

ProxFly: Robust Control for Close Proximity Quadcopter Flight via Residual Reinforcement Learning

Ruiqi Zhang, Dingqi Zhang, and Mark W. Mueller

Abstract—This paper proposes the ProxFly, a residual deep Reinforcement Learning (RL)-based controller for close proximity quadcopter flight. Specifically, we design a residual module on top of a cascaded controller (denoted as *basic controller*) to generate high-level control commands, which compensate for external disturbances and thrust loss caused by downwash effects from other quadcopters. First, our method takes only the ego state and controllers’ commands as inputs and does not rely on any communication between quadcopters, thereby reducing the bandwidth requirement. Through domain randomization, our method relaxes the requirement for accurate system identification and fine-tuned controller parameters, allowing it to adapt to changing system models. Meanwhile, our method not only reduces the proportion of unexplainable signals from the black box in control commands but also enables the RL training to skip the time-consuming exploration from scratch via guidance from the basic controller. We validate the effectiveness of the residual module in the simulation with different proximities. Moreover, we conduct the real close proximity flight test to compare ProxFly with the basic controller and an advanced model-based controller with complex aerodynamic compensation. Finally, we show that ProxFly can be used for challenging quadcopter in-air docking, where two quadcopters fly in extreme proximity, and strong airflow significantly disrupts flight. However, our method can stabilize the quadcopter in this case and accomplish docking. The resources are available at <https://github.com/ruiqizhang99/ProxFly>.

I. INTRODUCTION

Flying quadcopters in close proximity is a challenging task but has a number of real-world applications. Such scenarios arise during collaborative mapping and exploration missions, where the mission is confined to a limited workspace [1], [2]. Meanwhile, in some cases like aerial docking or payload transport [3], [4], a close proximity quadcopter flight control is intended. However, as a significant challenge, the complex aerodynamic interaction occurs when quadcopters fly close to each other, and poses an additional risk and constraint for motion planning [5]. For example, as one quadcopter flies above another, the lower quadcopter is subjected to the downwash effect caused by the upper one. Specifically, this effect results in thrust loss, complex external forces and torques on the lower quadcopter, which is difficult to model using conventional model-based approaches [6]. Previous work reveals the high-fidelity aerodynamics modeling for quadcopter control in both free-flight [6], [7] and over-actuated configuration [8] via the computational fluid dynamics (CFD) software and wind tunnel experiments. A good example is the flying-battery [4]. To realize the in-air docking

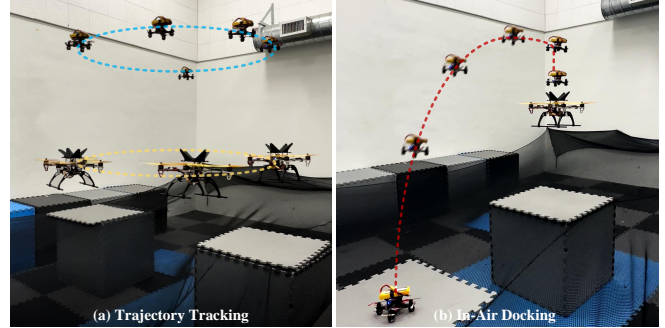


Fig. 1. Flying quadcopters in close proximity with ProxFly. (a) Two quadcopters are tracking circular trajectories. (b) The small quadcopter are taking off, approaching and docking with the large one in the air.

and charging for the flight time extension, researchers use explicit aerodynamic models for thrust compensation. However, these approaches rely on precise system identification and fine-tuning of controller parameters. Moreover, they are computationally expensive and can hardly be transferred to other diverse quadcopter models.

Recently, deep learning (DL) is a rapidly developing data-driven method for extracting complex latent representation from large amounts of data [9]. It enables many challenging tasks like visual-based navigation [10], [11] and near-ground flight [12], which provide new perspectives for achieving robust flight control. For example, learning a residual model to compensate for ground effects from real flight data and assist the stable control under complex aerodynamics effect [12]. However, these supervised learning methods rely on extensive data collected from human pilots and manual annotations, and the performance of learned controllers heavily depends on the skill level of the human pilots. DL can also be used to predict the downwash effect through real-world flight datasets, so that we can incorporate the learned model into motion planning, which are verified on high-speed navigation [13], homogeneous [14] and heterogeneous [15] quadcopter swarms. However, these methods are only validated on micro quadcopters with weak downwash effect and large spacing, so it is uncertain whether the conclusions can be generalized to regular quadcopters and other closer proximity flight tasks.

As a powerful sequential decision-making tool, reinforcement learning (RL) can be used for many challenging tasks like flying in wind field or with off-center payload [16]–[18]. With massive data from the simulation, RL can learn a controller that maximizes cumulative returns via reward signals and avoid the risks of real flight data collection [19]. However, the policies learned through this approach

The authors are with High Performance Robotics Lab, Department of Mechanical Engineering, University of California Berkeley, CA 94720, United States. Email: {richzhang, dingqi, mwm}@berkeley.edu

tend to fit specific model configurations and tasks, so they are unsuitable for close proximity flight where system dynamics change significantly. Recent studies propose the online system identification [16] and model-based meta-learning [20] to make quadcopters adapt to diverse external disturbances. Meanwhile, multi-agent RL control can take the state information from other robots and realize the active compensation [21]. However, multi-agent RL relies on the solid assumption of precise modeling and stable signal connection for communication [22]. Moreover, the lack of explicability in black box-based control remains a tricky and unsolved problem in the robotics community [23].

