Efficient Trajectory Optimization for Generalized Multirotors via Sequential Convex Programming and Convexity Exploitation

Zhipeng Shen, *Member, IEEE*, Jinjie Li, *Graduate Student Member, IEEE*, Shiyu Zhou, *Graduate Student Member, IEEE*, Moju Zhao, *Member, IEEE*, and Hailong Huang, *Senior Member, IEEE*

***Abstract*—Aerial interaction requires aerial robots capable of independent attitude control, leading to the rising popularity of omnidirectional tiltable multirotors (tilt-multirotors). Unlike standard multirotors that rely on differential flatness to generate high-quality trajectories, tilt-multirotors possess servo-integrated nonlinear dynamics. Consequently, their trajectory optimization becomes a challenging nonconvex problem, generally suffering from high initialization sensitivity and computational costs. To address these challenges, we propose the penalized trust-region sequential convex programming with line search (PTR-SCP-LS) framework. Specifically, we introduce a line search update strategy based on a merit function to mitigate sensitivity to penalty weights. Furthermore, the framework allows us to flexibly exploit the underlying convexity of specific tasks to reduce computational burden and enhance tractability. By recognizing that standard multirotors are special cases of tilt-multirotors, we subsume both under the unified framework of *generalized multirotors*. Extensive simulations and experiments evaluate the framework's performance and generality. In a filming task with special Euclidean group SE(3) constraints using a tilt-quadrotor, with the servo-integrated nonlinear model, the line search strategy improves practical convergence robustness in the tested cases and achieves 2x faster solving times compared to nonconvex baselines with comparable solution accuracy. The proposed framework lays the foundation of the trajectory optimization for generalized multirotors, paving the way for versatile interaction tasks.**

***Note to Practitioners*—Interaction-oriented aerial tasks (e.g., manipulation and constrained filming) often require independent control of position and full attitude. Tiltable multirotors enable this by decoupling thrust direction from attitude, but the resulting servo-integrated dynamics make trajectory optimization highly nonlinear and nonconvex, with practical sensitivity to initialization and penalty-weight tuning. We present PTR-SCP-LS, a sequential convex programming framework for *generalized multirotors* (covering both standard and tiltable platforms) that plans with full six-degree-of-freedom (6-DoF) dynamics and actuator constraints. For practitioners, the key value is reliability and ease of deployment: a merit-function line search reduces sensitivity to penalty weights and improves practical convergence behavior, while task-dependent convexity exploitation (e.g., convex gate-passage and SE(3) constraints) preserves tractability by keeping key constraints in a convex form. Benchmarks and hardware experiments demonstrate fast, repeatable planning for both quadrotor racing and SE(3)-constrained tilt-quadrotor filming. Overall, the proposed framework serves as a practical foundation for deploying tilt-multirotors in a broader range of aerial tasks.**

***Index Terms*—Tiltable multirotors, trajectory optimization, sequential convex programming, convexity exploitation, aerial interaction.**

Received 24 March 2026; revised 30 June 2026; accepted 7 August 2026. Date of publication 11 August 2026; date of current version 19 August 2026. This article was recommended for publication by Associate Editor G. Silano and Editor A. M. Ghalamzan Esfahani upon evaluation of the reviewers' comments. *(Corresponding author: Hailong Huang.)*

Zhipeng Shen and Hailong Huang are with the Department of Aeronautical and Aviation Engineering, The Hong Kong Polytechnic University, Hong Kong, China (e-mail: zhi-peng.shen@polyu.edu.hk; hailong.huang@polyu.edu.hk).

Jinjie Li and Moju Zhao are with the DRAGON Laboratory, Department of Mechanical Engineering, The University of Tokyo, Tokyo 113-8654, Japan (e-mail: jinjie-li@dragon.t.u-tokyo.ac.jp; chou@dragon.t.u-tokyo.ac.jp).

Shiyu Zhou is with the Department of Mechanical Engineering, City University of Hong Kong, Hong Kong, China (e-mail: shiyuzhou9-c@my.cityu.edu.hk).

This article has supplementary downloadable material available at https://doi.org/10.1109/TASE.2026.3722603, provided by the authors.

Digital Object Identifier 10.1109/TASE.2026.3722603

I. Introduction

AERIAL robots have been applied in numerous tasks in recent years, ranging from vision-based tasks, which require independent position and yaw control, to increasingly popular interaction-based tasks [1], which further require independent roll and pitch control. Standard multirotors, like quadrotors, are widely adopted for the former tasks due to their mechanical simplicity, reliability, and well-established control frameworks. However, these platforms suffer from a fundamental limitation: the coupling between thrust direction and vehicle attitude. This underactuation restricts the ability to generate forces in arbitrary directions without altering orientation, thereby limiting maneuverability and operational flexibility during complex manipulation tasks [2].

To address this limitation, overactuated tiltable multirotors (tilt-multirotors) have emerged as a promising alternative [3], [4], [5]. By equipping each rotor with a servo-actuated tilt mechanism, these platforms decouple thrust direction from vehicle attitude, enabling omnidirectional flight. This capability makes tilt-multirotors ideal for interaction tasks requiring independent control of position and orientation [2], [6]. The design and optimal control of an omnidirectional hexarotor were presented in [3], and active interaction force control for omnidirectional hexarotors was investigated in [2]. More recent works have introduced advanced controllers, including nonlinear model predictive control (NMPC) [4] and adaptive control [5], to improve performance and robustness under

diverse operating conditions. Despite this progress in design and control, *trajectory optimization for tilt-multirotors remains underexplored*.

Tilt-multirotors strictly subsume standard platforms: fixing all tilt angles to zero recovers the robot model to a standard multirotor. Consequently, model-based planning or control methods developed for generalized multirotors apply directly to standard multirotors. We therefore unify standard and tilt-multirotors under the term ***generalized multirotors***.

Throughout the remainder of this paper, we adopt the following terminology: 1) *multirotors* refer to standard platforms; 2) *tilt-multirotors* refer specifically to tiltable platforms; and 3) *generalized multirotors* encompass both categories.

A. Related Works and Motivations

1) Trajectory Planning for Generalized Multirotors: Trajectory planning for multirotors can be broadly categorized into two paradigms: model-based methods [7], [8], [9], [10], [11] and learning-based methods [12], [13]. While both offer benefits, learning-based approaches typically require task-specific training, high-fidelity simulation, and large-scale data collection, and they do not always provide explicit constraint-satisfaction certificates. Therefore, this study focuses on model-based trajectory optimization for generalized multirotors. Recent model-based planning studies have extended multirotor trajectory generation to richer tasks, including autonomous aerial tracking [14], perception-aware collaborative tracking [15], dynamic planning for aerial vehicle groups in unknown environments [16], and cooperative planning [17]. These works demonstrate the continued importance of trajectory-generation methods under sensing, coordination, and environmental constraints. Complementary to these directions, this paper studies full-dynamics trajectory optimization for generalized multirotors, with an emphasis on handling heterogeneous actuation mechanisms within a unified formulation.

Differential flatness is widely used in model-based trajectory optimization for multirotors: the system states and inputs can be parameterized by flat outputs, enabling efficient generation of dynamically feasible trajectories [7], [8], [9]. With total thrust rigidly aligned to the body z-axis, a desired acceleration and yaw uniquely determine roll, pitch, and thrust magnitude. In the flat-output domain, the dynamics can be reduced to a linear system. Hence, differential flatness effectively exploits convexity by revealing an equivalent linear representation that supports polynomial parameterizations and convex optimization [18], [19]. In contrast, tilt-multirotors can decouple thrust direction from attitude via rotor tilting, so a given desired acceleration can admit many combinations of attitude, tilt angles, and thrust magnitudes. As a result, generalized multirotors are *not differentially flat* with respect to the standard flat outputs used for conventional multirotors (absent assumptions or simplifications that enforce fixed thrust–attitude alignment). Consequently, prior differential flatness-based approaches (e.g., [7], [8], [9]) are not directly applicable. Thus, it is necessary to handle the full nonlinear dynamics and related constraints of generalized multirotors.

Since the resulting trajectory optimization problem for generalized multirotors involves nonlinear systems and is thus nonconvex, one straightforward approach is to tackle this nonconvex program directly. For example, [11] solves a full-dynamics quadrotor formulation via nonconvex programming. Similarly, [20] plans trajectories for omnidirectional aerial manipulators by coupling offline (nonconvex) end-effector trajectory generation with an online NMPC scheme. While effective, such nonconvex programs are often computationally demanding. Thus, developing *efficient* trajectory optimization methods that operate under full tilt-multirotor dynamics remains a central open challenge.

2) Sequential Convex Programming: Convex approximations that exploit problem structure are typically more *efficient* and numerically robust. This strategy, via convexification and convexity exploitation, has achieved notable success in aerospace and robotics trajectory optimization [21]. Beyond conventional aerial vehicles, recent trajectory-planning studies have also addressed complex robotic systems such as floating-base multi-link robots maneuvering in confined environments [22], further highlighting the need for scalable optimization tools under coupled dynamics and geometric constraints.

To handle the nonlinear dynamics, we adopt successive convexification, which iteratively approximates the original nonconvex problem with a sequence of convex subproblems solved via sequential convex programming (SCP). SCP has proved effective for nonconvex trajectory optimization across applications [21], e.g., spacecraft rendezvous [23], aircraft landing [24], boost-glide ascent [25], and lunar reentry [26]. Convergence of SCP has been established under suitable regularity conditions [27], [28]. Moreover, recent results [29] show that penalized trust-region SCP (PTR-SCP) can handle nonconvex constraints while ensuring continuous-time constraint satisfaction and convergence guarantees. In aerial robotics, PTR-SCP has been applied to quadrotor trajectory optimization [30], [31] and, in a continuous-time constrained form, to general 6-degrees-of-freedom (6-DoF) rigid-body dynamics [32]. However, for tilt-multirotors, servo actuation dynamics are critical [4] and cannot be neglected. Incorporating servo dynamics further increases nonlinearity and can aggravate PTR-SCP *sensitivity to penalty weight selection*, hindering convergence (see Section V-C). Inspired by line search methods in nonlinear programming [28], [33], we introduce a merit-function-based line-search update within the PTR-SCP framework to mitigate this sensitivity.

3) Standard-Quadrotor Validation Under Racing: Given the absence of established benchmarks for tilt-multirotors, we use multirotor racing as a challenging, well-benchmarked standard-quadrotor case to test whether the proposed generalized-multirotor framework also applies to ordinary multirotors. This racing case is not intended to compare offline trajectory optimization against all recent racing pipelines, but rather to provide a demanding full-dynamics trajectory-optimization instance with common baselines. Multirotor racing has been extensively studied [11], [30], [34], [35], [36], [37]. More broadly, recent high-speed drone naviga-

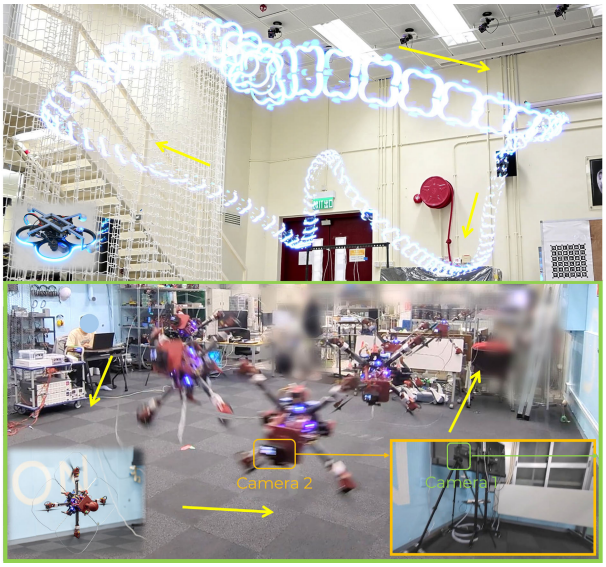


Fig. 1. Representative hardware demonstrations enabled by the proposed trajectory optimizer for generalized multirotors. The top row shows agile flight with a standard quadrotor, and the bottom row shows SE(3)-constrained flight with an overactuated tilt-quadrotor.

tion systems have demonstrated impressive safety-assured agile flight in cluttered environments [38]. Foehn et al. [11] presented a landmark study on quadrotor racing using nonconvex trajectory optimization to outperform top human pilots. However, the inherent nonconvexity of their formulation results in high computational costs and sensitivity to initial guesses [30]. While recent research has improved real-time performance by combining NMPC with simplified trajectory optimization [34], [35], [36], [37], e.g., using point-mass models [37], these approaches handle the nonconvexity in NMPC rather than the trajectory optimization stage. Variable-time-step model predictive control (MPC) further improves long-horizon agile multirotor planning by optimizing nonuniform prediction steps, but still relies on simplified point-mass or flat-output models rather than full generalized-multirotor dynamics with servo actuation [39]. As evidenced by Fig. 10 in [34] (see also related discussions), planning with simplified models can yield a large plan-track mismatch between the planned trajectory and the NMPC tracking result. In contrast, trajectory optimization with full nonlinear dynamics markedly reduces the plan-track gap, producing trajectories that are more faithful to the real platform.

Apart from the model-based methods, learning-based methods have demonstrated impressive capabilities in agile flight [40], [41], [42], [43], [44]. Although recent results suggest that reinforcement learning (RL) can outperform optimization-based pipelines in drone-racing tasks [42], this conclusion is drawn under a constraint design that does not explicitly enforce safe gate passage, i.e., waypoint-only constraints [11] rather than geometric gate-passage constraints. Recent benchmark results comparing learning-based and optimization-based quadrotor navigation further show that learning-based methods can be fast in inference and effective for high-speed flight, while optimization-based planners remain competitive in challenging cases such as sharp turns and occluded views [45].

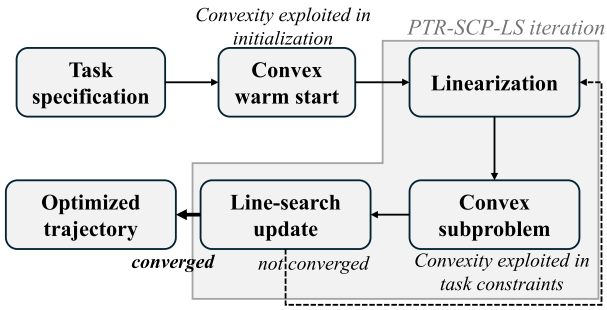


Fig. 2. Overall PTR-SCP-LS workflow. Convexity is exploited in the warm start and task-specific constraints, while full nonlinear dynamics are handled by successive convexification and rollout-mismatch line search.

While sufficient for simple tracks, waypoint-only constraints can lead to mission failure in complex environments (e.g., collisions when the vehicle does not fully pass through the gate plane). The same study observes that when optimization-based solvers successfully completed the racing without crashes, they can still surpass RL policies (see Fig. 2 in [42] and related discussions), suggesting substantial remaining potential for optimization methods.

Motivated by complex racing environments, we emphasize the need for reliable geometric gate-passage constraints to ensure safety. Within our framework, these constraints admit *convex* formulations, enabling seamless integration into convex optimization without relying on nonconvex constructs such as complementary progress constraints (CPCs) [11] or state-triggered constraints (STCs) [30].

