psi + s_\phi s_\theta s_\psi & -s_\phi c_\psi + c_\phi s_\theta s_\psi \\ -s_\theta & s_\phi c_\theta & c_\phi c_\theta \end{bmatrix}$$

and $Q(\lambda)$ denotes a transformation between angular velocities and Euler angles. $\mathbf{F} = [F_x, F_y, F_z]^T$ is the specific force acting on the quadcopter in the body frame which we model as a function of the body velocities and the propeller RPMs using a thrust and drag model based on [35]:

$$F_x = -k_x v_x^B \sum_{i=1}^{4} \omega_i \qquad F_y = -k_y v_y^B \sum_{i=1}^{4} \omega_i$$

$$F_z = -k_\omega \sum_{i=1}^{4} \omega_i^2 - k_z v_z^B \sum_{i=1}^{4} \omega_i - k_h(v_x^{B2} + v_y^{B2}) \qquad (2)$$

Table 1

Model parameters for the Parrot Bebop quadcopter. The moments of inertia I_x, I_y, I_z are obtained from [36]. All other parameters have been identified by means of linear regression with sensor data obtained from various flights.

k_x [rpm^{-1} s^{-1}]	k_y [rpm^{-1} s^{-1}]	k_ω [rpm^{-2} m s^{-2}]	k_z [rpm^{-1} s^{-1}]	k_h [m^{-1}]	I_x [kg m^2]	I_y [kg m^2]	I_z [kg m^2]
1.08e−05	9.65e−06	4.36e−08	2.79e−05	6.26e−02	0.000906	0.001242	0.002054

k_p [rpm^{-2} N m]	k_{pv} [N s]	k_q [rpm^{-2} N m]	k_{qv} [N s]	k_{r1} [rpm^{-1} N m]	k_{r2} [rpm^{-1} N m s]	k_{rr} [N m s]	τ [s]
1.41e−09	−7.97e−03	1.22e−09	1.29e−02	2.57e−06	4.11e−07	8.13e−04	0.06

Table 2

Validation loss for a variety of architectures after training on the hover-to-hover dataset for 10 epochs.

	60 neurons per layer	120 neurons per layer	180 neurons per layer
1 layer	0.00180	0.00113	0.00099
2 layers	0.00060	0.00025	0.00019
3 layers	0.00038	**0.00015**	0.00010
4 layers	0.00028	0.00014	0.00011

Similarly, $\mathbf{M} = [M_x, M_y, M_z]^T$ is the moment acting on the quadcopter which we model with the following equations:

$$\begin{aligned}
M_x &= k_p(\omega_1^2 - \omega_2^2 - \omega_3^2 + \omega_4^2) + k_{pv}v_y^B \\
M_y &= k_q(\omega_1^2 + \omega_2^2 - \omega_3^2 - \omega_4^2) + k_{qv}v_x^B \\
M_z &= k_{r1}(-\omega_1 + \omega_2 - \omega_3 + \omega_4) \\
&\quad + k_{r2}(-\dot\omega_1 + \dot\omega_2 - \dot\omega_3 + \dot\omega_4) - k_{rr}r
\end{aligned} \tag{3}$$

See Table 1 for the parameter values identified for our platform.

2.2. Energy optimal control problem

Given a state space X and set of admissible controls U, the goal is to find a control trajectory $\mathbf{u} : [0, T] \to U$ that steers the system from an initial state $\mathbf{x}_0$ to some target state $S \subset X$ in time T while minimizing some cost function. The energy optimal control problem considered in this paper is formulated as

$$\underset{\mathbf{u},T}{\text{minimize}} \quad E(\mathbf{u}, T) = \int_0^T \|\mathbf{u}(t)\|^2 dt \tag{4}$$
$$\text{subject to} \quad \dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}) \quad \mathbf{x}(0) = \mathbf{x}_0 \quad \mathbf{x}(T) \in S$$

It is worth highlighting that this objective is closely connected to the time-optimization criterion. As the motor command vector $\mathbf{u}$ is normalized within $[0, 1]$, it is anticipated to exhibit oscillations around the hover thrust, approximately at 0.5. Consequently, this behavior implies that the integral will be roughly proportional to the final time T.

Similar to [29] the control problem is transformed into a Nonlinear Programming (NLP) problem using Hermite Simpson transcription. The trajectories $\mathbf{x}(t), \mathbf{u}(t)$ are discretized into $N + 1$ points with a time step $\Delta t = T/N$ such that $\mathbf{x}_k = \mathbf{x}(k\Delta t)$ and $\mathbf{u}_k = \mathbf{u}(k\Delta t)$ Using the AMPL [37] modeling language with the SNOPT NLP solver [38], the optimal (discretized) trajectory $\mathbf{x}_0^* \dots \mathbf{x}_N^*$ and $\mathbf{u}_0^* \dots \mathbf{u}_N^*$ can be computed. To generate extensive datasets for these trajectories, we harness the power of parallel processing by executing the SNOPT algorithm on a server equipped with 256 CPUs, where the solver typically takes approximately 8 s to run on a single CPU.

2.3. Dataset generation and network training

A dataset is created by generating optimal trajectories for a range of initial conditions. From these trajectories, a dataset of state–action pairs can be obtained of the form $(\mathbf{x}_i^*, \mathbf{u}_i^*)$ $i = 0, \dots, N$. We use these state–action pairs to train a Neural Network $f_N : X \to U$ to approximate

the optimal feedback[1] that maps $\mathbf{x}_i^*$ to $\mathbf{u}_i^*$. In all our experiments we use a neural network with 3 hidden layers of 120 neurons with ReLU activation and an output layer of 4 neurons with Sigmoid activation (Fig. 1). This architecture was chosen because it achieved a sufficiently low loss with a modest number of weights. Increasing the number of neurons beyond this point did not significantly reduce the loss. For a comparative assessment of validation loss across various architectures, please refer to Table 2. Similar to [29] we use the mean squared error loss function:

$$l = \|f_N(\mathbf{x}_i^*) - \mathbf{u}_i^*\|^2$$

with mini-batch size 256 and a starting learning rate of 1e−3.

2.4. Adaptive method

We modify our model by assuming the existence of some constant external moment $\mathbf{M}_{ext} = [M_{ext,x}, M_{ext,y}, M_{ext,z}]^T$ acting on the system. The external moment can thus be considered part of our state vector $\mathbf{x} = [\mathbf{p}, \mathbf{v}, \lambda, \Omega, \omega, \mathbf{M}_{ext}]^T$ The modified system dynamics becomes:

$$\begin{aligned}
\dot{\mathbf{p}} &= \mathbf{v} & \dot{\mathbf{v}} &= \mathbf{g} + R(\lambda)\mathbf{F} \\
\dot\lambda &= Q(\lambda)\Omega & I\dot\Omega &= -\Omega \times I\Omega + \mathbf{M} + \mathbf{M}_{ext} \\
\dot{\mathbf{M}}_{ext} &= 0 & \dot\omega &= ((\omega_{max} - \omega_{min})\mathbf{u} + \omega_{min} - \omega)/\tau
\end{aligned} \tag{5}$$

Using the same approach as before, we can now generate optimal trajectories for this system and train a network to approximate the optimal state feedback. Additionally, the neural network will now have 3 extra inputs for $M_{ext,x}, M_{ext,y}, M_{ext,z}$. The obtained controller will now use these extra inputs to optimally compensate for the unmodeled moments (assuming they are constant). For the onboard implementation, we will obtain the values of $\mathbf{M}_{ext}$ by subtracting the modeled moment (Eq. (3)) from the measured moment

$$M_{measured} = I\dot\Omega + \Omega \times I\Omega \tag{6}$$

using filtered (8 Hz 2nd order Butterworth low-pass filter) gyroscope measurements. It is important to note that the filtering causes our estimates for $\mathbf{M}_{ext}$ to be slightly delayed. Furthermore, the controller's output is based on the assumption of a constant external moment so we can expect our method to only be effective if the modeling errors are in a sufficiently low-frequency range.