To address these challenges, we propose ProxFly, a controller for close proximity flight based on residual RL [24] (or residual policy learning [25]). ProxFly incorporates a residual module on top of a basic cascaded controller, which generates high-level thrust and body rate commands. It compensates for external disturbances caused by the downwash effect, impulse from another quadcopter and unknown payload. Meanwhile, it takes only ego states and command outputs but doesn't rely on any forms of communication. Through the domain randomization, residual module relaxes the requirements for accurate system identification, disturbance modeling, and fine-tuning of controller parameters, so that ProxFly can adapt to diverse disturbances during the flight. Moreover, this approach reduces the proportion of unexplainable signals from the black box in the overall command and leverages guidance from a basic controller, allowing to skip time-consuming exploration from scratch of the policy network.

In the experiment section, we first validate the effectiveness of the residual module in the simulator. After that, we conduct the comparative real flight test of ProxFly, the basic controller, and an advanced model-based controller established on the complex aerodynamics model [4]. Specifically, we test two quadcopters for hovering and trajectory tracking in close proximity. The results show it significantly improves the basic controller's position and attitude control accuracy and achieves a comparative performance with the advanced method. Finally, we demonstrate the capability of ProxFly in quadcopter aerial docking. This is a highly challenging task since two quadcopters are required to fly at extremely close proximity, where strong downwash and local vortices disturb and reduce the propeller efficiency. Quadcopter docking also introduces impulse interference and permanently alters the quadcopter's dynamics model, which sets a high standard of adaptation and robustness for the controller.

II. METHODOLOGY

In this section, we introduce the methodology of ProxFly. Specifically, we illustrate the principle, pipeline, setting and detailed implementation of our method.

A. Residual Policy Learning

As shown in Figure 2, ProxFly uses a superposition of two control signals from the cascaded model-based controller and the residual module. The fundamental cascaded controller

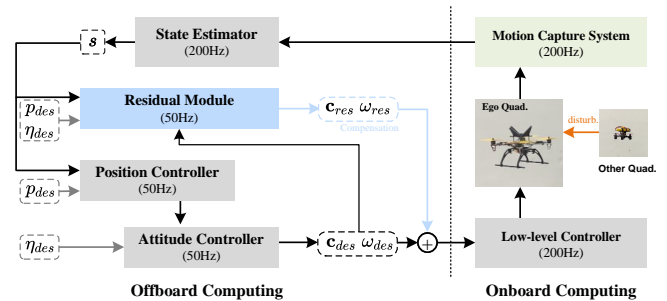


Fig. 2. **The pipeline of ProxFly.** The high-level position and attitude controllers generate the desired mass-normalized thrust c_{des} and the body rate ω_{des} . Meanwhile, the residual module takes the current state s , desired position p_{des} and attitude η_{des} and the last command from basic high-level controller. Then it generates the residual thrust c_{res} and body rates ω_{res} as a compensation of the basic controller. The overall commands can be calculated and sent to the model-based low-level controller to generate the motor speed commands. The ground truth of states are from motion capture system and the state estimator provides the estimated states to the controller.

only requires the predefined natural frequency, damping parameters and basic model constants. It is widely used in quadcopter control and path planning [6], [26]. Although this controller design is simple and efficient, it has imperfections and the performance depends heavily on accurate model parameters. Hence, we leverage the residual RL [24] to improve the performance on the top of the cascaded controller. Specifically, the output of our ProxFly can be represented as (1), where the \mathbf{u}_{cas} and π_{res} are the output from the cascaded controller and residual module, respectively. s and o presents the quadcopter state and observational vector. θ denotes the learnable parameter in the residual policy network.

$$\pi(o|\theta) = \mathbf{u}_{cas}(s) + \pi_{res}(o|\theta) \quad (1)$$

Note that during the training process, the gradient of policy satisfies the condition $\nabla_{\theta}\pi(o|\theta) = \nabla_{\theta}\pi_{res}(o|\theta)$, because \mathbf{u}_{cas} is not a function of the inner parameter θ . In other words, from the learning perspective, we can optimize the residual policy to maximize the superpose policy's reward, as their learning objectives are aligned. The advantages of this learning scheme are diverse. First, RL from scratch remains data-inefficient or intractable. However, learning a residual on top of the basic controller is simpler and skips the time-consuming exploration process at the initial stage [25], [27]. Second, residual policy learning contributes to modifying its steady state error and model inaccuracy of the model-based cascaded controller and substantially improves robustness [25], [28]. Third, this method helps reduce the proportion of signals generated by the black box and facilitates the explainable result analysis of the compensatory signals based on external disturbances and the basic controller output.

