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

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

***Abstract*—Swarm aerial systems have significant potential for many complex tasks, while their application suffers from the downwash effect, which comes from the downward airflow generated by other quadrotors inside the swarm. Dealing with this effect brings two challenges: first, the airflow is highly nonlinear and hard to model by classic mathematical methods; second, the motor speed of quadrotors is risky to reach its limit when resisting the disturbance.**

To solve these problems, we integrate a Deep Neural Network (DNN) observer with Nonlinear Model Predictive Control (NMPC) to design a trajectory-tracking system. Trained with spectral normalization to ensure robustness and safety on uncollected cases, the DNN observer can predict future disturbances from the relative motion of ego and other quadrotors. This prediction is then merged into the optimization scheme in NMPC, of which the constraints handling property is leveraged to guarantee the motor speed under the limit. Next, we design a quadrotor system, identify its parameters, and implement the proposed method onboard. Finally, we conduct an open-loop prediction experiment and a real-time closed-loop trajectory tracking experiment to verify the algorithm. The former confirms the efficacy and safety of the DNN observer, and the latter demonstrates a 75.37% reduction of tracking error in height under the downwash effect.

***Index Terms*—Model learning for control, model predictive control, aerial systems: applications.**

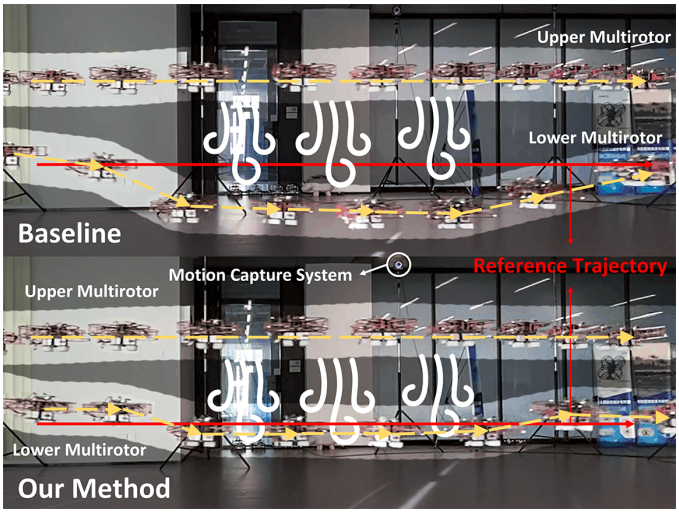


Fig. 1. The comparison of two methods in close-proximity quadrotor flight. In each scene, two multirotors are flying together to track the reference trajectories (red lines) from left to right, and the quadrotor below (yellow lines) suffers from the downwash effect of the upper drone. For the baseline method, the lower drone deviates from the reference for about 40cm, while for our method, this deviation is suppressed. Note that the camera lens distortion slightly affects the flight trajectories in this figure.

I. Introduction

ADVANCES in multi-robot systems have attracted more and more attention since they can achieve complex tasks impossible for one robot through the cooperation and coordination among a group of robots [1]. As one kind of multi-robot systems, the aerial multi-agent systems differ from ground robots in that each agent in the group is affected by downwash effect, the flow generated from its neighbors [2]. When an agent is unable to track its trajectory under the airflow disturbances accurately, it could influence other multirotors in normal flight, thereby amplifying the disturbance until it affects the entire system.

To tackle this problem, one idea is to treat the multirotor as an ellipsoid in the planning layer. Since the planning method considers the collision, the generated trajectories make the quadrotors fly far from each other, avoiding the downwash effect. However, this assumption reduces the *configuration space* of the aerial multi-robot system, unable to push the mobility into their boundary. The other idea is to regard the downwash effect as a *disturbance rejection* problem and solve it in the control layer. In this way, the planning layer only considers the collision radius of each agent, which maximizes the system's mobility. Therefore, it is necessary to model the downwash disturbance and compensate for it in the control loop.

The downwash effect in close flight is challenging to simulate in a general robotics simulator due to its high-degree nonlinearity and randomness. Traditionally, the airflow effect can be accurately modeled through some special flowfield capture devices [3], wind tunnels [4], or Computational Fluid Dynamics (CFD) simulation [5]. However, these approaches all pose high requirements for experimental equipment or computational time. The rapid development of deep learning in recent days renders the possibility to simulate some nonlinear phenomena such as airflow disturbance in low time and fund demanding [6]. Some researchers have begun to apply machine learning to the aerial robotics field with airflow phenomena, such as the ground effect when the quadrotor lands [7], the air disturbance in high-speed flight [8], and the disturbance among quadrotors in close distance [7], [9]. However, in general, the above control approaches are under the classic feedback scheme, which follows the process of system output, error computing, and feedback correction. This scheme needs

[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]H. Yu, Y. Lin, and L. Han are with the Sino-French Engineer School, Beihang University, Beijing, 100191, China {haoyang_yu, linyuheng, liang_han}@buaa.edu.cn

no future information which belongs to causal control. In addition, generated by the relative motions of quadrotors, the disturbance of close-proximity flight can be predicted through the exchange of reference trajectories. Thus, the potential of the predicted disturbances can be fully exploited by being integrated with the control approaches with prediction properties. The predictions, or the non-causal information, can reduce the disturbance in advance. Besides, the quadrotors under the downwash airflow need to generate a more significant output signal to suppress the disturbance, which results in a higher possibility of saturating the actuator. The above control methods neither consider the anti-saturation.

Model Predictive Control (MPC) is a suitable solution to these two problems. The model prediction phase of MPC can fuse the predictions of future disturbances, and its constraints handling capability can also avoid the motor saturation of quadrotors under disturbance environments.

In this letter, we design the trajectory tracking controller in close-proximity flight through the integration of neural network disturbance observer and Nonlinear MPC. First, we train a Multilayer Perceptron (MLP) network to predict the disturbance and utilize *spectral normalization* to ensure robustness and generalizability. Then, we synthesize the observer with NMPC to propose the trajectory tracking method. We also introduce the necessary algorithms to form the close loop, including trajectory generation and the estimation of hover throttle. Finally, we introduce the implementation details and hardware experiments, containing system identification, data collection, open-loop prediction, and closed-loop trajectory tracking. The experiments (Fig. 1) verify the effect of the proposed algorithm.