B. Contributions

To summarize, our main contributions are:

1) **An SCP-based framework for generalized multirotors:** We present the first SCP trajectory optimization framework for generalized multirotors, capable of handling both standard and tilt-multirotors within a single mathematical structure. Unlike differential flatness-based works [7], [8], [9], [18], [19], our approach directly handles full nonlinear 6-DoF dynamics and complex actuator constraints, ensuring physical feasibility for diverse aerial platforms.
2) **A robust PTR-SCP-LS algorithm:** We propose a robust PTR-SCP algorithm integrated with a merit-function-based line search. This mechanism automatically adapts update step sizes, mitigating the notorious sensitivity of standard PTR methods [29], [30], [31], [32] to penalty weights (see Section V-C). Figure 2 summarizes the overall workflow.
3) **Convexity exploitation:** We show how to exploit the convex structure within the proposed framework to improve tractability. As a primary example, we introduce a geometric decomposition for drone racing that expresses gate-passage constraints using convex constructions similar to [46]. This representation certifies full gate traversal without introducing integer variables or intricate nonconvex constraints [9], [11], [30]. As another example, we enforce SE(3) constraints in a

convex form. Finally, we use simplified convex models to produce high-quality initial guesses for the SCP.

4) **Extensive experimental validation:** We evaluate the framework through simulations and real-world experiments on both standard quadrotors (high-speed racing) and tilt-quadrotors (SE(3)-constrained filming). The results demonstrate that our approach achieves superior computational efficiency compared to nonconvex solvers [11], [47], [48], while incurring small optimality gaps. Extensive experiments verify the repeatability and high success rates (see also the supplementary video).

This paper is organized as follows. Section II introduces the notation and dynamics model, and formulates the multiphase free-final-time trajectory optimization problem. Section III presents the proposed PTR-SCP-LS algorithm and clarifies the convergence guarantees inherited from the prox-linear approach and the practical role of the line-search update. Section IV explains how convexity is exploited in the warm start and task constraints. Section V details the implementation and benchmarking methodology, reporting experiment results. Finally, Section VI provides concluding remarks.

II. Problem Formulation

This section formulates the trajectory optimization problem for generalized multirotors. The notations used in this paper are introduced in Section II-A. The system model is introduced in Section II-B, and the problem is formulated in Section II-C.

A. Notations

For the definitions of the world frame $\mathcal{F}_W$, the body frame $\mathcal{F}_B$, and the rotor frame $\mathcal{F}_{Ri}$, please refer to [4]. The position of the multirotor is denoted as $\boldsymbol{p} \in \mathbb{R}^3$, expressed in $\mathcal{F}_W$. The velocity is denoted as $\boldsymbol{v} \in \mathbb{R}^3$, also expressed in $\mathcal{F}_W$. The orientation is represented by a quaternion $\boldsymbol{q}_B^W \in \mathbb{R}^4$, which describes the rotation from the robot's body frame $\mathcal{F}_B$ to the world frame $\mathcal{F}_W$. The angular velocity is denoted as $\boldsymbol{\omega} \in \mathbb{R}^3$, expressed in $\mathcal{F}_B$. $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_{N_r}]^\top \in \mathbb{R}^{N_r}$ represents the tilting angles of the rotors, where N_r is the number of rotors and α_i is the tilting angle of the i-th rotor. The thrust force generated by the rotors is denoted as $\boldsymbol{f}_t = [f_1, \ldots, f_{N_r}]^\top \in \mathbb{R}^{N_r}$, and $f_i \in \mathbb{R}$ is the thrust force generated by the i-th rotor. The thrust vector of the i-th rotor $\boldsymbol{f}_i = [0, 0, f_i]^\top$ is expressed in the rotor frame $\mathcal{F}_{Ri}$. The torque generated by the rotors is denoted as $\boldsymbol{\tau} = [\boldsymbol{\tau}_1, \ldots, \boldsymbol{\tau}_{N_r}]^\top \in \mathbb{R}^{N_r \times 3}$, where $\boldsymbol{\tau}_i \in \mathbb{R}^3$ is the torque generated by the i-th rotor. The total thrust force is denoted as $\boldsymbol{f}_B \in \mathbb{R}^3$, expressed in the body frame $\mathcal{F}_B$. The gravitational acceleration vector is denoted as $\boldsymbol{g} = [0, 0, -g]^\top$, where g is the gravitational acceleration. The inertia matrix is denoted as $J_B \in \mathbb{R}^{3\times3}$. The resultant torque is denoted as $\boldsymbol{\tau}_B \in \mathbb{R}^3$, expressed in $\mathcal{F}_B$. m is the mass of the multirotor. R_B^W is the rotation matrix derived from the quaternion $\boldsymbol{q}_B^W$. $\otimes$ denotes quaternion multiplication. $\mathcal{H}(\boldsymbol{\omega}_B) = [0, \boldsymbol{\omega}_B^\top]^\top$ is the quaternion representation of angular velocity. $\times$ denotes the cross-product operator. $\boldsymbol{r}_i$ is the position vector of the i-th rotor in $\mathcal{F}_B$. R_i is the rotation matrix from the rotor frame $\mathcal{F}_{Ri}$ to $\mathcal{F}_B$. $\boldsymbol{x} = [\boldsymbol{p}^\top, \boldsymbol{v}^\top, \boldsymbol{q}_B^{W\top}, \boldsymbol{\omega}_B{}^\top, \boldsymbol{\alpha}^\top]^\top \in \mathbb{R}^{n_x}$ is the state vector, and $\boldsymbol{u} = [\boldsymbol{f}_t{}^\top, \boldsymbol{\alpha}_c{}^\top]^\top \in \mathbb{R}^{n_u}$ is the control input vector, where n_x and n_u are the dimensions of the state vector and control input vector, respectively. $\boldsymbol{\alpha}_c = [\alpha_{c,1}, \ldots, \alpha_{c,N_r}]^\top$ is the commanded tilting angle vector, where $\alpha_{c,i}$ is the commanded tilting angle of the i-th rotor. τ_s is the time constant of the servo. $\mathcal{K}_i^j$ denotes the integer set $\{i, \ldots, j\}$, where i and j are integers and $i \le j$. $\|\cdot\|$ denotes the Euclidean norm. $\|\cdot\|_p$ denotes the p-norm. I is the identity matrix.

B. System Model

We introduce the dynamics model of generalized multirotors. The motion is governed by a 6-DoF rigid-body dynamic model. Neglecting the aerodynamic drag effects, the dynamics are driven by gravitational forces and the rotor-generated wrenches. The equations of motion are given as follows:

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

$$\dot{\boldsymbol{v}} = \frac{1}{m}\left(R_B^W \boldsymbol{f}_B\right) + \boldsymbol{g}, \tag{1b}$$

$$\dot{\boldsymbol{q}}_B^W = \frac{1}{2}\boldsymbol{q}_B^W \otimes \mathcal{H}(\boldsymbol{\omega}_B), \tag{1c}$$

$$\dot{\boldsymbol{\omega}}_B = J_B^{-1}\left(\boldsymbol{\tau}_B - \boldsymbol{\omega}_B \times (J_B\boldsymbol{\omega}_B)\right). \tag{1d}$$

This drag-free model is adopted to keep the core formulation focused on the rotor-generated wrench and servo dynamics. It is appropriate for the indoor motion-capture experiments considered in this paper, where the flight speeds and workspace are moderate and residual aerodynamic effects are handled by feedback tracking. For high-speed outdoor flights, drag can be included by augmenting the translational dynamics with a term $\boldsymbol{f}_{\text{drag}}(\boldsymbol{x})$ and linearizing this term together with the nominal dynamics inside the SCP loop. This extension does not change the overall formulation.

The total force $\boldsymbol{f}_B$ and torque $\boldsymbol{\tau}_B$ generated by the rotors are computed as:

$$\boldsymbol{f}_B = \sum_{i=1}^{N_r} R_i \boldsymbol{f}_i, \tag{2}$$

$$\boldsymbol{\tau}_B = \sum_{i=1}^{N_r} \left(\boldsymbol{r}_i \times R_i \boldsymbol{f}_i + R_i \boldsymbol{\tau}_i\right), \tag{3}$$

where R_i is the rotation matrix from $\mathcal{F}_{Ri}$ to $\mathcal{F}_B$ and is determined by the position vector $\boldsymbol{r}_i$ of the i-th rotor and the tilting angle α_i.

The servo can be modeled as a first-order system:

$$\dot{\alpha}_i = \frac{(\alpha_{ci} - \alpha_i)}{\tau_s}. \tag{4}$$

The equations (1)-(4) form the complete dynamic model for generalized multirotors, capturing both translational and rotational behaviors. In general, this formulation applies to any multirotor configuration. As a specific example, for a tilt-quadrotor (see Fig. 1), given $\boldsymbol{r}_i = [r_{i,x}, r_{i,y}, 0]^\top$, the rotation matrix is computed as:

$$R_i = \begin{bmatrix} \frac{r_{i,x}}{\sqrt{r_{i,x}^2 + r_{i,y}^2}} & -\frac{r_{i,y}}{\sqrt{r_{i,x}^2 + r_{i,y}^2}} & 0 \\ \frac{r_{i,y}}{\sqrt{r_{i,x}^2 + r_{i,y}^2}} & \frac{r_{i,x}}{\sqrt{r_{i,x}^2 + r_{i,y}^2}} & 0 \\ 0 & 0 & 1 \end{bmatrix} R_{t,i}, \tag{5}$$

where $R_{t,i}$ is the rotation matrix corresponding to the tilting angle α_i of the i-th rotor:

$$R_{t,i} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha_i) & -\sin(\alpha_i) \\ 0 & \sin(\alpha_i) & \cos(\alpha_i) \end{bmatrix}. \tag{6}$$

Remark 1: If all tilting angles are zero, (6) reduces to the identity matrix and the system degenerates to a standard multirotor. This highlights that, both mathematically and physically, a standard multirotor is a special case of a tilt-multirotor.

C. Trajectory Optimization Problem

In this subsection, we formulate the trajectory optimization problem for generalized multirotors. A multiphase free-final-time problem is formulated.

The dynamics formulated in Section II-B can be expressed as a nonlinear system:

$$\dot{\boldsymbol{x}}(t) = f(\boldsymbol{x}(t), \boldsymbol{u}(t)). \tag{7}$$

Using the state and input definitions in Section II-A, the vector field in (7) is explicitly

$$f(\boldsymbol{x}, \boldsymbol{u}) = \begin{bmatrix} \boldsymbol{v} \\ \frac{1}{m} R_B^W(\boldsymbol{q}_B^W) \boldsymbol{f}_B(\boldsymbol{\alpha}, \boldsymbol{f}_t) + \boldsymbol{g} \\ \frac{1}{2} \boldsymbol{q}_B^W \otimes \mathcal{H}(\boldsymbol{\omega}_B) \\ J_B^{-1} \left(\boldsymbol{\tau}_B(\boldsymbol{\alpha}, \boldsymbol{f}_t) - \boldsymbol{\omega}_B \times (J_B \boldsymbol{\omega}_B) \right) \\ (\boldsymbol{\alpha}_c - \boldsymbol{\alpha})/\tau_s \end{bmatrix}. \tag{8}$$

Here, $\boldsymbol{f}_B(\boldsymbol{\alpha}, \boldsymbol{f}_t)$ and $\boldsymbol{\tau}_B(\boldsymbol{\alpha}, \boldsymbol{f}_t)$ are computed from the rotor geometry by (2)–(6). This explicit form shows how the rigid-body dynamics and servo dynamics are combined into the nonlinear system $f(\boldsymbol{x}, \boldsymbol{u})$.

The multiphase free-final-time problem is formulated as:

$$\begin{aligned} \min_{\boldsymbol{x}(\cdot), \boldsymbol{u}(\cdot), \boldsymbol{T}} \quad & \sum_{k=1}^{K} \int_{t_{k-1}}^{t_k} L_k(\boldsymbol{x}(t), \boldsymbol{u}(t), t)\, dt \\ & + \phi(\boldsymbol{x}(t_K), \boldsymbol{T}), && \text{(9a)} \\ \text{subject to:} \quad & \dot{\boldsymbol{x}}(t) = f(\boldsymbol{x}(t), \boldsymbol{u}(t)), \quad t \in [t_0, t_K], && \text{(9b)} \\ & \boldsymbol{x}(t_0) = \boldsymbol{x}_0, && \text{(9c)} \\ & \boldsymbol{x}(t_k) \in \mathcal{X}_{k,f}, \quad \forall k \in \mathcal{K}_1^K, && \text{(9d)} \\ & \boldsymbol{x}(t) \in \mathcal{X}_k, \quad t \in [t_{k-1}, t_k], \quad \forall k \in \mathcal{K}_1^K, && \text{(9e)} \\ & \boldsymbol{u}(t) \in \mathcal{U}_k, \quad t \in [t_{k-1}, t_k], \quad \forall k \in \mathcal{K}_1^K, && \text{(9f)} \end{aligned}$$

where K is the number of phases, $L_k(\boldsymbol{x}(t), \boldsymbol{u}(t), t)$ is the running cost for phase k, and $\phi(\boldsymbol{x}(t_K), \boldsymbol{T})$ is the terminal cost. The initial time of the trajectory is denoted as t_0, and the final time is denoted as t_K. The time variable is denoted as $\boldsymbol{T} = [t_0, t_1, \ldots, t_K]$, where t_k is the terminal time of phase $\forall k \in \mathcal{K}_1^K$. The sets $\mathcal{X}_k$, $\mathcal{U}_k$, and $\mathcal{X}_{k,f}$ are nonempty compact convex sets describing the admissible state, control, and phase-terminal state constraints, respectively. The initial state is denoted as $\boldsymbol{x}_0$.

The problem (9) can be adapted to various applications by specifying the running cost $L_k(\boldsymbol{x}(t), \boldsymbol{u}(t), t)$, terminal cost $\phi(\boldsymbol{x}(t_K), \boldsymbol{T})$, convex state constraints $\mathcal{X}_k$, convex control constraints $\mathcal{U}_k$, and final convex state constraints $\mathcal{X}_{k,f}$.

For time-optimal trajectory optimization, the running cost and terminal cost can be defined as:

$$L_k(\boldsymbol{x}(t), \boldsymbol{u}(t), t) = 0, \quad \phi(\boldsymbol{x}(t_K), \boldsymbol{T}) = t_K. \tag{10}$$

For minimizing control effort, the running cost and terminal cost can be defined as:

$$L_k(\boldsymbol{x}(t), \boldsymbol{u}(t), t) = \|\boldsymbol{u}(t)\|^2, \quad \phi(\boldsymbol{x}(t_K), \boldsymbol{T}) = 0. \tag{11}$$

Remark 2: Many practical constraints can be formulated within $\mathcal{X}_k$, $\mathcal{X}_{k,f}$, and $\mathcal{U}_k$ as convex constraints. As will be detailed in Section IV, even complex geometric constraints under time-optimal objectives can be handled effectively via convex set constraints within this multiphase formulation. The framework can also accommodate nonconvex constraints and enforce continuous-time constraint satisfaction by augmenting the nonlinear dynamics (see, e.g., [29], [32]). However, to maintain focus, this paper emphasizes exploiting convexity in trajectory optimization for generalized multirotors and does not further address nonconvex state constraints or continuous-time constraint satisfaction.