In Figure 2, our basic controller consists of cascaded high-level position-attitude controllers and a low-level onboard controller. The motion capture system provides ground truth position, attitude, linear velocity and body rates of the quadcopter at a frequency of $200Hz$, which is then fed back by the state estimator to the high-level controller running on the laptop. After that, it generates a four-dimensional high-level control command at $50Hz$, which includes the

desired thrust and three angular velocities. Our policy network generates a residual command as compensation for the high-level control command, and then we can compute their superposition as the overall command. After that, the overall command is transmitted to the low-level controller running at $500Hz$ on the quadcopter board, which converts the high-level command into the motor speed command for execution. Additionally, we configure a simulated signal of the motion capture system and state estimator at $200Hz$. Both the basic controller and the residual policy generate commands at a frequency of $50Hz$.

B. Training Settings

We train two independent full-enveloped flight policies for both quadcopters with the proximal policy optimization (PPO) [29] algorithm. The policy and value networks are three-layer multilayer perceptrons (MLPs) with 128 units. The first two layers use LeakyReLU [30] as the activation, while the last layer uses the tanh activation to normalize the output. The Adam optimizer [31] is employed for training these networks. For each episode during the training process, we simulate 20 seconds of flight (i.e., 10,000 time steps at the simulation frequency of 500). In the first 15 seconds, the takeoff and hover phases are simulated, followed by the landing phase in the last 5 seconds. At the beginning of each episode, the vehicle’s position is randomly initialized on the ground within a $2m \times 2m$ area, and we set the desired hovering position as $(0, 0, 1.2)$. During the last 5 seconds, the quadcopter is required to descend vertically at a speed of $0.2m/s$. We train the policy for 500 epochs with 10 episodes in each one, which can be finished in 15 minutes on a laptop with a Intel i7-13700H CPU.

1) *Action and Observation Space*: The policy network outputs the action $\mathbf{a}_{res} = \{\mathbf{c}_{res}, \boldsymbol{\omega}_{res}\} \sim \boldsymbol{\pi}_{res} \in \mathbb{R}^4$ that includes the residual thrust and the body rates in three directions. Empirically, we set the range of actions to $[-10m \cdot s^{-2}, 10m \cdot s^{-2}]$ for the mass-normalized thrust and to $[-1rad \cdot s^{-1}, 1rad \cdot s^{-1}]$ for the body rates. The observation information $\mathbf{o} = \{\Delta \mathbf{s}, \mathbf{u}_{cas}, \mathbf{a}_{res}\} \in \mathbb{R}^{20}$ consists of the error between the desired state and current state $\Delta \mathbf{s} = (\mathbf{s}_{des} - \mathbf{s}) \in \mathbb{R}^{12}$, the high-level command $\mathbf{u}_{cas} \in \mathbb{R}^4$ from the basic cascaded controller, and the action from residual module $\mathbf{a}_{res} \in \mathbb{R}^4$.

2) *Reward Shaping*: For a behavior trajectory sampled in the simulator for T time steps, we present it as $\tau = \{\mathbf{s}_0, \mathbf{a}_0, \mathbf{r}_0, \mathbf{s}_1, \mathbf{a}_1, \mathbf{r}_1, \dots, \mathbf{s}_T, \mathbf{a}_T, \mathbf{r}_T\}$. In (2), the optimization objective for RL is to maximize the return of the behavior trajectory, where $p(\tau|\boldsymbol{\pi}_\theta)$ is the likelihood of the trajectory τ under policy $\boldsymbol{\pi}_\theta$ and γ is the decay rate. As shown in (3), we design the following reward signals to encourage the quadcopter to track the position and linear velocity commands while avoiding oscillations. Empirically, we set a vector of scaling factors $\boldsymbol{\alpha} = [-1m^{-1}, -1, -0.01Kg \cdot N^{-1}, -0.1s \cdot rad^{-1}, 1]$ to balance different reward signals and unify them to unitless terms.

$$\mathcal{J}(\boldsymbol{\pi}) = \mathbb{E}_{\tau \sim p(\tau|\boldsymbol{\pi})} \left[\sum_{t=0}^T \gamma^t \mathbf{r}_t \right] \quad (2)$$

$$\mathbf{r}_t = \boldsymbol{\alpha} \cdot [e_{pos}, e_{att}, \mathcal{P}_c, \mathcal{P}_\omega, \mathbf{r}_{\Delta t}]^T \quad (3)$$

- **Position Error Penalty.** We use the L2-norm to set a penalty $e_{pos} = \|p_{des} - p_t\|$ for position error, where p_{des} is the desired position from the planner and p_t is the quadcopter position in the world frame.
- **Attitude Error Penalty.** We compute a unitless attitude error $e_{att} = 3 - \text{trace}(R_t^T \cdot R_{des}) = 2 - 2\cos(\omega)$, where R_t^T and R_{des} are the rotation matrix of the desired and estimated attitude, respectively. $\text{trace}(R_t^T \cdot R_{des})$ is the geodesic distance on $SO(3)$ between the desire and estimated attitude. ω is the smallest rotation angle from the estimated attitude to the desired one.
- **Thrust Command Penalty.** Large thrust command will increase the energy consumption and command oscillation will make motors overheat. To alleviate these problems, We use the sum of two L2-norm terms $\mathcal{P}_c = \|\mathbf{c}_t\| + 2\|\mathbf{c}_t - \mathbf{c}_{t-1}\|$ to punish large thrust command and the oscillation of overall command. We set an empirical factor 2 to scale the oscillation penalty.
- **Body Rate Command Penalty.** Similarly, we set a command penalty $\mathcal{P}_\omega = \|\boldsymbol{\omega}_t\| + 2\|\boldsymbol{\omega}_t - \boldsymbol{\omega}_{t-1}\|$ to punish large body rate commands and oscillation. The factor 2 is set to scale the oscillation penalty as well.
- **Survival Reward.** At each time step, a unitless positive reward signal $\mathbf{r}_{\Delta t} = 0.1$ is sent to encourage any helpful action for survive and avoid risky behaviors.

