Servo Integrated Nonlinear Model Predictive Control for Overactuated Tiltable-Quadrotors

Jinjie Li[1], *Graduate Student Member, IEEE RAS*, Junichiro Sugihara[2], and Moju Zhao[1], *Member, IEEE*

***Abstract*—Utilizing a servo to tilt each rotor transforms quadrotors from underactuated to overactuated systems, allowing for independent control of both attitude and position, which provides advantages for aerial manipulation. However, this enhancement also introduces model nonlinearity, sluggish servo response, and limited operational range into the system, posing challenges to dynamic control. In this study, we propose a control approach for tiltable-quadrotors based on nonlinear model predictive control (NMPC). Unlike conventional cascade methods, our approach preserves the full dynamics without simplification. It directly uses rotor thrust and servo angle as control inputs, where their limited working ranges are considered input constraints. Notably, we incorporate a first-order servo model within the NMPC framework. Simulation reveals that integrating the servo dynamics is not only an enhancement to control performance but also a critical factor for optimization convergence. To evaluate the effectiveness of our approach, we fabricate a tiltable-quadrotor and deploy the algorithm onboard at 100 Hz. Extensive real-world experiments demonstrate rapid, robust, and smooth pose-tracking performance.**

***Index Terms*—Aerial Systems: Mechanics and Control, Motion Control, MPC, Overactuated Robots**

I. Introduction

AERIAL robots have increasingly attracted attention due to their mobility in three-dimensional space. As the most popular aerial robot, quadrotors have been applied in various areas including transportation, inspection, and search & rescue [1]. Traditional quadrotors are *underactuated*, meaning that the number of independent control inputs (i.e., the rotation speed of four vertically oriented rotors) is fewer than the required six degrees of freedom (DoF) for full-pose motion, leading to the independent control of only four flat outputs: three-axis position and yaw angle. However, numerous applications require independent attitude control, particularly in the field of aerial manipulation [2].

To overcome the challenges associated with *underactuation*, researchers have developed two primary ways to achieve *fullactuation* or *overactuation* in rotor-based aerial robots. One way involves adding rotors with fixed tilting angles [3], [4], named as *fixed-rotor* robots. While these robots maintain mechanical and control simplicity comparable to their underactuated counterparts, they encounter issues like constant internal forces, reduced efficiency, and constrained wrench generation in certain directions. Alternatively, the *tilt-rotor* design introduces servo modules to vary thrust directions, reducing internal force and enhancing energy efficiency. If we aim to enable conventional quadrotors to control independently both position and attitude while maintaining the symmetry of roll and pitch, a relatively straightforward idea is to add a servo-tilting mechanism to each rotor, allowing it to rotate along the arm. Therefore, our research focuses on the tiltable-quadrotor, as exemplified in Fig. 1.

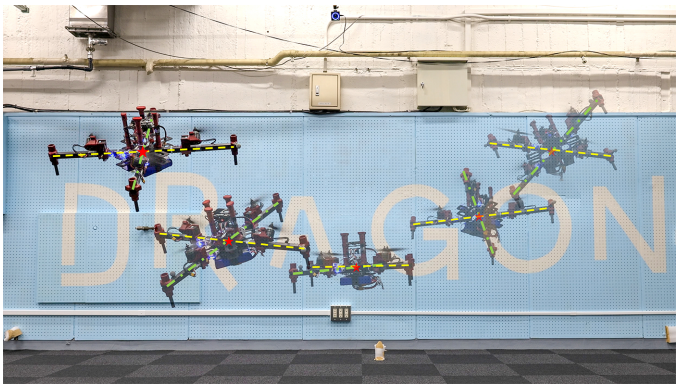


Fig. 1. Our self-made tiltable-quadrotor is tracking a pose trajectory based on the proposed NMPC method. It demonstrates the capability for independent control in position and attitude to track different Lemniscate curves.

The first real-world flight of a quadrotor equipped with tiltable rotors was achieved by Ryll et al., where the control approach is detailed in [5]. Subsequently, the Voliro project [6] introduced a hexacopter with tiltable rotors, marking a pioneering step towards real-world omnidirectional flight. In addition, Senkul et al. [7] developed a quadrotor capable of tilting its rotors along two axes, but they assumed the same thrust generation across all rotors, restricting the vehicle's maneuverability. Despite these advantages, flight control for tiltable-multirotors remains challenging.

The control complexity of tiltable-multirotors arises primarily from two factors. First, the introduction of additional DoFs to vary tilting angles significantly increases the system's nonlinearity. These extra DoFs make the system overactuated, resulting in multiple solutions in control allocation. Second, the inherent dynamics of tilting servos impede prompt motion control for agile trajectory tracking. This slow property can be caused by dead time or the servo's time constant, which has been reported as a critical factor in flight stability [5]. Similar challenges have been observed in other servo-equipped aerial robots, such as SPIDAR [8] and Perching Arm [9].

To address the first challenge, plenty of research has simplified this complex control problem by decoupling it into control and allocation components, as illustrated in Fig. 2a.

Manuscript received: April 5, 2024; Revised: July 6, 2024; Accepted: August 9, 2024. This letter was recommended for publication by Editor Jaydev P. Desai upon evaluation of the Reviewers' comments. *(Corresponding author: Moju Zhao)*

[1]J. Li, and M. Zhao are with the DRAGON Lab at the Department of Mechanical Engineering, the University of Tokyo, Tokyo, 113-8654, Japan {jinjie-li, chou}@dragon.t.u-tokyo.ac.jp

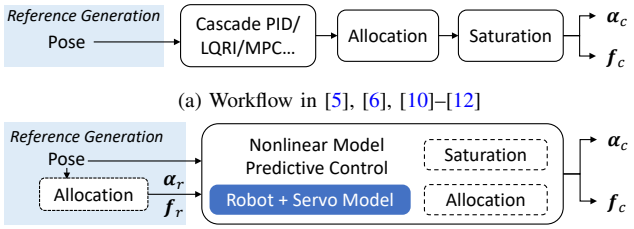
[2]J. Sugihara is with the JSK Lab at the Department of Mechano-Informatics, the University of Tokyo, Tokyo, 113-8654, Japan j-sugihara@jsk.imi.i.u-tokyo.ac.jp

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

For control, various strategies have been proposed based on established control theories, including feedback linearization [5], nonlinear inverse dynamics [10], cascade PID [6], LQRI [11], and NMPC [12]. For allocation, Moore–Penrose inverse is typically utilized to map control inputs (for most cases, wrench) to specific actuator commands. Although these approaches demonstrate effectiveness in real-world flights, their cascade structure may not fully leverage the potential of overactuated systems or elegantly address actuators' delay and constraints. To overcome these drawbacks, recent studies have started to integrate control and allocation within a single optimization framework, notably through nonlinear model predictive control (NMPC). This integrated approach has shown promise in conventional quadrotors [13], [14] and is gaining traction for tiltable-multirotors. For instance, Bicego et al. [15] proposed an NMPC framework suitable for various multirotor designs, tested on a hexrotor that all motors can tilt for the same angle. However, its applicability to drones with independently tiltable rotors remains unverified. Shawky et al. [16] developed an NMPC controller for tiltable hexacopters, albeit only validated in Gazebo simulation. Despite these advancements, the deployment of unified NMPC in real-world tiltable-multirotors is scarcely reported, and thus one highlight in this work is the implementation on a real tiltable-quadrotor.

