Imperative MPC: An End-to-End Self-Supervised Learning with Differentiable MPC for UAV Attitude Control

Haonan He HAONANH@ALUMNI.CMU.EDU
Yuheng Qiu YUHENGQ@ANDREW.CMU.EDU
Department of Mechanical Engineering, Carnegie Mellon University

Junyi Geng JGENG@PSU.EDU
Department of Aerospace Engineering, Pennsylvania State University

Editors: N. Ozay, L. Balzano, D. Panagou, A. Abate

Abstract

Modeling and control of nonlinear dynamics are critical in robotics, especially in scenarios with unpredictable external influences and complex dynamics. Traditional cascaded modular control pipelines often yield suboptimal performance due to conservative assumptions and tedious parameter tuning. Pure data-driven approaches promise robust performance but suffer from low sample efficiency, sim-to-real gaps, and reliance on extensive datasets. Hybrid methods combining learning-based and traditional model-based control in an end-to-end manner offer a promising alternative. This work presents a self-supervised learning framework combining learning-based inertial odometry (IO) module and differentiable model predictive control (d-MPC) for Unmanned Aerial Vehicle (UAV) attitude control. The IO denoises raw IMU measurements and predicts UAV attitudes, which are then optimized by MPC for control actions in a bi-level optimization (BLO) setup, where the inner MPC optimizes control actions and the upper level minimizes discrepancy between real-world and predicted performance. The framework is thus end-to-end and can be trained in a self-supervised manner. This approach combines the strength of learning-based perception with the interpretable model-based control. Results show the effectiveness even under strong wind. It can simultaneously enhance both the MPC parameter learning and IMU prediction performance.

Keywords:
Self-supervised learning, Differentiable Model Predictive Control, UAV Control

1. Introduction

Modeling and control of nonlinear dynamics precisely is essential for robotics, particularly in challenging settings like aerial robotics (Mousaei et al., 2022), aerial manipulation tasks (Geng and Langelaan, 2020). Such environments often feature complex nonlinear effects like coupled translational and rotational motion, or unpredictable external influences, complicating control design. Conventional methods like state feedback (Mellinger and Kumar, 2011; Blondin et al., 2019; Kozák, 2016) or optimization-based control (Garcia et al., 1989; Ji et al., 2022) require full knowledge of the system model. However, it is challenging to fully capture unpredictable factors such as wind gusts, boundary layer effects, uneven terrain, or hidden dynamics of chaotic nonlinear effects. These methods end up facing significant challenge and requiring tedious parameters tuning (Borase et al., 2021; Li and Wang, 2016; Alhajeri and Soroush, 2020).

Traditional control pipelines are typically modular, where perception & state estimation, planning and control are designed separately and then integrated, which often leads to suboptimal performance, as conservative assumptions in one component may cause local minima. The overall

© 2025 H. He, Y. Qiu & J. Geng.

performance is also constrained by the limitation of individual modules. In addition, iterative design update is required, where parameters of one module are fine-tuned while others are frozen, making the process tedious and time-consuming (Liu et al., 2023; Hanover et al., 2024). In contrast, data-driven approaches have gained more interest, which directly map sensor observation to action and offer robust learning-based perception. However, these methods usually require large amount of data, have low sample efficiency, and pose significant sim-to-real gap (Rawles et al., 2024; Ma et al., 2024). For example, some researchers train control policies by imitating expert actions. However, this approach demands extensive datasets to generalize across diverse environments (Xing et al., 2024). Another route is reinforcement learning (RL) and its variants (Williams et al., 2017), though promising, often exhibit slow convergence, low sample efficiency, and overfitting risks. Additionally, reward shaping can inadvertently introduce biases and result in suboptimal policies.

Recently, researchers have explored hybrid approaches that integrate learning-based methods with model-based approaches by embedding differentiable modules into learning frameworks. While much of this work focuses on perception, state estimation, and path planning, fewer studies address control. Amos et al. (2018) developed a differentiable model predictive control (MPC), combining physics-based modeling with data-driven methods to enable end-to-end learning of dynamics and control policies. However, many prior approaches rely on expert demonstrations or labeled data for supervised learning (Jin et al., 2020; Song and Scaramuzza, 2022), which, while effective on training datasets, struggle to generalize to unseen environments or handle external disturbances.

To address these issues, we propose a self-supervised learning framework combining learning-based perception with traditional model-based control. MPC can optimize control actions over a time horizon while satisfying system constraints. However, applying MPC in highly dynamic environments like wind gusts or moving obstacles in unmanned aerial vehicle (UAV) control remains challenging to balance optimality with real-time reactivity. For perception, UAVs face size, weight, power, and cost (SWAP-C) constraints (Mohsan et al., 2022; Youn et al., 2021), making inertial odometry (IO) based on lightweight, low-cost inertial measurement units (IMUs) ideal. Unlike vision or LiDAR sensors, IMUs avoid visual degradation factors such as motion blur or dynamic object interference (Zheng et al., 2007; Zhou et al., 2023), which are common in agile flights. However, traditional IO suffers from sensor noise and drift, leading to suboptimal performance (Tedaldi et al., 2014). Learning-based IO, alternatively, has gained attention for its ability to model and correct inherent data imperfections, thereby improving reliability and accuracy (Qiu et al., 2023). We incorporate differentiable MPC (d-MPC) into a self-supervised learning framework based on our prior work imperative learning (IL) (Wang et al., 2024), which we term imperative MPC. In our approach, a learning-based IO denoises raw IMU measurements and predicts UAV attitudes, then used by MPC for attitude control in a bi-level optimization (BLO) setup. The inner level solves standard MPC and the upper level optimizes discrepancy between control performance of the real vehicle and the predicted model. The whole framework is trained in a self-supervised manner. This joint training improves both MPC parameters and IMU prediction performance, leveraging robust learning-based perception and interpretable model-based control without modular separation.

