This paper has been accepted for publication at Robotics: Science and Systems, 2026

Learning Agile Quadrotor Flight in the Real World

Yunfan Ren, Zhiyuan Zhu, Jiaxu Xing, Davide Scaramuzza
Robotics and Perception Group, University of Zurich, Switzerland

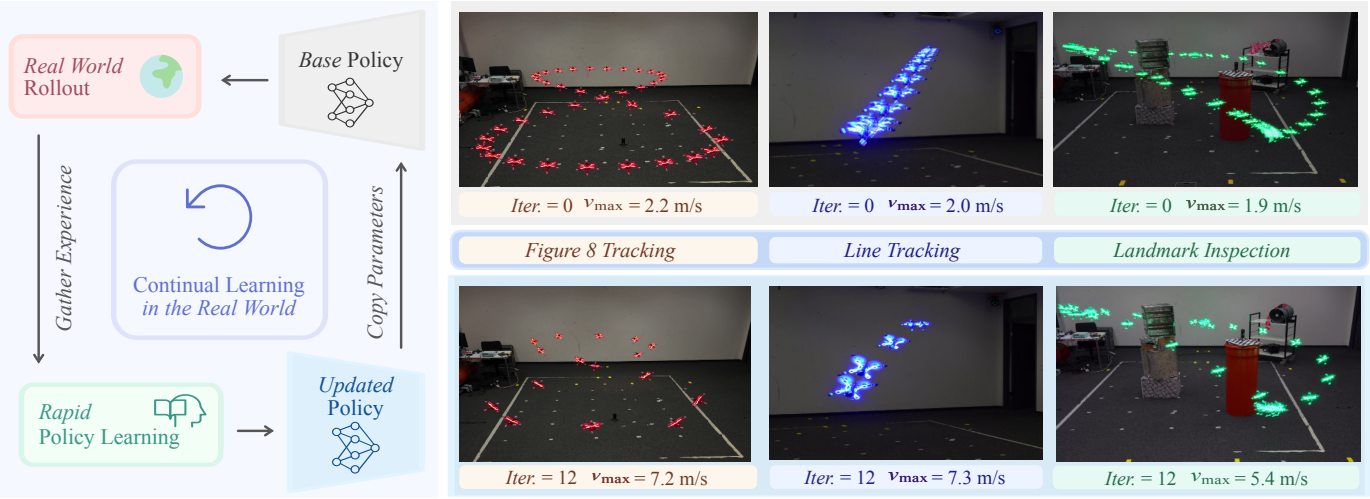

Fig. 1: **Autonomous online evolution of agile flight.** The figure illustrates the continuous loop between *Real-World Rollouts* and *Rapid Policy Learning*, comparing the initial conservative flights (Iter 0, top) with the evolved agile behaviors (Iter 12, bottom). Trajectories are composited with frames sampled at uniform 0.5 s intervals. Consequently, trajectory density reflects flight speed, where sparser segments indicate higher velocity. Driven by this online adaptation, the system pushes its physical limits without human intervention, tripling its speed within 100 s of flight time across three tasks.

Abstract*—Learning-based controllers have achieved impressive performance in agile quadrotor flight but typically rely on large-scale simulation training, necessitating accurate system identification for effective sim-to-real transfer. However, even with precise modeling, fixed policies remain susceptible to out-of-distribution scenarios, ranging from external aerodynamic disturbances to internal hardware degradation. To ensure safety under these evolving uncertainties, such controllers are forced to operate with conservative safety margins, inherently constraining their agility outside of controlled settings. While online adaptation offers a potential remedy, safely pushing the platform toward its agility envelope remains a critical bottleneck due to data scarcity and safety risks. To bridge this gap, we propose a self-adaptive framework that eliminates the need for precise system identification or offline sim-to-real transfer. We introduce Adaptive Temporal Scaling (ATS) to actively push the platform toward the agility ceiling exposed by its control interface, and employ online residual learning to augment a simple nominal model. Based on the learned hybrid model, we further propose Real-World Anchored Short-Horizon Backpropagation Through Time (RASH-BPTT) to achieve efficient and robust in-flight policy updates. Extensive experiments demonstrate that our quadrotor reliably executes agile maneuvers near the saturation limit of the CTBR command interface. The system evolves a conservative base policy from a peak speed of $2.0\,\mathrm{m\,s^{-1}}$ to $7.3\,\mathrm{m\,s^{-1}}$ within approximately $100\,\mathrm{s}$ of flight time. These findings underscore that real-world adaptation serves not merely to compensate for modeling errors, but as a practical mechanism for sustained performance improvement in aggressive flight regimes.

Supplementary Materials

Website: https://rpg.ifi.uzh.ch/lafr/
Code: https://github.com/uzh-rpg/LAFR

I. Introduction

The domain of agile control has witnessed a paradigm shift with the advent of data-driven policies, which now rival or exceed the capabilities of classical model-based methods [1, 2, 3, 4, 5, 6]. Inspired by human skill acquisition, most existing methods rely on imitation learning from expert demonstrations [7, 8, 9] or reinforcement learning through large-scale trial-and-error [3, 5, 6]. In practice, these policies are typically trained in simulation with extensive domain randomization [10] to approximate real-world variability and are kept fixed during real-world deployment. As a consequence, once deployed, the controller relies entirely on prior training experience rather than learning and adapting in the real world. More broadly, learning directly through real-world experience has begun to emerge at the task-planning level [11]. However, the capability to continually improve low-level control through real-world interaction, which is central to human motor learning, remains largely underexplored.

This limitation becomes especially critical in the context of agile quadrotor flight. Achieving high-speed, aggressive maneuvers requires precise control under tightly coupled and highly nonlinear dynamics, leaving little margin for model mismatch or unknown external disturbances [12].

However, real-world flight conditions are inherently non-stationary. Factors such as hardware wear, battery depletion, and complex aerodynamic effects introduce inevitable discrepancies between the training model and physical reality that fixed policies cannot mitigate. Consequently, relying on a fixed policy directly degrades performance and reliability,

often preventing the system from safely reaching its agility potential. In contrast, enabling controllers to learn directly in the real world fundamentally expands the utility of these systems. This capability empowers the agent to adapt not only to external environmental shifts but also to internal hardware variations, continuously exploiting platform-specific properties to push performance toward the agility ceiling imposed by the control interface without requiring expensive offline system identification [13].

However, learning directly in the real world introduces dual challenges of safety and data scarcity. First, agile quadrotors operate near the limits of stability, where even minor deviations in state or control can trigger catastrophic failure. Second, learning in the physical world demands high sample efficiency because data collection is costly and constrained by limited battery life, permitting only a few interactions. Consequently, a safe, adaptive, and efficient approach is required.

Existing work has made significant progress toward learning real-world agile quadrotor control. Sim-to-real approaches [14] based on extensive domain randomization [10] expose policies to diverse dynamics and disturbances during training, enabling them to tolerate moderate modeling errors. While effective in controlled settings, their performance can degrade significantly when real-world conditions deviate from the randomized training distribution. To further reduce the sim-to-real gap, Real2Sim2Real pipelines refine the simulation model using real-world data and retrain the policy offline before deployment [4, 15, 16]. Although effective, these methods require substantial data collection and repeated retraining cycles, which limit their practicality when system dynamics evolve or rapid adaptation is required. Recent advancements have begun to leverage differentiable simulation for rapid real-world adaptation of quadrotor control policies, notably in [17]. However, while this method facilitates rapid refinement during deployment, it prioritizes disturbance rejection over pushing the platform toward its agility envelope. Consequently, the resulting performance remains conservative, typically around $1\,\mathrm{m\,s^{-1}}$ to $2\,\mathrm{m\,s^{-1}}$, making the policy unsuitable for agile, high-performance behaviors.

Contributions

In this work, we propose a self-adaptive framework that bridges the gap between robustness and performance, enabling continuous, on-policy improvement directly in the real world without the need for precise system identification. Our approach relies on two key mechanisms to address the challenges of safe exploration and efficient learning. First, to tackle the safety-exploration paradox, we introduce *Adaptive Temporal Scaling (ATS)*. Unlike fixed-task baselines, ATS actively incentivizes the quadrotor to explore its physical capability boundaries by dynamically adjusting the aggressiveness of the reference trajectory based on the agent's current competence. Second, to ensure efficient learning, we employ a hybrid dynamics model that augments a simple nominal model with online residual learning. By introducing *Real-World Anchored Short-Horizon Backpropagation Through Time (RASH-BPTT)* using these hybrid dynamics, we achieve on-the-fly policy optimization, effectively turning limited flight data into sustained performance gains.