III. Sequential Convex Programming

In this section, we present the PTR-SCP-LS algorithm. The algorithm is designed to solve the problem formulated in Section II-C. Section III-A details the convexification process of the original nonconvex problem into convex subproblems. Section III-B describes the proposed PTR-SCP-LS algorithm. Finally, Section III-C clarifies the convergence guarantees inherited from the standard prox-linear approach and the scope of the proposed line-search update.

A. Successive Convexification

To enable numerical optimization for free final-time problems, we perform time scaling for each phase. For phase k, the original time interval $[t_{k-1}, t_k]$ is scaled to $[0, 1]$ via

$$s_k = \frac{t - t_{k-1}}{\Delta t_k}, \quad t = t_{k-1} + s_k(\Delta t_k), \quad s_k \in [0, 1], \tag{12}$$

where $\Delta t_k = t_k - t_{k-1}$ is the duration of phase k. The scaled time variable s_k varies from 0 to 1 as time progresses from t_{k-1} to t_k. Thus, for phase k, the dynamics become

$$\mathring{\boldsymbol{x}} \triangleq \frac{d\boldsymbol{x}}{ds_k} = \Delta t_k f(\boldsymbol{x}(s_k), \boldsymbol{u}(s_k)). \tag{13}$$

To accurately evaluate the integral running cost, we augment the state with the running cost variable $l_k(s_k)$:

$$\frac{dl_k}{ds_k} = \Delta t_k L_k(\boldsymbol{x}(s_k), \boldsymbol{u}(s_k), s_k). \tag{14}$$

For brevity, we omit s_k throughout the remainder of this paper, e.g., $\boldsymbol{x}(s_k)$ is written as $\boldsymbol{x}_k$. The extended state for phase k is $\tilde{\boldsymbol{x}}_k = [\boldsymbol{x}_k^\top, l_k]^\top$. The extended dynamics are

$$\mathring{\tilde{\boldsymbol{x}}}_k \triangleq \frac{d\tilde{\boldsymbol{x}}_k}{ds_k} = \Delta t_k \tilde{f}(\tilde{\boldsymbol{x}}_k, \boldsymbol{u}_k), \tag{15}$$

$$\tilde{f}(\tilde{\boldsymbol{x}}_k, \boldsymbol{u}_k) \triangleq \begin{bmatrix} f(\boldsymbol{x}_k, \boldsymbol{u}_k) \\ L_k(\boldsymbol{x}_k, \boldsymbol{u}_k, s_k) \end{bmatrix}. \tag{16}$$

To convexify the dynamics, we linearize the system around a reference trajectory $(\bar{\tilde{\boldsymbol{x}}}_k, \bar{\boldsymbol{u}}_k, \bar{\boldsymbol{T}})$ for $k \in \mathcal{K}_1^K$. $\bar{\tilde{\boldsymbol{x}}}_k = [\bar{\boldsymbol{x}}^\top, \bar{l}_k]^\top$ is the reference state trajectory, $\bar{\boldsymbol{u}}_k$ is the reference control input trajectory, and $\bar{\boldsymbol{T}} = [\bar{t}_0, \bar{t}_1, \ldots, \bar{t}_K]^\top$ is the reference time vector. The linearized dynamics are given by

$$\mathring{\tilde{\boldsymbol{x}}}_k \approx A_k \tilde{\boldsymbol{x}}_k + B_k \boldsymbol{u}_k + \Delta t_k \tilde{f}(\bar{\tilde{\boldsymbol{x}}}_k, \bar{\boldsymbol{u}}_k) + \boldsymbol{r}_k, \tag{17}$$

where $A_k = \Delta \bar{t}_k \frac{\partial \tilde{f}}{\partial \tilde{\boldsymbol{x}}_k}\big|_{(\bar{\tilde{\boldsymbol{x}}}_k, \bar{\boldsymbol{u}}_k)}$, $B_k = \Delta \bar{t}_k \frac{\partial \tilde{f}}{\partial \boldsymbol{u}_k}\big|_{(\bar{\tilde{\boldsymbol{x}}}_k, \bar{\boldsymbol{u}}_k)}$, $\Delta \bar{t}_k = \bar{t}_k - \bar{t}_{k-1}$, and $\boldsymbol{r}_k = -A_k \bar{\tilde{\boldsymbol{x}}}_k - B_k \bar{\boldsymbol{u}}_k$.

To numerically solve the continuous-time problem, we discretize each scaled interval of phase k into N_k+1 uniformly spaced nodes $s_{k,0}, s_{k,1}, \ldots, s_{k,N_k}$, resulting in N_k intervals. The extended state at node j is denoted as $\tilde{\boldsymbol{x}}_{k,j} = [\boldsymbol{x}_{k,j}^\top, l_{k,j}]^\top$, where $\boldsymbol{x}_{k,j}$ is the state at node j in phase k, $\boldsymbol{u}_{k,j}$ is the control input at node j in phase k, and $l_{k,j}$ is the running cost at node j in phase k. For each segment $[s_{k,j}, s_{k,j+1}]$, $j \in \mathcal{K}_0^{N_k-1}$, we use first-order hold (FOH) on the control:

$$\boldsymbol{u}(s_k) = \frac{s_{k,j+1} - s_k}{s_{k,j+1} - s_{k,j}} \boldsymbol{u}_{k,j} + \frac{s_k - s_{k,j}}{s_{k,j+1} - s_{k,j}} \boldsymbol{u}_{k,j+1}, \quad s_k \in [s_{k,j}, s_{k,j+1}]. \tag{18}$$

The linearized dynamics over each interval can be discretized using the state transition matrix and input matrices computed along the reference trajectory. For each $j \in \mathcal{K}_0^{N_k-1}$, the discrete-time dynamics are given by:

$$\tilde{\boldsymbol{x}}_{k,j+1} = \Phi_{k,j} \tilde{\boldsymbol{x}}_{k,j} + \Gamma_{k,j}^0 \boldsymbol{u}_{k,j} + \Gamma_{k,j}^1 \boldsymbol{u}_{k,j+1} + \boldsymbol{d}_{k,j} \Delta t_k + \boldsymbol{r}_{k,j} + \boldsymbol{v}_{k,j}, \tag{19}$$

$$\Phi\left(s_k, s_{k,j}\right) \triangleq I + \int_{s_{k,j}}^{s_k} A_k(\tau) \Phi\left(\tau, s_{k,j}\right) \mathrm{d}\tau, \tag{20a}$$

$$\Gamma_{k,j}^0 \triangleq \int_{s_{k,j}}^{s_{k,j+1}} \Phi\left(s_{k,j+1}, \tau\right) \frac{s_{k,j+1} - \tau}{s_{k,j+1} - s_{k,j}} B_k(\tau) \mathrm{d}\tau, \tag{20b}$$

$$\Gamma_{k,j}^1 \triangleq \int_{s_{k,j}}^{s_{k,j+1}} \Phi\left(s_{k,j+1}, \tau\right) \frac{\tau - s_{k,j}}{s_{k,j+1} - s_{k,j}} B_k(\tau) \mathrm{d}\tau, \tag{20c}$$

$$\boldsymbol{d}_{k,j} \triangleq \int_{s_{k,j}}^{s_{k,j+1}} \Phi\left(s_{k,j+1}, \tau\right) \tilde{f}(\bar{\tilde{\boldsymbol{x}}}_k(\tau), \bar{\boldsymbol{u}}_k(\tau)) \mathrm{d}\tau, \tag{20d}$$

$$\boldsymbol{r}_{k,j} \triangleq \int_{s_{k,j}}^{s_{k,j+1}} \Phi\left(s_{k,j+1}, \tau\right) \boldsymbol{r}_k(\tau) \mathrm{d}\tau, \tag{20e}$$

where $\Phi(\cdot)$ is the state transition matrix. $\boldsymbol{v}_{k,j}$ is a virtual control term added to avoid infeasibility due to linearization.

To maintain the validity of the linearization, we introduce a trust region mechanism. At each node, the deviation from the reference trajectory is bounded by slack variables $\zeta_{\{x\}k,j}$, $\zeta_{\{u\}k,j}$, and $\zeta_{\{t\}k}$. The trust region is penalized in the cost:

$$J_{\mathrm{tr}} = \lambda_{\mathrm{tr}} \sum_{k=1}^{K} \sum_{j=1}^{N_k} \zeta_{\{x\}k,j} + \lambda_{\mathrm{tr}} \sum_{k=1}^{K} \sum_{j=0}^{N_k} \zeta_{\{u\}k,j} + \lambda_{\mathrm{tr}} \sum_{k=1}^{K} \zeta_{\{t\}k}, \tag{21}$$

subject to:

$$\|\tilde{\boldsymbol{x}}_{k,j} - \bar{\tilde{\boldsymbol{x}}}_{k,j}\|^2 \le \zeta_{\{x\}k,j}, \tag{22a}$$

$$\|\boldsymbol{u}_{k,j} - \bar{\boldsymbol{u}}_{k,j}\|^2 \le \zeta_{\{u\}k,j}, \tag{22b}$$

$$\|t_k - \bar{t}_k\|^2 \le \zeta_{\{t\}k}, \tag{22c}$$

where $\lambda_{\mathrm{tr}} > 0$ is the trust region weight.

The virtual control term $\boldsymbol{v}_j$ is penalized to discourage its use except when necessary to avoid infeasibility:

$$J_{\mathrm{vc}} = \lambda_{\mathrm{vc}} \sum_{k=1}^{K} \sum_{j=0}^{N_k-1} \xi_{k,j}, \tag{23}$$

subject to:

$$\|\boldsymbol{v}_{k,j}\| \le \xi_{k,j}, \tag{24}$$

where $\lambda_{\mathrm{vc}} \gg 1$ is a large penalty weight.

By augmenting the running cost into the system dynamics as shown in (16), the cost function of the problem reduces to a terminal cost, defined as:

$$J_{\mathrm{cost}} = \tilde{\phi}(\tilde{\boldsymbol{x}}_{K,N_K}, \boldsymbol{T}), \tag{25}$$

where the convex function $\tilde{\phi}(\cdot)$ denotes the terminal cost.

In summary, the final discretized and convexified problem is formulated as:

$$\min_{\substack{\{\tilde{\boldsymbol{x}}_{k,j}, \boldsymbol{u}_{k,j}, \boldsymbol{v}_{k,j}, \zeta_{\{x\}k,j}, \\ \zeta_{\{u\}k,j}, \zeta_{\{t\}k,j}, \xi_{k,j}, \boldsymbol{T}\}}} J_{\mathrm{cost}} + J_{\mathrm{tr}} + J_{\mathrm{vc}}, \tag{26a}$$

$$\text{subject to: } \tilde{\boldsymbol{x}}_{k,j+1} = \Phi_{k,j} \tilde{\boldsymbol{x}}_{k,j} + \Gamma_{k,j}^0 \boldsymbol{u}_{k,j} + \Gamma_{k,j}^1 \boldsymbol{u}_{k,j+1} + \boldsymbol{d}_{k,j} \Delta t_k + \boldsymbol{r}_{k,j} + \boldsymbol{v}_{k,j}, \quad \forall j \in \mathcal{K}_0^{N_k-1}, \quad \forall k \in \mathcal{K}_1^K, \tag{26b}$$

$$\|\tilde{\boldsymbol{x}}_{k,j} - \bar{\tilde{\boldsymbol{x}}}_{k,j}\|^2 \le \zeta_{\{x\}k,j}, \quad \forall j \in \mathcal{K}_1^{N_k}, \quad \forall k \in \mathcal{K}_1^K, \tag{26c}$$

$$\|\boldsymbol{u}_{k,j} - \bar{\boldsymbol{u}}_{k,j}\|^2 \le \zeta_{\{u\}k,j}, \quad \forall j \in \mathcal{K}_0^{N_k}, \quad \forall k \in \mathcal{K}_1^K, \tag{26d}$$

$$\|t_k - \bar{t}_k\|^2 \le \zeta_{\{t\}k}, \quad \forall k \in \mathcal{K}_0^K, \tag{26e}$$

$$\|\boldsymbol{v}_{k,j}\| \le \xi_{k,j}, \quad \forall j \in \mathcal{K}_0^{N_k-1}, \ \forall k \in \mathcal{K}_1^K, \tag{26f}$$

$$\boldsymbol{x}_{k,j} \in \mathcal{X}_k, \ \boldsymbol{u}_{k,j} \in \mathcal{U}_k, \ \forall j \in \mathcal{K}_0^{N_k}, \quad \forall k \in \mathcal{K}_1^K, \tag{26g}$$

$$\tilde{\boldsymbol{x}}_{k,0} = \tilde{\boldsymbol{x}}_{k-1,N_{k-1}}, \quad \forall k \in \mathcal{K}_2^K, \tag{26h}$$

$$\tilde{\boldsymbol{x}}_{1,0} = [\boldsymbol{x}_0^\top, 0]^\top, \tag{26i}$$

$$\boldsymbol{x}_{k,N_k} \in \mathcal{X}_{k,f}, \quad \forall k \in \mathcal{K}_1^K, \tag{26j}$$

$$0 < \Delta t_{k,\min} \le \Delta t_k \le \Delta t_{k,\max} < \infty. \tag{26k}$$

Equation (26) also clarifies how trajectory, waypoint, gate, and SE(3) coordination requirements enter Algorithm 1. They are encoded through the convex sets $\mathcal{X}_k$, $\mathcal{U}_k$, and $\mathcal{X}_{k,f}$ inherited from (9d)–(9f), rather than being added as separate heuristic coordination steps inside the algorithm. The running cost has already been accumulated into the extended state l_k, so the convex subproblem objective is the terminal cost J_{cost} in (26), augmented by the trust-region and virtual-control penalties that regularize the SCP iteration. $t_{k,\min}$ and $t_{k,\max}$ are the minimum and maximum allowed durations for phase k, respectively. The problem (26) is a convex optimization problem with linear dynamics and convex set constraints. When the reference is fully updated, i.e., without the line search, the iteration can be interpreted as a prox-linear step for a penalized finite-dimensional optimal-control problem and inherits the corresponding stationary-point guarantees under the assumptions summarized in Section III-C.

