IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

SM-NMPC: Sliding Mode-Based Nonlinear Model Predictive Control for UAVs under Degraded Motor on Microcontrollers

Van Chung Nguyen, An Duy Nguyen, Pratik Walunj, Chuong Le, and Hung Manh La

***Abstract*—This paper presents a novel Sliding Mode-Based Nonlinear Model Predictive Control (SM-NMPC) for controlling Unmanned Aerial Vehicles (UAVs) such as Quadrotors and a 10-propeller drone (Cube-Drone). The proposed method combines Aggregated Hierarchical Sliding Mode Control (AHSMC) strategies with Nonlinear Model Predictive Control (NMPC), designed to operate on resource-constrained microcontrollers. First, an AHSMC that provided a virtual input reference is introduced to ensure the UAV's robustness, which is then leveraged by the NMPC to solve the optimization problem. A comprehensive comparison to existing approaches in terms of stability and computational efficiency demonstrates that the SM-NMPC framework excels, enabling quadrotor UAVs to accurately track reference trajectories even in the presence of a degraded motor. The proposed method also showcases the capability to implement robust optimal control on a microcontroller. Extensive experiments, both on real UAVs and their physical models in Gazebo/ROS2, are conducted to validate the effectiveness of the approach. A comparison to other state-of-the-art controllers further highlights the feasibility and superior performance of the proposed methodology. The open-source code has also been made available for further investigation: https://github.com/aralab-unr/SM-NMPC.**

***Index Terms*—Nonlinear Model Predictive Control, Sliding Mode-Based Control, Lyapunov Stability, Resource-Constrained Microcontroller, UAVs, Degraded Motor.**

I. INTRODUCTION

The pervasive integration of Unmanned Aerial Vehicles (UAVs) across diverse sectors, including aerial photography, agriculture, disaster response, and surveillance, has created many new opportunities in modern industry [1]. This technological advancement has revolutionized operational methodologies and attracted immense interest and engagement within both industrial and academic spheres. However, the design of controllers that allow the UAVs to perform accurately and effectively even under disturbances and uncertainty is necessary. Recently, numerous review papers [2] show that the controller for UAVs encompasses three primary categories: linear controllers, nonlinear controllers, and intelligent controllers. While linear controllers, such as the PID controller referenced in [3], [4], or the LQR controller mentioned in [5], can assist UAVs in tracking the desired trajectory, they often suffer from low performance. Additionally, linear controllers' robustness is contingent on various factors when they are highly sensitive to external disturbances and model uncertainties. Conversely, intelligent controllers, including artificial neural networks as discussed in [6], or reinforcement learning approaches as seen in [7], provide better solutions for enhanced trajectory tracking and improved robustness for UAVs. In addition, some intelligent controllers do not require a model of the UAVs as seen in [8]. Nevertheless, the application of intelligent controllers demands substantial data for learning and incurs high costs when applied to UAVs systems. The comparison of the proposed method to the other related works is provided in Table I.

To address the challenges associated with tracking performance, system stability, and cost reduction, nonlinear controllers have been widely applied in UAVs. Various nonlinear controllers, such as the backstepping controller [9], sliding mode control (SMC) [10], [11], [12], potential field (PF) control [13], [16], and nonlinear model predictive control (NMPC) [14], [15], have been successfully implemented in UAVs, demonstrating their effectiveness. However, the performance of nonlinear control laws depends significantly on the controller's design and parameter selection to ensure alignment with the UAV's model.

Since UAVs are underactuated systems [10], maintaining stability during control remains a significant challenge for developers. In [10], a second-order sliding mode control approach was proposed for quadrotor UAVs. This controller divides the UAV system into two parts: the actuated system (z, ψ) and the unactuated system (x, y, ϕ, θ). Controllers were then designed for each subsystem, and stability was analyzed using the Lyapunov function. The Lyapunov stability method also offers a promising solution for handling model uncertainties, as demonstrated in adaptive approaches [17], [18]. In [17], finite-time adaptive sliding mode control was employed to estimate internal moments and the UAV's mass. Additionally, in [19], sliding mode control driven by sliding mode disturbance observers was proposed to handle motor failure in UAVs. Another effective strategy is Nonlinear Model Predic-

Manuscript received: April, 18, 2025; Revised July, 19, 2025; Accepted August, 28, 2025.

This paper was recommended for publication by Editor Lucia Pallottino upon evaluation of the Associate Editor and Reviewers' comments.

This work was funded by the U.S. Department of Transportation under award number 69A3552348322 (Sub-award No. 000794), the NASA EPSCoR under grant number 80NSSC24M0141, the U.S. National Science Foundation (NSF) under grant NSF-CAREER: 1846513, and NSF-OIA-2148788 (Sub-award No. 24-52). The views, opinions, findings, and conclusions reflected in this publication are solely those of the authors and do not represent the official policy or position of the U.S. DOT, NASA, and NSF.

The authors are with the Advanced Robotics and Automation (ARA) Lab, Department of Computer Science and Engineering, University of Nevada, Reno, NV 89557, USA.

Corresponding author: Hung La, email: hla@unr.edu

Digital Object Identifier (DOI): see top of this page.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

TABLE I
QUADROTOR UAVS' CONTROLLERS COMPARISON. SYMBOL: ADDRESSED √, NOT ADDRESSED □

Method	Tracking ability	Stability under Degraded Motor	Complexity	Hardware	Results	Source codes
PID [3], [4]	√	□	Low	□	Simulink/Experiments	□
LQR [5]	√	□	Low	□	AirSim	□
Intelligent control [6], [8]	√	□	High	Raspberry Pi 3B Navio2	Experiments	□
Backstepping [9]	√	□	Low	Host Computer	Simulink & Experiments	□
SMC [10], [11], [12]	√	√	Low	On-board Computer	Simulink & Experiments	□
PF [13]	√	□	Low	□	Simulink	□
NMPC [14], [15]	√	√	High	Host/On-board Computer	Experiments	□
SM-NMPC (our)	√	√	High	Teensy 4.1	Gazebo & Experiments	√

tive Control (NMPC) [15]. In [15], linear parameter-varying and tube model predictive control methods were introduced to enhance UAVs control performance. These strategies have shown improved tracking performance compared to conventional approaches. However, integrating MPC into real-world UAV operations requires substantial computational resources, leading to increased implementation costs. Moreover, these control methodologies have not fully addressed UAV stability and the recursive feasibility of MPC, which warrant further investigation. Recent implementations of NMPC typically rely on powerful onboard computers or external host machines to solve optimization problems. This dependence poses a significant barrier to deploying NMPC on resource-constrained microcontrollers, which are increasingly used in embedded systems, robotics, and other applications.

To enable efficient real-time NMPC implementations under such constraints, code generation has emerged as a promising solution. Tools like AutoGenU [20] generate source code for the continuation/GMRES method, while solvers such as IPOPT (used in CasADi [21]) and OSQP [22] provide additional optimization capabilities. Similarly, CVXGEN [23] allows users to generate customized interior-point solvers for small-scale linear and quadratic programming problems, while qpOASES [24] employs active-set algorithms, making it another viable choice for MPC-based quadratic programming. Inspired by the ACADO Code Generation tool, which has successfully implemented NMPC on resource-limited platforms such as the Raspberry Pi [25], this paper presents a solution for implementing NMPC on the resource-constrained Teensy 4.1 microcontroller.