The main contributions are as follows:

1) proposing a NMPC-based trajectory tracking method with network disturbance prediction (NDP-NMPC) to integrate the future movement information and to solve the saturation constraints under close-proximity flight,
2) making a system including two quadrotors from scratch to implement the NDP-NMPC in real time, and
3) conducting open-loop experiments to verify the predictions and closed-loop experiments to test the trajectory tracking effect.

II. Related Work

A robust and reliable trajectory tracking module is the basis for top-level modules such as decision and planning in multi-agent systems. For rotor-type aerial robots, airflow disturbance is essential, including flight resistance, ground effect, wind blowing, and downwash effect. The downwash effect, generated by other quadrotors nearby, is slightly different from the first three. Even during a slow flight in an indoor and windless environment, the downwash effect still exists and has significant influence.

To tackle this problem, the research [10] and [11] regard the quadrotor as an axis-aligned ellipsoid model with tall height and then use the model to plan a collision-free trajectory. This procedure prevents the quadrotors fly at a close distance. However, crazyflie, the quadrotor used in [10], has a wheelbase of $0.13\mathrm{m}$, while requires the height of $0.6\mathrm{m}$ (4.62 times of the wheelbase) to avoid downwash effect. If we apply this ratio to a larger quadrotor with $0.27\mathrm{m}$ wheelbase, the ellipsoid height becomes $1.25\mathrm{m}$. In our real flight, the larger quadrotor generates a much stronger downwash flow, causing the disturbance even if the relative height achieves more than $2.5\mathrm{m}$. Therefore, for the quadrotor with a larger size, this ellipsoid assumption severely limits the motion space that the planning algorithm can choose.

The other direction is to model the downwash effect as a disturbance and then reject it in the model-based trajectory tracking methods. Some approaches model the airflow through self-made experimental equipment such as [3], [12], while the others utilize the Computational Fluid Dynamics (CFD) for simulation [13], [14]. Nevertheless, the downwash disturbance is complicated due to the coupled influences such as relative position and velocity, making the above methods challenging to cover all situations in a short time. Therefore, many works have begun to apply machine learning methods to capture the laws of airflow in recent years. Among them, [7] is a pioneer work. A neural network processed by spectral normalization is used to fit the ground effect for the first time, which ensures the stability of both learning and control algorithms. Subsequent work by Shi et al. extended this idea to tackle the downwash effect in homogeneous [15] and heterogeneous [9] aerial swarm systems, proving the effect of deep learning on airflow disturbance modeling problems.

Nevertheless, the above articles use the geometric control on SE(3) [16] to track the reference, unable to integrate the prediction, while the downwash disturbance can be predicted in advance. In addition, the method considers no actuator saturation, which tends to happen in the downwash rejected situation.

Model Predictive Control (MPC) can naturally handle these two problems. MPC controls the system by converting the control problem to an iterative, finite-horizon optimization problem of a system model. The MPC is powerful to deal with the actuator saturation constraints. Nonlinear Model Predictive Control (NMPC), the nonlinear version of MPC with much higher computation demands, is gradually applied to the quick-change system (such as quadrotors [17] and race cars [18]) due to the development of the onboard embedded computer and optimization tools. The work [19] has added the air resistance generated in high-speed flight to MPC in the form of prediction. These works lay the foundation for combining the MPC and neural network disturbance observer.

Our work is inspired by [9] and extends it to NMPC to fully exploit the power of prediction.

III. Methodology

In this section, we present our proposed control scheme with both Nonlinear Model Predictive Control (NMPC) and neural network disturbance observer to enhance the tracking performance in close-proximity flight. First, we present the overall workflow in Section III-A. Then, the coordinate system, notation, and a nominal quadrotor model are introduced in Section III-B. Based on these conventions, a neural network

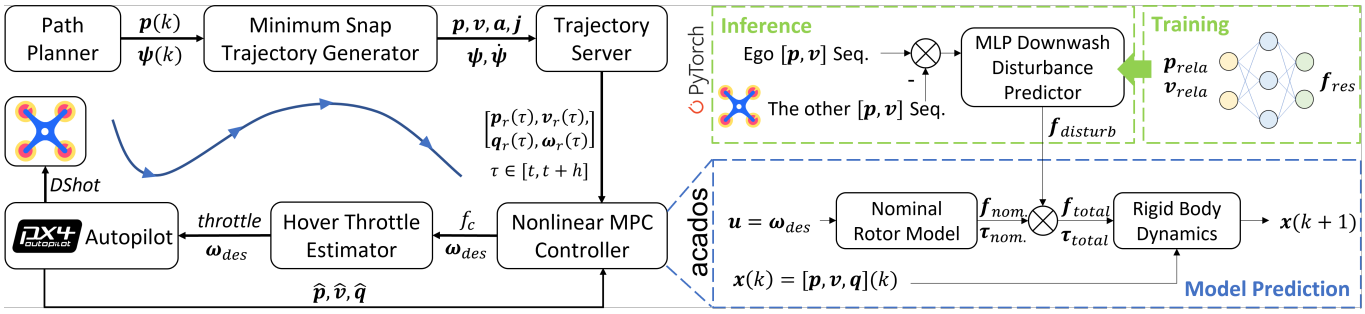


Fig. 2. The overall workflow of the proposed method. The method follows the *Path Planning-Trajectory Generation-Control* pipeline. Specifically, minimum snap is utilized to generate a trajectory, and Nonlinear Model Predictive Control (NMPC) is adopted for trajectory tracking, of which the model is a nominal quadrotor model with a predicted disturbance sequence calculated before every iteration. The disturbances are predicted by a pre-trained neural network from the error state sequence of its own and other quadrotors. Finally, The collective force output from the *Control* module is converted to throttle and then input to the PX4, an open-source autopilot. The measured states generated from the PX4 are sent to the NMPC module to form the close loop.

observer is implemented to predict the downwash disturbance in Section III-C. Finally, a modified NMPC trajectory tracking controller with disturbance prediction is proposed in Section III-D. We also briefly introduce the generation of the reference trajectory and the estimate of the hover throttle in the final part, which is essential to practical implementation.