To address the second challenge, some studies [6], [16] have chosen to overlook the servo effect, which consequently reduces the control performance. Ryll et al. [5] implemented a Smith predictor to address the slow servo response. Although effective, this method is complicated since conventional feedback controllers lack the predictive property. In contrast, NMPC inherently incorporates prediction, offering an advantageous framework for systems with sluggish actuators. For example, the works [17], [18] leverage NMPC to control an aerial manipulator comprising both an agile multirotor and relatively slow carriers/winches. For tiltable multirotors, one approach to indirectly consider servomechanism within NMPC is to constrain the change rate of the resultant wrench, a method proposed by Brunner et al. [12]. While this technique helps manage the servo dynamics, it cannot accurately capture the entire scope of servo behavior. More recent studies [15], [19] integrate the actuator model more explicitly by using their derivatives as control inputs (so-called "delta-input formulation" [17]) and trying to constrain the range of these derivatives. Although the "Δu formulation" can describe various systems including first-order models, this generality might be unnecessary if the system is already known to be first-order. Furthermore, this way requires an additional integrator to accurately obtain the control command, and the actuators' measurement noise presents challenges for identifying their derivatives' boundaries in practice. In contrast, we model the servo as a first-order system, eliminating the need for an integrator and simplifying the identification process.

In this work, we introduce a unified nonlinear model predictive control approach for tiltable-quadrotors, as illustrated in 2b. This approach leverages allocation as an external reference, rather than a module within the control loop. Our method fully incorporates the nonlinear dynamics into the NMPC framework, directly utilizing rotor thrust and servo angle as control inputs. Furthermore, we explicitly integrate the servo as a first-order system within the model. We find that the inclusion of servo dynamics not only improves model fidelity but also facilitates optimization convergence.



(a) Workflow in [5], [6], [10]–[12]

(b) Our workflow.

Fig. 2. Comparative analysis of the previous and proposed workflows. Unlike the previous cascade structure, the proposed method directly integrates constraints and allocation within the NMPC optimization. Furthermore, servos are explicitly modeled. We directly use the servo angle and thrust as commands, requiring no extra integrator as in other research [15].

The main contributions of this article are:

1) We propose a servo-integrated NMPC framework for tiltable-quadrotors, which explicitly considers the servo as a first-order system. This framework has no simplification of the nonlinear robot model, resulting in the full exploration of the hardware potential.
2) We point out that due to the servo angle's nonlinearity and working range, considering the servo model is more critical than thrust for flight performance. In addition, modeling the servo is crucial for optimization convergence. These findings are verified in simulations.
3) We thoroughly evaluate the real-world control performance on various references with a self-developed tiltable-quadrotor, where the NMPC controller is running at 100 Hz on an onboard computer. To the best of the authors' knowledge, this is the first time an actuator-level NMPC is implemented on a real tiltable-quadrotor.

The remainder of the article is organized as follows. The modeling for the tiltable-quadrotor is introduced in Sec. II. The control approach is presented in Sec. III, followed by simulation analysis in Sec. IV. We then show the experimental results in Sec. V and finally the conclusion in Sec. VI.

II. Modeling

A. Coordinate Systems and Notation

We denote scalars in unbold $x, X \in \mathbb{R}$, vectors in bold lowercase $\boldsymbol{x} \in \mathbb{R}^n$, and matrices in bold uppercase $\boldsymbol{X} \in \mathbb{R}^{n \times m}$.

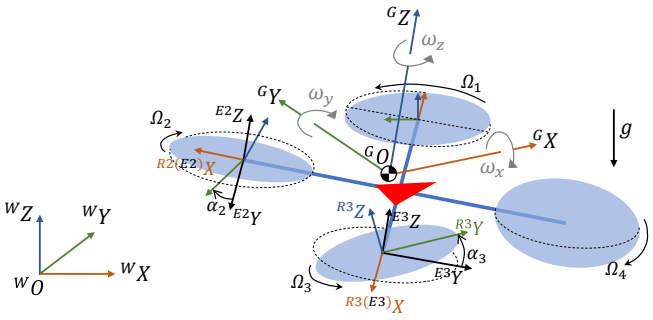


Fig. 3. Diagram of a tiltable-quadrotor with the ENU (X East, Y North, Z Up) inertial frame and the FLU (X Forward, Y Left, Z Up) body frame.

We use $\hat{\cdot}$ to denote estimated values. A vector in $\{\mathcal{W}\}$ can be denoted as ${}^{W}\boldsymbol{p}$, and the rotation from $\{\mathcal{G}\}$ to $\{\mathcal{W}\}$ is denoted as ${}^{W}_{G}\boldsymbol{R}$ (rotation matrix) or ${}^{W}_{G}\boldsymbol{q} = [q_w, q_x, q_y, q_z]^T$ (attitude quaternion).

Depicted in Fig. 3, the coordinate systems contain the world inertial frame $\{\mathcal{W}\}$, the body frame $\{\mathcal{G}\}$ whose origin is at the center of gravity (CoG), as well as the ith arm-end frame $\{\mathcal{E}_i\}$ and rotor frame $\{\mathcal{R}_i\}$ $(i = 1, 2, 3, 4)$. The origin of $\{\mathcal{E}_i\}$ is positioned at the extremity of the ith arm, where its X axis points outward, and its Z axis aligns parallel to the Z axis of $\{\mathcal{G}\}$. Frame $\{\mathcal{R}_i\}$ is derived from $\{\mathcal{E}_i\}$ through a rotation about the X axis, with the rotation angle denoted as α_i.

B. Tiltable-Quadrotor Model

The model for a tiltable-quadrotor has several parts. First, we establish the rotor model and get its resultant wrench. Then we introduce a first-order model to explicitly describe the servo dynamics. Finally, we apply the wrench to the rigid-body dynamics to obtain the full robot model.

1) Rotor Model: Motors rotate their propellers to generate thrust and torque. Here we neglect the angular acceleration of rotors and adopt commonly used quadratic fit to describe the motor dynamics

$$f_i = k_t\ \Omega_i^2, \quad \tau_i = k_q\ \Omega_i^2, \tag{1}$$

where Ω_i is the rotating speed of the ith propeller, k_t and k_q are coefficients identified from experiments. It is assumed that the motors are able to achieve the rotation speed Ω_i with negligible transients.

2) Resultant Wrench: To make the established model more general, we assume that N_p rotors are fixed on the robot, and we use d_i (whose value is ± 1) to denote the rotating direction of the ith motor and ${}^{G}\boldsymbol{p}_{r,i}$ to denote its position in $\{\mathcal{G}\}$.