1.1. Related Works

Learning-based Control. Hybrid approaches integrating learning-based methods with traditional model-based control become an emerging trend. These approaches range from combining learning residual dynamics with nominal dynamics and adaptive control to address real-time disturbances

and uncertainties (O'Connell et al., 2022), or incorporate large, complex neural network architectures as surrogate dynamics within an MPC (Salzmann et al., 2023), to policy search to optimize MPC parameters for improved adaptability (Song and Scaramuzza, 2022). However, these methods often follow a cascaded design, requiring separate stages for model learning and controller design. Another direction leverages physics-informed neural networks, embedding traditional control principles into loss functions. For instance, Mittal et al. (2020) integrates Lyapunov functions into the MPC framework, using neural networks to learn globally safe control policies. However, pure data-driven model introduces inaccuracies in learned Lyapunov functions or system dynamics, which degrade performance in dynamic environments. Differentiable optimization has recently gained attention for its end-to-end nature. Jin et al. (2020) proposed Pontryagin Differentiable Programming (PDP), embedding system dynamics and control strategies into a differentiable framework using Pontryagin's Minimum Principle (PMP). However, it is limited to local minima due to problem nonconvexity and first-order approximations. Furthermore, all these approaches heavily rely on supervised learning with substantial data and focus primarily on control, often neglecting upstream perception or planning in the autonomy pipeline.

Hybrid Learning and Optimization Framework for Autonomy. Many studies explore hybrid approaches in autonomy pipelines. Some integrate optimization-based planners into end-to-end learning (Han et al., 2024; Yang et al., 2023), but often ignore dynamics constraints or rely on supervised learning, or focusing solely on planning without control. Others combine model-based control with learning for decision-making or control-action initialization (Baek et al., 2022; Celestini et al., 2024), yet depend on modular training and offline-labeled data, limiting adaptability. Recent self-supervised methods, like Li et al. (2024)'s offline-to-online RL, improve flexibility but retain offline constraints. Similarly, RL-MPC hybrids (e.g., Romero et al. (2024)) leverage MPC's predictability and RL's exploration but assume known dynamics and face reward-shaping challenges. Our approach enables self-supervised reciprocal correction between neural and optimization components.

2. Preliminary

2.1. Differentiable Model Predictive Control

In model based control, MPC has dominated across many industrial applications due to its robust ability of handling complex tasks. It leverages a predictive model of the dynamical system $F(\cdot)$ and repeatedly solves an online optimization problem to minimize objective J in a receding horizon fashion with time horizon N, and produce the optimal states $\boldsymbol{x}$ and control $\boldsymbol{u}$ sequences $\boldsymbol{\mu}^* = \{\boldsymbol{\mu}_k\}_{1:N} = \{\boldsymbol{x}_k, \boldsymbol{u}_k\}_{1:N}$ (with only the first action is executed). Formally, at each time step, MPC solves an optimal control problem (OCP):

$$\begin{aligned} \min_{\boldsymbol{x}_{1:N}, \boldsymbol{u}_{1:N}} \; & J = \sum_k c_k(\boldsymbol{x}_k, \boldsymbol{u}_k) \\ \text{s.t.} \quad & \boldsymbol{x}_{k+1} = F(\boldsymbol{x}_k, \boldsymbol{u}_k), \quad k = 1, \cdots, N \\ & \boldsymbol{x}_1 = \boldsymbol{x}_{init};\ \boldsymbol{x}_k \in \mathcal{X};\ \boldsymbol{u}_k \in \mathcal{U} \end{aligned} \tag{1}$$

where $\boldsymbol{x}_k$ and $\boldsymbol{u}_k$ are the state and control input at time step k, $\mathcal{X}$ and $\mathcal{U}$ denote constraints on valid states and controls. From a learning perspective, the objective and dynamics as well as constraints of MPC may contains some unknown parameters, which can be predicted by some other autonomy components, e.g. $c_{\theta,k}(\cdot)$, $F_\theta(\cdot)$, $\boldsymbol{x}_{init,\theta}$, with the learnable parameters θ. The MPC thus can be integrated into a larger end-to-end systems. The learning process involves finding the derivatives of some loss function l, which are then used to update θ. D-MPC allows gradient back-propagation

through the traditional optimization process, enabling the end-to-end gradient can be obtained by:

$$\nabla_\theta l = \nabla_\theta \boldsymbol{\mu} \nabla_{\boldsymbol{\mu}} l \tag{2}$$

The gradient consists of $\nabla_\theta \boldsymbol{\mu}$ and $\nabla_{\boldsymbol{\mu}} l$. $\nabla_{\boldsymbol{\mu}} l$ can typically be computed analytically, while calculating $\nabla_\theta \boldsymbol{\mu}$ is more challenging. It can be obtained from simply unrolling and maintaining the entire computational graph throughout the iteration process, which incurs significant challenges in terms of memory usage. It may also face issues related to gradient divergence or vanishment (Finn et al., 2017). Another approach is to use implicit function differentiation, leveraging the necessary conditions for optimality, allowing the gradient to be computed without unrolling (Dontchev and Rockafellar, 2009). Denote $\xi \leq 0$ as the general constraints including the equality and inequality parts. The Lagrangian problem (1) with λ as the Lagrange multipliers can be formulated as:

$$\mathcal{L}(\boldsymbol{\mu}, \lambda) = J(\boldsymbol{\mu}) + \lambda^\top \xi(\boldsymbol{\mu}, \theta) \tag{3}$$

Then we can get the corresponding KKT (Karush-Kuhn-Tucker) conditions:

$$\begin{aligned} \nabla_{\boldsymbol{\mu}} J + \nabla_{\boldsymbol{\mu}} \xi \lambda^* = 0, \quad D(\lambda^*)\xi(\boldsymbol{\mu}^*, \theta) = 0, \\ \xi(\boldsymbol{\mu}^*, \theta) \leq 0, \quad \lambda^* \geq 0. \end{aligned} \tag{4}$$

where $D(\,\cdot\,)$ is a diagonal matrix derived from a vector. Eq. (4) can be easily reformulated as:

$$\begin{bmatrix} \mathrm{d}\boldsymbol{\mu} \\ \mathrm{d}\lambda \end{bmatrix} = - \begin{bmatrix} \nabla_{\boldsymbol{\mu},\boldsymbol{\mu}} J + (\nabla_{\boldsymbol{\mu},\boldsymbol{\mu}} \xi \lambda^*)^\top & \nabla_{\boldsymbol{\mu}} \xi \\ D(\lambda^*)\nabla_{\boldsymbol{\mu}} \xi & D(\xi) \end{bmatrix}^{-1} \begin{bmatrix} (\nabla_{\boldsymbol{\mu},\theta} \xi \lambda^*)^\top \\ D(\lambda^*)\nabla_\theta \xi \end{bmatrix} \mathrm{d}\theta \tag{5}$$

We thus analytically derives $\nabla_\theta \boldsymbol{\mu}$, which subsequently enables us to compute the gradient $\nabla_\theta L$ from Eq. (2). This enables efficient parameter updates within an end-to-end learning framework.