3) *Randomization*: To avoid the policy from overfitting to a specific configuration, we randomize the model constants of the quadcopter, as detailed in Table I. We set uniform distributed random domains for the mass, inertia, and propeller efficiency. The model randomization leads to the inconsistency between the model parameters set in the basic controller and the real model, which contributes to improving the generalizability of our residual module across different quadcopter dynamics. Due to the larger thrust-weight ratio of the large quadcopter (LQ), we set a $\pm 50\%$ mass error for it while a $\pm 20\%$ error for the small quadcopter (SQ). Additionally, according to the scaling law, the size of a quadcopter correlates positively with its mass and moment of inertia but is not strictly linear. Therefore, in practice, we multiply another random factor sampled from $\mathcal{U}(0.8, 1.2)$ to the mass factor as the overall inertia factor. Mass and Inertia randomization ensures that ProxFly can maintain stable flight even if the model parameters in the basic controller are inaccurate or if a payload is added during its flight.

For the proximity flight, the disturbance primarily comes from the downwash flow generated by the rotors. On the one hand, the downwash flow reduces the propeller efficiency of other quadcopters in the wind field, so the actual thrust decreases compared with it in the static air. On the other hand, quadcopter in wind field experiences external forces and torques acting on the body [6]. To simulate the propeller thrust loss and imbalance caused by downwash flow, we randomly initialize a propeller thrust factor from $\mathcal{U}(0.6, 1.2)$ for four propellers independently. Meanwhile, although we can calculate the force and torque disturbance

TABLE I
THE PARAMETERS OF QUADCOPTER MODELS AND DOMAIN
RANDOMIZATION SETTINGS IN THE TRAINING PROCESS

Parameters	Small Quad.	Large Quad.
Mass (Kg)	0.280	0.850
Mass Factor	$\sim \mathcal{U}(0.8, 1.2)$	$\sim \mathcal{U}(0.5, 1.5)$
Inertia around x, y ($Kg \cdot m^2$)	2.36e-4	5.51e-3
Inertia around z ($Kg \cdot m^2$)	3.03e-4	9.88e-3
Inertia Factor	$\sim \mathcal{U}(0.8, 1.2)$	$\sim \mathcal{U}(0.8, 1.2)$
Arm Length (m)	0.058	0.165
Propeller Thrust Factor ($N/(rad/s)^2$)	1.145e-7	7.640e-6
Thrust Factor	$\sim \mathcal{U}(0.6, 1.2)$	$\sim \mathcal{U}(0.6, 1.2)$
Force Disturbance Period (s)	$\sim \mathcal{U}(2, 8)$	$\sim \mathcal{U}(2, 8)$
Force Disturbance Amplitude on x, y (N)	$\sim \mathcal{U}(0, 0.5)$	$\sim \mathcal{U}(0, 2)$
Force Disturbance Amplitude on z (N)	$\sim \mathcal{U}(0, 2)$	$\sim \mathcal{U}(0, 8)$
Force Truncation in x, y (N)	0.25	1.00
Force Truncation in z (N)	1.00	4.00
Torque Disturbance ($N \cdot m$)	$\sim \mathcal{N}(0, 0.005)$	$\sim \mathcal{N}(0, 0.02)$

with the CFD software or wind tunnel, these methods are resource-intensive, time-consuming, and cannot cover all possible flight scenarios. Therefore, we propose to present the forces in the x, y, z directions as three independent triangular functions with random periods and amplitudes, as shown in Table I. Since downwash flow mainly acts in the vertical direction, the force disturbance in the z direction is more significant. For the torque disturbances around three axes, we simulate them using three independent Gaussian noises. Overall, this simulation approach allows quadcopters to resist airflow disturbances without any communication. In the next section, we conduct the simulation experiment to demonstrate that the policy trained with the proposed noise can resist diverse high-fidelity airflow disturbances.

III. EXPERIMENTS

In this section, we describe our experimental setup, results, and analysis in both simulation and real-world. Overall, we validate our algorithm in simulation by demonstrating its robustness under high-fidelity simulated disturbances. Subsequently, we showcase the performance of ProxFly compared to other advanced algorithms through various real-world flight tests. At last, we demonstrate its potential in challenging aerial quadcopter docking tasks.

