Journal of Intelligent & Robotic Systems (2021) 101: 21
https://doi.org/10.1007/s10846-020-01295-w

Controlling Draft Interactions Between Quadcopter Unmanned Aerial Vehicles with Physics-aware Modeling

Ion Matei[1] · Chen Zeng[2] · Souma Chowdhury[2] · Rahul Rai[2] · Johan de Kleer[1]

Received: 10 July 2020 / Accepted: 1 December 2020

Abstract

In this paper, we address the problem of multiple quadcopter control, where the quadcopters maneuver in close proximity resulting in interference due to air-drafts. We use sparse experimental data to estimate the interference area between palm sized quadcopters and to derive physics-infused models that describe how the air-draft generated by two quadcopters (flying one above the other) affect each other. The observed significant altitude deviations due to airdraft interactions, mainly in the lower quadcopter, is adequately captured by our physics infused machine learning model. We use two strategies to mitigate these effects. First, we propose non-invasive, online and offline trajectory re-planning strategies that allow avoiding the interference zone while reducing the deviations from desired minimum snap trajectories. Second, we propose invasive strategies that re-design control algorithms by incorporating the interference model. We demonstrate how to modify the standard quadcopter PID controller, and how to formulate a model predictive control approach when considering the interference model. Both invasive and non-invasive strategies show significant reduction in tracking error and control signal energy as compared to the case where the interference area is ignored.

Keywords Air draft interactions · Model predictive control · Physics-infused machine learning · Trajectory planning · Unmanned Aerial Vehicle (UAV)

1 Introduction

With growing popularity of unmanned aerial vehicles (UAVs), as more UAVs start operating in close proximity of each other it becomes important to consider potential interactions between UAVs in the context of control and operations planning. In this paper, we present new approaches to model air draft interactions among quadcopter UAVs with physics-infused machine learning, followed by trajectory planning and control solutions to mitigate detrimental impacts (e.g., on stability) of such interactions. The remainder of this introduction section motivates the need for these solutions, briefly surveys existing approaches w.r.t. modeling and regulating air-draft interactions between UAVs, and converges on the objectives of this paper.

✉ Souma Chowdhury
soumacho@buffalo.edu

[1] Palo Alto Research Center, Inc. (PARC), Palo Alto, CA 94304, USA

[2] Department of Mechanical and Aerospace Engineering, University at Buffalo, Buffalo, NY 14260, USA

1.1 Aerodynamic Interactions Between UAVs

Multirotor UAVs are seeing expanding applications in various civilian, commercial and humanitarian domains [19]. Chief among these emerging applications are close coordinated operation of multiple UAVs, namely UAV swarm and formation flight paradigms. Such paradigms involve a team of small UAVs collaborating in a certain manner to offer sensing, transportation, monitoring and other related solutions with increased redundancy and effectiveness [7, 40]) compared to sophisticated standalone alternatives. In such multi-UAV or shared airspace applications, UAVs often operate in close proximity of each other: examples of such scenarios include forming a narrow queue to pass through tight spaces [12], agricultural UAVs sweeping through the field in a line for uniform fertilization [13], or hundreds of UAVs with LED formations for aerial light shows [2].

Two important directions of research with regards to operational safety emerge from such increasing occurrence of close proximity operations: 1) collision avoidance between UAVs and 2) air draft interactions. While the former has received substantial attention in the past decade

or so [6, 14, 31], significantly less work has been done in the latter area, which is the focus of this paper. An aircraft creates aerodynamic disturbances around itself and along the flight path. Downwash is the airflow that blows downward, particularly created by rotor crafts [32, 33]. Wake turbulence on the other hand is a series of turbulent airflow that are present along the flight path of a moving aircraft [42, 49]. Both the downwash and wake turbulence have been extensively studied for manned aircraft to mitigate dangers in commercial and civil flights [15], and regulations mandate that manned aircraft maintain certain threshold separation distances from each other (usually in kilometers). In the case of small multirotor UAVs ($<$ 20 kg class), there exists a body of work in the aerodynamics domain that study the flow induced by the hovering [21, 47, 51], cruising [44] and maneuvering [24] flight states of standalone UAVs. Given the focus of this paper is on dynamics/control scale modeling and regulation of the interactions (of interest to the robotics and swarm system community), more comprehensive discussion of the literature on the flow physics of multi-rotor UAVs is outside the scope of this paper. Such details can be found in [50, 60].

While air-draft interactions between UAVs are in principle similar to those in the case of large manned aircraft, the solution of maintaining separation thresholds with high safety factors may be overtly restrictive to intended functionalities (e.g., in swarm/multi-UAV operations) or unnecessary (since considerations such as passenger comfort is not needed). This calls for investigations on modeling and mitigating air draft interactions specific to small UAVs. With proper modeling of UAV-UAV interactions and active countermeasures designed thereof, it would become possible for UAVs to fly beside, after, or on top of each other in very close proximity without drifting or losing control. Note that, given their significant complexity (in terms of motion planning and control), quadcopters provide a representative example to study the broader important problem of indirect physical interactions among intelligent systems. Furthermore, quadcopters are one of the most popular types of aerial robots, and their development and widespread usage has tangibly contributed to the growth of interest in robotics (among researchers, STEM students and the general population). This makes them an exciting vehicle to demonstrate the modeling benefits offered by the emerging physics-infused machine learning approaches in robotics and controls problems.

1.2 Methods to Model and Mitigate Interaction Effects

Conventionally, interactions between UAVs are simply modeled as distance-based disturbances. Hönig et al [23] modeled the effect in a form similar to that of the ground effect during path planning of the swarm. More refined approaches include reduced-order flow modeling (Yeo et al [59]), propeller velocity field modeling (Jain et al [25]), and data-driven numerical modeling (Shi et al [48]). Most of the aforementioned modeling approaches study single-UAV scenarios, focusing on simulating the ground effect for takeoff and landing. The computing cost of the fluid or velocity field models bring difficulties for applications to multi-UAV control. The data-driven models (e.g., empirically derived surrogate models [51] and machine learning models) do not provide any clear pathway to generalize beyond the specific platforms used for collecting the data.

From the motion control perspective, various approaches have been applied to evade or counteract the UAV interaction effects. Hönig et al [23] suggested maintaining wide separations that guarantees stability at a significant cost of restricting the range of motions. Motion planning approaches have also been studied to minimize these restrictions and the detrimental impact of interactions among UAVs [3, 58]. Our goal is to significantly extend this body of work by proposing both invasive and non-invasive, offline and online strategies to mitigate interference effects. The strategies apply to scenarios defined by the amount of available experimental data, access to onboard hardware firmware, control modalities and computational resources.

1.3 Research Objectives

In this paper, we focus on interactions between small quadcopter UAVs when they are roughly on top of each other. The formal problem descriptions and interaction-effect mitigation schemes are however extendable to other scenarios. Within this scope, the objectives of this paper are three-fold:

I. To formally describe the UAV-interactions problem, and design/use experiments to learn how to represent the interaction between quadcopter UAVs in a manner that can facilitate decisions in the control/planning time-scale.
II. To develop and test a *non invasive* approach for modifying the trajectories of interacting UAVs with the objective to minimize interference effects and deviations from desired trajectories.
III. To design and test (in simulation) interaction-aware controllers that will minimize drift and instability caused by the draft effects of the upper UAV, when trajectory modification is not viable (e.g., to preserve functionality).

Our *contributions* with respect to the stated objectives are illustrated in Fig. 1, and summarized below:

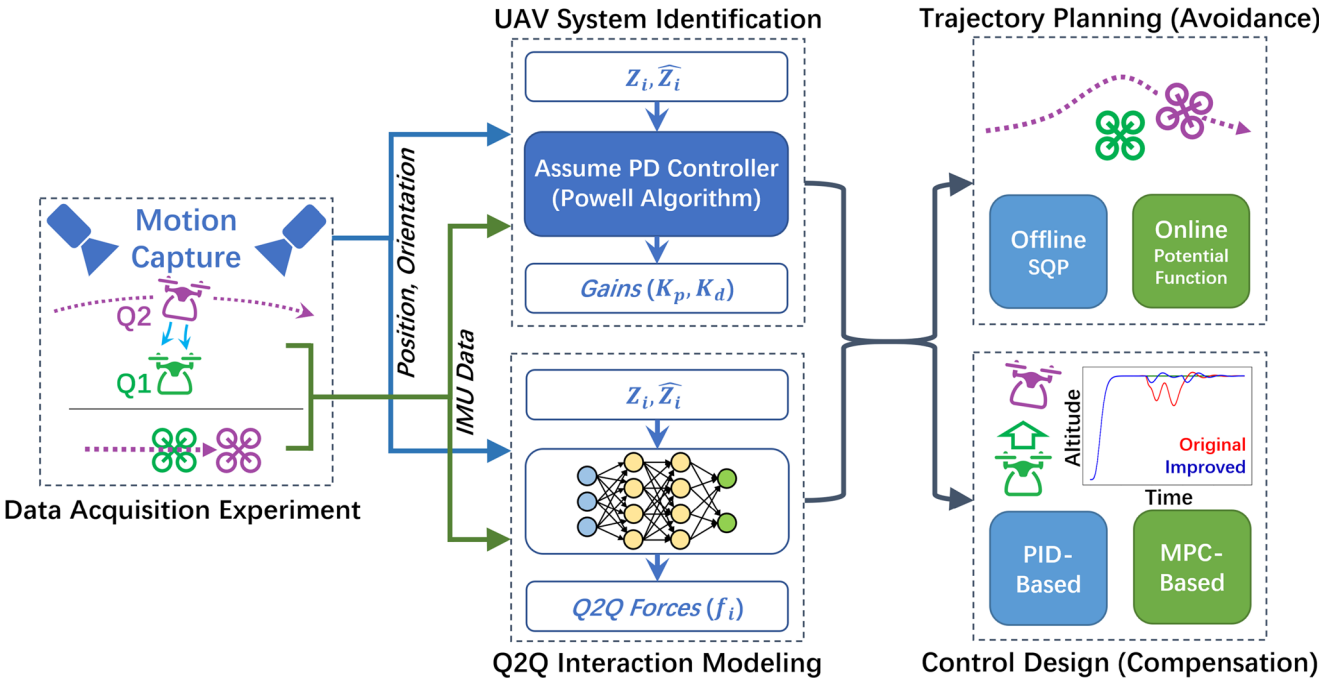

Fig. 1 Computational Framework: Modeling, avoiding (via trajectory planning) and mitigating (via controller design) effects of air-draft interactions between Quadcopters

1. *Interaction modeling*: We propose and train machine learning-driven interaction models, represented by a combination of neural networks and physics guided structure-imposing functions to deal with limited data and lack of controller information. The physics-awareness comes from the physical interpretation of the model (i.e., it is the gradient of a potential energy function), and from the imposed structure of the model (i.e., type of decay) determined from physical experimental data.
2. *Trajectory re-planning*: Design an optimisation-based forward-looking algorithm for computing minimum snap trajectories with interference zone avoidance. Design a low-complexity, myopic potential function-based algorithm for parsimoniously changing quadcopter trajectories to minimize interference effects.
3. *PID Controller re-design*: Re-design the standard PID controller by including the interaction models and accounting thereof for the interference effects. The algorithm has an inherent distributed implementation, i.e., each quadcopter uses only its own information and information form neighbors.
4. *MPC controller design*: Formulate a model predictive control (MPC) approach for both quadcopters to jointly compensate for interaction effects. The optimization algorithm is based on sequential quadratic programming (SQP); it uses global parameterizations of trajectories and inputs implemented through neural networks models, and automatic differentiation to generate state derivatives and Jacobians of objective and constraint functions, respectively.

The remainder of the paper is organized as follows: Section 2 formally describes the UAV-interactions problem and the physical experiments conducted with two quadcopters to collect data for learning the interactions model. Section 3 elaborates on the learning based model of UAV interactions. Section 4 introduces the non-invasive, forward looking and the myopic algorithms for trajectory modification to avoid or mitigate interactions, and Section 5 describes the control approaches aimed at minimizing the impact of the interactions. The paper ends with concluding remarks.

2 UAV Interaction: Problem Description

In this section, we introduce the dynamical models used for designing feedback control loops and enumerate the problems tackled in the following sections. In addition, we describe the experimental setup for collecting training data.

2.1 Models and Problems

Here we consider an X-shaped quadcopter UAV. The control design is based on the quadcopter dynamics model that

includes its behavior when operating in proximity to other quadcopters. We first start with the nominal quadcopter model, i.e., the model that does not include interferences. Let $\boldsymbol{a} = (\phi, \theta, \psi)$ denote the orientation angles (pitch, yaw, roll) and let $\mathrm{r} = (x, y, z)$ denote the quadcopter position. The single quadcopter dynamics model [9] defining the forces and torques acting on the quadcopter is given by:

$$\ddot{\phi} = \dot{\theta}\dot{\psi}a_1 + \dot{\theta}a_2\Omega_r + b_1U_2, \tag{1}$$

$$\ddot{\theta} = \dot{\phi}\dot{\psi}a_3 - \dot{\phi}a_4\Omega_r + b_2U_3, \tag{2}$$

$$\ddot{\psi} = \dot{\theta}\dot{\phi}a_5 + b_3U_4, \tag{3}$$

$$\ddot{x} = \frac{U_1}{m}u_x, \tag{4}$$

$$\ddot{y} = \frac{U_1}{m}u_y, \tag{5}$$

$$\ddot{z} = -g + \frac{U_1}{m}\cos\phi\cos\theta. \tag{6}$$

Here $a_1 = (I_{yy} - I_{zz})/I_{xx}$, $a_2 = J_r/I_{xx}$, $a_3 = (I_{zz} - I_{xx})/I_{yy}$, $a_4 = J_r/I_{yy}$, $a_5 = (I_{xx} - I_{yy})/I_{zz}$, $b_1 = l/I_{xx}$, $b_2 = l/I_{yy}$, $b_3 = l/I_{zz}$, $u_x = \cos\phi\sin\theta\cos\psi + \sin\phi\sin\psi$, and $u_y = \cos\phi\sin\theta\sin\psi - \sin\phi\cos\psi$. In the system dynamics, m is the quadcopter mass, l is its arm length, I_{xx}, I_{yy} and I_{zz} are the moment of inertia corresponding to the three axes, and U_i, $i \in \{1, 2, 3, 4\}$ are control inputs that depend on the propellers' angular velocities. We compactly re-write the quadcopter dynamics as $\ddot{\boldsymbol{a}} = f_a(\dot{\boldsymbol{a}}, \boldsymbol{U})$ and $\ddot{\mathrm{r}} = f_r(\boldsymbol{a}, \boldsymbol{U})$, where $\boldsymbol{U} = [U_1, U_2, U_3, U_4]$, f_a represents (1)-(3), and f_r represents (4)–(6).