Algorithm 1 PTR-SCP-LS Algorithm

Require: Initial guess $(\bar{\boldsymbol{T}}, \bar{\tilde{\boldsymbol{x}}}, \bar{\boldsymbol{u}})$, max iterations $i_{\max}$, tolerances $(\epsilon_{\mathrm{tr}}, \epsilon_{\mathrm{vc}})$, weights λ_{tr}, λ_{vc}, line-search bracket $\underline{\delta} \in [0, 1)$, line-search tolerance $\epsilon_{\mathrm{ls}} > 0$

$i \leftarrow 0$
repeat
  Form convex subproblem (26) around $(\bar{\boldsymbol{T}}, \bar{\tilde{\boldsymbol{x}}}, \bar{\boldsymbol{u}})$
  Solve (26) to obtain $(\boldsymbol{T}^\star, \tilde{\boldsymbol{x}}^\star, \boldsymbol{u}^\star)$, slacks $\{\zeta_{k,j}\}$, virtual controls $\{\boldsymbol{v}_{k,j}\}$
  if (29) and (30) are satisfied **then**
    return $(\boldsymbol{T}^\star, \tilde{\boldsymbol{x}}^\star, \boldsymbol{u}^\star)$
  end if
  Line search (golden section) on $\delta \in [\underline{\delta}, 1]$:
    $a \leftarrow \underline{\delta},\ b \leftarrow 1,\ \tau \leftarrow (\sqrt{5}-1)/2$
    $c \leftarrow b - \tau(b-a),\ d \leftarrow a + \tau(b-a)$
  while $b - a > \epsilon_{\mathrm{ls}}$ **do**
    if $\mathcal{M}(c) \le \mathcal{M}(d)$ **then**
      $b \leftarrow d,\ d \leftarrow c,\ c \leftarrow b - \tau(b-a)$
    else
      $a \leftarrow c,\ c \leftarrow d,\ d \leftarrow a + \tau(b-a)$
    end if
  end while
  $\delta^{\mathrm{gs}} \leftarrow (a+b)/2$
  $\delta^\star \leftarrow *arg\,min_{\delta \in \{\delta^{\mathrm{gs}}, 1\}} \mathcal{M}(\delta)$
  Update reference:
  $(\bar{\boldsymbol{T}}, \bar{\tilde{\boldsymbol{x}}}, \bar{\boldsymbol{u}}) \leftarrow (1-\delta^\star)(\bar{\boldsymbol{T}}, \bar{\tilde{\boldsymbol{x}}}, \bar{\boldsymbol{u}}) + \delta^\star(\boldsymbol{T}^\star, \tilde{\boldsymbol{x}}^\star, \boldsymbol{u}^\star)$
  $i \leftarrow i + 1$
until $i \ge i_{\max}$
return $(\boldsymbol{T}^\star, \tilde{\boldsymbol{x}}^\star, \boldsymbol{u}^\star)$

B. PTR-SCP With Line Search

The PTR-SCP algorithm iteratively refines the trajectory by solving a sequence of convex subproblems, specifically the subproblem (26) at each iteration. It begins with an initial guess for the trajectory, which can be generated by a simple trajectory planner (see Section IV). The main steps of the PTR-SCP algorithm are as follows: The algorithm terminates when the following convergence criteria are satisfied:

$$C_{\mathrm{tr}} \triangleq \Big\| \{\zeta_{\{x\}k,j}\}_{k\in\mathcal{K}_1^K,\, j\in\mathcal{K}_1^{N_k}} \cup \{\zeta_{\{u\}k,j}\}_{k\in\mathcal{K}_1^K,\, j\in\mathcal{K}_0^{N_k}} \cup \{\zeta_{\{t\}k}\}_{k\in\mathcal{K}_1^K} \Big\|_\infty^2, \tag{27}$$

$$C_{\mathrm{vc}} \triangleq \Big\| \{\xi_{k,j}\}_{k\in\mathcal{K}_1^K,\, j\in\mathcal{K}_0^{N_k-1}} \Big\|_\infty, \tag{28}$$

$$C_{\mathrm{tr}} \le \epsilon_{\mathrm{tr}}, \tag{29}$$

$$C_{\mathrm{vc}} \le \epsilon_{\mathrm{vc}}, \tag{30}$$

where $\epsilon_{\mathrm{tr}} > 0$ and $\epsilon_{\mathrm{vc}} > 0$ are user-defined tolerances for the trust region and virtual control, respectively. The virtual-control criterion measures dynamic consistency of the linearized solution, while the trust-region criterion measures the size of the update relative to the current reference.

Although PTR-SCP approaches [24], [29], [31], [32] can be used to iteratively solve (26) to convergence, they suffer from a notable limitation: sensitivity to the weighting parameters in the cost function. Poorly chosen weights can lead to slow convergence or even nonconvergence (see Section V-C), and tuning them is often challenging and time-consuming.

To mitigate this issue, this paper adopts a line-search update strategy that improves robustness and reduces sensitivity to weight selection. After solving the convex subproblem at each iteration, the reference is updated partially using a step size $\delta \in (0, 1]$, rather than fully replacing it. This gradual update is used as a practical safeguarding mechanism: it promotes dynamic consistency of the updated reference, but by itself does not constitute a new global convergence proof beyond the prox-linear result discussed in Section III-C.

We choose δ by minimizing the following merit function:

$$\mathcal{M}(\delta) \triangleq \sum_{k=1}^{K} \sum_{j=1}^{N_k} \left\| \hat{\tilde{\boldsymbol{x}}}_{k,j}(\delta) - \tilde{\boldsymbol{x}}_{k,j}^{\mathrm{ref}}(\delta) \right\|_2^2, \tag{31a}$$

$$\hat{\tilde{\boldsymbol{x}}}_{k,j}(\delta) \triangleq \tilde{\boldsymbol{x}}_{k,j-1}^{\mathrm{ref}}(\delta) + \int_{s_{k,j-1}}^{s_{k,j}} \mathring{\tilde{\boldsymbol{x}}}\big(\tilde{\boldsymbol{x}}_k^{\mathrm{ref}}(\tau,\delta), \boldsymbol{u}_k^{\mathrm{ref}}(\tau,\delta), \boldsymbol{T}^{\mathrm{ref}}(\tau,\delta)\big)\, \mathrm{d}\tau, \tag{31b}$$

$$\tilde{\boldsymbol{x}}_{k,j}^{\mathrm{ref}}(\delta) \triangleq (1-\delta)\, \bar{\tilde{\boldsymbol{x}}}_{k,j} + \delta\, \tilde{\boldsymbol{x}}_{k,j}^\star, \tag{31c}$$

$$\boldsymbol{u}_{k,j}^{\mathrm{ref}}(\delta) \triangleq (1-\delta)\, \bar{\boldsymbol{u}}_{k,j} + \delta\, \boldsymbol{u}_{k,j}^\star, \tag{31d}$$

$$\boldsymbol{T}^{\mathrm{ref}}(\delta) \triangleq (1-\delta)\, \bar{\boldsymbol{T}} + \delta\, \boldsymbol{T}^\star, \tag{31e}$$

where $\tilde{\boldsymbol{x}}_{k,j}^\star$, $\boldsymbol{u}_{k,j}^\star$, and $\boldsymbol{T}^\star$ are the solutions of the convex subproblem, respectively. The vector field $\mathring{\tilde{\boldsymbol{x}}}(\cdot)$ is the extended dynamics in (15). Equation (31b) propagates the state under the partially updated reference, using the FOH control. The merit $\mathcal{M}(\delta)$ measures the mismatch between this open-loop rollout and the candidate solution. In the implementation, we compute a one-dimensional search candidate $\delta^{\mathrm{gs}} \in [\underline{\delta}, 1]$ and then keep $\delta = 1$ as an explicit safeguard candidate:

$$\delta^{\mathrm{gs}} \leftarrow \text{golden-section search on } [\underline{\delta}, 1], \tag{32}$$

$$\delta^\star = *arg\,min_{\delta \in \{\delta^{\mathrm{gs}}, 1\}} \mathcal{M}(\delta). \tag{33}$$

This endpoint safeguard guarantees $\mathcal{M}(\delta^\star) \le \mathcal{M}(1)$ regardless of whether the merit profile is globally unimodal. Thus, the line search is used to select a dynamically more consistent reference along the segment from the current reference to the SCP solution. In practice, this one-dimensional problem is solved efficiently via a golden-section search. The PTR-SCP-LS algorithm is summarized in Algorithm 1.

C. Convergence Guarantees and Scope

This subsection states the inherited convergence guarantee and its scope. The formal descent and stationarity results apply to the canonical *unrelaxed* core of Algorithm 1, where the dynamics residual is linearized, a convex subproblem is solved, and the reference is fully replaced by the subproblem solution. Equivalently, the line-search step is disabled or returns $\delta^\star = 1$. For $\delta^\star < 1$, the rollout-mismatch line search is treated as a practical safeguard rather than a new global convergence theorem. Its empirical effect is evaluated in Section V-C.

For a fixed discretization and FOH control parameterization, collect all nodal states, controls, and phase durations into

$$\boldsymbol{y} = \big(\tilde{\boldsymbol{x}}_{1,0}^\top, \boldsymbol{u}_{1,0}^\top, \ldots, \tilde{\boldsymbol{x}}_{1,N_1}^\top, \boldsymbol{u}_{1,N_1}^\top, \Delta t_1, \ldots, \tilde{\boldsymbol{x}}_{K,N_K}^\top, \boldsymbol{u}_{K,N_K}^\top, \Delta t_K\big). \tag{34}$$

Let $\mathcal{Y}$ denote the set of reference-independent explicit convex constraints on $\boldsymbol{y}$, including the state, control, phase-duration, boundary, terminal, and task-specific convex constraints, but excluding the nonlinear dynamics-residual equalities and the auxiliary virtual-control, trust-region, and epigraph variables used to represent the implemented convex subproblem. Let $\eta_{k,j}$ collect $(\tilde{\boldsymbol{x}}_{k,j}, \boldsymbol{u}_{k,j}, \boldsymbol{u}_{k,j+1}, \Delta t_k)$. For each segment, define the nonlinear one-step flow as

$$\Psi_{k,j}(\eta_{k,j}) \triangleq \tilde{\boldsymbol{x}}_{k,j} + \int_{s_{k,j}}^{s_{k,j+1}} \Delta t_k \, \tilde{f}(\tilde{\boldsymbol{x}}(\tau), \boldsymbol{u}_{k,j}^{\mathrm{FOH}}(\tau), \tau) \, \mathrm{d}\tau, \tag{35}$$

where $\tilde{\boldsymbol{x}}(s_{k,j}) = \tilde{\boldsymbol{x}}_{k,j}$ and $\boldsymbol{u}_{k,j}^{\mathrm{FOH}}(\tau)$ is the first-order-hold interpolation between $\boldsymbol{u}_{k,j}$ and $\boldsymbol{u}_{k,j+1}$ defined in (18). We define the nonlinear discrete-time dynamics residual as

$$\boldsymbol{g}_{k,j}(\boldsymbol{y}) \triangleq \tilde{\boldsymbol{x}}_{k,j+1} - \Psi_{k,j}(\eta_{k,j}), \tag{36}$$

for $k \in \mathcal{K}_1^K$ and $j \in \mathcal{K}_0^{N_k-1}$, and stack all residuals as

$$G(\boldsymbol{y}) = \mathrm{vec}\big(\{\boldsymbol{g}_{k,j}(\boldsymbol{y})\}_{k,j}\big). \tag{37}$$

Thus, $G(\boldsymbol{y}) = \boldsymbol{0}$ is equivalent to satisfaction of the nonlinear discrete-time dynamics.

The nonlinear residual-penalized finite-dimensional problem whose linearized model underlies (26) can be written in the convex-composite form

$$\Xi(\boldsymbol{y}) = J(\boldsymbol{y}) + H(G(\boldsymbol{y})), \tag{38}$$

where

$$J(\boldsymbol{y}) = \tilde{\phi}(\tilde{\boldsymbol{x}}_{K,N_K}, \boldsymbol{T}(\boldsymbol{y})) + I_{\mathcal{Y}}(\boldsymbol{y}), \tag{39}$$

$$H(\boldsymbol{z}) = \lambda_{\mathrm{vc}} \sum_{k=1}^{K} \sum_{j=0}^{N_k-1} \|\boldsymbol{z}_{k,j}\|_2. \tag{40}$$

Here, $I_{\mathcal{Y}}$ is the indicator function of $\mathcal{Y}$, and $\boldsymbol{T}(\boldsymbol{y})$ denotes the time vector reconstructed from the phase durations in $\boldsymbol{y}$ and the fixed initial time. Linearizing G at the current reference $\boldsymbol{y}_i$ and adding a quadratic proximal regularizer gives the canonical prox-linear subproblem

$$\boldsymbol{y}_{i+1}^{\mathrm{pl}} = \arg\min_{\boldsymbol{y}} \quad J(\boldsymbol{y}) + H\big(G(\boldsymbol{y}_i) + \nabla G(\boldsymbol{y}_i)(\boldsymbol{y} - \boldsymbol{y}_i)\big) + \frac{1}{2\rho}\|\boldsymbol{y} - \boldsymbol{y}_i\|_2^2. \tag{41}$$

After eliminating the auxiliary virtual-control and epigraph variables, (26) provides a conic representation of the linearized residual penalty

$$H\big(G(\boldsymbol{y}_i) + \nabla G(\boldsymbol{y}_i)(\boldsymbol{y} - \boldsymbol{y}_i)\big). \tag{42}$$

The soft trust-region penalty in (22) plays the same regularizing role as the proximal term in (41), although the implemented formulation uses separated norm penalties for state, control, and time rather than the single quadratic proximal term in the standard prox-linear subproblem.

The statements below use the compact state, control, terminal, and phase-duration constraints from Section II-C. We additionally restrict the accumulated-cost coordinate to a sufficiently large interval $\mathcal{L}_k = [\ell_{k,\min}, \ell_{k,\max}]$, set $\tilde{\mathcal{X}}_k = \mathcal{X}_k \times \mathcal{L}_k$, and consider References initialized in the resulting compact convex set $\mathcal{Y}$. Because the line-search update is a convex combination of points in $\mathcal{Y}$, accepted references remain in $\mathcal{Y}$. Thus, smoothness of the residual map is needed only on this compact operating set.

Proposition 1 (Convex-Composite Embedding of the Proposed Discretized Problem): Consider a fixed discretization and the first-order-hold control parameterization in (18). Suppose that the terminal cost $\tilde{\phi}$ is closed, proper, and convex, and that the explicit constraint sets $\mathcal{X}_k$, $\mathcal{U}_k$, and $\mathcal{X}_{k,f}$ are nonempty compact convex sets. Let the accumulated-cost coordinate be restricted to a compact interval $\mathcal{L}_k$, and define $\tilde{\mathcal{X}}_k = \mathcal{X}_k \times \mathcal{L}_k$. Suppose further that

$$0 < \Delta t_{k,\min} \le \Delta t_k \le \Delta t_{k,\max} < \infty, \tag{43}$$

and that, for each k, the scaled extended vector field

$$(\tilde{\boldsymbol{x}}, \boldsymbol{u}, \Delta t_k, s) \mapsto \Delta t_k \, \tilde{f}(\tilde{\boldsymbol{x}}, \boldsymbol{u}, s) \tag{44}$$

is twice continuously differentiable on an open neighborhood containing all one-step rollouts generated from

$$\tilde{\mathcal{X}}_k \times \mathcal{U}_k \times \mathcal{U}_k \times [\Delta t_{k,\min}, \Delta t_{k,\max}] \times [0, 1]. \tag{45}$$

Then the nonlinear residual-penalized finite-dimensional trajectory-optimization problem whose linearized model is used in (26) can be written in the convex-composite form (38). Moreover, $\mathcal{Y}$ is compact and convex, J is closed, proper, and convex, H is convex and Lipschitz continuous in finite dimensions, and the nonlinear dynamics-residual map G has a Lipschitz continuous Jacobian on $\mathcal{Y}$. The convex subproblem (26) is the corresponding conic PTR-SCP realization of this linearized residual-penalty model, with the weighted/separated proximal regularization described above.