In this paper, we use the d-MPC developed in our previous work PyPose, an open-source library for robot learning (Wang et al., 2023). This module formulated the MPC as an iterative Linear Quadratic Regulator (iLQR) problem to tackle the non-convex and nonlinear scenarios. Unlike the existing method of KKT conditions, which is problem-specific, PyPose implements one extra optimization iteration of iLQR at the stationary point in the forward pass, allowing automatic gradient calculation in the backward pass, which is more versatile and problem-agnostic with the price of small approximation error (Bolte et al., 2023). This approach eliminates the need to compute gradients for the entire unrolled chain or derive analytical solutions tailored to each specific problem.

2.2. Imperative Learning

Our prior work introduced Imperative Learning (IL), a framework that integrates a neural system, a reasoning engine, and a memory module to enhance robotic learning and decision-making (Wang et al., 2024). IL is formulated as a BLO problem, where the upper-level (UL) objective U optimizes neural parameters related to perception, while the lower-level (LL) objective L optimizes parameters related to reasoning and memory:

$$\begin{aligned} \min_{\boldsymbol{\psi} \doteq [\boldsymbol{\theta}^\top, \boldsymbol{\gamma}^\top]^\top} & U\left(f_\theta(\boldsymbol{z}), g(\boldsymbol{\mu}^*), M(\boldsymbol{\gamma}, \boldsymbol{\nu}^*)\right) \\ \text{s.t.} \quad \boldsymbol{\phi}^* \in & \underset{\boldsymbol{\phi} \doteq [\boldsymbol{\mu}^\top, \boldsymbol{\nu}^\top]^\top}{\arg\min} L(f_\theta(\boldsymbol{z}), g(\boldsymbol{\mu}), M(\boldsymbol{\gamma}, \boldsymbol{\nu})) \\ \text{s.t.} \quad & \xi(M(\boldsymbol{\gamma}, \boldsymbol{\nu}), \boldsymbol{\mu}, f_\theta(\boldsymbol{z})) = \text{ or } \leq 0 \end{aligned} \tag{6}$$

where $f_\theta(\boldsymbol{z})$ is the neural outputs such as semantic attributes, $\boldsymbol{z}$ represents the sensor measurements, and $\boldsymbol{\theta}$ is the perception-related learnable parameters. The reasoning engine is $g(f, M, \boldsymbol{\mu})$ with parameters $\boldsymbol{\mu}$, while the memory system is denoted as $M(\boldsymbol{\gamma}, \boldsymbol{\nu})$ with perception-related parameters $\boldsymbol{\gamma}$ as well as reasoning-related parameters $\boldsymbol{\nu}$ (Wang et al., 2021). ξ is a general constraint; and $\boldsymbol{\psi} \doteq [\boldsymbol{\theta}^\top, \boldsymbol{\gamma}^\top]^\top$ are stacked UL variables and $\boldsymbol{\phi} \doteq [\boldsymbol{\mu}^\top, \boldsymbol{\nu}^\top]^\top$ are stacked LL variables, respectively. IL

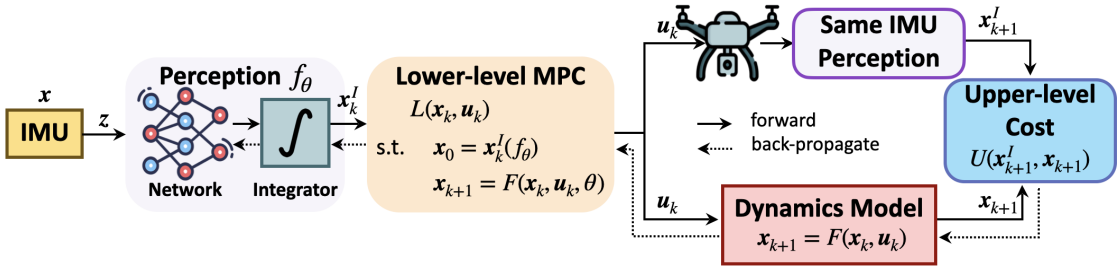


Figure 1: The proposed framework. The IMU model predicts the current state $\boldsymbol{x}_k^I$. The d-MPC solves for the optimal action $\boldsymbol{u}_k$ under lower-level L, which controls the dynamics model to the next state ($\boldsymbol{x}_{k+1}$) and actuates the real system to next state measured by the same IMU ($\boldsymbol{x}_{k+1}^I$). The upper-level U minimizes the discrepancy between $\boldsymbol{x}_{k+1}$ and $\boldsymbol{x}_{k+1}^I$.

also leverages implicit differentiation to compute gradients for interdependent neural and symbolic parameters efficiently. The solution to IL Eq. (6) mainly involves solving $\boldsymbol{\theta}$, $\boldsymbol{\gamma}$, $\boldsymbol{\mu}$, and $\boldsymbol{\nu}$.

The perception module extracts semantic features from sensor data, the symbolic reasoning module enforces logical or physical constraints, and the memory module stores relevant knowledge for long- and short-term retrieval when necessary such as in the application of SLAM. Through reciprocal learning, IL enables a self-supervised mechanism in which perception, reasoning, and memory modules co-evolve by iteratively refining each other to satisfy a global objective. This structure allows IL to combine the strength of both data-driven learning and physics knowledge. While the previous IL works mainly focus on SLAM, planning, and preliminary control (Yang et al., 2023; Fu et al., 2024; Wang et al., 2024), this work thoroughly introduces the usage of IL on control and provides high-fidelity validation results.

