Nonlinear MPC for Quadrotors in Close-Proximity Flight with Neural Network Downwash Prediction

Jinjie Li[1], Liang Han[2]*, Haoyang Yu[2], Yuheng Lin[2], Qingdong Li[1], Zhang Ren[1]

Abstract—Swarm aerial robots are required to maintain close proximity to successfully traverse narrow areas in cluttered environments. However, this movement is affected by the downwash effect generated from other quadrotors in the swarm. This aerodynamic effect is highly nonlinear and hard to describe through mathematical modeling. Additionally, the existence of the downwash disturbance can be predicted based on the states of neighboring quadrotors. If this prediction is considered, the control loop can proactively handle the disturbance, resulting in improved performance.

To address these challenges, we propose an approach that integrates a Neural network Downwash Predictor with Nonlinear Model Predictive Control (NDP-NMPC). The neural network is trained with spectral normalization to ensure robustness and safety in uncollected cases. The predicted disturbances are then incorporated into the optimization scheme in NMPC, which enforces constraints to ensure that states and inputs remain within safe limits. We also design a quadrotor system, identify its parameters, and implement the proposed method on board. Finally, we conduct a prediction experiment to validate the safety and effectiveness of the network. In addition, a real-time trajectory tracking experiment is performed with the entire system, demonstrating a 75.37% reduction in tracking error in height under the downwash effect.

Supplementary Material

We make the code and dataset available to the community at: `https://github.com/Li-Jinjie/ndp_nmpc_qd`.

I. Introduction

Advances in swarm robots have attracted significant attention since they can utilize cooperation and coordination among a group of robots [1] to achieve complex tasks that are impossible for a single robot. As one type of swarm robot, aerial swarm robots differ from ground robots in that each agent can be affected by the strong airflow generated from its upper neighbors, which is referred to as the downwash effect [2]. When an agent cannot accurately track its trajectory due to the airflow disturbances, it could influence other agents in normal flight (Fig. 1), thereby amplifying the
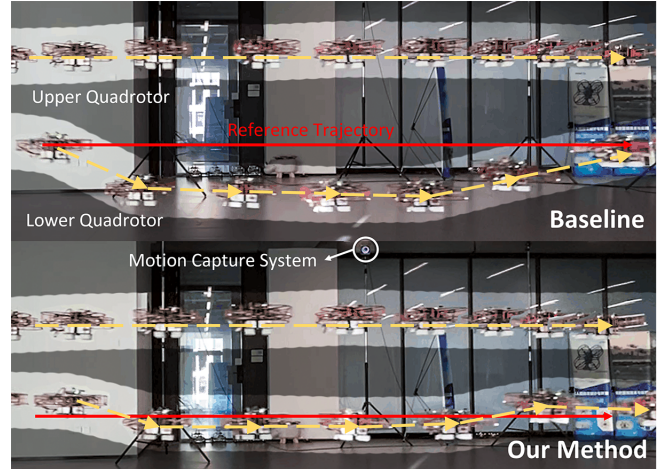

Fig. 1. Comparison of two methods in real-world close-proximity flight. In each scenario, two quadrotors fly together to track reference trajectories from left to right, with the lower quadrotor experiencing downwash effects from the upper drone.

disturbance until it affects the entire system. Thus, addressing the downwash effect is crucial for ensuring the flight safety of aerial swarm robots.

As the distances among aerial robots vary, the downwash effect exhibits a corresponding variation in strength, with greater intensity observed at shorter distances and weaker effects observed at longer distances. Hence, some researches [3], [4] treat the quadrotor as an ellipsoid with a long vertical axis, attempting to avoid close vertical flight when planning trajectories. However, this assumption reduces the *reachable set* of the aerial swarm robots, preventing pushing their mobility to the boundary. A better approach is to consider the downwash effect as a *disturbance rejection* problem and address it within the control layer [5]. In this way, the planning layer only needs to consider the collision radius of each agent and thus maximizes mobility. Therefore, it is necessary to model the downwash disturbance and incorporate it into the control loop.

The downwash effect is difficult to model by mathematical equations due to its high-degree nonlinearity. Traditionally, the airflow effect can be accurately modeled through real-world experiments [6], [7] or Computational Fluid Dynamics (CFD) simulation [8], while these approaches all pose high requirements for experimental equipment or computational time. On the other hand, the rapid development of deep learning in recent years renders the possibility to simulate nonlinear phenomena such as airflow disturbance in low time and fund demands [9]. Shi et al. conduct a series of pioneer

[1]J. Li, Q. Li, and Z. Ren are with the School of Automation Science and Electrical Engineering, Beihang University, Beijing, 100191, China `{lijinjie, liqingdong, renzhang}@buaa.edu.cn`

[2]L. Han (*Corresponding author), H. Yu, Y. Lin are with the Sino-French Engineer School, Beihang University, Beijing, 100191, China `{liang_han, haoyang_yu, linyuheng}@buaa.edu.cn`
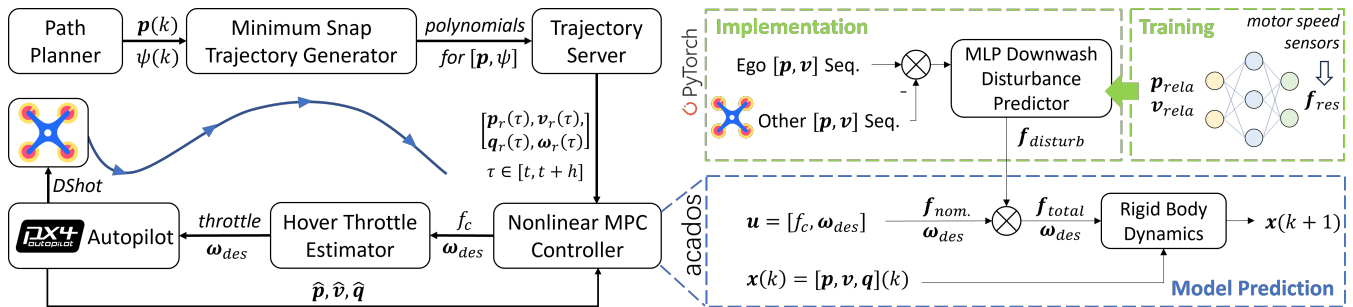
Fig. 2. The overall workflow of the proposed method, which follows the *Path Planning-Trajectory Generation-Control* pipeline. Specifically, Nonlinear Model Predictive Control (NMPC) is adopted for trajectory tracking, of which the model is a nominal quadrotor model assisted by a predicted disturbance sequence calculated before every iteration. These disturbances are predicted by a network using the error state sequence of its own and other quadrotors.

