Geometrically Constrained Trajectory Optimization for Multicopters

Zhepei Wang, Xin Zhou, Chao Xu, and Fei Gao

Abstract—In this article, we present an optimization-based framework for multicopter trajectory planning subject to geometrical configuration constraints and user-defined dynamic constraints. The basis of the framework is a novel trajectory representation built upon our novel optimality conditions for unconstrained control effort minimization. We design linear-complexity operations on this representation to conduct spatial–temporal deformation under various planning requirements. Smooth maps are utilized to exactly eliminate geometrical constraints in a lightweight fashion. A variety of state-input constraints are supported by the decoupling of dense constraint evaluation from sparse parameterization and the backward differentiation of flatness map. As a result, this framework transforms a generally constrained multicopter planning problem into an unconstrained optimization that can be solved reliably and efficiently. Our framework bridges the gap among solution quality, planning efficiency, and constraint fidelity for a multicopter with limited resources and maneuvering capability. Its generality and robustness are both demonstrated by applications to different flight tasks. Extensive simulations and benchmarks are also conducted to show its capability of generating high-quality solutions while retaining the computation speed against other specialized methods by orders of magnitude.

Index Terms—Aerial systems, applications, autonomous vehicle navigation, collision avoidance, motion and path planning.

I. INTRODUCTION

MULTICOPTERS rely on robust and efficient trajectory planning for safe yet agile autonomous navigation in complex environments [1]–[6]. For robotics, precisely incorporating dynamics, smoothness, and safety are essential to generate high-quality motions. Moreover, lightweight robots, such as multicopters under size, weight, and power constraints, put further hard requirements on the real-time computing using limited onboard resources. Despite that various successful tools in general-purpose kinodynamic planning or optimal control have been presented, few of them guarantee efficient online planning while also considering general constraints on dynamics for multicopters. Consequently, existing applications often use oversimplified requirements on trajectories for better computation efficiency, thus limiting the full exploitation of vehicle’s capability.

The high-performance planning mentioned above possesses four major algorithmic challenges. First, ensuring safety often involves frequent interactions with a large volume of highly discretized environment data. Second, the nonlinearity of vehicle dynamics brings difficulties to directly enforcing physically acceptable states and inputs when the multicopter is flying at the limit of its capabilities. Third, high-quality motions conventionally need fine discretization of the dynamic process, where requirements for task resources tend to be unrealistic. Fourth, methods that use sparse representation for trajectories lack an effective way to optimize the temporal profile while satisfying continuous-time constraints.

In this article, we overcome these challenges by designing a lightweight and flexible optimization framework to meet user-defined requirements based on a novel trajectory class.

As the theoretical foundation of our framework, we present necessary and sufficient optimality conditions to multistage control effort minimization for the concerned linear dynamics, which are given for the first time to the best of our knowledge. The conditions are easy to use in that the unique optimal solution can be directly constructed with linear complexity in both the time and space aspects. More importantly, the existence and uniqueness of conditions further provide crucial information on the smoothness of the problem parameter sensitivity.

To ease computation burden without sacrificing trajectory quality, it is essential to use sparse parameterization while keeping the flexibility to suit multicopter dynamics. Therefore, we design a novel trajectory class based on our optimality conditions. Any element in this class is by default an unconstrained control effort minimizer; thus, we name it as MINCO (minimum control). MINCO differs from conventional splines that majorly focus on the smoothness of the geometrical shape, such as B-Splines and Bézier curves. Its sparse parameters are designed to directly control both the spatial and temporal profiles of a trajectory, which are of equal importance for dynamic feasibility. Besides, a spatial–temporal deformation scheme is also designed such that MINCO can be optimized under any user-defined objective.

Our framework utilizes the geometrical approximation of low-dimensional free space based on the results of sampling-based or search-based global methods. The safety is ensured via configuration constraints formed by the union of obstacle-free convex primitives. Constraint elimination schemes are proposed...
such that MINCO can be freely deformed through unconstrained optimization. The schemes exactly eliminate constraints that are directly defined on decision variables without introducing extra local minima.

Reliable motion planning requires admissible states and inputs, while most of the existing flatness-based methods only support differential constraints. To ensure high-fidelity feasibility, we propose a systematic way to enforce user-defined state-input constraints for our sparse parameterization without resorting to a fine discretization of trajectories. We exploit the backward differentiation of flatness map such that the constraint violation can be reflected in their gradient w.r.t. sparse parameters. Besides, a differentiable penalty functional is also proposed to enforce general continuous-time constraints.

Our framework focuses on computationally efficient yet high-quality trajectory planning for multicopters where there are complex constraints for safety, critical limits on dynamics, and task-specified requirements. To validate its effectiveness, we conduct extensive benchmarks against various cutting-edge multicopter trajectory planning methods. Results show that our method exceeds existing methods for orders of magnitude in efficiency and retains comparable solution quality against general-purpose optimal-control solvers. We also conduct versatile simulations and extreme real-world flights to show the practical performance of our approach.

The contributions of this article are as follows.

1) Optimality conditions in a general form on multistage control effort minimization are proposed with a proof of both the necessity and sufficiency for the first time.

2) A novel trajectory class is designed to meet user-defined objectives while retaining local smoothness by spatial-temporal deformation via linear-complexity operations.

3) A flexible trajectory planning framework that leverages both constraint elimination and constraint transcription is proposed for multicopter systems with user-defined state-input constraints.

4) A set of simulations and experiments that validate our method significantly outperforms state-of-the-art works in efficiency, optimality, robustness, and generality.

II. RELATED WORK

Despite various planning approaches in the existing literature, there has yet to emerge a complete framework to accomplish time-critical large-scale trajectory planning for multicopters while incorporating user-defined continuous-time constraints on state and control. Our framework bridges this gap by exploring and exploiting different capabilities from both optimal control and motion planning.

A. Differentially Flat Multicopters

The concept of differential flatness has been introduced by Fliess et al. [7] and drawn great attentions in robotics trajectory planning [8]–[10]. The property makes it possible to recover the full state and input of a flat system from finite derivatives of its flat outputs. Mellinger and Kumar [11] validate the flatness of quadcopters with aligned propellers, which take the thrust and 3-D torque as inputs. Watterson and Kumar [12] use the Hopf fibration to decompose the quadcopter rotation, thus achieving the minimum singularity number in flatness maps. Ferrin et al. [13] show the flatness of a hexacopter whose inputs are desired orientation and thrust. They utilize the flatness to compute the nominal state where a linear–quadratic regulator is applied. Faessler et al. [14] further consider linear drags that produce extra linear and angular accelerations. They show the flatness of parallel-rotor multicopters subject to the drag effect. Moreover, Mu and Chirarattananon [15] investigate underactuated multicopters with tilted propellers. They prove that the flatness holds for a wide range of tricopters, quadcopters, and hexacopters as long as the input rank condition is satisfied.

The flatness of a multicopter, if holds, benefits trajectory generation and tracking control in obtaining the reference state and input without integrating differential equations. Literature mentioned above uses flatness to avoid confronting system dynamics during planning. However, dynamics of a real physical system are only valid for reasonable state and admissible input. Although our framework also utilizes the flatness property, it differs from previous works in that a general form of state-input constraints is formally supported.

B. Sampling-Based Motion Planning

Sampling-based motion planners focus on global solutions of problems by exploration and exploitation, where the complexity mainly originates from the configuration space. The probabilistic roadmap (PRM) [16] and the rapidly exploring random tree (RRT) [17] are both probabilistically complete since their probability of failure decays to zero exponentially as the sample number goes to infinity [18]. Karaman and Frazzoli [19] propose asymptotically optimal variants of PRM and RRT, known as PRM* and RRT*, which ensure the convergence to globally optimal solutions as the sample number goes to infinity. There are also algorithms [20]–[22] that further improve the efficiency or applicability of randomized motion planning. Our method exploits sampling-based planners to overcome the complexity from environments. It accomplishes the optimization of a dynamically feasible trajectory that is homotopic to a given low-dimensional collision-free path. It is designed to flexibly incorporate system state-input constraints, which is not the strength of sampling-based methods. In this way, the complexity from both the environments and dynamics is divided and conquered.

C. Optimization-Based Motion Planning

Optimization-based planners focus on local solutions by using high-order information of the problems. They depend on specific environment preprocessing methods such that the obstacle information is encoded into the optimization. Trajectory optimization has long been studied for general systems in the control community [23]. Many general-purpose methods are designed for high-quality solutions, such as the collocation-based method GPOPS-II [24], and the shooting-based one ACADO [25]. They transcribe the original problem into a nonlinear programming (NLP) using a lot of equalities and variables and, then, resort to well-established NLP solvers, such as SNOPT [26] or IPOPT [27]. However, trajectory planning in robotics may impose hard-to-formulate constraints, nonsmoothness, and integer variables. Besides, general-purpose methods often take a long computation time, making them inappropriate for time-critical tasks. For example, Bry et al. [28] report that direct collocation with SNOPT takes several minutes to optimize a 4.5-m trajectory for a 12-state airplane flying among cylindrical
obstacles [29]. Therefore, specialized methods are on calling to overcome these difficulties.

For differentially flat multicopters, motion planning can be transformed into the optimization of low-dimensional trajectories of flat outputs. Mellinger and Kumar [11] use fixed-duration splines to represent quadcopter trajectories. A quadratic programming (QP) is formulated by the quadratic cost of snap and linear constraints of safety. However, perturbation problems need to be solved in finite difference to estimate the gradient for time allocation. Its actuator constraints are also oversimplified. Bry et al. [28] propose a closed-form solution for this QP without safety constraints. They heuristically add waypoints from a collision-free path of RRT* to recompute the solution until the safety is satisfied. This method is admittedly efficient but cannot guarantee high-quality solutions in obstacle-rich environments. Besides, it involves the inverse of a matrix whose nonsingularity is never discussed. Deits and Tedrake [30] approximate the free space using polytopes. The safety of each piece of trajectory is equivalent to a sum-of-square (SOS) condition if it entirely lies inside a polytope. They solve interval assignment using mixed-interger second-order cone programming (MISOCP). It generates globally optimal trajectories, while the computation time is unacceptable. Gao et al. [31] also use the polyhedral-shaped free space representation. They alternately optimize the geometrical and temporal profiles of a trajectory. The safety is enforced by the convex-hull property of Bézier curves, and the dynamic profile is optimized via time-optimal path parameterization (TOPP) [32]. There are also variants that propose improvements over the above methods. For example, Tordesillas et al. [33] improve the efficiency of [30] by substituting SOS conditions on polynomials with linear constraints on Bézier curves at the cost of conservatism [34]. Sun et al. [35] avoid integer variables by optimizing time allocation instead, where the sensitivity of a bilevel optimization is exploited.

These specialized methods utilize the continuous-time trajectory parameterization to avoid the computation burden from the fine discretization. However, they do not support flexibly optimizing its time allocation, decoupling temporal resolutions of constraints from decision variable dimensions, or enforcing high-fidelity constraints except for restrictions on derivative norms. In this article, our framework supports all these features by introducing unified techniques for a novel sparse parameterization. Moreover, the solution quality is comparable with that of the general-purpose optimal-control solvers.