A. System Overview

The system architecture is illustrated in Fig. 2. From top to bottom, a sequence of position points and yaw angles are first transmitted from a *Path Planner* to a *Trajectory Generator*. The latter module then generates a continuous polynomial trajectory, including multiple derivatives of position and yaw angle. Next, a *Trajectory Server* discretizes the trajectory and then calculates the desired orientation and body rate through differential flatness. These desired state points are sent to a Nonlinear Model Predictive Control (MPC, or NMPC) *Controller* at high frequency to calculate control output. Converted from force to throttle by the *Hover Throttle Estimator*, the control command is eventually executed by a PX4 *Autopilot* to fly the quadrotor. The estimated states are feedback from the *Autopilot* to the *Controller*.

The downwash effect is considered in the controller. The model for the NMPC controller consists of two parts of forces and moments, one comes from a nominal rotor model, and the other comes from a neural network disturbance predictor. The network is trained before the flight. During the flight, it takes the difference between the state of quadrotors as input and then outputs a disturbance force prediction sequence. The disturbance is assumed to be constant if the controller executes at high speed.

B. Nominal Quadrotor Model

We denote constants in non-italic style x and variables in italic style x. 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 frame $\mathcal{I}$ is denoted as ${}^I\boldsymbol{p}$, and the rotation from $\mathcal{B}$ to $\mathcal{I}$ are denoted as ${}^I_B\boldsymbol{R}$ (rotation matrix) or ${}^I_B\boldsymbol{q}$ (attitude quaternion).

We use $\boldsymbol{q} = [q_w, q_x, q_y, q_z]^T \in \mathbb{H}$ to denote the attitude quaternion in *Hamilton-convention* [20], $\overline{\boldsymbol{q}} = [q_w, -q_x, -q_y, -q_z]^T$ to denote the quaternion conjugation, and $\cdot$ to denote the quaternion multiplication operator. The attitude quaternion is a unit quaternion ($\|\boldsymbol{q}\|$=1), and thus the inverse of quaternion $\boldsymbol{q}^{-1}$ is the same as $\overline{\boldsymbol{q}}$. 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 $\mathcal{V}^*(\boldsymbol{p}) := [0, \boldsymbol{p}]^T, \mathbb{R}^3 \to \mathbb{H}$. Then full SE3 transformations from $\mathcal{B}$ to $\mathcal{I}$ can be represented as ${}^I\boldsymbol{p} = \mathcal{V}({}^I_B\boldsymbol{q} \cdot \mathcal{V}^*({}^B\boldsymbol{p}) \cdot {}^I_B\overline{\boldsymbol{q}}) + {}^I\boldsymbol{p}_{Bo} = {}^I_B\boldsymbol{R}(\boldsymbol{q}){}^B\boldsymbol{p} + {}^I\boldsymbol{p}_{Bo}$, where ${}^I\boldsymbol{p}_{Bo}$ is the position of $\mathcal{B}$ frame's origin in the $\mathcal{A}$ frame, and $\boldsymbol{R}(\boldsymbol{q})$ is the rotation matrix from a quaternion following:

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

We assume that the origin of the body frame $\mathcal{B}$ is the same as the center of mass, and four rotors are all placed in the $\mathcal{B}$ frame's xy-plane. Established from 6-DoF rigid body kinematic and dynamic equations, the quadrotor model is

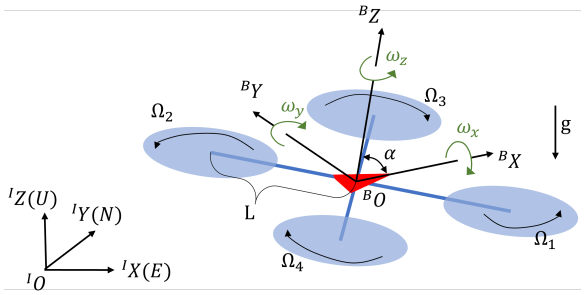


Fig. 3. Diagram of the quadrotor model with the ENU (X East, Y North, Z Up) inertial frame and the FLU (X Forward, Y Left, Z Up) body frame. This definition and propeller numbering convention is compatible with the MAVROS toolkit, a ROS package to communicate with various autopilots such as PX4.

written as follows [17]

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

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

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

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

where ${}^{I}\mathbf{g} = [0, 0, -\mathrm{g}]^T$ is the gravity vector, $\mathbf{I} = \mathtt{diag}(\mathrm{I_{xx}}, \mathrm{I_{yy}}, \mathrm{I_{zz}})$ is the inertia matrix, ${}^{B}\boldsymbol{f}_u$ and ${}^{B}\boldsymbol{\tau}_u$ are the force and torque caused by the rotors, ${}^{I}\boldsymbol{f}_d$ and ${}^{B}\boldsymbol{\tau}_d$ are the same effect caused by disturbances, and ${}^{B}\boldsymbol{\omega} = [\omega_x, \omega_y, \omega_z]^T$ is the angular rate vector expressed in the body frame.

The thrust generated by rotors is assumed to be vertical to the 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 of each propeller:

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

where $\mathrm{k_t}$ and $\mathrm{k_q}$ are the thrust coefficient and the torque coefficient, respectively, as well as Ω represents motor speed in RPM. Then the $[f_c, \tau_x, \tau_y, \tau_z]^T$ 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

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

where L and α are geometric parameters shown in Fig. 3.

The quadrotor nominal model established above is used to design both disturbance observer and controller. The disturbing ${}^{I}\boldsymbol{f}_d$ and ${}^{B}\boldsymbol{\tau}_d$ are estimated in the next part.

C. Neural Network Observer for Downwash Effect

The evolution of deep learning renders the possibility to observe the complicated downwash effect in real flight. In this part, we introduce a neural network observer to model the disturbance between quadrotors in vertical-aligned flight.