In this paper, we consider interactions between two quadcopters only, although the approach proposed can be extended to more quadcopters. In what follows, we use indices to refer to a particular quadcopter, e.g., r_i denotes the position of quadcopter i, with $i \in \{1, 2\}$. We denote by $\Delta\mathrm{r}$ the relative position between the two quadcopters, and by $\mathrm{r}_{i,\mathtt{ref}}$ the reference trajectories for the quadcopters' positions.

We use two methods for accounting for Q2Q interactions. In the first method we retain the original controllers, but minimally change the reference trajectories such that they avoid high interference zones. Hence we need to mathematically characterize the interference area and use it to re-generate the reference trajectories. This first method has the advantage that it is *non-invasive*. However on the downside, this could restrict certain coordinated functionalities by the UAVs that are tightly coupled with their action trajectories, e.g., synced mapping or scanning, coordinated transportation of object, and coordinated rope/cable manipulation. In the second approach we *augment the quadcopter model* with additional behavior that mimics the interference effects, and use the augmented model to *redesign* the controllers. In particular, we computationally model the interference effects and additional forces acting on the quadcopters. In our second method, we perform control redesign with two approaches: PID control and model predictive control (MPC). This method, while being invasive, alleviates any restriction on the coordinated functionality where the quadcopters need to operate close to each other without compromising stability. Below we formally state the problems addressed in this paper.

Problem 1 Find a mathematical description for a compact set $\mathcal{I}$ defined in the 3-d space of the relative position, so that if $\Delta\mathrm{r}(t) \bigcap \mathcal{I} = \{\emptyset\}$, the quadcopters' behavior is nominal.

Problem 2 Given the interference zone $\mathcal{I}$, redesign the reference trajectories $\mathrm{r}_{j,ref}(t)$, so that $\Delta\mathrm{r}_{\mathtt{ref}}(t) \bigcap \mathcal{I} = \{\emptyset\}$ for all t, where $\Delta\mathrm{r}_{\mathtt{ref}}(t) = \mathrm{r}_{1,\mathtt{ref}} - \mathrm{r}_{2,\mathtt{ref}}$.

Problem 3 Learn the interference-generated forces $\boldsymbol{F}(\Delta\mathrm{r})$ that contribute to the quadcopter linear accelerations $\ddot{\mathrm{r}}$.

Problem 4 Use the augmented quadcopter model to redesign the controllers such that they account for the interference-generated forces $\boldsymbol{F}(\Delta\mathrm{r})$.

The estimation of the interference zone and its effects are done using training data obtained through physical experiments. The following subsection describes the experimental setup used to collect the data.

2.2 Experimental Setup

Here our goal is to first gather the flight data of the UAV platform for the system identification process. To this end, we established a hardware implementation platform utilizing a VICON motion capture system to obtain experimental data describing quadcopters behavior. The hardware implementation platform comprises of a ROS software environment running on mobile workstations, a motion capture environment, and two demonstration aircraft namely Crazyflie 2.0 quadcopter platforms [1]. The ROS environment is installed on a Linux computer that acts as the UAV path planner, flight control, and data recorder. A 20-camera VICON motion capture facility captures the positions and orientations of the UAV in 100 Hz. The quadcopter platforms were mildly modified to include collision resilient rotor guards. Further specifications of the quadcopter platforms used here are summarized in Table 1. This UAV platform has been used by other researchers for related studies [18, 51, 52]. The ROS environment on the workstation obtains the aircraft localization information from the motion capture system, and computes and send the control inputs to the Crazyflie quadcopters. This off-board system is required due to the small size of the Crazyflie UAV, which makes it difficult to mount any practical computing node onboard. Figures 2 and 3 show

Table 1 Crazyflie 2.0 UAV Specifications [16]

Parameter	Value
Takeoff Weight	34 g
Size (WxHxD)	92x92x29 mm
Inertia ($kg \cdot m^2$)	$\begin{bmatrix} 3.14 & 5.00 & 4.64 \\ & 3.15 & 4.78 \\ & & 7.06 \end{bmatrix} \times 10^{-5}$
Processing Unit	Cortex-M4, 168MHz, 192kb SRAM, 1Mb flash
Sensing Ability	10 DoF IMU
Communication	20 dBm radio amplifier & Bluetooth
Flight Time	5 minutes

our quadcopter test platforms and the motion capture environment, respectively.

The Q2Q vertical interaction experiments record data to study the disturbances induced when one quadcopter flies over another quadcopter. We conducted 6 experiment flights, in which the Q1 hovers at a fixed point while Q2 passes Q1 from above, causing a down-wash disturbance. The design of (flight) experiments featured various altitudinal distances, flyover speeds, and durations of trajectory overlap, as summarized in Table 2. These flight samples were designed to ensure the occurrence of interference between the UAVs, without resulting in catastrophic failures, and while staying within their operational bounds. The quantities of interest recorded during the experiments include 1) the global position and orientation of the two quadcopters, 2) linear acceleration and angular velocity recorded by onboard IMUs, and 3) attitude control commands (not used in this paper).

The experimental data is used for two main purposes: 1) learning the non-interference quadcopter behavior, i.e., nominal behavior; and 2) learning the interaction model between quadcopters. For the latter, we make assumptions on the model structure (e.g., dependency of the distance between quadcopters), and choose parameterizations that we learn using the experimental data. The data acquired

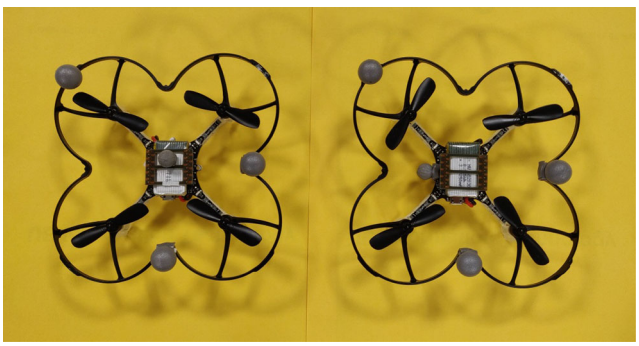

Fig. 2 Crazyflie quadcopters used as the platform for data collection

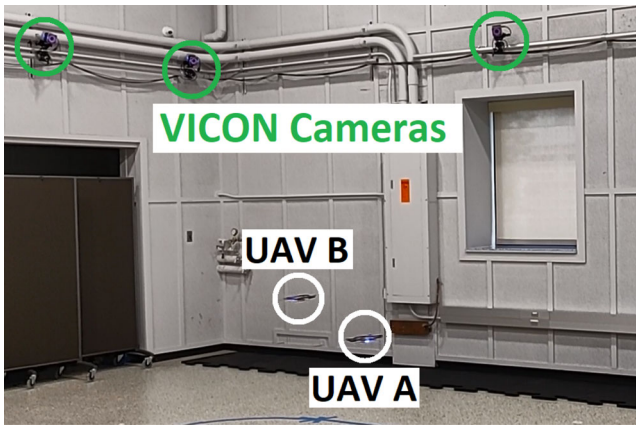

Fig. 3 A snapshot of a demonstration flight in the motion capture environment

by the VICON system, although less noisy than IMU data, did contain outliers. We built a rule based outlier detection schemes, where positions and velocities greater than a threshold determined empirically were considered outliers. The eliminated outliers were replaced by interpolating neighboring points. This scheme was sufficient since the outliers typically had large magnitudes as compared to their neighboring points. The data was visually inspected to ensure no outlier escaped detection.

3 Quadcopter-Quadcopter (Q2Q) Interactions Model

As seen in our experiments, the air draft generated by a quadcopter when being on top of another quadcopter can have significant implications on the trajectory of the latter. The upper quadcopter air-draft acts as an air disturbance on the lower quadcopter and it translates into additional forces and torques acting on the lower quadcopter. We can design robust controllers that are able to tolerate significant disturbances, but they come at a cost of accuracy of the trajectory tracking. A more efficient way to deal with quadcopter-to-quadcopter (Q2Q) interaction is to account for them in the controller design process. Hence we first need to learn a model that describes such an interaction. This is accomplished through interference area estimation,

Table 2 List of flight scenarios for the Q2Q interaction experiments

Property	#1	#2	#3	#4	#5	#6
Distance (m)	0.5	1	0.5	1	0.5	1
Flyover Speed (m/s)	0.5	0.5	1.5	1.5	0.5	0.5
Overlapping Time (s)	<0.5	<0.5	<0.5	<0.5	1	1

followed by learning of the Q2Q interaction forces, as described in the following subsections.

3.1 Interference Area Estimation

In what follows we consider the interaction between two quadcopters. We define the interaction manifold as the set $\mathcal{I}_{\varepsilon,d} = \{\Delta \mathrm{r} |\ |\Delta x|^2 + |\Delta y|^2 \leq \varepsilon^2, |\Delta z| \leq d\}$, for some non-negative scalars ε and d. This formulation is in accordance with our intuition that the upper quadcopter influences the lower quadcopter if they are close enough on the (x, y) plane and at some minimum distance on the z axis. In learning the interaction model our goal is two fold: (*i*) learn the parameters ε and d, and (*ii*) learn the magnitude of the disturbance (e.g., additional forces and torques) between the quadcopters. The characterization of the interaction manifold is particularly relevant when we account for the Q2Q interaction through trajectory (re)-planning. Namely, the reference trajectories of the two quadcopters are re-designed so that the relative distance $\Delta \mathrm{r}$ satisfies $\mathrm{dist}(\Delta \mathrm{r}, \mathcal{I}_{\varepsilon,d}) > 0$. In other words, the two quadcopters avoid the interference areas.

The ε parameter was determined from the geometric specifications of the quadcopters. The distance between two diagonally opposing motors is 92 mm, hence two quadcopters do not overlap on the (x, y) plane if $\sqrt{\Delta x^2 + \Delta y^2} > 92$ mm. As a consequence we can set $\varepsilon = 0.1$ m when defining the interference manifold.

To determine the d parameter, we used the experimental data to evaluate the magnitude of the disturbance induced by quadcopter-2 (Q2) on quadcopter-1 (Q1) when Q2 flies over Q1. For each considered experiment, we select the z-axis behavior only for time instances for which the inequality $d_{xy} = \sqrt{\Delta x^2(t) + \Delta y^2(t)} \leq \varepsilon$ holds and for time instances that belong to the steady state regime, i.e., $t \geq 15$ sec. This results in a subset of time instances $\mathcal{T}_{\{d_{xy} \leq \varepsilon\}}$. We quantify the disturbance affecting the quadcopters as the z-axis trajectory deviations from the reference values. In particular, for each experiment i, we compute the quantity

$$\mathrm{dev}^{(i)} = \frac{1}{|\mathcal{T}_{\{d_{xy} \leq \varepsilon\}}|} \sum_{t \in \mathcal{T}_{\{d_{xy} \leq \varepsilon\}}} |z^{(i)}_{1,\mathtt{ref}}(t) - z^{(i)}_1(t)| + |z^{(i)}_{2,\mathtt{ref}}(t) - z^{(i)}_2(t)|.$$

For each experiment i, we compute the average z-axis distance between quadcopters where only time samples $\mathcal{T}_{\{d_{xy} \leq \varepsilon\}}$ are included, that is, $\Delta z^{(i)}_{av} = \frac{1}{|\mathcal{T}_{\{d_{xy} \leq \varepsilon\}}|} \sum_{t \in \mathcal{T}_{\{d_{xy} \leq \varepsilon\}}} \Delta z^{(i)}(t)$. We make the following assumptions on the disturbance behavior: (i) it asymptotically decays to zero as the distance between quadcopters on the z axis increases, and (ii) it attains the maximum at zero distance. We represent the disturbance magnitude as a decaying exponential: $\mathrm{dev}(\Delta z_{av}) = \alpha e^{-\beta |\Delta z_{av}|}$. We fit the parameters of the decaying exponential model and obtain a decaying model as shown in Fig. 4.

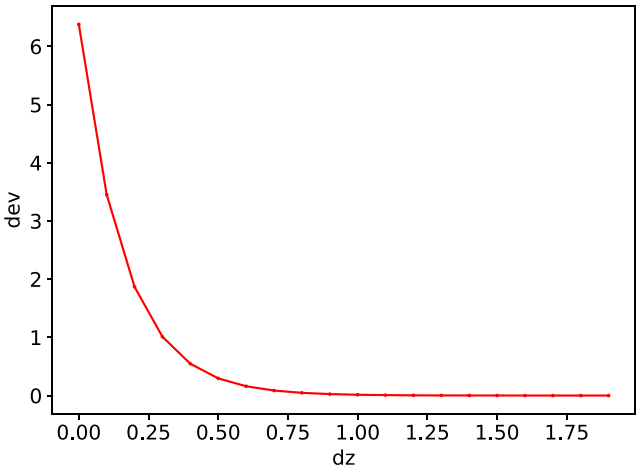

Fig. 4 Disturbance `dev` decays as a function of the relative distance Δz

The decay model is similar to other models in the literature [11] where short-range repulsion and long-range attraction are represented as scaled exponential functions as well. In our case, we concentrate on the z-axis only, since this is where the repulsion is the strongest. Based on the decaying model, we select d to be 1.1 meters, where we allow for some small disturbance that can be taken care by the controllers.