### III. Preliminaries

#### A. Differential Flatness

Consider a dynamical system of the following type:

$$\dot{x} = f(x) + g(x)u$$  \hspace{1cm} (1)

with $f: \mathbb{R}^n \rightarrow \mathbb{R}^n$, $g: \mathbb{R}^n \rightarrow \mathbb{R}^{n \times m}$, state $x \in \mathbb{R}^n$, and input $u \in \mathbb{R}^m$. The map $g$ is assumed to have rank $m$. The system is said to be differentially flat [7] if there exists a flat output $z \in \mathbb{R}^m$ determined by $x$ and finite derivatives of $u$, such that $x$ and $u$ can both be parameterized by finite derivatives of $z$:

$$x = \Psi_x(z, \dot{z}, \ldots, z^{(s-1)})$$  \hspace{1cm} (2)

$$u = \Psi_u(z, \dot{z}, \ldots, z^{(s)})$$  \hspace{1cm} (3)

where $\Psi_x: (\mathbb{R}^m)^{s-1} \rightarrow \mathbb{R}^n$ and $\Psi_u: (\mathbb{R}^m)^s \rightarrow \mathbb{R}^m$ are both induced by $f$ and $g$. Intuitively, the state and control can be determined from $z$ without explicit integration of the system dynamics (1).

Leveraging the flatness of a system, the trajectory generation is convenient when there are only differential constraints in (1). If we introduce a new control variable $v = z^{(s)}$ and denote $z^{[s-1]} \in \mathbb{R}^m$ as

$$z^{[s-1]} = (z_T, z^T, \ldots, z^{(s-1)^T})^T$$  \hspace{1cm} (4)

the input $u = \Psi_u(z^{[s-1]}, v)$ then exactly linearizes the original flat system into $m$ decoupled chains of $s$-integrators. Let $z_i$ denote the $i$th entry in $z$, $v_i$ the $i$th entry in $v$, and $z_i^{[s-1]} = (z_i, \dot{z}_i, \ldots, z_i^{(s-1)})^T$. The $i$th integrator chain is

$$\dot{z}_i^{[s-1]} = \begin{pmatrix} 0 & I_{s-1} & 0 \\ 0 & 0 & 1 \end{pmatrix} z_i^{[s-1]} + \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} v_i$$  \hspace{1cm} (5)

where $0$ and $I$ are a zero matrix and an identity matrix with appropriate sizes, respectively. Given an initial state and a goal state, boundary values of each integrator chain (5) can be algebraically computed. Thus, any trajectory integrated from these $m$ integrator chains can be transformed into a feasible trajectory [8] for the original flat system via (2) and (3).

For dynamics with a small $m$, the flatness maps $\Psi_x$ and $\Psi_u$ further reduce the trajectory dimension and eliminate the differential constraints (1), which is illustrated in Fig. 1. As a side effect, nonlinearity coming from both $\Psi_x$ and $\Psi_u$ brings additional difficulties in trajectory generation for $z$ when there are additional state-input constraints for (1). However, such an effect is relieved if the flat-output space coincides with the configuration space of the relevant planning problem.

#### B. Direct Optimization in Flat-Output Space

Fortunately, the differential flatness of multicopters has been well studied and shown to have physically meaningful flat-output space, which overlaps with the configuration space. Explicit forms of $\Psi_x$ and $\Psi_u$ are available in [11]–[15] for a variety of underactuated multicopters. More importantly, their flat outputs share the same form in general

$$z = (p_x, p_y, p_z, \psi)^T$$  \hspace{1cm} (6)

where $(p_x, p_y, p_z)^T$ is the translation of the center of gravity and $\psi$ is the yaw angle of the vehicle. The flat output $z$, especially
its translation, provides a lot of convenience for the multicopter motion planning with complex spatial constraints.

To generate feasible motions for a multicopter, we first optimize the trajectory \( z(t) : [0, T] \rightarrow \mathbb{R}^m \) in its flat-output space such that most of the spatial constraints are directly enforced. Then, the flatness maps \( \Psi_z \) and \( \Psi_u \) are applied to transform \( z(t) \) into the state-input trajectory \( x(t) \) and \( u(t) \).

For motion smoothness, the quadratic control effort [36] with time regularization is adopted as a cost functional of \( z(t) \). General constraints on multicopters can be classified into configuration constraints and user-defined dynamic constraints. Normally, a collision-free motion implies

\[
z(t) \in \mathcal{F} \quad \forall t \in [0, T]
\]

where \( \mathcal{F} \) is the concerned obstacle-free region in configuration space. Besides, user-defined state-input constraints, such as actuator limits or task-specific constraints, are denoted by

\[
G_D(z(t), u(t)) \leq 0 \quad \forall t \in [0, T].
\]

Exploiting \( \Psi_z \) and \( \Psi_u \), the corresponding constraints on \( z(t) \) are computed as

\[
G_D(\Psi_z(z^{[s-1]}(t)), \Psi_u(z^{[s]}(t))) \leq 0 \quad \forall t \in [0, T].
\]

Apparently, via the flatness, a constraint on \( x \) and \( u \) has its equivalent form on the finite derivatives of \( z(t) \). For simplicity, we denote (9) hereafter by

\[
G(z(t), \dot{z}(t), \ldots, z^{(s)}(t)) \leq 0 \quad \forall t \in [0, T]
\]

where \( G \) consists of \( n_x \) equivalent constraints.

It is worth noting that we do not make further assumptions on the multicopter dynamics and flatness maps. In other words, the proposed framework supports a wide range of multicopters, including, but not limited to the ones in [11]–[15].

### C. Problem Formulation

Concluding above descriptions gives the following problem:

\[
\begin{alignat}{2}
\min_{z(t), T} & \quad \int_0^T v(t)^T W v(t) dt + \rho(T) \\
\text{s.t.} & \quad z^{(s)}(t) = v(t) \quad \forall t \in [0, T] \\
& \quad G(z(t), \ldots, z^{(s)}(t)) \leq 0 \quad \forall t \in [0, T] \\
& \quad z(t) \in \mathcal{F} \quad \forall t \in [0, T] \\
& \quad z^{[s-1]}(0) = \bar{z}_o, z^{[s-1]}(T) = \bar{z}_f
\end{alignat}
\]

where \( W \in \mathbb{R}^{m \times m} \) is a positive diagonal matrix, \( \rho : [0, \infty) \rightarrow [0, \infty] \) is the time regularization, \( \bar{z}_o \in \mathbb{R}^{ms} \) is the initial condition, and \( \bar{z}_f \in \mathbb{R}^{ms} \) is the terminal condition. The control input \( v \) is allowed to be discontinuous in a finite number of time instants, as is commonly assumed in the existing literature [36].

The trajectory optimization (11) is nontrivial because of the continuous-time constraints \( G \) and the nonconvex set \( \mathcal{F} \). We further specify some reasonable conditions to make it a well-defined problem. As for time regularization \( \rho \), it trades off between the control effort and the expectation of total time

\[
\rho(T) = \sum_{i=0}^{M_T} b_i T^i
\]

where \( b_{M_T} \) is positive. Common choices are \( \rho_s(T) = k_s T \) and \( \rho_u(T) = k_u (T - T_S)^2 \) with an expected time \( T_S \). Besides, \( \rho \) can also be defined to strictly fix the total time

\[
\rho_f(T) = \begin{cases} 
0, & \text{if } T = T_S \\
\infty, & \text{if } T \neq T_S.
\end{cases}
\]

As for nonlinear constraints \( G \), they are required to be \( C^2 \), i.e., twice continuously differentiable. As for the feasible region \( \mathcal{F} \) in the configuration space, we approximate it geometrically by the union of \( M_P \) closed convex sets as

\[
\mathcal{F} \simeq \mathcal{F} = \bigcup_{i=1}^{M_P} \mathcal{P}_i.
\]

For simplicity, locally sequential connection is assumed on these convex sets

\[
\begin{cases}
\mathcal{P}_i \cap \mathcal{P}_j = \emptyset, & \text{if } |i - j| = 2 \\
\text{Int}(\mathcal{P}_i \cap \mathcal{P}_j) \neq \emptyset, & \text{if } |i - j| \leq 1
\end{cases}
\]

where \( \text{Int}(\cdot) \) means the interior of a set. The translation of \( \bar{z}_o \) and \( \bar{z}_f \) is inscribed in \( \mathcal{P}_i \) and \( \mathcal{P}_{M_P} \), respectively. As for \( \mathcal{F} \), we consider the case that each \( \mathcal{P}_i \) is a closed \( m \)-dimensional ball:

\[
\mathcal{P}_i^{B} = \{ x \in \mathbb{R}^m \mid \| x - o_i \|_2 \leq r_i \}
\]

or, more generally, a bounded convex polytope described by its \( \mathcal{H} \)-representation [37] with potentially redundant constraints

\[
\mathcal{P}_i^{R} = \{ x \in \mathbb{R}^m \mid A_i x \leq b_i \}.
\]

For the optimization in (11), we aim to construct a computationally efficient solver while retaining the flexibility to handle different task-specific constraints \( G_D \) in (8).

### IV. MULTISTAGE CONTROL EFFORT MINIMIZATION

In this section, we analyze the multistage control effort minimization without functional constraints. For this problem, we propose easy-to-use optimality conditions for general cases, which are proved to be necessary and sufficient. Leveraging our conditions, the optimal trajectory is directly constructed in linear complexity of time and space, without evaluating the cost functional explicitly or implicitly. Based on them, a novel trajectory class along with linear-complexity spatial–temporal deformation is designed to meet user-defined objectives in various trajectory planning scenarios.

#### A. Unconstrained Control Effort Minimization

When constraint \( \mathcal{F} \) exists, adjusting the waypoints [28] or control points [33] of a trajectory helps to ensure safety. When constraint \( G \) exists, adjusting the time allocation also helps to enforce physical limits [38]. Therefore, spatial and temporal parameters are both vital to a flexible trajectory representation. Then, the natural problem is to generate a smooth trajectory subject to these parameters. We solve linear–quadratic minimum-time (LQMT) problems to generate trajectories from spatial–temporal parameters. Although the LQMT problems have extensive studies and applications, only single-stage problems are considered in the literature [39]–[41]. We study the multistage problems where intermediate points and time vector
are fixed in advance for multiple piecewise trajectories. Consider an \( M \)-stage control effort minimization without \( F \) and \( G \)

\[
\min_{z(t)} \int_{t_0}^{t_M} v(t)^T W v(t) dt \quad (18a)
\]

s.t. \( z(t) = v(t) \quad \forall t \in [t_0, t_M] \) \( \quad (18b) \)

\[
z^{[s-1]}(t_0) = \bar{z}_o, z^{[s-1]}(t_M) = \bar{z}_f \quad (18c)
\]

\[
z^{[d_i-1]}(t_i) = \bar{z}_{i}, 1 \leq i < M \quad (18d)
\]

\[
t_{i-1} < t_i, 1 \leq i < M. \quad (18e)
\]

The time interval \([t_0, t_M]\) is split into \( M \) stages by \( M + 1 \) fixed timestamps, with constant boundary conditions \( \bar{z}_o, \bar{z}_f \in \mathbb{R}^{m_s} \). Intermediate conditions \( \bar{z}_i \in \mathbb{R}^{m_d} \), with \( d_i \leq s \) specify the value of \( z(t_i), \bar{z}(t_i), \ldots, z^{(d_i-1)}(t_i) \), where \( d_i \) is the number of derivatives fixed at \( t_i \). For example, if \( z(t) \) is only required to pass a given position at \( t_i \), then \( d_i = 1 \) because \( \bar{z}_i \) contains the zero-order derivative and nothing else.