research to apply deep learning to model the airflow phenomena in aerial robotics, including the ground effect when the quadrotor lands [10], the downwash disturbances among quadrotors [5], and the disturbances in strong winds [11], which demonstrate the feasibility of modeling downwash effect using neural networks (NNs).

When integrating the NN model inside the control loop, the research [5] adopts a hierarchical feedback-linearization controller, which generates control inputs based on only the current states. However, the downwash effect is generated by the relative motions of other quadrotors and hence can be predicted through the exchange of reference trajectories. If the controller can fully exploit this prediction, the overshoots when tracking trajectories will be reduced, and thus the tracking performance is improved. As a prediction-based method, Model Predictive Control (MPC) can look forward and is a suitable solution to integrate the downwash prediction. Besides, MPC has the advantage of handling constraints [12], which can avoid the input saturation of quadrotors when resisting disturbances.

Numerous studies have attempted to combine deep networks with MPC and to assess their performance on a single quadrotor [13]–[15], while limited research tries to apply this combination to the downwash problem. Matei et al. [16] apply an MPC-based controller with a learning-driven interaction model to solve the downwash problem. Nevertheless, their approach uses a network as the MPC dynamics to accelerate computation, which is less accurate than a physics-based model and cannot be run onboard in real-time.

In this work, we design a trajectory tracking system for close-proximity flight by integrating a neural network disturbance predictor with Nonlinear Model Predictive Control (NMPC). The proposed approach is inspired by Shi et al. [5] and extends it to NMPC to fully exploit predictive power. First, using motor speed sensors and a physical model to collect downwash data, we train a Multi-Layer Perceptron (MLP) to predict disturbances and utilize *spectral normalization* to ensure robustness. Then, we integrate the predictor with NMPC to propose the trajectory tracking method. We also briefly introduce the trajectory generation algorithm to close the loop. Finally, we implement the proposed approach

on two quadrotors to verify its effectiveness.

The main contributions are as follows:

1) an NMPC-based trajectory tracking method with network disturbance prediction (NDP-NMPC) to exploit predicted movement information and to address the saturation constraints under close-proximity flight,
2) real-time experiments to verify trajectory tracking performance under downwash effects, and
3) provision of open-source code and a dataset to support further research in this area.

II. METHODOLOGY

This section outlines our proposed control scheme that combines a neural network disturbance observer with NMPC to enhance tracking performance during close-proximity flight. The overall workflow is presented in Section III-A. Then, subsections B and C introduce notation, coordinate systems, and a nominal quadrotor model. Leveraging these conventions, a neural network observer is implemented to predict the downwash disturbance in Section III-D. Finally, a modified NMPC trajectory tracking controller with disturbance prediction is proposed in Section III-E. This subsection also briefly introduces the generation of reference trajectories, which is essential to practical implementation.

A. System Overview

The system architecture is illustrated in Fig. 2. Moving from left to right, a sequence of position points and yaw angles is initially transmitted from a *Path Planner* to a *Trajectory Generator*. The latter module then leverages the minimum snap method [17] to generate a continuous polynomial trajectory from multiple derivatives of position and yaw angle. Subsequently, a *Trajectory Server* discretizes the trajectory and computes the desired full states through differential flatness [17]. These full-state points are transmitted as control reference to an *NMPC Controller* at a high frequency for computing control outputs. The control command, after being converted from force to throttle by a *Hover Throttle Estimator*, is ultimately executed by the *PX4 Autopilot* [18] to operate the quadrotor. The estimated states are fed back from the *Autopilot* to the *Controller* for closing the loop.

The downwash effect is taken into account within the NMPC controller. NMPC is a model-based control approach,
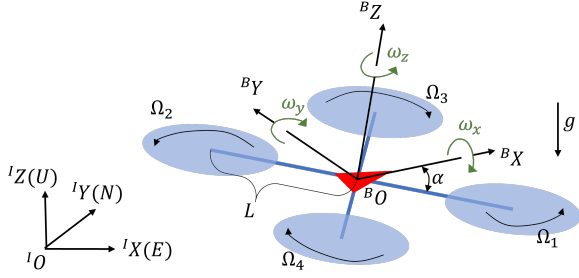
Fig. 3. Diagram of a quadrotor model with the ENU (X East, Y North, Z Up) inertial frame and FLU (X Forward, Y Left, Z Up) body frame.

and its model consists of two components: one based on a nominal quadrotor model and the other based on a neural network disturbance predictor. Prior to flight, the network is trained to predict disturbance forces based on relative states with nearby quadrotors. During flight, the sequence of disturbance predictions is employed by the controller to mitigate the downwash effect. By predicting the downwash effect using a neural network, the quadrotor can achieve more accurate trajectory-tracking performance during vertically-aligned flight.

B. Notation and Coordinate Systems

We denote scalars in lowercase $x \in \mathbb{R}$, vectors in bold lowercase $\boldsymbol{x} \in \mathbb{R}^n$, and matrices in bold uppercase $\boldsymbol{X} \in \mathbb{R}^{n \times m}$. We use $[\cdot]$ to denote arrays and $(\cdot)$ to denote functions. We use $\hat{\cdot}$ to denote estimated values. The coordinate systems, depicted in Fig. 3, contain the world inertial frame $\mathcal{I}$, the body frame $\mathcal{B}$, as well as the propeller numbering convention. The vector in the frame $\mathcal{I}$ is denoted as $^I\boldsymbol{p}$, and the rotation from $\mathcal{B}$ to $\mathcal{I}$ is denoted as $^I_B\boldsymbol{R}$ (rotation matrix) or $^I_B\boldsymbol{q}$ (attitude quaternion). We use the ENU inertial frame and FLU body frame to ensure compatibility with MAVROS, a toolkit for communication with autopilots like PX4 [18].