Proof: The explicit set $\mathcal{Y}$ is obtained by intersecting finite products of compact convex state, control, terminal, cost-coordinate, and duration sets with affine boundary/phase-linking constraints and the convex task constraints in (26). Therefore, $\mathcal{Y}$ is compact and convex. Since $I_{\mathcal{Y}}$ is the indicator of a nonempty closed convex set, J is closed, proper, and convex under the stated assumptions on $\tilde{\phi}$. Likewise, H is a nonnegative weighted sum of Euclidean norms and is convex and Lipschitz. For $N_G = \sum_{k=1}^{K} N_k$,

$$|H(\boldsymbol{z}) - H(\boldsymbol{z}')| \le \lambda_{\mathrm{vc}} \sqrt{N_G} \, \|\boldsymbol{z} - \boldsymbol{z}'\|_2.$$

For each interval, FOH interpolation keeps the control in $\mathcal{U}_k$, and the duration lies in a compact positive interval. The assumed C^2 regularity of the scaled vector field on a neighborhood of the compact rollout set gives bounded second derivatives. By smooth dependence of ordinary differential equation (ODE) solutions on initial conditions and parameters, each one-step flow $\Psi_{k,j}$, and hence each residual $\boldsymbol{g}_{k,j}$, has a Lipschitz continuous Jacobian on $\mathcal{Y}$. Stacking finitely many residuals preserves this property for G. Linearizing G gives the dynamics residual used in (26), whose virtual-control and epigraph variables form a conic representation of the linearized residual penalty. This establishes the embedding. ∎

We use the following standard prox-linear descent result only for the canonical unrelaxed PTR-SCP core associated

with (38). The implemented norm-penalized trust region and line-search update are discussed separately.

Lemma 1 (Prox-Linear Descent; Lemma 5.1 of [49]): Assume that J is closed, proper, and convex, that H is α_Ξ-Lipschitz continuous, and that G has a β_Ξ-Lipschitz continuous Jacobian. Let $\boldsymbol{y}_{i+1}^{\mathrm{pl}}$ be the solution of (41), and define the prox-gradient mapping

$$\mathcal{G}_\rho(\boldsymbol{y}_i) \triangleq \frac{1}{\rho}\left(\boldsymbol{y}_i - \boldsymbol{y}_{i+1}^{\mathrm{pl}}\right). \tag{46}$$

If $0 < \rho < 2/(\alpha_\Xi \beta_\Xi)$, then

$$\Xi(\boldsymbol{y}_{i+1}^{\mathrm{pl}}) \le \Xi(\boldsymbol{y}_i) - \frac{\rho}{2}\left(2 - \rho\alpha_\Xi\beta_\Xi\right)\|\mathcal{G}_\rho(\boldsymbol{y}_i)\|_2^2. \tag{47}$$

In particular, the sufficient condition $\rho \le 1/(\alpha_\Xi\beta_\Xi)$ yields monotone decrease of Ξ for the unrelaxed prox-linear update.

Lemma 1 applies here through the convex-composite embedding above: G is the stacked dynamics residual, H is the ℓ_1/ℓ_2 residual penalty, and $I_{\mathcal{Y}}$ encodes the reference-independent convex constraints, as in the viewpoint of [29].

Corollary 1 (Finite-Iteration Stationarity of the Proposed Unrelaxed PTR-SCP Core): Under the assumptions of Proposition 1, the penalized objective Ξ is bounded below on $\mathcal{Y}$. Let $\Xi^\star$ denote such a lower bound. If Algorithm 1 is run in its unrelaxed prox-linear core form, i.e., the line-search update is disabled or returns $\delta^\star = 1$, and if $0 < \rho \le 1/(\alpha_\Xi\beta_\Xi)$, then for any $N \ge 1$,

$$\min_{0\le i<N} \|\mathcal{G}_\rho(\boldsymbol{y}_i)\|_2^2 \le \frac{2(\Xi(\boldsymbol{y}_0) - \Xi^\star)}{\rho N}. \tag{48}$$

Consequently, for any tolerance $\epsilon_\Xi > 0$, there exists an iterate $i \in \{0, \ldots, N-1\}$ satisfying

$$\|\mathcal{G}_\rho(\boldsymbol{y}_i)\|_2^2 \le \epsilon_\Xi \tag{49}$$

whenever

$$N \ge \left\lceil \frac{2(\Xi(\boldsymbol{y}_0) - \Xi^\star)}{\rho\epsilon_\Xi} \right\rceil. \tag{50}$$

Proof: For the unrelaxed core, $\boldsymbol{y}_{i+1} = \boldsymbol{y}_{i+1}^{\mathrm{pl}}$. Let $c_\rho = \frac{\rho}{2}(2 - \rho\alpha_\Xi\beta_\Xi)$, so $c_\rho \ge \rho/2 > 0$. Lemma 1 gives

$$c_\rho \sum_{i=0}^{N-1} \|\mathcal{G}_\rho(\boldsymbol{y}_i)\|_2^2 \le \Xi(\boldsymbol{y}_0) - \Xi(\boldsymbol{y}_N) \le \Xi(\boldsymbol{y}_0) - \Xi^\star.$$

Dividing by N and using $c_\rho \ge \rho/2$ yields (48). The iteration-complexity statement follows by requiring its right-hand side to be no larger than ϵ_Ξ. ∎

Corollary 1 is the formal stationarity guarantee for the unrelaxed PTR-SCP core. The bound on ρ is sufficient for the monotone-descent proof, not necessary for practical convergence. Existing prox-linear/PTR-SCP and exact-penalty SCP analyses further imply that, under appropriate regularity, a feasible stationary point of the penalized discretized problem corresponds to a Karush-Kuhn-Tucker (KKT) point of the control-parameterized problem [29]. The line-search update is handled below.

For the implemented PTR-SCP-LS update, let $\boldsymbol{y}_{i+1}^{\mathrm{scp}}$ denote the optimizer of the convex subproblem (26), and define

$$\boldsymbol{d}_i = \boldsymbol{y}_{i+1}^{\mathrm{scp}} - \boldsymbol{y}_i, \qquad \boldsymbol{y}_i(\delta) = \boldsymbol{y}_i + \delta\boldsymbol{d}_i. \tag{51}$$

Algorithm 1 chooses the next reference using the rollout-mismatch merit function $\mathcal{M}(\delta)$ in (31a). The following proposition states the precise safeguard property introduced by this line-search update.

Proposition 2 (Rollout-Mismatch Safeguard of PTR-SCP-LS): Suppose that Algorithm 1 retains the full step $\delta = 1$ as an admissible candidate and selects the accepted step by comparing the rollout-mismatch merit function $\mathcal{M}(\delta)$ in (31a). Then the accepted step satisfies

$$\mathcal{M}(\delta^\star) \le \mathcal{M}(1). \tag{52}$$

Proof: Because $\delta = 1$ is explicitly included in the candidate set used to select $\delta^\star$, the selected step cannot have a larger merit value than the full-step candidate. Hence, (52) follows directly. ∎

Proposition 2 is specific to the proposed PTR-SCP-LS update. It does not establish monotone decrease of Ξ for arbitrary partial-step sequences, nor does it require global optimality or unimodality of $\mathcal{M}(\delta)$. Overall, Proposition 1, Lemma 1, and Corollary 1 give the inherited prox-linear/PTR-SCP stationarity guarantee for the unrelaxed core, while Proposition 2 states the line-search safeguard: the accepted reference does not increase the nonlinear rollout mismatch relative to the full SCP candidate. The practical benefit is demonstrated in Section V-C.

IV. Convexity Exploitation

A. Convexity Exploitation With Flexibility

In this work, we exploit the latent convex structure of the problem in two complementary ways to ensure both computational efficiency and trajectory flexibility: the *initial guess* and the *problem formulation*.

1) Convex Initial Guess: In SCP, the initial guess is critical for both convergence speed and feasibility. A judicious choice reduces the risk of infeasibility and accelerates convergence. For many practical systems, one can exploit latent convex structure to construct a high-quality warm start, e.g., by optimizing simplified linear models.

Considering a point-mass model with a fixed final time, its dynamics admit a linear discrete-time representation. Combined with convex state and input constraints (e.g., constraints introduced in Section IV-B), the resulting problem is convex and can be solved efficiently to produce a reliable warm start. In this work, we use the point-mass model to initialize position and velocity. Attitude is linearly interpolated between the boundary values, while angular rates and control inputs are initialized as constants. This yields an initial trajectory that is close to feasibility. Moreover, because the initialization is obtained by solving a convex program, it is computationally inexpensive (see Section V-B).

Importantly, the SCP method does not require a dynamically feasible initial guess. The algorithm permits infeasible iterates and uses mechanisms such as virtual control terms (heavily penalized in the cost) and trust-region constraints to recover feasibility and guide convergence. Consequently, a simple convex warm start is often effective [31].

2) Convex Formulation via Multiphase Free Final Time: The second avenue for convexity exploitation is the problem formulation itself. We adopt a multiphase free-final-time formulation, which allows us to impose convex constraints over specific trajectory segments without solving a complex time-allocation problem. Since the duration of each phase is treated as an optimization variable, the solver automatically determines the optimal time allocation. This provides significant flexibility, allowing the trajectory to satisfy convex spatial or manifold constraints efficiently without the need for integer variables [9] or nonconvex constraints [11], [30].

We demonstrate the efficacy of this formulation through two specific examples: gate traversing (Section IV-B) and SE(3)-constrained flight (Section IV-C).

B. Example 1: Multi-Gate Racing

A significant advantage of the multiphase formulation is that it allows the imposition of constraints, such as waypoints or gates, at phase boundaries. Since phase durations are free, time-optimal flight through these constraints can be achieved directly by imposing convex set constraints, avoiding the additional optimization variables and nonconvex constraints (e.g., integer variables [9], CPCs [11], or STCs [30]) found in prior works. This approach reduces problem complexity and improves computational efficiency.

1) Waypoint Constraints: A waypoint constraint at a phase boundary can be formulated as:

$$\boldsymbol{x}(t_k) \in \mathcal{X}_{k,f}, \quad k \in \mathcal{K}_1^K, \tag{53}$$

where $\mathcal{X}_{k,f}$ is a convex set representing the desired state (e.g., position, velocity, orientation) at the k-th waypoint. For example, if the position at the waypoint is required to be within a ball of radius ϵ around $\boldsymbol{p}_k^*$, the constraint is:

$$\|\boldsymbol{p}(t_k) - \boldsymbol{p}_k^*\|_2 \le \epsilon. \tag{54}$$

2) Gate Constraints: In addition, the formulation enables us to guide the multirotor through flight corridors defined by gates (e.g., in racing scenarios) by introducing additional convex set constraints (see Fig. 3). Specifically, we define two convex cylindrical regions associated with each gate: one in front of the gate and one behind it.

Let $\boldsymbol{n}_k$ be the unit normal vector of the gate plane, and $\boldsymbol{p}_k^*$ be the center position of the gate at phase k. The projection operator onto the gate plane is defined as $A_n = I - \boldsymbol{n}_k \boldsymbol{n}_k^\top$. The cylinder in front of the gate is defined as:

$$\mathcal{G}_k^{\text{front}} = \Big\{ \boldsymbol{p} \in \mathbb{R}^3 \,\Big|\, d_1 \le (\boldsymbol{p} - \boldsymbol{p}_k^*) \cdot \boldsymbol{n}_k \le d_2, \; \|A_n(\boldsymbol{p} - \boldsymbol{p}_k^*)\|_2 \le r_k \Big\}, \tag{55}$$

and the cylinder behind the gate is defined as:

$$\mathcal{G}_k^{\text{back}} = \Big\{ \boldsymbol{p} \in \mathbb{R}^3 \,\Big|\, d_3 \le (\boldsymbol{p} - \boldsymbol{p}_k^*) \cdot \boldsymbol{n}_k \le d_4, \; \|A_n(\boldsymbol{p} - \boldsymbol{p}_k^*)\|_2 \le r_k \Big\}, \tag{56}$$

where r_k is the gate radius tolerance, and $d_1, \ldots, d_4$ define the longitudinal depth of the regions. To implement this, we enforce the convex set constraint on the discretization node closest to the phase boundary.

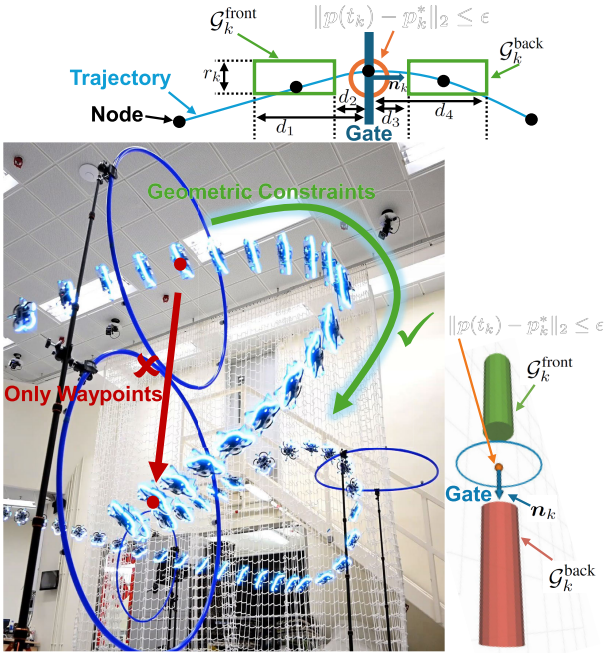


Fig. 3. Gate-passage constraint using convex sets. The multirotor must pass through two cylindrical regions (front and back) defined with respect to the gate plane, ensuring safe traversal. By contrast, waypoint-only constraints can yield trajectories that intersect the gate frame (e.g., with closely stacked gates), causing collisions. Free phase durations enable time-optimal allocation through these regions.

Remark 3: Compared with prior trajectory-optimization approaches for racing that enforce only waypoint constraints (e.g., [11], [30]), our multiphase free-final-time formulation imposes gate passage via convex sets. Waypoint-only constraints can be insufficient when gates are closely stacked (Fig. 3), leading to trajectories that intersect the gate plane and risk collisions. By modeling each gate as convex regions and constraining the phase boundary to lie within them, the optimizer enforces safe traversal while preserving flexibility: phase durations remain decision variables, enabling time-optimal allocation without additional nonconvex constructs.

Remark 4: Flight corridors defined by convex polyhedra or ellipsoids can be seamlessly integrated. Let $\mathcal{C}_k$ denote a convex set (polyhedron or ellipsoid) representing the k-th corridor. The constraint for phase k is imposed as $\boldsymbol{p}(t) \in \mathcal{C}_k$ for $t \in [t_{k-1}, t_k]$. At the phase boundary t_k, constraints from both adjacent regions are enforced simultaneously: $\boldsymbol{p}(t_k) \in \mathcal{C}_k \cap \mathcal{C}_{k+1}$. By treating phase durations as decision variables, the multirotor optimally allocates time within each corridor without resorting to mixed-integer programming (e.g., [9]).

C. Example 2: SE(3)-Constrained Flight

While Example 1 focuses on spatial constraints (position), many advanced maneuvers require precise control over the full rigid-body state. Our multiphase formulation naturally extends to SE(3) constraints, allowing us to specify not just the location but also the orientation of the multirotor at specific instances or intervals.

Analogous to the waypoint formulation, we can divide the trajectory into phases based on a sequence of desired poses. This is essential for tasks such as docking, passing through orientation-dependent gaps, filming, or grasping objects. The constraint at time t_k is formulated as:

$$\boldsymbol{x}(t_k) \in \mathcal{X}_k^{\text{pose}}, \tag{57}$$

where $\mathcal{X}_k^{\text{pose}}$ represents a convex region in the state space centered around a target position $\boldsymbol{p}_k^*$ and a target attitude $\boldsymbol{q}_k^*$. For example, this can be realized by enforcing:

$$\|\boldsymbol{p}(t_k) - \boldsymbol{p}_k^*\|_2 \leq \epsilon_p, \tag{58}$$

$$\|\boldsymbol{q}_B^W(t_k) - \boldsymbol{q}_k^*\|_2 \leq \epsilon_q, \tag{59}$$

where ϵ_p and ϵ_θ are tolerances for position and orientation, respectively. By treating the phase duration as a variable, the optimizer can adjust the flight time to ensure the vehicle has sufficient time to meet these boundary conditions dynamically.