If we use $\boldsymbol{f}_c = \left[f_1, \cdots, f_{N_p}\right]^T$ as input and try to express the thrust and torque in the rotor frame, the wrench becomes

$${}^{R_i}\boldsymbol{f}_i = [0, 0, f_i]^T, \quad f_i \in [f_{i,\min}, f_{i,\max}], \tag{2a}$$

$${}^{R_i}\boldsymbol{\tau}_i = \left[0, 0, -d_i\ f_i\ \frac{k_q}{k_t}\right]^T. \tag{2b}$$

Based on the coordinate systems, the tilt angle α_i influences the rotation from $\{\mathcal{R}_i\}$ to $\{\mathcal{E}_i\}$ along X axis, denoted as

$${}^{E_i}_{R_i}\boldsymbol{R} = \boldsymbol{R}_X(\alpha_i), \quad \alpha_i \in [\alpha_{i,\min}, \alpha_{i,\max}]. \tag{3}$$

In addition to the established torque, tilting a rotating propeller can generate an additional torque caused by gyroscopic effects. For aerial robots with small sizes, this effect is 2-3 orders of magnitude smaller [15] and can be treated as high-order terms to neglect. Then the resultant force and torque generated by propellers are derived as

$${}^{G}\boldsymbol{f}_u = \sum_{i=1}^{N_p} {}^{G}_{E_i}\boldsymbol{R}\ {}^{E_i}_{R_i}\boldsymbol{R}\ {}^{R_i}\boldsymbol{f}_i, \tag{4a}$$

$$\begin{aligned} {}^{G}\boldsymbol{\tau}_u = \sum_{i=1}^{N_p} \Big(& {}^{G}_{E_i}\boldsymbol{R}\ {}^{E_i}_{R_i}\boldsymbol{R}\ {}^{R_i}\boldsymbol{\tau}_i \\ & + {}^{G}\boldsymbol{p}_{r,i} \times {}^{G}_{E_i}\boldsymbol{R}\ {}^{E_i}_{R_i}\boldsymbol{R}\ {}^{R_i}\boldsymbol{f}_i \Big), \end{aligned} \tag{4b}$$

where ${}^{G}_{E_i}\boldsymbol{R}$ can be obtained from geometric properties.

3) Servo Model: The servo can be modeled with angle, angular velocity, or even torque as input. Considering most low-cost servos only support angle control, we hereby model the servo's motion as a first-order model

$$\dot{\boldsymbol{\alpha}} = \frac{1}{t_{\text{servo}}}(\boldsymbol{\alpha}_c - \boldsymbol{\alpha}), \tag{5}$$

where $\boldsymbol{\alpha} = \left[\alpha_1, \alpha_2, \ldots, \alpha_{N_p}\right]^T$ denotes the vector containing all servo angles, and $\boldsymbol{\alpha}_c = \left[\alpha_{c1}, \alpha_{c2}, \ldots, \alpha_{cN_p}\right]^T$ is the symbol for servo angle commands.

4) Rigid-Body Model: Neglecting the aerodynamic drag during flying, the motion of the robot is caused only by gravitational force and rotor wrench. Using position ${}^{W}\boldsymbol{p}$, velocity ${}^{W}\boldsymbol{v}$, quaternion ${}^{W}_{G}\boldsymbol{q}$, and angular velocity ${}^{G}\boldsymbol{\omega}$ (angular velocity of $\{\mathcal{G}\}$ w.r.t $\{\mathcal{W}\}$ and expressed in $\{\mathcal{G}\}$) as states, a six-DoF rigid body dynamics can be established as follows

$${}^{W}\dot{\boldsymbol{p}} = {}^{W}\boldsymbol{v}, \tag{6a}$$

$${}^{W}\dot{\boldsymbol{v}} = \left({}^{W}_{G}\boldsymbol{R}(\boldsymbol{q})\ {}^{G}\boldsymbol{f}_u + {}^{W}\boldsymbol{f}_d\right)/m + {}^{W}\boldsymbol{g}, \tag{6b}$$

$${}^{W}_{G}\dot{\boldsymbol{q}} = \frac{1}{2}\ {}^{W}_{G}\boldsymbol{q} \circ \mathcal{H}({}^{G}\boldsymbol{\omega}), \tag{6c}$$

$${}^{G}\dot{\boldsymbol{\omega}} = \boldsymbol{I}^{-1}\left(-{}^{G}\boldsymbol{\omega} \times \left(\boldsymbol{I}\ {}^{G}\boldsymbol{\omega}\right) + {}^{G}\boldsymbol{\tau}_u + {}^{G}\boldsymbol{\tau}_d\right), \tag{6d}$$

where $\boldsymbol{R}(\boldsymbol{q})$ denotes the rotation matrix converted from a quaternion, $\circ$ refers to quaternion multiplication, $\mathcal{H}(\cdot)$ means homogenizing a 3D vector $\mathcal{H}(\boldsymbol{p}) := [0, \boldsymbol{p}]^T$, ${}^{G}\boldsymbol{f}_u$ and ${}^{G}\boldsymbol{\tau}_u$ are expressed in (4a) and (4b), ${}^{W}\boldsymbol{f}_d$ and ${}^{G}\boldsymbol{\tau}_d$ are the force and torque caused by disturbances, as well as m, ${}^{W}\boldsymbol{g} = [0, 0, -g]^T$, and $\boldsymbol{I} = \texttt{diag}(I_{xx}, I_{yy}, I_{zz})$ are the mass, the gravity vector, and the inertia matrix, respectively.

The whole model established above is utilized for control and simulation. Note that this model is created for multirotors, and the tiltable-quadrotor is the special case when $N_p = 4$.

III. Control

Unlike some cascade control algorithms, our proposed NMPC controller leverages the fully nonlinear dynamics as the control model, emphasizing a concise and elegant "end-to-end" style. During this section, we first describe the generation of control reference in Sec. III-A. Then this target is converted into a cost function inside a finite-time optimal control problem (OCP), which is the core concept of NMPC in Sec. III-B. One drawback of NMPC is its sensitivity to model error, which leads to a steady-state error during flight, especially influenced by the ground effect in the Z axis. To solve this problem, a simple integral unit is introduced in Sec. III-C.

A. Generation of Control Reference

Although NMPC can work with only ${}^{W}\boldsymbol{p}_r$ and ${}^{W}_{G}\boldsymbol{q}_r$, the optimizer can converge faster by computing reference states as comprehensively as possible. This calculation is worthwhile since the computation cost is trivial. Thus, all our subsequent experiments are conducted with the full-state reference, and the generation method is introduced here.

Compared with normal quadrotors, tiltable-quadrotors are able to independently track three-dimensional attitudes at the

same time. Hence, we need both reference position ${}^{W}\boldsymbol{p}_r$ and attitude ${}^{W}_{G}\boldsymbol{q}_r$ as control targets. Another difference in control reference comes from the number, where NMPC requires a sequence of reference points as the control target, not just one point for traditional feedback controllers.

The control task can be divided into two types: tracking a constant point or tracking a trajectory. In our system, if the robot tracks a constant point at time t, then all reference points after t are set the same, and the reference velocity ${}^{W}\boldsymbol{v}_r$, acceleration ${}^{W}\dot{\boldsymbol{v}}_r$, angular velocity ${}^{G}\boldsymbol{\omega}_r$, and angular acceleration ${}^{G}\dot{\boldsymbol{\omega}}_r$ are set to zero. For tracking a trajectory, the reference points are calculated with a shifting prediction time interval, and ${}^{W}\boldsymbol{v}_r$, ${}^{W}\dot{\boldsymbol{v}}_r$, ${}^{G}\boldsymbol{\omega}_r$, and ${}^{G}\dot{\boldsymbol{\omega}}_r$ can be calculated by differentiating position and attitude. Then the desired ${}^{G}\boldsymbol{f}_{u,r}$ and ${}^{G}\boldsymbol{\tau}_{u,r}$ can be calculated from (6b) and (6d), where the gyroscopic term is omitted for simplicity.