Existing works focus on the necessary conditions for special cases of (18). In aerial robotic areas, theQP formulation \([11]\) and the closed-form one \([28]\) implicitly or explicitly optimize unknown knot derivatives, taking parameterization as \( a \) \textit{priori}. This extra computation actually makes them less efficient. In control area, a special case where \( d_i = 1 \) is also studied in \([42]\) and \([43]\) via controllability Gramian. The result is for general linear systems with possibly nonpolynomial solutions, while it is less intuitive considering the computational aspect. These necessary conditions can cause potential degeneracy in trajectory representation and sensitivity if further parametric optimization on spatial–temporal parameters is needed.

**B. Optimality Conditions**

We propose necessary and sufficient optimality conditions for (18) with all possible settings of \( d_i, \bar{z}_i, \) and \( t_i \). Thus, an optimal trajectory can be directly constructed from spatial–temporal parameters. Furthermore, the existence and uniqueness of the optimal trajectory are always guaranteed.

We transform (18) into the Mayer form \([23]\), in which a new state \( \bar{y} \in \mathbb{R}^{m_s+1} \) augmented by \( \bar{y} \in \mathbb{R} \) is defined as

\[
\bar{y} = \begin{pmatrix} z^{[s-1]} \\ \bar{y} \end{pmatrix}. \quad (19)
\]

The augmented system \( \bar{y} = f(y, v) \) has the structure

\[
\bar{y} = \begin{pmatrix} \bar{A} & 0 \\ 0^T & 0 \end{pmatrix} y + \begin{pmatrix} 0 \\ v \end{pmatrix} v^T W v \quad (20)
\]

where

\[
\bar{A} = \begin{pmatrix} 0 & I_{m(s-1)} \\ 0_{m \times m} & 0^T \end{pmatrix} \in \mathbb{R}^{m_s \times m_s}. \quad (21)
\]

We design a running process for the augmented system in \( M \) stages, of which the \( i \)th is \( \Delta_i = [t_{i-1}, t_i] \). It is worth noting that state switching occurs in this running process. Strictly speaking, the state switching only occurs on \( \bar{y} \) at the beginning of each stage. Denote by \( \bar{y}[i] : \Delta_i \mapsto \mathbb{R}^{m_s+1} \) the augmented state trajectory in the \( i \)th stage, which consists of two parts: \( z^{[s-1]}[i] \) and \( \bar{y}[i] \). At each timestamp \( t_i \), the state transfers from \( y[i] \) to \( y[i+1] \), and the part \( \bar{y} \) is reset as

\[
\bar{y}[i+1](t_i) = 0, \quad 0 \leq i < M \quad (22)
\]

thus switching the partial state from \( \bar{y}[i](t_i) \) to 0. The \( z^{[s-1]} \) part remains continuous between stages, which means

\[
z^{[s-1]}[i](t_i) = z^{[s-1]}[i+1](t_i), \quad 1 \leq i < M. \quad (23)
\]

The conditions in (18c) and (18d) are still satisfied, i.e.,

\[
z^{[s-1]}[1](t_0) = \bar{z}_o, z^{[s-1]}[M](t_M) = \bar{z}_f \quad (24)
\]

\[
z^{[d_i-1]}[i](t_i) = \bar{z}_{i}, \quad 1 \leq i < M. \quad (25)
\]

In this process, the cost functional in (18) is converted into the sum of terminal cost of each stage for the augmented system, i.e., \( \sum_{i=1}^{M} \bar{y}[i](t_i) \). Therefore, the optimal trajectories for the augmented system and the original one are identical in \( z^{[s-1]} \).

We utilize the hybrid maximum principle \([44]\) to derive necessary conditions for the optimal solution.

**Theorem 1 (Hybrid Maximum Principle):** Let \( t_0 < \ldots < t_M \) be real numbers and \( \Delta_k = [t_{k-1}, t_k] \). For any collection of absolute continuous functions \( x_k : \Delta_k \mapsto \mathbb{R}^{n_k} \), define a vector, \( \Sigma_k \in \mathbb{R}^n \), where \( n = 2 \sum_{k=1}^{M} n_k \), as

\[
\Sigma_k = (x_1^T(t_0), x_1^T(t_1), \ldots, x_M^T(t_{M-1}), x_M^T(t_M))^T. \quad (26)
\]

On the time interval \( \Delta = [0, t_M] \), consider the problem

\[
\min_{u_k, x_k} J(x_{\Sigma}) \quad (27a)
\]

s.t. \( \dot{x}_k(t) = f_k(x_k(t), u_k(t)) \quad (27b) \)

\[
u_k(t) \in U_k \subseteq \mathbb{R}^{r_k} \quad (27c)
\]

\[
\forall t \in \Delta_k, \quad k = 1, \ldots, M. \quad (27d)
\]

\[
\eta(x_{\Sigma}) = 0 \quad (27e)
\]

where \( f_k : \mathbb{R}^{n_k} \times \mathbb{R}^{r_k} \mapsto \mathbb{R}^{n_k}, J : \mathbb{R}^n \mapsto \mathbb{R}, \) and \( \eta : \mathbb{R}^n \mapsto \mathbb{R}^q \) are continuously differentiable, and \( u_k : \mathbb{R} \mapsto \mathbb{R}^{r_k} \) are measurable and bounded on the corresponding \( \Delta_k \).

Denote an optimal process for (27) by \( (x^*(t), u^*(t)) \). Then, there exists a collection \( (\alpha, \gamma, \psi_1, \ldots, \psi_M) \), where \( \alpha \geq 0, \gamma \in \mathbb{R}^q \) and \( \psi_k : \Delta_k \mapsto \mathbb{R}^{n_k} \) are Lipschitz continuous. It generates \( M \) Pontryagin functions

\[
H_k(\psi_k, x_k, u_k) = \psi_k^T f_k(x_k, u_k), \quad t \in \Delta_k \quad (28)
\]

and a Lagrange function \( L(x_{\Sigma}) = \alpha J(x_{\Sigma}) + \gamma^T \eta(x_{\Sigma}) \). The following conditions are satisfied for all \( k = 1, \ldots, M \).

1) Nontriviality condition:

\[
(\alpha, \gamma^T) \neq 0. \quad (29)
\]

2) Adjoint equations: for almost all \( t \in \Delta_k \),

\[
\dot{\psi}_k(t) = -\frac{\partial H_k}{\partial x_k}(\psi_k(t), x^*_k(t), u^*_k(t)). \quad (30)
\]

3) Transversality conditions:

\[
\begin{cases}
\psi_k(t_{k-1}) = L_{x_k(t_{k-1})}(x^*_k) \\
\psi_k(t_k) = -L_{x_k(t_k)}(x^*_k). \quad (31)
\end{cases}
\]
4) Maximality conditions: for all $t \in \Delta_k$,
\[
H_k(\psi_k(t), x_k^*(t), u_k(t)) = \sup_{u_k \in U_k} H_k(\psi_k(t), x_k^*(t), u_k) = 0.
\]

**Proof:** The proof can be directly adapted from [44, Th. 4]. Here, we only consider each system $f_k$ to be time invariant and all intervals $\Delta_k$ to be fixed. Besides, no inequality constraints are specified on $x_\Sigma$.

According to Theorem 1, the costate $\psi_i : T_i \mapsto \mathbb{R}^{m_{i,s}+1}$ in the $i$th stage is defined as
\[
\psi_i = \left( \lambda_i, \mu_i \right) = \left( \lambda_i^T, \lambda_i^T, \ldots, \lambda_i^T, \mu_i^T \right)^T
\]
where $\lambda_i : \Delta_i \mapsto \mathbb{R}$ and $\mu_i : \Delta_i \mapsto \mathbb{R}^m$ is the $j$th map in $\lambda_i$, $\lambda_i$. The $i$th Pontryagin function of (20) is
\[
H_i(\psi_i, y_i, v_i) = \psi_i^T \dot{f}(y_i, v_i) + \lambda_i^T \dot{A}_i \lambda_i + \mu_i^T W_i v_i.
\]

By applying the adjoint equation [30] for $\mu_i$, we have $\mu_i = 0$, which means $\mu_i(t) = \bar{\mu}_i \in \mathbb{R}$ is a constant in $\Delta_i$. Therefore, $H_i$ is always a quadratic function of $v_i$.

\[
H_i(\psi_i, y_i, v_i) = \chi_i^T \dot{A}_i \lambda_i + \mu_i^T W_i v_i.
\]

By applying the adjoint equation for $\lambda_i$, we obtain
\[
\dot{\lambda}_i = \bar{A}^T \lambda_i
\]

which is expanded as
\[
\dot{\lambda}_i^j = \begin{cases} 0, & \text{if } j = 1 \\ -\lambda_i^j, & \text{if } 2 \leq j \leq s \end{cases}
\]

It is obvious that $\lambda_i^j(t) = \lambda_i^j(s - 1)$ is an $(s - 1)$-degree polynomial.

According to maximality conditions (32), the supremum of $H_i$ is always 0 in $\Delta_i$. Thus, the positive definiteness of $W$ implies $\bar{\mu}_i \leq 0$. If $\bar{\mu}_i = 0$, then (35) becomes a linear function of $v_i$. The zero supremum means that $\lambda_i^j(t) = 0$ in $\Delta_i$. As the result of (36), $\psi_i(t) = 0$ holds for all $t \in \Delta_i$. In such a case, a contradiction occurs for the nontriviality condition (29) and the transversality conditions (31) cannot be satisfied at the same time. Therefore, $\mu_i = 0$ always holds in the whole $\Delta_i$. The optimal control $v_i^* = W_i^{-1} \lambda_i$ can be obtained from
\[
\frac{\partial H_i}{\partial v_i}(\psi_i, y_i, v_i) = \lambda_i + 2\bar{\mu}_i W_i v_i = 0
\]
i.e.,
\[
v_i^*(t) = -\frac{1}{2\bar{\mu}_i} W_i^{-1} \lambda_i^j(t), \quad \forall t \in \Delta_i.
\]

Then, $z_i$ produced by a chain of $s$-integrators from $\lambda_i^j(t)$, is a $(s - 1)$-degree polynomial.

To further explore the structures of the solution, we generate the Lagrange function using the cost of augmented system along with all constraints in (23)–(25). We have
\[
L(y_{\Sigma}) = \alpha \sum_{i=1}^{M} \bar{y}_i(t_i) + \sum_{i=0}^{M-1} \gamma_i \bar{y}_{i+1}(t_i)
\]
\[
+ \sum_{i=1}^{M-1} \left( \gamma_i^T, \sigma_i^T \right) \left( z_{i+1}(t_i) - z_{i+1}(t_{i+1}) \right)
\]
\[
+ \theta_i^T \left( z_{i+1}(t_0) - \bar{z}_0 \right) + \theta_f^T \left( z_{M+1}(t_M) - \bar{z}_f \right)
\]
\[
+ \sum_{i=1}^{M-1} \theta_i^T \left( z_{i+1}^d(t_i) - z_i \right)
\]
where $\gamma_i \in \mathbb{R}$, $\zeta_i \in \mathbb{R}^m$, $\sigma_i \in \mathbb{R}^m$, $\theta_0 \in \mathbb{R}^m$, $\theta_f \in \mathbb{R}^m$, and $\theta_i \in \mathbb{R}^m$ is a constant coefficient of corresponding constraints, and $y_{\Sigma}$ is defined as in (26). Following transversality conditions (31), taking the derivative of $L$ w.r.t. $y_{\Sigma}$ gives the boundary values of costates $\psi_i$ and $\psi_{i+1}$, i.e.,
\[
\lambda_i(t_i) = -\left( \zeta_i + \theta_i \right), \lambda_{i+1}(t_i) = -\left( \zeta_i / \sigma_i \right)
\]
Because $\mu_{i+1}(t_i) = \bar{\mu}_{i+1} + \Delta_i$, we have
\[
\bar{\mu}_i = -\alpha, \quad 1 \leq i \leq M.
\]

Finally, by substituting (36), (41), and (43) into (39), we obtain that the optimal controls of two consecutive stages satisfy
\[
v_{i+1}^*(t) = v_{i+2}^*(t-1), \quad 0 < j < (s - d_i).
\]

We finally know that the optimal control of the problem (18) is actually $s - d_i$ – times continuously differentiable at the timestamp $t_i$. Accordingly, the optimal state trajectory, consisting of $M$ polynomials with $2s - 1$ degree, is the continuous differentiable at $t_i$. Now, we conclude the conditions derived from both (39) and (44) in the following theorem, which are proved to be necessary and sufficient optimality conditions of (18).