3.2 Physics-infused Learning of Q2Q Interaction Forces

In quantifying the magnitude of the interference, we consider only the additional forces affecting the linear accelerations and neglect the interference effects on the quadcopter orientation. Hence we get the following model for the quadcopter accelerations:

$$m\ddot{x}_i = U_{1,i} u_{x,i} + F_x(\Delta \mathrm{r}) \quad (7)$$

$$m\ddot{y}_i = U_{1,i} u_{y,i} + F_y(\Delta \mathrm{r}) \quad (8)$$

$$m\ddot{z}_i = -mg + U_{1,i} \cos\phi_i \cos\theta_i + F_z(\Delta \mathrm{r}), \quad (9)$$

where index i refers to quadcopter-i, and $F_x(\Delta \mathrm{r})$, $F_y(\Delta \mathrm{r})$, and $F_z(\Delta \mathrm{r})$ are the interference induced forces on the x, y, z axes, respectively. The first step in learning the Q2Q interaction forces is learning the quadcopter model operating without any interference. This is important considering that the control system on off-the-shelf quadcopters are often not accessible (when proprietary), or discernible (even when open source), and the real dynamics could deviate significantly from idealized representations given by typical quadcopter dynamics models. This realism gap, if not tackled early on can significantly impact the quality of the trajectory planning and controller redesign

efforts. We decided to learn the closed-loop model rather the input-output mode, where the control inputs are based on a PID controller. In particular we learn the parameters of the controller such that the simulated behavior closely matches the observed behavior. Consider the altitude control loop, that is, the control loop that acts on the z axis, whose dynamics are given by

$$\ddot{z} = -g + \frac{U_1}{m}\cos\phi\cos\theta. \tag{10}$$

The quadcopter altitude can be controlled by selecting a particular controller structure, namely $U_1 = \frac{\cos\phi\cos\theta}{m}(g + V_1)$, which transforms the altitude dynamics into a double integrator: $\ddot{z} = V_1$. Given a reference signal $z_{\texttt{ref}}$, our objective is to select a control input to drive the error $\varepsilon_z = z - z_{\texttt{ref}}$ to zero. Consider the second order dynamics $\ddot{\varepsilon}_z + K_d\dot{\varepsilon}_z + K_p\varepsilon_z = 0$, with $K_d, K_p > 0$. Such dynamics converge to zero at a rate dependent on the parameters K_d, K_p. Let $V_1 = \ddot{z}_{\texttt{ref}} + K_d(\dot{z}_{\texttt{ref}} - \dot{z}) + K_p(z_{\texttt{ref}} - z)$, or

$$U_1 = \frac{\cos\phi\cos\theta}{m}\left(g + \ddot{z}_{\texttt{ref}} + K_d(\dot{z}_{\texttt{ref}} - \dot{z}) + K_p(z_{\texttt{ref}} - z)\right). \tag{11}$$

To learn the parameters of the altitude controller, we select an experiment where Q1 presets small interference from Q2. In particular, we select an experiment where Q2 flies high, at reduced speed above Q1. Hence interference between them, although present, is small. We formulate an optimization problem in terms of the two parameters of the PD controllers. The loss function is the mean square error (MSE) between the measured and simulated trajectories when using the PD controller:

$$\min_{K_d\geq 0, K_p\geq 0} \ \sum_{i=1}^{N}\|z_1(t_i) - \hat{z}_1(t_i)\|^2 + \sum_{i=1}^{N}\|z_2(t_i) - \hat{z}_2(t_i)\|^2 \tag{12}$$

$$\text{s.t: } \ddot{\hat{z}}_j = \ddot{z}_{j,\texttt{ref}} + K_d(\dot{z}_{j,\texttt{ref}} - \dot{\hat{z}}_j) + K_p(z_{j,\texttt{ref}} - \hat{z}_j), \tag{13}$$

where $j \in \{1, 2\}$ refers to a particular quadcopter and where z_i and $\hat{z}_j$ denote the measured and simulated trajectories, respectively. Since we have two optimization parameters, we used the gradient free `Powell` algorithm to learn the parameters K_d and K_p coupled with an ODE solver. Details about the optimization algorithm are shown in Table 3. Powell algorithm is one example of derivative free optimization algorithm. Other algorithms include the Nelder-Mead method [38], genetic algorithm [20], particle swarm optimization [27] or cuckoo search algorithm [54].

Table 3 Closed loop model learning algorithm information

Algorithm	Powell
xtol	10^{-8}
ftol	10^{-8}
Library	Python/Scipy

Figure 5 shows a comparison between the measured trajectories and the trajectories generated with learned controller parameters. Note that we experimented with introducing an integral term as well in the controller. Since it did not significantly change the results, the integral term was excluded thereafter.

We are essentially using a surrogate models to learn the Q2Q interference effects. In particular, we are using the model

$$\ddot{z}_j = \ddot{z}_{j,\texttt{ref}} + K_d(\dot{z}_{j,\texttt{ref}} - \dot{z}_j) + K_p(z_{j,\texttt{ref}} - z_j) + \eta_j(\Delta\text{r}) \tag{14}$$

where η_j is the additional acceleration affecting quadcopter-j due to air drafts. Note that η_j is the additional force due to air drafts scaled by the mass of the quadcopter. The training data is extracted from the physical experiments where the Q2Q interference is significant. In addition, we make the following assumptions on the interaction function: (i) the forces depend on the relative distance on the $x - y$

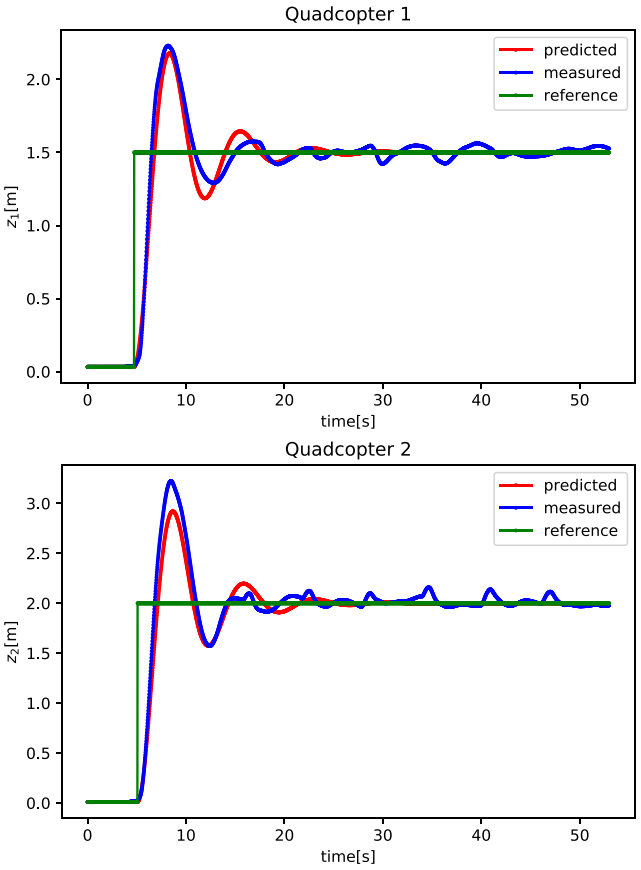

Fig. 5 Quadcopter z-axis trajectories generated using learned PD controller parameters $K_p = 0.81073$, $K_d = 0.42952$. The measured versus predicted trajectories MSE are: $MSE_1 = 0.004$ and $MSE_2 = 0.011$

plane $\Delta r_{xy} = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$ and on the relative distance along the z-axis, $\Delta r_z = z_1 - z_2$. In addition, to cope with the absence of training data for large relative distances, we modulate the forces by a Gaussian function concentrated on the interference zone, as learned previously. Hence we consider the following form for the force function:

$$\eta_j(\Delta \mathrm{r}) = e^{-\Delta r_{xy}^2/(2\varepsilon^2)} e^{-\Delta r_z^2/(2d^2)} f_j(\Delta r_{xy}, \Delta r_z). \tag{15}$$

Here $f_j : \mathbb{R}^2 \to \mathbb{R}$ is a function that we need to learn for each of the two quadcopters. We parameterize functions f_j by neural networks. To train the parameters of the interaction function we formulate a constrained optimization problem that searches for parameters of the two neural networks (ANNs) such that the simulated versus measured trajectories match as closely as possible.

$$\begin{aligned} \min_{w} \quad & \sum_{i=1}^{N} \|z_1(t_i) - \hat{z}_1(t_i)\|^2 + \sum_{i=1}^{N} \|z_2(t_i) - \hat{z}_2(t_i)\|^2 \\ \text{s.t:} \quad & \ddot{\hat{z}}_j = \ddot{z}_{j,\mathrm{ref}} + K_d(\dot{z}_{j,\mathrm{ref}} - \dot{\hat{z}}_j) + K_p(z_{j,\mathrm{ref}} - \hat{z}_j) + \eta_j(\Delta \mathrm{r}; w), \\ & \eta_j(\Delta \mathrm{r}; w) = e^{-\Delta r_{xy}^2/(2\varepsilon^2)} e^{-\Delta r_z^2/(2d^2)} f_j(\Delta r_{xy}, \Delta r_z; w), \end{aligned} \tag{16}$$

where $j \in \{1, 2\}$, and w is the vector of weights and biases for the two ANN modeling the functions $f_j(\Delta r_{xy}, \Delta r_z; w)$. We solved this optimization problem using `Pytorch` [41] since it supports ODE solvers featuring automatic differentiation. Hence we can eliminate the equality constraints and end-up with an unconstrained optimization problem. The ANNs have one hidden layer of size 40 and use `tanh` activation functions. We intentionally chose smaller size models to reduce overfitting. We use the MSE between the measured and predicted trajectories as the loss function. After training, the MSE corresponding to the z-axis trajectories of quadcopters 1 and 2 are 1.13×10^{-3} and 1.87×10^{-3}, respectively. Details of the learning algorithm are shown in Table 4. Note that the training is done on CPU since we have empirically observed that for small ANN models, computations executed on CPU are faster.

Figures 6 and 7 depict the measured and simulated trajectories of Q1 and Q2 respectively, after learning the interaction model. A close match between measured and simulated trajectories is observed in the case of both quadcopters, thereby supporting the use of the learned model for later use in active trajectory modification and controller redesign. In addition, note the significant oscillations experienced by the lower quadcopter (Q1) due to the air draft generated by Q2.

Figures 8 and 9 show the additional accelerations imposed due to the quadcopters getting close to each other. As expected, their magnitudes increase as the relative distance on the z axis decreases.

Table 4 Interference model learning: Neural network and learning settings

Algorithm	Adam [29]
Step-size	10^{-4}
Time per iteration	0.7 sec
Library	Pytorch [41]

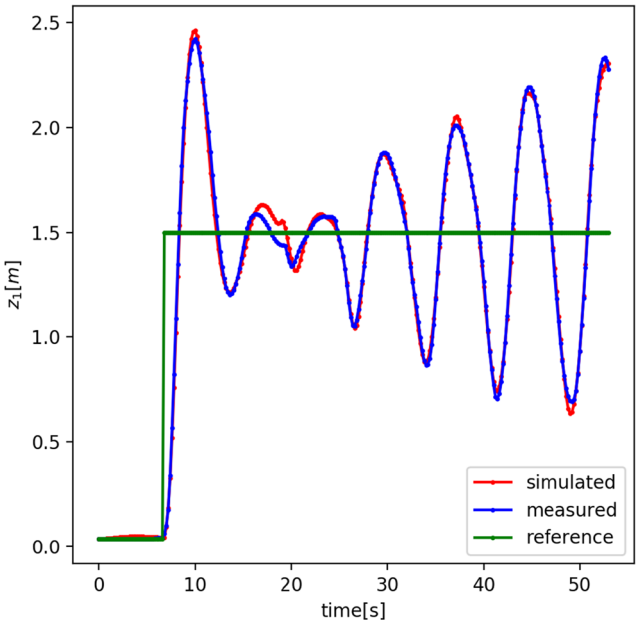

Fig. 6 Q1 (lower UAV): x-axis trajectory including the simulated trajectory after learning the interaction models, the measured trajectory and the reference signal

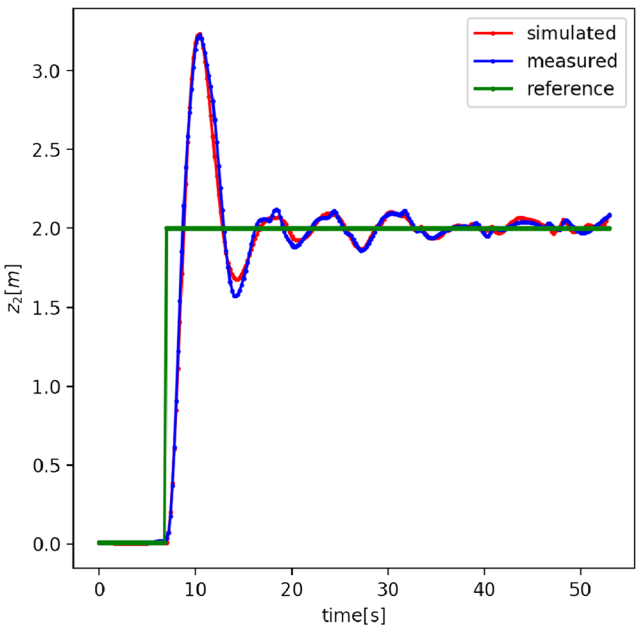

Fig. 7 Q2 (upper UAV): x-axis trajectory including the simulated trajectory after learning the interaction models, the measured trajectory and the reference signal

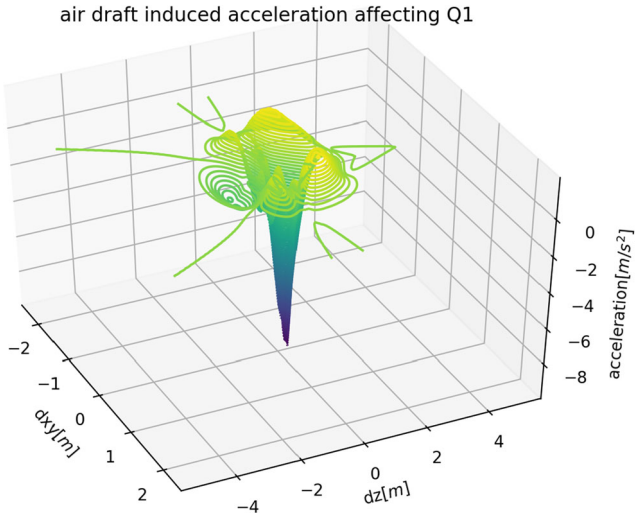

Fig. 8 The additional acceleration (scaled force) affecting Q1 while in proximity of Q2

4 Q2Q Interaction Aware Trajectory Planning

We assume that a planner generates trajectories that the quadcopters are supposed to follow based on some objective. They can be generated using a minimum snap type of approach for example [36]. When accounting for Q2Q interference we add an additional module that re-processes the reference trajectories so that the interference manifold is avoided. We consider two approaches. In the first approach, the reference trajectories over some future time horizon are generated and modified to account for interference. The advantage here is that we can minimize the differences between the original reference trajectories and the modified ones by considering a time horizon, which then helps in preserving the coordinated functionality of the UAVs (often linked with the waypoints they are originally tasked to follow). These minimal differences however come at an increased numerical complexity of the trajectory optimization process. In the second approach, the changes in the trajectories are based only on initial state. This myopic approach is less expensive and easier to implement onboard UAVs; however it is likely to induce larger differences between the original and modified trajectories since a time horizon is not considered.