2.5. Differential-flatness-based controller (DFBC)

DFBC is a state-of-the-art method for generating aggressive trajectories using piece-wise high-order polynomials $\mathbf{p}(t) = [x(t), y(t), z(t), \psi(t)]^T$ that pass through a set of waypoints while minimizing the 'Snap' defined by the following integral [4]:

$$\int_0^T \mu_r \left[x^{(4)}(t) + y^{(4)}(t) + z^{(4)}(t) \right]^2 + \mu_\psi \left[\psi^{(2)}(t)^2 \right] dt$$

In this problem, the final time is fixed, and the polynomial coefficients are found by solving a quadratic constraint optimization problem. As

---

[1] From [30]: "the Hamilton–Jacobi-Bellman equations are important here as they imply the existence and uniqueness of an optimal state-feedback $\mathbf{u}^*(\mathbf{x})$ which, in turn, allow to consider universal function approximators such as deep neural networks to represent it".
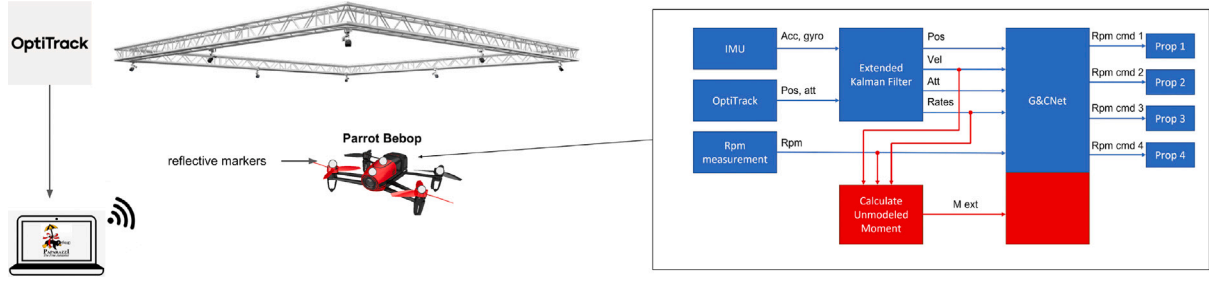
Fig. 3. Experimental setup: The Parrot Bebop's position and attitude are tracked with OptiTrack and sent via WiFi, while an onboard extended Kalman filter fuses the OptiTrack and IMU data to get an accurate state estimate for the G&CNet.

shown in [4], if we change the final time by a factor of α, the new minimum snap solution is simply a time-scaled version of the original polynomial $\mathbf{p}(\alpha t)$. By changing the value of α, the trajectory can be faster or slower without having to recompute the optimal solution. In order to achieve accurate tracking, we use an outer-loop INDI controller where the velocity and acceleration feed-forward commands are directly computed from the polynomials.

3. Experimental setup

The quadcopter used in our experiment is the Parrot Bebop 1 which has its onboard software replaced by the Paparazzi-UAV open-source autopilot project [39]. All computations will run in real time on the Parrot P7 dual-core CPU Cortex A9 processor. The Parrot Bebop has an MPU6050 IMU sensor that will be used to obtain measurements of the specific force and angular velocity along the body axes. Additionally, the Bebop can measure the angular velocities (in rpm) of each of the propellers, which is a requirement for our control method.

All flight tests are performed in The CyberZoo which is a research and test laboratory in the faculty of Aerospace Engineering at the TU Delft. This lab consists of a 10 by 10 meter area surrounded by nets with an OptiTrack motion capture system that can provide position and attitude data in real-time. An extended Kalman filter is used to fuse the OptiTrack and IMU data to obtain an estimate of the position, velocity, attitude and body rates. These estimated state variables, along with RPM measurements obtained from the ESC, serve as inputs for the G&CNet, which operates at a 500 Hz frequency. The outputs of the network will be directly used as rpm commands to the motors. The DFBC method will use the same state estimates to obtain the feedforward terms for the INDI controller. See Fig. 3 for an overview of the experimental setup.

4. Results

4.1. Identifying the reality gap

4.1.1. Nominal G&CNet

Using the system dynamics from Eq. (1) we generate a dataset of 100,000 energy-optimal trajectories with a target hover state defined by $\mathbf{x}, \mathbf{v}, \lambda, \boldsymbol{\Omega}, \dot{\mathbf{v}}, \dot{\boldsymbol{\Omega}}, \dot{\boldsymbol{\omega}} = 0$. The rpm limits are set to $\omega_{min} = 5000$, $\omega_{max} = 10000$ and the initial conditions are uniformly sampled from the following intervals:

$x \in [-5, 5]$	$y \in [-5, 5]$	$z \in [-1, 1]$
$v_x \in [-\frac{1}{2}, \frac{1}{2}]$	$v_y \in [-\frac{1}{2}, \frac{1}{2}]$	$v_z \in [-\frac{1}{2}, \frac{1}{2}]$
$\phi \in [-\frac{2\pi}{9}, \frac{2\pi}{9}]$	$\theta \in [-\frac{2\pi}{9}, \frac{2\pi}{9}]$	$\psi \in [-\pi, \pi]$
$p \in [-1, 1]$	$q \in [-1, 1]$	$r \in [-1, 1]$

$\boldsymbol{\omega} \in [\omega_{min}, \omega_{max}]^4$

We split this dataset into a training set of 90,000 trajectories and a test set of 10,000 trajectories. The G&CNet is trained until a mean squared error of ~0.0003 is obtained on the test set.

4.1.2. Simulation and flight test

With the trained nominal G&CNet, we simulate the closed loop system dynamics and do a flight test where the drone flies from hover to hover in a 3×4 m rectangle. In order to fly to the target waypoints, we subtract the waypoint coordinates from the x, y and z neural network inputs. Both in simulation and the flight test, the drone flies 10 laps in which the target waypoint is switched every 4 s. In Fig. 4 a top-down view of the trajectory can be seen for the simulation and the flight test. As expected, in the simulation, the trajectories show significant overlap and the drone consistently arrives at the waypoint without overshooting. In the flight test, the trajectories are more spread out and a large deviation can be seen in the positive x-direction. The unmodeled effects are especially visible in the forward translation maneuver where the drone speeds up too much and overshoots the next waypoint. In Fig. 5 these forward trajectories are shown from a sideways view. It can be seen that the drone loses too much altitude causing it to speed up and overshoot.

4.1.3. Unmodeled effects

We investigate the unmodeled aerodynamic effects from the forward translation flight by comparing the measured and modeled moments and specific forces. The measured moments and forces are obtained by using the filtered (16 Hz 2nd order Butterworth non-causal filter) gyroscope and accelerometer measurements. Fig. 6 shows these measured and modeled quantities for one of the forward translation trajectories of the nominal G&CNets from Fig. 5.