Remark 5: Although exploiting convexity within a multi-phase free-final-time framework is a natural design choice, it is frequently overlooked in the literature in favor of mixed-integer formulations [9] or complex nonconvex constraints [11]. We emphasize that leveraging these convex substructures is critical for improving computational efficiency and tractability. As will be demonstrated in Section V-B, this strategy yields significant performance gains: even when employing a general-purpose nonconvex solver (for the nonlinear dynamics), the proposed formulation substantially reduces both the mean and standard deviation of the computation time.

V. Experiments

We evaluate the proposed framework through two representative applications chosen to test its applicability to both standard and tiltable multirotor platforms.

First, we use aggressive quadrotor racing as a challenging and well-benchmarked standard-multirotor case. The purpose of this case is to show that the proposed generalized-multirotor trajectory optimizer also applies to ordinary quadrotors and can exploit convex task structure in a demanding setting. Second, we address SE(3)-constrained filming flight using a tilt-quadrotor, where convergence behavior, model fidelity, and solution quality are analyzed. This application tests the framework's ability to handle complex, over-actuated dynamics and strict orientation requirements that standard quadrotors cannot fulfill. Finally, real-world experiments are presented for both scenarios to validate the physical feasibility and repeatability.

A. Experimental Settings

The proposed PTR-SCP-LS algorithm is implemented in Python. Numba [50] is used to just-in-time compile performance-critical routines, such as integrating the state transition matrices in (20). Each convex subproblem is modeled using CVXPY [51] and solved with the Mosek[1] solver. To reduce repeated solve overhead, the disciplined parametrized programming ruleset in CVXPY is leveraged: after the first solve, the parameter-to-problem-data mapping is cached, making subsequent canonicalization and solves significantly faster. This is especially beneficial when the same parametrized convex problem must be solved many times, yielding substantial savings in rewrite and setup time.

[1]https://www.mosek.com/

TABLE I
Computational Time Statistics for Quadrotor Racing (50 Trials)

Method	Max [s]	Min [s]	Median [s]	Mean [s]	Std Dev [s]
Initial Guess	0.0062	0.0048	0.0050	0.0051	0.0003
Proposed	7.63	**0.65**	**1.47**	**2.12**	1.77
CasADi+Ipopt [47], [48]	**7.60**	2.09	3.21	3.51	**1.11**
CPC [11]	31.37	4.28	7.99	9.10	4.98

All computational time results and convergence analysis of trajectory optimization are obtained on a laptop computer equipped with an Intel Core i7-13650HX CPU and 16 GB of RAM. For hardware tests with quadrotors, the PX4 firmware is utilized with an NMPC controller computing body angular rates and collective thrust commands. For real-world experiments with tilt-quadrotors, NMPC is employed for trajectory tracking [4]. State estimation in both experiments is provided by motion-capture systems.

For quadrotor racing, we additionally include the CPC method proposed by [11] as a benchmark, which has been used successfully in prior drone racing studies [34], [37], [42]. Unlike directly solving the nonconvex problem here with CasADi+Ipopt, it enforces only waypoint constraints.

For trajectory optimization for tilt-multirotors, the benchmark solves the original nonconvex problem using a fourth-order Runge-Kutta discretization in CasADi [48] with Ipopt [47]. We refer to this benchmark as "CasADi+Ipopt" hereafter.

The benchmark set is chosen to keep the comparison within comparable trajectory-optimization problem classes. CasADi+Ipopt solves the corresponding nonconvex optimal-control problem with the same full dynamics or task constraints, while CPC is a widely used racing benchmark that exposes the difference between waypoint-only and geometric gate-passage formulations. It is also used as a benchmark in recent model predictive contouring control (MPCC) and RL racing studies [34], [37], [42], which makes it a meaningful common reference for the racing example. Recent MPCC, NMPC, and RL racing pipelines are discussed in Section I because they are highly relevant and each has distinct advantages and limitations. However, these works are not the same offline full-dynamics trajectory-optimization problem studied here: they combine different closed-loop control structures, simplified planning models, perception/training pipelines, or constraint semantics. Directly timing them against the present optimizer would therefore conflate solver efficiency with controller design, training cost, and tracking implementation. Thus, the racing experiment is used mainly as a standard-quadrotor stress test and generality check, rather than as a claim that the proposed trajectory optimizer dominates all racing pipelines.

Unless otherwise specified, we use $\lambda_{\text{vc}} = 100$, $\epsilon_{\text{vc}} = 10^{-3}$, $i_{\max} = 500$, $\underline{\delta} = 0$, $\epsilon_{\text{ls}} = 0.1$, $\lambda_{\text{tr}} = 0.3$, and $\epsilon_{\text{tr}} = 0.5$.

TABLE II

PROBLEM INSTANCES, OBJECTIVES, CONSTRAINTS, AND COMPARISON SCOPE USED IN THE EXPERIMENTS

Instance	Objective	Key constraints	Comparison scope
Quadrotor racing	Time-optimal trajectory: $L_k = 0,\ \phi = t_K$.	Full rigid-body dynamics; thrust bounds; free phase durations with lower/upper bounds; initial/final boundary conditions; geometric gate-passage constraints.	CasADi+Ipopt uses the same full-dynamics model and geometric gate-passage formulation to compare solver efficiency and solution quality. A waypoint-only ablation isolates the effect of the proposed gate-passage constraints. CPC [11] is included as a common waypoint-style racing benchmark.
SE(3)-constrained filming	Control-effort minimization: $L_k = \|\boldsymbol{u}\|^2,\ \phi = 0$.	Servo-integrated tilt-quadrotor dynamics; thrust and tilt-angle bounds; free phase durations with lower/upper bounds; prescribed position-and-attitude constraints; initial/final boundary conditions.	CasADi+Ipopt uses the same servo-integrated nonlinear model, objective, and boundary/pose constraints to compare solve time, objective value, and rollout consistency. The no-line-search ablation uses the same PTR-SCP formulation with a full-step update to isolate the practical benefit of the line-search safeguard.

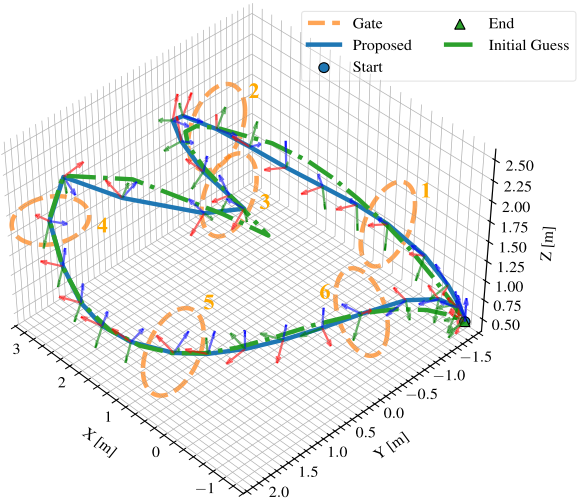


Fig. 4. Optimized racing trajectory using the proposed framework. All gate constraints are satisfied while minimizing flight time.

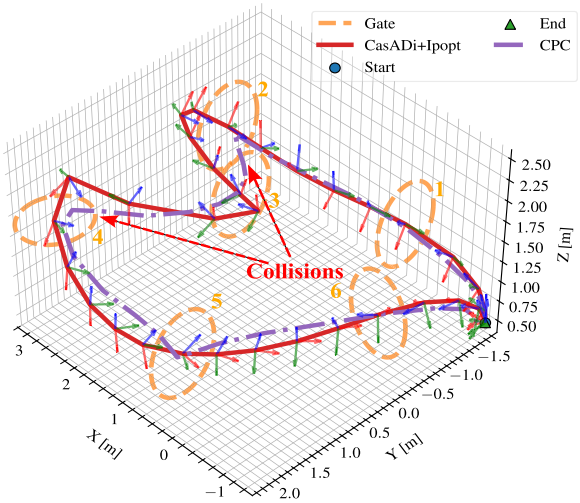


Fig. 5. Benchmark trajectory using CasADi+Ipopt [47], [48] and CPC [11]. Due to the use of waypoint-only constraints, collisions with the gates may occur along the CPC trajectory between the second and third gates.

Table II summarizes the objectives, key constraints, and comparison scope for the two experimental instances. The same generic formulation in (9) and the same convexified subproblem (26) are used. Only the task-specific cost and convex constraint sets are changed.

B. Standard-Quadrotor Case: Multi-Gate Racing

For the racing scenario, the quadrotor must traverse 6 gates while minimizing total flight time. The optimized trajectory is shown in Fig. 4. The initialization (point-mass convex program) already satisfies the geometric gate constraints and is obtained by solving a single convex problem, yielding very low solve time (see Table I). The benchmark solution using CasADi+Ipopt is shown in Fig. 5. The CPC method [11] is also included. CPC may lead to infeasible (colliding) trajectories in tighter layouts (e.g. collision between the second and the third gate in Fig. 5), whereas the proposed framework enforces all gate-passage constraints.

A Monte Carlo study with 50 randomized gate configurations evaluates efficiency and robustness. All methods use the same number of discretization nodes (36 nodes). Timing statistics are reported in Table I and Fig. 6. The proposed

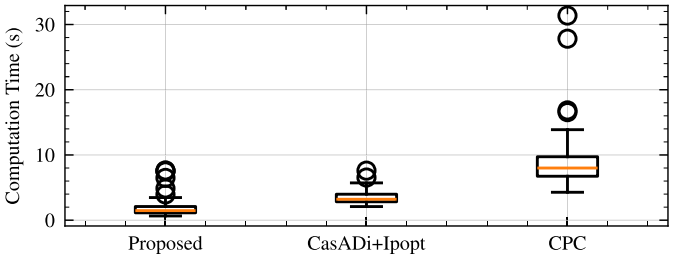
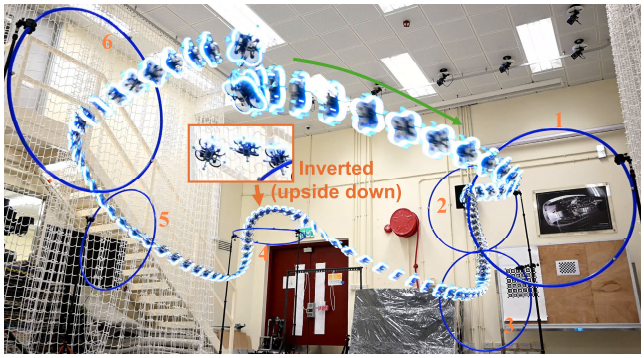


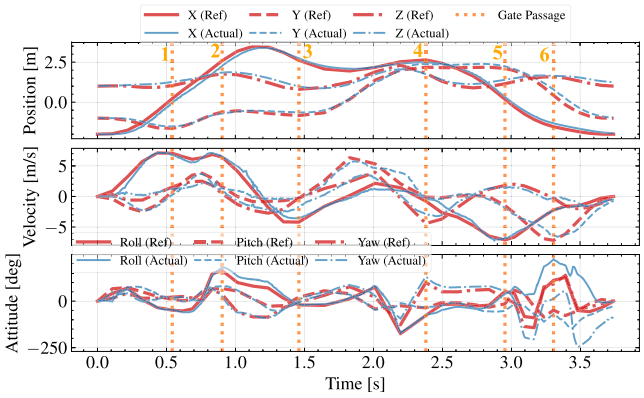
Fig. 6. Monte carlo timing distribution for 50 randomized gate configurations of the proposed method, CasADi+Ipopt [47], [48], and CPC [11].

method attains the lowest mean and median solve times with reduced variability versus CPC. CPC shows the largest dispersion and the highest average time. Both the proposed approach and CasADi+Ipopt benefit from exploiting problem convexity, explaining their superior tractability relative to CPC (see Table I), which uses the same nonlinear solver stack as CasADi+Ipopt with waypoint constraints.

In terms of optimality, the proposed method achieves $J_{\text{cost}} = 3.78$. A finer-discretization (50 nodes) CasADi+Ipopt solve yields $J^\star = 3.73$, giving a 1.3% gap. CPC attains $J_{\text{cost}} = 3.47$ but ignores full gate pass constraints, trading physical feasibility for marginal cost improvement.



(a) Executed gate traversal and the real-world flight.



(b) Tracking performance during flight.

Fig. 7. Hardware quadrotor racing experiment. The vehicle satisfies all gate constraints and tracks the optimized trajectory with small errors.

A hardware experiment validates practical performance (see Fig. 7). The quadrotor successfully tracks the optimized trajectory, traversing all gates by following the geometric constraints. Notably, the quadrotor performs an inverted maneuver during flight to generate downward thrust for rapid passage through a horizontal gate. This inverted motion emerges naturally from the optimization and is not provided in the initial guess. The position tracking root-mean-square errors (RMSEs) along the three Cartesian axes are 0.180 m, 0.108 m, and 0.146 m, respectively (Fig. 7b). Additional flights with varied, randomly arranged gate layouts were completed without collisions (see supplementary video).

C. Generality: SE(3)-Constrained Filming for a Tilt-Quad

The tilt-quadrotor is tasked with passing three waypoints and returning to the start position while minimizing control effort. Between the first and third waypoints, the body pitch is constrained to 90°, enforced via the fixed attitude quaternion. The problem is divided into four phases, each discretized with six nodes in the initial study.

We first run the algorithm up to $i_{\max}$ iterations to characterize convergence behavior. The optimized trajectories using different λ_{tr} after 500 iterations are shown in Fig. 8. The tilt-quadrotor visits all waypoints while satisfying the prescribed 90° pitch constraint between the first and third waypoints.

Convergence diagnostics are summarized in Fig. 9. To assess dynamic consistency, we generate an open-loop rollout

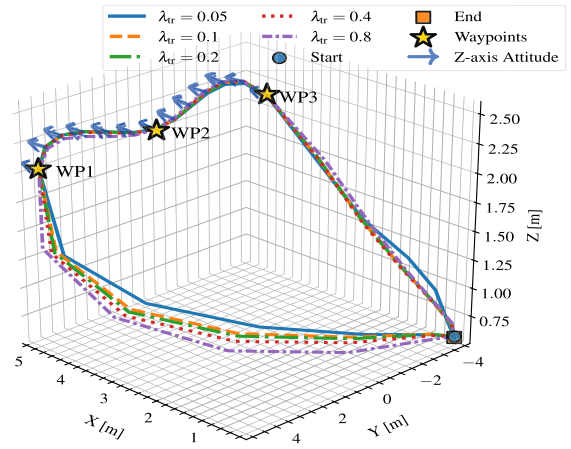


Fig. 8. Optimized trajectories for the SE(3)-constrained tiltable multicopter with different λ_{tr}. Waypoints are reached while enforcing the 90° pitch constraint between the first and third waypoints.