**Theorem 2 (Optimality Conditions):** A trajectory, denoted by $z^*(t) : [t_0, t_M] \mapsto \mathbb{R}^m$, is optimal for the problem (18) if and only if the following conditions are satisfied.

1) The map $z^*(t) : [t_{i-1}, t_i] \mapsto \mathbb{R}^m$ is parameterized as a $(2s - 1)$-degree polynomial for any $1 \leq i \leq M$.

2) The boundary conditions in (18c).

3) The intermediate conditions in (18d).

4) $z^*(t) = d_i - 1$ times continuously differentiable at $t_i$ for any $1 \leq i < M$ where $d_i = 2s - d_i$.

Moreover, a unique trajectory exists for these conditions.

**Proof sketch:** The proof of necessity is evident in the analyses from (33) to (44) that are directly derived from Theorem 1. The proof of sufficiency is sketched as follows: 1) the first and fourth conditions always determine a linear spline space of dimension $2s + \sum_{i=1}^{M-1} d_i$ for any sequence of $d_i$; 2) the second and third conditions form a square coefficient matrix on a basis of the spline space, implying the existence of solution; 3) the matrix is proved to be nonsingular since $t_{i-1} < t_i$ for each $i$, giving the uniqueness of solution; and 4) the existence and uniqueness for the necessary conditions yield their sufficiency. This proof of sufficiency is detailed in Appendix A.
To further explain the optimality conditions, we take the multistage jerk minimization as an example. In this example, the position, velocity, and acceleration are states of the jerk-controlled system \((s = 3)\). There are intermediate points \((d_i = 1)\) that the trajectory should pass through at certain timestamps. The continuity of state only requires the continuity up to acceleration of the minimum-jerk trajectory, while jerk and snap of the optimal trajectory are also continuous everywhere. Accordingly, if we enforce all these continuity conditions, then Theorem 2 guarantees that only one trajectory exists, which is exactly the optimal one.

C. Minimization Without Cost Functional

Theorem 2 provides a direct way to construct the unique optimal trajectory. The computation enjoys linear complexity in time and space. It does not even require explicit or implicit evaluation of the cost functional or its gradient.

Consider an \(m\)-dimensional trajectory whose \(i\)th piece is denoted by an \((N = 2s - 1)\)-degree polynomial:

\[
p_i(t) = c_i^T \beta(t - t_{i-1}), \quad t \in [t_{i-1}, t_i]
\]

where \(\beta(x) = (1, x, \ldots, x^N)^T\) is the basis and \(c_i \in \mathbb{R}^{2s \times m}\) are the coefficients. For simplicity, we use the timeline relative to \(t_0 = 0\). The trajectory is described by a coefficient matrix \(c \in \mathbb{R}^{2Ms \times m}\) and a time vector \(T \in \mathbb{R}^{M \times 1}\) defined by

\[
c = (c_1^T, \ldots, c_M^T)^T, \quad T = (T_1, \ldots, T_M)^T
\]

where \(T_i\) means the duration of the \(i\)th piece. Then, we have the time stamp \(t_i = \sum_{j=1}^i T_j\) and the total duration \(T = \|T\|_1\). The \(M\)-piece trajectory \(p : [0, T] \mapsto \mathbb{R}^m\) is defined by

\[
p(t) = p_i(t) \quad \forall t \in [t_{i-1}, t_i) \forall i \in \{1, \ldots, M\}.
\]

To compute the unique solution for (18), we directly enforce optimality conditions on the coefficient matrix \(c\). Denote by \(D_0, D_M \in \mathbb{R}^{s \times m}\) and \(D_i \in \mathbb{R}^{d_i \times m}\) the specified derivatives at boundaries and intermediate timestamp \(t_i\), respectively. Each column of \(D_i\) is related to one dimension. Then, conditions at \(t_i\) are formulated by \(E_i, F_i \in \mathbb{R}^{2s \times 2s}\)

\[
\begin{pmatrix}
E_i & F_i
\end{pmatrix}
\begin{pmatrix}
c_i \\
c_{i+1}
\end{pmatrix}
= \begin{pmatrix}
D_i \\
0_{d_i \times m}
\end{pmatrix}
\]

\[
E_i = (\beta(T_i), \ldots, \beta(d_i-1)(T_i)),
\]

\[
F_i = (0, -\beta(0), \ldots, -\beta(d_i-1)(0))^T.
\]

Especially, define \(F_0, E_M \in \mathbb{R}^{s \times 2s}\) as

\[
F_0 = (\beta(0), \ldots, \beta(s-1)(0))^T
\]

\[
E_M = (\beta(T_M), \ldots, \beta(s-1)(T_M))^T.
\]

The linear system for the optimal coefficient matrix is

\[
M c = b
\]

where \(M \in \mathbb{R}^{2Ms \times 2Ms}\) and \(b \in \mathbb{R}^{2Ms \times m}\) are

\[
M = \begin{pmatrix}
F_0 & 0 & 0 & \cdots & 0 \\
E_1 & F_1 & 0 & \cdots & 0 \\
& \vdots & \ddots & \ddots & \vdots \\
& 0 & \cdots & 0 & F_{M-1} \\
0 & 0 & \cdots & 0 & E_M
\end{pmatrix}
\]

\[
b = (D_0^T, D_1^T, 0_{m \times d_1}, \ldots, D_{M-1}^T, 0_{m \times d_{M-1}}, D_M^T)^T.
\]

It is essential that the uniqueness in Theorem 2 ensures the nonsingularity of \(M\) for any time vector \(T > 0\). Consequently, the unique solution \(c\) can be obtained via linear equation system (53) with a banded matrix \(M\), i.e., a banded system. As for a nonsingular banded system, its banded PLU factorization always exists [45], which can be employed to compute the solution with \(O(M)\) time and space complexity [46]. Therefore, without the need of cost functional, the unique solution of problem (18) is obtained in the lowest complexity, by directly applying our optimality conditions.

D. MINCO Trajectories With Spatial–Temporal Deformation

For multicopters, there are often task-specific requirements apart from feasibility, such as the perception quality in active simultaneous localization and mapping [47] or the occlusion rate in aerial videography [48]. These user-defined requirements majorly need to flexibly and adaptively deform both the spatial and temporal profiles of a trajectory. Therefore, we select the intermediate points and the time vector as two salient parameters in (18). Fortunately, the existence and uniqueness of solution guarantee the smoothness of sensitivity for them. An iterative procedure is then designed to conduct the spatial–temporal deformation with the lowest computation complexity per iteration.

We denote the intermediate points by \(q = (q_1, \ldots, q_{M-1})\), where \(q_i \in \mathbb{R}^m\) is a specified zero-order derivative at \(t_i\). Denote by \(T = (T_1, \ldots, T_M)^T\) the time vector where \(T_i \in \mathbb{R}_{>0}\). For any pair of \(q\) and \(T\), Theorem 2 naturally determines a trajectory belonging to a class of control effort minimizers, named MINCO hereafter. The MINCO trajectory class, denoted by \(\mathcal{T}_{\text{MINCO}}\), is defined as

\[
\mathcal{T}_{\text{MINCO}} = \left\{ p(t) : [0, T] \mapsto \mathbb{R}^m \bigg| c = c(q, T) \text{ determined by Theorem2}, \forall q \in \mathbb{R}^{m \times (M-1)}, T \in \mathbb{R}^M \right\}.
\]

The dimension \(m\), the system order \(s\), and initial and terminal conditions are omitted here for brevity. Intuitively, all the trajectories in \(\mathcal{T}_{\text{MINCO}}\) are compactly parameterized by only \(q\) and \(T\). Evaluating an element in \(\mathcal{T}_{\text{MINCO}}\) directly follows our linear-complexity formulation.

We denote any user-defined objective (or constraint) on a trajectory by a \(C^2\) function \(K(c, T)\) with available gradient. This objective on \(\mathcal{T}_{\text{MINCO}}\) can be computed as

\[
\mathcal{V}(q, T) = K(c(q, T), T).
\]

To accomplish deformation of \(\mathcal{T}_{\text{MINCO}}\), the function \(\mathcal{V}\) together with its gradient \(\partial \mathcal{V}/\partial q\) and \(\partial \mathcal{V}/\partial T\) is needed for a high-level optimizer to optimize the objective. Obviously, evaluating
\( W \) shares the same complexity as evaluating any trajectory in \( \mathcal{Z}_{\text{MINCO}} \). The key procedure is to compute the gradient. Now, we give a linear-complexity scheme to compute \( \partial W / \partial q \) and \( \partial W / \partial T \) from the given \( \partial K / \partial c \) and \( \partial K / \partial T \). We first rewrite the linear equation system (53) as

\[
M(T)c(q, T) = b(q). \tag{57}
\]

Without causing ambiguity, we omit parameters in \( M, b, c, K, \) and \( W \) temporarily for simplicity. Any notation involving \( c \) is interpreted as \( c(q, T) \). Denote by \( q_{i,j} \) the \( j \)th entry in \( q_i \).

As for the gradient w.r.t. \( q \), we first differentiate both sides of (57) w.r.t. \( q_{i,j} \), which gives

\[
\frac{\partial c}{\partial q_{i,j}} = M^{-1} \frac{\partial b}{\partial q_{i,j}}, \tag{58}
\]

Then

\[
\frac{\partial W}{\partial q_{i,j}} = \text{Tr} \left\{ \left( \frac{\partial c}{\partial q_{i,j}} \right)^T \frac{\partial K}{\partial c} \right\} = \text{Tr} \left\{ \left( M^{-1} \frac{\partial b}{\partial q_{i,j}} \right)^T \frac{\partial K}{\partial c} \right\} = \text{Tr} \left\{ \left( \frac{\partial b}{\partial q_{i,j}} \right)^T \left( M^{-1} \frac{\partial K}{\partial c} \right) \right\} \tag{59}
\]

where \( \text{Tr}(\cdot) \) is the trace operation. The definition of \( b(q) \) in (55) implies that \( \partial b / \partial q_{i,j} \) only has a single nonzero entry 1 at its \((2i-1)s + j\) row and \( j \) column. Thus, stacking all the resultant scalars gives

\[
\frac{\partial W}{\partial q_i} = \left( M^{-1} \frac{\partial K}{\partial c} \right)^T e_{(2i-1)s+1} \tag{60}
\]

where \( e_j \) is the \( j \)th column of \( I_{2Ms} \). Now that we have already conducted the banded PLU factorization for \( M \) when we compute \( c \), we can reuse the factorization to avoid inverting \( M^T \).

Define a matrix \( G \in \mathbb{R}^{2Ms \times Ms} \) as

\[
M^T G = \frac{\partial K}{\partial c}. \tag{61}
\]

We only need to compute \( G \) once to obtain \( \partial W / \partial q_i \) for all \( 1 \leq i < M \). Denote the factorization of \( M \) as \( M = PLU \).