1) Neural Network Disturbance Observer: The disturbance observer can estimate the disturbing force or torque based on the input states. For the downwash effect, the input variables contain relative states are ego quadrotor and the other quadrotor $\left[{}^{I}\boldsymbol{p}_{rela}, {}^{I}\boldsymbol{v}_{rela}, {}^{I}_{B}\boldsymbol{q}_{rela}, {}^{B}\boldsymbol{\omega}_{rela}\right]$, and the output value is the disturbances ${}^{I}\hat{\boldsymbol{f}}_d$ and ${}^{B}\hat{\boldsymbol{\tau}}_d$.

Due to the nonlinearity and fast-changing property of airflow, we apply a neural network to estimate the disturbance. Specifically, we utilize a Multi-Layer Perceptron (MLP) to observe the disturbance. A trained MLP can be viewed as a mapping function $f(\cdot;\boldsymbol{\theta}) : \mathbb{R}^i \to \mathbb{R}^o$ from the input tensor $\boldsymbol{x}$ to the output result $\boldsymbol{y}$, where $\boldsymbol{\theta} := \left\{\boldsymbol{W}^1, \cdots, \boldsymbol{W}^{\mathrm{H}+1}\right\}$ represents the weight parameters, and H is the number of hidden layers. The element-wise ReLU function $\phi(x) = \max(0, x)$ is selected as the activation function, and then the MLP network can be written as

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

When we collected training data in the experiments, it is obvious that the data is impossible to cover the entire state space. In this case, the network is prone to overfit, and hence the output of the network in those data-uncovered states is critical to flight safety. To alleviate this problem, the spectral normalization from the Generative Adversarial Networks (GAN) [21] field is adopted.

2) Spectral Normalization: The spectral normalization can limit the Lipschitz constant of the network and then raise its robustness and generalizability. The Lipschitz constant of a function $\|f\|_{\mathrm{Lip}}$ is defined as the smallest value M that

$$ \|f(\boldsymbol{x}) - f(\boldsymbol{x'})\| / \|\boldsymbol{x} - \boldsymbol{x'}\| \leq M \tag{9} $$

for any $\boldsymbol{x}$ and $\boldsymbol{x'}$, where $\|\cdot\|$ is the l_2 norm. Mathematically, the Lipschitz constant $\|g(\boldsymbol{x})\|_{\mathrm{Lip}}$ is equal to the maximum spectral norm (maximum singular value) $\sigma(\cdot)$ of the function's gradient $\sup_{\boldsymbol{x}} \sigma\left(\nabla g(\boldsymbol{x})\right)$. We can apply (8) to this definition and then obtain

$$ \|f(\boldsymbol{x})\|_{\mathrm{Lip}} = \sigma\left(\prod_{l=1}^{\mathrm{H}+1} \boldsymbol{W}^l\right) \leq \prod_{l=1}^{\mathrm{H}+1} \sigma\left(\boldsymbol{W}^l\right). \tag{10} $$

If the weight matrix $\boldsymbol{W}^l$ is normalized by the spectral norm $\sigma(\cdot)$ and the scale ratio γ at each training epoch

$$ \overline{\boldsymbol{W}}^l := \gamma \cdot \boldsymbol{W}^l / \sigma\left(\boldsymbol{W}^l\right), \tag{11} $$

then the Lipschitz constant is constrained as

$$ \|f(\boldsymbol{x})\|_{\mathrm{Lip}} \leq \gamma^{\mathrm{H}+1}. \tag{12} $$

The spectral normalization actually limits the rate of function's change, making the function more uniform. This uniformity prevents the network from overfitting. Our subsequent experiments also proved this feature.

3) Data Acquisition: To obtain the training set, real-world flight data are collected (see Section IV-B for details) using the odometry module and the nominal quadrotor model.

The odometry module can provide the estimation of velocity and body rate, and then the real resultant force and torque on the body is obtained through classical mechanics:

$$ {}^{I}\boldsymbol{f} = \mathrm{m} \cdot {}^{I}\dot{\hat{\boldsymbol{v}}}, \quad {}^{B}\boldsymbol{\tau} = \mathbf{I} \cdot {}^{B}\dot{\hat{\boldsymbol{\omega}}}. \tag{13} $$

Note that the derivative of $\hat{\boldsymbol{v}}$ and $\hat{\boldsymbol{\omega}}$ are numerically calculated through the Tustin's method. We also tried to obtain the resultant force directly from the inertial measurement unit (IMU), but the noise is unacceptable.

For the nominal dynamics model, the nominal resultant force and torque is calculated as follows:

$$ {}^{I}\boldsymbol{f}_n = {}^{I}_{B}\boldsymbol{R}(\boldsymbol{q}) \cdot {}^{B}\boldsymbol{f}_u + \mathrm{m} \cdot {}^{I}\mathbf{g}, \quad {}^{B}\boldsymbol{\tau}_n = {}^{B}\boldsymbol{\tau}_u, \tag{14} $$

where ${}^{B}\boldsymbol{f}_u$ and ${}^{B}\boldsymbol{\tau}_u$ come from (5) and (6) given the measured motor speeds Ω_i.

Finally, the disturbance ${}^{I}\boldsymbol{f}_d$ and ${}^{B}\boldsymbol{\tau}_d$ is attained through

$$ {}^{I}\boldsymbol{f}_d = {}^{I}\boldsymbol{f} - {}^{I}\boldsymbol{f}_n, \quad {}^{B}\boldsymbol{\tau}_d = {}^{B}\boldsymbol{\tau} - {}^{B}\boldsymbol{\tau}_n. \tag{15} $$

Now the input $\left[{}^{I}\boldsymbol{p}_{rela}, {}^{I}\boldsymbol{v}_{rela}, {}^{I}_{B}\boldsymbol{q}_{rela}, {}^{B}\boldsymbol{\omega}_{rela}\right]$ and the output $\left[{}^{I}\boldsymbol{f}_d, {}^{B}\boldsymbol{\tau}_d\right]$ has been constructed for neural network training.