Building on the advantages of the Lyapunov stability method and the optimization capabilities of NMPC, this paper introduces a novel approach to UAV control: Aggregated Hierarchical Sliding Mode Control (AHSMC) within a Nonlinear Model Predictive Control framework. First, AHSMC formulates a nonlinear control law and provides its stability analysis. Then, NMPC leverages this virtual control law to solve the optimization problem. Furthermore, this study evaluates the proposed approach using a UAV model under a single-propeller failure scenario. The results demonstrate the effectiveness of the proposed control method to stabilize the UAVs even when reducing the throttle. The main contributions of our proposed method are outlined as follows:

- We introduce an SM-NMPC framework for quadrotor UAVs running on microcontrollers. Compared to existing methods, the proposed controller leverages the stability guarantees of the Lyapunov-based Sliding Mode Control (SMC) while optimizing control performance through NMPC. To the best of our knowledge, this is the first implementation of real-time NMPC on a resource-constrained microcontroller that also integrates the stability benefits of a Sliding Mode Controller.
- We provide the hardware setup, experimental results, and simulations of the quasi-physical model on Gazebo/ROS2 for the proposed method on Quadrotor UAVs and the Cube-Drone, along with comparisons to other related methods.
- Open source code and instructions are available for further investigation

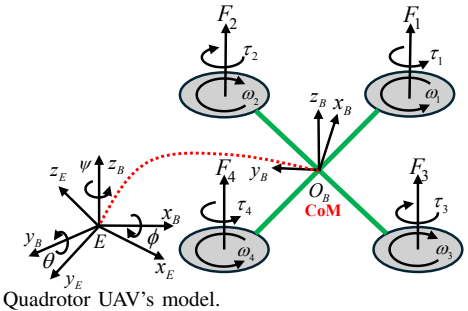

Fig. 1. Quadrotor UAV's model.

The rest of the paper is organized as follows: Section II introduces the UAV model and the proposed method. Section III presents simulation results in Gazebo/ROS2, as well as experimental results conducted on a single MCU for the UAVs. Finally, Section IV concludes the paper.

II. METHODOLOGY

In this research, we consider the full kinematic model of the UAVs as in Figure 1 as follows:

$$m\ddot{x} = (s_\psi s_\phi + c_\psi s_\theta c_\phi)U_1 - K_{dx}\dot{x} \quad (1)$$
$$m\ddot{y} = (s_\psi s_\theta c_\phi - c_\psi s_\phi)U_1 - K_{dy}\dot{y} \quad (2)$$
$$m\ddot{z} = c_\theta c_\phi U_1 - mg - K_{dz}\dot{z} \quad (3)$$
$$I_{xx}\ddot{\phi} = \dot{\theta}\dot{\psi}(I_{yy} - I_{zz}) + U_2 \quad (4)$$
$$I_{yy}\ddot{\theta} = \dot{\psi}\dot{\phi}(I_{zz} - I_{xx}) + U_3 \quad (5)$$
$$I_{zz}\ddot{\psi} = \dot{\phi}\dot{\theta}(I_{xx} - I_{yy}) + U_4, \quad (6)$$

where the control inputs are U_1, U_2, U_3, U_4, m is the mass of the UAV; K_{dx}, K_{dy}, K_{dz} are the drag coefficients; x, y, z denote the position of the UAVs; ϕ, θ, ψ denote the roll, pitch, and yaw angles of the UAVs, respectively; I_{xx}, I_{yy}, I_{zz} are the rotational inertias; and c and s are the shorthand notations for cosine and sine, respectively. The control inputs are defined:

$$\begin{bmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \end{bmatrix} = \begin{bmatrix} k_t & k_t & k_t & k_t \\ -lk_t & lk_t & -lk_t & lk_t \\ -lk_t & -lk_t & lk_t & lk_t \\ k_d & -k_d & -k_d & k_d \end{bmatrix} \begin{bmatrix} w_1^2 \\ w_2^2 \\ w_3^2 \\ w_4^2 \end{bmatrix}. \quad (7)$$

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

Due to the clockwise orientation of w_i in Figures 1, the propeller's velocity is defined as:

$$w_1 = -\sqrt{\frac{U_1 k_d l - U_2 k_d - U_3 k_d - U_4 k_t l}{4 k_d k_t l}} \tag{8}$$

$$w_2 = \sqrt{\frac{U_1 k_d l + U_2 k_d - U_3 k_d + U_4 k_t l}{4 k_d k_t l}} \tag{9}$$

$$w_3 = \sqrt{\frac{U_1 k_d l - U_2 k_d + U_3 k_d + U_4 k_t l}{4 k_d k_t l}} \tag{10}$$

$$w_4 = -\sqrt{\frac{U_1 k_d l + U_2 k_d + U_3 k_d - U_4 k_t l}{4 k_d k_t l}}. \tag{11}$$

where l indicates the distance from the motor to the center of mass (CoM) in the body frame, and k_t, k_d represent the thrust and torque coefficients, respectively.

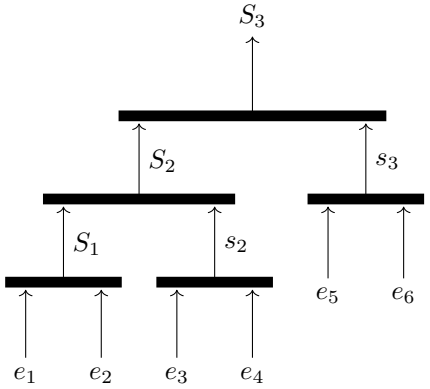

Fig. 2. Structure of AHSMC.

A. Nonlinear Model Predictive Control

To control the UAVs, we consider the NMPC in the following form:

$$\min_{\mathbf{x}(\cdot),\mathbf{u}(\cdot)} J(\mathbf{x}(\cdot),\mathbf{u}(\cdot)) = \|\mathbf{x}(t+T) - \mathbf{x}^{\mathrm{r}}(t+T)\|_R^2 + \int_t^{t+T} (\|\mathbf{x}(\tau) - \mathbf{x}^{\mathrm{r}}(\tau)\|_P^2 + \|\mathbf{u}(\tau) - \mathbf{u}^{\mathrm{r}}(\tau)\|_Q^2)\, d\tau \tag{12}$$

s.t.

$$\mathbf{x}(t) = \mathbf{x}_t \tag{13}$$

$$\dot{\mathbf{x}}(\tau) = \mathbf{f}(\mathbf{x}(\tau), \mathbf{u}(\tau)) \quad \text{for all } \tau \in [t, t+T] \tag{14}$$

$$\underline{\mathbf{u}} \le \mathbf{u}(\tau) \le \overline{\mathbf{u}} \quad \text{for all } \tau \in [t, t+T] \tag{15}$$

$$\underline{\mathbf{x}} \le \mathbf{x}(\tau) \le \overline{\mathbf{x}} \quad \text{for all } \tau \in [t, t+T]. \tag{16}$$