We use $\boldsymbol{q} = [q_w, q_x, q_y, q_z]^T \in \mathbb{H}$ to denote the attitude quaternion in *Hamilton-convention* [19], $\boldsymbol{q}^* = [q_w, -q_x, -q_y, -q_z]^T$ to denote the quaternion conjugation, and $\circ$ to denote the quaternion multiplication operator. The attitude quaternion is a unit quaternion ($\|\boldsymbol{q}\|$=1), and thus its inverse $\boldsymbol{q}^{-1}$ is the same as $\boldsymbol{q}^*$. We use $\mathcal{V}(\cdot)$ to represent the vector part of the quaternion $\mathcal{V}(\boldsymbol{q}) := [q_x, q_y, q_z]^T, \mathbb{H} \to \mathbb{R}^3$, and $\mathcal{V}^*(\cdot)$ to denote the reverse mapping from a position point $\mathcal{V}^*(\boldsymbol{p}) := [0, \boldsymbol{p}]^T, \mathbb{R}^3 \to \mathbb{H}$. Then full SE3 transformations from $\mathcal{B}$ to $\mathcal{I}$ can be represented as $^I\boldsymbol{p} = \mathcal{V}(^I_B\boldsymbol{q} \circ \mathcal{V}^*(^B\boldsymbol{p}) \circ ^I_B\boldsymbol{q}^*) + ^I\boldsymbol{p}_{Bo} = ^I_B\boldsymbol{R}(\boldsymbol{q})^B\boldsymbol{p} + ^I\boldsymbol{p}_{Bo}$, where $^I\boldsymbol{p}_{Bo}$ is the position of $\mathcal{B}$ frame's origin in the $\mathcal{I}$ frame, and $\boldsymbol{R}(\boldsymbol{q})$ is the rotation matrix from a quaternion following:

$$\boldsymbol{R} = \begin{bmatrix} 1 - 2q_y^2 - 2q_z^2 & 2q_xq_y - 2q_wq_z & 2q_xq_z + 2q_wq_y \\ 2q_xq_y + 2q_wq_z & 1 - 2q_x^2 - 2q_z^2 & 2q_yq_z - 2q_wq_x \\ 2q_xq_z - 2q_wq_y & 2q_yq_z + 2q_wq_x & 1 - 2q_x^2 - 2q_y^2 \end{bmatrix}.$$

C. Nominal Quadrotor Model

We assume that the origin of the body frame $\mathcal{B}$ is at the center of mass, and four rotors are all placed in the $\mathcal{B}$ frame's XY-plane. Established from 6-DoF rigid body dynamics, the

quadrotor model is written as follows [12]

$$^I\dot{\boldsymbol{p}} = {}^I\boldsymbol{v}, \tag{1}$$

$$^I\dot{\boldsymbol{v}} = \left(^I_B\boldsymbol{R}(\boldsymbol{q}) \cdot {}^B\boldsymbol{f}_u + {}^I\boldsymbol{f}_d\right)/m + {}^I\boldsymbol{g}, \tag{2}$$

$$^I_B\dot{\boldsymbol{q}} = 1/2 \cdot {}^I_B\boldsymbol{q} \circ \mathcal{V}^*(^B\boldsymbol{\omega}), \tag{3}$$

$$^B\dot{\boldsymbol{\omega}} = \boldsymbol{I}^{-1} \cdot \left(-^B\boldsymbol{\omega} \times \left(\boldsymbol{I} \cdot {}^B\boldsymbol{\omega}\right) + {}^B\boldsymbol{\tau}_u + {}^B\boldsymbol{\tau}_d\right), \tag{4}$$

where m is the mass, $^I\boldsymbol{g} = [0, 0, -g]^T$ is the gravity vector, $\boldsymbol{I} = \mathtt{diag}(I_{xx}, I_{yy}, I_{zz})$ is the inertia matrix assuming that quadrotors exhibit symmetry across all three axes, $^B\boldsymbol{f}_u$ and $^B\boldsymbol{\tau}_u$ are the force and torque caused by rotors, $^I\boldsymbol{f}_d$ and $^B\boldsymbol{\tau}_d$ are the force and torque caused by disturbances, and $^B\boldsymbol{\omega} = [\omega_x, \omega_y, \omega_z]^T$ is the angular rate expressed in the $\mathcal{B}$ frame.

The thrust generated by rotors is assumed to be vertical to the $\mathcal{B}$ frame's XY-plane, and we therefore obtain $^B\boldsymbol{f}_u = [0, 0, f_c]^T$ and $^B\boldsymbol{\tau}_u = [\tau_x, \tau_y, \tau_z]^T$, where f_c is the collective force of four rotors. We use a quadratic fit to model the thrust and torque for each propeller:

$$f_i = k_t \cdot \Omega^2, \quad \tau_i = k_q \cdot \Omega^2, \tag{5}$$

where k_t and k_q are the thrust coefficient and torque coefficient, respectively, as well as Ω represents motor speed in kRPM. Then $[f_c, \tau_x, \tau_y, \tau_z]^T$ and the thrust of each rotor f_i is connected by

$$[f_c, \tau_x, \tau_y, \tau_z]^T = \boldsymbol{G} \cdot [f_1, f_2, f_3, f_4]^T, \tag{6}$$

in which the control allocation matrix $\boldsymbol{G}$ is

$$\boldsymbol{G} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ -L\sin\alpha & L\sin\alpha & L\sin\alpha & -L\sin\alpha \\ -L\cos\alpha & L\cos\alpha & -L\cos\alpha & L\cos\alpha \\ -k_q/k_t & -k_q/k_t & k_q/k_t & k_q/k_t \end{bmatrix}, \tag{7}$$

where L and α are the geometric parameters as in Fig. 3.