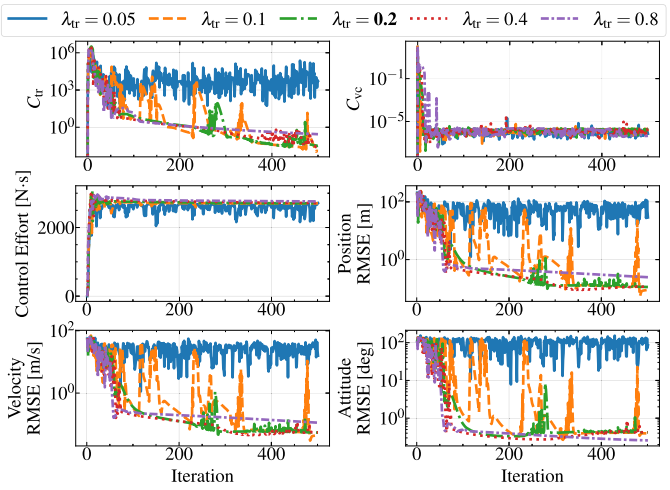


Fig. 9. Convergence of the C_{tr}, C_{vc}, objective (control effort), and open-loop rollout RMSEs across different λ_{tr} values using the proposed algorithm.

by forward-simulating the full nonlinear model using the optimized control sequence with FOH on the inputs. The RMSE between the optimized and rolled-out trajectories is reported alongside the objective (control effort) and the convergence criteria C_{tr} and C_{vc}. The results indicate that: (i) for $\lambda_{\text{tr}} \geq 0.1$, the algorithm converges reliably; (ii) C_{tr} is positively correlated with the open-loop RMSE, suggesting tighter trust regions improve dynamic consistency; and (iii) very large λ_{tr} (e.g., 0.8) yields stable reduction of C_{tr} but can degrade the control-effort objective. Moderate values (e.g., 0.2–0.4) strike a favorable balance between convergence speed, stability, and optimality.

To assess the role of the line search strategy, we perform an ablation that disables the line-search update. As shown in Fig. 10, the SCP fails to converge without line search for all tested values of λ_{tr}. In these runs, the trust-region criterion C_{tr} remains large, and the resulting trajectory is dynamically infeasible. Sensitivity to penalty weights is more pronounced in tilt-multirotors than in standard multirotors, as their dynamics are more complex and exhibit greater nonlinearity.

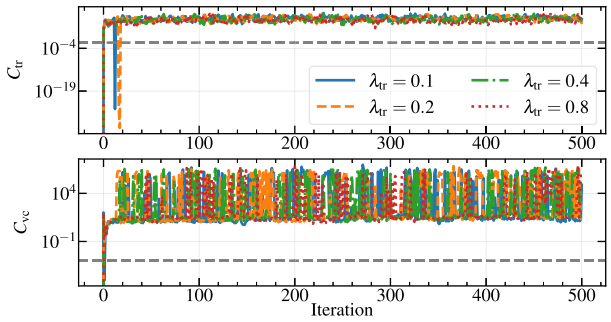


Fig. 10. Ablation without line search. Across λ_{tr} values, the SCP iterations do not converge: C_{tr} and C_{vc} remain high (compare also with the converged cases in Fig. 9).

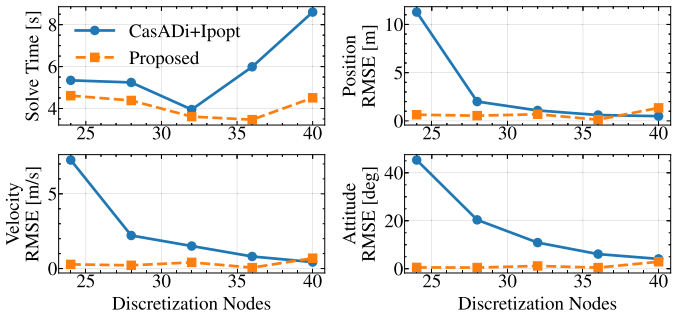


Fig. 11. Solve time and open-loop rollout RMSEs versus the number of discretization nodes for the proposed method and CasADi+Ipopt [47], [48] on the SE(3) flight.

TABLE III

PER-ITERATION TIMING VS. DISCRETIZATION NODES (SE(3) TASK) WITH OPTIMALITY GAP RELATIVE TO $J^\star$

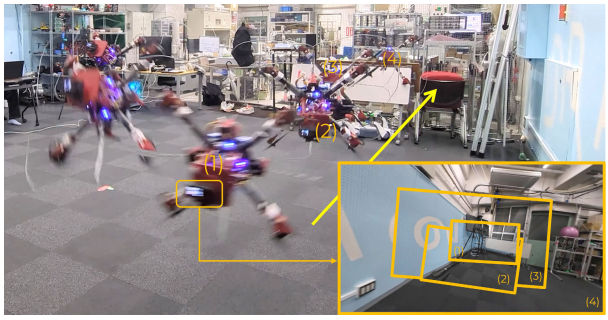
Nodes	Its	AvgTot [s]	SolveTot [s]	SolveAvg [s]	DiscAvg [s]	LSAvg [s]	Gap [%]
24	110	**0.041**	4.52	0.035	**0.005**	0.001	3.85
28	105	0.041	4.29	**0.034**	0.006	0.001	3.49
32	77	0.046	3.53	0.038	0.006	0.001	3.56
36	**67**	0.051	**3.39**	0.042	0.007	0.001	3.16
40	78	0.057	4.41	0.047	0.008	0.001	**2.82**

Nodes: discretization nodes. Its: iterations to convergence. **AvgTot**: average total wall time per iteration. **SolveTot**: cumulative solve time. **SolveAvg**: average convex solver time per iteration. **DiscAvg**: average dynamics discretization time per iteration. **LSAvg**: average line-search time per iteration. **Gap**: $(J_{\text{cost}} - J^\star)/J^\star \times 100\%$.

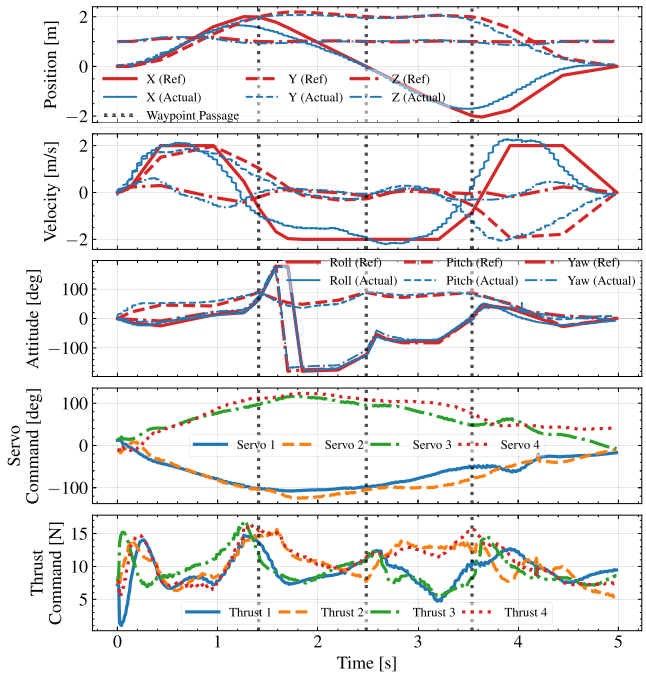
Guided by the above analysis, $\lambda_{tr} = 0.2$ and $\epsilon_{tr} = 6$ are used in the remainder. We then study solve time and open-loop accuracy versus the number of discretization nodes and compare against the CasADi+Ipopt benchmark. As shown in Fig. 11, the proposed method is faster across all discretizations. Moreover, it maintains strong open-loop fidelity even with fewer nodes, owing to the multiple-shooting transcription (20) (see also [30]). The open-loop accuracy exhibits consistent behavior as the discretization is refined. When CasADi+Ipopt achieves similar open-loop accuracy, its solve time is about twice that of the proposed algorithm.

Detailed timing statistics are summarized in Table III. As the number of nodes increases, the average per-iteration wall time grows moderately, driven mainly by the convex subproblem solves. Discretization and line-search times (DiscAvg and LSAvg in Table III, respectively) remain small. The optimality gap decreases with discretization refinement, yet stays within a few percent even for coarse discretizations, indicating efficiency. The reference cost $J^\star = 2646.07$ is obtained by solving the nonconvex problem in CasADi with Ipopt using a fine discretization of 80 nodes. The gap is $(J_{\text{cost}} - J^\star)/J^\star \times 100\%$, where J_{cost} is the control-effort cost from the proposed method.

To validate the method in practice, we enforce attitude constraints at the waypoints (instead of the whole interval), which suffices to demonstrate the approach (see Figs. 12a and 13a). From both the onboard and external camera views, the vehicle's attitude can be clearly observed as it maneuvers through the waypoints while adhering to the specified SE(3) constraints. Two hardware experiments are performed with a tiltable quadrotor: (i) pass three waypoints while maintaining a 90° pitch at each waypoint; and (ii) pass the waypoints with a −90° roll. In the first experiment, the onboard camera observes the external camera during the 90° pitch maneuver, confirming that the vehicle achieves and maintains the

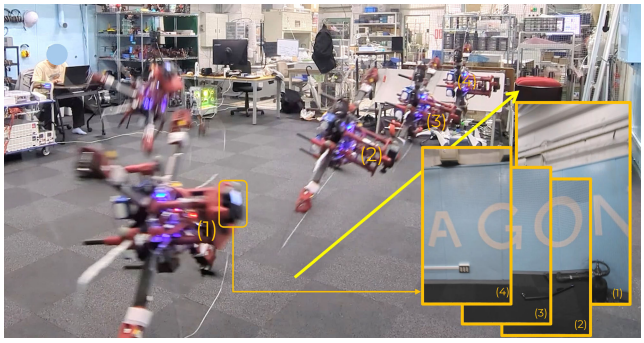


(a) Experiment snapshots with onboard camera view.

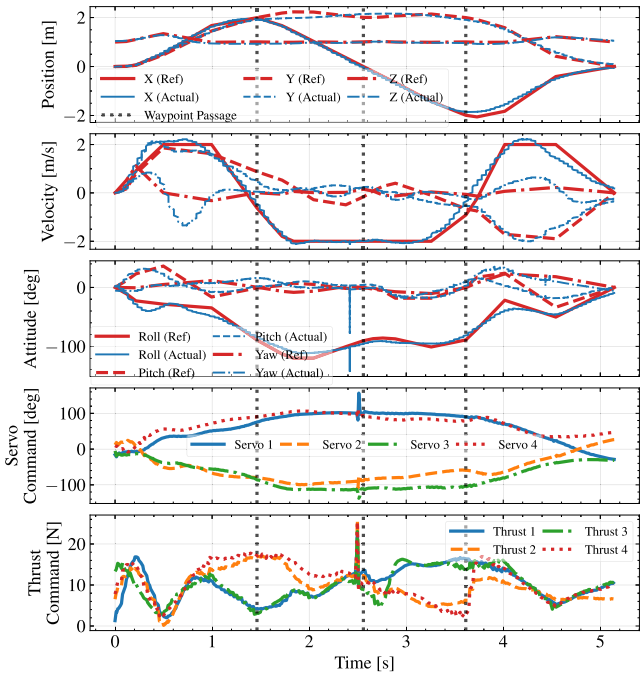


(b) Tracking performance and commanded inputs during flight.

Fig. 12. Hardware demonstration of SE(3)-constrained flight with a tiltable quadrotor. The vehicle passes three waypoints while maintaining a 90° pitch at each waypoint (the time points illustrated by vertical black dotted lines).



(a) Experiment snapshots with onboard camera view.



(b) Tracking performance and commanded inputs during flight.

Fig. 13. Hardware demonstration of SE(3)-constrained flight with a tiltable quadrotor. The vehicle passes three waypoints while maintaining a −90° roll at each waypoint (the time points illustrated by vertical black dotted lines).

commanded attitude. In the second experiment, the onboard camera records the roll maneuver, yielding a continuous view of the room wall. The results (Figs. 12 and 13) show successful task completion while adhering to the prescribed orientation constraints. The tracking results (Figs. 12b and 13b) exhibit small deviations from the optimized trajectories, indicating high-fidelity execution. In Fig. 13b, a spike in the actual attitude curve is observed, attributable to state-estimation noise induced by the motion-capture system. Additional experiments are provided in the supplementary video.

D. Discussions

1) Importance of Line Search: The ablation study in Section V-C underscores the critical role of the line search strategy. In this experiment, removing the line-search safeguard causes the SCP to fail to converge (Fig. 10), leading to dynamically infeasible trajectories. This empirical observation supports the practical robustness motivation discussed in Section III-C.

2) Emergent Behaviors and Generality: The framework's ability to handle diverse dynamics is evidenced by its success on both standard quadrotors and tilt-multirotors. In the racing experiment, the emergence of an inverted flight maneuver, without explicit encoding in the initial guess, demonstrates the solver's capability to explore the full dynamic range of the vehicle to minimize time.

3) Benefits of Convexity Exploitation: A central finding is that exploiting convexity within the multiphase free-final-time formulation significantly enhances computational tractability. As observed in the quadrotor racing comparison (Table I), our method demonstrates not only lower mean solve times but also substantially reduced variance compared to the CPC benchmark. This structural advantage allows for consistent performance even when using the same nonconvex solver.

4) Real-World Feasibility and Repeatability: Finally, the successful hardware experiments validate the physical feasibility and repeatability of the optimized trajectories.

5) Aerodynamic Drag and Extension to More Complex Constraints: The drag-free model is used consistently in both the proposed optimizer and the compared optimization baselines. In the indoor hardware experiments, residual model mismatch, including unmodeled drag, is handled by the low-level NMPC controller and feedback tracking. This modeling choice is also consistent with the full-dynamics racing formulation in [11], where aerodynamic drag is not explicitly included but successful real-world racing performance is still demonstrated. In our hardware racing experiment, the tracking-error plot in Fig. 7b also shows small deviations, indicating that the current model is adequate for the experimental setting considered in this work.

Nevertheless, drag can become important for high-speed outdoor flight and may noticeably affect trajectory accuracy. The proposed framework can incorporate such effects by augmenting the translational dynamics. For example, one may replace (1b) with $\dot{\boldsymbol{v}} = \frac{1}{m}(R_B^W \boldsymbol{f}_B + \boldsymbol{f}_{\text{drag}}(\boldsymbol{x})) + \boldsymbol{g}$, similar to the drag-aware model used in [34]. The additional term $\boldsymbol{f}_{\text{drag}}(\boldsymbol{x})$ can then be linearized together with the nominal dynamics in each SCP iteration. This modification increases model fidelity but does not change the overall successive-convexification structure.

The framework can also be extended to handle more complex constraints beyond the convex state/control constraints emphasized in this paper. For example, continuous-time constraints and other nonconvex constraints can be addressed through successive linearization, see, e.g., [29], [32]. Since this manuscript focuses on exploiting convex structures for generalized multirotor trajectory optimization, a detailed treatment of these extensions is left for future work.

VI. Conclusion

This work presented a trajectory optimization framework tailored for generalized multirotors with complex dynamics. We formulated a multiphase, free-final-time optimal control problem that incorporates whole-body 6-DoF dynamics and convex set constraints. To solve this nonconvex problem efficiently, we developed the PTR-SCP-LS algorithm. A key

contribution of this algorithm is the integration of a line-search mechanism, which significantly enhances robustness and reduces sensitivity to penalty hyperparameters. The theoretical discussion clarifies that the formal stationary-point guarantee is inherited from the unrelaxed prox-linear core, whereas the proposed line-search update serves as a practical safeguarding mechanism that reduces nonlinear rollout mismatch. Extensive simulations and hardware experiments validated both the performance and the generality of the framework. The proposed framework demonstrated superior computational efficiency and stability compared to nonconvex solvers, achieving trajectories with few-percent optimality gaps.