D. Nonlinear MPC with Network Disturbance Prediction

In this subsection, we introduce the trajectory tracking controller using the Nonlinear Model Predictive Control (NMPC) method. During the close-proximity flight, we assume that each quadrotor can obtain the future trajectory of other quadrotors, which makes the prediction of future disturbance available. In addition, the control reference generation step and the post-processing step are also briefly introduced.

1) Reference Trajectory Generation: Trajectory generation refers to the process that generating a smooth, dynamic feasible, and time-indexed curve to pass a set of points including start point, several pre-defined waypoints, and the end point. As the quadrotors are mathematically *differential flatness*, the trajectory generation can be converted to a polynomial optimization problem for the four flat output $[x, y, z, \psi]$, which are 3D position and yaw angle. We implement the *minimum snap* algorithm based on [22], except that the time allocation between points is achieved by simply dividing the relative distance by the user-defined average velocity.

The generated trajectory is a set of parametric equations. Therefore, a trajectory server is needed to discretize the curve, to choose the future reference states compatible with the predict horizon of NMPC, and to publish these reference at a frequency no lower than the control loop.

2) Nonlinear MPC: Nonlinear MPC is chosen in close-proximity flight because of two reasons. First, the thrust command is highly possible to saturate when the lower quadrotor is influenced by the upper one's downwash airflow, and MPC is powerful for saturation handling. Second, the ideal trajectory of other quadrotor can be obtained through the network, making the prediction of future disturbance available. This prediction is helpful to improve the flight performance in theory.

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

$$\min_{\boldsymbol{u}_k} \left(\overline{\boldsymbol{x}}_N^T \mathbf{Q}_N \overline{\boldsymbol{x}}_N + \sum_{k=0}^{N-1} \left(\overline{\boldsymbol{x}}_k^T \mathbf{Q}_x \overline{\boldsymbol{x}}_k + \overline{\boldsymbol{u}}_k^T \mathbf{R} \overline{\boldsymbol{u}}_k \right) \right) \tag{16}$$

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

$$\begin{aligned} \boldsymbol{x}_{k+1} &= f\left(\boldsymbol{x}_k, \boldsymbol{u}_k\right), \\ \boldsymbol{x}_0 &= \boldsymbol{x}_{\text{init}}, \\ \boldsymbol{u}_k &\in \left[\boldsymbol{u}_{\min}, \boldsymbol{u}_{\max}\right], \end{aligned} \tag{17}$$

where the symbol $\overline{(\cdot)} = (\cdot) - (\cdot)_r$ denotes the difference with respect to the reference, and $f(\cdot)$ is the quadrotor nominal model (1-3) discretized by the 4-order Runge-Kutta method. Specifically, the state is $\boldsymbol{x}_k = \left[{}^I\boldsymbol{p}_k, {}^I\boldsymbol{v}_k, {}^I\boldsymbol{q}_k\right]^T$, and the control input is $\boldsymbol{u}_k = \left[f_c, {}^B\boldsymbol{\omega}, {}^I\boldsymbol{f}_d\right]^T$. Note that the input ${}^I\boldsymbol{f}_d$ is actually a fixed value bounded by ${}^I\boldsymbol{f}_{d\min} = {}^I\boldsymbol{f}_{d\max} = {}^I\hat{\boldsymbol{f}}_d$. In this way, the disturbance predictions of the neural network observer are incorporated into the MPC system. Then *Warm-Starting*, *Real-Time Iteration (RTI)*, and *Multi-Shooting* techniques are applied to accelerate the computing speed. This process of problem construction is illustrated in Fig. 4. Finally,

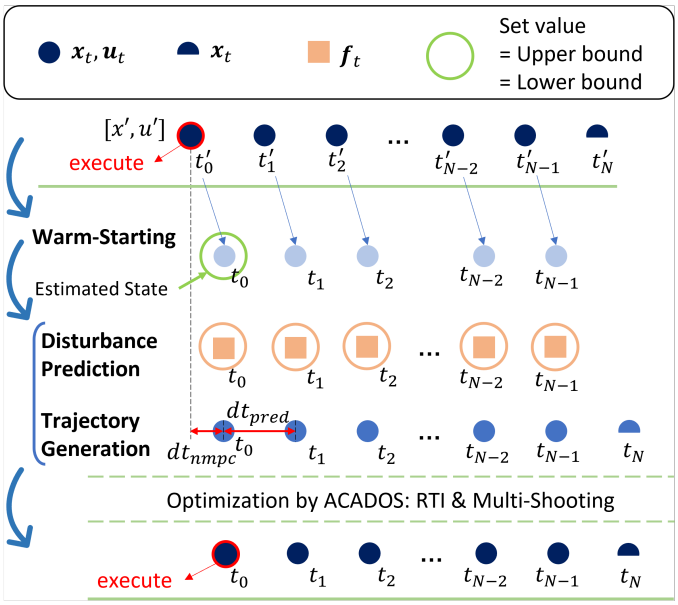


Fig. 4. A detailed diagram to illustrate NDP-NMPC algorithm. From top to bottom, every state points' initial values in this iteration comes from the result of last iteration, which is called warm-stating. Then the first state is constrained to the real system state, introducing the feedback mechanism into MPC. Next, the disturbances are predicted by the neural network, and the reference states are obtained by trajectory generation. Eventually these data points are sent into ACADOS for optimization. Note that the predictive horizon dt_{pred} is usually longer than the control period dt_{nmpc}.

the control command is extracted from the optimized result sequence:

$$\boldsymbol{u}_{\text{NDP-NMPC}} = \boldsymbol{u}_0^* = \left[f_c', \omega_x', \omega_y', \omega_z'\right]. \tag{18}$$

Note that we drop (4) in quadrotor nominal model since the PX4 contains a body rate controller working in 1kHz, which provides an effective compensation for the disturbance torque $\boldsymbol{\tau}_d$ in real flight. This is also the reason why we exclude the $\boldsymbol{\tau}_d$ from the input.