The quadrotor's nominal model established above is utilized for designing both the disturbance observer and controller. The disturbance $^B\boldsymbol{\tau}_d$ is compensated by a high-frequency body-rate controller within the autopilot, and the $^I\boldsymbol{f}_d$ is estimated in the subsequent section.

D. Neural Network Observer for Downwash Effect

This part introduces a neural network observer to model the disturbance between quadrotors in close-proximity flight.

1) Neural Network Disturbance Observer: Considering the high nonlinearity of airflow, we employ a Multi-Layer Perceptron (MLP) to estimate the disturbances. A trained MLP can be viewed as a mapping function $\boldsymbol{y} = f(\boldsymbol{x}; \boldsymbol{\theta}) : \mathbb{R}^i \to \mathbb{R}^o$ from the input $\boldsymbol{x}$ to the output $\boldsymbol{y}$, where $\boldsymbol{\theta} := \{\boldsymbol{W}^1, b^1, \cdots, \boldsymbol{W}^{H+1}, b^{H+1}\}$ represent the weight and bias parameters, and H is the number of hidden layers. By choosing the element-wise ReLU $\phi(\boldsymbol{x}) = \max(0, \boldsymbol{x})$ as the activation function, the MLP network can be written as

$$f(\boldsymbol{x}; \boldsymbol{\theta}) = \boldsymbol{W}^{H+1} \cdot \phi\left(\cdots \phi\left(\boldsymbol{W}^1\boldsymbol{x} + b^1\right)\cdots\right) + b^{H+1}. \tag{8}$$

When applying the network to model the downwash effect, the input variables encompass the relative position and velocity of the ego quadrotor and the other one as

$\boldsymbol{x} = [^I\boldsymbol{p}_{rela}, ^I\boldsymbol{v}_{rela}]^T$, while the outputs encompass the disturbances as $\boldsymbol{y} = ^I\hat{\boldsymbol{f}}_d$.

2) Spectral Normalization: The training set we collected is impossible to cover the entire state space, and thus the output of the network in those data-uncovered states is critical to flight safety. Spectral normalization has been demonstrated in recent papers [10], [20] that it can enhance the robustness and generalization of neural networks and hence be adopted.

The Lipschitz constant of a function $\|f\|_{\text{Lip}}$ is defined as the smallest value such that

$$\forall \boldsymbol{x}, \boldsymbol{x}' : \|f(\boldsymbol{x}) - f(\boldsymbol{x}')\|_2 \, / \, \|\boldsymbol{x} - \boldsymbol{x}'\|_2 \leq \|f\|_{\text{Lip}}. \quad (9)$$

Let $l(\boldsymbol{x}) = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}$, and then we can get the Lipschitz norm of one layer in (8):

$$\frac{\|\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b} - \boldsymbol{W}\boldsymbol{x}' - \boldsymbol{b}\|_2}{\|\boldsymbol{x} - \boldsymbol{x}'\|_2} \leq \|\boldsymbol{W}\|_2 = \sigma(\boldsymbol{W}) = \|l\|_{\text{Lip}}, \quad (10)$$

where $\sigma(\cdot)$ represents spectral norm, i.e., the maximum singular value. Leveraging the Lipschitz constant inequality of composite functions $\|g_1 \circ g_2\|_{\text{Lip}} \leq \|g_1\|_{\text{Lip}} \cdot \|g_2\|_{\text{Lip}}$, and the fact that the Lipschitz constant of ReLU function $\|\phi(\boldsymbol{x})\|_{\text{Lip}} = 1$, we can obtain the bound of the MLP (8):

$$\|f(\boldsymbol{x})\|_{\text{Lip}} \leq \sigma(\boldsymbol{W}^{H+1}) \cdot 1 \cdots \sigma(\boldsymbol{W}) = \prod_{l=1}^{H+1} \sigma\left(\boldsymbol{W}^l\right). \quad (11)$$

In training phase, if every $\boldsymbol{W}^l$ is normalized by its spectral norm $\sigma(\cdot)$ and the scale ratio γ at each training epoch

$$\overline{\boldsymbol{W}}^l := \gamma \cdot \boldsymbol{W}^l / \sigma\left(\boldsymbol{W}^l\right), \quad (12)$$

then the Lipschitz norm of the network can be bounded as

$$\left\|f(\boldsymbol{x}; \overline{\boldsymbol{W}}^l)\right\|_{\text{Lip}} \leq \gamma^{H+1}. \quad (13)$$

Spectral normalization effectively limits the change rate of the network's output and leads to a more uniform output. This uniformity is further substantiated by our subsequent experiments.

3) Data Acquisition: Collecting the disturbance force is vital for training the network. Assuming that the physical model obtained through parameter identification is accurate, the disturbance force can be computed by subtracting the nominal force from the real one.

The nominal resultant force $^I\boldsymbol{f}_n$ can be calculated as follows assuming that the motor speed Ω_i is measured during the training phase:

$$^I\boldsymbol{f}_n = ^I_B\boldsymbol{R}(\boldsymbol{q}) \cdot ^B\boldsymbol{f}_u + m \cdot ^I\boldsymbol{g}, \quad (14)$$

where $^B\boldsymbol{f}_u$ comes from (5) and (6). In addition, the real resultant force $^I\boldsymbol{f}$ can be acquired using odometry. The odometry module of an autopilot offers velocity estimates, and its derivative $^I\dot{\boldsymbol{v}}$ is numerically calculated using Tustin's method. Subsequently, the real resultant force on the body is obtained through classical mechanics:

$$^I\boldsymbol{f} = m \cdot ^I\dot{\boldsymbol{v}}. \quad (15)$$

Finally, the disturbance $^I\boldsymbol{f}_d$ is determined by

$$^I\boldsymbol{f}_d = ^I\boldsymbol{f} - ^I\boldsymbol{f}_n. \quad (16)$$

Now the inputs $\left[^I\boldsymbol{p}_{rela}, ^I\boldsymbol{v}_{rela}\right]$ and outputs $^I\boldsymbol{f}_d$ have been constructed for training the neural network.