It can be observed that the pitch moment seems to have a significant low-frequency model mismatch. The unmodeled pitch moment is mostly negative which might explain why the drone is diving down so much in the flight test. Because our current parametric model cannot capture this effect, we choose to go for an adaptive control strategy.

4.1.4. Adaptive G&CNet

We use the modified system dynamics with external moments from Eq. (5) to generate another 100,000 energy-optimal trajectories with the same target state and initial conditions as before, only now we also uniformly sample the external moments from the following intervals:

$$M_{x,ext}, M_{y,ext} \in [-0.04, 0.04] \quad M_{z,ext} \in [-0.01, 0.01]$$

With the generated dataset we train the adaptive G&CNet with 3 extra M_{ext} inputs to learn the optimal state feedback for the modified system. Again, we train until a mean squared error of ~0.0003 is achieved.

With the adaptive G&CNet, we perform the same flight test using the 4 waypoints and compare the results to the nominal network. In Figs. 4 and 5 the trajectory is compared to the previous nominal network and the simulation. It can be seen that the trajectory no longer deviates towards the positive x-direction and the overshoot in the forward translation maneuver is significantly reduced. Furthermore the box-plot in Fig. 7 shows the arrival time T and energy $E(T) = \int_0^T \|\mathbf{u}(t)\|^2 dt$ corresponding to the trajectories from Fig. 5. As one might expect, the performance gain of the adaptive network is most significant in terms of Energy. However, the arrival time and energy in the flight
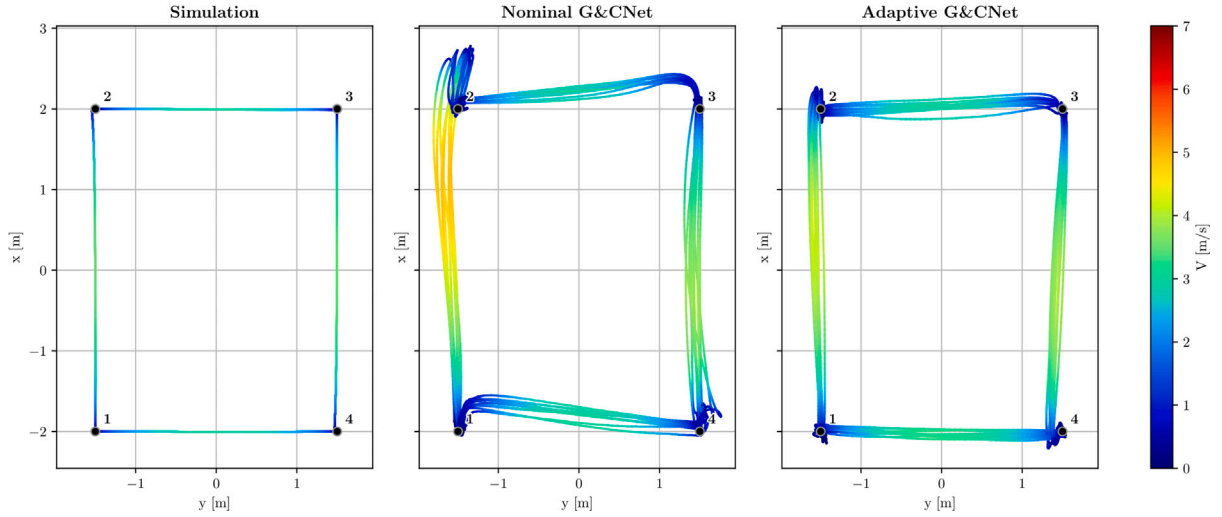
Fig. 4. Top-down view of the simulated trajectory next to the Nominal- and Adaptive G&CNet flight test.
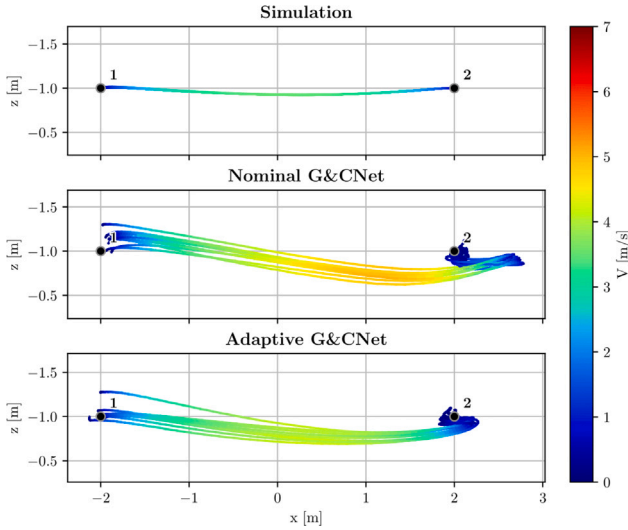

Fig. 5. Sideways view of the trajectories between waypoints 1 and 2: Simulation next to the Nominal- and Adaptive G&CNet flight test.
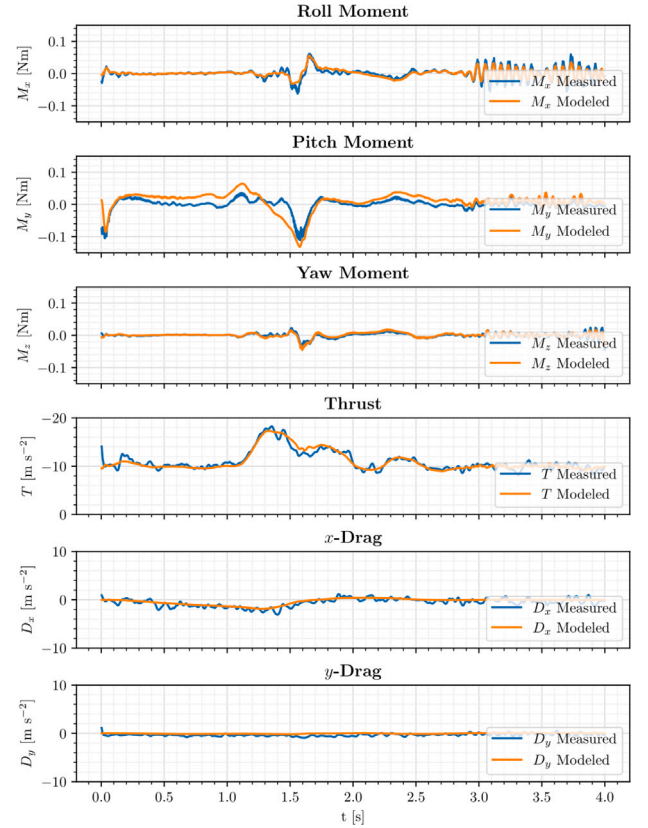

Fig. 6. Comparison of the measured and modeled moments and (specific) forces encountered in one of 'Nominal G&CNet' flights from Fig. 5.

tests are still significantly higher than in simulation which is probably due to the remaining unmodeled effects causing the overshoot at the 2nd waypoint

4.2. Bench-marking: Adaptive G&CNet vs. DFBC

4.2.1. Adaptive G&CNet

For the task of flying through consecutive waypoints, we will train an adaptive G&CNet to reach the waypoint with a forward final velocity in the direction of a 45° yaw angle. Using the modified system dynamics from Eq. (5) we generate a dataset of 10,000 energy-optimal trajectories with a target state given by:

$$x, y, z, v_z, p, q, r, \dot{p}, \dot{q}, \dot{r} = 0, \frac{v_y}{v_x} = \tan(\frac{\pi}{4}), \psi = \frac{\pi}{4}$$