On the basis of these states, we can obtain the reference thrust $f_{i,r}$ and servo angle $\alpha_{i,r}$ through control allocation. The relations between the reference wrench and a group of virtual input $\boldsymbol{z}$ can be expressed as

$$\left[{}^{G}\boldsymbol{f}_{u,r}, {}^{G}\boldsymbol{\tau}_{u,r}\right]^{T} = \boldsymbol{A}\ \boldsymbol{z}, \quad \text{where} \tag{7a}$$

$$\boldsymbol{z} = \begin{bmatrix} f_{1,r,h} \\ f_{1,r,v} \\ \vdots \\ f_{N_p,r,h} \\ f_{N_p,r,v} \end{bmatrix} = \begin{bmatrix} f_{1,r}\sin\alpha_{1,r} \\ f_{1,r}\cos\alpha_{1,r} \\ \vdots \\ f_{N_p,r}\sin\alpha_{N_p,r} \\ f_{N_p,r}\cos\alpha_{N_p,r} \end{bmatrix}, \tag{7b}$$

where the allocation matrix $\boldsymbol{A}$ can be derived from (4a) and (4b) using symbolic computation tools. Then we can calculate $f_{i,r}$ and $\alpha_{i,r}$ as follows

$$\boldsymbol{z} = \boldsymbol{A}^{\dagger}\ \left[{}^{G}\boldsymbol{f}_{u,r}, {}^{G}\boldsymbol{\tau}_{u,r}\right]^{T}, \tag{8a}$$

$$f_{i,r} = \sqrt{f_{i,r,h}^2 + f_{i,r,v}^2}, \tag{8b}$$

$$\alpha_{i,r} = \operatorname{atan2}\left(f_{i,r,h}, f_{i,r,v}\right), \tag{8c}$$

where $()^{\dagger}$ means the Moore–Penrose inverse operation. Note that $\boldsymbol{A}^{\dagger}$ is unchanged and only needs to be computed once.

As revealed by [13], NMPC has superiority in tracking dynamically infeasible trajectories. Hence, our requirement in this part is a middle-quality reference with a trivial computational burden, which has the potential for online replanning. The pose reference can be generated from parametric equations. If several pose points are given, the reference can be also obtained from the minimum-acceleration planning method [20] or other evolutionary versions.

B. Nonlinear Model Predictive Control

Nonlinear model predictive control converts the control problem into a constrained nonlinear optimization problem. For quadrotors, depending on the coordinate space, we can choose either a nonlinear model with linear constraints and a simple cost function, or a linear model with nonlinear constraints and a complex cost function. The former is easier to understand while the latter may have advantages in computational speed (please read [21] for discussions about normal quadrotors). Considering that the former one is more intuitive and has been deployed onboard successfully in many recent research [13], [14], we select the nonlinear model version.

We select the state as $\boldsymbol{x} = \left[{}^{W}\boldsymbol{p}, {}^{W}\boldsymbol{v}, {}^{W}_{G}\boldsymbol{q}, {}^{G}\boldsymbol{\omega}, \boldsymbol{\alpha}\right]^{T}$ and the control input as $\boldsymbol{u} = \left[\boldsymbol{f}_c, \boldsymbol{\alpha}_c\right]^{T}$. Given the reference $\boldsymbol{x}_r$, $\boldsymbol{u}_r$, we define the state error as $\overline{\boldsymbol{x}} = \left[\overline{\boldsymbol{p}}, \overline{\boldsymbol{v}}, \mathcal{V}(\boldsymbol{q}_e), \overline{\boldsymbol{\omega}}, \overline{\boldsymbol{\alpha}}\right]^{T}$ and the control input error as $\overline{\boldsymbol{u}} = \left[\overline{\boldsymbol{f}_c}, \boldsymbol{\alpha}_c - \boldsymbol{\alpha}\right]^{T}$, where the over-line symbol denotes $\overline{(\cdot)} = (\cdot) - (\cdot)_r$ if no special explanation, $\mathcal{V}(\cdot)$ represents the vector part of a quaternion $\mathcal{V}(\boldsymbol{q}) := [q_x, q_y, q_z]^{T}$, and $\boldsymbol{q}_e = \boldsymbol{q} \circ \boldsymbol{q}_r^{-1}$ represents the quaternion error. Note that the error term for the servo angle command is defined as $\overline{\boldsymbol{\alpha}}_c = \boldsymbol{\alpha}_c - \boldsymbol{\alpha}$ instead of the error w.r.t. reference $\overline{\boldsymbol{\alpha}}_c' = \boldsymbol{\alpha}_c - \boldsymbol{\alpha}_r$, aiming to penalize the right-hand of (5) to avoid the sudden change of servo commands.

Then the optimal control problem in NMPC can be formulated as a nonlinear least-square problem

$$\underset{\boldsymbol{x}_k, \boldsymbol{u}_k}{\text{minimize}} \quad \sum_{k=0}^{N-1} \left(\overline{\boldsymbol{x}}_k^{T}\boldsymbol{Q}\overline{\boldsymbol{x}}_k + \overline{\boldsymbol{u}}_k^{T}\boldsymbol{R}\overline{\boldsymbol{u}}_k\right) + \overline{\boldsymbol{x}}_N^{T}\boldsymbol{Q}_N\overline{\boldsymbol{x}}_N, \tag{9a}$$

$$\text{subject to} \quad \boldsymbol{x}_{k+1} = \boldsymbol{f}\left(\boldsymbol{x}_k, \boldsymbol{u}_k\right), \quad k = 0 : N-1, \tag{9b}$$

$$\boldsymbol{x}_0 = \hat{\boldsymbol{x}}, \tag{9c}$$

$$\|v_{x,y,z}\| \le v_{\text{limit}}, \quad \|\omega_{x,y,z}\| \le \omega_{\text{limit}}, \tag{9d}$$

$$\boldsymbol{u}_{\min} \le \boldsymbol{u}_k \le \boldsymbol{u}_{\max}, \tag{9e}$$

where $\boldsymbol{Q}$, $\boldsymbol{R}$, $\boldsymbol{Q}_N$ are positive diagonal matrices representing weights for state cost, control energy cost, and terminal cost, respectively. The (9b) indicates the dynamics constraint, where $\boldsymbol{f}(\cdot)$ refers to the full tiltable-quadrotor model established from (2) to (6). This model is discretized by the *fourth-order Runge–Kutta* method with t_{integ} integrating time. The (9c) denotes the initial value constraint, a critical component for feedback in NMPC, wherein $\hat{\boldsymbol{x}}$ represents the estimated state from the estimator. Finally, the (9d) and (9e) refer to the state and input constraints, respectively, which are mainly decided from safety concerns and physical limits.

After calculation, the first element of the optimized sequence $\boldsymbol{u}^*$ is transmitted to a low-level autopilot for execution:

$$\boldsymbol{u}_{\text{now}} = \boldsymbol{u}_0^*. \tag{10}$$

In implementation, we leverage several typical techniques in the NMPC community to accelerate computation. Specifically, *warm-starting*, *real-time iteration (RTI)*, and *multiple-shooting* are adopted. The *warm-starting* tries to accelerate by giving an initial guess near the final solution, where the guess comes from the last round's result. The RTI only calculates the sequential quadratic programming (SQP) once in one control iteration, preferring speed instead of optimality. Finally, the *multiple-shooting* divides the original OCP into several smaller optimization problems and tries to solve them in parallel. We recommend [22] to interested readers for more details.

C. Integral Term for Ground Effect

When rotor-based drones fly near the ground, the aerodynamic interaction between the drone's propellers and the environment can increase the lift, resulting in higher force in the Z axis. We empirically find that the model error caused by

this phenomenon is much larger than other axes, so we adopt an integral term as follows to compensate for the Z error

$$f_{d,z} = \mathrm{ITerm}\left({}^{W}\hat{p}_z - {}^{W}p_{z,r}\right). \tag{11}$$

The digital version of the integral term $u[k+1] = \mathrm{ITerm}(e[k+1])$ with trapezoidal rule and anti-windup is given as [23]

$$\begin{aligned} I'[k+1] &= I[k] + \frac{t_s}{2}\left(e[k] + e[k+1]\right), \\ u'[k+1] &= k_I\ I'[k+1], \\ u[k+1] &= \max\left(\min\left(u'[k+1], u_{\max}\right), u_{\min}\right), \\ I[k+1] &= I'[k+1] + \frac{1}{k_I}\left(u[k+1] - u'[k+1]\right). \end{aligned} \tag{12}$$

The compensating force ${}^{W}\boldsymbol{f}_d = [0, 0, f_{d,z}]^T$ is calculated before each NMPC round and then transmitted as parameters into the optimization process through (6b).