3. Methodology

3.1. Pipeline Overview

As illustrated in Fig. 1, our framework consists of a perception network $f_\theta(\boldsymbol{z})$, in this work is the learning-based IO, and a physics-based optimization based on MPC. Specifically, the encoder $f(\cdot)$ takes into raw sensor measurements $\boldsymbol{z}$, and here are IMU measurements to predict the high-level parameters for MPC, such as objective function, dynamic model, initial condition or constraints. Here is UAV's current attitude as initial state $\boldsymbol{x}_0$. Then, MPC takes the current attitude as initial states and solves for the the optimal states $\boldsymbol{x}$ and control $\boldsymbol{u}$ sequences $\boldsymbol{\mu}^* = \{\boldsymbol{\mu}_k\}_{1:N} = \{\boldsymbol{x}_k, \boldsymbol{u}_k\}_{1:N}$ over a time horizon N under constraints of system dynamics $F(\cdot)$, actuator limit, and other factors. Then, only the first action is executed. The problem is be formulated as a BLO problem:

$$\begin{aligned} \min_{\boldsymbol{\theta}} \quad & U\left(f_\theta(\boldsymbol{z}), g(\boldsymbol{\mu}^*)\right) \\ \text{s.t.} \quad & \boldsymbol{\mu}^* = \arg\min_{\boldsymbol{\mu}} L(f, g(\boldsymbol{\mu})) \\ & \text{s.t.} \quad \boldsymbol{x}_{k+1} = F(\boldsymbol{x}_k, \boldsymbol{u}_k), \quad k = 1, \cdots, N \\ & \qquad \boldsymbol{x}_1 = \boldsymbol{x}(t_1);\ \boldsymbol{x}_k \in \mathcal{X};\ \boldsymbol{u}_k \in \mathcal{U} \end{aligned} \tag{7}$$

where the lower level (LL) optimization is a standard MPC to output optimal control sequences $\boldsymbol{u}$ sequences $\boldsymbol{\mu}^* = \{\boldsymbol{\mu}_k\}_{1:N}$ with $\boldsymbol{x}_k$ and $\boldsymbol{u}_k$ are the state and control input at time step k, $\mathcal{X}$ and $\mathcal{U}$ are the constraints on valid states and controls. Notice in this scenario, there is no need for the memory system $M(\boldsymbol{\gamma}, \boldsymbol{\nu})$ as in Eq. (6). The objective function L is designed to minimize both tracking error and control effort:

$$L(\boldsymbol{\mu}_k) \doteq \sum_{k=0}^{N-1} \left(\Delta\boldsymbol{\mu}_k^\top \boldsymbol{Q}_k \Delta\boldsymbol{\mu}_k + \boldsymbol{p}_k \Delta\boldsymbol{\mu}_k\right) \tag{8}$$

Here $\Delta\boldsymbol{\mu}_k = \boldsymbol{\mu}_k - \boldsymbol{\mu}_k^{\text{ref}}$, where $\boldsymbol{\mu}_k^{\text{ref}}$ represents the reference state trajectory, and $\boldsymbol{Q}_k$ and $\boldsymbol{p}_k$ are the weight matrices to balance tracking performance and control effort, respectively. The optimal control action $\boldsymbol{u}_k$ is then applied to both the modeled dynamics and the physical robot, resulting in the subsequent states $\boldsymbol{x}_{k+1}$ and $\boldsymbol{x}_{k+1}^I$ (the latter measured by the IMU network).

The upper level (UL) cost is defined as the Euclidean distance between $\boldsymbol{x}_{k+1}$ and $\boldsymbol{x}_{k+1}^I$:

$$U(\boldsymbol{\theta}) \doteq \|\boldsymbol{x}_{k+1}^I - \boldsymbol{x}_{k+1}\|_2. \tag{9}$$

This discrepancy between the predicted states and the IMU network measurement captures the imperfectness of perception and dynamics model, suggesting the necessitates to adjust the learnable parameters $\boldsymbol{\theta}$ within both the network and MPC module. Notice that both x_k^I and x_{k+1}^I are indeed outputs of the perception module f_θ. However, gradients are explicitly backpropagated only through x_{k+1}^I, treating the current-step state x_k^I as a fixed initial condition within the optimization loop.

A critical step in (7) involves calculating the implicit gradient $\nabla_\theta \boldsymbol{\mu}^*$. We leveraging PyPose's d-MPC to eliminate the necessity to calculate gradients for the whole unrolled chain or analytical derivatives. The parameters in both IMU network and d-MPC can thus be jointly learned. Additionally, our framework reduces dependency on annotated labels by utilizing self-supervision from the physical model and achieves dynamically feasible predictions through the MPC module.

3.2. Perception Networks–Learning based IO

For the perception model, we leverage a neural network to model the noise inhere in both the accelerometer and the gyroscope. After filtering the noise, we employ the differentiable IMU pre-integrator from Qiu et al. (2023) to estimate the state $\boldsymbol{x}_{k+1}^I = f_\theta(z)$, where z is the measurements of the IMU including accelerometer and the gyroscope. The differentiability of the IMU pre-integrator enables gradient flow from the integrated state $\boldsymbol{x}_{k+1}^I$ back to the noise modeling network during training. To reduce computational complexity, we adopt a lightweight design: a 3-layer multi-layer perceptron (MLP) encoder extracts feature embeddings, followed by a 2-layer MLP decoder that predicts sensor noise. We randomly initialize the network weights and train the model from scratch.

3.3. UAV Dynamics Model

We consider a quadrotor UAV. The dynamics can be modeled using the Netwon-Euler method:

$$\begin{aligned} m\ddot{\mathbf{r}} &= -mg\mathbf{z}_W + T\mathbf{z}_B \\ \mathbf{J}\dot{\boldsymbol{\omega}} &= -\boldsymbol{\omega} \times \mathbf{J}\boldsymbol{\omega} + \boldsymbol{\tau} \end{aligned} \tag{10}$$

where m is the UAV mass, $\mathbf{r} = [x, y, z]$ is the UAV position vector in world frame $\mathcal{W}$ with the $x_W - y_W - z_W$ axes, g is the gravity vector, T is the body thrust force magnitude, $\mathbf{J}$ is the moment of inertia matrix referenced to the center of mass along the body $x_B - y_B - z_B$ axes. $\boldsymbol{\omega}$ is the robot angular velocity of frame $\mathcal{B}$ in frame $\mathcal{W}$, $\boldsymbol{\tau} = [\tau_x, \tau_y, \tau_z]$ is the body moment. We also use 3-2-1 Euler angles to define the roll, pitch, and yaw angles $\boldsymbol{\Omega} = [\phi, \theta, \psi]$ as the orientation. The state of the system is given by the position and velocity of the center of mass and the orientation and the angular velocity: $\mathbf{x} = [\mathbf{r}, \boldsymbol{\Omega}, \dot{\mathbf{r}}, \boldsymbol{\omega}]^\top$. The control input is defined as $\mathbf{u} = [T; \boldsymbol{\tau}]^\top$.

Eq. (10) will be incorporated as constraints $F(\cdot)$ of Eq. (4) in (7). The optimization problem will be solved to obtain states and control inputs is dynamic feasible. The optimal control action $\boldsymbol{u}_k$ will be sent to both the modeled UAV dynamics and the real vehicle accordingly.