The rpm limits are set to $\omega_{min} = 3000$, $\omega_{max} = 12000$ and the initial conditions are uniformly sampled from the following intervals:

$$x \in [-5, -2] \qquad y \in [-1, 1] \qquad z \in [-\tfrac{1}{2}, \tfrac{1}{2}]$$

$$v_x \in [-\tfrac{1}{2}, 5] \qquad v_y \in [-3, 3] \qquad v_z \in [-1, 1]$$

$$\phi \in [-\tfrac{2\pi}{9}, \tfrac{2\pi}{9}] \qquad \theta \in [-\tfrac{2\pi}{9}, \tfrac{2\pi}{9}] \qquad \psi \in [-\tfrac{\pi}{3}, \tfrac{\pi}{3}]$$
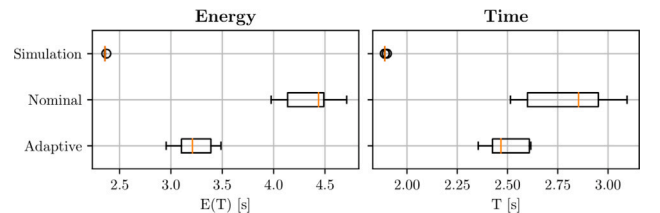

Fig. 7. Energy and time comparison during the 4 m forward flight between waypoints 1 and 2: Simulation compared to Nominal and Adaptive G&CNet.
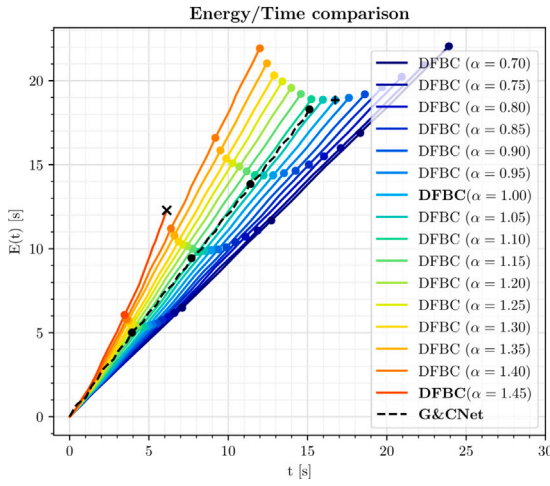
Fig. 8. Energy plotted over time during 4 laps of the 3×4 m track. The points in time where a lap is completed are represented by a dot. The DFBC method that uses the least energy is marked with a "+". The flight that crashes is marked with an "×".

$$p \in [-1, 1] \qquad q \in [-1, 1] \qquad r \in [-1, 1]$$
$$\boldsymbol{\omega} \in [\omega_{min}, \omega_{max}]^4$$

We split this dataset into a training set of 9000 trajectories and a test set of 1000 trajectories and train until a mean squared error of $\sim$0.0003 is obtained on the test set. With the trained adaptive G&CNet we perform a flight test where we fly through 4 waypoints in a 3×4m rectangle (See Figs. 9 and 11). The controller switches to the next target waypoint and changes the coordinate system once the drone is within 1.2 m from the current target. When switching to the next waypoint, we rotate our coordinate system by 90° (around the z-axis) and set the next waypoint as the origin.

4.2.2. DFBC

We generate a piece-wise 6th order polynomial $\mathbf{p}(t) = [x(t), y(t), z(t), \psi(t)]^T$ that passes through 10 laps of the 4×3 m track with a final time of 40 s. To make sure the trajectory starts in hover, the initial velocity, acceleration and yaw of the trajectory are set to 0. At the 2nd waypoint, we constrain the yaw angle to be 45° which we increment by 90° for each of the following waypoints. Additionally, at these waypoints, we constrain the velocity to be aligned with the yaw direction. Using the time scaling values starting at $\alpha = 0.7$ we generate faster and faster trajectories by incrementing α by 0.05. We then track these trajectories with the INDI controller for 4 laps. We increased alpha until the INDI controller could no longer track the trajectory (resulting in a crash). An overview of all the performed flights can be found in Figs. 14 and 15 in the appendix.

4.2.3. Energy/time comparison

We now compare the lap times and the energy integral obtained from the flight tests. In Fig. 8 the energy integral $E(t) = \int_0^t \|\mathbf{u}(\tau)\|^2 d\tau$ is plotted over time for the adaptive G&CNet flight and all DFBC flights. It can be noted that the fastest DFBC method finishes the 4 laps significantly faster than the G&CNet. In terms of energy, however, the adaptive G&CNet outperforms all of the DFBC methods. The DFBC method that uses the least energy ($\alpha = 1.0$) still uses more energy and time to finish the track. In Fig. 9, a top-down view of the trajectory of the 'energy optimal' DFBC method is plotted next to the adaptive G&CNets flight. It can be seen that the DFBC method travels in a smooth circular trajectory at a relatively high velocity, while the G&CNet takes tighter corners and flies at a lower velocity while still finishing the 4 laps quicker.
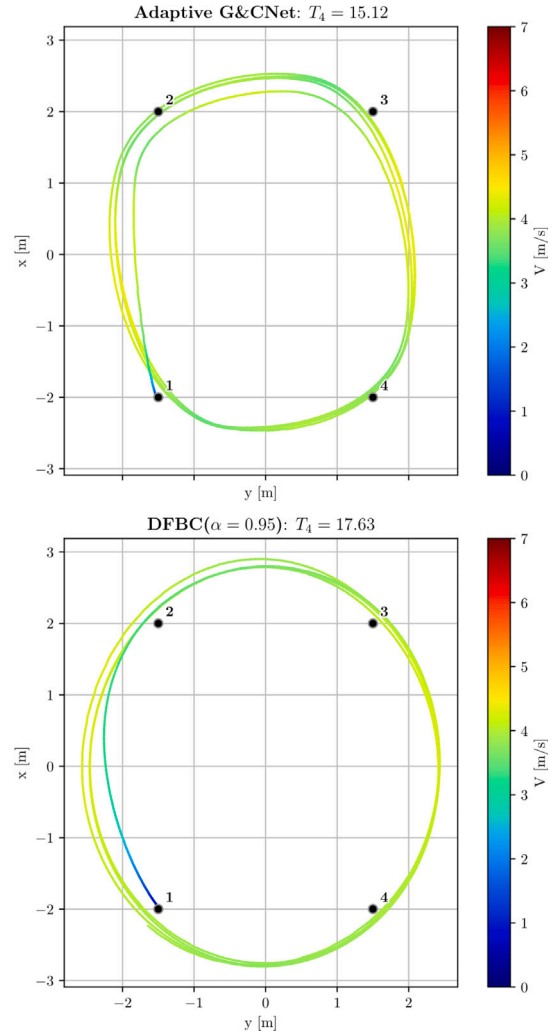

Fig. 9. Top-down view of the adaptive G&CNet's flight and the 'energy optimal' DFBC's flight at $\alpha = 1.0$.