A. Simulation Experiments and Analysis

We conduct an experiment in simulator to verify that the proposed disturbances contribute to making the residual module adaptive to high-fidelity simulated disturbances. Based on conclusions from previous research [6], we approximate the effect of real downwash flow as a function of the relative positions and model parameters of two quadcopters. In our setup, the LQ takes off from the ground and hovers at $(0, 0, 1.2)$, while the SQ takes off vertically from $(-1, 0, 0)$ and flies along the positive x -axis at a speed of 0.2 m/s at a certain height. After hovering above the LQ for 5 seconds, the SQ continues along the positive x -axis and lands

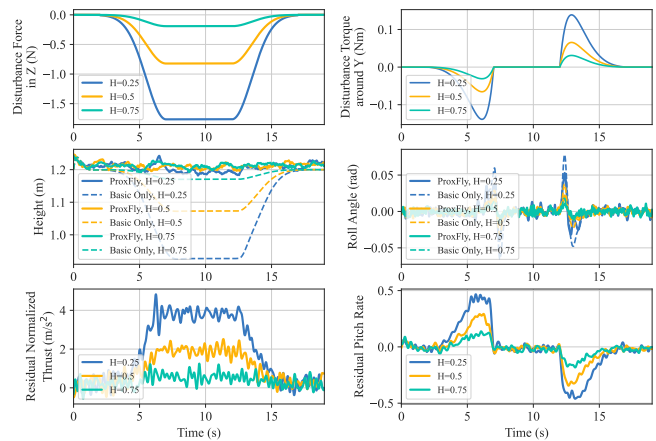


Fig. 3. The results of simulated experiments. **First row:** The external forces in the z direction and torques around the x and y axes from the SQ at three different height differences $H = [0.25, 0.5, 0.75]$ based on the aerodynamics model in [6]. **Second row:** The performance comparison of altitude, roll, and pitch attitude control using the basic controller (denoted as *Basic Only*) and *ProxFly*. **Third row:** The residual commands of mass-normalized thrust, roll rate and pitch rate.

vertically at $(1, 0, 0)$. We set the vertical distance between the SQ and LQ as $\{0.25, 0.5, 0.75\}$ to reveal the effect of downwash flow disturbances with different magnitudes and their corresponding compensatory behaviors generated by the residual module.

As shown in Figure 3, when the SQ approaches along the x -axis, the downwash force on the LQ increases exponentially as the horizontal distance between SQ and LQ decreases and reaches the peak when the SQ hovers directly above the LQ. This procedure also induces a negative torque around the y -axis on the SQ, which first increases and then decreases in magnitude. When the SQ leaves, the change rate of disturbances is reversed compared to the approach phase. In this process, the basic controller can correct the pitch, while it also leads to the steady-state error in altitude control, and the error becomes larger with the downwash flow increasing. On the other hand, the residual module generates thrust and body rate compensation without the relative position between the SQ and LQ. In other words, ProxFly can correct the altitude error and reduce pitch oscillations of the basic controller with only its estimated and desired states. These behaviors are fully learned within our randomized external force and torque, which are fundamentally different from the high-fidelity simulated disturbances. However, these disturbances are derived from complex CFD simulations and wind tunnel experiments that require domain expertise. Conversely, our proposed disturbance is easy to implement.

B. Real-World Experiments and Analysis

1) *Comparative Experiments:* We aim to explain the performance improvements of ProxFly over the basic controller for close proximity flight control through real-world experiments. For this, we set the following tasks.

- **Close Proximity Hovering.** The small quadcopter (SQ) hovers 50cm above the large quadcopter (LQ) for 10 seconds. This experiment demonstrates the robustness

TABLE II
THE RESULTS OF REAL-WORLD COMPARATIVE EXPERIMENTS

Tasks & Metrics		Basic Controller	FB-AeroComp [4]	ProxFly (Ours)
Hovering	E_{pos} (m)	0.1199	0.1113	0.0882
	E_{att} (rad)	0.1710	0.1818	0.0794
Circling (same)	E_{pos} (m)	0.1867	0.0832	0.1385
	E_{att} (rad)	0.1976	0.1238	0.1252
Circling (reversed)	E_{pos} (m)	0.1451	0.0983	0.0940
	E_{att} (rad)	0.1714	0.0930	0.0996
Average	E_{pos} (m)	0.1506	0.0976	0.1069
	E_{att} (rad)	0.1800	0.1329	0.1014

of position and attitude control under continuous downwash flow disturbance.

- **Circling in the same direction.** The SQ flies 50cm above the LQ, and both vehicles track a circular trajectory with a diameter of 1.5m counterclockwise with a period of 7.5 seconds. This experiment evaluates the controller’s robustness and trajectory tracking accuracy under continuous air disturbance.
- **Circling in the reversed direction.** The SQ flies 50cm above the LQ and both vehicles track a circular trajectory with a diameter of 1.5m with a period of 7.5 seconds. The SQ tracks counterclockwise while the LQ tracks clockwise. This experiment evaluates the controller’s robustness and tracking accuracy under sudden air disturbance.

Baselines. First, as an ablation experiment, we evaluate the performance of our basic cascaded controller (denoted as *Basic Controller*) in Table II, which illustrates the performance improvements brought by our residual module. Meanwhile, we compare it with the model-based fine-tuned cascaded controller [6], which can handle complex close proximity flight tasks with the same hardware used in this paper. Technically, it first models the aerodynamics of quadcopters via both CFD software and real-world verification, and then calculates the thrust compensation from the relative position between quadcopters. Here, we denoted it as *FB-AeroComp*.