Through extensive real-world experiments, we demonstrate that our framework allows a quadrotor to reliably learn agile maneuvers near the saturation limit of the CTBR command interface. Empirically, the system evolves a conservative base policy from a peak speed of $2.0\,\mathrm{m\,s^{-1}}$ to $7.3\,\mathrm{m\,s^{-1}}$ within approximately $100\,\mathrm{s}$ of flight time. Beyond pure agility, we demonstrate the practical utility of our method in a multi-point inspection mission under strong external wind disturbances (Fig. 1). Notably, after only $2\,\mathrm{min}$ of real-world training, the quadrotor reduces mission completion time by 42% while maintaining low tracking error under unknown aerodynamic disturbances. These results underscore that real-world adaptation serves not merely to compensate for modeling errors, but as a practical mechanism for aggressive performance improvement in dynamic environments.

II. Related Work

A. Learning-based Agile Flight Control

In recent years, learning-based approaches have gained significant attention for agile quadrotor flight control, owing to their ability to handle highly nonlinear dynamics and operate under aggressive flight conditions. Early work demonstrated the feasibility of applying reinforcement learning to quadrotor control, enabling waypoint tracking and recovery from challenging initial conditions [18]. Subsequent research showed that reinforcement learning can support highly agile behaviors, with RL-based controllers achieving state-of-the-art performance in autonomous drone racing [3, 19]. In these settings, learning-based controllers have even surpassed expert human performance in competitive racing scenarios [3, 19]. Advances in training efficiency further indicate that, under carefully optimized pipelines, reinforcement learning policies for quadrotor control can be trained within seconds [20]. Beyond reinforcement learning, imitation learning has played an important role in accelerating training and improving stability for agile flight. By leveraging expert demonstrations, imitation learning has been applied both as a standalone approach and as an initialization or regularization strategy for reinforcement learning, substantially improving sample efficiency and overall control performance [21, 9]. Despite these impressive results, learning-based quadrotor controllers are typically trained offline and remain fixed during deployment, limiting their ability to adapt to changing real-world conditions.

B. Efficient Control Policy Learning

Traditional reinforcement learning methods often require extensive training time, though significant acceleration is possible through highly optimized physics simulation [20]. Despite these improvements, these methods remain sample-inefficient due to the high variance associated with zeroth-order policy gradient estimation. An alternative paradigm is policy learning via differentiable simulation, which exploits smooth and differentiable dynamics and reward formulations

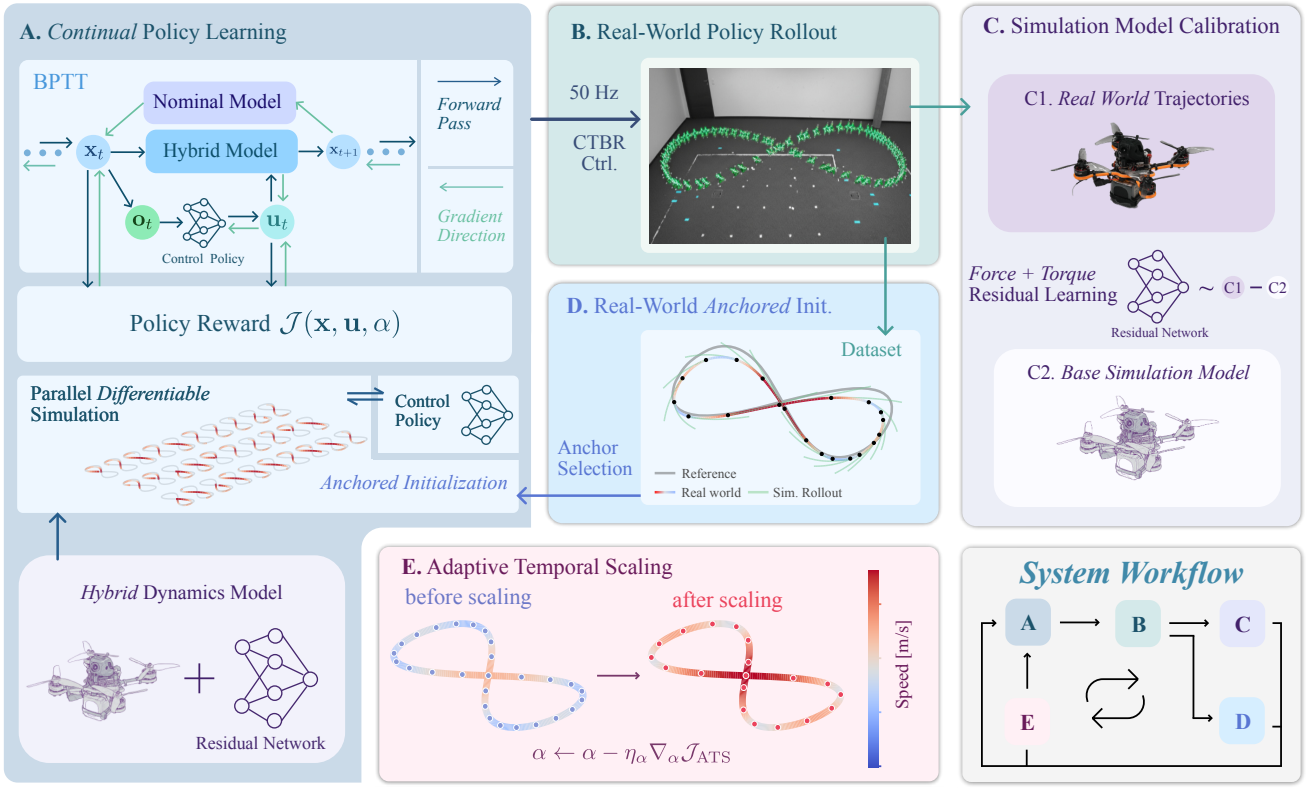

Fig. 2: **Overview of the self-adaptive autonomous flight framework.** The system operates as a continuous closed-loop cycle (bottom right) bridging physical execution and differentiable simulation: **(A) Policy Learning:** Leveraging a learned hybrid dynamics model in a differentiable simulator, we perform **RASH-BPTT** to optimize the control policy via massively parallelized rollouts. **(B) Real-World Rollout:** The agent executes the current policy on the physical quadrotor to collect state-action-transition data. **(C) Model Calibration:** Collected data is used to update the *Hybrid Dynamics Model* online, where a neural residual network learns to compensate for the reality gap (e.g., unmodeled aerodynamics, delays) of the nominal rigid-body model. **(D) Anchored Initialization:** To mitigate compounding prediction errors, simulation rollouts are initialized (*anchored*) using the most recent real-world state estimates rather than random resets. **(E) Adaptive Temporal Scaling (ATS):** Integrated directly with policy optimization, the trajectory time-scale α is jointly optimized using real-world rollouts. By leveraging analytical gradients derived from closed-loop sensitivity, ATS maximizes agility while enforcing safety constraints via a barrier function.

to compute *first-order* policy gradients. This enables substantially faster and more sample-efficient learning compared to standard RL methods [22]. Differentiable simulation has been successfully applied to direct policy parameterizations, including parametric trajectory representations for underwater [23] and sinusoidal control policies for robotic cutting tasks [24]. However, extending these techniques to neural network policies remains challenging. In practice, unstable gradients often limit applicability to short-horizon tasks with simplified contact dynamics and restricted variation in initial conditions [25]. To address these limitations, prior work has explored a range of stabilizing techniques, including early termination of simulations at contact events, truncated backpropagation through time [26], and reward shaping or augmentation with learned critics [27, 28]. However, these methods have yet to effectively address the challenge of safely pushing the platform toward its agility envelope to enable online learning of agile flight.