Remark: The resultant force can also be estimated directly using the inertial measurement unit (IMU), but the noise level is unacceptable. In addition, predictions for a sequence of states can be computed collectively in a single batch when utilizing the network in real flight.

E. Nonlinear MPC with Network Disturbance Prediction

This subsection first introduces the generation of the control target, i.e., the reference trajectory. Subsequently, the detailed presentation of the proposed NMPC-based control approach follows.

1) Reference Trajectory Generation: Trajectory generation involves the creation of a smooth, dynamically feasible, and time-indexed curve that traverses a set of points, including a start point, multiple predefined waypoints, and an end point. Leveraging the inherent mathematical property of *differential flatness* in quadrotors, the process of trajectory generation can be reformulated into a polynomial optimization problem for the four flat outputs $[x, y, z, \psi]$ corresponding to 3D position and yaw angle. To achieve this, we implement the *minimum snap* algorithm as outlined in [17], except that the time allocation among points is accomplished by straightforwardly dividing the relative distance by the user-defined average velocity.

The generated trajectory comprises a collection of parametric equations, which requires a trajectory server to discretize the curve. This server is also responsible for selecting the future reference states that align with the prediction horizon of NMPC, and then publishing these references. The publication frequency is set to be the same as the control frequency.

2) Nonlinear MPC: During close-proximity flight, we assume that each quadrotor employs an NMPC controller. This enables them to share predictions among themselves, ensuring the possibility of future disturbance predictions. Under above assumption, we select Nonlinear MPC for close-proximity flight due to two primary reasons. First, the ego quadrotor can prepare for the downwash effect in advance through the prediction trajectory of another quadrotor. Second, the potential saturation of the thrust command, which arises as one quadrotor experiences the downwash airflow, can be powerfully handled by NMPC.

Given the reference trajectory, the cost function is defined as the accumulated error between predicted and reference states over the time horizon. Then a constrained nonlinear optimization problem is formulated as

$$\min_{\boldsymbol{u}_k} \left(\overline{\boldsymbol{x}}_N^T \boldsymbol{Q}_N \overline{\boldsymbol{x}}_N + \sum_{k=0}^{N-1} \left(\overline{\boldsymbol{x}}_k^T \boldsymbol{Q} \overline{\boldsymbol{x}}_k + \overline{\boldsymbol{u}}_k^T \boldsymbol{R} \overline{\boldsymbol{u}}_k \right) \right) \quad (17)$$
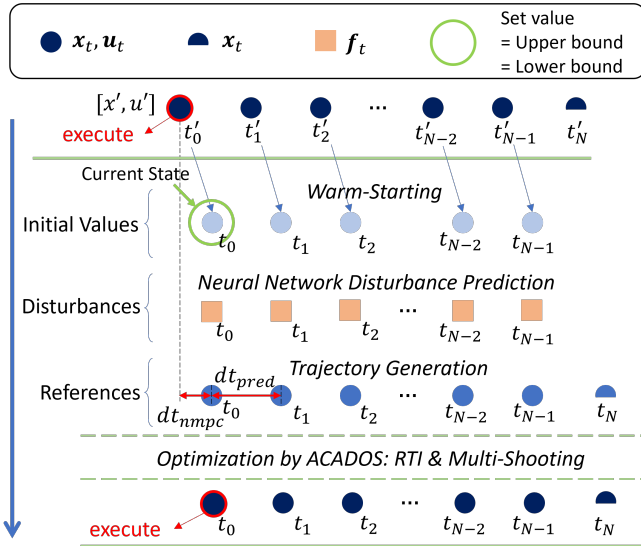
Fig. 4. A diagram illustrating the NDP-NMPC algorithm. From top to bottom, the initial value of each state point in this iteration is derived from the result of the previous iteration, a concept known as *warm-starting*. Then the first state is constrained to match the current state, introducing a feedback mechanism into NMPC. Next, the disturbances are predicted by a neural network and used as parameters for the optimizer, as well as the reference states are provided by the trajectory server. Finally, these data points are transmitted to ACADOS [22] for optimization.

with constraints of dynamics, initial values, and control inputs:

$$\begin{aligned} \boldsymbol{x}_{k+1} &= f\left(\boldsymbol{x}_k, \boldsymbol{u}_k\right), \\ \boldsymbol{x}_0 &= \boldsymbol{x}_{\text{init}}, \\ \boldsymbol{u}_k &\in \left[\boldsymbol{u}_{\min}, \boldsymbol{u}_{\max}\right], \end{aligned} \quad (18)$$

where the symbol $\overline{(\cdot)} = (\cdot) - (\cdot)_r$ denotes the error w.r.t. the reference, $\boldsymbol{x}_{\text{init}}$ is the current quadrotor state, the diagonal matrices $\boldsymbol{Q}_N, \boldsymbol{Q}, \boldsymbol{R}$ represent weights for terminal cost, state cost, and control energy cost, respectively, and $f(\cdot)$ is the quadrotor nominal model (1-3) discretized by the 4th-order *Runge-Kutta* method. Specifically, the state $\boldsymbol{x}_k$ equals to $[^I\boldsymbol{p}_k, {}^I\boldsymbol{v}_k, {}^I\boldsymbol{q}_k]^T$, and the control input $\boldsymbol{u}_k$ equals to $[f_c, {}^B\boldsymbol{\omega}]^T$. Note that (17) is a nonlinear least squares cost due to the nonlinearity introduced by quaternions. If the quaternion error is denoted as $\boldsymbol{q}_e = \boldsymbol{q} \circ \boldsymbol{q}_r^{-1}$, considering the fact that only three variables in a quaternion are independent, we can write the quaternion term in the cost function as
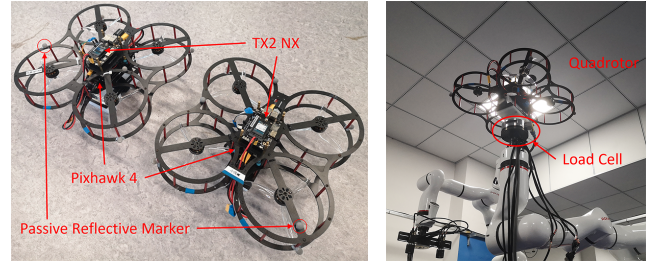
$$\overline{\boldsymbol{q}}_k^T \boldsymbol{Q}_q \overline{\boldsymbol{q}}_k = \|\text{sgn}(q_{ew}) \cdot \mathcal{V}(\boldsymbol{q}_e)\|_{\boldsymbol{Q}}^2 = \mathcal{V}(\boldsymbol{q}_e)^T \boldsymbol{Q}_q \mathcal{V}(\boldsymbol{q}_e), \quad (19)$$