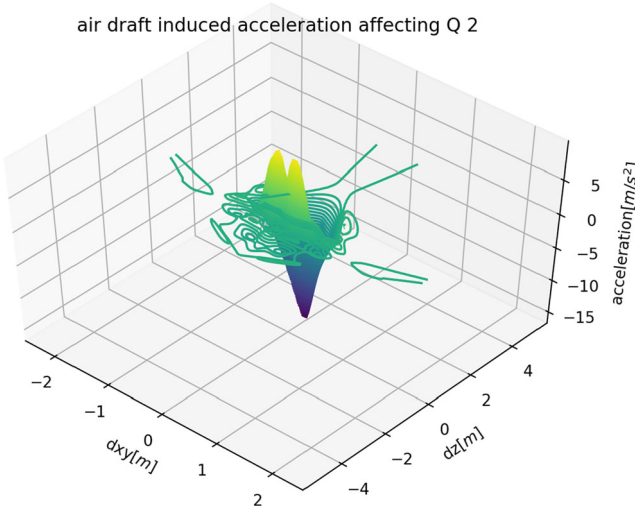

Fig. 9 The additional acceleration (scaled force) affecting Q2 while in proximity of Q1

4.1 Trajectory Generation with Interference Avoidance

We use a similar strategy as in [36] to generate minimum snap trajectories. Let $\xi^T = [\mathrm{r}^T, \psi]$ denote the vector of position and roll angle, and $\{(t^{(i)}, \xi^{(i)})\}_{i=1}^{m}$ be a set of time instants and associated way-points. The minimum snap trajectory is then the solution of the following optimization problem:

$$\min \int_{t_0}^{t_m} \mu_r \left\| \frac{d^{k_r}\mathrm{r}}{dt^{k_r}} \right\|^2 + \mu_\psi \left\| \frac{d^{k_\psi}\psi}{dt^{k_\psi}} \right\|^2 dt \tag{17}$$

$$\begin{aligned} \text{subject to:} \quad & \xi(t_i) = \xi^{(i)}, \ i \in \{1, \ldots, m\}, \\ & \left.\frac{d^p \mathrm{r}}{dt^p}\right|_{t=t^{(i)}} = 0, \ p \in \{1, \ldots, k_r\}, \\ & \left.\frac{d^p \psi}{dt^p}\right|_{t=t^{(i)}} = 0, \ p \in \{1, \ldots, k_\psi\}, \end{aligned}$$

where μ_r and μ_ψ are constants that makes the integrand nondimensional and where $k_r = 4$, $k_\psi = 2$ are typical choices.

To account for Q2Q interaction, we formulate a modified optimization problem that jointly computes minimum snap trajectories for two quadcopters and avoids the interference area $\mathcal{I}_{\varepsilon,d}$. As such we have

$$\min \sum_{j=1}^{2} \int_{t_0}^{t_m} \mu_r \left\| \frac{d^{k_r}\mathrm{r}_j}{dt^{k_r}} \right\|^2 + \mu_\psi \left\| \frac{d^{k_\psi}\psi_j}{dt^{k_\psi}} \right\|^2 dt \tag{18}$$

$$\begin{aligned} \text{s. t.:} \quad & \xi_j(t_i) = \xi_j^{(i)}, \ i \in \{1, \ldots, m\}, \\ & \left.\frac{d^p \mathrm{r}_j}{dt^p}\right|_{t=t^{(i)}} = 0, \ \forall i, j, p \in \{1, \ldots, k_r\}, \\ & \left.\frac{d^p \psi_j}{dt^p}\right|_{t=t^{(i)}} = 0, \ \forall i, j, p \in \{1, \ldots, k_\psi\}, \\ & \mathbb{1}_{\{\Delta x(t)^2 + \Delta y(t)^2 \le \varepsilon^2\}} \left(\Delta z(t)^2 - d^2\right) \ge 0, \forall t \in [t_0, t_m], \end{aligned}$$

where index j corresponds to the j^{th} quadcopter, the relative distance between the two quadcopters is defined by $\Delta \mathrm{r} = [\Delta x, \Delta y, \Delta z]$, and $\mathbb{1}$ is the indicator function. The last two constraints ensure that the generated trajectories are outside of the interference areas. In [36] the trajectories are represented as piecewise polynomials.

We take advantage of computational libraries featuring automatic differentiation such as Jax [10] to automatically compute time derivatives of the trajectories and choose neural networks as representations for the trajectories. In other words, we choose $\mathrm{r}(t) = \mathrm{r}(t;\beta)$ to be a neural network that takes as input the time t and generate a three dimensional output representing the position over time, i.e., $(x, y, z)|_t = f_{\texttt{ANN},\texttt{traj}}(t)$. The advantage of this type of parameterization is that we can impose the time derivative constraints with ease since we compute them automatically. We approximate the integral in the objective function using the trapezoidal rule and a finer time discretization. The same fine time discretization is used to evaluate the interference constraints. In addition, we again use Jax to compute the gradients/Jacobian matrices of the objective and constraint functions and pass them to the optimization algorithm. We use sequential quadratic programming to solve the optimization problem (18). Since Jax enables batch evaluation of functions and their gradients, we can afford to choose a time discretization for computing the integral and the interference constraints that introduce small numerical error due to the numerical approximation.

We solved the optimization problems for trajectory generation under the cases with and without "interference avoidance" constraints. We considered the time interval [0, 50] sec, with the following time instants and way-points tuples:
Q1: {(0, [−1, 0, 0, 0]), (20, [0, 0, 2.5, 0]), (50, [1, 0, 0, 0])}, and
Q2: {(0, [0, −1, 0, 0]), (30, [0, 0, 2.0, 0]), (50, [0, 1, 0, 0])}, corresponding to quadcopters 1 and 2, respectively. We parameterized the quadcopter trajectories as neural networks with one hidden layer of size 30 and `tanh` as activation function. The learning algorithm details are shown in Table 5.

Figure 10 show the quadcopter minimum snap trajectories and their relative distance when the interference zone constraints are excluded and included in the optimization problem, respectively. The effects of the interference zone constraints are obvious: the relative distance does not intersect the interference area. Figure 11 shows the quadcopters' z-axis component of the trajectories for the cases where the minimum-snap trajectories are generated without (Fig. 11a–b) and with (Fig. 11c–d) accounting for the interference area; in all the subfigures, the ideal reference z-axis trajectory component is shown in green. The simulation models include the closed-loop PD controller and the interference models learned in Section 3. The root mean square errors (RMS) between the quadcopter z-axis trajectories and the reference trajectories for Q1 and Q2 are respectively $\mathrm{RMS}_1 = 0.2002$ and $\mathrm{RMS}_2 = 0.2380$ when the interference zone is not accounted for. When it is accounted for, this RMS value for Q1 and Q2 decreases to $\mathrm{RMS}_1 = 0.1118$ and $\mathrm{RMS}_2 = 0.1381$, respectively. Hence, from these results and Fig. 11 it can be observed that we have reduced the z-axis deviations of the trajectories by roughly 50% when accounting for interference at the trajectory planning stage.

Table 5 Minimum snap trajectories learning: Neural network and learning settings

Algorithm	Sequential Least Squares Programming
xtol	10^{-8}
ftol	10^{-8}
Library	Python/Scipy

4.2 Online Interference Area Avoidance

Here, we develop a (computationally) more tractable approach to trajectory generation, amenable to online and onboard deployment. This method uses a potential function approach to modify the trajectory based on the practical assumption that we have pre-computed reference trajectories that are at least twice continuously differentiable. Our objective is to generate new trajectories that track as close as possible the original trajectories while avoiding the interference area. Let $\tilde{\mathrm{r}}$ denote the modified trajectory. To ensure that we track the original trajectory we must have $\lim_{t\to\infty} \boldsymbol{\epsilon} = 0$, where $\boldsymbol{\epsilon} = \tilde{\mathrm{r}} - \mathrm{r}$. The differential equation $\ddot{\boldsymbol{\epsilon}} + k_d\dot{\boldsymbol{\epsilon}} + k_p\boldsymbol{\epsilon} = 0$ has zero steady state solution if you choose for example k_d and k_p as diagonal matrices with positive entries. To make sure that the modified trajectories avoid the interference area, we add a potential function that repels the quadcopters as they get close to each other. The repulsion force is activated once the quadcopters approach the interference zone. Intuitively, we would like to model a spring that is active within the interference area only. To this end, we define a Gaussian spring force function,

$$F_s(\Delta\mathrm{r}) = e^{-\frac{\Delta x^2+\Delta y^2}{2(\alpha\varepsilon^2)}} e^{-\frac{\Delta z^2}{2(\beta\varepsilon^2)}} \boldsymbol{K}_s \Delta\mathrm{r}, \text{ where } \Delta\mathrm{r} = \mathrm{r}_1 - \mathrm{r}_2 \tag{19}$$

Here, $\boldsymbol{K}_s$ is a diagonal matrix with positive entries, and α, β are positive scalars controlling the width of the Gaussian like function. When the two trajectories are far from the interference zone, the Gaussian functions are very small, and hence no repulsion force is generated. We can choose the magnitude of $\boldsymbol{K}_s$ entries large enough so that the quadcopters never enter the interference areas. For the two quadcopters scenario, we have the following dynamical equations that ensure tracking and proximity avoidance:

$$\ddot{\tilde{\mathrm{r}}}_1 = \ddot{\mathrm{r}}_1 - k_d(\dot{\tilde{\mathrm{r}}}_1 - \dot{\mathrm{r}}_1) - k_p(\tilde{\mathrm{r}}_1 - \mathrm{r}_1) + F_s(\Delta\tilde{\mathrm{r}}), \tag{20}$$

$$\ddot{\tilde{\mathrm{r}}}_2 = \ddot{\mathrm{r}}_2 - k_d(\dot{\tilde{\mathrm{r}}}_2 - \dot{\mathrm{r}}_2) - k_p(\tilde{\mathrm{r}}_1 - \mathrm{r}_1) - F_s(\Delta\tilde{\mathrm{r}}), \tag{21}$$

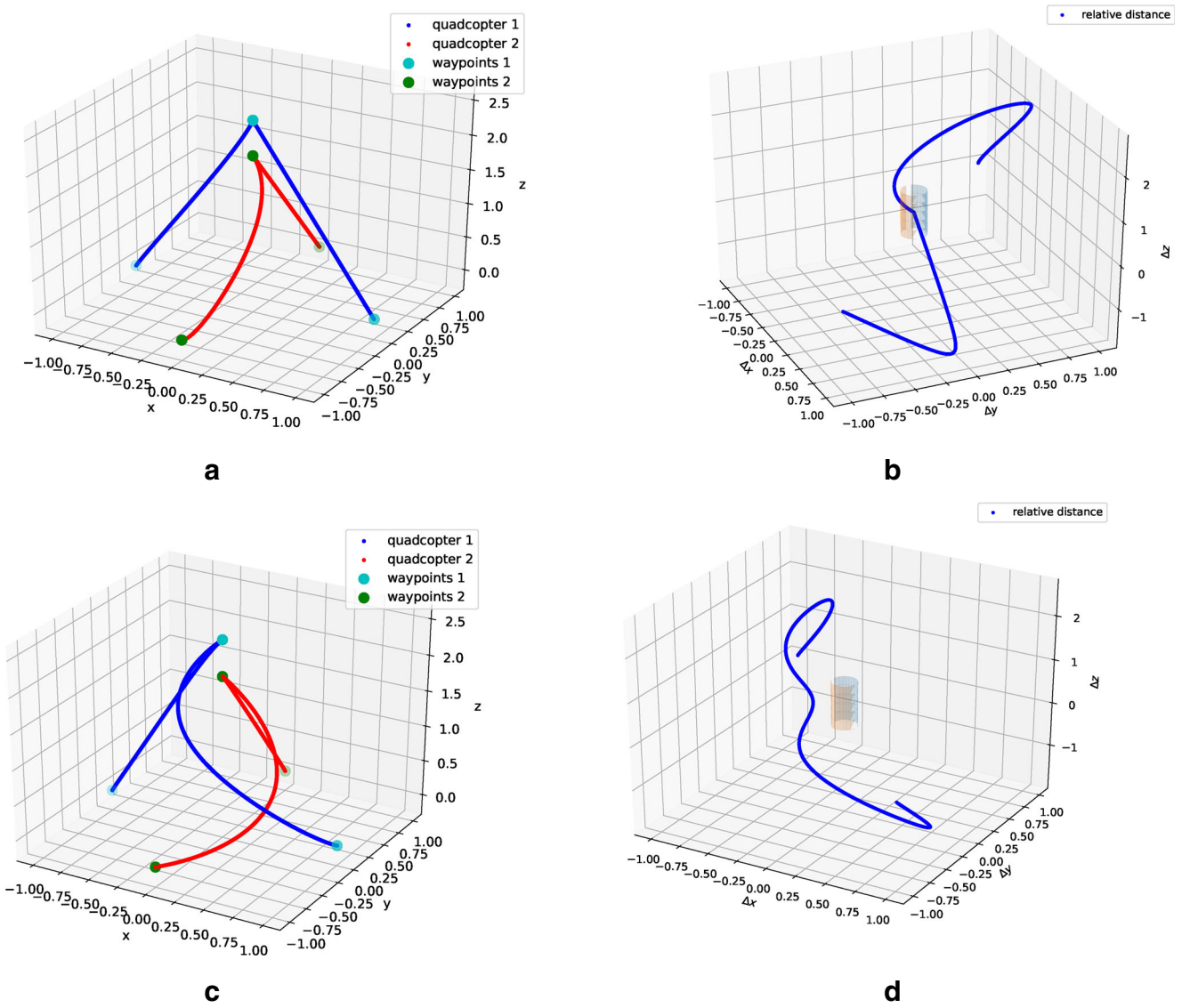

Fig. 10 Trajectory generation considering future time horizon: Without and with avoidance of the interference zone. **a** Without Interference Avoidance: Minimum snap trajectories for the two quadcopters w.r.t. the interference zone. **b** Without Interference Avoidance: Relative distance between the two quadcopters. **c** With Interference Avoidance: Minimum snap trajectories for the two quadcopters w.r.t. the interference zone. **d** With Interference Avoidance: Relative distance between the two quadcopters

where $\Delta\tilde{\mathrm{r}} = \tilde{\mathrm{r}}_1 - \tilde{\mathrm{r}}_2$. The above system of equations can be seen as a feedback control loop where the reference are trajectories that do not account for Q2Q interference, where the input is modeled as a force that is the gradient of a potential function.