III. Methodology

In this section, we introduce our self-adaptive framework. As illustrated in Fig. 2, the core workflow operates as a continuous closed-loop cycle bridging physical execution and differentiable simulation. The process begins with an initialization phase, where we employ standard BPTT to train a base policy on the nominal quadrotor model. Benefiting from the sample efficiency of differentiable simulation, this pre-training typically converges within seconds. Once deployed, the agent commences the real-world adaptation loop. First, the system collects flight data to update the *Hybrid Dynamics Model*, using a neural residual network to capture both internal dynamic discrepancies and external disturbances. Subsequently, leveraging this calibrated model, we execute *RASH-BPTT* to optimize the policy via massively parallelized rollouts. To mitigate compounding prediction errors, these rollouts utilize *Anchored Initialization*, resetting the simulation to the most recent real-world state estimates. Tightly coupled with policy optimization, *Adaptive Temporal Scaling (ATS)* is applied to dynamically adjust the reference trajectory evolution. By optimizing a learnable parameter α through analytical gradients derived from closed-loop sensitivity directly on real-world rollouts, the system maximizes agility while maintaining safety constraints.

The framework consists of three hierarchical components:

1) *Online Residual Learning*: Calibrates the online hybrid dynamics model to compensate for the reality gap using

real-world data (Section III-A).
2) *RASH-BPTT*: Performs online policy optimization via differentiable simulation, anchored to the current physical state to ensure robust updates (Section III-B).
3) *Adaptive Temporal Scaling (ATS)*: Jointly optimizes the reference trajectory's time evolution to balance flight agility and safety (Section III-C).

A. Online Residual Dynamics Learning

To avoid precise offline system identification and to adapt to external disturbances and platform-specific discrepancies, we employ a hybrid dynamics model. Specifically, we learn a neural residual term that augments the nominal rigid-body dynamics and is trained by minimizing one-step state prediction error, inspired by Neural ODEs [29].

1) Hybrid Continuous-Time Dynamics: We define the quadrotor state as

$$\mathbf{x} \triangleq (\mathbf{p}, \mathbf{v}, \mathbf{R}) \in \mathbb{R}^3 \times \mathbb{R}^3 \times \mathrm{SO}(3),$$

where $\mathbf{p}$ and $\mathbf{v}$ are the position and velocity expressed in the world frame, and $\mathbf{R}$ maps vectors from the body frame to the world frame. Let $\mathbf{e}_3 \triangleq [0,0,1]^\top$ denote the third canonical basis vector; thus, $\mathbf{R}\mathbf{e}_3$ is the body-frame thrust axis expressed in the world frame. Throughout the paper, gravity is taken to act along $-g\mathbf{e}_3$. The control input is a collective-thrust/body-rate (CTBR) command $\mathbf{u} \triangleq \begin{bmatrix} c & \boldsymbol{\omega}_{\mathrm{cmd}}^\top \end{bmatrix}^\top \in \mathcal{U}$, with admissible input set

$$\mathcal{U} \triangleq [c_{\min}, c_{\max}] \times \left\{ \boldsymbol{\omega} \in \mathbb{R}^3 \,\middle|\, \|\boldsymbol{\omega}\|_\infty \le \omega_{\max} \right\}.$$

Here, c is the mass-normalized collective thrust, with units of $\mathrm{m\,s^{-2}}$, and $\boldsymbol{\omega}_{\mathrm{cmd}} \in \mathbb{R}^3$ is the commanded body rate. The infinity-norm constraint imposes an axis-wise saturation on the commanded body rate. We use the CTBR as an abstraction over individual rotor thrusts: it simplifies policy learning and leverages a well-tuned low-level rate controller, while making the bound on $\|\boldsymbol{\omega}_{\mathrm{cmd}}\|_\infty$ a controller-level command ceiling rather than a fundamental hardware limit.

We model the continuous-time quadrotor dynamics as a nominal CTBR model augmented with neural residual terms:

$$\dot{\mathbf{p}} = \mathbf{v}, \tag{1a}$$

$$\dot{\mathbf{v}} = c\,\mathbf{R}\mathbf{e}_3 - g\mathbf{e}_3 + \mathbf{a}_{\mathrm{res}}(\boldsymbol{\zeta};\theta), \tag{1b}$$

$$\dot{\mathbf{R}} = \mathbf{R}\left(\boldsymbol{\omega}_{\mathrm{cmd}} + \boldsymbol{\omega}_{\mathrm{res}}(\boldsymbol{\zeta};\theta)\right)^\wedge. \tag{1c}$$

This defines the hybrid vector field $\mathbf{f}_{\mathrm{hybrid}}(\mathbf{x},\mathbf{u};\theta)$. Here, $(\cdot)^\wedge : \mathbb{R}^3 \to \mathfrak{so}(3)$ denotes the skew-symmetric map to the Lie algebra $\mathfrak{so}(3)$, $\boldsymbol{\zeta}$ denotes the residual-model input features, and θ denotes the residual-model parameters. The residual terms $\mathbf{a}_{\mathrm{res}} \in \mathbb{R}^3$ and $\boldsymbol{\omega}_{\mathrm{res}} \in \mathbb{R}^3$ capture unmodeled translational accelerations and body-rate discrepancies, respectively. This formulation assumes that the low-level body-rate loop tracks CTBR commands sufficiently fast, so that its dominant tracking errors can be represented as additive residuals on the commanded body rates.

The residual network $\mathbf{f}_{\mathrm{res}}(\boldsymbol{\zeta};\theta)$ takes the feature vector

$$\boldsymbol{\zeta} \triangleq [\mathbf{p}^\top,\ \mathbf{v}^\top,\ \mathrm{vec}(\mathbf{R})^\top,\ \mathbf{u}^\top]^\top, \tag{2}$$

where $\mathrm{vec}(\cdot)$ denotes column-stacking vectorization, and outputs $\mathbf{f}_{\mathrm{res}}(\boldsymbol{\zeta};\theta) = \begin{bmatrix} \mathbf{a}_{\mathrm{res}}^\top,\ \boldsymbol{\omega}_{\mathrm{res}}^\top \end{bmatrix}^\top$.

2) Differentiable Integration and Online Training: To obtain a discrete-time transition, we employ a differentiable fourth-order Runge–Kutta (RK4) integrator Φ_{RK4}. Beyond improved numerical accuracy, RK4 is important for our one-step residual learning: its intermediate stages partially advance $\mathbf{R}$ within the step, making the translational prediction $(\mathbf{p}_{k+1}, \mathbf{v}_{k+1})$ sensitive to $\boldsymbol{\omega}_{\mathrm{res}}$ and thus allowing position/velocity errors to supervise attitude-related residuals. We define the discrete dynamics map

$$\mathbf{x}_{k+1} = \mathbf{F}_{\Delta t}(\mathbf{x}_k, \mathbf{u}_k; \theta) \triangleq \Phi_{\mathrm{RK4}}(\mathbf{x}_k, \mathbf{u}_k, \mathbf{f}_{\mathrm{hybrid}}; \Delta t). \tag{3}$$

In implementation, Φ_{RK4} is realized as a Lie-group integrator that updates $\mathbf{R}$ on $SO(3)$ using the exponential map, which prevents attitude drift and preserves $\mathbf{R} \in SO(3)$ while remaining fully differentiable.

We optimize parameters θ online using a sliding-window replay buffer $\mathcal{B}$ of recent transitions $(\mathbf{x}_k, \mathbf{u}_k, \mathbf{x}_{k+1})$ collected from the state estimator. Instead of explicitly supervising acceleration targets (which are often noisy), we minimize the *integrated* one-step prediction error between the simulated next state $\hat{\mathbf{x}}_{k+1} = \mathbf{F}_{\Delta t}(\mathbf{x}_k, \mathbf{u}_k; \theta)$ and the measured state $\mathbf{x}_{k+1}$:

$$\mathcal{L}_{\mathrm{res}}(\theta) = \frac{1}{|\mathcal{B}|} \sum_{k \in \mathcal{B}} \mathcal{D}(\mathbf{x}_{k+1}, \hat{\mathbf{x}}_{k+1}) + \lambda_{\mathrm{reg}} \sum_l \|\mathbf{W}_l\|_\sigma. \tag{4}$$

The discrepancy metric $\mathcal{D}(\cdot,\cdot)$ combines the Euclidean translation error and the geodesic distance on the rotation manifold. The rotational term is formulated as $\|\mathrm{Log}(\hat{\mathbf{R}}^\top \mathbf{R})^\vee\|_2^2$, where $\mathrm{Log} : SO(3) \to \mathfrak{so}(3)$ denotes the logarithmic map to the Lie algebra, and $(\cdot)^\vee$ maps the skew-symmetric matrix to its corresponding vector in $\mathbb{R}^3$. This ensures the loss strictly respects the geometry of $SO(3)$. The regularization term penalizes the spectral norm $\|\mathbf{W}_l\|_\sigma$ of each network layer. As motivated by [30], constraining the Lipschitz constant of the residual dynamics is crucial for bounding gradient variance and ensuring numerical stability during long-horizon recursive optimization.