where $\text{sgn}(\cdot)$ denotes the sign function. The sign term, which can avoid the unwinding phenomenon in quaternion-based control [21], is able to be eliminated in the quadratic cost.

Subsequently, *warm-starting*, *real-time iteration (RTI)*, and *multi-shooting* techniques [22] are applied to accelerate the NMPC computation, which is illustrated in Fig. 4. Finally, the control command is extracted from the optimized result sequence:

$$\boldsymbol{u}_{\text{NDP-NMPC}} = \boldsymbol{u}_0^* = \left[f_c', \omega_x', \omega_y', \omega_z'\right]. \quad (20)$$

Note that the thrust command f_c' needs to be normalized into $[0, 1]$ to align with the MAVROS toolkit. This normal-

(a) Flight platform

(b) Parameter identification

Fig. 5. (a) Two self-made quadrotors with onboard computing resources. (b) The quadrotor is identified for rotor parameters and inertial parameters.

TABLE I: IDENTIFIED PARAMETERS

Parameter(s)	Value(s)	Unit
L	0.1372	m
α	45	deg
m	1.5344	kg
g	9.81	m/s^2
I_{xx}	0.0094	kg $\cdot$ m^2
I_{yy}	0.0134	kg $\cdot$ m^2
I_{zz}	0.0145	kg $\cdot$ m^2
k_q	3.7611 E-4	N $\cdot$ m/kRPM^2
k_t	2.8158 E-2	N/kRPM^2
$[\Omega_{\min}, \Omega_{\max}]$	[2.6, 24.0]	kRPM
thrust/weight	4.3100	—
flight time	705	s

ization can be implemented either by a Kalman Filter to estimate the hover throttle, similar to the approach used in the PX4 autopilot [23], or through a calibration mapping that relates thrust to throttle.

III. EXPERIMENTS

This section introduces the experimental setup and analyzes the results. We begin by presenting the system identification of our hardware platform. Following that, we discuss the data collection related to the downwash effect. Next, we describe an open-loop experiment conducted for disturbance prediction. Finally, we perform a closed-loop trajectory tracking experiment to validate the control performance.

A. Parameter Identification for Quadrotors

We constructed our flight platform as depicted in Fig. 5a. It was noteworthy that the selected Electronic Speed Controllers (ESCs) supported rotor speed measurement through the DShot protocol[1]. Then, we leveraged a load cell as illustrated in Fig. 5b to identify the rotor parameters. Specifically, a 3D-printed connector was used to position the quadrotor on the load cell, and square wave signals were applied to drive a pair of propellers diagonally. This setup took into account the airflow effect of the body on the rotors. Assuming that the quadrotor is symmetrical in all three axes, we measured its inertial matrix using the bifilar-pendulum method [24]. The identified parameters listed in Table I were employed for both PX4 SITL simulation and control.
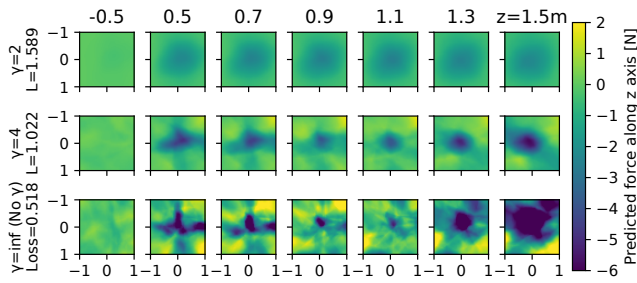
---

[1]https://docs.px4.io/main/en/peripherals/dshot.html

Fig. 6. Disturbance predictions of the neural network under different spectral normalization ratios γ. Each square represents the predicted Z force in a 1m×1m X-Y area. Given zero relative velocity, we change the γ to observe the output at different heights. The relative heights of the other quadrotor to the ego one are listed from left to right, where the negative number indicates flying above. The disturbance predictions for the X and Y axes are not displayed, as they are considered negligible when compared to the Z force.

B. Data Collection, Training, and Testing

To collect data points under the downwash effect, two quadrotors were operated by pilots to fly over each other. The lower quadrotor remained stationary while the upper quadrotor was moved to increase the likelihood of overlap. The height of the moving quadrotor was varied to diversify the data collection. Data from both quadrotors, including their states and estimated disturbances along with timestamps, were captured at a rate of 100Hz using the ROS tool `rosbag`. A total of 570-second data were collected for both quadrotors, and this data size can be doubled due to the similarity of drones. Then the data were bias eliminated by subtracting the average disturbance force in the hover state and time-aligned. Finally, the collected data were shuffled with a ratio of 0.75 for training and 0.25 for testing.

In training, the network was implemented in PyTorch with the parameters detailed in Table II. As previously mentioned, the input consisted of the 3-axis relative position and velocity, while the output included the 3-axis disturbance force. The network was trained with various spectral normalization ratios γ to determine the optimal value.

The spectral normalization ratio γ is critical to the balance of accuracy and robustness. We recorded the losses for different values of γ and plotted the predictions at different heights as Fig. 6 to evaluate performance in cases where no data are available.