IV. Simulation

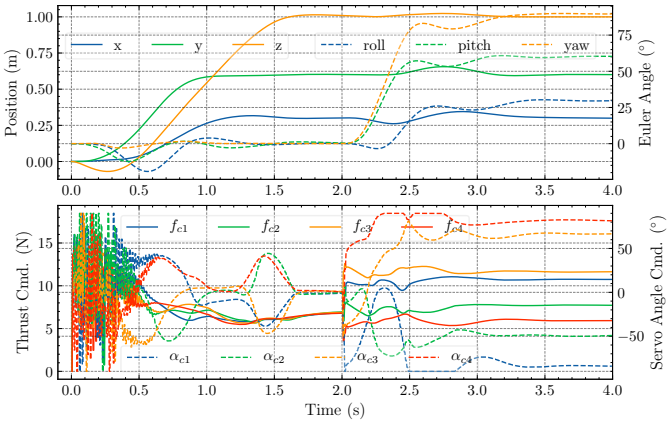
In this section, we perform simulations to reveal the impact of the servo model on the NMPC performance. To eliminate the influence of model error, disturbance, and other unknown effects, we implement an ideal simulation environment without model error and noise.

When simulating, the target is first to track a position of $\boldsymbol{p}_r = [0.3, 0.6, 1.0]^T$ from $t = 0$ s, then to track an attitude of $\boldsymbol{q}_r = \mathrm{RPY2Quat}\left(30°, 60°, 90°\right)^T$ from $t = 2$ s, where $\mathrm{RPY2Quat}(\cdot)$ denotes converting Euler angles to quaternions. These targets are given as step signals to represent the challenging dynamically infeasible cases. In the subsequent comparison, we choose the control frequency as 100 Hz and the simulation frequency as 200 Hz to align with the real-world experiments. To make the model more realistic, besides the servo-integrated nonlinear model (2)-(6), we also consider the thrust model as a first-order system similar to (5), where the time constant t_{thrust} is set from the identification result (Fig. 7b). The disturbances ${}^{W}\boldsymbol{f}_d$ and ${}^{G}\boldsymbol{\tau}_d$ are all set to zero, and other parameters are set the same as in Table I. The simulation results are depicted in Fig. 4.

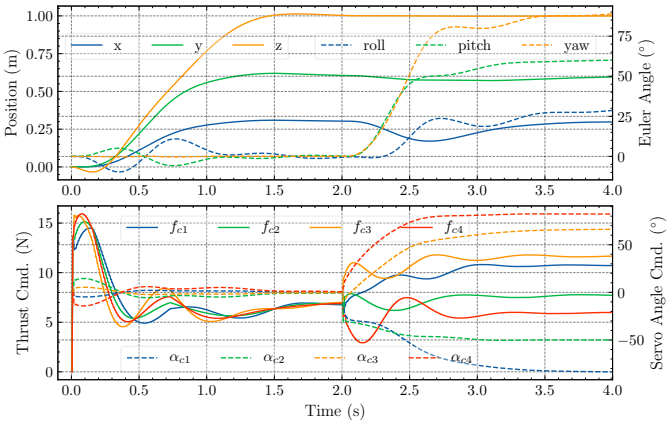
A. The Influence of Servo Model on Optimization Convergence

It is natural to think that adding a model increases the OCP complexity, resulting in higher difficulty in convergence. However, as revealed by Fig. 4a and Fig. 4b, the servo-integrated NMPC achieves less oscillation and faster optimization convergence. We believe this is due to the narrower search space introduced implicitly by the first-order servo model. Without this model, the next servo angle command can be any value within the physical limits. With this model, the command is limited in the nearby area of the real servo angle, and this extra limitation accelerates the convergence. Although the no-servo version achieves optimization convergence in the end, we report it diverging and crashing in noise-existing simulation, not mentioned in the real world. Hence, the inclusion of the servo model is a must for real-world usage.

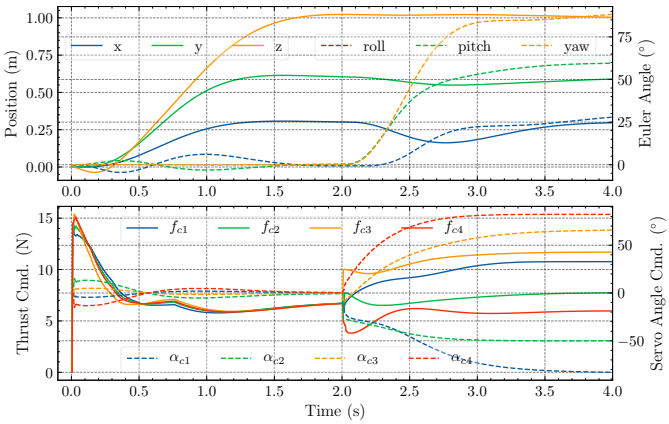
We also observe that increasing the servo angle's weight to a large number (above 50) can eliminate the oscillation. However, this way relies heavily on the accuracy of $\boldsymbol{\alpha}_r$ and leads to unacceptable error when tracking infeasible trajectories.



(a) Pose tracking for NMPC without servo and thrust model



(b) Pose tracking for NMPC with servo and without thrust model.



(c) Pose tracking for NMPC with servo and thrust model

Fig. 4. Comparative analysis of NMPC considering different models in an ideal simulation, where the control targets are described in the main text. The drastic command oscillation in (a) leads to optimization errors in noisy simulations such as Gazebo, and incorporating a servo model in (b) significantly mitigates this phenomenon. From (b) and (c), adding a thrust model merely results in a smoother tracking performance. The result of only considering the thrust is similar to (a).

B. The Influence of Servo Model on Flight Performance

On the basis of optimization convergence, considering the servo dynamics can reduce the fluctuations and hence increase the flight performance as revealed by Fig. 4a and Fig. 4b.

It is worth considering whether the thrust model should be incorporated when its time constant is comparable to that of the servo. Here we emphasize that, even if their

time constants are similar, the servo delay still has a more significant influence on the flight performance than thrust, and the reasons come from their different working ranges and servo angle's nonlinearity. During the mild flight, since the rotors of tiltable-quadrotors are always tilted upward, the relative working range of thrust is much smaller than the servo angle. This fact can be observed from Fig. 4b: after 2 s, the servo angle α_{c1} decreases from 0° to -80° while the thrust f_{c1} increases from 7 N to 11 N. If they have the same time constant and all left 20% to reach the target, the lagging angle and thrust are $(-80^\circ - 0^\circ) \times 20\% = -16^\circ$ and $(11-7)) \times 20\% = 0.8$ N, respectively. Mathematically, the wrench (4a) and (4b) are the sum of terms such as $c_x \sin(\alpha_i) f_i$ or $c_y \cos(\alpha_i) f_i$, where c denotes the constant term. Thus, a -16° lag causes $(\cos(-64^\circ) - \cos(-80^\circ))/\cos(-80^\circ) \times 100\% = 152.45\%$ relative error, while the 0.8 N only causes $(10.2-11)/11 \times 100\% = -7.27\%$ relative error. For other ranges where the cosine term is trivial, the sine term is important. The greater importance of servo angle than thrust is also verified in Fig. 4, where they have unequal influences on performance.