We tested the online obstacle avoidance using the minimum snap approach summarized in Eq. 17 with the same time instance and way-points tuples as used in the previous section. The control parameters and the parameters of the repulsion force can be tuned to minimize the error between the original and modified trajectories while avoiding the interference zone. Figure 12a shows a comparison between minimum snap trajectories and trajectories modified using the online interference zone avoidance mechanism. We note that the trajectories do converge to the original reference trajectories. Figure 12b depicts the relative distance between quadcopters when using the online modified trajectories. It can be seen that the interference zone is successfully avoided. The parameters for the avoidance mechanism used in the simulation were $k_d = 3$, $k_p = 5$ and $(\boldsymbol{K}_s)_{i,i} = 10$, $i \in \{1, 2, 3\}$, $\alpha = 0.7$ $\beta = 0.5$, and $\varepsilon = 0.3$.

Figure 13 shows the trajectories generated using avoidance potential functions, and the response of the closed loop system that includes the Q2Q interaction model and the PD controller learned in Section 3. The root mean square errors (RMS) between the quadcopter z-axis trajectories and the reference trajectories for the two quadcopters are

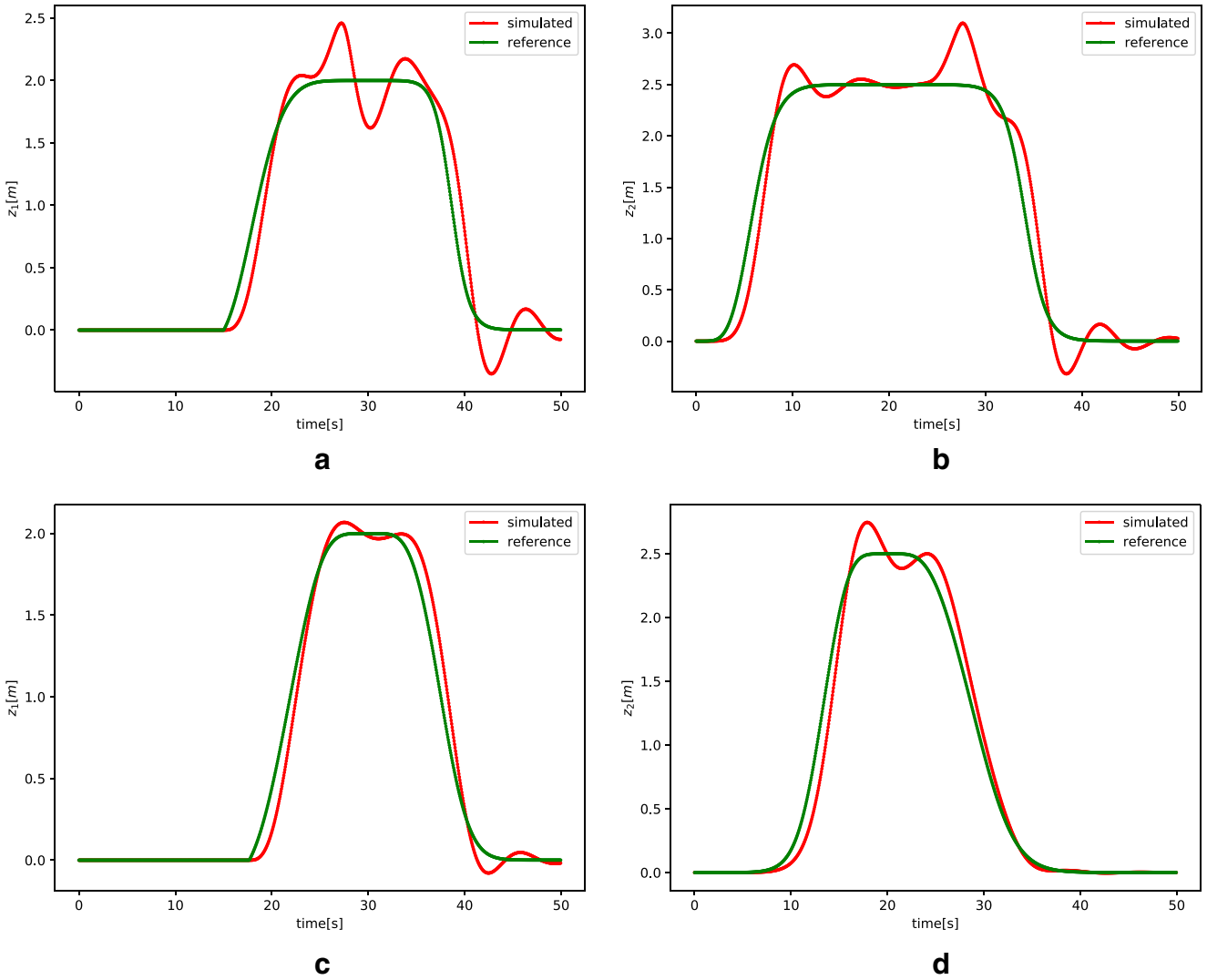

Fig. 11 Trajectory generation considering future time horizon: Simulated z-axis trajectories generated without and with the consideration of interference area constraints (quadcopter 1 is the lower UAV and quadcopter 2 is the upper UAV). **a** Without Interference Avoidance: Quadcopter 1 z-axis trajectory. **b** Without Interference Avoidance: Quadcopter 2 z-axis trajectory. **c** With Interference Avoidance: Quadcopter 1 z-axis trajectory. **d** ith Interference Avoidance: Quadcopter 2 z-axis trajectory

$\text{RMS}_1 = 0.1787$ and $\text{RMS}_2 = 0.2082$, respectively. Note that although the results are not as attractive as those in the case of the forward looking re-planning approach presented earlier, noticeable reduction in tracking errors was still accomplished, in comparison to the case where the interference area is not accounted for.

5 Q2Q Interaction Aware Control Design

In this section we discuss control strategies to cope with air disturbances affecting quadcopters when operating in the proximity of other quadcopters, and where a trajectory deviation is not desirable. We assume that neighboring quadcopters are aware of their neighbors positions so that they can estimate their influence. We present two control strategies: a PID based control and a MPC based control. The first strategy is commonly used in practice, mainly because it is easy to implement and is computationally efficient. The MPC approach, although computationally expensive, has the advantage of looking forward in time, as well as reducing potential overshoots. Hence, it can find strategies that not only track reference trajectories, but can better account for future Q2Q interference effects. These two control approaches mitigate the effect of interference induced disturbance.

5.1 PID-based Control

The typical PID-quadcopter control approach is based on three inter-dependent controllers: altitude control, position control and attitude control. Altitude control computes input

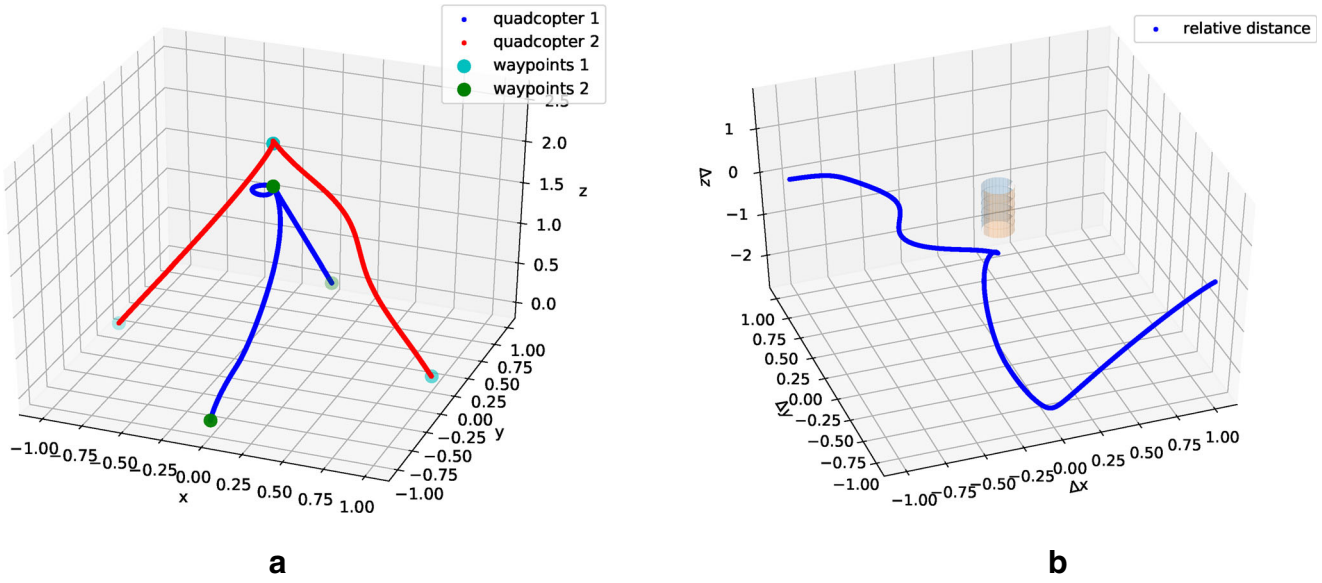

Fig. 12 Online trajectory modification with potential function to avoid the interference zone. **a** Online re-planned trajectories for the two quadcopters. **b** Relative distance between quadcopters w/ replanned trajectory

U_1 such that the variable z representing the quadcopter altitude tracks a desired reference. The position control is done indirectly through the quadcopter orientation, namely the attitude angles ϕ, θ and ψ. The desired (x, y) position is achieved by controlling the quantities u_x and u_y such that (x, y) track a desired position reference. The position control determines the desired angles ϕ_d, θ_d and ψ_d that are passed to the attitude controllers. Note that the position dynamics is not feedback linearlizable, and hence a linear version based on the small angle approximations is used to compute the desired angles. The attitude control determines control inputs U_2, U_3 and U_4 such that the angles track the angles ϕ_d, θ_d and ψ_d needed by the position controller. We describe the basics of the altitude and position control. The attitude control approach is similar to the altitude control, and thus it is omitted.

To control the quadcopter altitude, first the z dynamics are linearized by choosing $U_1 = \frac{m}{\cos\phi\cos\theta}(g + v)$, where

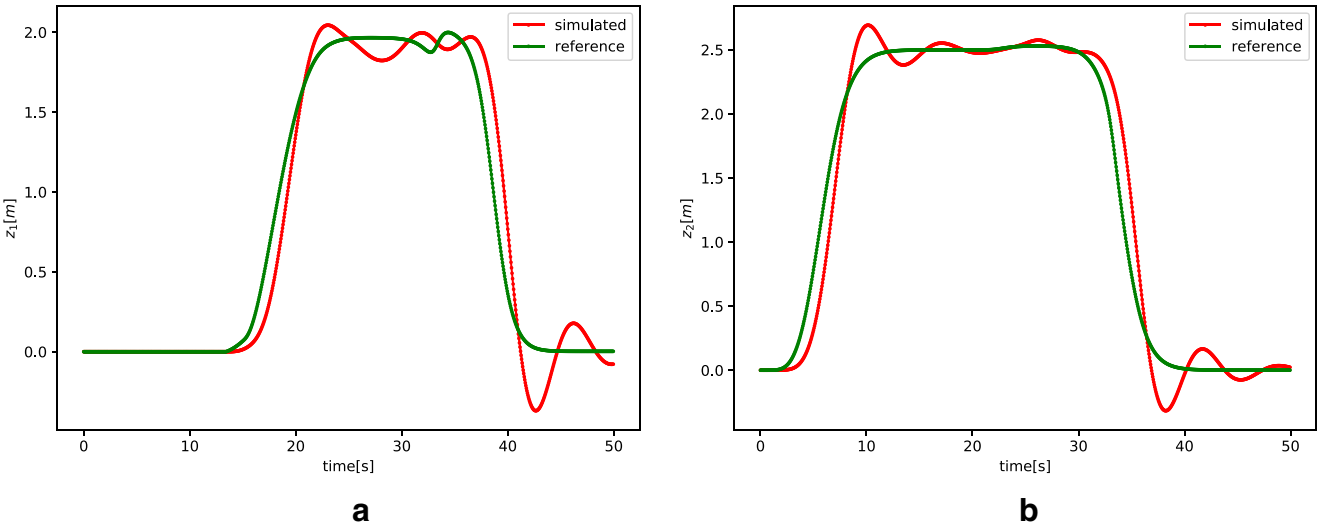

Fig. 13 Online trajectory modification with potential function: Simulated z-axis trajectories to avoid interference zone (quadcopter 1 is the lower UAV and quadcopter 2 is the upper UAV). **a** With Avoidance Potential Function: Quadcopter 1 z-axis trajectory. **b** With Avoidance Potential Function: Quadcopter 2 z-axis trajectory

v is the control input for a double integrator: $\ddot{z} = v$. Next v is chosen as the output of a PID controller leading to the following closed-loop dynamics:

$$\ddot{z} = \ddot{z}_r + K_p^z(z_r - z) + K_d^z(\dot{z}_r - \dot{z}) + K_i^z \xi^z \quad (22)$$

$$\dot{\xi}^z = z_r - z, \quad (23)$$

where K_p^z, K_d^z, K_i^z are tunable positive constants, and ξ^z represents the integration of the tracking error. For the position control, we need to select the orientation angles such that

$$u_x^d = \frac{m}{U_1}\left[\ddot{x}_r + K_p^x(x_r - x) + K_d^x(\dot{x}_r - \dot{x}) + K_i^x \xi^x\right], \quad (24)$$

$$u_y^d = \frac{m}{U_1}\left[\ddot{y}_r + K_p^y(y_r - y) + K_d^y(\dot{y}_r - \dot{y}) + K_i^y \xi^y\right] \quad (25)$$

where $\xi^x(t) = \int_{t_0}^{t}(x_r(\tau) - x(\tau))d\tau$ and $\xi^y(t) = \int_{t_0}^{t}(y_r(\tau) - y(\tau))d\tau$. After using the small angles approximation for the expressions of u_x and u_y and after solving a linear system of equations, we obtain the approximate desired angles $\theta^d = u_x^d \cos\psi + u_y^d \sin\psi$ and $\phi^d = u_x^d \sin\psi - u_y^d \cos\psi$. The desired angles are references for the attitude controllers.