4.2.4. Robustness experiment

In order to compare robustness, we apply an external moment to the drone by adding a bumper with a weight on the left side of the Bebop (Fig. 1). With this alteration, we perform the same flight tests as before. In Fig. 10 we again show the energy/time plot for all of the performed flights. It can be seen that the DFBC controller fails a lot earlier at $\alpha = 1.05$. In terms of time, the adaptive G&CNet demonstrated superior performance, with the quadcopter flying faster than all of the DFBC flights. The trajectories of the G&CNet and the fastest DFBC flight can be seen in Fig. 11. Another interesting observation is that the G&CNet flies slower with this added weight than it did in the previous flight. Here our method exhibits a clear advantage over DFBC, as it does not require a reference trajectory, and can dynamically adjust its course in real time. Furthermore, if we compare the rpm commands of both methods (Fig. 12) it can be seen that the G&CNet can handle sustained rpm saturations, while the DFBC method at $\alpha = 1.05$ experiences similar saturations (at the same propeller) and crashes.

5. Discussion & future work

It is important to contextualize our results compared to current state of the art control methods. In comparison with traditional controllers like DFBC or similarly NMPC, our approach offers a unique solution. While these traditional methods involve a multistage process,
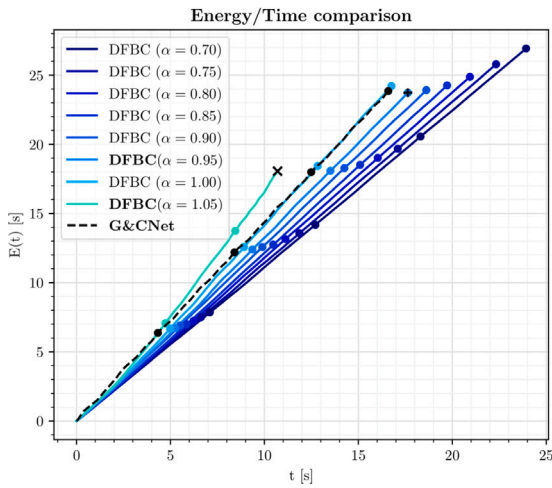
Fig. 10. Energy plotted over time during 4 laps of the 3 × 4 m track with the added weight. The points in time where a lap is completed are represented by a dot. The DFBC method that uses the least energy is marked with a "+". The flight that crashes is marked with an "×".
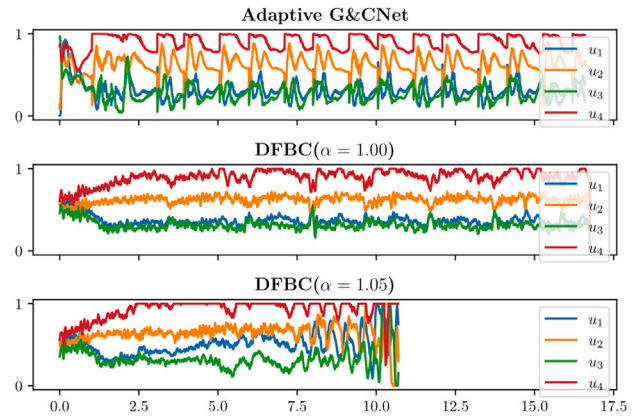

Fig. 12. Comparison of the normalized RPM commands of the Adaptive G&CNet compared to the fastest DFBC flight at $\alpha = 1.00$ and the failed flight at $\alpha = 1.05$.
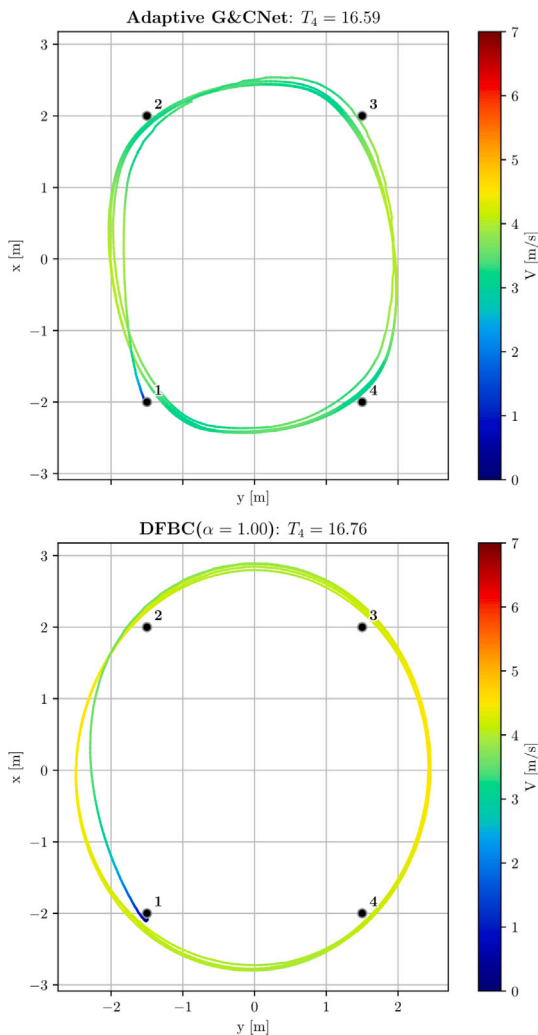

Fig. 11. Top-down view of the adaptive G&CNet's flight and the fastest DFBC's flight at $\alpha = 1.0$ with the added weight.
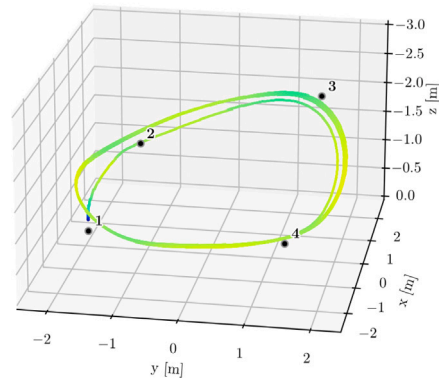

Fig. 13. Trajectory of the adaptive G&CNet through a 4 × 3 m track where the 3rd waypoint is raised by 1 meter.

with pre-computed trajectories and onboard tracking controllers, our methodology integrates planning and tracking into a unified control system. We achieve this by training neural networks with optimal trajectory data, allowing them to approximate the best path for each state. Therefore, our work can be viewed as a specialized form of MPC, where the network offers an approximate solution to the end-to-end trajectory optimization problem without any limitations tied to a finite time horizon. Our method boasts several significant advantages. Firstly, it achieves a substantial reduction in onboard computation time, notably highlighted when comparing the approximately 8 s required by a powerful server to solve the optimal control problem using SNOPT versus the mere 0.002 s it takes the trained network to calculate optimal control on the drone's limited processor. Our approach enhances optimality by streamlining control layers found in traditional methods and exhibits improved robustness against unmodeled moments, as shown in our robustness experiment. However the main disadvantage of our method is that it requires substantial upfront offline computation for generating extensive trajectory datasets and training the network, which may pose a limitation.