Although considering the thrust model can theoretically improve performance, focusing solely on the servo model is easier for implementation. Adding the thrust model to NMPC needs more computation, and also requires extra state-feedback of thrust, which can be calculated from rotor speed through (1). Many servos support angle measurement, but few ESCs can measure rotor speed at high frequencies. Therefore, we recommend only considering the servo model at the beginning. If oscillations are observed as in Fig. 4b, modeling the thrust should then be considered.

V. Experiments

A. Robot Platform

We made a tiltable-quadrotor as shown in Fig. 5 to verify the proposed approach, and the basic parameters are listed in Table I. The main modules and their communications are illustrated in Fig. 6, where the Robot Operating System (ROS) is used for communication. The NMPC controller is running in a Khadas VIM4 onboard computer, which has a 2.2 GHz Quad-Core ARM Cortex-A73 and a 2.0 GHz Quad-Core Cortex-A53 CPU. Then it sends control commands to the self-designed flight control unit "Spinal", which employs an STM32H7 series processor and has been employed in many

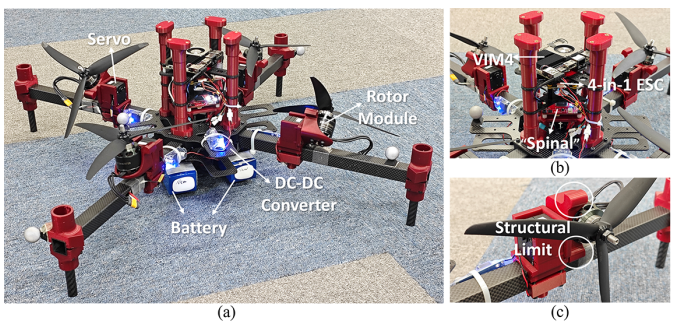


Fig. 5. (a) Snapshot of our self-build tiltable-quadrotor. (b) An onboard computer VIM4, a 4-in-1 electronic speed controller (ESC), and a self-designed flight control unit "Spinal" are centrally mounted on the robot's body. (c) The feasible servo angle is structurally limited to within $\pm\pi/2$.

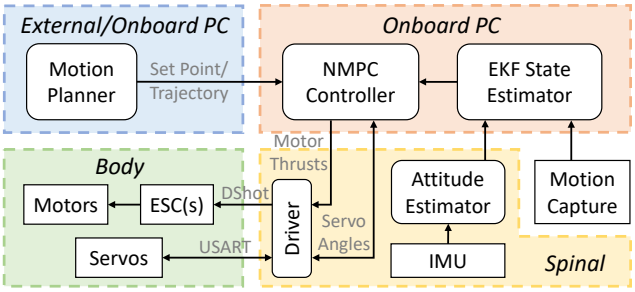


Fig. 6. Diagram of information flow among main modules within our system, where normal rectangles denote hardware modules and rounded rectangles indicate software modules.

previous works [8], [9]. Next, "Spinal" sends signals to control actuators. Specifically, we choose a T-Motor F55A PROII 6S 4IN1 ESC with DShot protocol to power four T-Motor AT2814 KV900 motors. Each motor is equipped with a 3-blade 9045 propeller. We select Kondo KRS-3302 as servos.

For state estimation, an IMU is installed on "Spinal", and its data are filtered in an attitude estimator to obtain roll and pitch angles as well as attitude velocity. In addition, we use an OptiTrack Motion Capture system to get position and attitude. These measurements are input to an EKF-based state estimator to calculate ${}^W\hat{\boldsymbol{p}}$, ${}^W\hat{\boldsymbol{v}}$, ${}^W_G\hat{\boldsymbol{q}}$, and ${}^G\hat{\boldsymbol{\omega}}$. The servo angle $\hat{\boldsymbol{\alpha}}$ is directly obtained by the Kondo servo without filtering. These estimated states are sent to NMPC for control.

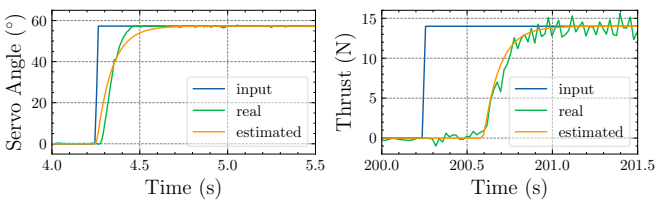
B. Parameter Identification

Our parameter identification experiments can be separated into three categories: geometry and inertial parameters, rotor parameters, as well as servo parameters.

The geometric parameters for the ith rotor ${}^G\boldsymbol{p}_{r,i}$ were directly obtained from the CAD model. The mass parameter was weighted using a weighing scale, and the inertial matrix was identified by the bifilar-pendulum method [24].

The rotor parameters k_t, k_q, $f_{i,\min}$, $f_{i,\max}$ as well as the mapping between command and thrust were identified by an ATI six-axis force/torque sensor fixed on a testbed. To make the result closer to reality, we 3D-printed a servo structure and mounted it near the rotor with 0° tilting. Compared to directly fixing the rotor on the force sensor, we observed a 19.53% thrust loss for the same command.

To identify the servo model, we removed the propellers for safety, commanded the motor to rotate at 60% throttle, and sent a step signal to stimulate the servo to 1 rad (57.30°). The data were collected using `rosbag`, and then the time constant of the servo model was identified by the MATLAB System



(a) Result for servo identification (b) Result for rotor identification

Fig. 7. For servos, the identified time constant is 0.0859 s, achieving an average accuracy of 93.4%. For rotors, the time constant is 0.0942 s with a 0.35-s dead time, where this large dead time is only observed when starting from zero speed. Thus, the time constants for servo and thrust can be similar.

TABLE I
MODEL & CONTROL PARAMETERS

Param.	Value	Param.	Value	Param.	Value
Wheelbase	0.4 m	m	2.773 kg	α_{limit}	$\pm\pi/2$
I_{xx}	0.0417	I_{yy}	0.0395	I_{zz}	0.0707 kg m^2
N_p	4	k_q/k_t	0.0153	t_{servo}	0.0859 s
N	20	t_{integ}	0.1 s	t_s	0.01 s
$Q_{p,\text{xy}}$	300	$Q_{p,\text{z}}$	400	$Q_{\text{v,xy}}$	10
$Q_{v,\text{z}}$	10	$Q_{q,\text{xy}}$	300	$Q_{\text{q,z}}$	600
$Q_{\omega,\text{xy}}$	5	$Q_{\omega,\text{z}}$	5	Q_α	2
R_f	2	R_α	250		
v_{limit}	$\pm$1 m/s	ω_{limit}	$\pm$6 rad/s	$\alpha_{i,\text{limit}}$	$\pm\pi/2$
$f_{i,\text{min}}$	0 N	$f_{i,\text{max}}$	30 N	$\alpha_{ci,\text{limit}}$	$\pm\pi/2$
$k_{I,z}$	5	$f_{d,\text{limit}}$	5 N		

Identification Toolbox, where Prediction Error Minimization is used as the evaluation method. The final result is the average of four servos, with one servo's result as shown in Fig. 7. Although this no propeller setting may introduce some errors, this accuracy is enough for real-world flight.