4. Experiments

We conducted experiments to validate the effectiveness of iMPC for UAV attitude control. We evaluate the system performance under different initial condition, wind gusts, and model uncertainty.

4.1. Experiment Setup

We test our approach through both a customized Python simulation and a high-fidelity Gazebo simulator with widely used ROS and commercial PX4 Autopilot. The customized simulation is used to train our proposed framework and evaluate the controller performance. Validation in the Gazebo PX4 SITL (Software-in-the-Loop) simulation captures real-world effects, including sensor and actuator dynamics, environmental conditions, and communication delays, serving as a digital twin and final checkpoint prior to real-world flight testing.

The Python simulation consists of standard 6-degree-of-freedom (DoF) quadrotor UAV dynamics described in Section 3.3, running at 1 KHz, and a simulated IMU sensor at 200 Hz. To emulate real-world effects such as environmental disturbances, actuator uncertainties, sensor noise, and other unknown behaviors, we inject zero-mean Gaussian noise with a standard deviation of 1×10^{-4} into the control input, and zero-mean Gaussian noise with a standard deviation of 8.73×10^{-2} into the UAV attitude at 1 KHz. Additionally, we employ sensor noise models documented by the `Epson G365` IMU, including initial bias, bias instability, and random walk. This produces the same noise level as typical UAV tracking systems employing visual-inertial odometry (Qin et al., 2018) for attitude estimation. For wind disturbance experiments, we introduce external forces and torques based on the drag equation: $F = \frac{1}{2} \cdot C_d \cdot \rho \cdot A \cdot V^2$ and $\tau = r \times F$, where C_d is the drag coefficient, ρ air density, A the frontal area of the object facing the wind, V wind speed, and r the lever arm length.

The Gazebo PX4 SITL simulation environment leverages the PX4 autopilot firmware as the low-level control and communicates with high-level decision-making via Mavros. Here, the MPC generates total thrust and angular rate commands, which are sent to the low-level controller. A 3DR iris quadrotor UAV model is used as the active agent, including all actuators and sensor plugins, such as the servo motors, IMU, etc. We directly use the Gazebo wind gusts to inject wind disturbance.

To demonstrate the effectiveness of jointly optimizing the IMU model and MPC, we investigate four scenarios: (1) IMU+MPC: classic IMU integrator with a regular d-MPC; (2) IMU$^+$+MPC: data-driven IMU model with a regular d-MPC; (3) IMU+MPC$^+$: classic IMU integrator with a d-MPC with learnable parameters, e.g. mass, moment of inertia (MOI); and (4) iMPC (ours): IMU$^+$ with MPC$^+$ trained with IL framework, where "IMU$^+$" refers to the data-driven IMU denoising and integration model from our previous work (Qiu et al., 2023). Additionally, for the wind disturbance test, we also implement a RL approach with a commonly used Proximal Policy Optimization (PPO) (Schulman et al., 2017) as the baseline. Specifically, we take pre-integrated IMU measurements as the observation, output UAV action (thrust and torques) in OpenAI gym (Brockman et al., 2016), and designed the reward to minimize the difference between the current and target UAV pose.

We use three metrics to evaluate the control performance: settling time (*ST*), root-mean-square error (*RMSE*), and steady-state error (*SSE*). Specifically, *ST* is the time for the UAV to enter and remain within $\pm 1.5^\circ$ of its final steady attitude, measuring how quickly the system settles; *RMSE* is the root-mean-square of the difference between estimated and the desired attitude; and *SSE* is the absolute difference between the steady and desired attitude, evaluating the control accuracy. All experiments are repeated ten times for generalizability.

Table 1: The performance of UAV attitude control under different initial conditions. The "IMU" means the RMSE (unit: $\times 10^{-3}\mathrm{rad}$) of attitude estimation for the corresponding method.

	IMU+MPC			IMU^{+}+MPC			IMU+MPC^{+}			**iMPC (ours)**		
Initial	10°	15°	20°	10°	15°	20°	10°	15°	20°	10°	15°	20°
ST (s)	0.287	0.330	0.336	0.281	0.299	0.318	0.283	0.321	0.324	**0.275**	**0.296**	**0.317**
RMSE (°)	0.745	0.726	0.728	0.692	0.691	0.685	0.726	0.721	0.715	**0.691**	**0.688**	**0.684**
SSE (°)	0.250	0.262	0.263	0.193	0.213	0.217	0.216	0.230	0.224	**0.185**	**0.201**	**0.208**
IMU	7.730	8.390	8.370	6.470	7.020	7.270	7.370	7.980	8.090	**6.310**	**6.800**	**7.010**

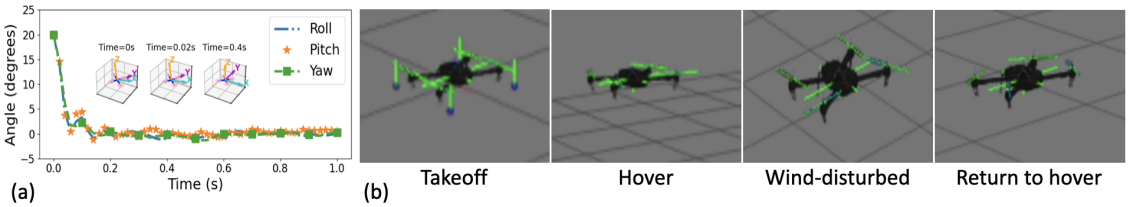


Figure 2: UAV Performances. (a) The UAV attitude quickly returns to a stable hover for an initial condition of 20° using iMPC. (b) Snapshots of iMPC under 20 m/s wind disturbance in Gazebo, including takeoff, hover, being disturbed by the wind, and returning to the hover.

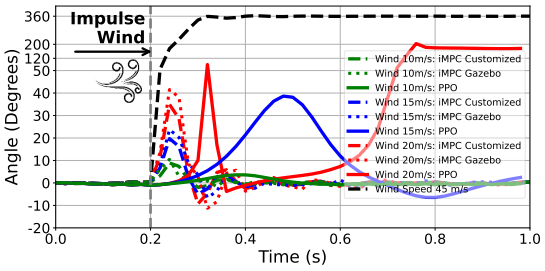
4.2. Results

Different Initial Conditions: We first evaluate system control performance under various initial conditions using our customized simulation, which offers flexibility and avoids the physical setup challenges of Gazebo. Table 1 compares performance across scenarios with initial attitudes ranging from 10° to 20° for all three Euler angles (roll, pitch, and yaw). Our method reliably stabilizes the UAV under all tested conditions, with all exhibiting stable performance and negligible standard deviations. Compared to baseline methods, our framework achieves lower *SSE*, *RMSE*, and *ST*. Fig. 2 demonstrates how rapidly the UAV returns to a stable hover from an initial attitude of 20° using our approach. To investigate how the LL MPC assists UL IMU parameter learning, we include IMU attitude estimation results in the last column of Table 1. Thanks to the physics knowledge from dynamics as well as the LL MPC, the denoising and prediction performance of the IMU module get significantly improved compared to the separately training. Consequently, our self-supervised learning framework improves both IMU network learning and overall control performance.