In the broader context, the field of neural network based optimal quadcopter control is still in its early stages, yet it has already delivered extremely impressive results such as those detailed in the Nature paper [27] in which a neural controller beats human drone racing champions. However, current methods are highly specialized and lack generalizability [21,27,28]. These approaches often rely on neural
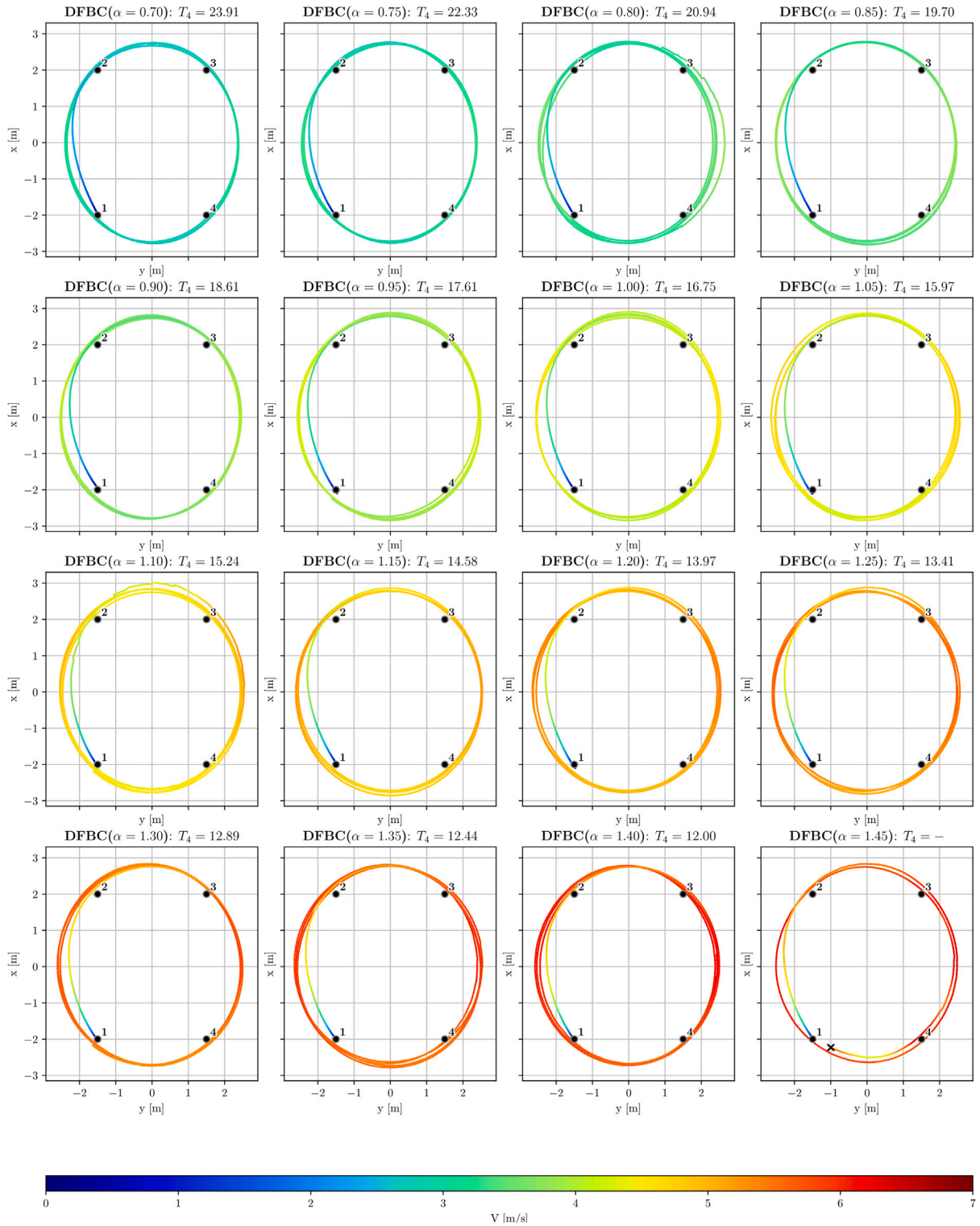
Fig. 14. Top down view of all the DFBC trajectories used in the energy/time comparison from Section 4.2.3.

networks trained for specific track layouts and require complex modeling of aerodynamic effects, including parametric models, Gaussian processes, and neural networks, sometimes even predicting modeling errors based on global position coordinates [27]. We acknowledge that, akin to other studies, our network is highly specialized, as it results from the utilization of a finely tuned quadcopter model and tailored trajectory datasets for specific flights. Nevertheless, we have taken measures to move towards a more versatile form of control. Through

training our network on a dataset encompassing a varied range of initial conditions and modeling errors, we have already made notable advancements in this direction. Even within the constraints of our approach, our neural network has demonstrated its ability to execute various trajectory types, as detailed in Appendix A of the paper.

Furthermore it is important to emphasize that our primary focus is introducing end-to-end optimal quadcopter control for high-speed flight. While there are some limits in terms of generalizability, we have
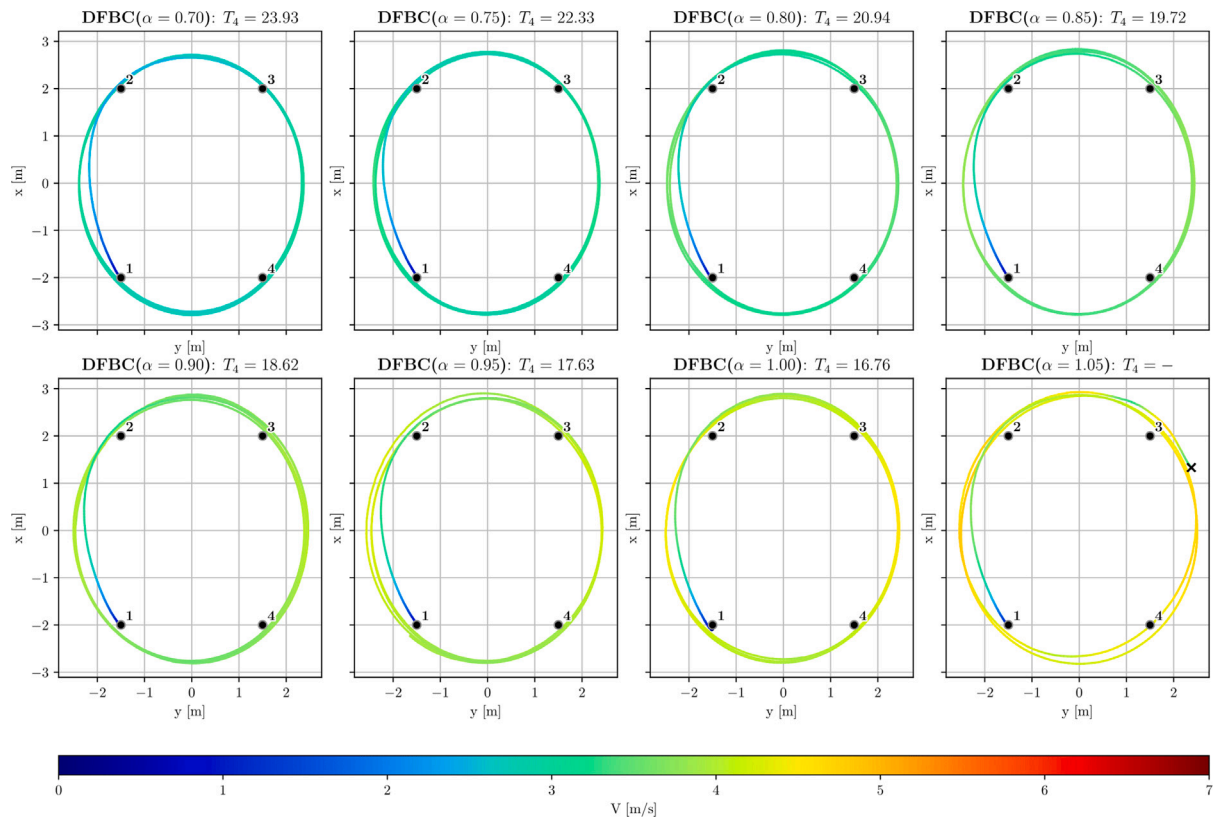
Fig. 15. Top down view of all the DFBC trajectories used in the robustness experiment from Section 4.2.4.

effectively demonstrated the unique aspects of our approach, including real-time optimal control at a high on-board frequency of 500 Hz, the seamless integration of guidance and control, and the development of a method to bridge the gap between simulation and real-world applications.