B. Real-World Anchored Short-Horizon BPTT

Building on the learned hybrid dynamics, we propose *Real-World Anchored Short-Horizon BPTT (RASH-BPTT)*. To address *model exploitation* caused by compounding errors [31], our method *anchors* each rollout to the instantaneous real-world state. Furthermore, by restricting optimization to a *short horizon* (Fig. 2D), we prevent error accumulation, ensuring that policy gradients remain grounded in reality and optimized specifically for the agent's current dynamic conditions.

We parameterize the control policy π_ϕ as a multilayer perceptron (MLP) with two hidden layers of 256 units. The policy takes an observation vector $\mathbf{o}_k = [\mathbf{x}_k, \mathbf{x}_{\mathrm{ref},k}, \mathbf{h}_k]$, composed of the current state, the reference trajectory state, and a history of past actions $\mathbf{h}_k$ to encourage control smoothness. The output is a four-dimensional CTBR command $\mathbf{u}_k \in \mathbb{R}^4$.

At each real-world time step t, we initialize the differentiable rollout with the current state estimate $\mathbf{x}_0 := \mathbf{x}_{\mathrm{real}}(t)$.

The policy generates control actions conditioned on the current rollout state $\mathbf{u}_k := \pi_\phi(\mathbf{o}_k)$. We then unroll the short-horizon dynamics for H steps using the learned discrete map from Eq. (3):

$$\mathbf{x}_{k+1} = \mathbf{F}_{\Delta t}(\mathbf{x}_k, \mathbf{u}_k; \theta), \quad k = 0, \dots, H-1. \tag{5}$$

The optimization objective is to maximize the cumulative discounted reward along this predicted trajectory:

$$\mathcal{J}(\phi) = \sum_{k=0}^{H-1} \gamma^k \, r(\mathbf{x}_k, \mathbf{u}_k, \mathbf{x}_{\text{ref},k}), \tag{6}$$

where $\gamma \in (0, 1]$ is the discount factor and $r(\cdot)$ is the task reward function. We obtain the gradient $\nabla_\phi \mathcal{J}$ by performing BPTT through the unrolled computation graph. This is implemented via JAX's automatic differentiation engine [32], allowing us to differentiate through the RK4 integrator and neural residuals exactly. The policy parameters are updated via stochastic gradient ascent: $\phi \leftarrow \phi + \eta_\pi \nabla_\phi \mathcal{J}$. By combining autodiff with JAX's Just-In-Time (JIT) compilation and automatic vectorization (vmap), we achieve highly efficient, parallelized updates on the GPU.

C. Adaptive Temporal Scaling via Differentiable Optimization

While RASH-BPTT optimizes the control policy parameters, achieving time-optimal performance under varying conditions requires dynamically adapting the reference trajectory's evolution. To this end, we introduce *Adaptive Temporal Scaling (ATS)*, a closed-loop mechanism that decouples spatial path planning from the temporal execution profile. This approach gives the system a form of "temporal elasticity": it aggressively compresses the time domain (reducing α) to maximize agility when tracking is precise, yet automatically relaxes kinematic demands (increasing α) to restore stability upon encountering significant disturbances or model mismatches. Unlike heuristic speed adjustment rules, ATS is formulated as a holistic online optimization problem. Inspired by [33], we explicitly minimize the future discrepancy between the *real* and *reference* trajectories, utilizing the differentiable hybrid model as a proxy to anticipate the closed-loop effects of time-scale adjustments.

1) Parameterization: We parameterize the reference trajectory as a piecewise polynomial sequence. For each segment, the spatial path is defined by a coefficient matrix $\mathbf{C} \in \mathbb{R}^{3\times N}$ (representing a polynomial of degree $N-1$ in 3D space) and a temporal basis vector $\boldsymbol{\beta}(t) = [1, t, t^2, \cdots, t^{N-1}]^\top$. To control the execution rate, we introduce a scalar scaling parameter $\alpha \in \mathbb{R}_{>0}$. The reference position $\mathbf{p}_{\text{ref}}$ at a specific trajectory time τ is formulated as:

$$\mathbf{p}_{\text{ref}}(\tau; \alpha) = \mathbf{C}\boldsymbol{\beta}\left(\frac{\tau}{\alpha}\right). \tag{7}$$

Here, α serves as a *time dilation factor*. By exploiting the differential flatness property [34], we map the derivatives to the full reference state $\mathbf{x}_{\text{ref}}(\tau; \alpha)$.

2) Optimization Objective and Update: Direct optimization of α using real-world rollouts is intractable, as the physical system execution is non-differentiable and does not provide analytical gradients. To address this, we employ the differentiable hybrid model as a *proxy* to perform *counterfactual inference* [35]. At each update step, given a recent real-world rollout $\{(\bar{\mathbf{x}}_k, \bar{\mathbf{u}}_k)\}_{k=0}^{H-1}$, we analyze the sensitivity of the closed-loop performance to the time-scale α by constructing a *counterfactual* state sequence $\hat{\mathbf{x}}_k(\alpha)$. This allows us to formulate a composite potential function $\mathcal{J}_{\text{ATS}}$ over the horizon H that balances execution speed with safety:

$$\mathcal{J}_{\text{ATS}}(\alpha) = \lambda_{\text{speed}}\, \alpha + \lambda_{\text{safe}} \sum_{k=0}^{H-1} \Psi(\mathcal{E}_k(\alpha) - \mathcal{E}_{\text{th}}), \tag{8}$$

where the error metric is defined as

$$\mathcal{E}_k(\alpha) \triangleq \mathcal{D}(\hat{\mathbf{x}}_k(\alpha), \mathbf{x}_{\text{ref},k}(\alpha)). \tag{9}$$

Here, $\mathcal{D}(\cdot,\cdot)$ denotes the geometry-aware discrepancy from Eq. (4), and $\mathcal{E}_{\text{th}}$ is a predefined safety threshold.

Crucially, the counterfactual state $\hat{\mathbf{x}}_k(\alpha)$ serves as a differentiable surrogate for the physical state. It is anchored to the real observations, satisfying $\hat{\mathbf{x}}_k(\alpha_0) = \bar{\mathbf{x}}_k$ at the current time-scale α_0. However, unlike the fixed real history $\bar{\mathbf{x}}_k$, the evolution of $\hat{\mathbf{x}}_k(\alpha)$ depends on α. Its local sensitivity is approximated by linearizing the hybrid model dynamics along the real rollout $\{(\bar{\mathbf{x}}_k, \bar{\mathbf{u}}_k)\}$. Here, λ_{speed} and λ_{safe} are weighting factors, and $\Psi(z) = \frac{1}{\kappa}\ln(1 + \exp(\kappa z))$ is the Softplus barrier with sharpness κ.