Metrics. We compare the performance of different controllers based on the tracking accuracy of position and attitude. For position accuracy, we evaluate it by the root mean square error (RMSE) between the estimated and desired positions E_{pos} , as shown in (4). Meanwhile, similar to the reward shaping, for attitude accuracy, we use the RMSE of the minimum rotational angle E_{att} between the estimated and desired attitude as (5).

$$E_{pos} = \sqrt{\frac{1}{T} \sum_{t=1}^T (p_{des} - p_t)^2} \quad (4)$$

$$E_{att} = \sqrt{\frac{1}{T} \sum_{t=1}^T \left[\cos^{-1} \left(\frac{\text{trace}(R_t^T \cdot R_{des}) - 1}{2} \right) \right]^2} \quad (5)$$

Analysis. Compared to the basic controller, ProxFly demonstrates significant performance improvements across all three tasks. Specifically, the residual module reduces the averaged RMSE of position control by 29.0% and averaged attitude error by 43.7% across the three tasks. In the circling in

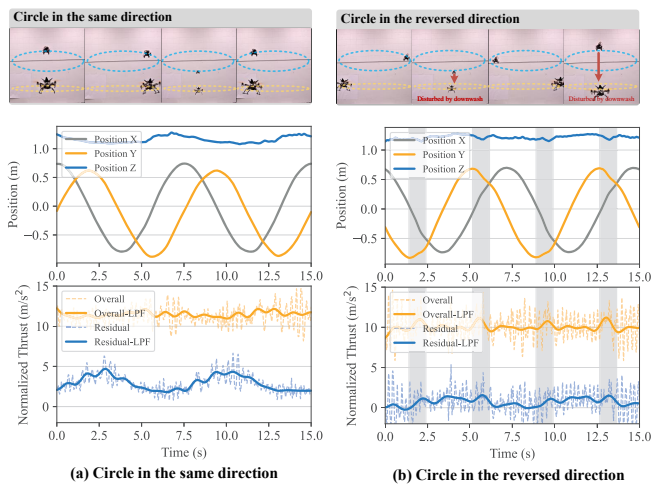


Fig. 4. **The demonstration of circular trajectory tracking.** (a) Two quadcopters are tracking the circular trajectory counterclockwise. The residual module generates positive thrust commands to compensate the thrust loss and downward force caused by downwash flow. (b) Two quadcopters are tracking in reversed directions. When the small quadcopter is passing above the large one, the controller on the large quadcopter increases the thrust for about 1s for compensation. (*LPF: low-pass filtered*)

the same direction and hovering tasks, these improvements result from the residual commands correcting the steady-state errors caused by continuous external aerodynamic disturbances. For the circling in the reversed direction task, the residual module allows the LQ to recover more quickly to the desired altitude after a sudden downward impulse. Meanwhile, compared to FB-AeroComp with complex aerodynamic modeling, ProxFly achieves comparable performance. In the hovering task, ProxFly reduces position error by 20.8% and attitude error by 56.3% compared to FB-AeroComp. For circular trajectory tracking tasks, as shown in Figure 4, the residual module of ProxFly compensates for the downward force and thrust loss when the downwash effect happens. For example, when two quadcopters circle in the same direction, the residual module generates the lasting positive thrust command for compensation. While circling in two reversed directions, every time when the SQ flies over the LQ, the residual module generates a temporary thrust compensation. Our method achieves the comparative performance with FB-AeroComp. Note that ProxFly is mainly trained for hovering rather than trajectory tracking. In these tasks, we implement trajectory tracking by continuously changing the hovering position, and ProxFly is highly sensitive to position and attitude errors, which leads to more aggressive actions. Meanwhile, the output commands still oscillate at a high frequency, although we have set a penalty for it during reward shaping. However, our approach does not depend on any prior assumptions of accurate models and any forms of communication between quadcopters. Essentially, ProxFly adapts to various disturbances by using a residual module to correct the behaviors from a simple controller, which simplifies the time-consuming procedure of dynamics modeling and controller fine-tuning.