From the figure, we can observe that a high value of γ (the third row) describes the disturbances well, while a low value of γ (the first row) is too conservative. In other words, the stronger the influence of spectral normalization, the lower the accuracy in fitting data. However, this low γ value results in safer predictions for points where no data have been collected. For the third row without spectral normalization,

TABLE II: SETTINGS FOR TRAINING THE NETWORK

Setting	Layers	Initialization	Activation Function
Value	6-128-64-128-3	normal	ReLU

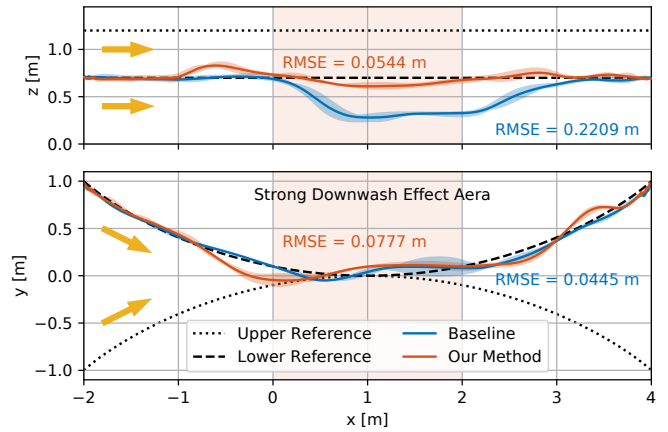| Optimizer	Epoch	Learning Rate	Loss Function
Adam	20,000	1e-4	Mean Squared Error

Fig. 7. Comparison of closed-loop tracking results between the baseline [25] and the proposed method. Each curve represents the average of three rounds, with the shaded area indicating the range of values. The shaded middle region highlights the zone with a strong downwash effect.

the predicted force suddenly increases at 1.3m due to the lack of data. Nevertheless, the first and second rows with Low γ mitigate this trend and guarantee flight safety. In conclusion, we decide to choose $\gamma = 4$ as the balance between data fitting and safety.

C. Trajectory Tracking under Downwash Effect

In this section, we aim to close the loop and verify the feasibility of the proposed method considering hardware limitations such as system latency and constrained computational resources for embedded platforms. The entire system was developed in Python and leveraged Robot Operating System (ROS) for communication. The system operated on both an onboard TX2 NX computer and a desktop PC. Specifically, the trajectory server, NDP-NMPC controller, and hover throttle estimator ran on the TX2 NX, while the PC was responsible for trajectory generation, broadcasting motion capture (MoCap) data, and serving as the ROS master node. The NMPC method was implemented using ACADOS with parameters listed in Table III. We assigned high weights to positions in order to minimize tracking errors.

We conducted quadrotor flights within a room measuring $7 \times 4 \times 3$ and used an OptiTrack MoCap system for localization. The flight trajectories were carefully designed to ensure an overlapping area. During the experiments, two quadrotors initiated from the same side but at different altitudes, and then executed back-and-forth movements. To evaluate the performance of closed-loop tracking, we conducted multiple runs, and the resulting trajectories are illustrated in Fig. 7. We established the NMPC method without disturbance prediction [25] as the baseline.

From Fig. 7, the baseline quadrotor is severely affected and thus deviates from the reference. In contrast, NDP-NMPC

TABLE III: CONTROL PARAMETERS

Parameter	Value	Parameter	Value	Parameter	Value
N	20	dt_{nmpc}	1/60s	dt_{pred}	0.1s
$Q_{p,\mathrm{xy}}$	300	$Q_{p,\mathrm{z}}$	400	$Q_{v,\mathrm{xyz}}$	1
$Q_{q,\mathrm{xyz}}$	0.1	$R_{\omega,\mathrm{xyz}}$	10	R_{f_c}	10

significantly mitigates the impact of disturbances, leading to a noteworthy 75.37% reduction in the tracking Root-Mean-Square Error (RMSE). This reduction emphasizes the positive impact of NDP-NMPC on close-proximity flight. Additionally, we observe that the lower quadrotor suddenly jumps before it enters the downwash area and then is pushed down to the reference height. This phenomenon is caused by the inaccuracy of neural network predictions. In other words, the neural network predicts a disturbance that does not exist in reality and has no corresponding counterbalancing force, which results in an unexpected ascent of the lower quadrotor. This prediction inaccuracy also introduces fluctuations in the horizontal plane as shown in Fig. 7. Therefore, when attempting to integrate predictions to enhance the system performance, it is critical to carefully weigh the potential implications.

IV. CONCLUSION

In this article, we proposed NDP-NMPC, a trajectory tracking method to alleviate the disturbance from downwash airflow in close-proximity flight. This approach utilized a neural network with spectral normalization to predict the disturbances caused by other quadrotors flying above. The network observer was then combined with an NMPC controller. To test the method, we trained a neural network and evaluated the performance of the disturbance prediction. Finally, we executed the algorithm on two quadrotors in real-time for trajectory tracking. We reported that the network was robust, and the proposed approach reduced 75.37% tracking error in the Z axis.

In the future, the proposed approach should be compared with non-prediction methods to better highlight the advantage of integrating predictions. Additionally, future directions will extend the proposed method to large-scale swarm drones and consider the uncertainty of predictions inside the NMPC workflow.

REFERENCES

[1] X. Zhou, X. Wen, Z. Wang, Y. Gao, H. Li, Q. Wang, T. Yang, H. Lu, Y. Cao, C. Xu, and Fei Gao, "Swarm of micro flying robots in the wild," *Science Robotics*, vol. 7, no. 66, pp. 1–17, May 2022.

[2] S.-J. Chung, A. A. Paranjape, P. Dames, S. Shen, and V. Kumar, "A Survey on Aerial Swarm Robotics," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 837–855, Aug. 2018.

[3] W. Hönig, J. A. Preiss, T. K. S. Kumar, G. S. Sukhatme, and N. Ayanian, "Trajectory Planning for Quadrotor Swarms," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 856–869, Aug. 2018.