C. Tracking Test

The tool acados [25] is utilized to solve the NMPC problem, where we use *Partial Condensing HPIPM* as QP solver and *Explicit Runge-Kutta* as integrator. The control parameters are listed in Table I. We conducted three experiments to evaluate the proposed algorithm. Initially, the tiltable-quadrotor took off and then was poked by a stick to test disturbance rejection. Next, we carried out an experiment to track several pose points, aiming to evaluate the performance of physically infeasible references. Finally, the robot was controlled to track a 6-DoF pose trajectory with different speeds.

1) Takeoff, Anti-Disturbance, and Landing: Takeoff is the basis for all other experiments. During takeoff, the position command with the current X-Y position and $p_{r,z} = 0.6$ m was sent to the robot. After takeoff, the robot entered a hovering state and was then disturbed by a stick. After that, the robot landed. The data are shown in Fig. 8.

From Fig. 8, the robot can achieve the target height with a 14.83% overshoot, and then the integral term slowly corrects the Z position to the reference within 6 s. After giving disturbances to the robot, it can recover to the hovering state within 1 s, demonstrating the robustness of the proposed controller. After the disturbances ($t > 21$ s), the tilting angles of about 10° still exist, possibly implying that the optimizer has fallen into a local minimum. More analysis can be done in the future.

2) Set-Pose Tracking: The capability of tracking static points was tested, and the reference in each DoF can be seen as a step signal. Initially, the pose point $\boldsymbol{p}_r[\text{m}] = [0.3, 0.2, 1.2]^T$, $\boldsymbol{q}_r = \text{RPY2Quat}\,(0.5, 0.0, 0.3)^T$ was sent to the robot. After eight seconds, another pose point $\boldsymbol{p}_r[\text{m}] = [-0.3, 0.0, 1.0]^T$, $\boldsymbol{q}_r = \text{RPY2Quat}\,(0.5, 0.5, -0.3)^T$ was sent. After another eight seconds, the robot was commanded to the starting point. The result is plotted in Fig. 9. Note that we use radian in code but degree in figures for readability.

From Fig. 9, the robot can track the position and attitude changes simultaneously due to the non-simplification of the nonlinear robot model. The tracking RMSE for all directions

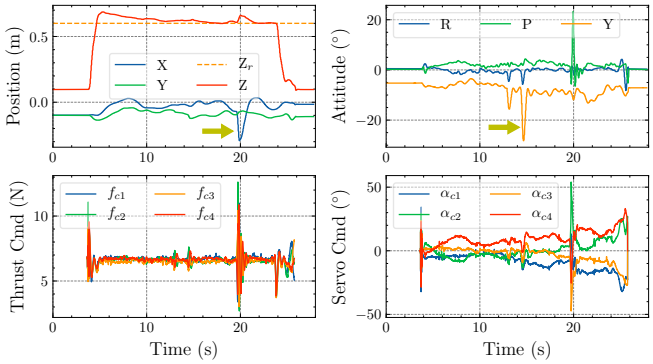


Fig. 8. Pose data from the takeoff, anti-disturbance, and landing experiment. A disturbance is given to yaw at about 14.5s (the yellow arrow in the left-upper figure), and then another disturbance is given to X at about 20s (the yellow arrow in the right-upper figure).

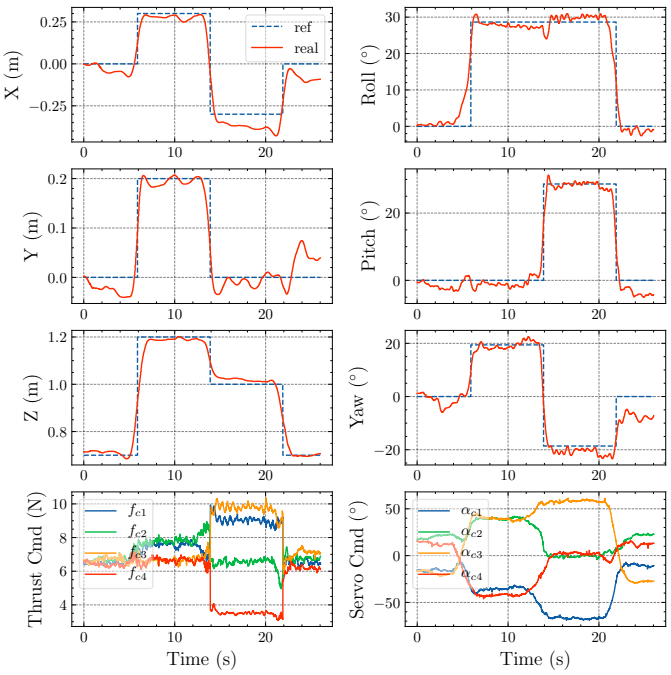


Fig. 9. Pose data from the set point tracking experiment. The robot tracks two pose points and finally goes back to the starting point.

are X: 0.072 m, Y: 0.029 m, Z: 0.044 m; Roll: 3.250°, Pitch: 2.810°, Yaw: 4.342°. Some oscillations in Y exist from 16 to 22 s, possibly due to aerodynamic disturbances from environments. In addition, the X position has an error of about 0.08 m from 14 to 22 s, indicating the existence of model error. Overall, we demonstrate the capability of setpoint tracking.

3) Pose Trajectory Tracking: Finally, a pose trajectory was sent to the robot. Let $\omega = 2\pi/T$, then the position $\boldsymbol{p}_r[\text{m}]$ was set as $p_x(t) = \cos(\omega t)$, $p_y(t) = \sin(2\omega t)/2$, $p_z(t) = 0.3\sin(2\omega t + \pi/2) + 1.0$, as well as the attitude $\boldsymbol{q}_r$ was set from Euler angles as $roll(t) = -\sin(2\omega t)/2$, $pitch(t) = 0.5\cos(\omega t)$, $yaw(t) = \pi/2 \cdot \sin(\omega t + \pi) + \pi/2$. We set $T = 20$ s (1x) and $T = 10$ s (2x) for one trajectory with different difficulties, where the tracking results are displayed in Fig. 10, and the real-world flight snapshot is presented in Fig. 1.

The tracking RMSE for 1x trajectory are X: 0.071 m, Y: 0.067 m, Z: 0.018 m; Roll: 4.934°, Pitch: 2.023°, Yaw: 2.789°. RMSE for 2x trajectory are X: 0.103 m, Y: 0.085 m, Z: 0.029

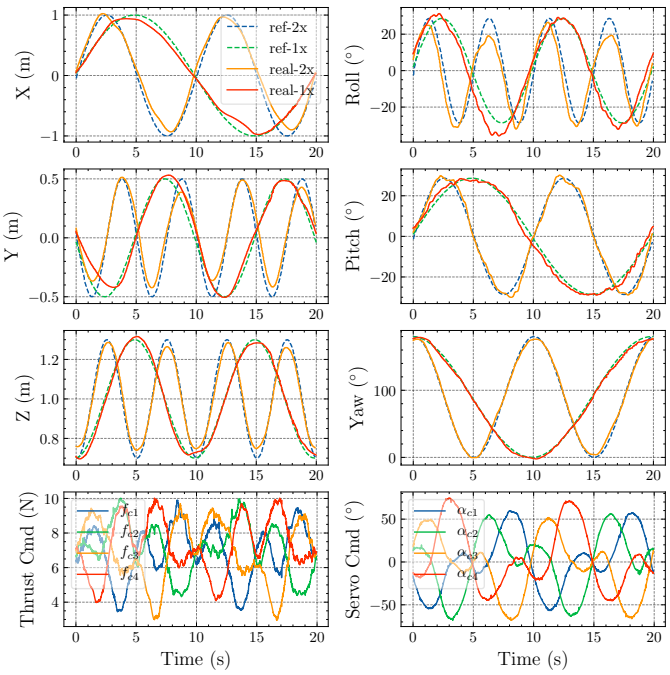


Fig. 10. Pose data for tracking one trajectory with different speeds, where 1x means using 20 s and 2x means using 10 s to finish. The plotted commands are for 2x trajectory. The actual servo angles follow their commands with a slight delay.

m; Roll: 6.740°, Pitch: 1.857°, Yaw: 3.622°, worse than 1x case. The tracking error in some quick turns (e.g., at 6.5 s for Roll-2x) may be due to model error or physical infeasibility, which can be resolved in the future. Overall, we demonstrate the feasibility of the proposed controller for trajectory tracking.