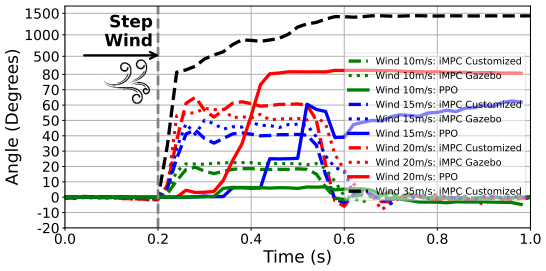
Wind Disturbance: To validate robustness, we evaluate system performance under wind disturbances in both customized and Gazebo simulations. We conducted two experiments: (1) impulse wind gusts opposite to the UAV's heading during hover at wind speeds of $10/15/20/45$ m/s, and (2) step winds in the same direction during hover for 0.3 s at speeds of $10/15/20/35$ m/s, beginning 0.2 s after stable hover. During RL (PPO) training, we apply external wind disturbances of up to 10m/s. In contrast, our hybrid MPC approach provides robustness without exposure to diverse wind conditions during training. Table 2 compares iMPC with other methods, showing that iMPC yields lower *SSE*, shorter *ST*, and more accurate attitude prediction. Particularly compared to IMU+MPC and IMU^{+}+MPC, our method significantly improves IMU denoising and reduces RMSE. Even under substantial disturbances, iMPC ensures robust attitude control, demonstrating its ability to simultaneously enhance both MPC parameter learning and IMU denoising and prediction.

Table 2: UAV attitude control performance under different **impulse** and **step wind** disturbances.

			Impulse Wind Disturbance				**Step Wind Disturbance**			
	Wind	Methods	*ST* (s)	*RMSE* (°)	*SSE* (°)	IMU	*ST* (s)	*RMSE* (°)	*SSE* (°)	IMU
Customized	10 m/s	IMU+MPC	0.406	0.360	0.169	9.190	0.596	0.396	0.189	6.870
		IMU$^+$+MPC	0.392	0.355	0.142	9.010	0.594	0.373	0.183	6.110
		IMU+MPC$^+$	0.406	0.360	0.168	9.180	0.588	0.392	0.179	6.830
		RL (PPO)	0.563	1.721	1.398	✗	0.747	1.785	1.454	✗
		iMPC (ours)	**0.368**	**0.353**	**0.132**	**8.660**	**0.574**	**0.352**	**0.168**	**5.990**
	15 m/s	IMU+MPC	0.454	0.328	0.120	8.160	0.684	0.337	0.172	7.190
		IMU$^+$+MPC	0.450	0.316	0.065	7.730	0.670	0.309	0.167	6.740
		IMU+MPC$^+$	0.454	0.325	0.117	8.140	0.684	0.337	0.171	7.180
		RL (PPO)	✗	✗	✗	✗	✗	✗	✗	✗
		iMPC (ours)	**0.430**	**0.312**	**0.060**	**7.570**	**0.620**	**0.297**	**0.149**	**6.730**
	20 m/s	IMU+MPC	0.486	0.361	0.186	6.550	0.704	0.376	0.240	6.300
		IMU$^+$+MPC	0.476	0.356	0.139	6.330	0.690	0.317	0.173	6.120
		IMU+MPC$^+$	0.484	0.361	0.185	6.520	0.704	0.372	0.253	6.300
		RL (PPO)	✗	✗	✗	✗	✗	✗	✗	✗
		iMPC (ours)	**0.470**	**0.354**	**0.135**	**6.160**	**0.676**	**0.311**	**0.150**	**6.110**
Gazebo	10 m/s	IMU+MPC	0.654	0.644	0.259	10.180	0.778	0.665	0.311	10.540
		IMU$^+$+MPC	0.622	0.636	0.246	9.970	0.766	0.646	0.302	10.330
		IMU+MPC$^+$	0.649	0.342	0.255	10.140	0.769	0.665	0.307	10.530
		iMPC (ours)	**0.616**	**0.630**	**0.238**	**9.560**	**0.759**	**0.640**	**0.289**	**10.120**
	15 m/s	IMU+MPC	0.688	0.658	0.240	8.880	0.843	0.673	0.324	9.010
		IMU$^+$+MPC	0.647	0.641	0.211	8.270	0.841	0.638	0.315	8.950
		IMU+MPC$^+$	0.668	0.652	0.237	8.800	0.835	0.671	0.311	9.010
		iMPC (ours)	**0.635**	**0.637**	**0.206**	**8.150**	**0.827**	**0.626**	**0.306**	**8.740**
	20 m/s	IMU+MPC	0.729	0.667	0.275	7.490	0.996	0.690	0.322	7.090
		IMU$^+$+MPC	0.718	0.665	0.254	7.230	0.975	0.655	0.298	6.800
		IMU+MPC$^+$	0.720	0.667	0.273	7.440	0.996	0.684	0.320	7.090
		iMPC (ours)	**0.711**	**0.659**	**0.243**	**7.210**	**0.963**	**0.647**	**0.293**	**6.550**



(*a*) UAV pitch angle response when encountering an **impulse wind** disturbance at 0.2 s for different speeds. PPO loses control at a wind speed of 15 m/s, while iMPC loses control at 45 m/s.



(*b*) UAV pitch angle response when the encountering a **step wind** disturbance at 0.2 s and lasting for 0.3 s for different speeds. PPO loses control at a wind speed of 15 m/s, while iMPC loses control at 35 m/s.

Figure 3: Control performance of iMPC and RL (PPO) under different levels of wind disturbance.

Fig. 2 (b) shows the entire process of the UAV initially hovering, being disturbed by wind, and returning to stable hover, demonstrating that the UAV in Gazebo simulation can still resist wind and recover quickly, even under 20 m/s wind. UAV pitch angle responses using our iMPC and

Table 3: The learned UAV MOI and mass error using our method (iMPC).