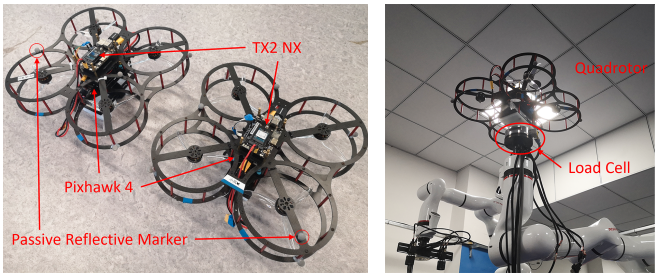
3) Hover Throttle Estimate: The Nonlinear MPC generates the collective thrust and torque as control input. However, the PX4 autopilot requires the thrust to be normalized into $[0, 1]$. We denote this normalized thrust as h and then the hover throttle as h_h. In actual flight, the normalization ratio reduces slowly due to the decrease of the battery voltage. To provide an accurate estimation of the hover throttle, we implement a simple Kalman Filter estimator similar to the PX4 autopilot.

The estimator uses $\hat{\boldsymbol{x}} = [\hat{f}_c, \hat{\gamma}_h]^T$ as the state and $z = {}^B a_{sp,z}$ as the measurement, where $\hat{f}_c$ is the collective force generated by all rotors, $\hat{\gamma}_h := \text{mg}/h_h$ is the hover throttle ratio, and $a_{sp,z}$ is the *specific force* of z axis measured from the IMU. If we assume that the specific force all comes from the rotors, then the system model and the measurement model can be written as

$$\hat{\boldsymbol{x}}(k) = \begin{bmatrix} 0 & h(k) \\ 0 & 1 \end{bmatrix} \cdot \hat{\boldsymbol{x}}(k-1), \tag{19}$$

$$z(k) = \left[\frac{1}{\text{m}}, 0\right] \cdot \hat{\boldsymbol{x}}(k). \tag{20}$$

Following the standard equations of Kalman Filter, the hover throttle ratio $\hat{\gamma}_h$ can be identified in the flight. Finally we



(a) Flight platform (b) System identification

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

normalize the thrust force as $h' = f'_c/\hat{\gamma}_h$ and then feed $\left[h', \omega'_x, \omega'_y, \omega'_z\right]$ into the PX4 autopilot.

It is noted that the assumption we made before neglects all disturbances, and hence the estimation only works when the quadrotors are not affected by each other. In the real flight, we update the hover throttle ratio at the time before trajectory tracking.

IV. Experiments

In this section, we introduce the setup of experiments and the analysis of results. We first present the system identification of hardware platforms, then the data collection of the downwash effect, next an open-loop experiment for disturbance prediction, and finally a closed-loop trajectory tracking experiment to verify the whole system.

A. Hardware and System Identification

To verify the proposed algorithm, we made our flight platform as shown in Fig. 5a. The platform contained a pixhawk 4 for low-level flight control and state estimation, an NVIDIA TX2 NX for algorithm execution, several passive reflective markers for indoor localization, and other electrical devices essential to flight such as WIFI module, receiver, and ESC. Note that the ESC we used supported rotor speed data transmission.

We followed the similar procedure in [23] to identify the rotor parameters with a load cell, as shown in Fig. 5b. We 3D printed a connector to place the whole quadrotor on the load cell, and then gave square wave signals to stimulate a pair of propellers on the diagonal. Compared with only placing one motor on the load cell, this setup took the body's disturbance on rotors into consideration. We assumed that the quadrotor was symmetrical in all three axes, and then the inertial matrix became a diagonal matrix, of which the diagonal elements were measured using the bifilar pendulum method. All parameters are listed in Table I.

B. Data Collection, Learning, and Open-Loop Prediction

Two quadrotors were controller by pilots to fly over each other to collect data points under the downwash circumstances. We fixed one quadrotor and moved the other to raise the possibility of overlapping. The height of the moving one varied in each flight to increase the data diversity. Quadrotors' states and estimated disturbances data with a timestamp are collected at 100 Hz using the ROS tool `rosbag`. Finally, $57,000$ data points are successfully collected for each quadrotor. Then the data are aligned in time, and the average disturbance force in the hover state is subtracted to remove the bias. Note that two quadrotors are totally symmetrical, and thus the data sets can be doubled.

When training the network, we choose the hidden layer $\mathrm{H} = 3$. The network used Adam [24] as the optimizer and Mean Squared Error (MSE) as the loss function. We fixed the epoch number as 10000 and the learning rate as 1e-4. Collected data were shuffled, and 70 percent of them were selected as the training data. The spectral normalization ratio γ was critical to the balance of accuracy and robustness, especially the performance in the cases where data did not cover. Therefore, we executed some experiments to find the best value. The losses on testing data for various γ were listed in Table II. The prediction of different γ is shown as Fig. 6.

From the table, it is obvious that the loss increases with the decrease of γ, i.e., the stronger the influence of spectral normalization, the weaker the ability to fit the data. Nevertheless, this shortage comes with a safer prediction on points without data collection, which can be observed in Fig. 6. This figure shows the predicted disturbance force on the Z axis for different γ at different heights. The results at $\gamma = (2, 4, 8)$ are from top to bottom. The relative heights of another quadrotor to the ego one are listed from left to right, where the negative numbers indicate flying above. The disturbance is high in the middle and low at the edges in all figures. For the second and third rows, the trend is correct: when the two quadrotors fly away from each other, the downwash effect decreases. However, the first row is too conservative and shows almost no differences. For the first column on the right, where no data points are collected, $\gamma = 8$ outputs too much disturbance, while $\gamma = 4$ and $\gamma = 2$ maintain continuity. Therefore, we choose $\gamma = 4$ as the balance of data fitting and safety. The