When considering Q2Q interactions modeled as additional forces acting on the quadcopters, the changes to the PID controller are straightforward. In the case of attitude control, the input U_1 takes the form $U_1 = \frac{m}{\cos\phi\cos\theta}(g - \eta^z(\Delta \mathrm{r}) + v)$, where η^z is the air draft induced acceleration (scaled force) that depends on the relative distance Δr. In the case of position control, the changes follow the same idea, namely: $u_x^d = \frac{m}{U_1}\left[-\eta^x(\Delta \mathrm{r}) + \ddot{x}_r + K_p^x(x_r - x) + K_d^x(\dot{x}_r - \dot{x}) + K_i^x \xi^x\right]$, $u_y^d = \frac{m}{U_1}\left[-\eta^y(\Delta \mathrm{r}) + \ddot{y}_r + K_p^y(y_r - y) + K_d^y(\dot{y}_r - \dot{y}) + K_i^y \xi^y\right]$. When changing the controllers as described above, and not limiting the control magnitude, the interference effects are completely eliminated. In reality, the inputs are bounded by the physical limitations of the quadcopters, e.g., maximum thrust. To account for potential saturation, we limited the control U_1 based on the maximum magnitude attained when there is no interference, i.e., $|U_1| \leq 6.6$. Figure 14 show the simulated quadcopters altitudes (Fig. 14a and b) and altitude control inputs (Fig. 14c and d) for two quadcopter when accounting and disregarding the Q2Q interference in the control design. We used the PD controller learned from experimental data for the altitude control. In the presented scenario, Q1 remains hovering at a fix position (1.5 meters), and Q2 passes from right to left above Q1. As expected, Q1 altitude is affected by Q2 passing by. This phenomenon is more significant when not accounting for the Q2Q interference. To quantitatively emphasize the differences in behavior, we use as metrics the RMS between trajectories and the reference trajectories, i.e., $\sqrt{\frac{1}{T}\int_0^T (z(t) - z_{ref}(t))^2 dt}$, and the altitude control signal energy, i.e., $\int_0^T U_1(t)^2 dt$. The results are shown in Table 6.

We note that although the effects of the air-draft caused disturbance has been attenuated in Q1, it has not been eliminated completely, when control input saturation is considered. In the next section we show an MPC approach to control design that looks forward in time and can better account for the Q2Q interference even with limited control inputs.

5.2 Model Predictive Control

In this section we describe a model predictive control (MPC) approach for joint control of two interacting quadcopters. The interaction scenario is the same as in the previous section: Q1 hovers at a fixed altitude (1.5 m) while Q2 passes slowly above Q1, hovering for roughly 10 sec on top of Q1, at 2 meters. Q1 remains at $(0, 0)$ on the (x, y) plane, while Q2 remains at $x = 0$, varying its y position from $y = 2$ meters to $y = -2$ while passing Q1. Using a minimum snap approach we derive the reference trajectories for the two quadcopters. Our objective is to compute the control inputs so that the quadcopters follow their prescribed reference trajectories. Our objective is to show that we can do better with an MPC approach than the bounded PID controller.

MPC [17] is the standard approach for control of nonlinear systems and requires solving a nonlinear programming program with constraints. The optimization variables are the state trajectories and control inputs over time. The most important constraint is the system dynamics constraint that makes sure that the state trajectories respect the system dynamics. Other constraints include setting the initial and final values of the state variables, or limiting the magnitude of the states and control inputs.

The MPC formulation assumes a discretization of the time domain $\mathcal{T} = \{t_0, t_1, \ldots, t_N\}$ and the optimization variables are the state and control variables at discrete time instances, that is, $\mathrm{r}(t_k)$, $\boldsymbol{a}(t_k)$, $\dot{\mathrm{r}}(t_k)$, $\dot{\boldsymbol{a}}(t_k)$ and $\boldsymbol{U}(t_k)$, for $k \in \{0, 1, \ldots, N\}$. The key step in the formulation of the system dynamics constraint is the representation of the time derivatives of $\mathrm{r}(t_k)$ and $\boldsymbol{a}(t_k)$, namely their velocities and accelerations. Approaches based on trapezoidal collocation or Hermite-Simpson collocation [26] are typically used to transform the continuous dynamics into a discrete set of equality constraints. Such methods locally approximate the state and input trajectories while imposing equality constraints at the collocation points. We use a different approach to solve the MPC problem: a global representation of the state and input trajectories along with automatic differentiation to compute time derivatives. We choose neural networks to represent the orientation, position and inputs over time, that is $\boldsymbol{a} : \mathbb{R} \to \mathbb{R}^3$, $\mathrm{r} : \mathbb{R} \to \mathbb{R}^3$ and $\boldsymbol{U} : \mathbb{R} \to \mathbb{R}^4$. In particular r, $\boldsymbol{a}$ and $\boldsymbol{U}$ are ANNs with one hidden layer of size 30, `tanh` activation function,

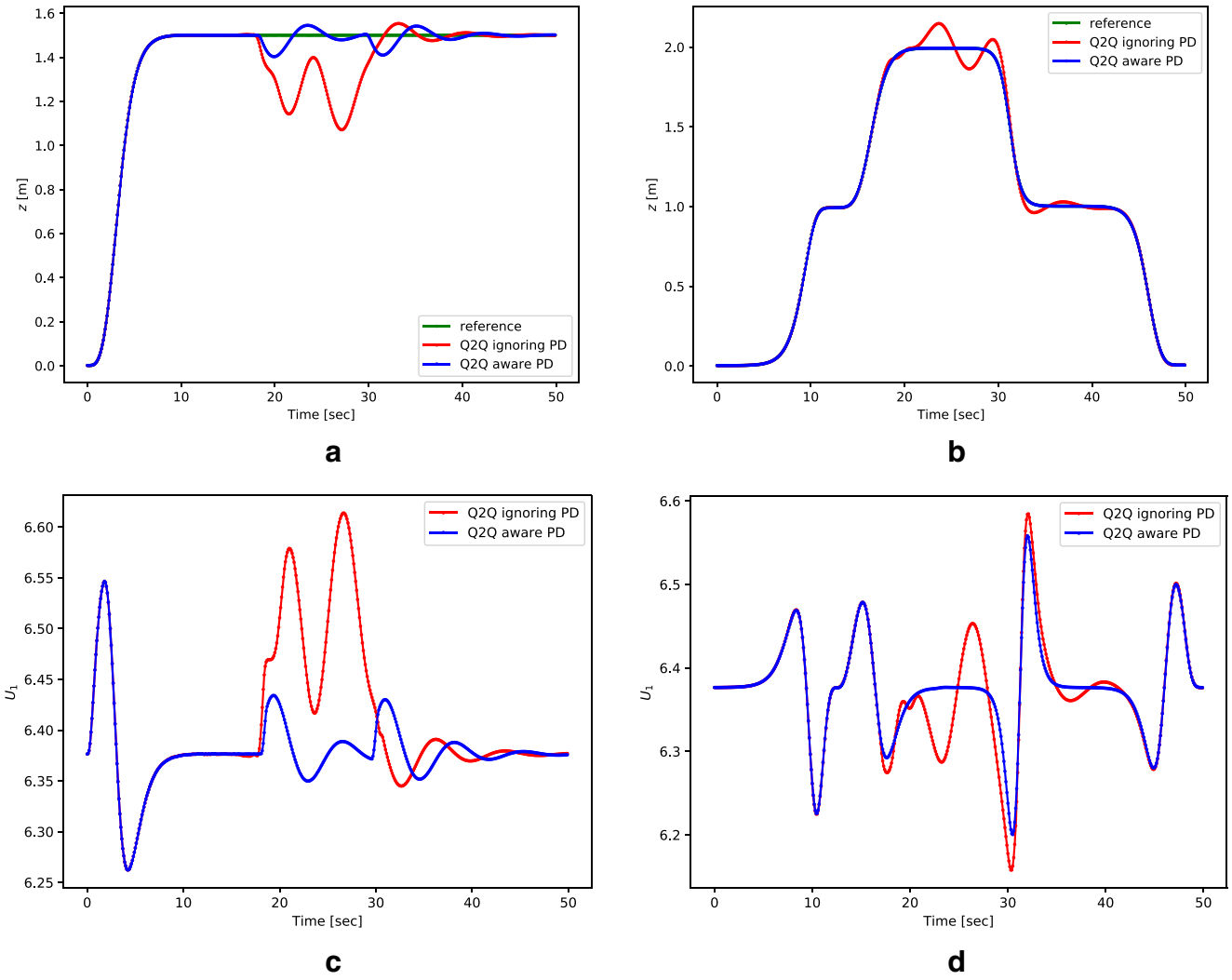

Fig. 14 PID controller design: z-axis trajectories and control inputs for the two quadcopters, with (Q2Q aware) and without (Q2Q ignoring) accounting for the Q2Q interference (quadcopter 1 is the lower UAV and quadcopter 2 is the upper UAV). **a** Quadcopter 1 z-axis reference and simulated trajectories. . **b** Quadcopter 2 z-axis reference and simulated trajectories. **c** Quadcopter 1 U1 control input. **d** Quadcopter 2 U1 control input

one input (the time) and respectively three, three and four dimensional outputs. We use Jax's automatic differentiation [10] feature to explicitly compute the time derivative of the state vector. This way, we can avoid the need for collocation-like methods. More importantly, Jax features batch evaluation of derivatives, that is we can evaluate simultaneously the derivatives for all time samples, avoiding this say the use of an inefficient `for loop`.

Table 6 Comparing the altitude of the quadcopters and corresponding control inputs when considering (Q2Q aware) and ignoring the Q2Q interference in the PD control design

Quantity	Q2Q aware	Q2Q ignoring
Q1 z-axis RMSE (m)	0.0280	0.1315
Q2 z-axis RMSE (m)	5.35×10^{-5}	0.0446
Q1 U_1 energy*	2035.09	2052.87
Q2 U_1 energy*	2033.33	2031.43

*Integration of squared signal over time

Let w denote the combined parameters of the state and control input representations. The constrained nonlinear program for the MPC formulation is then given by:

$$\min_{w} \quad \sum_{k=0}^{N}\sum_{j=1}^{2} \|\mathrm{r}_j(t_k; w) - \mathrm{r}_{j,\mathrm{ref}}(t_k)\|^2 \tag{26}$$

$$\begin{aligned}
\text{subject to} \quad & \ddot{\boldsymbol{a}}_j(t_k; w) = f_a(\boldsymbol{a}_j(t_k; w), \dot{\boldsymbol{a}}_j(t_k; w), \boldsymbol{U}_j(t_k; w)), \forall k, j \\
& \ddot{\mathrm{r}}_j(t_k; w) = f_r(\mathrm{r}_j(t_k; w), \boldsymbol{a}_j(t_k; w), \boldsymbol{U}_j(t_k; w)), \forall k, j \\
& \psi_j(t_k; w) = 0, \forall k, j \\
& |\boldsymbol{U}_j(t_k; w)| \leq U_{max}, \forall k, j, \\
& \boldsymbol{a}_j(t_0; w) = \boldsymbol{a}_{j,0}, \mathrm{r}_j(t_0; w) = \mathrm{r}_{j,0}, \forall j, \\
& \dot{\boldsymbol{a}}_j(t_0; w) = \dot{\boldsymbol{a}}_{j,0}, \dot{\mathrm{r}}_j(t_0; w) = \dot{\mathrm{r}}_{j,0}, \forall j,
\end{aligned}$$

where k and j are time and quadcopter indices, respectively, $\mathrm{r}_{j,\mathtt{ref}}$ are the minimum snap reference trajectories, $\boldsymbol{a}_{j,0}$, $\dot{\boldsymbol{a}}_{j,0}$, $\mathrm{r}_{j,0}$, $\dot{\mathrm{r}}_{j,0}$ are initial conditions for the orientations, angular velocities, position and linear velocities, respectively, and U_{max} is the vector of bounds on the control magnitudes. Note that due to the parameterization of the state and input, we are not dependent on a particular discretization scheme. In fact, during the optimization procedure, we can randomly select time instances from the time domain. In turn, this will reduce overfitting of the optimization solution to a particular time discretization scheme. The optimization solution parameterization has the same flavor as the PDE solution parameterizations studied in [4]. We prefer to use a constrained optimization formulation that strictly enforces the equality constraints to avoid the painful exercise of properly scaling the different components of the loss function in case we add the constraints (as regularizers) in the loss function.

We choose neural network representations for the state and control input for practical reasons: we enable batch executions to evaluate the state, state derivatives and input for all time samples. When choosing the optimization algorithm, we consider the availability of automatic differentiation to compute gradients and Jacobian matrices of the loss and constrain functions.In this context, sequential quadratic programming (SQP) algorithm [39] serves a suitable role as the solver for the nonlinear program to compute the MPC control inputs. This algorithm solves a sequence of quadratic, convex optimization problems with equality constraints. Automatic differentiation deals with the linearization difficulties that sometimes plague the SQP algorithm. The optimization problem (26) was solved in Python, using the SQP algorithm that is part of the SciPy optimization library. The gradients and Jacobians of the loss and constraint functions are generated using Jax to avoid noisy numerical approximations. When solving (26), we used 5000 time samples, with a time horizon of 50 sec. We chose the U_{max} upper bound such that $|U_{1,j}| \leq 6.6$, a bound similar to that in the PID case. The characteristics of the optimization algorithm are similar to the ones shown in Table 5. When the MPC problem is solved online at each iteration, we solve the implicit MPC. When implementing the MPC in real time rather than solving the MPC problem at each iteration, we solve a sequence of optimization problems offline and use their solution in real time; this is the explicit MPC formulation [8]. This case is where neural networks shine since we can take advantage of optimization solution based on GPU/TPU enabling us to solve a large number of optimization problem much faster than CPU-based implementations.

Figure 15 depict a comparison between the Q2Q aware MPC and Q2Q aware (limited) PD controllers for two interacting quadcopters. The quantitative results similar to the PID case are shown in Table 7. We note that the MPC approach generates more balanced trajectory tracking results. In addition, the control signals energy are slightly reduced as compared to the PID case. Interestingly, the MPC and PD based control signals share some similarities over the time horizon, except the close to the interference area.