We then compute the gradient $\nabla_\alpha \mathcal{J}_{\text{ATS}}$ using closed-loop sensitivity analysis. By applying the chain rule, the gradient combines the analytic gradient of the reference with the recursive sensitivity of the system state, approximated by the hybrid model dynamics $\mathbf{F}_{\Delta t}$:

$$\frac{\mathrm{d}\mathcal{J}_{\text{ATS}}}{\mathrm{d}\alpha} = \lambda_{\text{speed}} + \lambda_{\text{safe}} \sum_{k=0}^{H-1} \Psi'(\mathcal{E}_k(\alpha) - \mathcal{E}_{\text{th}}) \, \frac{\mathrm{d}\mathcal{E}_k(\alpha)}{\mathrm{d}\alpha}, \tag{10}$$

where

$$\frac{\mathrm{d}\mathcal{E}_k}{\mathrm{d}\alpha} = \frac{\partial \mathcal{E}_k}{\partial \mathbf{x}_{\text{ref},k}} \frac{\partial \mathbf{x}_{\text{ref},k}}{\partial \alpha} + \frac{\partial \mathcal{E}_k}{\partial \hat{\mathbf{x}}_k} \mathbf{S}_k. \tag{11}$$

Here, $\mathbf{S}_k \triangleq \frac{\mathrm{d}\hat{\mathbf{x}}_k}{\mathrm{d}\alpha}$ is the closed-loop state sensitivity, with $\mathbf{S}_0 =$

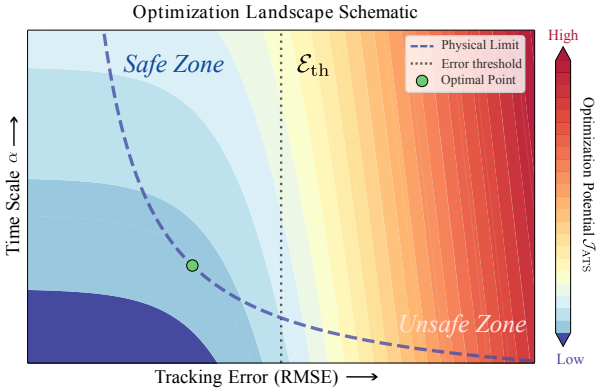

Fig. 3: **The optimization landscape for ATS.** The heatmap visualizes the composite potential $\mathcal{J}_{\text{ATS}}$, balancing agility (low α) against safety. The landscape transitions from the Safe Zone (blue) to the Unsafe Zone (red) determined by the tracking error threshold $\mathcal{E}_{\text{th}}$ (dotted line) and a schematic representation of the system's physical limits (dashed curve). The green dot marks the optimal equilibrium: the most aggressive time scale achievable within safe tracking bounds.

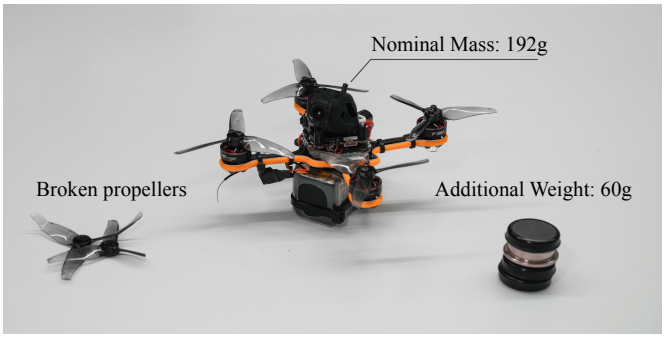

Fig. 4: **Experimental platform with extreme modifications.** To validate robustness, the nominal quadrotor (192 g) is subjected to drastic degradations: mechanically clipped propellers (inducing aerodynamic loss) and a 60 g payload. This 31% mass increase significantly alters the inertial properties and reduces the thrust-to-weight ratio.

$\mathbf{0}$, and evolves as:

$$\mathbf{S}_{k+1} = \mathbf{A}_k \mathbf{S}_k + \mathbf{B}_k \frac{\mathrm{d}\mathbf{u}_k}{\mathrm{d}\alpha}, \tag{12}$$

with Jacobians evaluated on the real rollout:

$$\mathbf{A}_k \triangleq \left.\frac{\partial \mathbf{F}_{\Delta t}}{\partial \mathbf{x}}\right|_{(\bar{\mathbf{x}}_k, \bar{\mathbf{u}}_k)}, \mathbf{B}_k \triangleq \left.\frac{\partial \mathbf{F}_{\Delta t}}{\partial \mathbf{u}}\right|_{(\bar{\mathbf{x}}_k, \bar{\mathbf{u}}_k)}. \tag{13}$$

The action sensitivity is obtained by differentiating through the policy:

$$\frac{\mathrm{d}\mathbf{u}_k}{\mathrm{d}\alpha} = \frac{\partial \pi_\phi}{\partial \mathbf{o}_k}\left(\frac{\partial \mathbf{o}_k}{\partial \hat{\mathbf{x}}_k}\mathbf{S}_k + \frac{\partial \mathbf{o}_k}{\partial \mathbf{x}_{\mathrm{ref},k}}\frac{\partial \mathbf{x}_{\mathrm{ref},k}}{\partial \alpha}\right), \tag{14}$$

where the observation $\mathbf{o}_k$ is a function of the state $\hat{\mathbf{x}}_k$ and the reference $\mathbf{x}_{\mathrm{ref},k}(\alpha)$. While the functional dependency on $\hat{\mathbf{x}}_k$ allows for differentiation, all Jacobian matrices (e.g., $\frac{\partial \pi_\phi}{\partial \mathbf{o}_k}, \frac{\partial \mathbf{o}_k}{\partial \hat{\mathbf{x}}_k}$) are evaluated at the nominal operating point defined by the real rollout $(\bar{\mathbf{x}}_k, \bar{\mathbf{u}}_k)$. Additional dependencies in $\mathbf{o}_k$ (e.g., action history) can be included analogously and are handled automatically in our autodiff implementation.

The parameter is updated via projected gradient descent:

$$\alpha \leftarrow \Pi_{[\alpha_{\min}, \alpha_{\max}]}(\alpha - \eta_\alpha \nabla_\alpha \mathcal{J}_{\mathrm{ATS}}). \tag{15}$$

This update rule naturally maintains an equilibrium near the safety boundary, as visualized in Fig. 3. In the *safe regime* ($\mathcal{E} \ll \mathcal{E}_{\mathrm{th}}$), the barrier gradient vanishes, and the linear term λ_{speed} drives α down to accelerate execution. Conversely, in the *unsafe regime*, the barrier gradient dominates, using the model-based sensitivity to increase α and restore feasibility.

IV. Results and Experiments

In this section, we present comprehensive experiments to validate the effectiveness of our self-adaptive framework. We conduct both real-world flight tests and extensive simulation comparisons to answer the following research questions:

- *RQ1 (Agility):* Can the proposed framework push the quadrotor to the saturation limit of its CTBR command interface in real time, starting from a conservative policy?
- *RQ2 (Robustness):* Is the framework capable of adapting to internal dynamic changes (e.g., payload, damage) and unknown external disturbances (e.g., wind) while maintaining safety?
- *RQ3 (Ablation):* How do the individual components, residual learning and real-world anchored initialization, contribute to the overall performance?

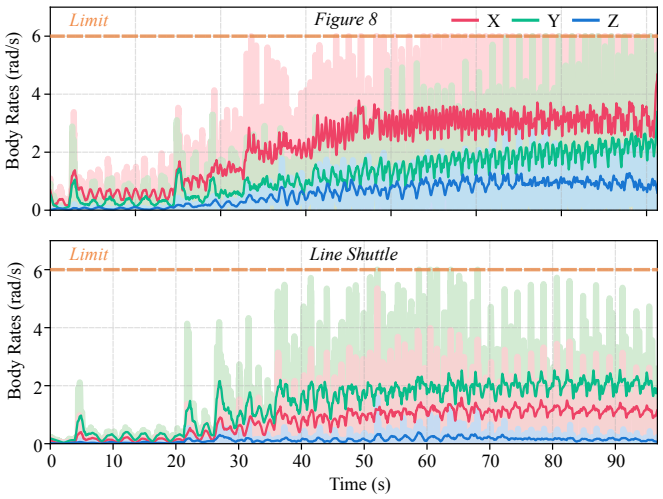

Fig. 5: Commanded body-rate $\boldsymbol{\omega}_{\mathrm{cmd}}$ during real-world Figure-8 and Linear Shuttle flights. Solid curves denote the 2 s sliding-window mean, overlaid on raw measurements (background traces). The orange dashed line marks the actuation limit ($6\,\mathrm{rad\,s^{-1}}$).

We validate the proposed framework entirely in real-world flight using a quadrotor based on the Agilicious platform [36], as shown in Fig. 4. A motion-capture system provides state estimates at 100 Hz. An off-board workstation runs the online learning loop and sends commands to the onboard flight controller at 50 Hz. The workstation is equipped with an NVIDIA RTX 4090 GPU; using JAX with JIT compilation and `vmap`, each adaptation step (100 epochs: ∼3 s residual + ∼6 s policy) completes in ∼9 s. The 50 Hz command loop requires only a forward pass through a lightweight MLP (∼100 μs), so policy execution is decoupled from learning.