TABLE I
Identified Parameters

Parameter(s)	Value(s)	Unit
L	0.1372	m
α	45	deg
m	1.5344	kg
g	9.81	$\mathrm{m/s^2}$
$\mathrm{I_{xx}}$	0.0094	$\mathrm{kg \cdot m^2}$
$\mathrm{I_{yy}}$	0.0134	$\mathrm{kg \cdot m^2}$
$\mathrm{I_{zz}}$	0.0145	$\mathrm{kg \cdot m^2}$
$\mathrm{k_q}$	3.7611 E-10	$\mathrm{N \cdot m/RPM^2}$
$\mathrm{k_t}$	2.8158 E-8	$\mathrm{N/RPM^2}$
$[\Omega_{\min}, \Omega_{\max}]$	[2600, 24000]	RPM
thrust/weight	4.3100	—
flight time	705	s

TABLE II
Loss in various spectral normalization ratio γ

γ	2	3	4	5	8
Loss	1.6283	1.2955	1.0477	0.9051	0.6212

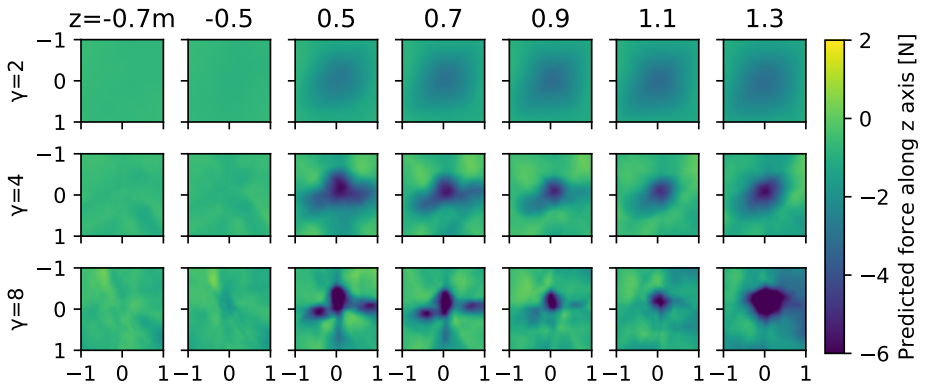


Fig. 6. Disturbance predictions of the neural network under different spectral normalization ratios γ. We fix the relative velocity to zero and change the γ to observe the output at different heights. When the γ increases, the prediction is more specific and easier to overfit. For the third row, we can observe that the predicted force suddenly increases at 1.3m due to the lack of data. A high γ (the middle and up rows) factor mitigates this trend. The two left columns indicate that one quadrotor receives almost no disturbance when it flies over the other. The figure also illustrates that it is safe for quadrotors to fly within 0.5 meters with this disturbance observer.

disturbance prediction is more trivial for the X and Y axis than the Z axis, so they are not displayed.

C. Closed-Loop Trajectory Tracking

In this part, we made the whole system a closed loop to track the given trajectories. The aim was to verify the feasibility of the proposed method considering hardware limitations such as system delay or limited computing power for embedded platforms. We implemented the algorithm in python on the onboard TX2 NX computer and flew the quadrotors at a $7 \times 4 \times 3$ room with an OptiTrack motion capture system. The frequency for the method was approximately 60 Hz. We utilized the Robot Operation System (ROS) for communication. During the experiment, a PC ran the ROS master node and broadcasted mocap data. It also controlled the experimental process. Waypoints for each quadrotor were pre-defined, and then reference trajectories were generated on this PC. These trajectories were designed to ensure an overlapping area in flight. As shown in Fig. 7, two quadrotors started from the same side at different heights and then moved back and forth. The strong downwash effect existed in the middle area ($0 \leq x \leq 2$). Finally, these trajectory points were sent to the quadrotors to track.

We executed multiple runs to test the performance of closed-loop tracking, and we set the PAMPC method [25] as the baseline. The result is shown as Fig. 8. From Fig. 8a, the quadrotor using the baseline method is severely affected and deviates from the reference at $1 \leq x \leq 2$. However, the proposed method significantly reduces the impact of disturbances, decreasing the tracking RMSE by 75.37%. The reduction verifies the effect of the proposed method on the close-proximity flight. From Fig. 8b, the proposed approach produces a slightly stronger fluctuation.

In reality, we also observed that the lower quadrotor suddenly jumped before it entered the downwash flow area. Then the drone was pushed down to the reference height. This phenomenon is caused by the inaccuracy of neural network predictions. The network thinks there will be a disturbance in the future while actually there is not, so the lower quadrotor is raised in advance with no force to counteract this trend. This

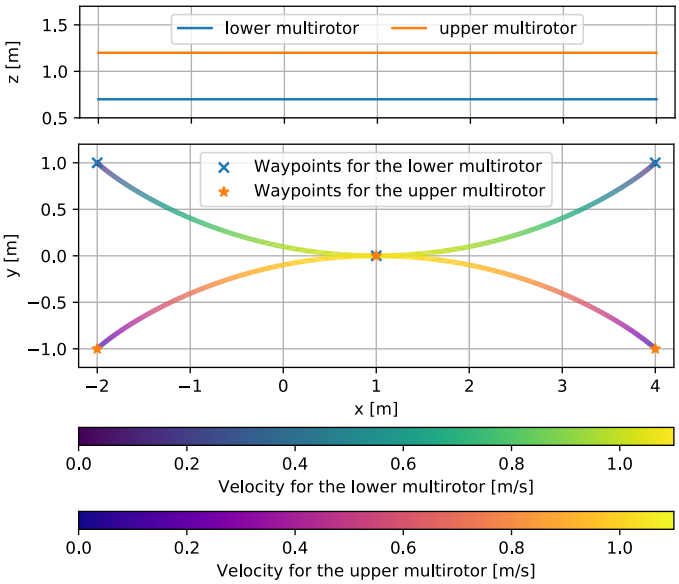
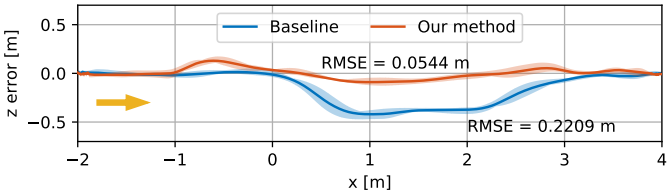


Fig. 7. The fly trajectories for the closed-loop experiment. Two quadrotors move at different heights (0.7m for the lower one and 1.2m for the higher one). They start together from one side and move to the other side, overlapping in the middle area. They finally cycle back and forth.