Specifically, \( L \) is a banded matrix with zero upper bandwidth and all-ones diagonal entries. \( U \) is a banded matrix with zero lower bandwidth and nonzero diagonal entries because of the nonsingularity of \( M \). The pivoting matrix \( P \) simply changes the row order of the operand, satisfying \( P^T P = I \). Consequently, the transpose also has a \textit{banded LUP factorization} [45]. Specifically, \( M^T = LUP^T \), where

\[
\tilde{L} = U^T (U \circ I)^{-1}, \quad \tilde{U} = (I \circ U) L^T \tag{62}
\]

where the inverse is only done for a diagonal matrix and \( \circ \) is the Hadamard product. Then, \( G \) can also be computed in linear complexity through such a factorization. For convenience, we partition \( G \) into

\[
G = (G_0^T, G_1^T, \ldots, G_{M-1}^T, G_M^T)^T \tag{63}
\]

such that \( G_0, G_M \in \mathbb{R}^{s \times m} \) and \( G_i \in \mathbb{R}^{2s \times m} \) for \( 1 \leq i < M \). After that, the gradient of \( W \) w.r.t. \( q \) is determined as

\[
\frac{\partial W}{\partial q} = \left( G_1^T e_1, \ldots, G_{M-1}^T e_1 \right) \tag{64}
\]

where \( e_1 \) is the first column of \( I_{2s} \). This operation takes out \( M-1 \) specific columns in \( G^T \).

As for the gradient w.r.t. \( T \), differentiating both sides of (57) w.r.t. \( T_i \) gives

\[
\frac{\partial M}{\partial T_i} c + M \frac{\partial c}{\partial T_i} = 0. \tag{65}
\]

Thus

\[
\frac{\partial W}{\partial T_i} = \frac{\partial K}{\partial T_i} - \text{Tr} \left\{ \left( \frac{\partial M}{\partial T_i} c \right)^T M^{-T} \frac{\partial K}{\partial c} \right\} = \frac{\partial K}{\partial T_i} - \text{Tr} \left\{ G_i^T \frac{\partial M}{\partial T_i} c \right\}. \tag{66}
\]

The banded structure of \( M \) implies that

\[
G_i^T \frac{\partial M}{\partial T_i} c = G_i^T \frac{\partial E_i}{\partial T_i} e_i. \tag{67}
\]

Then, we obtain the gradient w.r.t. \( T_i \), computed as

\[
\frac{\partial W}{\partial T_i} = \frac{\partial K}{\partial T_i} - \text{Tr} \left\{ G_i^T \frac{\partial E_i}{\partial T_i} e_i \right\} \tag{68}
\]

where \( \frac{\partial E_i}{\partial T_i} \) can be analytically derived from (49). Computing (68) for every \( 1 \leq i \leq M \) gives \( \partial W / \partial T \).

Finally, we finish the computation of \( \partial W / \partial q \) and \( \partial W / \partial T \). It can be verified from both (64) and (68) that the gradient propagation is also done in \( O(M) \) complexity. As for \( K \), we make no assumption on its concrete form. Actually, the smoothness of \( K \) is not even needed if only the resultant \( W \) is \( C^2 \). In other words, the linear-complexity gradient propagation enjoys both efficiency and flexibility. By incorporating it into common optimizers, we can accomplish the spatial–temporal deformation of \( \mathcal{Z}_{\text{MINCO}} \) for a wide range of planning purposes while maintaining the local smoothness of trajectories.

V. GEOMETRICALLY CONSTRAINED FLIGHT TRAJECTORY OPTIMIZATION

In this section, we provide a unified framework for flight trajectory optimization with different time regularization \( \rho(T) \), spatial constraints \( F \), and continuous-time constraints \( G \). This framework indeed relaxes the original problem into \( \mathcal{Z}_{\text{MINCO}} \). The spatial–temporal deformation is utilized to meet various feasibility requirements. Lightweight schemes are especially designed to eliminate geometrical constraints such that the trajectory can be freely deformed. For continuous-time constraints, a time integral penalty functional is proposed to ensure the feasibility without sacrificing the scalability. Finally, our framework transforms the constrained trajectory optimization into a sparse unconstrained one, which can be reliably solved.
Optimizing $\tau$ requires gradient propagation. We partition the gradient as $\partial J_{q}/\partial \tau = (g_{a}^{T}, g_{b})^{T}$, where $g_{a}, g_{b} \in \mathbb{R}^{M-1}$ and $g_{a}, g_{b} \in \mathbb{R}$. Differentiating the layer in (71) yields the gradient of $J$ w.r.t. $\tau$

$$\frac{\partial J}{\partial \tau} = (g_{a} - g_{b}1) e^{\tau} (1 + \|e^{\tau}\|_{1}) - \frac{(g_{a}^{T} e^{\tau} - g_{b}^{T} e^{\tau}^{2})}{(1 + \|e^{\tau}\|_{1})^{2}}$$  

(72)

where $e^{[1]}$ is the entrywise exponential map and 1 is an all-ones vector. If an initial guess $T$ is specified, the corresponding $\tau$ can be computed via the inverse map of the diffeomorphism, given by $\tau_{i} = \ln (T_{i}/T_{M})$ for $1 \leq i < M$. As for $\rho_{s}$ in (12), only $T_{i} > 0$ needs to be ensured. It suffices to use $T = e^{\tau}$ as the diffeomorphism between $\mathbb{R}^{M}$ and $\mathbb{R}_{>0}^{M}$.

For either $\rho_{f}$ or $\rho_{s}$, we denote the diffeomorphism by $T(\tau)$. Unconstrained optimization on $\tau$ can be directly conducted to minimize $J(q, T(\tau))$. Although $T(\tau)$ does not preserve convexity, the original cost $J(q, T)$ is already nonconvex as given in (57). Thus, the only concern is whether $T(\tau)$ brings new local minima in the space of $T$ or eliminates local minima in the space of $T$.

**Proposition 2:** Denote by $F : \mathbb{D}_{F} \mapsto \mathbb{R}$ any $C^{2}$ function with a convex open domain $\mathbb{D}_{F} \subseteq \mathbb{R}^{N}$. Given any $C^{2}$ diffeomorphism $G : \mathbb{R}^{N} \mapsto \mathbb{D}_{F}$, define $H : \mathbb{R}^{N} \mapsto \mathbb{R}$ as $H(y) = F(G(y))$ for $y \in \mathbb{R}^{N}$. For any $x \in \mathbb{D}_{F}$ and $y \in \mathbb{R}^{N}$ satisfying $x = G(y)$ or equivalently $y = G^{-1}(x)$, the following statements always hold:

1. $\nabla F(x) = 0$ if and only if $\nabla H(y) = 0$.
2. $\nabla^{2} F(x)$ is positive-definite (or positive-semidefinite) at $\nabla F(x) = 0$, if and only if $\nabla^{2} H(y)$ is positive-definite (or positive-semidefinite) at $\nabla H(y) = 0$.

**Proof:** See Appendix B.

Proposition 2 confirms that $T(\tau)$ preserves the first/second-order necessary optimality conditions and second-order sufficient optimality conditions [49]. It is also applicable to substitute the exponential map in this subsection with any $C^{2}$ diffeomorphism from $\mathbb{R}$ to $\mathbb{R}_{>0}$ for a better numerical condition. In the sense of commonly used optimality conditions, our constraint elimination does not produce extra spurious local minima or cancel any existing one.

**B. Spatial Constraint Elimination**

We enforce motion safety by confining trajectories into the feasible region $\mathcal{F}$. Although $\mathcal{F}$ is nonconvex, it is a union of convex primitives that are sequentially connected. If all pieces have been assigned into these primitives, the safety constraint on each piece becomes convex and, thus, can be conveniently encoded in $\mathcal{G}$. Owing to the feature of MINCO, the traverse time for every primitive can be directly optimized. Thus, we fix the piece assignment before optimization, rather than resorting to integer variables during optimization [33]. Consequently, intermediate points should be contained by the overlap between primitives, forming inequalities. For inequality constrained problems (ICPs), general methods successively approximate the constraints via additional parameters. However, we aim to apply the constraints directly and efficiently. Therefore, we propose spatial constraint elimination to enforce them exactly, leveraging their geometrical properties.

Consider the constraint $q \in \mathcal{P} \subset \mathbb{R}^{n}$, where $\mathcal{P}$ is a closed ball. Its dimension satisfies $n \leq m$ since a low-dimensional constraint also exists in $\mathbb{R}^{m}$. If $\mathcal{P}$ is a closed ball $\mathcal{P}_{\mathcal{B}}$ centered
at point $o$ with radius $r$

$$P^B = \left\{ x \in \mathbb{R}^n \mid \|x - o\|_2 \leq r \right\}. \quad (73)$$

We utilize a smooth surjection to map $\mathbb{R}^n$ to $P^B$ such that optimization over $\mathbb{R}^n$ implicitly satisfies the constraint $P^B$. As illustrated in Fig. 3, the map is a composition of the inverse stereographic projection and the orthographic projection. First, we utilize the inverse stereographic projection to map $\mathbb{R}^n$ to $S^n_{\odot}$, where $S^n_{\odot}$ is a unit sphere without north pole, i.e.,

$$S^n_{\odot} = \left\{ x \in \mathbb{R}^{n+1} \mid \|x\|_2 = 1, x_{n+1} < 1 \right\}. \quad (74)$$

The inverse stereographic projection $f_s$ is defined as

$$f_s(x) = \frac{(2x^T, x^T x - 1)^T}{x^T x + 1} \in S^n_{\odot} \quad \forall x \in \mathbb{R}^n. \quad (75)$$

Note that $f_s$ is a diffeomorphism between $\mathbb{R}^n$ and $S^n_{\odot}$ [50]. We then project $S^n_{\odot}$ back in $\mathbb{R}^n$ to obtain

$$B^n = \left\{ x \in \mathbb{R}^n \mid \|x\|_2 \leq 1 \right\}. \quad (76)$$

The map is described by

$$f_o(x) = (x_1, \ldots, x_n)^T \in B^n \quad \forall x \in S^n_{\odot} \quad (77)$$

which is indeed a smooth surjection onto $B^n$. Each point in $B^n$, except the center, is paired with two points in $S^n_{\odot}$. The composition of $f_s$, $f_o$, and a linear transformation, is a smooth surjection

$$f_B(x) = o + \frac{2rx}{x^T x + 1} \in P^B \quad \forall x \in \mathbb{R}^n. \quad (78)$$

The map $f_B$ introduces a new coordinate, denoted by $\xi$, such that optimizing $\xi$ over $\mathbb{R}^n$ always satisfies the constraint on $q$ described by $P^B$. For the ith intermediate point $q_i$, denote by $\xi_i$ the corresponding new coordinate. Accordingly, denote by $\xi$ the new coordinate for $q$. Optimizing $\xi$ requires gradient propagation for $\partial J / \partial q_i$. Denote by $q_i$ the ith entry $\partial J / \partial q_i$ in $\partial J / \partial q$. Differentiating the layer $f_B$ gives the gradient

$$\frac{\partial J}{\partial \xi_i} = \frac{2r_i q_i}{\xi_i^2 + 1} - \frac{4r_i (\xi_i^2 q_i) \xi_i}{(\xi_i^2 + 1)^2}. \quad (79)$$

If the optimization needs to start from an initial guess $q$, the backward evaluation of $\xi$ can be done by using a local inverse of $f_B$, given by $\xi_i$ for $1 \leq i < M$ as follows:

$$\xi_i = \frac{r_i - \sqrt{r_i^2 - \|q_i - o_i\|_2^2}}{\|q_i - o_i\|_2^2} (q_i - o_i). \quad (80)$$

Similarly, we analyze influences that the smooth surjection $f_B$ imposes on the constrained local minima in $P^B$. Although $f_B$ lacks the one-to-one correspondence as diffeomorphisms possess, its components are all well-formed. First, $f_o$ only takes the first $n$ entries of a point. This operation preserves at least the first-order necessary conditions for local minima in either $B^n$ or $S^n_{\odot}$. Second, $f_s$ is a diffeomorphism between $S^n_{\odot}$ and $\mathbb{R}^n$, thus satisfying Proposition 2. Therefore, we can also confirm that $f_B$ does not produce extra spurious local minima or cancel any existing one. As shown in Fig. 4, the constrained minimum within a 2-D ball is transformed into an unconstrained minimum.