ACKNOWLEDGMENT

This study was supported by Otto Poon Charitable Foundation Smart Cities Research Institute, The Hong Kong Polytechnic University (CDC3).

REFERENCES

[1] M. Zhao, K. Okada, and M. Inaba, "Versatile articulated aerial robot DRAGON: Aerial manipulation and grasping by vectorable thrust control," *Int. J. Robot. Res.*, vol. 42, nos. 4–5, pp. 214–248, Apr. 2023.

[2] K. Bodie et al., "Active interaction force control for contact-based inspection with a fully actuated aerial vehicle," *IEEE Trans. Robot.*, vol. 37, no. 3, pp. 709–722, Jun. 2021.

[3] M. Allenspach et al., "Design and optimal control of a tiltrotor micro-aerial vehicle for efficient omnidirectional flight," *Int. J. Robot. Res.*, vol. 39, nos. 10–11, pp. 1305–1325, Sep. 2020.

[4] J. Li, J. Sugihara, and M. Zhao, "Servo integrated nonlinear model predictive control for overactuated tiltable-quadrotors," *IEEE Robot. Autom. Lett.*, vol. 9, no. 10, pp. 8770–8777, Oct. 2024.

[5] R. Guan, R. Xing, N. Hao, F. He, H. Luo, and Y. Yao, "A novel coaxial dual-propeller omnidirectional tiltrotor aircraft: Design and control," *IEEE Trans. Ind. Electron.*, vol. 73, no. 2, pp. 1–12, Feb. 2026.

[6] J. Li, J. Li, K. Kaneko, H. Liu, L. Shu, and M. Zhao, "Six-DoF hand-based teleoperation for omnidirectional aerial robots," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2025, pp. 15482–15488.

[7] Z. Wang, X. Zhou, C. Xu, and F. Gao, "Geometrically constrained trajectory optimization for multicopters," *IEEE Trans. Robot.*, vol. 38, no. 5, pp. 3259–3278, Oct. 2022.

[8] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *Proc. IEEE Int. Conf. Robot. Autom.*, Shanghai, China, May 2011, pp. 2520–2525.

[9] J. Tordesillas, B. T. Lopez, M. Everett, and J. P. How, "FASTER: Fast and safe trajectory planner for navigation in unknown environments," *IEEE Trans. Robot.*, vol. 38, no. 2, pp. 922–938, Apr. 2022.

[10] W. Deng, H. Chen, B. Ye, H. Chen, Z. Li, and X. Lyu, "Whole-body integrated motion planning for aerial manipulators," *IEEE Trans. Robot.*, vol. 41, pp. 1–20, 2025.

[11] P. Foehn, A. Romero, and D. Scaramuzza, "Time-optimal planning for quadrotor waypoint flight," *Sci. Robot.*, vol. 6, no. 56, p. 1221, Jul. 2021.

[12] J. Lu, X. Zhang, H. Shen, L. Xu, and B. Tian, "You only plan once: A learning-based one-stage planner with guidance learning," *IEEE Robot. Autom. Lett.*, vol. 9, no. 7, pp. 6083–6090, Jul. 2024.

[13] J. Lu et al., "YOPOv2-tracker: An end-to-end agile tracking and navigation framework from perception to action," 2025, *arXiv:2505.06923*.

[14] Y. Lee, J. Park, S. Jung, B. Jeon, D. Oh, and H. J. Kim, "QP chaser: Polynomial trajectory generation for autonomous aerial tracking," *IEEE Trans. Autom. Sci. Eng.*, vol. 22, pp. 24177–24194, 2025.

[15] H. Wang, X. Zhang, Y. Liu, G. Sun, X. Zhang, and Y. Zhuang, "PCDCT: Perception-complementarity-driven collaborative trajectory generation for vision-based aerial tracking," *IEEE Trans. Autom. Sci. Eng.*, vol. 22, pp. 10520–10532, 2025.

[16] F. Yang, Q. Lu, B. Zhang, N. Huang, and Y. Choi, "Dynamic trajectory planning for a group of unmanned aerial vehicles in unknown environments," *IEEE Trans. Autom. Sci. Eng.*, vol. 23, pp. 882–899, 2026.

[17] J. Yan, Z. Zhou, and B. Chen, "Multi-UAV cooperative trajectory planning with dynamic programming," *IEEE Trans. Autom. Sci. Eng.*, vol. 23, pp. 8800–8811, 2026.

[18] Z. Han et al., "Hierarchically depicting vehicle trajectory with stability in complex environments," *Sci. Robot.*, vol. 10, no. 103, p. 4551, Jun. 2025.

[19] V. Freire and X. Xu, "Flatness-based quadcopter trajectory planning and tracking with continuous-time safety guarantees," *IEEE Trans. Control Syst. Technol.*, vol. 31, no. 6, pp. 2319–2334, Nov. 2023.

[20] D. Lee, B. Kim, and H. J. Kim, "Autonomous aerial manipulation at arbitrary pose in SE(3) with robust control and whole-body planning," *Int. J. Robot. Res.*, vol. 45, no. 7, pp. 1–25, Jun. 2026.

[21] D. Malyuta et al., "Convex optimization for trajectory generation: A tutorial on generating dynamically feasible trajectories reliably and efficiently," *IEEE Control Syst. Mag.*, vol. 42, no. 5, pp. 40–113, Oct. 2022.

[22] Y. Chen, J. Li, H. Liu, Z. Luo, K. Kaneko, and M. Zhao, "Hierarchical trajectory planning of floating-base multi-link robot for maneuvering in confined environments," *IEEE Trans. Autom. Sci. Eng.*, vol. 23, pp. 5460–5477, 2026.

[23] D. Malyuta and B. Açıkmeşe, "Fast homotopy for spacecraft rendezvous trajectory optimization with discrete logic," *J. Guid., Control, Dyn.*, vol. 46, no. 7, pp. 1262–1279, Jul. 2023.

[24] T. Kim, A. G. Kamath, N. Rahimi, B. Açıkmeşe, M. Mesbahi, and J. Corleis, "Six-degree-of-freedom aircraft landing trajectory planning with runway alignment," *J. Guid., Control, Dyn.*, vol. 48, no. 10, pp. 2225–2242, Oct. 2025.

[25] P. Zhang, W. Li, and S. Gong, "Ascent trajectory optimization for boost-glide vehicle using homotopy approximation function sequential convex programming," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 61, no. 3, pp. 7576–7596, Jun. 2025.

[26] L. Xie, Z. Gong, H. Zhang, G. Tang, and S. Li, "Penalty separation-based sequential convex programming for lunar spacecraft reentry trajectory optimization," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 62, pp. 646–660, 2026.

[27] R. Bonalli, T. Lew, and M. Pavone, "Analysis of theoretical and numerical properties of sequential convex programming for continuous-time optimal control," *IEEE Trans. Autom. Control*, vol. 68, no. 8, pp. 4570–4585, Aug. 2023.

[28] L. Xie, X. Zhou, H.-B. Zhang, and G.-J. Tang, "Descent property in sequential second-order cone programming for nonlinear trajectory optimization," *J. Guid., Control, Dyn.*, vol. 46, no. 12, pp. 2346–2361, Dec. 2023.

[29] P. Elango, D. Luo, A. G. Kamath, S. Uzun, T. Kim, and B. Açıkmeşe, "Continuous-time successive convexification for constrained trajectory optimization," *Automatica*, vol. 180, Oct. 2025, Art. no. 112464.

[30] Z. Shen, G. Zhou, and H. Huang, "Sequential convex programming for time-optimal quadrotor waypoint flight," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2024, pp. 3108–3115.

[31] Z. Shen, G. Zhou, H. Huang, C. Huang, Y. Wang, and F.-Y. Wang, "Convex optimization-based trajectory planning for quadrotors landing on aerial vehicle carriers," *IEEE Trans. Intell. Vehicles*, vol. 9, no. 1, pp. 138–150, Jan. 2024.

[32] C. R. Hayner, J. M. Carson, B. Açıkmeşe, and K. Leung, "Continuous-time line-of-sight constrained trajectory planning for 6-degree of freedom systems," *IEEE Robot. Autom. Lett.*, vol. 10, no. 5, pp. 4332–4339, May 2025.

[33] J. Nocedal and S. J. Wright, *Numerical Optimization*. Cham, Switzerland: Springer, 2006.

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

[35] M. Krinner, A. Romero, L. Bauersfeld, M. Zeilinger, A. Carron, and D. Scaramuzza, "MPCC++: Model predictive contouring control for time-optimal flight with safety constraints," in *Proc. Robot., Sci. Syst.*, Jul. 2024, p. 109.

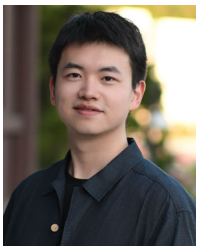
[36] F. Zhao, J. Mei, J. Zhou, Y. Chen, J. Chen, and S. Li, "Gate-aware online planning for two-player autonomous drone racing," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2025, pp. 8536–8542.

[37] A. Romero, R. Penicka, and D. Scaramuzza, "Time-optimal online replanning for agile quadrotor flight," *IEEE Robot. Autom. Lett.*, vol. 7, no. 3, pp. 7730–7737, Jul. 2022.

[38] Y. Ren et al., "Safety-assured high-speed navigation for MAVs," *Sci. Robot.*, vol. 10, no. 98, p. 6187, Jan. 2025.

[39] A. Ghotavadekar, F. Nekovář, M. Saska, and J. Faigl, "Variable time-step MPC for agile multi-rotor UAV interception of dynamic targets," *IEEE Robot. Autom. Lett.*, vol. 10, no. 2, pp. 1249–1256, Feb. 2025.

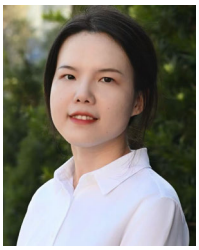
[40] Y. Zhang, Y. Hu, Y. Song, D. Zou, and W. Lin, "Learning vision-based agile flight via differentiable physics," *Nature Mach. Intell.*, vol. 7, no. 6, pp. 954–966, Jun. 2025.
[41] E. Kaufmann, L. Bauersfeld, A. Loquercio, M. Müller, V. Koltun, and D. Scaramuzza, "Champion-level drone racing using deep reinforcement learning," *Nature*, vol. 620, no. 7976, pp. 982–987, Aug. 2023.
[42] Y. Song, A. Romero, M. Müller, V. Koltun, and D. Scaramuzza, "Reaching the limit in autonomous racing: Optimal control versus reinforcement learning," *Sci. Robot.*, vol. 8, no. 82, p. 1462, Sep. 2023.
[43] H. Zhao, N. Schlüter, L. Brunke, and A. P. Schoellig, "Improving drone racing performance through iterative learning MPC," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2025, pp. 6667–6674.
[44] A. Verraest, S. Bahnam, R. Ferede, G. de Croon, and C. De Wagter, "SkyDreamer: Interpretable end-to-end vision-based drone racing with model-based reinforcement learning," 2025, *arXiv:2510.14783*.
[45] S.-A. Yu, C. Yu, F. Gao, Y. Wu, and Y. Wang, "FlightBench: Benchmarking learning-based methods for ego-vision-based quadrotors navigation," *IEEE Robot. Autom. Lett.*, vol. 10, no. 7, pp. 6888–6895, Jul. 2025.
[46] M. Bos, W. Decre, J. Swevers, and G. Pipeleers, "Multi-stage optimal control problem formulation for drone racing through gates and tunnels," in *Proc. IEEE 17th Int. Conf. Adv. Motion Control (AMC)*, Feb. 2022, pp. 376–382.
[47] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Math. Program.*, vol. 106, no. 1, pp. 25–57, Mar. 2006.
[48] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi: A software framework for nonlinear optimization and optimal control," *Math. Program. Comput.*, vol. 11, no. 1, pp. 1–36, Mar. 2019.
[49] D. Drusvyatskiy and A. S. Lewis, "Error bounds, quadratic growth, and linear convergence of proximal methods," *Math. Oper. Res.*, vol. 43, no. 3, pp. 919–948, Aug. 2018.
[50] S. K. Lam, A. Pitrou, and S. Seibert, "Numba: A LLVM-based Python JIT compiler," in *Proc. 2nd Workshop LLVM Compiler Infrastruct. HPC*, Nov. 2015, pp. 1–6.
[51] S. Diamond and S. Boyd, "CVXPY: A Python-embedded modeling language for convex optimization," *J. Mach. Learn. Res.*, vol. 17, no. 83, pp. 1–5, 2016.



Zhipeng Shen (Member, IEEE) received the B.Eng. degree in aircraft design and engineering from Northwestern Polytechnical University, Xi'an, China, in 2019, the M.Eng. degree in control engineering from Beihang University, Beijing, China, in 2022, and the Ph.D. degree from The Hong Kong Polytechnic University, Hong Kong, in 2025. His research interests include trajectory optimization, machine learning, and optimal control in robotics.

Jinjie Li (Graduate Student Member, IEEE) received the B.Eng. degree in automation and the M.Sc. degree in control science and engineering from Beihang University, Beijing, China, in 2020 and 2023, respectively. He is currently pursuing the Ph.D. degree with the Department of Mechanical Engineering, The University of Tokyo, Tokyo, Japan.

His research interests include optimization-based control for aerial manipulation, aiming to make aerial robots function as flying hands rather than just eyes.



Shiyu Zhou (Graduate Student Member, IEEE) received the B.Eng. degree in automation from Northwestern Polytechnical University, Xi'an, China, in 2019, and the M.Eng. degree in control science and engineering from Beihang University, Beijing, China, in 2022. She is currently pursuing the Ph.D. degree with the Department of Mechanical Engineering, City University of Hong Kong, Hong Kong.

Her current research interests include cooperative control of multiagent systems and synchronization of complex networks.

Moju Zhao (Member, IEEE) received the Ph.D. degree from the Department of Mechano-Informatics, The University of Tokyo, in 2018.

He is currently a Lecturer (a Junior Associate Professor) with The University of Tokyo. His research interests include mechanical design, modeling and control, motion planning, and vision-based recognition in aerial robotics. His main achievements include articulated aerial robots, such as DRAGON and SPIDAR.

Dr. Zhao has received several awards at conferences and in journals, including the Best Paper Award at IEEE ICRA 2018. He has been an Associate Editor of IEEE TRANSACTIONS ON ROBOTICS since 2026.

Hailong Huang (Senior Member, IEEE) received the Ph.D. degree in systems and control from the University of New South Wales, Sydney, Australia, in 2018.

He is currently an Assistant Professor with the Department of Aeronautical and Aviation Engineering, The Hong Kong Polytechnic University, Hong Kong. His research interests include guidance, navigation, and control of UAVs and mobile robots.

Dr. Huang is an Associate Editor of IEEE TRANSACTIONS ON AUTOMATION SCIENCE AND ENGINEERING, *Journal of Field Robotics*, IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, and IEEE TRANSACTIONS ON INTELLIGENT VEHICLES.