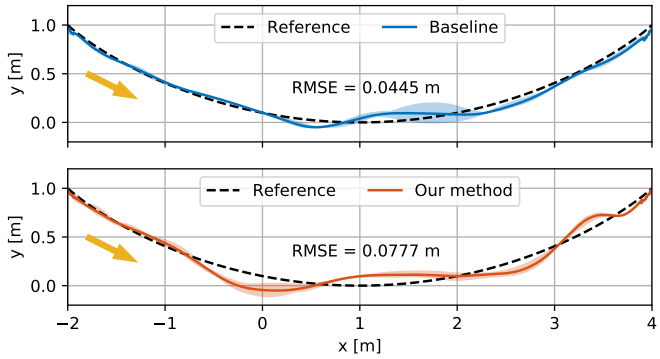
inaccuracy also causes fluctuations on the horizontal plane, which can be observed in Fig. 8a. Therefore, when attempting to combine future information to improve performance, the effect of future inaccuracy must be considered with caution.

V. Conclusion

In this letter, 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 a Nonlinear Model Predictive Control controller. The NMPC method had natural specifications to handle constraints, so it was adopted to avoid the actuator saturation in close flight. To test the method, we identified the quadrotor hardware platform, collected data under downwash disturbances, trained the neural network, and studied



(a) Flight trajectories on the XZ plane. Z-axis data are offset.



(b) Flight trajectories on the XY plane

Fig. 8. Closed-loop tracking results of the baseline [25] and the proposed method

the performance for open-loop prediction. Finally, we ran the algorithm in real-time on two quadrotors to execute the closed-loop trajectory tracking experiment. We reported that the network could successfully generalize to the unseen data, and the proposed approach reduced 75.37% tracking error in the Z axis. We plan to extend the proposed method to more drones in the future. In addition, we will reduce the impact of future prediction's uncertainty while ensuring the prediction effect.

Acknowledgments

We thanks Dr. Sihao Sun for insightful discussions. In addition, we are grateful to Prof. Qinglei Hu for providing the devices for system identification. We thanks Zilu Chen for mechanical design.

References

[1] X. Zhou, X. Wen, Z. Wang, Y. Gao, H. Li, Q. Wang, T. Yang, H. Lu, Y. Cao, C. Xu, and F. Gao, "Swarm of micro flying robots in the wild," *Science Robotics*, vol. 7, no. 66, 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] 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, 2021.

[4] Z. Ma, E. J. Smeur, and G. C. de Croon, "Wind tunnel tests of a wing at all angles of attack," *International Journal of Micro Air Vehicles*, vol. 14, pp. 1–11, Jan. 2022.

[5] Q. Guo, Y. Zhu, Y. Tang, C. Hou, Y. He, J. Zhuang, Y. Zheng, and S. Luo, "CFD simulation and experimental verification of the spatial and temporal distributions of the downwash airflow of a quad-rotor agricultural UAV in hover," *Computers and Electronics in Agriculture*, vol. 172, p. 105343, May 2020.

[6] J. Willard, X. Jia, S. Xu, M. Steinbach, and V. Kumar, "Integrating Scientific Knowledge with Machine Learning for Engineering and Environmental Systems," Jul. 2021.

[7] 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 *2019 IEEE International Conference on Robotics and Automation (ICRA)*. Montreal, QC, Canada: IEEE, May 2019, pp. 9784–9790.

[8] L. Bauersfeld, E. Kaufmann, P. Foehn, S. Sun, and D. Scaramuzza, "NeuroBEM: Hybrid Aerodynamic Quadrotor Model," in *Robotics: Science and Systems XVII*. Held Virtually: Robotics: Science and Systems Foundation, Jul. 2021.

[9] 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.

[10] 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.

[11] 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.

[12] D. Yeo, E. Shrestha, D. A. Paley, and E. M. Atkins, "An Empirical Model of Rotorcrafy UAV Downwash for Disturbance Localization and Avoidance," in *AIAA Atmospheric Flight Mechanics Conference*. Kissimmee, Florida: American Institute of Aeronautics and Astronautics, Jan. 2015.

[13] P. Sanchez-Cuevas, G. Heredia, and A. Ollero, "Characterization of the Aerodynamic Ground Effect and Its Influence in Multirotor Control," *International Journal of Aerospace Engineering*, vol. 2017, p. e1823056, Aug. 2017.

[14] 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.

[15] G. Shi, W. Hönig, Y. Yue, and S.-J. Chung, "Neural-Swarm: Decentralized Close-Proximity Multirotor Control Using Learned Interactions," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. Paris, France: IEEE, May 2020, pp. 3241–3247.

[16] T. Lee, M. Leok, and N. H. McClamroch, "Geometric tracking control of a quadrotor UAV on SE(3)," in *49th IEEE Conference on Decision and Control (CDC)*, 2010, pp. 5420–5425.

[17] 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," Feb. 2022, arXiv: 2109.01365. [Online]. Available: http://arxiv.org/abs/2109.01365

[18] J. Kabzan, L. Hewing, A. Liniger, and M. N. Zeilinger, "Learning-Based Model Predictive Control for Autonomous Racing," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3363–3370, Oct. 2019.

[19] G. Torrente, E. Kaufmann, P. Föhn, and D. Scaramuzza, "Data-Driven MPC for Quadrotors," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3769–3776, Apr. 2021.

[20] 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, p. 72, Sep. 2018.

[21] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida, "Spectral Normalization for Generative Adversarial Networks," Feb. 2018.

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

[23] J. Förster, "System Identification of the Crazyflie 2.0 Nano Quadrocopter," Bachelor's thesis, ETH Zurich, Aug. 2015, accepted: 2017-12-15T11:43:49Z Publisher: ETH Zurich. [Online]. Available: https://www.research-collection.ethz.ch/handle/20.500.11850/214143

[24] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *arXiv:1412.6980 [cs]*, Jan. 2017.

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