C. Polyhedral Spatial Constraint Elimination

Now, we consider the elimination of polyhedral constraints. Specifically, $P$ is a closed convex polytope $P^H$ defined by

$$P^H = \left\{ x \in \mathbb{R}^n \mid Ax \leq b \right\} \quad (81)$$

where $\text{Int}(P^H) \neq \emptyset$ according to (15). Common optimization algorithms use the $H$-representation of $P^H$ as linear inequality constraints. In our framework, we exploit their geometrical property to eliminate these constraints so that $S_{\text{MINCO}}$ can be freely deformed. To achieve this, we use the $V$-representation of $P^H$ instead, where any $q \in P^H$ has a (general) barycentric coordinate, i.e., a convex combination of vertices. To obtain the vertices, we apply the efficient convex hull algorithm [51] to the dual of $P^H$ based on an interior point calculated by Seidel’s algorithm [52]. Note that this procedure produces negligible overhead in our case ($n \leq 4$).

The procedure to eliminate a polytope constraint is illustrated in Fig. 5. We denote all $\hat{n} + 1$ vertices of $P^H$ by $(v_0, \ldots, v_{\hat{n}})$, where $v_i \in \mathbb{R}^n$ for each $i$. The barycentric coordinate of a point $q \in P^H$ consists of the weights for these vertices. To obtain a more compact form, define $\hat{v}_i = v_i - v_0$ and $\hat{V} = (\hat{v}_1, \ldots, \hat{v}_{\hat{n}})$; then, the position can be calculated as

$$q = v_0 + \hat{V} w \quad (82)$$
where \( w = (w_1, \ldots, w_n)^T \in \mathbb{R}^n \) is the last \( n \) entries in the barycentric coordinate. The set of coordinates in convex combinations can also be written as

\[
\mathcal{P}_w^N = \left\{ w \in \mathbb{R}^n \mid w \geq 0, \|w\|_1 \leq 1 \right\}.
\]

Concretely, the inequality constraints are \(-w \preceq 0\). By introducing additional variables \( x \), the equivalent equality constraints are \(-w + |x|^2 = 0\), yielding \( w = |x|^2 \). Such type of constraint conversion is proved to preserve first-/second-order necessary conditions and second-order sufficient conditions for ICPs by Bertsekas [55, Sec. 4.3]. We confirm that the additional nonlinearity in \( f_H \) does not exclude the desired minimum or produce undesired minimum practically.

Direct constraints on \( q \) are eliminated for either \( \mathcal{P}_B \) or \( \mathcal{P}_w^N \) using a smooth surjection \( q(\xi) \). We can conduct unconstrained optimization on \( \xi \) to minimize \( J(q(\xi), \mathbf{T}(\tau)) \) hereafter.

\[
\mathcal{D}. \text{ Time Integral Penalty Functional}
\]

After eliminating direct constraints, \( \mathcal{T}_{\text{MINCO}} \) can be freely deformed to meet the continuous-time constraints \( \mathcal{G} \). However, enforcing \( \mathcal{G} \) over the entire trajectory involves infinitely many inequalities that cannot be solved by constrained optimization. It further needs temporal discretization that usually produces a large number of decision variables. To preserve the sparsity of trajectory parameterization, we decouple the resolution of constraint evaluation from the number of decision variables. Inspired by the constraint transcription [56], we transform \( \mathcal{G} \) into finite constraints by the integral of constraint violations.

For a trajectory \( p : [0, T] \rightarrow \mathbb{R}^m \), we define

\[
I_G^k[p] = \int_0^T \max \{ |\mathcal{G}(p(t), \ldots, p^{(s)}(t)), 0 |^k \} dt
\]

where \( k \in \mathbb{R}_{>0} \) and \( \max[\cdot, 0]^k \) is the composition of the entrywise maximum and an entrywise power function. Specifically, smoothing is needed if \( k \leq 1 \). The functional-type constraint is then equivalent to equality constraints \( I_G^k[p] = 0 \). Actually, \( I_G^k[p] \) is a function of trajectory parameters, which we adopt as penalty terms.

\[
\mathcal{D}. \text{ Time Integral Penalty Functional}
\]

After eliminating direct constraints, \( \mathcal{T}_{\text{MINCO}} \) can be freely deformed to meet the continuous-time constraints \( \mathcal{G} \). However, enforcing \( \mathcal{G} \) over the entire trajectory involves infinitely many inequalities that cannot be solved by constrained optimization. It further needs temporal discretization that usually produces a large number of decision variables. To preserve the sparsity of trajectory parameterization, we decouple the resolution of constraint evaluation from the number of decision variables. Inspired by the constraint transcription [56], we transform \( \mathcal{G} \) into finite constraints by the integral of constraint violations.

For a trajectory \( p : [0, T] \rightarrow \mathbb{R}^m \), we define

\[
I_G^k[p] = \int_0^T \max \{ |\mathcal{G}(p(t), \ldots, p^{(s)}(t)), 0 |^k \} dt
\]

where \( k \in \mathbb{R}_{>0} \) and \( \max[\cdot, 0]^k \) is the composition of the entrywise maximum and an entrywise power function. Specifically, smoothing is needed if \( k \leq 1 \). The functional-type constraint is then equivalent to equality constraints \( I_G^k[p] = 0 \). Actually, \( I_G^k[p] \) is a function of trajectory parameters, which we adopt as penalty terms. If \( k = 1 \), it forms a nonsmooth but exact penalty. If \( k > 1 \), it forms a differentiable strictly convex penalty. Thus, either \( I_G^k[p] \) or a smoothing approximation of \( I_G^k[p] \) can be adopted.

For simplicity, we utilize \( I_G^k[p] \) hereafter unless otherwise specified. There are two reasons for choosing a penalty function method. First, the integral in (87) can only be evaluated numerically, making the constraint approximation inevitable. Second, penalty methods have no requirement on a feasible initial guess, which is nontrivial to construct.

We define the time integral penalty functional for \( p(t) \) as

\[
I_G[p] = \chi^T I_G^k[p]
\]

where \( \chi \in \mathbb{R}^{n_x} \) is a weight vector. Normally, \( \chi \) should contain large constants. If no constraint is violated, \( I_G[p] \) remains zero. Otherwise, if any part on \( p(t) \) violates any constraint in \( \mathcal{G} \), the penalty functional \( I_G[p] \) grows rapidly. By incorporating \( I_G[p] \) into the cost functional, continuous-time constraints are enforced within an acceptable tolerance.

Practically, \( I_G[p] \) can only be evaluated by quadrature. To conduct the quadrature, we first define a sampled function \( \mathcal{G}_r : \mathbb{R}^{2n_x m} \times \mathbb{R}_{>0} \times [0, 1] \rightarrow \mathbb{R}^{n_y} \) as

\[
\mathcal{G}_r(c_r, T_r, \tau) = \mathcal{G} \left( c_r^T \beta(T_r, \tau), \ldots, c_r^T \beta^{(s)}(T_r, \tau) \right)
\]

where \( \tau \in [0, 1] \) is a normalized stamp. Then, the quadrature for \( I_G[p] \), denoted by \( I : \mathbb{R}^{2n_x m} \times \mathbb{R}_{>0} \rightarrow \mathbb{R}_{>0} \), is computed as
a weighted sum of the sampled penalty
\[
I(c, T) = \sum_{i=1}^{M} \frac{T_i}{\kappa_i} \sum_{j=0}^{\kappa_i} \omega_j^T \max \left[ \mathcal{G}_0 \left( c_i, T_i, \frac{j}{\kappa_i} \right) \right], \quad 0^k (90)
\]
where \(\kappa_i\) controls the resolution. We choose the trapezoidal rule is the quadrature of penalty functional (90). Note that any integrands in our practice.

Intuitively, \(I(c, T)\) is a differentiable approximation to \(I_0[p]\), whose precision is adjustable through \(\kappa_i\). The value and gradient at most timestamps can be parallely computed then directly combined as one.

E. Trajectory Optimization via Unconstrained NLP

Due to \(\mathcal{G}\) and \(\mathcal{F}\) in (11), the optimal trajectory parameterization is generally hard to know. Unlike traditional methods approximating solutions via a large number of variables [23], we propose to solve a lightweight relaxed optimization via unconstrained NLP, where the spatial–temporal deformation of \(\bar{\mathcal{S}}_{\text{MINCO}}\) is applied. The relaxation to (11) is defined as
\[
\min_{\xi, \tau} J(q(\xi), T(\tau)) + I(c(q(\xi), T(\tau)), T(\tau)) \quad (91)
\]
where \(J\) is the time-regularized control effort (69) for \(\bar{\mathcal{S}}_{\text{MINCO}}\) and \(I\) is the quadrature of penalty functional (90). Note that any task-specific requirement, either objectives or constraints, can be combined in (91) without affecting its structure.

To generate trajectories for a flat multicopter, we first parameterize its flat-output trajectory as \(\bar{\mathcal{S}}_{\text{MINCO}}\). After assigning a fixed number of pieces into each \(\mathcal{P}_i\), variable transformations are applied to eliminate all direct constraints. User-defined \(\mathcal{G}_n\) are also transformed into \(\mathcal{G}\) via \(\Psi_x\) and \(\Psi_u\). Finally, we obtain the cost function (91). Apparently, the gradient propagation is derived for all layers except \(\Psi_x\) and \(\Psi_u\). One can either apply automatic differentiation (AD) [57] to \(\Psi_x\) and \(\Psi_u\) or derive the gradient propagation analytically by following the reverse-mode AD. The efficiency is the same as the flatness map as ensured by the Baur–Strassen theorem [58]. The differentiation is only needed for the given flat dynamics once and for all. With available gradient, the relaxation (91) is then solved by the limited-memory Broyden–Fletcher–Goldfarb–Shanno algorithm [59].

VI. APPLICATIONS

A. Large-Scale Unconstrained Control Effort Minimization

We benchmark several existing schemes over problem (18), including the QP formulation by Mellinger and Kumar [11], the closed-form solution by Bry et al. [28], and the linear-complexity scheme by Burke et al. [60]. We implement all these schemes in C++11 without any explicit hardware acceleration. Mellinger’s scheme is implemented using OSQP [61]. Bry’s solution is evaluated by both a dense solver and a sparse one [62]. Burke’s scheme is reimplemented here for fairness, which is faster than the original one [60]. The benchmark is conducted on an Intel Core i7-8700 CPU under Linux.