[4] S. H. Arul and D. Manocha, "DCAD: Decentralized Collision Avoidance With Dynamics Constraints for Agile Quadrotor Swarms," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 1191–1198, Apr. 2020.

[5] G. Shi, W. Hönig, X. Shi, Y. Yue, and S.-J. Chung, "Neural-Swarm2: Planning and Control of Heterogeneous Multirotor Swarms Using Learned Interactions," *IEEE Transactions on Robotics*, vol. 38, no. 2, pp. 1063–1079, Apr. 2021.

[6] D. Yeo, E. Shrestha, D. A. Paley, and E. M. Atkins, "An Empirical Model of Rotorcrafy UAV Downwash for Disturbance Localization and Avoidance," in *Proceedings of AIAA Atmospheric Flight Mechanics Conference*, Jan. 2015, pp. 1–14.

[7] D. J. Carter, L. Bouchard, and D. B. Quinn, "Influence of the Ground, Ceiling, and Sidewall on Micro-Quadrotors," *AIAA Journal*, vol. 59, no. 4, pp. 1398–1405, Apr. 2021.

[8] A. E. Jimenez-Cano, P. J. Sanchez-Cuevas, P. Grau, A. Ollero, and G. Heredia, "Contact-Based Bridge Inspection Multirotors: Design, Modeling, and Control Considering the Ceiling Effect," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3561–3568, Oct. 2019.

[9] J. Willard, X. Jia, S. Xu, M. Steinbach, and V. Kumar, "Integrating scientific knowledge with machine learning for engineering and environmental systems," *ACM Computing Surveys*, vol. 55, no. 4, pp. 1–37, Nov. 2022.

[10] G. Shi, X. Shi, M. O'Connell, R. Yu, K. Azizzadenesheli, A. Anandkumar, Y. Yue, and S.-J. Chung, "Neural Lander: Stable Drone Landing Control Using Learned Dynamics," in *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, May 2019, pp. 9784–9790.

[11] M. O'Connell, G. Shi, X. Shi, K. Azizzadenesheli, A. Anandkumar, Y. Yue, and Soon-Jo Chung, "Neural-Fly enables rapid learning for agile flight in strong winds," *Science Robotics*, vol. 7, no. 66, pp. 1–15, May 2022.

[12] S. Sun, A. Romero, P. Foehn, E. Kaufmann, and D. Scaramuzza, "A Comparative Study of Nonlinear MPC and Differential-Flatness-Based Control for Quadrotor Agile Flight," *IEEE Transactions on Robotics*, vol. 38, no. 6, pp. 3357–3373, Dec. 2022.

[13] T. Salzmann, E. Kaufmann, J. Arrizabalaga, M. Pavone, D. Scaramuzza, and M. Ryll, "Real-time neural MPC: Deep learning model predictive control for quadrotors and agile robotic platforms," *IEEE Robotics and Automation Letters*, vol. 8, no. 4, pp. 2397–2404, Feb. 2023.

[14] K. Y. Chee, T. Z. Jiahao, and M. A. Hsieh, "KNODE-MPC: A Knowledge-Based Data-Driven Predictive Control Framework for Aerial Robots," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 2819–2826, Apr. 2022.

[15] L. Bauersfeld, E. Kaufmann, P. Foehn, S. Sun, and D. Scaramuzza, "NeuroBEM: Hybrid Aerodynamic Quadrotor Model," in *Proceedings of Robotics: Science and Systems (RSS)*, Jul. 2021, pp. 1–11.

[16] I. Matei, C. Zeng, S. Chowdhury, R. Rai, and J. de Kleer, "Controlling Draft Interactions Between Quadcopter Unmanned Aerial Vehicles with Physics-aware Modeling," *Journal of Intelligent & Robotic Systems*, vol. 101, no. 1, pp. 1–21, Jan. 2021.

[17] D. Mellinger and V. Kumar, "Minimum Snap Trajectory Generation and Control for Quadrotors," in *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, May 2011, pp. 2520–2525.

[18] L. Meier, D. Honegger, and M. Pollefeys, "PX4: A node-based multithreaded open source robotics framework for deeply embedded platforms," in *Proceedings of IEEE International Conference on Robotics and Automation*, May 2015, pp. 6235–6240.

[19] H. Sommer, I. Gilitschenski, M. Bloesch, S. Weiss, R. Siegwart, and J. Nieto, "Why and How to Avoid the Flipped Quaternion Multiplication," *Aerospace*, vol. 5, no. 3, pp. 1–15, Sep. 2018.

[20] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida, "Spectral Normalization for Generative Adversarial Networks," in *Proceedings of International Conference on Learning Representations (ICLR)*, Feb. 2018, pp. 1–14.

[21] C. G. Mayhew, R. G. Sanfelice, and A. R. Teel, "On quaternion-based attitude control and the unwinding phenomenon," in *Proceedings of American Control Conference (ACC)*, Jun. 2011, pp. 299–304.

[22] R. Verschueren, G. Frison, D. Kouzoupis, J. Frey, N. v. Duijkeren, A. Zanelli, B. Novoselnik, T. Albin, R. Quirynen, and M. Diehl, "acados—a modular open-source framework for fast embedded optimal control," *Mathematical Programming Computation*, vol. 14, no. 1, pp. 147–183, Mar. 2022.

[23] M. Grob, "Quaternion based Estimation and Control for Attitude Tracking of a Quadcopter using IMU sensors," Master's thesis, ETH Zürich, Zürich, Switzerland, Sep. 2016.

[24] M. R. Jardin and E. R. Mueller, "Optimized Measurements of Unmanned-Air-Vehicle Mass Moment of Inertia with a Bifilar Pendulum," *Journal of Aircraft*, vol. 46, no. 3, pp. 763–775, May 2009.

[25] D. Falanga, P. Foehn, P. Lu, and D. Scaramuzza, "PAMPC: Perception-Aware Model Predictive Control for Quadrotors," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct. 2018, pp. 1–8.