Here, $\mathbf{x}$ and $\mathbf{u}$ represent the differential state and control input, respectively; t denotes the current running time of the NMPC. $\mathbf{x}_t$ represents the initial state, which is the current state of the robot, and T denotes the length of the prediction horizon. The upper and lower constraints for the control input and differential states are $\underline{\mathbf{u}}$, $\overline{\mathbf{u}}$, $\underline{\mathbf{x}}$, and $\overline{\mathbf{x}}$, respectively. The cost function objective is weighted with positive-definite matrices P, Q, and R. Additionally, the variables $\mathbf{x}^{\mathrm{r}}$ and $\mathbf{u}^{\mathrm{r}}$ denote the reference for the system, and the term $\|\mathbf{x}(t+T) - \mathbf{x}^{\mathrm{r}}(t+T)\|$ represents the terminal cost for the NMPC. Note that this problem can be identified as a parameterized optimal control problem (OCP), where the input is the initial condition, which is the current state of the robot $\mathbf{x}_t$. The NMPC solves the OCP at each time step based on the initial condition and provides the control input for the system. Moreover, the cost function is extended to follow a desired virtual control reference $\mathbf{u}^{\mathrm{r}}$, to facilitate the stability and tracking performance of the controller. The desired control law is provided by the AHSMC, which will be proposed in the next subsection.

B. Aggregated Hierarchical Sliding Mode Control

In order to apply the HSMC for the UAVs, the model of the UAVs in (1)-(6) can be rewritten as:

$$\begin{cases} \dot{x}_1 = x_2,\ \dot{x}_2 = f_x + b_x u_1 \\ \dot{x}_3 = x_4,\ \dot{x}_4 = f_y + b_y u_1 \\ \dot{x}_5 = x_6,\ \dot{x}_6 = f_z + b_z u_1 \\ \dot{x}_7 = x_8,\ \dot{x}_8 = f_\phi + b_\phi u_2 \\ \dot{x}_9 = x_{10},\ \dot{x}_{10} = f_\theta + b_\theta u_3 \\ \dot{x}_{11} = x_{12},\ \dot{x}_{12} = f_\psi + b_\psi u_4, \end{cases} \tag{17}$$

where $x_1 = x$, $x_2 = \dot{x}$, $x_3 = y$, $x_4 = \dot{y}$ $x_5 = z$, $x_6 = \dot{z}$, $x_7 = \phi$, $x_8 = \dot{\phi}$, $x_9 = \theta$, $x_{10} = \dot{\theta}$, $x_{11} = \psi$, $x_{12} = \dot{\psi}$, $s_{x_i} = sin(x_i)$, $c_{x_i} = cos(x_i)$, $[u_1, u_2, u_3, u_4] = [U_1, U_2, U_3, U_4]$, and $f_i, b_i (i = x, y, z, \phi, \theta, \psi)$ can be identified based on (1)-(6), respectively. First, we propose the Aggregated HSMC (AHSMC) for the position control. Inspired by the idea in [26], the AHSMC is designed for three class variables x, y, z with one control input u_1. The structure of AHSMC is depicted in Figure 2. The basic idea of AHSMC is to pair two variable states as a lower layer, and the higher layer considers the lower layer. The lower layer is defined as follows:

$$s_1 = c_1 e_1 + e_2,\ s_2 = c_2 e_3 + e_4,\ s_3 = c_3 e_5 + e_6, \tag{18}$$

where c_1, c_2, c_3 are positive numbers, $e_1 = x_r - x$, $e_2 = \dot{x}_r - \dot{x}$, $e_3 = y_r - y$, $e_4 = \dot{y}_r - \dot{y}$, $e_5 = z_r - z$, $e_6 = \dot{z}_r - \dot{z}$ are control errors. The higher layers are defined as:

$$S_1 = s_1,\ S_2 = \lambda_1 S_1 + s_2,\ S_3 = \lambda_2 S_2 + s_3, \tag{19}$$

where λ_1, λ_2 are control parameter. The equivalent controller that makes the sliding surface's derivative equal to zero is calculated based on:

$$\begin{aligned} \dot{s}_1 &= c_1 \dot{e}_1 + \dot{e}_2 = c_1(\dot{x}_r - \dot{x}) + (\ddot{x}_r - \ddot{x}) \\ &= c_1(\dot{x}_r - \dot{x}) + (\ddot{x}_r - f_x - b_x u_1). \end{aligned} \tag{20}$$

Therefore, the equivalent control law for the sliding surface s_1 is chosen as:

$$u_{eqx} = \frac{c_1(\dot{x}_r - \dot{x}) + \ddot{x}_r - f_x}{b_x}. \tag{21}$$

Similar to the equivalent control law of sliding surface s_2, s_3:

$$u_{eqy} = \frac{c_2(\dot{y}_r - \dot{y}) + \ddot{y}_r - f_y}{b_y} \tag{22}$$

$$u_{eqz} = \frac{c_3(\dot{z}_r - \dot{z}) + \ddot{z}_r - f_z}{b_z}. \tag{23}$$

The switching control law of the AHSMC is proposed as:

$$\begin{aligned} u_{sw} = &- \frac{\lambda_1 \lambda_2 b_x (u_{eqy} + u_{eqz}) + \lambda_2 b_y (u_{eqx} + u_{eqz})}{\lambda_2 \lambda_1 b_x + \lambda_2 b_y + b_z} \\ &- \frac{b_z (u_{eqx} + u_{eqy}) - K_a S_3 - \eta \mathrm{sat}(S_3)}{\lambda_2 \lambda_1 b_x + \lambda_2 b_y + b_z}, \end{aligned} \tag{24}$$

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

where K_a, η are positive numbers, and $\text{sat}(S_3)$ is the saturation sign function, which can be defined as in (33).

Theorem 2.1: Considering the UAVs with the state space (17), if the AHSMC controller is proposed as:

$$u_{1r} = u_{eqx} + u_{eqy} + u_{eqz} + u_{sw}, \tag{25}$$

then the sliding surface S_3 is asymptotically stable.