It should not be a surprise that the control generated using the MPC approach fairs better than the PID controller, due to its forward looking characteristic. The results show the behavior when solving (26) once. In practice, (26) needs to be solved at each time sample during flight. Interestingly, Jax generated derivatives and gradients can be compiled into functions that can be connected to optimization libraries (e.g., NLOpt) built in C++. Therefore, the MPC optimization problem has the potential to be executed in real time on reasonable onboard platforms.

6 Comparison with State of the Art and Results Discussion

6.1 State of the art Comparison

Interaction modeling Physics-based interaction models have been previously proposed in the literature. They typically require costly experiments, specialized test platforms and sensing capabilities e.g., air speed probe [59], high-speed stereo particle image velocimetry [51]) or numerically intense Computational Fluid Dynamics (CFD) simulations [44, 49, 50, 60]. Closer to our problem setup, physics-based interaction models are discussed in [25], where the authors learn an interaction model that includes drag and change in propeller thrust due to oncoming flow. The parameters of such a model require low level hardware measurements (e.g., propeller thrust) and IMU measured attitudes. Our models require less output measurements, i.e., only position and do not use attitude measurements since their signal-to-noise ratio is too low to be of use. A neural network-based interaction model for quadrotors operating in close proximity is introduced in [48]. The authors use a neural network with four layer having as input the entire state vector, requiring accurate measurements of the position, angles, velocities and angular velocities. In addition, training such models required repeated experiments under different configuration scenarios. Our interaction model are also based on neural network, but use only two layers. Moreover, we use only position and velocity measurements and use less

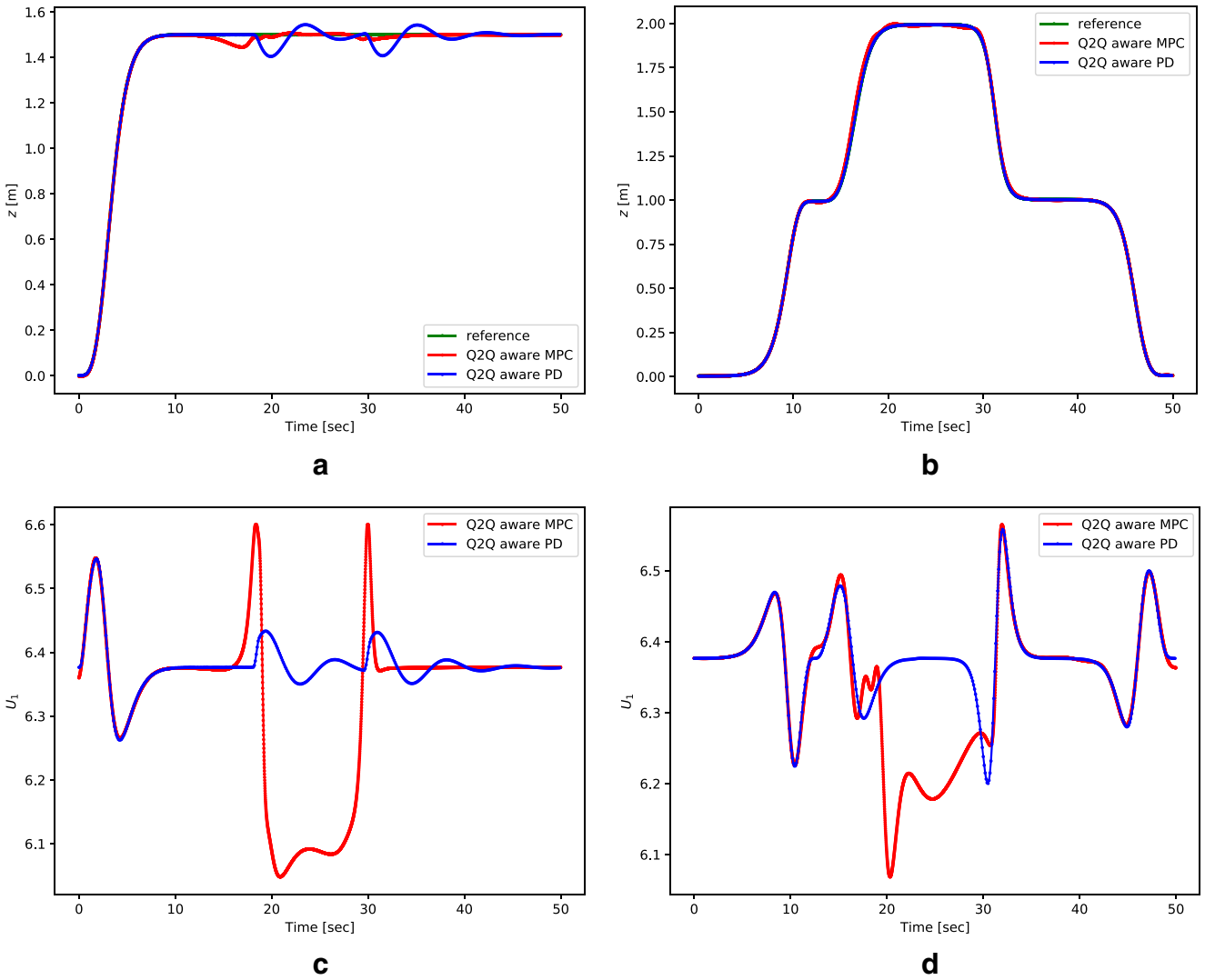

Fig. 15 MPC controller design: z-axis trajectories and control inputs for the two quadcopters, with (Q2Q aware) and without (Q2Q ignoring) accounting for the Q2Q interference (quadcopter 1 is the lower UAV and quadcopter 2 is the upper UAV). **a** Quadcopter 1 z-axis reference and simulated trajectories. **b** Quadcopter 2 z-axis reference and simulated trajectories. **c** Quadcopter 1 U1 control input. **d** Quadcopter 2 U1 control input

training data due to structural constraints imposed to the model, i.e., exponential functions that decay as the relative distances on the (x, y) plane and on the z-axis increase.

Table 7 Comparing the altitude of the quadcopters and corresponding control inputs generated by the Q2Q aware MPC and Q2Q aware (limited) PD controllers

Quantity	Q2Q aware MPC	Q2Q aware PD
Q1 z-axis RMSE (m)	0.01265	0.028
Q2 z-axis RMSE (m)	0.01264	5.35×10^{-5}
Q1 U_1 energy*	2003.071	2035.09
Q2 U_1 energy*	2011.99	2033.33

*Integration of squared signal over time

Trajectory re-planning Both trajectory re-planning approaches make use of the characterization of a cylinder shaped interaction area identified from experimental data. A trajectory planning strategy for quadrotor swarms is proposed in [23]. Similar to our approach, the authors use an interference area modeled through an ellipsoid whose parameters are determined from experiments. Their approach include a discrete planning step which are typically computationally expensive, especially in the case of a large number of quadrotors that generates a set of discrete waypoints. Obstacle avoiding continuous trajectories are generated by solving hard-margin support vector machine (SVM) problem that includes the ellipsoidal interference area, followed by a trajectory optimization problem that includes safe corridor constraint determined by the solution

of the hard-margin SVM problem. In our case, we use a minimum snap trajectory optimization problem that includes the interference area as constraints. Our formulation is centralized but can be extended to a distributed formulation using approaches described in [34, 35]. In the second strategy, we used a control algorithm approach to generated computationally cheap trajectories, where the disturbance is modeled as gradients as potential functions. Potential field control methods are popular for obstacle avoidance [28, 57]. In our case, we use it for trajectory re-planning rather than for direct control.

Controller re-design Due to its simplcity, the most common control algorithm used for quadcopter control is the PID controller [9]. We have re-purposed this type of controller and changed to account for the augmented quadcopter model that includes the Q2Q interference behavior. It has an inherent distributed implementation when considering a swarm of quadcopters since each quadcopter requires only local information, i.e., its own measurements and the measurements from its neighbors. It has some similarities with the quadcopter interaction aware control introduced in [48], although we remain closed to the original quadcopter PID control and provide explicit formulas for position control, as well as for attitude control.

MPC controller design MPC has been used for both single UAV control [5, 30, 46] and swarm of UAVs [14]. Unlike typical implementations based on collocation methods to convert the quadcopter dynamics to a set of equality constraints, we use global parameterizations for both the state vector and input. In addition, we use automatic differentiation to compute batch evaluation enabled state derivatives and gradients of the loss and constraint functions. Similar to the MPC literature, we use SQP algorithm to solve the constrained nonlinear program.

6.2 Discussion of the Results

In our approach we update the dynamical model of the quadcopter by adding the interference model and improve this way the control performance. Other possible approaches could be based on robust control [43] or fault tolerant control [56], where control algorithms deal with unknown parts of the system dynamics. Typically, we have a trade-off between control robustness and accuracy, and many such approaches are designed for particular classes of nonlinear systems.

In practical applications the main difficulty of our approach comes from the need of experimental data for training interference models. From the implementation practical implementation point of view, the quadcopters must be aware of the position of this neighbors. This can be achieved through a centralized system that tracks the quadcopter positions (e.g., by using the VICON system) and communicates them the quadcopters, or by enabling Q2Q communication. The PID controllers have low complexity, requiring a second order dynamical model for implementation, when considering the integral component and the approximation of the derivative part by a causal transfer function. Due to their linearity the PID controller can be exactly expressed in the discrete time domain, requiring to store the previous state of the controller. Implicit MPC is computationally more expensive, since it requires solving a constraint optimization problem at each time instant, based on sequential quadratic programming approached efficiently solved in real time using interior point methods [22, 45]. It is shown that it takes mili-seconds to compute the optimal control input at one time instant for a system with 12 states and inputs. In the case of box constraints, an MPC implementation based on the fast gradient method [37] can provide the optimal control input in micro-seconds (e.g., for a system with 6 states and 2 inputs). A computationally cheaper MPC implementation is the explicit MPC [8] that uses offline computations to determine all operating regions in which the optimal control moves. Explicit MPC controllers require fewer run-time computations than traditional (implicit) model predictive controllers. Further complexity reduction can be achieved by limiting the MPC approach to the position only, as in [46], while keeping the PID algorithm for attitude control. Given such evidence, we conclude that for the linearized quadcopter dynamics, optimal control inputs can be computed in mili-seconds, which is enough to ensure stability based on the dynamics time constants. The quality of the controller is a function of the accuracy of the model used for control design. In our quadcopter model based on [9], we determined the parameters based on the technical specification summarized in Table 1. In case specifications are insufficient, we can execute a system identification process as described in [46]. In the case the experimental data comes from experiments that reflect drone interactions, they can be treated as faults, and state and parameters estimation techniques in the presence of faults can be used to learn the system parameters [53, 55].

7 Conclusions

Quadcopters operating in close proximity affect each other through vertical airdrafts leading to unwanted trajectory deviations and loss of stability. We proposed strategies for coping with the effects of airdrafts that are applied based on the assumed operating conditions. Our overall methodology to tackle this problem provide the following principal features: i) the ability to model the inter-quadcopter

interference with sparse data by introducing physics-aware learning approaches, ii) the ability to modify trajectories (both offline and online) to avoid interference zones where viable, as well as iii) the ability to jointly compensate for the interference effects through modified control where trajectory deviations are not advisable due to application constraints. To further elaborate, our proposed approach is founded on the estimation of an interference area and of the additional forces acting on the quadcopters within this area. Modeling of these interference quantities were accomplished by combining data collected from UAV flight experiments with physics based models through the use of optimization and neural network modeling. Subsequently, to address situations where access to the quadcopter firmware is not possible and stability considerations are more important than trajectory deviations, we designed forward looking and myopic trajectory re-planning strategies that apply depending on the available computational resources. The forward looking strategy minimizes the difference between the original desired trajectories and the interference area-avoiding trajectories over some time horizon, but come at an increased computational cost. We achieved a reduction by 50% of the reference tracking error. The myopic strategy is suboptimal, while providing ease of real-time implementation. To consider situations when access to the quadcopter firmware is feasible and trajectory deviations are undesirable to maintain UAV functionality, we redesign the controllers by including the physics-infused interference model in the quadcopter dynamics. We show how the PID controllers need to be changed to accommodate the interference augmented model. In addition, we demonstrate an MPC approach that jointly design control strategies for two quadcopters operating in close proximity. Both control strategies are based on a physics-infused learning approach, where the original model is augmented with a physical interpretable data-driven model accounting for the interference effects. By accounting for the interference model in the control design we achieved 80% reduction of the RMSE with respect to tracking error and 20% reduction in control signal energy, that translates to longer flying time.

The proposed strategies were designed within the constraints of limited experimental data and unknown firmware for the quadcopter controllers. Further work will focus on additional experimental validation for the trajectory planning and control strategies through physical experiments. In particular we will study the feasibility of implementing automatic-differentiation enabled real-time constrained optimization algorithm for solving the MPC formulation, which will shed more light on the trade-offs between control and computational performance in real-time settings. Our recorded experimental data and interference model outcomes also provide insight on suitable settings for formation control implementations with UAVs (where such settings are conventionally made in a heuristic manner, as opposed to based on measurable evidence of the interference effects). Further aerodynamics (flow simulation) studies on such interference effects is expected to enlighten the applicability of the methods presented in the paper, and the usefulness of the physics infused neural network models that can use data both from experiments and high-fidelity simulations. Lastly, to more comprehensively address such airdraft interference issues in close coordinated UAV swarm applications, future work must investigate approaches to extend the models and planning/control mechanisms to handle simultaneous interactions of more than two quadcopters.

Acknowledgements This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) under Agreement No. HR0011-18-9-0037. Any opinions, findings, conclusions, or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the DARPA.