Our method can be extended in several ways to enhance its capabilities. A logical next step would be to widen our datasets to include a greater variety of starting conditions, allowing us to perform a wider range of maneuvers using our approach. Additionally, we could enhance our neural networks by introducing additional inputs. This would enable them to represent solutions to parametric optimal control problems, accommodating variations in final conditions and additional waypoint constraints. Furthermore, our approach has the potential for extension to real-world scenarios where obstacle avoidance plays a critical role. Given that our neural networks offer an computationally efficient means of planning optimal trajectories, they could be integrated with obstacle avoidance algorithms. In this setup, the obstacle avoidance algorithm would identify safe waypoints devoid of obstacles, and subsequently, the G&CNets would be employed to plan optimal trajectories towards these waypoints. This approach is reminiscent of [40], which incorporates obstacle-free guidance systems as local planners alongside a probabilistic waypoint planner, opening up promising paths for future exploration. While the effectiveness of these strategies awaits further research and exploration, the prospects are encouraging.

6. Conclusion

We have presented a novel G&CNet setup to perform energy-optimal end-to-end quadcopter control. Our approach introduces an efficient algorithm for on-board optimal control computation, eliminating the requirement for a reference trajectory or inner-loop controller, albeit

with the initial offline computation time as a minor trade-off. With real flights, we have investigated the performance of this G&CNet, revealing that unmodeled moments were a significant contributor to the reality gap. To mitigate these effects, we proposed and implemented an adaptive control strategy that shows a significant improvement in flight performance. Furthermore, we compare our proposed adaptive G&CNet to a DFBC method in consecutive waypoint flight scenarios, revealing clear advantages of our method over DFBC. Specifically, our method is more energy efficient, robust against large disturbances, and more flexible, with the ability to dynamically adjust its path in real time without relying on a reference trajectory.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

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

Appendix A. Varying altitude

The adaptive G&CNet utilized in Section 4.2 has, thus far, only been used to fly through a set of waypoints constrained to a horizontal plane. To exhibit the versatility of the trained G&CNet, we will now execute a flight along the same 3×4 m track, but with one waypoint

positioned 1 meter higher in altitude. Fig. 13 shows the trajectory of this flight. Remarkably, even though the network was trained with a narrow range of $+-0.5$ m in z variation, it adeptly navigates through all the waypoints. This demonstration not only underscores the network's capability to navigate complex 3D paths but also its ability to generalize to some degree beyond the provided dataset.

Appendix B. DFBC trajectories

Figs. 14 and 15 show a top-down view of all the performed DFBC flights.

Appendix C. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.robot.2023.104588.

References

[1] M. Hassanalian, A. Abdelkefi, Classifications, applications, and design challenges of drones: A review, Prog. Aerosp. Sci. 91 (2017) 99–131.

[2] L. Bauersfeld, D. Scaramuzza, Range, endurance, and optimal speed estimates for multicopters, CoRR abs/2109.04741, 2021, URL https://arxiv.org/abs/2109.04741.

[3] M.J. Van Nieuwstadt, R.M. Murray, Real-time trajectory generation for differentially flat systems, Internat. J. Robust Nonlinear Control 8 (11) (1998) 995–1020.

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

[5] M. Faessler, A. Franchi, D. Scaramuzza, Differential flatness of quadrotor dynamics subject to rotor drag for accurate tracking of high-speed trajectories, IEEE Robot. Autom. Lett. 3 (2) (2017) 620–626.

[6] E. Tal, S. Karaman, Accurate tracking of aggressive quadrotor trajectories using incremental nonlinear dynamic inversion and differential flatness, IEEE Trans. Control Syst. Technol. 29 (3) (2020) 1203–1218.

[7] P. Ru, K. Subbarao, Nonlinear model predictive control for unmanned aerial vehicles, Aerospace 4 (2) (2017) URL https://www.mdpi.com/2226-4310/4/2/31.

[8] D. Bicego, J. Mazzetto, R. Carli, M. Farina, A. Franchi, Nonlinear model predictive control with enhanced actuator model for multi-rotor aerial vehicles with generic designs, J. Intell. Robot. Syst. 100 (3–4) (2020) 1213–1247.

[9] C. Liu, H. Lu, W.-H. Chen, An explicit MPC for quadrotor trajectory tracking, in: 2015 34th Chinese Control Conference, CCC, 2015, pp. 4055–4060.

[10] G. Torrente, E. Kaufmann, P. Foehn, D. Scaramuzza, Data-driven MPC for quadrotors, IEEE Robot. Autom. Lett. 6 (2) (2021) 3769–3776, http://dx.doi.org/10.1109/lra.2021.3061307, arXiv:2102.05773.

[11] A. Romero, S. Sun, P. Foehn, D. Scaramuzza, Model predictive contouring control for near-time-optimal quadrotor flight, CoRR abs/2108.13205, 2021, URL https://arxiv.org/abs/2108.13205.

[12] A. Romero, R. Penicka, D. Scaramuzza, Time-optimal online replanning for agile quadrotor flight, IEEE Robot. Autom. Lett. 7 (3) (2022) 7730–7737.

[13] S. Sun, A. Romero, P. Foehn, E. Kaufmann, D. Scaramuzza, A comparative study of nonlinear mpc and differential-flatness-based control for quadrotor agile flight, IEEE Trans. Robot. (2022).

[14] D. Hanover, P. Foehn, S. Sun, E. Kaufmann, D. Scaramuzza, Performance, precision, and payloads: Adaptive nonlinear MPC for quadrotors, IEEE Robot. Autom. Lett. 7 (2) (2022) 690–697.

[15] P. Foehn, A. Romero, D. Scaramuzza, Time-optimal planning for quadrotor waypoint flight, Science Robotics 6 (56) (2021) eabh1221.

[16] M.W. Mueller, M. Hehn, R. D'Andrea, A computationally efficient motion primitive for quadrocopter trajectory generation, IEEE Trans. Robot. 31 (6) (2015) 1294–1310.

[17] S. Tankasala, C. Pehlivanturk, E. Bakolas, M. Pryor, Smooth time optimal trajectory generation for drones, 2022, arXiv preprint arXiv:2202.09392.

[18] M. Geisert, N. Mansard, Trajectory generation for quadrotor based systems using numerical optimal control, CoRR abs/1602.01949, 2016, URL http://arxiv.org/abs/1602.01949.

[19] Y. Mao, M. Szmuk, X. Xu, B. Açikmese, Successive convexification: A superlinearly convergent algorithm for non-convex optimal control problems, 2018, arXiv preprint arXiv:1804.06539.

[20] Y. Yu, K. Nagpal, S. Mceowen, B. Açikmeşe, U. Topcu, Real-time quadrotor trajectory optimization with time-triggered corridor constraints, 2022, arXiv preprint arXiv:2208.07259.