Proof: According to Lyapunov's stability, considering the Lyapunov candidate as:

$$V_a = \frac{1}{2} S_3^2. \tag{26}$$

Differentiating V_a with respect to time t, the derivative of V_a yield is expressed as:

$$\dot{V}_a = S_3 \dot{S}_3 = S_3(\lambda_1 \lambda_2 \dot{s}_1 + \lambda_2 \dot{s}_2 + \dot{s}_3). \tag{27}$$

Substituting the control law (25) ,(20) and the similar $\dot{s}_2, \dot{s}_3$ into the state space (27), the derivative becomes:

$$\begin{aligned} \dot{V}_a =& - S_3 \Big[\lambda_1 \lambda_2 b_x (u_{eqy} + u_{eqz} + u_{sw}) + \lambda_2 b_y (u_{eqx} \\ &+ u_{eqz} + u_{sw}) + b_z (u_{eqx} + u_{eqy} + u_{sw}) \Big] \\ =& - K_a S_3^2 - \eta S_3 \text{sat}(S_3). \end{aligned} \tag{28}$$

Hence, the sliding surface is stable. Moreover, applying Barbalat's lemma [27], the sliding surface will converge to zero. Since the sliding surface S_3 is asymptotically stable from the time domain $[0, t_f]$, the stability of sliding surfaces S_1, S_2 can be achieved in the time domain $[t_f, \infty]$. The proof of this stability is in [26]. ■

For the attitude controller of the UAVs, the SMC attitude controller (SMC-AC) is proposed to create the desired roll and pitch angles for the UAVs. For the attitude controller, we consider ϕ, θ, ψ (the roll, pitch, yaw angles) of the UAVs as the class of actuated states, which can be controlled by u_2, u_3, u_4, respectively. Considering the sliding surface as:

$$s_\phi = c_\phi e_7 + e_8, \tag{29}$$

where c_ϕ is a positive constant, and $e_7 = \phi - \phi_r, e_8 = \dot{\phi} - \dot{\phi}_r$, and ϕ_r is the desired roll angle.
The Lyapunov function is chosen as:

$$V_\phi = \frac{1}{2} s_\phi^2, \tag{30}$$

and its derivative:

$$\begin{aligned} \dot{V}_\phi &= s_\phi \dot{s}_\phi = s_\phi (c_\phi (\dot{\phi} - \dot{\phi}_r) + \ddot{\phi} - \ddot{\phi}_r) \\ &= s_\phi (c_\phi (\dot{\phi} - \dot{\phi}_r) + \ddot{\phi} - \ddot{\phi}_r). \end{aligned} \tag{31}$$

Therefore, the virtual reference control law u_{2r} is proposed as:

$$\begin{aligned} u_{2r} = \frac{1}{b_\phi} [&- K_y c_\phi \phi - (c_\phi + K_y) \dot{\phi} + K_y c_\phi \phi_r + (c_\phi + K_y) \dot{\phi}_r \\ &- f_\phi - K_\phi \text{sat}(s_\phi) + \ddot{\phi}_r], \end{aligned} \tag{32}$$

where K_y, K_ϕ are positive numbers, $\text{sat}(s_\phi)$ is the saturation sign function, which is defined by:

$$\text{sat}(s_\phi) = \begin{cases} -0.1 \text{ if } s_\phi \leq -0.1 \\ s_\phi \text{ if } -0.1 < s_\phi < 0.1 \\ 0.1 \text{ if } s_\phi \geq 0.1. \end{cases} \tag{33}$$

Substituting the control law (32) and the dynamic equation (17) into (31), the derivative of the Lyapunov candidate becomes:

$$\begin{aligned} \dot{V}_\phi &= s_\phi [- K_y c_\phi (\phi - \phi_r) - K_y (\dot{\phi} - \dot{\phi}_r) - K_\phi \text{sat}(s_\phi)] \\ &= - K_y s_\phi^2 - s_\phi K_\phi \text{sat}(s_\phi). \end{aligned} \tag{34}$$

Hence, the system is stable. Moreover, applying Barbalat's lemma [27], the sliding surface will converge to zero, which means $c_\phi e_7 + e_8 \to 0$. Solving the differential equation, we have the control error $\phi - \phi_r \to 0$, and then the UAVs can track references. For the pitch and yaw angles, let's define the sliding surface as:

$$s_\theta = c_\theta e_9 + e_{10} \tag{35}$$

$$s_\psi = c_\psi e_{11} + e_{12}. \tag{36}$$

Defining the Lyapunov function as:

$$V_\theta = \frac{1}{2} s_\theta^2 \tag{37}$$

$$V_\psi = \frac{1}{2} s_\psi^2. \tag{38}$$

Using the same technique with the roll angle, the virtual reference control input u_{3r}, u_{4r} can be obtained by:

$$\begin{aligned} u_{3r} = \frac{1}{b_\theta} [&- K_x c_\theta \theta - (c_\theta + K_x) \dot{\theta} + K_x c_\theta \theta_r + (c_\theta + K_x) \dot{\theta}_r \\ &- f_\theta - K_\theta \text{sat}(s_\theta) + \ddot{\theta}_r] \end{aligned} \tag{39}$$

$$\begin{aligned} u_{4r} = \frac{1}{b_\psi} [&- K_z c_\psi \psi - (c_\psi + K_z) \dot{\psi} + K_z c_\psi \psi_r + (c_\psi + K_z) \dot{\psi}_r \\ &- f_\psi - K_\psi \text{sat}(s_\psi) + \ddot{\psi}_r], \end{aligned} \tag{40}$$

then the SMC can help the UAVs track the roll-pitch-yaw references. However, for the reference roll and pitch angle, the roll angle is approximated as the first-order system of $(y_r – y)$, where y_r is the desired trajectory. Therefore, the roll and the pitch references, respectively, are defined as:
Roll controller:

$$\text{Roll}_r = k_{py}(y_r - y) - k_{dy}(\dot{y}_r - \dot{y}), \tag{41}$$

Pitch controller:

$$\text{Pitch}_r = k_{px}(x - x_r) - k_{dx}(\dot{x} - \dot{x}_r), \tag{42}$$

where $k_{px}, k_{dx}, k_{py}, k_{dy}$ are the controller parameters, respectively. Following the rotational direction of the roll, pitch, and yaw angles, the desired roll and pitch angles, respectively, are calculated as follows:

$$\phi_r = \text{Roll}_r \cos(\psi) + \text{Pitch}_r \sin(\psi) \tag{43}$$

$$\theta_r = -\text{Roll}_r \sin(\psi) + \text{Pitch}_r \cos(\psi). \tag{44}$$

Here, the virtual reference $\mathbf{u}^{\text{r}} = [u_{1r}, u_{2r}, u_{3r}, u_{4r}]^T$ proposed by AHSMC will be fed to the NMPC. The NMPC will solve the optimization problem by tracking the reference and the virtual input to ensure the stability of the system.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

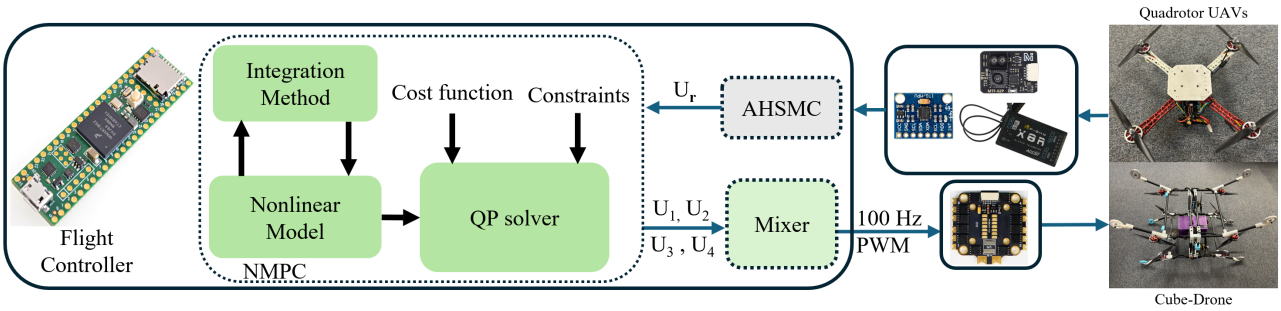

Fig. 3. Close-loop control structure & hardware setup of the proposed method.