The performance is reported in Fig. 6. Both jerk \(s = 3\) and snap \(s = 4\) are minimized, as defined in (18). Mellinger’s scheme [11] only performs better than the dense evaluation of Bry’s closed-form solution [28] on middle-scale problems \((10^3 < M < 10^5)\). Burke’s scheme [60] benefits from its linear complexity; thus, it can solve large-scale problems \((10^4 < M < 10^6)\). Our scheme improves the computation speed by orders of magnitude against the others at any problem scale while retaining \(O(M)\) complexity.

In conclusion, our optimality conditions provide a practical way to directly construct the solution of problem (18), which possesses simplicity, efficiency, stability, and scalability. The trajectory class \(\bar{\mathcal{S}}_{\text{MINCO}}\) can serve as a reliable submodule of our optimization framework.

B. Trajectory Generation Within Safe Flight Corridors

As a special case of the problem (11), trajectory generation within 3-D safe flight corridors (SFCs) has been widely adopted in real-world applications, such as [31], [63], and [64]. The SFCs are usually generated by the front end of a trajectory planning framework as an abstraction of the concerned configuration space, such as the parallel convex cluster inflation [31], the regional inflation by line search [38], the safe-region RRT* expansion [64], or the iterative regional inflation by semidefinite programming [65]. We assume that an SFC, either polyhedron-shaped or ball-shaped, is already obtained here as in (14) and (15). Optimizing dynamically feasible trajectories within SFCs is usually taken as a back end of such kind of frameworks.

As is illustrated in Fig. 7, we consider two kinds of SFCs. Each convex primitive is assigned with \(K\) trajectory pieces; thus, \(M = M_F K\). The \(i\)th trajectory piece \(p_i(t) : [0, T_i] \rightarrow \mathbb{R}^3\) is assigned to \(\mathcal{P}_{[i/K]}\). The intermediate point assignment of
are automatically satisfied, such as $\rho \leq \rho^*_f$ for all $i$. Proposed methods assign their total time using trapezoidal velocity profiles. Patterson* and Deits* suffer from combinatorial explosion, but they are faster than Patterson* on small-scale problems. Methods not supporting time or interval optimization consume less computation time at the sacrifice of quality.

2) Patterson* [24]: The LQMT problem of a jerk-controlled system is solved using Gauss pseudospectral method. Each trajectory phase is confined within one polytope. Dynamic limits are enforced through path constraints.

3) Gao* [31]: A geometrical curve is optimized via QP formed by jerk energy cost and linear safety constraints on control points of Bézier curves. Its temporal profile is then optimized by an SOCP for TOPP under (92).

4) Deits* [30]: The jerk energy and interval allocation of a trajectory is optimized by an MISOCP. Safety constraints and dynamic limits on the $L_1$-norm of trajectory derivatives are exactly enforced through SOS conditions. Each trajectory piece is a three-degree polynomial.

5) Deits: Details are the same as Deits* except that intervals are allocated heuristically. No integer variable exists.

6) Tordesillas* [33]: Details are the same as Deits* except that safety is ensured by linear constraints on control points of Bézier curves. An mixed-integer QP is solved instead. The total time is determined by a well-designed algorithm.

7) Mellinger [11]: A trajectory is optimized in a QP formed by quadratic cost on jerk and linear safety constraints on sampled points. Its time allocation is generated with trapezoidal velocity profiles. Dynamic limits in (92) are enforced by time scaling [38].

8) Sun* [35]: A trajectory is optimized in a bilevel framework. The low-level QP is exact the same as Tordesillas* except that six-degree polynomials are used. Its time allocation is optimized in the upper level optimization using analytical sensitivity of the low-level one.

A method is asterisked if it supports optimizing time allocation or interval allocation. Dynamic limits are treated as the same for either $L_1$ or $L_2$-norm. Thus, constraints are indeed much tighter on methods from the Proposed*, Gao*, Mellinger, and Patterson*, which restrict the $L_2$-norm of derivatives. As for total time, Deits* and Sun* need preassigned values; thus, we set their total time using trapezoidal velocity profiles. Patterson* handles the original problem directly, taking advantage of the exponential convergence of global collocation [24]. Therefore, we take its trajectory as the ground truth.

The benchmark is conducted in randomly generated environments, one of which is shown in Fig. 10. The corridor size $M_P$...
ranges from 2 to 64, where ten SFCs are generated for each size. The facet number of $P_i^H$ ranges from 8 to 30. We set $K = 1$, $K_P = 1024.0$, $v_{\text{max}} = 5.0$ m/s, $a_{\text{max}} = 7.0$ m/s², $\kappa_i = 16$, the timeout as 3 min, and the relative tolerance as $10^{-4}$. Static boundary conditions are assumed. As for programs, methods from the Proposed* and Mellinger are both implemented in C++11 with a single thread for sequential computing. The general-purpose solver [24] is directly adopted for Patterson*. A C++11 implementation of the original MATLAB one [30] is adopted for both Deits* and Deits. Methods from Gao*, Tordesillas*, and Sun* are taken from their open-source implementations. Besides, the commercial solver Gurobi [66] is used by Deits* and Tordesillas* with six threads enabled for parallel computing. The commercial solver MOSEK [67] is used by both Gao* and Sun*.

The computation efficiency is provided in Fig. 9. Clearly, Deits* and Tordesillas* have to optimize integer variables, thus possessing approximately exponential complexity as $M_P$ grows. Nonetheless, Tordesillas* achieves acceptable performance for small $M_P$ by using a more conservative but easier constraints than Deits*. Methods from Deits and Mellinger achieve satisfactory performance by tackling time allocation or interval allocation heuristically. Methods from Gao* and Sun* perform well in their scalability, while the overhead for small $M_P$ does not suit real-time applications. The method from Patterson* suits offline scenarios, where computation time is far less important than solution quality. The Proposed* method improves the speed by more than an order of magnitude, while retaining optimization on time allocation.

The geometrical profile of trajectories is provided in Fig. 10. Methods that do not optimize time or interval allocation are more likely to deviate from the ground truth. Trajectories by Deits* and Tordesillas* also deviate a lot from the ground truth because of the limited resolution of intervals. The success rates, relative control effort, and flight durations are all given in Fig. 11. Interval-allocation-based methods have relatively low success rates. All the control efforts are normalized by that of the Proposed* one, whose total time is fixed accordingly for fairness. Clearly, heuristic time or interval allocation causes relatively high control effort. Besides, the flight duration from the Proposed* method is the closest to the ground truth.

To explore the temporal profile, we also test four complete methods in a long-distance flight, as shown in Fig. 12. The trajectory from Gao* is less aggressive than the others. The trajectory from Tordesillas* has discontinuous jerk since three-degree polynomials are used. The results from the Proposed* one have nearly the same quality as the ground truth. Profiting from the effectiveness of the penalty functional, our method can also achieve the maximum speed persistently.

In simulations, our method achieves comparable trajectory quality to the collocation-based method [24] in both the geometrical and temporal profiles, while having superior computational speed against all benchmarked ones.
C. SE(3) Motion Planning in Quotient Space

In dense obstacle environments, safe motions often do not exist for narrow spaces unless a multicopter agiley adjusts its attitude to avoid collisions. Therefore, we consider SE(3) motion planning in our framework. An important property for planning in SE(3) as a manifold with structure \( \mathbb{R}^3 \times SO(3) \) is the necessary condition that a feasible pose for a rigid body at least contains a feasible translation for a dimensionless point. The subspace \( \mathbb{R}^3 \) is referred to as a quotient space [69]. Exploiting such a quotient-space decomposition [70], we consider the rotational safety based on a translational trajectory, instead of handling them jointly. Therefore, we can relax assumptions for (14) such that \( \bar{F} \) is just a free region in the quotient space without considering multicopter’s actual size.

We consider simplified quadcopter dynamics whose configuration is defined by its translation \( p \) and rotation \( R \)

\[
\begin{aligned}
\dot{p} &= v \\
\dot{m}v \mathbf{e}_3 &= -m \ddot{g} \mathbf{e}_3 + R \dot{f} \mathbf{e}_3 \\
\dot{R} &= R \dot{\omega}
\end{aligned}
\]

where \( e_i \) is the \( i \)th column of \( I_3 \), \( \ddot{g} \) is the gravitational acceleration, \( \dot{f} \) is the thrust, \( \omega \) is the body rate input, and \( m \) is the vehicle mass. The hat map \( \hat{\cdot} : \mathbb{R}^3 \rightarrow \mathbb{R}^3 \times \mathbb{R}^3 \) is defined by \( \hat{ab} = a \times b \) for all \( a, b \in \mathbb{R}^3 \). Moreover, we model the geometrical shape of a symmetric multicopter as its outer Löwner–John ellipsoid [37]

\[
E(t) = \{ R(t)Qx + p(t) \lVert x \rVert_2 \leq 1 \}
\]

where \( Q = \text{Diag}\{ r_x, r_y, h_z \} \), \( r_x \) and \( h_z \) are the radius and the height of the multicopter, respectively.

A feasible motion satisfies the safety and dynamic limits. By safety, we mean \( E(t) \subseteq \bar{F}, \forall t \in [0, T], \) where \( T \) is the total time of the motion. However, this safety constraint is indeed hard to enforce. We further make an assumption on \( \bar{F} \) that all \( \mathcal{P}_i^H \) or their intersections are able to contain at least one ellipsoid of the multicopter. This assumption can be reasonably satisfied when \( \bar{F} \) is generated incrementally. As a result, we can ensure safety through

\[
\forall t \in [0, T], \exists 1 \leq i \leq M_p, \text{s.t.} E(t) \subset \mathcal{P}_i^H.
\]

By dynamic limits, we mean the velocity, thrust, and body rate should have reasonable magnitude

\[
\begin{aligned}
\lVert p(t) \rVert_2^2 &\leq v_{\text{max}}^2, &\forall t \in [0, T] \\
\rho_{\text{min}} &\leq \dot{f}(t) \leq \rho_{\text{max}}, &\forall t \in [0, T]. \\
\lVert \omega(t) \rVert_2^2 &\leq \omega_{\text{max}}^2, &\forall t \in [0, T].
\end{aligned}
\]

Given a quotient-space trajectory \( p(t) : [0, T) \rightarrow \mathbb{R}^3, \) state-control trajectories of \( p, v, R, \) and \( \omega \) are all algebraically computed by flatness maps \( \Psi_p \) and \( \Phi_p \) of the dynamics (93). The concrete forms of the algebraic maps are detailed in [11] with fixes on the body rate [14] for simple quadcopters and, thus, are omitted here. Consequently, the entire SE(3) trajectory is also obtained. Denote by \( R(t) \) its rotational part. To generate \( p(t) \) in \( \bar{F} \), we follow the methodology of our previous experiment but with different constraints here.

The \( i \)th trajectory piece \( p_i(t) : [0, T_i) \rightarrow \mathbb{R}^3 \) is assigned to the polytope \( \mathcal{P}_i^H \) defined in (17) with \( j = \lfloor i/K \rfloor \). We denote by \( E_i(t) \) the ellipsoid induced by \( p_i(t) \) and the corresponding \( R_i(t) \) as is defined in (94). As proposed by Wu et al. [71], ensuring
safety by confining the vehicle ellipsoid in a polyhedron also has an analytical form. Specifically

$$\mathcal{E}_j(t) \in \mathcal{P}^H_j \quad \forall t \in [0, T_i] \quad (97)$$

is equivalent to

$$\left[ [A_j R_i(t) Q_j]^2 \mathbf{1} \right]^{1/2} + A_j p_i(t) - b_j \preceq 0, \quad (98a)$$

$$j = [i/K] \quad \forall t \in [0, T_i] \quad (98b)$$

where \( \mathbf{1} \) is an all-ones vector with an appropriate length and \([j]^2\) and \([j]^{1/2}\) are entrywise square and square root, respectively. Finally, we obtained the state-control constraint \( \mathcal{G}_D \) in (8) for the considered dynamics in (93). We choose to minimize \( s = 3 \) because it is the highest derivative order for flatness of (93) and also helpful in smoothing the angular rate.