[21] Y. Song, M. Steinweg, E. Kaufmann, 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.

[22] G. Tang, W. Sun, K. Hauser, Learning trajectories for real-time optimal control of quadrotors, in: IEEE/RSJ Intl Conf on Intelligent Robots and Systems, IEEE, 2018, URL https://par.nsf.gov/biblio/10100589.

[23] Q. Li, J. Qian, Z. Zhu, X. Bao, M.K. Helwa, A.P. Schoellig, Deep neural networks for improved, impromptu trajectory tracking of quadrotors, in: 2017 IEEE International Conference on Robotics and Automation, ICRA, IEEE, 2017, pp. 5183–5189.

[24] S. Li, Y. Wang, J. Tan, Y. Zheng, Adaptive RBFNNs/integral sliding mode control for a quadrotor aircraft, Neurocomputing 216 (2016) 126–134, URL https://www.sciencedirect.com/science/article/pii/S0925231216307780.

[25] J. Hwangbo, I. Sa, R. Siegwart, M. Hutter, Control of a quadrotor with reinforcement learning, IEEE Robot. Autom. Lett. 2 (4) (2017) 2096–2103.

[26] E. Kaufmann, A. Loquercio, R. Ranftl, M. Müller, V. Koltun, D. Scaramuzza, Deep Drone Acrobatics, in: RSS: Robotics: Science, and Systems, Robotics: Science and Systems Foundation, Corvalis, Oregon, USA, 2020, pp. 1–10, arXiv:2006.05768.

[27] E. Kaufmann, L. Bauersfeld, A. Loquercio, M. Müller, V. Koltun, D. Scaramuzza, Champion-level drone racing using deep reinforcement learning, Nature 620 (7976) (2023) 982–987, http://dx.doi.org/10.1038/s41586-023-06419-4.

[28] Y. Song, A. Romero, M. Müller, V. Koltun, D. Scaramuzza, Reaching the limit in autonomous racing: Optimal control versus reinforcement learning, Science Robotics 8 (82) (2023) eadg1462.

[29] S. Li, E. Öztürk, C.D. Wagter, G.C.H.E. de Croon, D. Izzo, Aggressive online control of a quadrotor via deep network representations of optimality principles, CoRR abs/1912.07067, 2019, URL http://arxiv.org/abs/1912.07067.

[30] C. Sánchez-Sánchez, D. Izzo, Real-time optimal control via deep neural networks: Study on landing problems, J. Guid. Control Dyn. 41 (2016).

[31] D. Tailor, D. Izzo, Learning the optimal state-feedback via supervised imitation learning, Astrodynamics 3 (4) (2019) 361–374.

[32] L. Cheng, Z. Wang, F. Jiang, C. Zhou, Real-time optimal control for spacecraft orbit transfer via multiscale deep neural networks, IEEE Trans. Aerosp. Electron. Syst. 55 (5) (2018) 2436–2450.

[33] L. Cheng, Z. Wang, Y. Song, F. Jiang, Real-time optimal control for irregular asteroid landings using deep neural networks, Acta Astronaut. 170 (2020) 66–79.

[34] E.J.J. Smeur, Q. Chu, G.C.H.E. de Croon, Adaptive incremental nonlinear dynamic inversion for attitude control of micro air vehicles, J. Guid. Control Dyn. 39 (3) (2016) 450–461.

[35] J. Svacha, K. Mohta, V.R. Kumar, Improving quadrotor trajectory tracking by compensating for aerodynamic effects, in: 2017 International Conference on Unmanned Aircraft Systems, ICUAS, 2017, pp. 860–866.

[36] S. Sun, C.C. de Visser, Q. Chu, Quadrotor gray-box model identification from high-speed flight data, J. Aircr. 56 (2) (2019) 645–661.

[37] R. Fourer, D.M. Gay, B.W. Kernighan, A modeling language for mathematical programming, Manage. Sci. 36 (5) (1990) 519–554.

[38] P.E. Gill, W. Murray, M.A. Saunders, SNOPT: An SQP algorithm for large-scale constrained optimization, SIAM Rev. 47 (1) (2005) 99–131.

[39] B. Gati, Open source autopilot for academic research-the paparazzi system, in: American Control Conference, ACC, 2013, IEEE, Washington, DC, 2013, pp. 1478–1481, http://dx.doi.org/10.1109/ACC.2013.6580045.

[40] E. Frazzoli, M.A. Dahleh, E. Feron, Real-time motion planning for agile autonomous vehicles, J. Guid. Control Dyn. 25 (1) (2002) 116–129.

Robin Ferede received the M.Sc. degree in aerospace engineering from Delft University of Technology, Delft, The Netherlands, in 2022. His graduation work focused on end-to-end neural network-based optimal quadcopter control. Since 2022, he has been working toward a Ph.D. degree. His research interests lie in combining optimal control theory with machine learning algorithms to address the challenges associated with autonomous quadcopter flight.

Guido de Croon received his M.Sc. and Ph.D. in the field of Artificial Intelligence (AI) at Maastricht University, the Netherlands. His research interest lies in computationally efficient algorithms for robot autonomy, with an emphasis on computer vision and evolutionary robotics. Since 2008 he has worked on algorithms for achieving autonomous flight with small and lightweight flying robots, such as the DelFly flapping wing MAV. In 2011-2012, he was a research fellow in the Advanced Concepts Team of the European Space Agency, where he studied topics such as optical flow-based control algorithms for extraterrestrial landing scenarios. Currently, he is a full professor at TU Delft and scientific lead of the Micro Air Vehicle lab (MAV-lab) of the Delft University of Technology.
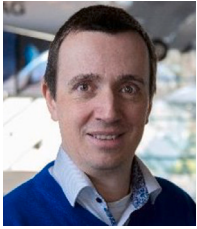
Christophe De Wagter received his M.Sc. in Aerospace Engineering at the Delft University of Technology in 2004 on the topic of vision-based control. In 2005 he created the Micro Air Vehicle Lab where he worked as a researcher until he obtained a Ph.D. in robotics. His areas of interest range from control theory and sensor fusion to computer vision, electronics and AI. He proposed novel concepts like the DelFly, and worked on the DelftaCopter and hydrogen-powered Nederdrone. In parallel, he has been a freelance electronics and software developer for local startup companies and is a private pilot, glider pilot, and certified drone pilot. He is also the safety manager for the MAVLab drone operations. Over the years he won many awards ranging from the 1st prize for "Best Fully Autonomous Indoor MAV" at the EMAV 2008 in Braunschweig, to the "World Champion in Artificial Intelligence Drone Racing", at the AIRR-2019 in the United States.

Dario Izzo graduated as a Doctor of Aeronautical Engineering from the University Sapienza of Rome (Italy). He then took a second master in Satellite Platforms at the University of Cranfield in the United Kingdom and completed his Ph.D. in Mathematical Modelling at the University Sapienza of Rome where he lectured classical mechanics and space flight mechanics. Dario Izzo later joined the European Space Agency and became the scientific coordinator of its Advanced Concepts Team. He devised and managed the Global Trajectory Optimization Competitions events, the ESA Summer of Code in Space and the Kelvins innovation and competition platform. He published more than 170 papers in international journals and conferences making key contributions to the understanding of flight mechanics and spacecraft control and pioneering techniques based on evolutionary and machine-learning approaches. Dario Izzo received the Humies Gold Medal and led the team winning the 8th edition of the Global Trajectory Optimization Competition.