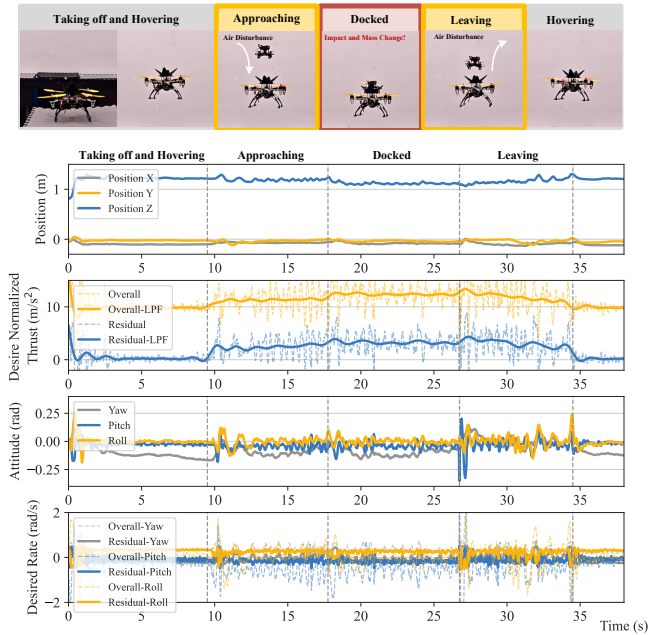


Fig. 5. **The procedure of quadcopter in-air docking.** **Stage 1: Taking off and hovering.** The residual RL controller only assists the basic controller on the large quadcopter (LQ) to achieve faster responses. **Stage 2: Small quadcopter approaching.** Small quadcopter (SQ) hovers above the LQ and vertically approaches it, which generates strong downwash flow disturbances, and the residual RL controller generates thrust and rate compensation to help stabilize LQ’s position and attitude. (*LPF: low-pass filtered*) **Stage 3: Docked with LQ.** The SQ falls freely from 5cm above the LQ, gets docked with the LQ and generates an impulse. The overall mass changes and the thrust compensation from the RL controller on LQ reaches the peak. **Stage 4: SQ leaving.** SQ takes off from the LQ vertically, and the downwash airflow reappears and gradually decreases while the thrust compensation from the RL controller also gradually decreases.

2) *Quadcopter Aerial Docking:* In-air charging to achieve infinite flight time can be accomplished through quadcopter docking [4]. Due to the extreme proximity between the quadcopters, strong and complex local vortices are generated and cause severe thrust loss in both the LQ and SQ, which makes the aerodynamic modeling and control for docking highly challenging. Additionally, when the SQ lands on top of the LQ, it causes an impulse and a permanent change in dynamics. However, our ProxFly can adapt to unknown impulses and payloads and achieves robust position and attitude control. The LQ first takes off and hovers at $(0, 0, 1.2)$. Then, the SQ approaches directly above the LQ from $(0, -1, 1.7)$ and then descends vertically at a speed of 0.1m/s until it is 10cm above the LQ. When the horizontal distance between the two quadcopters is less than 5cm , the SQ shuts down its motors and falls freely onto the top of the LQ, and the docking is finished. The SQ exerts an impulse on the LQ and the overall mass changes. When the undocking command is received, the SQ turns on the motors and rakes off vertically, where the thrust loss caused by downwash flow reappears.

The position, attitude, and controller outputs of the LQ are shown in Figure 5. As the SQ approaches the LQ, the downwash flow affects the propeller’s efficiency on one side of the LQ and generates a downward force and a roll torque disturbance. At this point, the residual module generates an

instantaneous body rate command in the opposite direction of the roll to offset the rotation. When the SQ approaches vertically, the downward force disturbance and thrust loss acting on the LQ gradually increase. The residual module then produces a correspondingly increasing compensatory thrust command to help the LQ maintain its altitude. When the SQ gets docked with the LQ, the magnitude of the residual thrust command becomes stable. Note that the preset model parameters in the basic controller have significant errors at this point, and docking would fail if only the model-based basic controller were used. We manually send the undocking signal to the SQ via a joystick. As the undocking begins, the downwash flow and complex local vortices gradually intensify, leading to the thrust loss and imbalance of the LQ once again. At this point, the LQ experiences severe roll and pitch oscillations, and the residual module therefore generates commands opposite to the roll and pitch directions to stabilize the body. As the SQ gradually moves away from the LQ, the downward force disturbance acting on the SQ decreases. Consequently, the thrust commands generated by the residual module also gradually reduce to zero.

IV. CONCLUSION

This paper proposes the ProxFly, which leverages a cascaded controller combined with residual RL for diverse close proximity tasks, including hovering, trajectory tracking, in-air docking, and undocking. This approach does not rely on any communication or accurate knowledge of system parameters but only uses a lightweight MLP to learn how to compensate for high-level commands. ProxFly leverages the strengths of both classical control and deep RL. On the one hand, it requires less domain expertise and avoids time-consuming controller parameter tuning. On the other hand, compared to end-to-end RL, it improves sampling efficiency and reduces the proportion of unexplained signals from the black box in the overall control signals.

However, ProxFly’s residual module can potentially cause oscillations, which might lead to motor overheating. Meanwhile, although ProxFly relaxes the requirement of accurate modeling and is adaptive to different systems, it still needs to keep the system parameters in a reasonable range. Finally, due to hardware limitations, we only validated the performance on two regular-sized quadcopters. More types of quadcopters and experiments are needed to verify the generality of the conclusions in this paper.

ACKNOWLEDGEMENT

This work is supported by Hong Kong Center for Logistics Robotics. We thank William Su from the department of aerospace engineering, University of California, Berkeley for his design and modification of the quadcopter docking platform. The experimental testbed at the HiPer Lab is the result of contributions of many people, a full list of which can be found at <https://hiperlab.berkeley.edu/members/>.