C. ACADO code generation and online QP solver qpOASES

In this section, we provide a brief overview of C++ ACADO code generation and the qpOASES solver to embed the code to the resources-constrainted microcontroller. An example of the generated code, along with instructions on how to generate the code and include qpOASES, can be found in [28].

ACADO code generation is designed to address nonlinear optimal control problems, which are typically formulated as general quadratic programs (QPs). To solve these nonlinear optimal control problems, ACADO discretizes the nonlinear dynamics using the multiple shooting method [29]. This approach transforms the problem into a nonlinear programming problem (NLP), which can then be solved using the sequential quadratic programming (SQP) method. Furthermore, within the tracking term of the cost function in (12), ACADO employs the Gauss-Newton method to approximate the Hessian matrices. This enables the generation of real-time iterations, where only one SQP step is required per time step. To solve the SQP problem, ACADO incorporates two tailored algorithms: CVXGEN [23] and qpOASES [24]. CVXGEN utilizes a primal-dual interior-point solver, and the exported code is implemented in plain C, using only static memory. This results in relatively constant computation times for each time step. On the other hand, qpOASES employs active-set algorithms, which represent a second class of QP solvers. For ACADO code generation, modifications such as hard-coded dimensions and static memory allocation are utilized. While the computation time for active-set methods is harder to predict due to its dependence on the number of active-set changes, it benefits from dedicated hot-starting capabilities and is generally faster than interior-point iterations.

Given these advantages, we employ the qpOASES solver in this paper. The implementation of NMPC on resource-constrained microcontrollers requires minimal static memory consumption and a fast algorithm to ensure real-time performance.

III. RESULTS

In this section, we present simulation results conducted in Gazebo/ROS2 alongside experimental outcomes on real UAVs: The Quadrotor UAVs and the Cube-Drone to demonstrate the effectiveness and feasibility of our proposed method. Additionally, a comparison to the Second-Order Sliding Mode Control [10] and the Model Predictive Contouring Control (MPCC) [30] is included to highlight the robustness of the hardware setup and the superior performance of our control approach. The UAVs models are designed in SolidWorks, and URDF files are exported for use in the robot model. The robot and control parameters are set as shown in Table II. We set the initial position of the flying robot as $[x_0, y_0, z_0] = [0, 0, 0.15]$, and the desired trajectory is set as:

TABLE II
ROBOT & CONTROL PARAMETERS

Quadrotor UAVs	SM-NMPC controller
$m_{QUAD} = 1.85(kg)$	$K_x = K_y = 32.5, K_z = 27.5$
$m_{CUBE} = 5.0(kg)$	$c_\phi = c_\theta = 12.5, c_\psi = 12.5$
$I_{xx} = I_{yy} = 0.0785$	$K_\phi = K_\theta = 1.75, K_\psi = 0.75$
$I_{zz} = 0.105$	$c_1 = c_2 = 0.015, c_3 = 0.375$
$K_{dx} = K_{dy} = 0.0000267$	$\lambda_1 = \lambda_2 = 0.05, K_a = 10.75, \eta = 0.25$
$K_{dz} = 0.0000625$	$k_{px} = k_{py} = 0.05, k_{dx} = k_{dy} = 0.1$
$l_{QUAD} = 0.149(m)$	$P = \text{diag}(12.75, 75.5, 75.5, 75.5)$
$lx_{CUBE} = 0.219(m)$	$Q = \text{diag}(70.5, 70.5, 70.5, 70.5)$
$ly_{CUBE} = 0.1845(m)$	$R = \text{diag}(10.25, 10.25, 10.25, 10.25)$

$$\begin{bmatrix} x_r(t) \\ y_r(t) \end{bmatrix}' = \begin{cases} [5,\ 5], & 0 \le t \le 10 \\ [5,\ -5], & 10 < t \le 20 \\ [-5,\ -5], & 20 < t \le 30 \\ [-5,\ 5], & 30 < t \le 40 \\ [0,\ 0], & 40 < t \le 50 \\ [5\sin(0.157q)),\ 5\sin(0.314q)], & 50 < t \le 100 \\ [0,\ 0], & t > 100 \end{cases}$$

with $q = t - 50$, and the desired height of the robot is set as $z(t) = 5$. The simulation results are shown in two scenarios: with and without a degraded motor. In both scenarios, the control parameters remain the same to showcase the robustness of the controller under failure conditions.

A. Simulation under Normal Conditions

The simulation results are presented in Figures 4 to 9. The simulations are performed on a computer equipped with an Intel i7-9700F CPU, 16GB of RAM, and an NVIDIA GeForce RTX 2060 GPU. For tracking performance evaluation, the position and angular states of the robot are shown in Figures 4–6. Although the SO-SMC, AHSMC, and SM-NMPC exhibit small overshoots, MPCC can eliminate them because it incorporates navigation within the optimization problem. However, the tracking performance of SO-SMC, AHSMC, and SM-NMPC remains similar and does not experience small fluctuations like MPCC. Finally, we benchmark the performance of the controllers in Table III. The benchmark criteria include Overshoot (in %) (we take the maximum overshoot along x,y, and z-axis), Integral Square Error (ISE),

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

TABLE III
THE CONTROL PERFORMANCE BENCHMARK

Methods	Overshoot (%)	ITAE	IAE	ISE
SO-SMC [10]	18	6873	170	938
MPCC [30]	0.02	20397	419	2103
AHSMC (our)	38	6813	175	860
SM-NMPC (our)	18	6602	167	851
MPCC Failure	–	–	–	–
SO-SMC Failure	999	281693	4728	199259
AHSMC Failure	31	9662	220	985
SM-NMPC Failure	47	16517	337	1475

Integral of Time Multiplied Square Error (ITSE), and Integral Absolute Error (IAE).

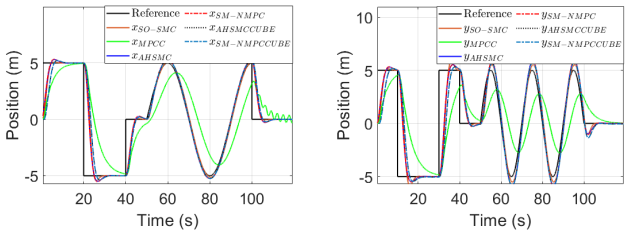

Fig. 4. Tracking performance along the x and y-axis.