References

1. Crazyfly 2.0 design specifications: https://www.bitcraze.io/crazyflie-2/. Accessed 22 Jul 2019
2. 1, 000 drones decorate the sky to celebrate hit's 100th anniversary: https://news.cgtn.com/news/2020-06-08/1-000-drones-decorate-the-sky-to-celebrate-HIT-s-100th-anniversary-R9tddkPzgc/index.htm (2020)
3. Abaee Shoushtary, M., Hoseini Nasab, H., Fakhrzad, M.B.: Team robot motion planning in dynamics environments using a new hybrid algorithm (honey bee mating optimization-tabu list). Chinese J. Eng. 2014 (2014)
4. Al-Aradi, A., Correia, A., Naiff, D., Jardim, G., Saporito Y.: Solving nonlinear and high-dimensional partial differential equations via deep learning (2018)
5. Andrade, R., Raffo, G.V., Normey-Rico, J.E.: Model predictive control of a Tilt-Rotor Uav for load transportation. In: 2016 European Control Conference (ECC), pp. 2165–2170 (2016)
6. Behjat, A., Paul, S., Chowdhury, S.: Learning reciprocal actions for cooperative collision avoidance in quadrotor unmanned aerial vehicles. Robot. Auton. Syst. **121**, 103270 (2019)
7. Bekmezci, I., Sahingoz, O.K., Temel, Ş.: Flying ad-hoc networks (fanets): a survey. Ad Hoc Netw. **11**(3), 1254–1270 (2013)
8. Bemporad, A., Morari, M., Dua, V., Pistikopoulos, E.N.: The explicit linear quadratic regulator for constrained systems. Automatica **38**(1), 3–20 (2002). https://doi.org/10.1016/S0005-1098(01)00174-1. http://www.sciencedirect.com/science/article/pii/S0005109801001741
9. Bouabdallah, S., Siegwart, R.: Full control of a quadrotor. In: 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 153–158 (2007). https://doi.org/10.1109/IROS.2007.4399042
10. Bradbury, J., Frostig, R., Hawkins, P., Johnson, M.J., Leary, C., Maclaurin, D., Wanderman-Milne, S.: JAX: composable transformations of Python+NumPy programs. http://github.com/google/jax (2018)
11. Carrillo, J.A., Fornasier, M., Toscani, G., Vecil, F.: Particle, Kinetic, and Hydrodynamic Models of Swarming, pp. 297–336. Birkhäuser, Boston (2010)

12. Causa, F., Vetrella, A.R., Fasano, G., Accardo, D.: Multi-uav formation geometries for cooperative navigation in Gnss-challenging environments. In: 2018 IEEE/ION Position, Location and Navigation Symposium (PLANS), pp. 775–785 (2018)
13. Costa, F.G., Ueyama, J., Braun, T., Pessin, G., Osório, F.S., Vargas, P.A.: The use of unmanned aerial vehicles and wireless sensor network in agricultural applications. In: 2012 IEEE International Geoscience and Remote Sensing Symposium, IEEE, pp. 5045–5048 (2012)
14. Dentler, J., Rosalie, M., Danoy, G., Bouvry, P., Kannan, S., Olivares-Mendez, M.A., Voos, H.: Collision avoidance effects on the mobility of a uav swarm using chaotic ant colony with model predictive control. J.Intell. Robot. Syst. **93**(1-2), 227–243 (2019)
15. Finegan, F., Higgins, R., Nichols, F.: Runway acceptance rate improvements. In: 8Th Aerospace Sciences Meeting, p. 74 (1970)
16. Förster, J.: System Identification of the Crazyflie 2.0 Nano Quadrocopter. B.S. thesis, ETH Zurich (2015)
17. Garcia, C.E., Prett, D.M., Morari, M.: Model predictive control: Theory and practice - A survey. Automatica **25**(3), 335–348 (1989). http://www.sciencedirect.com/science/article/pii/0005109889900022
18. Giernacki, W., Skwierczyński, M., Witwicki, W., Wroński, P., Kozierski, P.: Crazyflie 2.0 Quadrotor as a platform for research and education in robotics and control engineering. In: 2017 22Nd International Conference on Methods and Models in Automation and Robotics (MMAR), pp. 37–42. IEEE (2017)
19. Giones, F., Brem, A.: From toys to tools: the co-evolution of technological and entrepreneurial developments in the drone industry. Bus. Horiz. **60**(6), 875–884 (2017)
20. Goldberg, D.E. Genetic Algorithms in Search, Optimization and Machine Learning, 1st edn. Addison-wesley Longman Publishing Co., Inc, USA (1989)
21. Griffiths, D.A.: A study of dual-rotor interference and ground effect using a Free-Vortex wake model. In: American Helicopter Society 58th Annual Forum, Montreal, Canada, June 11-13, 2002 (2002)
22. Hansson, A.: A primal-dual interior-point method for robust optimal control of linear discrete-time systems. IEEE Trans. Autom. Control **45**(9), 1639–1655 (2000)
23. Hönig, W., Preiss, J.A., Kumar, T.S., Sukhatme, G.S., Ayanian, N.: Trajectory planning for quadrotor swarms. IEEE Trans. Robot. **34**(4), 856–869 (2018)
24. Huang, H., Hoffmann, G.M., Waslander, S.L., Tomlin, C.J.: Aerodynamics and control of autonomous Quadrotor helicopters in aggressive maneuvering. In: 2009 IEEE International Conference on Robotics and Automation, pp. 3277–3282. IEEE (2009)
25. Jain, K.P., Fortmuller, T., Byun, J., Mäkiharju, S.A., Mueller, M.W.: Modeling of aerodynamic disturbances for proximity flight of multirotors. In: 2019 International Conference on Unmanned Aircraft Systems (ICUAS), pp. 1261–1269. IEEE (2019)
26. Kelly, M.: An introduction to trajectory optimization: How to do your own direct collocation. SIAM Rev. **59**(4), 849–904 (2017). https://doi.org/10.1137/16M1062569
27. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proceedings of ICNN'95 - International Conference on Neural Networks, vol. 4, pp. 1942–1948 (1995)
28. Kim, J., Khosla, P.K.: Real-time obstacle avoidance using harmonic potential functions. IEEE Trans. Robot. Autom. **8**(3), 338–349 (1992)
29. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: arXiv:1412.6980. Cite arxiv:1412.6980comment: Published as a Conference Paper at the 3rd International Conference for Learning Representations San Diego, 2015 (2014)
30. Koo, S., Kim, S., Suk, J.: Model predictive control for uav automatic landing on moving carrier deck with heave motion. IFAC-PapersOnLine **48**(5), 59–64 (2015). https://doi.org/10.1016/j.ifacol.2015.06.464. http://www.sciencedirect.com/science/article/pii/S240589631500703X. 3rd IFAC Workshop on Multivehicle Systems
31. Kuriki, Y., Namerikawa, T.: Consensus-Based cooperative formation control with collision avoidance for a Multi-Uav system. In: 2014 American Control Conference, pp. 2077–2082. IEEE (2014)
32. Leese, G.W.: Helicopter downwash blast effects study. 3 US Army Engineer Waterways Experiment Station (1964)
33. Lei, Y., Bai, Y., Xu, Z.: Wind effect on aerodynamic optimization for non-planar rotor pairs using full-scale measurements. J. Intell. Robot. Syst. **87**(3-4), 615–626 (2017)
34. Matei, I., Baras, J.: Distributed algorithms for optimization problems with equality constraints. In: Decision and Control (CDC), 2013 IEEE 52nd Annual Conference On, pp. 2352–2357 (2013). https://doi.org/10.1109/CDC.2013.6760232
35. Matei, I., Baras, J., Nabi, M., Kurtoglu, T.: An extension of the method of multipliers for distributed nonlinear programming. In: Decision and Control (CDC), 2014 IEEE 53nd Annual Conference On, pp. 6951–6956 (2014)
36. Mellinger, D., Kumar, V.: Minimum snap trajectory generation and control for Quadrotors. In: 2011 IEEE International Conference on Robotics and Automation, pp. 2520–2525 (2011)
37. Morabito, B., Kögel, M., Bullinger, E., Pannocchia, G., Findeisen, R.: Simple and efficient moving horizon estimation based on the fast gradient method. IFAC-PapersOnLine **48**(23), 428–433 (2015). https://doi.org/10.1016/j.ifacol.2015.11.316. http://www.sciencedirect.com/science/article/pii/S2405896315026002. 5th IFAC Conference on Nonlinear Model Predictive Control NMPC 2015
38. Nelder, J.A., Mead, R.: A simplex method for function minimization. Comput. J. **7**, 308–313 (1965)
39. Nocedal, J., Wright, S.J. Numerical Optimization, 2nd edn. Springer, USA (2006)
40. Odonkor, P., Ball, Z., Chowdhury, S.: Distributed operation of collaborating unmanned aerial vehicles for time-sensitive oil spill mapping. Swarm Evol. Comput. **46**, 52–68 (2019)
41. Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., Lerer, A.: Automatic differentiation in pytorch (2017)
42. Patnaik, B., Wei, G.: Controlling wake turbulence. Phys. Rev. Lett. **88**(5), 054502 (2002)
43. Qu, Z., Jin, Y.F.: Robust control of nonlinear systems in the presence of unknown exogenous dynamics. vol. 3 pp. 2784–2790, https://doi.org/10.1109/.2001.980695 (2001)
44. Radhakrishnan, A., Schmitz, F.: An experimental investigation of a quad tilt rotor in ground effect. In: 21st AIAA Applied Aerodynamics Conference, p. 3517 (2003)
45. Rao, C.V., Wright, S.J., Rawlings, J.B.: Application of interior-point methods to model predictive control. J. Opt. Theory Appl. **99**, 723–757 (1998)
46. Sa, I., Kamel, M., Khanna, R., Popović, M., Nieto, J., Siegwart, R.: Dynamic System Identification, and Control for a Cost-Effective and Open-Source Multi-Rotor Mav. In: Hutter, M., Siegwart, R. (eds.) Field and Service Robotics, pp. 605-620. Springer International Publishing, Cham (2018)
47. Sartori, D., Yu, W.: Experimental characterization of a propulsion system for multi-rotor uavs. J. Intell. Robot. Syst. **96**(3-4), 529–540 (2019)
48. Shi, G., Hönig, W., Yue, Y., Chung, S.J.: Neural-swarm:, Decentralized close-proximity multirotor control using learned interactions. arXiv:2003.02992 (2020)

49. Shin, H.S., Antoniadis, A.F., Tsourdos, A.: Parametric study on formation flying effectiveness for a blended-wing uav. J. Intell. Robot. Syst. **93**(1-2), 179–191 (2019)
50. Shukla, D., Hiremath, N., Patel, S., Komerath, N.: Aerodynamic interactions study on low-re coaxial and quad-rotor configurations. In: ASME International Mechanical Engineering Congress and Exposition, vol. 58424, p. V007t09a023. American Society of Mechanical Engineers (2017)
51. Shukla, D., Komerath, N.: Multirotor drone aerodynamic interaction investigation. Drones **2**(4), 43 (2018)
52. Silano, G., Aucone, E., Iannelli, L.: Crazys: a software-in-the-loop platform for the Crazyflie 2.0 Nano-Quadcopter. In: 2018 26Th Mediterranean Conference on Control and Automation (MED), pp. 1–6. IEEE (2018)
53. Stojanovic, V., He, S., Zhang, B.: State and parameter joint estimation of linear stochastic systems in presence of faults and non-gaussian noises. International Journal of Robust and Nonlinear Control. https://doi.org/10.1002/rnc.5131 (2020)
54. Stojanovic, V., Nedic, N., Prsic, D., Dubonjic, L., Djordjevic, V.: Application of cuckoo search algorithm to constrained control problem of a parallel robot platform. Int. J. Adv. Manuf. Technol. **87** https://doi.org/10.1007/s00170-016-8627-z (2016)
55. Stojanovic, V., Prsic, D.: Robust identification for fault detection in the presence of non-gaussian noises: application to hydraulic servo drives. Nonlinear Dyn. **100** https://doi.org/10.1007/s11071-020-05616-4 (2020)
56. Sun, K., Liu, L., Qiu, J., Feng, G.: Fuzzy adaptive finite-time fault-tolerant control for strict-feedback nonlinear systems. IEEE Trans. Fuzzy Syst. 1–1 (2020)
57. Tazibt, C.Y., Achir, N., Muhlethaler, P., Djamah, T.: uav-based data gathering using an artificial potential fields approach. In: 2018 IEEE 88Th Vehicular Technology Conference (VTC-Fall), pp. 1–5 (2018)
58. Wang, Z., Spica, R., Schwager, M.: Game theoretic motion planning for multi-robot racing. In: Distributed Autonomous Robotic Systems, pp. 225–238. Springer, New York (2019)
59. Yeo, D., Shrestha, E., Paley, D.A., Atkins, E.M.: An empirical model of rotorcraft Uav Downwash for disturbance localization and avoidance. In: AIAA Atmospheric Flight Mechanics Conference, p. 1685 (2015)
60. Yoon, S., Lee, H.C., Pulliam, T.H.: Computational analysis of multi-rotor flows. In: 54Th AIAA Aerospace Sciences Meeting, p. 0812 (2016)

Dr. Ion Matei is a Principal Scientist with Intelligent Systems Laboratory, at Palo Alto Research Center (PARC). He received his doctoral degree in Electrical Engineering from the University of Maryland, College Park. Before joining PARC, Dr. Matei was a Research Associate with the National Institute for Standards and Technology (NIST). His research interests focus on optimization, machine learning and control. He has extensive experience in statistical and physics-based modeling in multiple physical domains.

Mr. Chen Zeng is a Ph.D. student in Mechanical and Aerospace Engineering at the University at Buffalo. He received his M.S. in Mechanical Engineering from University at Buffalo and his B.S. in Geophysics from Tongji University in China. His current research focuses on concurrent approaches for design and control of intelligent systems subject to uncertainties, with applications to UAVs and flexible space structures.

Dr. Souma Chowdhury is an Assistant Professor of Mechanical and Aerospace Engineering at the University at Buffalo. Previously, he was a research faculty at Mississippi State University. He received his Ph.D. in Mechanical Engineering in 2012 from Rensselaer Polytechnic Institute at Troy. His research interests lie at the interface of optimization, evolutionary computing and machine learning, with application to design and autonomy of UAVs, swarm robotic systems and cyber-physical systems.

Dr. Rahul Rai is Dean's Distinguished Professor in the Clemson University International Center for Automotive Research (CU-ICAR). He directs the Geometric Reasoning and Artificial Intelligence Lab (GRAIL) at Clemson University. He earned his doctoral degree in Mechanical Engineering from The University of Texas at Austin USA in 2006. By combining engineering innovations with methods from machine learning, AI, statistics and optimization, and geometric reasoning, his research strives to solve important problems in the cyber-physical system design and fabrication domains.

Dr. Johan de Kleer is a Research Fellow in the Intelligent Systems Laboratory, at Palo Alto Research Center (PARC). He co-invented the field of Qualitative Reasoning and continues to publish widely on topics including temporal reasoning, abstraction and modeling. He invented the Assumption-Based Truth Maintenance System, widely used in model-based diagnosis and constraint solvers. Dr. de Kleer invented the field of model-based diagnosis - most current model-based systems trace their roots to the General Diagnosis Engine invented at PARC.