REFERENCES

- [1] J. A. Preiss, W. Honig, G. S. Sukhatme, and N. Ayanian, "Crazyswarm: A large nano-quadcopter swarm," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3299–3304, 2017.
- [2] G. Vásárhelyi, C. Virágh, G. Somorjai, T. Nepusz, A. E. Eiben, and T. Vicsek, "Optimized flocking of autonomous drones in confined environments," *Science Robotics*, vol. 3, no. 20, p. eaat3536, 2018.
- [3] A. Shankar, S. Elbaum, and C. Detweiler, "Dynamic path generation for multirotor aerial docking in forward flight," in *2020 59th IEEE Conference on Decision and Control (CDC)*, pp. 1564–1571, 2020.
- [4] K. P. Jain and M. W. Mueller, "Flying batteries: In-flight battery switching to increase multirotor flight time," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3510–3516, 2020.
- [5] H. Smith, A. Shankar, J. Gielis, J. Blumenkamp, and A. Prorok, "So(2)-equivariant downwash models for close proximity flight," *IEEE Robotics and Automation Letters*, vol. 9, no. 2, pp. 1174–1181, 2024.
- [6] K. P. Jain, T. Fortmuller, J. Byun, S. A. Mäkiharju, and M. W. Mueller, "Modeling of aerodynamic disturbances for proximity flight of multirotors," in *2019 International Conference on Unmanned Aircraft Systems (ICUAS)*, pp. 1261–1269, 2019.
- [7] J. Panerati, H. Zheng, S. Zhou, J. Xu, A. Prorok, and A. P. Schoellig, "Learning to fly—a gym environment with pybullet physics for reinforcement learning of multi-agent quadcopter control," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 7512–7519, 2021.
- [8] Y. Su, C. Chu, M. Wang, J. Li, L. Yang, Y. Zhu, and H. Liu, "Downwash-aware control allocation for over-actuated uav platforms," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 10478–10485, 2022.
- [9] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.
- [10] S. Ross, N. Melik-Barkhudarov, K. S. Shankar, A. Wendel, D. Dey, J. A. Bagnell, and M. Hebert, "Learning monocular reactive uav control in cluttered natural environments," in *2013 IEEE International Conference on Robotics and Automation*, pp. 1765–1772, 2013.
- [11] A. Loquercio, A. I. Maqueda, C. R. del Blanco, and D. Scaramuzza, "Dronet: Learning to fly by driving," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 1088–1095, 2018.
- [12] 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 International Conference on Robotics and Automation (ICRA)*, pp. 9784–9790, 2019.
- [13] E. Kaufmann, L. Bauersfeld, A. Loquercio, M. Müller, V. Koltun, and D. Scaramuzza, "Champion-level drone racing using deep reinforcement learning," *Nature*, vol. 620, no. 7976, pp. 982–987, 2023.
- [14] 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)*, pp. 3241–3247, 2020.
- [15] 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, 2022.
- [16] D. Zhang, A. Loquercio, X. Wu, A. Kumar, J. Malik, and M. W. Mueller, "Learning a single near-hover position controller for vastly different quadcopters," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1263–1269, 2023.
- [17] J. Eschmann, D. Albani, and G. Loianno, "Learning to fly in seconds," *IEEE Robotics and Automation Letters*, vol. 9, no. 7, pp. 6336–6343, 2024.
- [18] R. Ferede, C. De Wagter, D. Izzo, and G. C. de Croon, "End-to-end reinforcement learning for time-optimal quadcopter flight," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6172–6177, 2024.
- [19] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.
- [20] S. Belkhal, R. Li, G. Kahn, R. McAllister, R. Calandra, and S. Levine, "Model-based meta-reinforcement learning for flight with suspended payloads," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1471–1478, 2021.
- [21] R. Zhang, J. Hou, F. Walter, S. Gu, J. Guan, F. Röhrbein, Y. Du, P. Cai, G. Chen, and A. Knoll, "Multi-agent reinforcement learning for autonomous driving: A survey," *arXiv preprint arXiv:2408.09675*, 2024.
- [22] J. Qin, Q. Ma, Y. Shi, and L. Wang, "Recent advances in consensus of multi-agent systems: A brief survey," *IEEE Transactions on Industrial Electronics*, vol. 64, no. 6, pp. 4972–4983, 2017.
- [23] B. Beyret, A. Shafti, and A. A. Faisal, "Dot-to-dot: Explainable hierarchical reinforcement learning for robotic manipulation," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5014–5019, 2019.
- [24] T. Johannink, S. Bahl, A. Nair, J. Luo, A. Kumar, M. Loskyll, J. A. Ojea, E. Solowjow, and S. Levine, "Residual reinforcement learning for robot control," in *2019 International Conference on Robotics and Automation (ICRA)*, pp. 6023–6029, 2019.
- [25] T. Silver, K. Allen, J. Tenenbaum, and L. Kaelbling, "Residual policy learning," *arXiv preprint arXiv:1812.06298*, 2018.
- [26] N. Bucki, J. Lee, and M. W. Mueller, "Rectangular pyramid partitioning using integrated depth sensors (rappids): A fast planner for multicopter navigation," *IEEE Robotics and Automation Letters*, vol. 5, no. 3, pp. 4626–4633, 2020.
- [27] R. Zhang, J. Hou, G. Chen, Z. Li, J. Chen, and A. Knoll, "Residual policy learning facilitates efficient model-free autonomous racing," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 11625–11632, 2022.
- [28] R. Zhang, G. Chen, J. Hou, Z. Li, and A. Knoll, "Pipo: Policy optimization with permutation-invariant constraint for distributed multi-robot navigation," in *2022 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, pp. 1–7, 2022.
- [29] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.
- [30] B. Xu, N. Wang, T. Chen, and M. Li, "Empirical evaluation of rectified activations in convolutional network," *arXiv preprint arXiv:1505.00853*, 2015.
- [31] D. P. Kingma, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.