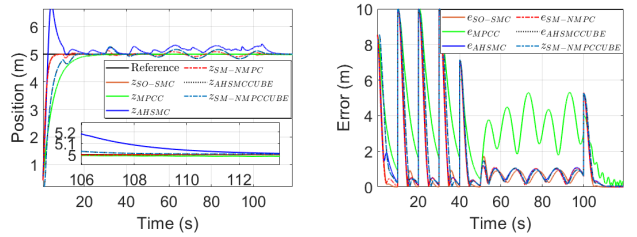

Fig. 5. Tracking performance along the z-axis and tracking error.

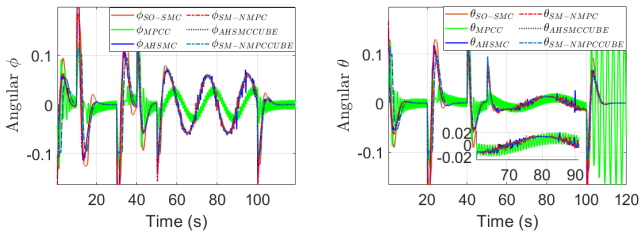

Fig. 6. The angular ϕ, and θ.

B. Simulation under Degraded Motor Scenarios

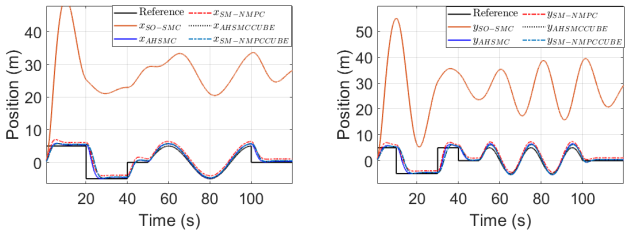

Fig. 7. Tracking performance along the x and y-axis under failure case.

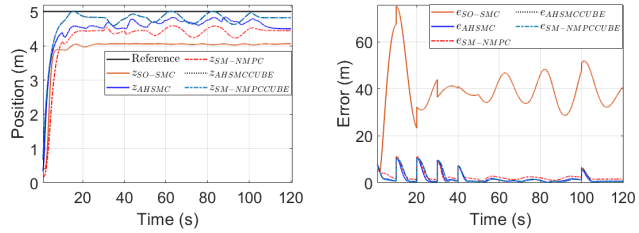

Fig. 8. Robot's position along the z-axis and tracking error in the failure case.

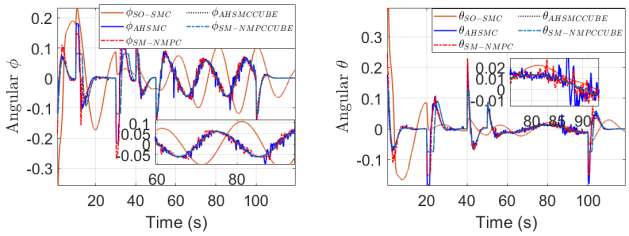

Fig. 9. The angular ϕ, and θ under failure case.

For the degraded motor case in the simulation, we reduced the throttle of one motor by 50%. Without loss of generality, we assume that motor 2 produces only 50% of the input throttle. We then test the stability and tracking performance of the proposed controller and other state-of-the-art controllers. Specifically, the MPCC fails to keep the UAVs airborne, while the SO-SMC, AHSMC, and SM-NMPC successfully stabilize the system. However, SO-SMC exhibits a large overshoot and fails to track the reference trajectory. The proposed method can track the reference but exhibits a small, steady-state error due to the lack of control forces.

Moreover, under these conditions, the results indicate that the sliding mode approach is robust to uncertainty, whereas the model predictive control approach relies on an accurate model of the robot. Therefore, by integrating both approaches, SM-NMPC leverages the stability properties of SMC while addressing model uncertainties. The results of this scenario are shown in Figures 7 to 9. The results demonstrate that the proposed SM-NMPC utilizes the optimal solution while maintaining the stability of AHSMC. Unlike AHSMC, the control law of SM-NMPC does not exhibit overshoot and can still stabilize the system. For the performance benchmark, it shows that the AHSMC performs well since there are no constraints for the control laws U_2, U_3, while the SM-NMPC has control input constraints that cause a larger static error under failure cases.

C. Experiments

We verified the efficiency of our control method for real-time execution on the resource-constrained Teensy 4.1 microcontroller and implemented our flight controller under a motor failure scenario to demonstrate the stability of the proposed controller. Additionally, we evaluated the SM-NMPC solving time on the Teensy 4.1, showing that our approach can achieve high-frequency execution on resource-constrained hardware despite the computational demands of the NMPC. The solving time and memory usage when executing the SM-NMPC on the Teensy 4.1 are shown in Figure 10. The Teensy 4.1 features an ARM Cortex-M7 microcontroller operating at 600 MHz, with 7.75 MB of flash memory and 512 kB of RAM. Therefore, the memory usage must not exceed this RAM limit. Moreover, the solving time must be fast enough for real-time implementation.

1) Hardware setup: The custom quadrotor runs on the Teensy 4.1 microcontroller, with ACADO code generation in C++ using the qpOASES solver. The generated code is then embedded in the microcontroller. The drone's angular and position data are obtained from the MPU 6050 and the Micoair MTF-02P optical flow and LiDAR sensor. We use the FrSky X8R radio receiver to send the desired commands to the UAVs.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

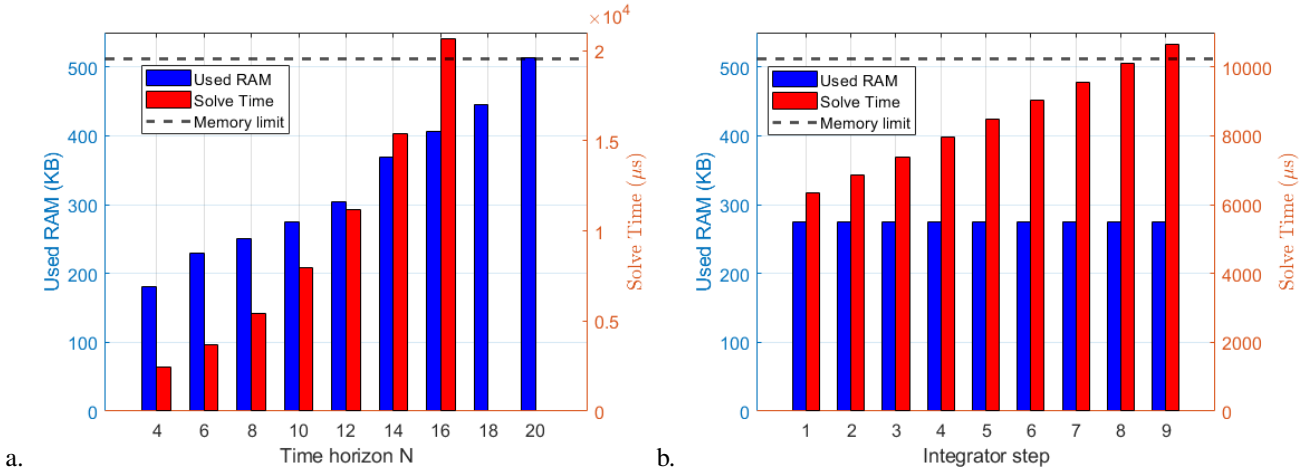

Fig. 10. A benchmark of memory usage and solution time for each interaction of embedded NMPC on the Teensy 4.1 development board (ARM Cortex-M7 running at 600 MHz with 32-bit floating-point support, 7.75 MB of flash, 512 kB of tightly coupled RAM1, and 512 kB of RAM2). In this research, we limited the memory usage to 512 kB for all variables and programmed the system to use only RAM1. At a, with the integrator step set to 4, the time horizon limit is 18, beyond which the microcontroller fails to solve the optimization problem. At b, with the time horizon set to 10, the memory consumption remains consistent regardless of the integrator step. However, as the integrator step increases, the solution time also increases. This represents a trade-off between the accuracy and the solution time of the SM-NMPC.