A. Real-World Agile Flight and Adaptation

1) Approaching the CTBR Interface Limit (RQ1): We evaluate whether our framework can adapt from conservative flight to near-limit agility on two canonical trajectories: a linear shuttle and a Figure-8. As shown in Fig. 1, on the *Linear Shuttle*, the vehicle starts from a conservative maximum speed of 2.0 m/s and reaches 7.3 m/s after roughly 100 s of in-flight adaptation. On the *Figure-8*, the speed similarly increases from 2.2 m/s to 7.2 m/s over the same adaptation horizon.

Across both trajectories, the adapted policies reach the axis-wise saturation limit of the CTBR body-rate command interface. Fig. 5 shows the commanded body rate $\boldsymbol{\omega}_{\mathrm{cmd}}$, where the raw commands and their 2 s sliding-window mean indicate persistent saturation at the input bound, $\|\boldsymbol{\omega}_{\mathrm{cmd}}\|_\infty = 6$ rad/s, during the later stages of adaptation. This indicates that ATS does not merely improve tracking at a fixed aggressiveness level; rather, it safely exploits the improved predictive accuracy provided by residual learning and RASH-BPTT to push the trajectory toward the boundary of the CTBR command regime.

To contextualize the achieved lap times, we compute two zero-delay offline lower bounds via trajectory optimization: a *CTBR bound* under ideal rigid-body dynamics with the same command limits $c \leq 25\ \mathrm{m\,s^{-2}}$ and $\|\boldsymbol{\omega}_{\mathrm{cmd}}\|_\infty \leq 6$ rad/s, and a *rotor-thrust bound* that directly constrains individual

TABLE I: Real-world lap times versus zero-delay offline lower bounds.

Method	Figure-8 (s)	Shuttle (s)
Rotor-thrust bound†	2.02	1.42
CTBR bound	2.24	1.49
Ours	2.60	2.30

†Not reachable through the CTBR interface; the optimized trajectory requires body rates exceeding 28 rad/s.

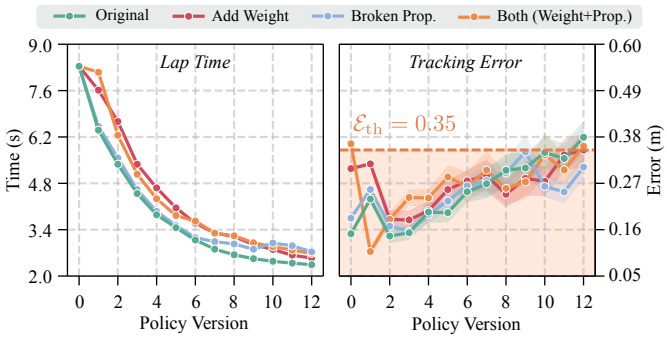

Fig. 6: Adaptation to hardware variations. The framework is tested with (i) added mass, (ii) propeller damage, and (iii) combined conditions. In all cases, the residual learning module swiftly compensates for the dynamic mismatch within one iteration, enabling the ATS to safely push the compromised hardware to its new physical limits.

motor thrusts without the CTBR body-rate ceiling. Table I summarizes the comparison.

Our Figure-8 result, 2.60 s, approaches the CTBR bound of 2.24 s, indicating operation near the command-interface ceiling. The larger shuttle gap, 2.30 s versus 1.49 s, reflects the trajectory's stronger sensitivity to actuation delays, rapid acceleration reversals, and aerodynamic effects absent from the zero-delay reference.

2) Robustness to Hardware Variations (RQ2): To evaluate the system's resilience to internal dynamic shifts, we conducted flight tests on the Figure-8 trajectory under three distinct hardware degradations: *Added Mass* (adding a 60 g payload), *Propeller Damage* (clipping propeller tips), and *Combined* (both mass increase and propeller damage).

As illustrated in Fig. 6, compared to the intact baseline, the achievable maximum speeds in all three scenarios naturally decreased, reflecting the reduced physical authority of the compromised hardware. In contrast, our framework successfully identified the *new* feasible operating envelope for each configuration, allowing the drone to operate at the boundary of CTBR body-rate saturation while maintaining stability.

A key observation underscores the efficacy of our Residual Dynamics Learning. Immediately following hardware alteration, the nominal policy exhibited significant tracking errors due to severe model mismatch. Remarkably, after just a *single update iteration* of the residual model, the tracking error dropped precipitously. This rapid adaptation enabled ATS to confidently decrease α, thereby compressing the time scale and increasing agility. This confirms that the residual network effectively captures platform-specific properties, enabling the ATS+RASH-BPTT loop to safely explore the unknown performance envelope.

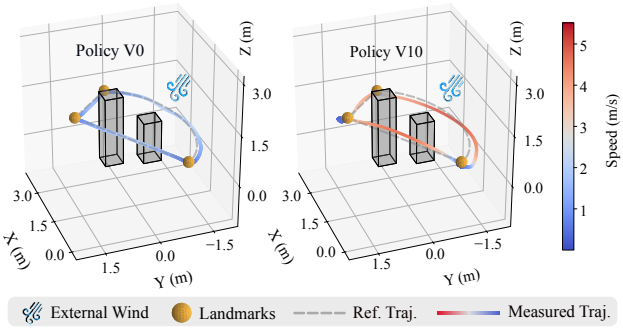

Fig. 7: Evolution of flight agility during the inspection mission. The drone must traverse three landmarks while battling airflow from a fan (top-right). Starting from a baseline of 12 s (Policy V0, left), the framework adapts to the disturbance and accelerates the execution to 7 s (Policy V10, right).

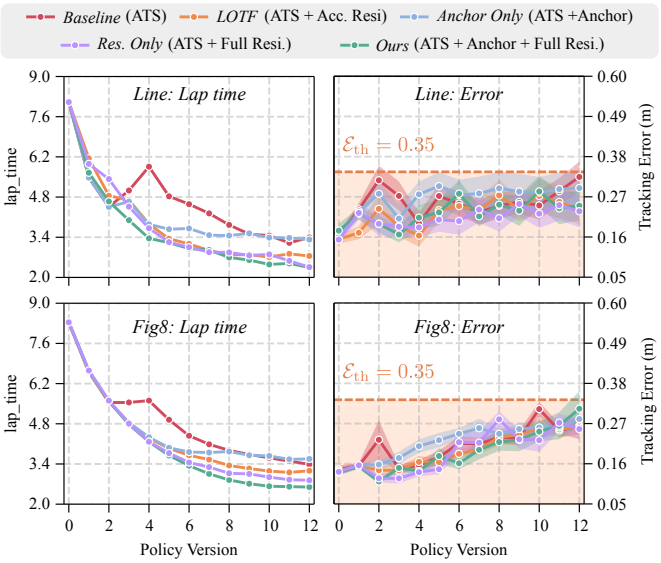

Fig. 8: **Ablation study of residual dynamics and anchored rollouts.** *Note:* Since no existing methods support online learning at the physical limit in real-world settings, all baselines are augmented with our ATS framework to enable feasible deployment. We compare the *Baseline* (nominal), *LOTF* [17] (acceleration residual only), component ablations (*Anchor/Residual Only*), and *Ours* (full system). Enabled by our ATS framework, all methods successfully maintain a bounded tracking error less than $\mathcal{E}_{\text{th}}$ while reducing lap time, demonstrating the framework's effectiveness. Furthermore, *Ours* achieves the shortest lap time, demonstrating that anchored initialization combined with full hybrid residual modeling is critical for pushing the agility envelope.

3) Application: Time-Optimal Inspection under Unknown Disturbances: To demonstrate practical utility, we evaluated the framework in a realistic multi-stage inspection mission subject to unknown wind fields between landmarks (Fig. 1). Despite turbulent aerodynamic disturbances, the system successfully maintained stable tracking via online residual adaptation. This robustness empowered the ATS mechanism to progressively compress the execution duration, ultimately reducing the total motion time by 42% (from 12 s to 7 s) while adhering to safety constraints, as visualized in Fig. 7.