	Initial Offset	Initial	Error	Initial	Error	Initial	Error	Initial	Error
MOI	50%	0°	0.96%	10°	2.67%	15°	3.41%	20°	2.22%
Mass	50%	0°	1.69%	10°	0.85%	15°	1.43%	20°	0.32%

PPO under different wind disturbances are shown in Fig. 3. The wind affects the UAV attitude with different peak values under different speeds. Even under 20 $\mathrm{m/s}$ impulse or step wind, the UAV overcomes the disturbance and quickly returns to hover. Note that the system tolerates higher speeds under impulse wind compared to step wind, due to the latter's longer duration. When the wind speed is too high, actuator limits prevent compensation, leading to failure and crash. Compared to PPO, iMPC exhibits notably better performance, especially under wind disturbances not seen during training. While PPO performs comparably under seen wind conditions, it completely loses control under out-of-distribution disturbances. These results highlight the robustness of our framework, benefiting from a hybrid structure that integrates prior physical knowledge.

Learned Dynamics Parameters: We also evaluate the learning performance in the d-MPC. In particular, we treat the UAV mass and moment of inertia (MOI) as the learnable parameters. The goal is to validate the final estimated mass and MOI match the real value after the learning process. To illustrate this, we set an initial guess of the vehicle MOI and mass with 50% offset from its true value and show the final estimated error using our method in Table 3. It can be seen that iMPC achieves a final MOI error of less than 3.5%, and a final mass error of less than 1.7%. By jointly considering the performance in Table 1 and Table 3 and compare iMPC with IMU$^+$+MPC, and IMU+MPC$^+$ with IMU+MPC, where "MPC$^+$" indicates the learned MOI and mass are in the control loop, we conclude that better-learned dynamics parameters lead to better attitude control performance, with smaller SSE and settling time.

Efficiency Analysis: Imperative MPC matches IMU + MPC in runtime because MPC dominates computation and the learning-based IMU inference adds negligible overhead. It runs at 50 Hz with a 200 Hz IMU , which is at the same level as standard MPC. Compared to RL, although the inference time of iMPC is longer as expected due to the need of solving an online optimization, its physics-based prior helps overcome the typical low sampling efficiency issue of RL during training. In fact, our experiment shows the PPO needs 221.28% more training time.

5. Conclusions

This paper proposes a novel end-to-end self-supervised learning framework that integrates a learning-based IO module with d-MPC for UAV attitude control. The framework addresses the challenges of nonlinear dynamics and unpredictable external disturbances by combining the robustness of data-driven perception with the interpretability of physics-based control. The problem was formulated as a bi-level optimization, where the inner MPC optimizes control actions and the upper level minimizes discrepancy between real-world and the predicted performance. The framework is thus end-to-end and can be trained in a self-supervised manner. The framework was validated in both a customized Python and PX4 Gazebo simulations. Results show the effectiveness even under strong external wind of up to 20 m/s, achieving a steady-state error with an accuracy of 0.243 degrees. The joint learning mechanism simultaneously enhances both the MPC parameter learning and IMU denoising and prediction performance. Future works includes directly deploying the PX4-based integration in to the real world UAVs and considering uncertainty in the perception module.

Acknowledgments

We thank Dr. Chen Wang and Qihang Li at the University at Buffalo for their suggestion and support to this research.

References

Mohammed Alhajeri and Masoud Soroush. Tuning guidelines for model-predictive control. *Industrial & Engineering Chemistry Research*, 59(10):4177–4191, 2020.

Brandon Amos, Ivan Jimenez, Jacob Sacks, Byron Boots, and J Zico Kolter. Differentiable mpc for end-to-end planning and control. *Advances in neural information processing systems*, 31, 2018.

Donghoon Baek, Amartya Purushottam, and Joao Ramos. Hybrid lmc: Hybrid learning and model-based control for wheeled humanoid robot via ensemble deep reinforcement learning. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 9347–9354. IEEE, 2022.

Maude Josée Blondin, Javier Sanchis Sáez, and Panos M Pardalos. Control engineering from classical to intelligent control theory—an overview. *Computational Intelligence and Optimization Methods for Control Engineering*, pages 1–30, 2019.

Jérôme Bolte, Edouard Pauwels, and Samuel Vaiter. One-step differentiation of iterative algorithms. *Advances in Neural Information Processing Systems*, 36:77089–77103, 2023.

Rakesh P Borase, DK Maghade, SY Sondkar, and SN Pawar. A review of pid control, tuning methods and applications. *International Journal of Dynamics and Control*, 9:818–827, 2021.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

Davide Celestini, Daniele Gammelli, Tommaso Guffanti, Simone D'Amico, Elisa Capello, and Marco Pavone. Transformer-based model predictive control: Trajectory optimization via sequence modeling. *IEEE Robotics and Automation Letters*, 2024.

Asen L Dontchev and R Tyrrell Rockafellar. *Implicit functions and solution mappings*, volume 543. Springer, 2009.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pages 1126–1135. PMLR, 2017.

Taimeng Fu, Shaoshu Su, Yiren Lu, and Chen Wang. islam: Imperative slam. *IEEE Robotics and Automation Letters*, 2024.

Carlos E Garcia, David M Prett, and Manfred Morari. Model predictive control: Theory and practice—a survey. *Automatica*, 25(3):335–348, 1989.

Junyi Geng and Jack W Langelaan. Cooperative transport of a slung load using load-leading control. *Journal of Guidance, Control, and Dynamics*, 43(7):1313–1331, 2020.

Zhichao Han, Long Xu, Liuao Pei, and Fei Gao. Learning to plan maneuverable and agile flight trajectory with optimization embedded networks. *arXiv preprint arXiv:2405.07736*, 2024.

Drew Hanover, Antonio Loquercio, Leonard Bauersfeld, Angel Romero, Robert Penicka, Yunlong Song, Giovanni Cioffi, Elia Kaufmann, and Davide Scaramuzza. Autonomous drone racing: A survey. *IEEE Transactions on Robotics*, 2024.

Tianchen Ji, Junyi Geng, and Katherine Driggs-Campbell. Robust model predictive control with state estimation under set-membership uncertainty. In *2022 IEEE Conference on Decision and Control (CDC)*. IEEE, 2022.

Wanxin Jin, Zhaoran Wang, Zhuoran Yang, and Shaoshuai Mou. Pontryagin differentiable programming: An end-to-end learning and control framework. *Advances in Neural Information Processing Systems*, 33:7979–7992, 2020.