For the degraded motor configuration, the diameter of the right rear (motor 3) rotor propeller is reduced by 2 inches, as shown in Figure 14, which approximately reduces its effectiveness by 30% according to flight test results.

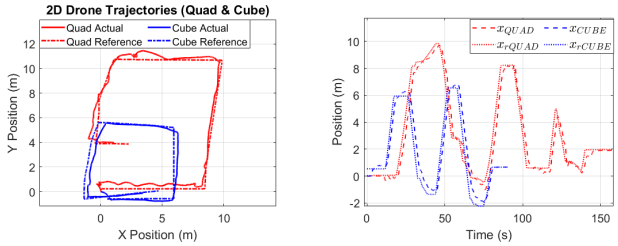

Fig. 11. Tracking performance and x-position in the normal case.

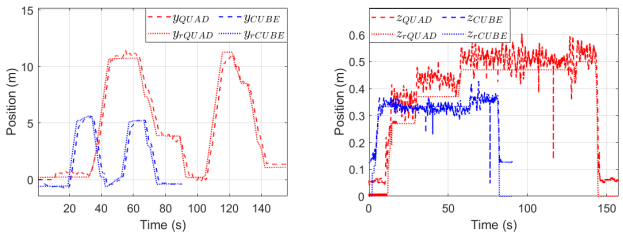

Fig. 12. Y-position and Z-position in the normal case.

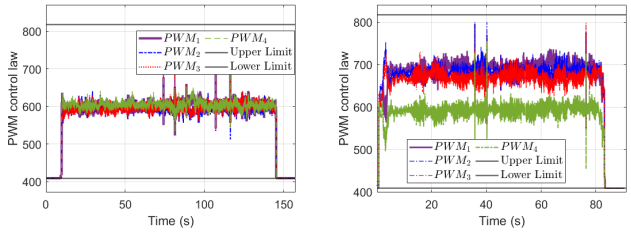

Fig. 13. PWM of normal conditions : (Right) Quadrotor, (Left) Cube-Drone.

2) Experimental Results in the Normal Scenario: The results of the experiment under normal conditions are presented in Figures 11 to 13. The results demonstrate that, under the proposed method, the quadrotor UAVs and the Cube-Drone can effectively track the reference trajectory. The PWM signal remains within the allowable range, thanks to the constraints imposed by the NMPC.

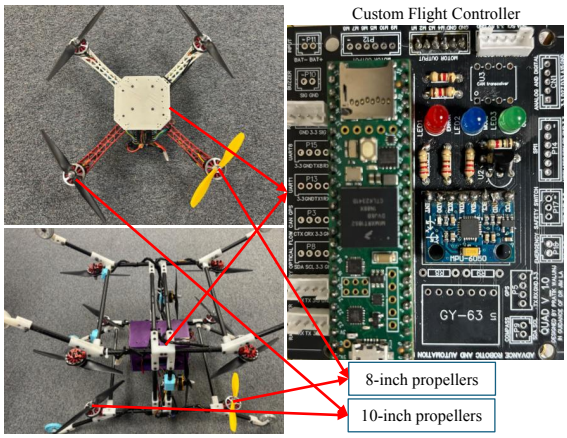

Fig. 14. The hardware setup under failure cases & the custom flight controller.

3) Experimental Results in the Degraded Motor Scenario: The experimental results under degraded motor scenarios are presented in Figures 15 to 17. These results demonstrate that, under the proposed method, both the quadrotor UAVs and the Cube-Drone can maintain stability and effectively track the reference trajectory despite the motor failure. A small steady-state error is observed, primarily due to the reduced thrust on motor 3. Additionally, the PWM signal for motor 3 is slightly higher than in the normal case to compensate for the failure; however, it remains within the allowable range, thanks to the constraints enforced by the NMPC. Notably, the Cube-Drone exhibits greater stability compared to the quadrotor UAVs. This is attributed to its octocopter configuration, where the top motor compensates for the failed one, distributing the thrust more effectively. Nonetheless, both aerial platforms can remain airborne and track the reference trajectory under the proposed control strategy.

IV. CONCLUSION

In this paper, we present an optimal controller for the UAVs, designed to run on resource-constrained microcontrollers, even

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

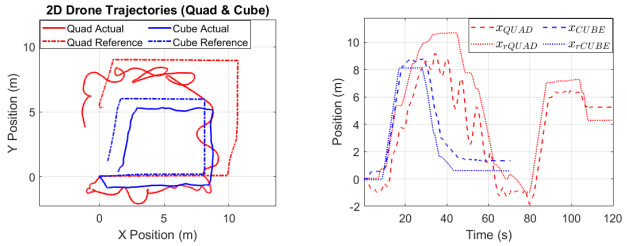

Fig. 15. Tracking performance along the x and y-axis under failure case.

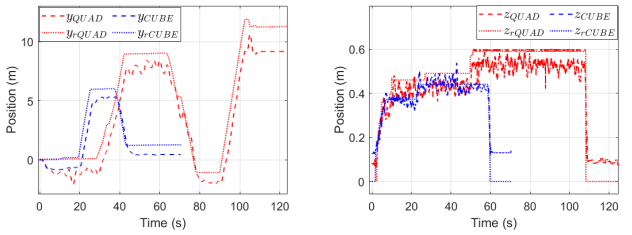

Fig. 16. Y-position and Z-position in the failure case.

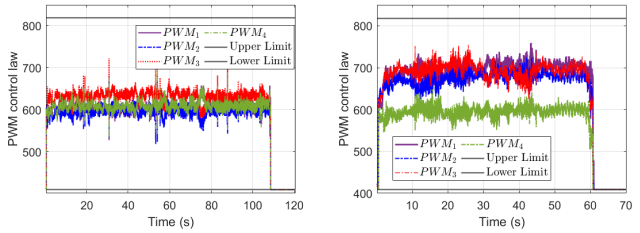

Fig. 17. PWM signals under failure conditions: (Right) Quadrotor, (Left) Cube-Drone.

in the presence of motor failure. By leveraging the advantages of AHSMC with Lyapunov stability, the NMPC ensures both tracking accuracy and system stability. The use of ACADO code generation facilitates the implementation of the proposed control on the microcontroller. In conclusion, the proposed method is not only easy to implement on microcontrollers but also ensures robustness in performance. Furthermore, it simplifies the hardware implementation, making it a promising approach for deploying optimal control on low-level platforms, not only for the UAVs but also for other systems. The open-source code has also been made available for further investigation and implementation of the optimal control on resource-constrained microcontrollers.