VI. Conclusion

In this article, we proposed an NMPC-based control framework for tiltable-quadrotors. Leveraging a fully nonlinear model with servo dynamics, the method directly generated rotor thrust and servo angle as control inputs. We found in the simulation that the inclusion of servo dynamics not only enhanced the control performance but also assisted in the optimization convergence. Finally, the algorithm was verified in the real world using a self-made robot.

In the future, we plan to implement a disturbance observer within the system to achieve offset-free tracking. Additionally, the proposed controller can be generalized to other aerial robots with tilting structures. We also prepare to extend our approach to handle physical interactions.

References

[1] D. Floreano and R. J. Wood, "Science, technology and the future of small autonomous drones," *Nature*, vol. 521, no. 7553, pp. 460–466, May 2015.

[2] A. Ollero, M. Tognon, A. Suarez, D. Lee, and A. Franchi, "Past, present, and future of aerial robotic manipulators," *IEEE Transactions on Robotics*, vol. 38, no. 1, pp. 626–645, Feb. 2022.

[3] D. Brescianini and R. D'Andrea, "Design, modeling and control of an omni-directional aerial vehicle," in *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, Stockholm, Sweden, May 2016, pp. 3261–3266.

[4] G. Flores, A. M. De Oca, and A. Flores, "Robust nonlinear control for the fully actuated hexa-rotor: Theory and experiments," *IEEE Control Systems Letters*, vol. 7, pp. 277–282, Jul. 2022.

[5] M. Ryll, H. H. Bulthoff, and P. R. Giordano, "A novel overactuated quadrotor unmanned aerial vehicle: Modeling, control, and experimental validation," *IEEE Transactions on Control Systems Technology*, vol. 23, no. 2, pp. 540–556, Mar. 2015.

[6] M. Kamel *et al.*, "The Voliro omniorientational hexacopter: An agile and maneuverable tiltable-rotor aerial vehicle," *IEEE Robotics & Automation Magazine*, vol. 25, no. 4, pp. 34–44, Dec. 2018.

[7] A. F. Şenkul and E. Altuğ, "System design of a novel tilt-roll rotor quadrotor UAV," *Journal of Intelligent & Robotic Systems*, vol. 84, no. 1-4, pp. 575–599, Dec. 2016.

[8] M. Zhao, T. Anzai, and T. Nishio, "Design, modeling, and control of a quadruped robot SPIDAR: Spherically vectorable and distributed rotors assisted air-ground quadruped robot," *IEEE Robotics and Automation Letters*, vol. 8, no. 7, pp. 3923–3930, Jul. 2023.

[9] T. Nishio, M. Zhao, K. Okada, and M. Inaba, "Design, control, and motion planning for a root-perching rotor-distributed manipulator," *IEEE Transactions on Robotics*, vol. 40, pp. 660–676, 2024.

[10] G. Scholz and G. F. Trommer, "Model based control of a quadrotor with tiltable rotors," *Gyroscopy and Navigation*, vol. 7, no. 1, pp. 72–81, Jan. 2016.

[11] M. Allenspach *et al.*, "Design and optimal control of a tiltrotor micro-aerial vehicle for efficient omnidirectional flight," *The International Journal of Robotics Research*, vol. 39, no. 10-11, pp. 1305–1325, Sep. 2020.

[12] M. Brunner *et al.*, "Trajectory tracking nonlinear model predictive control for an overactuated MAV," in *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, Paris, France, May 2020, pp. 5342–5348.

[13] S. Sun, A. Romero, P. Foehn, E. Kaufmann, and D. Scaramuzza, "A comparative study of nonlinear MPC and differential-flatness-based control for quadrotor agile flight," *IEEE Transactions on Robotics*, vol. 38, no. 6, pp. 3357–3373, Dec. 2022.

[14] J. Li, L. Han, H. Yu, Y. Lin, Q. Li, and Z. Ren, "Nonlinear MPC for quadrotors in close-proximity flight with neural network downwash prediction," in *Proceedings of the 62nd IEEE Conference on Decision and Control (CDC)*, Singapore, Singapore, Dec. 2023, pp. 2122–2128.

[15] D. Bicego, J. Mazzetto, R. Carli, M. Farina, and A. Franchi, "Nonlinear model predictive control with enhanced actuator model for multi-rotor aerial vehicles with generic designs," *Journal of Intelligent & Robotic Systems*, vol. 100, no. 3-4, pp. 1213–1247, Dec. 2020.

[16] D. Shawky, C. Yao, and K. Janschek, "Nonlinear model predictive control for trajectory tracking of a hexarotor with actively tiltable propellers," in *Proceedings of the 7th International Conference on Automation, Robotics and Applications (ICARA)*, Prague, Czech Republic, Feb. 2021, pp. 128–134.

[17] A. Yiğit, L. Cuvillon, M. A. Perozo, S. Durand, and J. Gangloff, "Dynamic control of a macro–mini aerial manipulator with elastic suspension," *IEEE Transactions on Robotics*, vol. 39, no. 6, pp. 4820–4836, Dec. 2023.

[18] L. Cuvillon, M. A. Perozo, A. Yiğit, S. Durand, and J. Gangloff, "Offset-free nonlinear model predictive control for improving dynamics of cable-driven parallel robots with on-board thrusters," *Mechanism and Machine Theory*, vol. 180, pp. 1–26, Feb. 2023.

[19] M. Brunner, W. Zhang, A. Roumie, M. Tognon, and R. Siegwart, "MPC with learned residual dynamics with application on omnidirectional MAVs," Jul. 2022, arXiv:2207.01451 [cs]. [Online]. Available: http://arxiv.org/abs/2207.01451

[20] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 2011, pp. 2520–2525.

[21] M. Greeff and A. P. Schoellig, "Flatness-based model predictive control for quadrotor trajectory tracking," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Madrid, Spain, 2018, pp. 6740–6745.

[22] S. Gros, M. Zanon, R. Quirynen, A. Bemporad, and M. Diehl, "From linear to nonlinear MPC: bridging the gap via the real-time iteration," *International Journal of Control*, vol. 93, no. 1, pp. 62–80, Jan. 2020.

[23] R. W. Beard and T. W. McLain, *Small Unmanned Aircraft: Theory and Practice*. Princeton, NJ, USA: Princeton University Press, Feb. 2012.

[24] M. R. Jardin and E. R. Mueller, "Optimized measurements of unmanned-air-vehicle mass moment of inertia with a bifilar pendulum," *Journal of Aircraft*, vol. 46, no. 3, pp. 763–775, May 2009.

[25] R. Verschueren *et al.*, "acados—A modular open-source framework for fast embedded optimal control," *Mathematical Programming Computation*, vol. 14, no. 1, pp. 147–183, Mar. 2022.