B. Simulation Evaluation

1) Ablation Study: While our real-world experiments demonstrated the system's overall efficacy, we use simulation to disentangle the contributions of individual modules. Since no existing method supports online learning while pushing to-

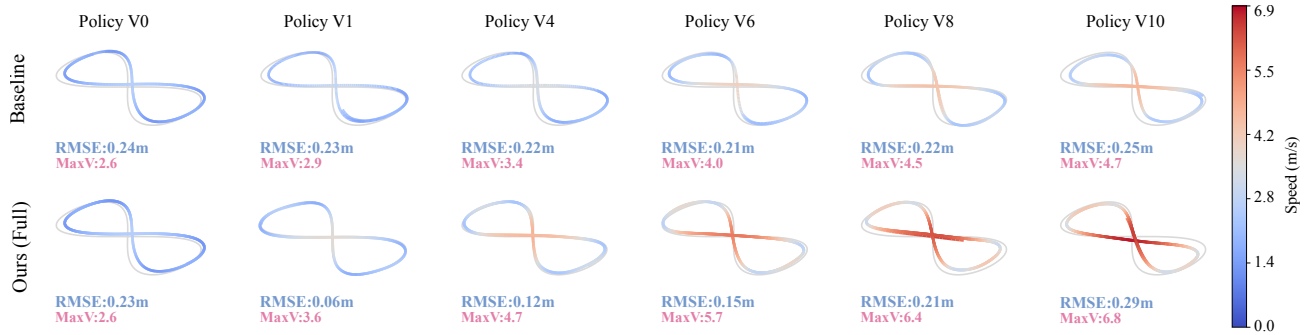

Fig. 9: **Online adaptation under wind disturbance.** Comparison between the *Baseline* and *Ours*. The *Baseline* fails to accelerate due to unmodeled wind drag, remaining stuck at lower speeds of 4.7 m/s. In contrast, *Ours* demonstrates a rapid adaptation cycle: the residual network first suppresses the tracking error (V0 $\rightarrow$ V1, RMSE drops to 0.06 m), creating a safety margin. Subsequently, ATS exploits this margin to aggressively scale up agility, achieving a top speed of $6.8\,\mathrm{m\,s^{-1}}$ (V10) while maintaining the trajectory tracking error within the threshold.

ward the agility envelope in real-world agile-flight settings, we integrate our ATS framework into all baselines to ensure a fair comparison. Consequently, the comparisons below represent ablated variants of our holistic system, evaluated on tracking accuracy (RMSE) and the maximum speed optimized by ATS:

- *Baseline:* Relies solely on the *nominal* dynamics model optimized via standard BPTT, serving as a lower bound to benchmark the efficacy of residual learning.
- *LOTF* [17]: Adapts the LOTF architecture to our framework. It models residual *linear acceleration* but omits rotational dynamics.
- *Anchor Only:* A variant that applies RASH-BPTT with anchored rollouts while disabling the learned residual.
- *Residual Only:* Incorporates the full hybrid residual model (linear acceleration and body rates) but reverts to standard BPTT optimization.
- *Ours:* The complete approach that integrates the full hybrid residual dynamics with anchored rollouts for optimal performance.

We set the tracking error threshold $\mathcal{E}_{\text{th}}$ to 0.35 m. The results, summarized in Fig. 8, yield three key insights:

Residual learning lifts the agility ceiling that ATS alone cannot break through. Across all methods, incorporating residual dynamics substantially reduces tracking errors. Since the ATS mechanism allows for time compression (decreasing α) *only* when the predicted error remains within the safety margin, lower RMSE directly translates into more aggressive time-scaling capabilities. Without residuals, the baseline hits the error threshold early, effectively capping the maximum speed.

Angular dynamics fidelity is critical in high-speed regimes. Compared to LOTF (acceleration residuals only), our full residual model (acceleration + angular velocity) achieves superior tracking accuracy as speed increases. This indicates that compensating for rotational dynamics mismatches (e.g., aerodynamic drag moments) is essential for maintaining prediction fidelity during aggressive maneuvers, which in turn empowers ATS to push performance boundaries further.

Anchored rollouts enhance optimization stability. Comparing *Anchor Only* against standard initialization reveals that anchoring the simulation rollout to the current real-world state mitigates the *distribution shift* between optimization and deployment. Practically, this keeps the optimization trajectory aligned with the on-policy region, yielding more stable convergence than optimizing from a reset reference state.

Overall, our full method achieves the optimal speed–accuracy trade-off, demonstrating a synergistic effect between residual dynamics (ensuring prediction accuracy) and anchored optimization (ensuring stable adaptation).

2) Online Adaptation under Wind Disturbance: We further evaluate robustness by benchmarking *Ours* against the *Baseline* in a simulated $5\,\mathrm{m\,s^{-1}}$ wind field (Fig. 9). Within just a single update, the residual-enhanced policy reduces tracking RMSE by approximately a factor of four compared to the baseline. This rapid error suppression is critical: it creates the necessary safety margin for ATS to initiate aggressive time-scaling early in the flight, validating that accurate dynamics are a prerequisite for agility. Over ten iterations, our method increases peak speed by a factor of 2.6, from $2.6\,\mathrm{m\,s^{-1}}$ to $6.8\,\mathrm{m\,s^{-1}}$, while maintaining a bounded tracking error below $\mathcal{E}_{\text{th}} = 0.35\,\mathrm{m}$. In contrast, the baseline fails to accelerate meaningfully due to persistent model mismatch. This confirms that residual learning effectively compensates for external disturbances, empowering the ATS optimizer to safely push the system toward the saturation limit of the CTBR command interface.

V. Discussion

We present a self-adaptive framework that enables autonomous quadrotors to evolve from conservative operation to the agility ceiling of their CTBR command interface *in real time*. Unlike methods requiring offline identification or extensive pre-training, our approach achieves this progression within 100 seconds of flight without any prior system identification. By tightly coupling differentiable simulation, online residual learning, and Adaptive Temporal Scaling (ATS), the system autonomously bridges the gap between conservative planning and agile execution, grounding performance dynamically in the robot's instantaneous capabilities.

A. Synergy between Residual Learning and ATS

Ablation studies reveal a critical dependency: *accurate short-horizon prediction is a prerequisite for safely exploiting agility*. Without residual learning, significant model mismatch

(e.g., unmodeled drag or delays) increases real-world tracking error, causing the ATS safety barrier (Eq. (8)) to prevent time compression. By introducing the hybrid residual model to minimize this dynamics mismatch, we expand the feasible safety margin. ATS immediately exploits this margin by decreasing α, pushing the system toward the CTBR command-interface ceiling (i.e., commanded body-rate saturation) rather than being bounded by modeling artifacts. This mechanism explains why our full method attains substantially higher terminal speeds: improved prediction accuracy unlocks agility without sacrificing safety.

B. Adaptation versus Static Robustness

Hardware variation experiments (Fig. 6) highlight the distinction between *static robustness* and *adaptive evolution*. Classical robust control typically ensures stability under a worst-case uncertainty set, often at the cost of nominal performance. In contrast, our framework continuously identifies and optimizes for the *current* dynamics. This is evident in the propeller damage scenario: rather than maintaining flight in an overly conservative regime, the system rapidly learns the altered dynamics and re-optimizes temporal scaling. This online specialization allows the policy to adapt to degradation without overestimating capabilities or imposing unnecessary conservatism.

C. Limitations and Future Work

The present implementation has several limitations that suggest promising directions for future work. First, the saturation observed in our experiments is imposed by the CTBR command interface, not by the rotors themselves: Table I shows residual headroom between the CTBR (2.24 s) and rotor-thrust (2.02 s) bounds on Figure-8. Operating at the individual-rotor-thrust level could expose this headroom, at the cost of a more challenging sim-to-real transfer and safety problem. Second, ATS currently optimizes only the temporal scaling of fixed path geometries; extending it to joint spatial–temporal optimization [37], in the spirit of MPCC [38], would enable more flexible adaptation and direct incorporation of obstacle-avoidance constraints. Third, the residual model is learned passively during stable task execution, leaving boundary regions of the state–action space under-modeled; safe active exploration could enable the controller to probe high-uncertainty regions while preserving stability. Finally, this work assumes accurate, low-latency state estimation, whereas aggressive flight can induce perception noise and latency; integrating perception-aware constraints into the ATS barrier would keep optimized trajectories within the sensing limits of the platform.

Acknowledgments

This work was supported by the European Union's Horizon Europe Research and Innovation Programme under grant agreement No. 101120732 (AUTOASSESS) and the European Research Council (ERC) under grant agreement No. 864042 (AGILEFLIGHT). The authors thank Ismail Geles, Yixi Cai, and Leonard Bauersfeld for discussions and internal review.