REFERENCES

[1] Y. Alghamdi, A. Munir, and H. M. La, "Architecture, classification, and applications of contemporary unmanned aerial vehicles," *IEEE Consumer Electronics Magazine*, vol. 10, no. 6, pp. 9–20, 2021.

[2] B. Han, Y. Zhou, K. K. Deveerasetty, and C. Hu, "A review of control algorithms for quadrotor," in *2018 IEEE international conference on information and automation (ICIA)*. IEEE, 2018, pp. 951–956.

[3] O. Bouaiss, R. Mechgoug, and R. Ajgou, "Modeling, control and simulation of quadrotor uav," in *2020 1st International Conference on Communications, Control Systems and Signal Processing (CCSSP)*. IEEE, 2020, pp. 340–345.

[4] H. X. Pham, H. M. La, D. Feil-Seifer, and L. V. Nguyen, "Autonomous uav navigation using reinforcement learning," *arXiv preprint arXiv:1801.05086*, 2018.

[5] A. S. Elkhatem and S. N. Engin, "Robust lqr and lqr-pi control strategies based on adaptive weighting matrix selection for a uav position and attitude tracking control," *Alexandria Engineering Journal*, vol. 61, no. 8, pp. 6275–6292, 2022.

[6] O. A. Hay, M. Chehadeh, A. Ayyad, M. Wahbah, M. A. Humais, I. Boiko, L. Seneviratne, and Y. Zweiri, "Noise-tolerant identification and tuning approach using deep neural networks for visual servoing applications," *IEEE Transactions on Robotics*, 2023.

[7] W. Koch, R. Mancuso, R. West, and A. Bestavros, "Reinforcement learning for uav attitude control," *ACM Transactions on Cyber-Physical Systems*, vol. 3, no. 2, pp. 1–21, 2019.

[8] H. Müller, V. Niculescu, T. Polonelli, M. Magno, and L. Benini, "Robust and efficient depth-based obstacle avoidance for autonomous miniaturized uavs," *IEEE Transactions on Robotics*, 2023.

[9] N. Koksal, H. An, and B. Fidan, "Backstepping-based adaptive control of a quadrotor uav with guaranteed tracking performance," *ISA transactions*, vol. 105, pp. 98–110, 2020.

[10] E.-H. Zheng, J.-J. Xiong, and J.-L. Luo, "Second order sliding mode control for a quadrotor uav," *ISA transactions*, vol. 53, no. 4, pp. 1350–1356, 2014.

[11] X. Shao, G. Sun, W. Yao, J. Liu, and L. Wu, "Adaptive sliding mode control for quadrotor uavs with input saturation," *IEEE/ASME Transactions on Mechatronics*, vol. 27, no. 3, pp. 1498–1509, 2021.

[12] V. C. Nguyen and H. M. La, "A class of hierarchical sliding mode control based on extended kalman filter for quadrotor uavs," *arXiv preprint arXiv:2504.02851*, 2025.

[13] A. Ma'Arif, W. Rahmaniar, M. A. M. Vera, A. A. Nuryono, R. Majdoubi, and A. Çakan, "Artificial potential field algorithm for obstacle avoidance in uav quadrotor for dynamic environment," in *2021 IEEE International Conference on Communication, Networks and Satellite (COMNETSAT)*. IEEE, 2021, pp. 184–189.

[14] K. Zhang, Y. Shi, and H. Sheng, "Robust nonlinear model predictive control based visual servoing of quadrotor uavs," *IEEE/ASME Transactions on Mechatronics*, vol. 26, no. 2, pp. 700–708, 2021.

[15] L. Cavanini, G. Ippoliti, and E. F. Camacho, "Model predictive control for a linear parameter varying model of an uav," *Journal of Intelligent & Robotic Systems*, vol. 101, no. 3, p. 57, 2021.

[16] A. C. Woods and H. M. La, "A novel potential field controller for use on aerial robots," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 49, no. 4, pp. 665–676, 2017.

[17] O. Mofid and S. Mobayen, "Adaptive sliding mode control for finite-time stability of quad-rotor uavs with parametric uncertainties," *ISA transactions*, vol. 72, pp. 1–14, 2018.

[18] S. Lian, W. Meng, Z. Lin, K. Shao, J. Zheng, H. Li, and R. Lu, "Adaptive attitude control of a quadrotor using fast nonsingular terminal sliding mode," *IEEE Transactions on Industrial Electronics*, vol. 69, no. 2, pp. 1597–1607, 2021.

[19] X. Wang, S. Sun, E.-J. van Kampen, and Q. Chu, "Quadrotor fault tolerant incremental sliding mode control driven by sliding mode disturbance observers," *Aerospace Science and Technology*, vol. 87, pp. 417–430, 2019.

[20] B. Houska, H. J. Ferreau, and M. Diehl, "Acado toolkit—an open-source framework for automatic control and dynamic optimization," *Optimal control applications and methods*, vol. 32, no. 3, pp. 298–312, 2011.

[21] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi – A software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation*, vol. 11, no. 1, pp. 1–36, 2019.

[22] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd, "OSQP: an operator splitting solver for quadratic programs," *Mathematical Programming Computation*, vol. 12, no. 4, pp. 637–672, 2020. [Online]. Available: https://doi.org/10.1007/s12532-020-00179-2

[23] J. Mattingley and S. Boyd, "Cvxgen: A code generator for embedded convex optimization," *Optimization and Engineering*, vol. 13, pp. 1–27, 2012.

[24] H. Ferreau, C. Kirches, A. Potschka, H. Bock, and M. Diehl, "qpOASES: A parametric active-set algorithm for quadratic programming," *Mathematical Programming Computation*, vol. 6, no. 4, pp. 327–363, 2014.

[25] S. Adhau, S. Patil, D. Ingole, and D. Sonawane, "Implementation and analysis of nonlinear model predictive controller on embedded systems for real-time applications," in *2019 18th European Control Conference (ECC)*. IEEE, 2019, pp. 3359–3364.

[26] D. Qian and J. Yi, "Hierarchical sliding mode control for under-actuated cranes," *Heidelberg, Ber: Springer*, 2016.

[27] J.-J. E. Slotine, "Applied nonlinear control," *PRENTICE-HALL google schola*, vol. 2, pp. 1123–1131, 1991.

[28] B. Houska, H. J. Ferreau, and M. Diehl, "An auto-generated real-time iteration algorithm for nonlinear mpc in the microsecond range," *Automatica*, vol. 47, no. 10, pp. 2279–2285, 2011.

[29] H. G. Bock and K.-J. Plitt, "A multiple shooting algorithm for direct solution of optimal control problems," *IFAC Proceedings Volumes*, vol. 17, no. 2, pp. 1603–1608, 1984.

[30] 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.