Štefan Kozák. From pid to mpc: Control engineering methods development and applications. In *2016 cybernetics & informatics (K&I)*, pages 1–7. IEEE, 2016.

Shuxia Li and Jiesheng Wang. Research on engineering tuning methods of pid controller parameters and its application. In *Intelligent Computing Methodologies: 12th International Conference, ICIC 2016, Lanzhou, China, August 2-5, 2016, Proceedings, Part III 12*, pages 563–570. Springer, 2016.

Zhaoxin Li, Letian Chen, Rohan Paleja, Subramanya Nageshrao, and Matthew Gombolay. Faster model predictive control via self-supervised initialization learning. *arXiv preprint arXiv:2408.03394*, 2024.

Lixing Liu, Xu Wang, Xin Yang, Hongjie Liu, Jianping Li, and Pengfei Wang. Path planning techniques for mobile robots: Review and prospect. *Expert Systems with Applications*, 227: 120254, 2023.

Xiao Ma, Sumit Patidar, Iain Haughton, and Stephen James. Hierarchical diffusion policy for kinematics-aware multi-task robotic manipulation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18081–18090, 2024.

Daniel Mellinger and Vijay Kumar. Minimum snap trajectory generation and control for quadrotors. In *2011 IEEE international conference on robotics and automation*, pages 2520–2525. IEEE, 2011.

Mayank Mittal, Marco Gallieri, Alessio Quaglino, Seyed Sina Mirrazavi Salehian, and Jan Koutník. Neural lyapunov model predictive control: Learning safe global controllers from sub-optimal examples. *arXiv preprint arXiv:2002.10451*, 2020.

Syed Agha Hassnain Mohsan, Muhammad Asghar Khan, Fazal Noor, Insaf Ullah, and Mohammed H Alsharif. Towards the unmanned aerial vehicles (uavs): A comprehensive review. *Drones*, 6(6):147, 2022.

Mohammadreza Mousaei, Junyi Geng, Azarakhsh Keipour, Dongwei Bai, and Sebastian Scherer. Design, modeling and control for a tilt-rotor vtol uav in the presence of actuator failure. In *2022*

IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 4310–4317. IEEE, 2022.

Michael O'Connell, Guanya Shi, Xichen Shi, Kamyar Azizzadenesheli, Anima Anandkumar, Yisong Yue, and Soon-Jo Chung. Neural-fly enables rapid learning for agile flight in strong winds. *Science Robotics*, 7(66):eabm6597, 2022.

Tong Qin, Peiliang Li, and Shaojie Shen. Vins-mono: A robust and versatile monocular visual-inertial state estimator. *IEEE transactions on robotics*, 34(4):1004–1020, 2018.

Yuheng Qiu, Chen Wang, Xunfei Zhou, Youjie Xia, and Sebastian Scherer. Airimu: Learning uncertainty propagation for inertial odometry. *arXiv preprint arXiv:2310.04874*, 2023.

Christopher Rawles, Alice Li, Daniel Rodriguez, Oriana Riva, and Timothy Lillicrap. Androidinthewild: A large-scale dataset for android device control. *Advances in Neural Information Processing Systems*, 36, 2024.

Angel Romero, Yunlong Song, and Davide Scaramuzza. Actor-critic model predictive control. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 14777–14784. IEEE, 2024.

Tim Salzmann, Elia Kaufmann, Jon Arrizabalaga, Marco Pavone, Davide Scaramuzza, and Markus Ryll. Real-time neural mpc: Deep learning model predictive control for quadrotors and agile robotic platforms. *IEEE Robotics and Automation Letters*, 8(4):2397–2404, 2023.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Yunlong Song and Davide Scaramuzza. Policy search for model predictive control with application to agile drone flight. *IEEE Transactions on Robotics*, 38(4):2114–2130, 2022.

David Tedaldi, Alberto Pretto, and Emanuele Menegatti. A robust and easy to implement method for imu calibration without external equipments. In *2014 IEEE international conference on robotics and automation (ICRA)*, pages 3042–3049. IEEE, 2014.

`Epson G365`. *Epson Electronics M-G365 Datasheet*, 2020.

Chen Wang, Yuheng Qiu, Wenshan Wang, Yafei Hu, Seungchan Kim, and Sebastian Scherer. Unsupervised online learning for robotic interestingness with visual memory. *IEEE Transactions on Robotics*, 38(4):2446–2461, 2021.

Chen Wang, Dasong Gao, Kuan Xu, Junyi Geng, Yaoyu Hu, Yuheng Qiu, Bowen Li, Fan Yang, Brady Moon, Abhinav Pandey, et al. Pypose: A library for robot learning with physics-based optimization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22024–22034, 2023.

Chen Wang, Kaiyi Ji, Junyi Geng, Zhongqiang Ren, Taimeng Fu, Fan Yang, Yifan Guo, Haonan He, Xiangyu Chen, Zitong Zhan, et al. Imperative learning: A self-supervised neural-symbolic learning framework for robot autonomy. *arXiv preprint arXiv:2406.16087*, 2024.

Grady Williams, Nolan Wagener, Brian Goldfain, Paul Drews, James M Rehg, Byron Boots, and Evangelos A Theodorou. Information theoretic mpc for model-based reinforcement learning. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 1714–1721. IEEE, 2017.

Jiaxu Xing, Angel Romero, Leonard Bauersfeld, and Davide Scaramuzza. Bootstrapping reinforcement learning with imitation for vision-based agile flight. *arXiv preprint arXiv:2403.12203*, 2024.

Fan Yang, Chen Wang, Cesar Cadena, and Marco Hutter. iplanner: Imperative path planning. *arXiv preprint arXiv:2302.11434*, 2023.

Wonkeun Youn, Hayoon Ko, Hyungsik Choi, Inho Choi, Joong-Hwan Baek, and Hyun Myung. Collision-free autonomous navigation of a small uav using low-cost sensors in gps-denied environments. *International Journal of Control, Automation and Systems*, 19(2):953–968, 2021.

Nanning Zheng, George Loizou, Xiaoyi Jiang, Xuguang Lan, and Xuelong Li. Computer vision and pattern recognition, 2007.

Chu Zhou, Minggui Teng, Jin Han, Jinxiu Liang, Chao Xu, Gang Cao, and Boxin Shi. Deblurring low-light images with events. *International Journal of Computer Vision*, 131(5):1284–1298, 2023.