References

[1] C. Tang, B. Abbatematteo, J. Hu, R. Chandra, R. Martín-Martín, and P. Stone, "Deep reinforcement learning for robotics: A survey of real-world successes," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 8, no. 1, pp. 153–188, 2025.

[2] A. Loquercio, E. Kaufmann, R. Ranftl, M. Müller, V. Koltun, and D. Scaramuzza, "Learning high-speed flight in the wild," *Science Robotics*, vol. 6, no. 59, p. eabg5810, 2021.

[3] 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.

[4] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning quadrupedal locomotion over challenging terrain," *Science Robotics*, vol. 5, no. 47, p. eabc5986, 2020.

[5] T. Miki, J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning robust perceptive locomotion for quadrupedal robots in the wild," *Science Robotics*, vol. 7, no. 62, p. eabk2822, 2022.

[6] T. He, C. Zhang, W. Xiao, G. He, C. Liu, and G. Shi, "Agile But Safe: Learning Collision-Free High-Speed Legged Locomotion," in *Proceedings of Robotics: Science and Systems*, (Delft, Netherlands), July 2024.

[7] C. Chi, Z. Xu, S. Feng, E. Cousineau, Y. Du, B. Burchfiel, R. Tedrake, and S. Song, "Diffusion policy: Visuomotor policy learning via action diffusion," *The International Journal of Robotics Research*, vol. 44, no. 10-11, pp. 1684–1704, 2025.

[8] C. Chi, Z. Xu, C. Pan, E. Cousineau, B. Burchfiel, S. Feng, R. Tedrake, and S. Song, "Universal manipulation interface: In-the-wild robot teaching without in-the-wild robots," *arXiv preprint arXiv:2402.10329*, 2024.

[9] J. Xing, L. Bauersfeld, Y. Song, C. Xing, and D. Scaramuzza, "Contrastive learning for enhancing robust scene transfer in vision-based agile flight," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5330–5337, IEEE, 2024.

[10] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pp. 23–30, IEEE, 2017.

[11] K. Qu, G. Lan, R. Zurbrügg, C. Chen, C. E. Mower, H. Bou-Ammar, and M. Hutter, "A pragmatist robot: Learning to plan tasks by experiencing the real world," *IEEE Robotics and Automation Letters*, 2026.

[12] D. Hanover, A. Loquercio, L. Bauersfeld, A. Romero, R. Penicka, Y. Song, G. Cioffi, E. Kaufmann, and D. Scaramuzza, "Autonomous drone racing: A survey," *IEEE Transactions on Robotics*, vol. 40, pp. 3044–3067, 2024.

[13] M. O'Connell, G. Shi, X. Shi, K. Azizzadenesheli,

A. Anandkumar, Y. Yue, and S.-J. Chung, "Neural-fly enables rapid learning for agile flight in strong winds," *Science Robotics*, vol. 7, no. 66, p. eabm6597, 2022.

[14] E. Aljalbout, J. Xing, A. Romero, I. Akinola, C. R. Garrett, E. Heiden, A. Gupta, T. Hermans, Y. Narang, D. Fox, *et al.*, "The reality gap in robotics: Challenges, solutions, and best practices," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 9, 2025.

[15] L. Bauersfeld, E. Kaufmann, P. Foehn, S. Sun, and D. Scaramuzza, "Neurobem: Hybrid aerodynamic quadrotor model," in *Proceedings of Robotics: Science and Systems*, 2021.

[16] N. Sobanbabu, G. He, T. He, Y. Yang, and G. Shi, "Sampling-based system identification with active exploration for legged sim2real learning," in *Conference on Robot Learning*, pp. 578–598, PMLR, 2025.

[17] J. Pan, J. Xing, R. Reiter, Y. Zhai, E. Aljalbout, and D. Scaramuzza, "Learning on the fly: Rapid policy adaptation via differentiable simulation," *IEEE Robotics and Automation Letters*, 2026.

[18] J. Hwangbo, I. Sa, R. Siegwart, and M. Hutter, "Control of a quadrotor with reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 2, no. 4, pp. 2096–2103, 2017.

[19] Y. Song, A. Romero, M. Müller, V. Koltun, and D. Scaramuzza, "Reaching the limit in autonomous racing: Optimal control versus reinforcement learning," *Science Robotics*, vol. 8, no. 82, p. eadg1462, 2023.

[20] 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.

[21] J. Xing, A. Romero, L. Bauersfeld, and D. Scaramuzza, "Bootstrapping reinforcement learning with imitation for vision-based agile flight," in *Conference on Robot Learning*, pp. 2542–2556, PMLR, 2025.

[22] J. Heeg, Y. Song, and D. Scaramuzza, "Learning quadrotor control from visual features using differentiable simulation," in *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4033–4039, IEEE, 2025.

[23] E. Nava, J. Z. Zhang, M. Y. Michelis, T. Du, P. Ma, B. F. Grewe, W. Matusik, and R. K. Katzschmann, "Fast aquatic swimmer optimization with differentiable projective dynamics and neural network hydrodynamic models," in *International Conference on Machine Learning*, pp. 16413–16427, PMLR, 2022.

[24] E. Heiden, M. Macklin, Y. S. Narang, D. Fox, A. Garg, and F. Ramos, "DiSECt: A Differentiable Simulation Engine for Autonomous Robotic Cutting," in *Proceedings of Robotics: Science and Systems*, (Virtual), July 2021.

[25] J. Xu, S. Kim, T. Chen, A. R. Garcia, P. Agrawal, W. Matusik, and S. Sueda, "Efficient tactile simulation with differentiability for robotic manipulation," in *Conference on Robot Learning*, pp. 1488–1498, PMLR, 2023.

[26] J. Xu, V. Makoviychuk, Y. Narang, F. Ramos, W. Matusik, A. Garg, and M. Macklin, "Accelerated policy learning with parallel differentiable simulation," in *International Conference on Learning Representations*, 2022.

[27] I. Georgiev, K. Srinivasan, J. Xu, E. Heiden, and A. Garg, "Adaptive horizon actor-critic for policy learning in contact-rich differentiable simulation," in *Proceedings of the 41st International Conference on Machine Learning*, pp. 15418–15437, 2024.

[28] J. Y. Luo, Y. Song, V. Klemm, F. Shi, D. Scaramuzza, and M. Hutter, "Residual policy learning for perceptive quadruped control using differentiable simulation," in *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1–8, IEEE, 2025.

[29] R. T. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud, "Neural ordinary differential equations," *Advances in neural information processing systems*, vol. 31, 2018.

[30] 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)*, pp. 9784–9790, IEEE, 2019.

[31] M. Janner, J. Fu, M. Zhang, and S. Levine, "When to trust your model: Model-based policy optimization," *Advances in neural information processing systems*, vol. 32, 2019.

[32] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang, "JAX: composable transformations of Python+NumPy programs," 2018.

[33] F. Nan, H. Ma, Q. Guan, J. Hughes, M. Muehlebach, and M. Hutter, "Efficient model-based reinforcement learning for robot control via online learning," *arXiv preprint arXiv:2510.18518*, 2025.

[34] M. Faessler, A. Franchi, and D. Scaramuzza, "Differential flatness of quadrotor dynamics subject to rotor drag for accurate tracking of high-speed trajectories," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 620–626, 2017.

[35] L. Buesing, T. Weber, Y. Zwols, N. Heess, S. Racaniere, A. Guez, and J.-B. Lespiau, "Woulda, coulda, shoulda: Counterfactually-guided policy search," in *International Conference on Learning Representations*, 2018.

[36] P. Foehn, E. Kaufmann, A. Romero, R. Penicka, S. Sun, L. Bauersfeld, T. Laengle, G. Cioffi, Y. Song, A. Loquercio, *et al.*, "Agilicious: Open-source and open-hardware agile quadrotor for vision-based flight," *Science Robotics*, vol. 7, no. 67, p. eabl6259, 2022.

[37] Y. Ren, F. Zhu, G. Lu, Y. Cai, L. Yin, F. Kong, J. Lin, N. Chen, and F. Zhang, "Safety-assured high-speed navigation for mavs," *Science Robotics*, vol. 10, no. 98, p. eado6187, 2025.

[38] A. Romero, S. Sun, P. Foehn, and D. Scaramuzza, "Model predictive contouring control for time-optimal quadrotor flight," *IEEE Transactions on Robotics*, vol. 38, no. 6, pp. 3340–3356, 2022.