We validate our framework in simulations where a relatively large quadcopter is required to fly through a narrow gap with much smaller width, as shown in Fig. 15. The settings are \( r_e = 0.5 \text{ m}, h_e = 0.1 \text{ m}, f_{\text{min}}/\bar{m} = 5.0 \text{ m/s}^2, f_{\text{max}}/\bar{m} = 18.5 \text{ m/s}^2, v_{\text{max}} = 6.5 \text{ m/s}, \) and \( \omega_{\text{max}} = 5.2 \text{ rad/s}. \) Intuitively, the quadcopter can only achieve no more than 1 revolution/s, making it less agile than small quadcopters [72] that can achieve 5 revolutions/s. The computation times, required roll angles, and SE(3) motions for different \( d_{\text{gap}} \) are shown in Table I and Fig. 15(b)–(f).

As the gap becomes narrower, the required roll angle becomes larger and the feasible space becomes smaller in view of dynamic limits. Our method is still able to find all the feasible motions. The superior computation speed makes it possible to solve SE(3) planning at a high frequency (at least 100 Hz). Constraint functions are visualized in Fig. 15(g). The body rate and thrust satisfy dynamic limits all the time. The continuous-time tightness of \( f_{\text{min}} \) for \( \phi_{\text{gap}} \in \{60^\circ, 75^\circ, 85^\circ\} \) shows the effectiveness of our penalty functional.

We evaluate the performance of our planner in a real-world experiment, where a quadcopter flies through several narrow windows. Sizes of the quadcopter and windows are given in Fig. 16. The quadcopter weights 794.2 g. The safety margin of the short side is only 5.4 cm, implying that the feasible motion space is extremely small. The settings are \( r_e = 20.0 \text{ cm}, h_e = 4.6 \text{ cm}, v_{\text{max}} = 4.0 \text{ m/s}, f_{\text{min}}/\bar{m} = 3.0 \text{ m/s}^2, f_{\text{max}}/\bar{m} = \)

### Table I

| \( d_{\text{gap}} \) | 0.88m | 0.76m | 0.60m | 0.40m | 0.25m |
|------------------|------|------|------|------|------|
| \( \phi_{\text{gap}} \) | 30°  | 45°  | 60°  | 75°  | 85°  |
| \( t_{\text{comp}} \) | 4.7ms | 4.4ms | 6.0ms | 6.6ms | 7.4ms |

Fig. 15. SFC layout for a narrow gap, the SE(3) trajectories under different widths of gaps, and the control inputs for different motions. As the gap becomes narrower, larger angular rates and higher thrust are needed for a safe flight. The proposed method persistently enforces limits on these control inputs under different settings while retaining millisecond-level computation time. (a) SFC layout. (b) \( \phi_{\text{gap}} = 30^\circ \). (c) \( \phi_{\text{gap}} = 45^\circ \). (d) \( \phi_{\text{gap}} = 60^\circ \). (e) \( \phi_{\text{gap}} = 75^\circ \). (f) \( \phi_{\text{gap}} = 85^\circ \). (g) Magnitude of angular velocity and the normalized thrust for different SE(3) trajectories.

Fig. 16. Sizes of the quadcopter and the narrow window. (a) Custom-made quadcopter. (b) Window.
Fig. 17. Experiment results for three SE(3) planning scenarios. (a) Interactive scenario. (b) Flying through two consecutive windows. (c) Flying through three consecutive windows. (d) SE(3) planning for two windows. (e) SE(3) planning for three windows.

18.0 m/s², ω_{max} = 6.0 \text{ rad/s}, and K = 2. The flying space is a restricted volume of 6.5 × 6.0 × 2.0 m³. All poses of narrow windows and the quadcopter are provided by a motion capture system running at 100 Hz. The obstacle-free region \( \mathcal{F} \) is geometrically computed for multiple narrow windows in the free volume. The planner is run on an offboard computer, where a human operator arbitrarily chooses the goal position. We adopt the control algorithm by Faessler et al. [14] for onboard SE(3) trajectory tracking.

The first scenario contains consecutive windows with roll angles ranging from 30° to 90°. The quadcopter has to fly through them and reach a randomly selected goal, as shown in Fig. 17(b)–(e). The second scenario is an interactive one where a human operator randomly holds a narrow window for real-time planning, as given in Fig. 17(a). The third scenario requires the quadcopter persistently fly back and forth through multiple windows for a long duration, as shown in Fig. 14(b) and (d). Our planner guides the quadcopter to fly back and forth through windows for about 20.0 s, while ensuring the safety and physical limits all the time. More details about this experiment are given in the attached multimedia.

In this experiment, the short distance between consecutive windows, the small acceleration/deceleration space, and the limited vehicle maneuverability are challenges that our planner must confront. We believe that these results constitute a strong evidence for its constraint fidelity, motion quality, computation efficiency, and robustness. However, we do observe the limitation of optimization-based methods. For example, if two 90° windows are asymmetrically placed, a multicopter has to pass them in sequence. Each window only allows two roll angles ±90°. The combinations are four locally optimal maneuvers, but only one can be the global optimum. Thus, the other three are shallow local minima inevitable for local methods.

VII. DISCUSSION AND CONCLUSION

A. Extensions

Profiting from the flexibility and efficiency, our framework has many applicative and algorithmic extensions. First, no assumption is ever made on concrete forms of vehicle dynamics and \( \mathcal{G}_D \). More accurate dynamics, such as the rotor drag [14], can be adopted to fully exploit physical limits via real-time high-fidelity planning and control. Time-dependent constraints for moving obstacles can also be supported by \( \mathcal{G}_D \). Second, our framework is inherently parallelizable to further squeeze its performance. Computation-demanding operations on \( I_0[p] \) are independent at each timestamp; thus, parallelization can effectively speedup our optimization. Moreover, it is possible to extend our methodology to other vehicle types whose flat-output space overlaps the configuration space. An example is the fixed-wing aircraft in [28], whose flights are mainly restricted by the trajectory curvature. Bry et al. [28] propose Dubins-polynomial trajectories for this restriction, while the curvature constraint is a special case of \( \mathcal{G} \) for MINCO.

To demonstrate the extendibility, we apply our framework to a swarm of multicopters to enable their autonomous navigation in unknown environments. All the details of the formulation (11) and real-world flights are given in a technical report [73].

B. Limitations

Our framework, like most optimization-based ones, focuses on local solutions of trajectory planning, thus suffering from shallow local minima. This can be alleviated by interleaving sampling-based or graph-search-based strategies into our framework, as proposed in [74]–[76]. A major limitation of the framework originates from MINCO itself. If \( \mathcal{G} \) exists, optimal solutions cannot, in general, be represented by polynomial splines, let alone MINCO. Thus, optimizing MINCO is just a relaxation to the original problem. However, our results show that MINCO can still represent high-quality solutions comparable to the ground truth, but with several orders of magnitudes faster computing. There are also limitations caused by the penalty functional. To achieve zero constraint violations, an unbounded smoothing factor or penalty weight and an unbounded quadrature resolution are both required. However, small constraint violations are empirically acceptable for multicopter navigation. As a reward, this method does not need initial feasible guesses.
C. Conclusion

In this article, we proposed a flexible multicopter trajectory planning framework powered by several core features, such as the MINCO trajectory based on our optimality conditions, constraint elimination schemes based on smooth maps, the penalty functional method based on constraint transcription, and the backward differentiation of the flatness maps from flat outputs. All these components enjoyed the efficiency and generality originating from low complexity and less preliminary assumptions. We performed extensive benchmarks against many kinds of multicopter trajectory planning methods to show the speedup over orders of magnitude and the top-level solution quality. A variety of applications demonstrated the versatility of our framework. We also presented further discussions about several unlisted applications or extensions as future work.

APPENDIX

A. Proof of Sufficiency in Theorem 2

Proof: We consider the space of $M$-piece polynomial 2s-order splines defined over $[t_0, t_M]$, where consecutive pieces on any $x : [t_0, t_M] \mapsto \mathbb{R}$ satisfy $x^{(j)}(t_i) = x^{(j)}(t_i)$ for $0 \leq j < d_i$ and $1 \leq i < M$. In (18), $d_i \leq s$ holds for each $i$. For brevity, we define $D_{i,j}$ as $D_{i,j} = \sum_{k=1}^{s} d_k$. According to [77, Th. 4.4], this spline space is actually a linear space of dimension $\tilde{D} = D_{2,M-1}$.

Moreover, an explicit basis of the space exists. Based on the original partition $t_0 < t_1 \ldots < t_M$, we define an extended partition $\bar{t}_1 \leq \bar{t}_2 \leq \ldots \leq \bar{t}_M$ of length $\tilde{M} = M_{2,M-1}$ as

$$\bar{t}_i = \begin{cases} t_0, & \text{if } 1 \leq i \leq D_{2,0} \\ t_j, & \text{if } D_{2,j-1} < i \leq D_{2,j} \\ t_M, & \text{if } D_{2,M-1} < i \leq \tilde{M} \end{cases}$$

(99)

Based on this extended partition, [77, Th. 4.9] explicitly constructs $D$ functions $\{B_i(t) : [t_0, t_M] \mapsto \mathbb{R}\}^{D_{1,M}}_{i=1}$, which form a basis for the considered spline space.

Now, we consider (18c) and (18d) in the spanned linear space. These conditions specify derivative values on timestamps of the original partition to be interpolated by the basis $\{B_i(t)\}^{D}_{i=1}$. We only need the specified orders along with their timestamps instead of the specified derivative values. Denote by $\nu_i$ the ith specified timestamps, where

$$\nu_i = \begin{cases} i-1, & \text{if } 1 \leq i \leq D_{i,0} \\ i - 1 - D_{i,j-1}, & \text{if } D_{i,j-1} < i \leq D_{i,j} \\ i - 1 - D_{i,M-1}, & \text{if } D_{i,M-1} < i \leq \tilde{D} \end{cases}$$

(100)

Denote by $\nu_i$ the specified order at $\tau_i$, written as

Then, the conditions (18c) and (18d) generate a linear equation system on the basis, whose coefficient matrix is

$$B = \begin{pmatrix} B_1^{(\nu_1)}(\tau_1) & B_2^{(\nu_1)}(\tau_1) & \cdots & B_{D}^{(\nu_1)}(\tau_1) \\ B_1^{(\nu_2)}(\tau_2) & B_2^{(\nu_2)}(\tau_2) & \cdots & B_{D}^{(\nu_2)}(\tau_2) \\ \vdots & \vdots & \ddots & \vdots \\ B_1^{(\nu_D)}(\tau_D) & B_2^{(\nu_D)}(\tau_D) & \cdots & B_{D}^{(\nu_D)}(\tau_D) \end{pmatrix}.$$ (102)

Since $B$ is a square matrix, there always exists a solution for all conditions in Theorem 2 in each dimension.

According to [77, Th. 4.67], $B$ is nonsingular if and only if

$$\tau_i \in \delta_i = \begin{cases} [\bar{t}_i, \bar{t}_{i+2s}], & \text{if } \nu_i + \alpha_i - 2s \geq 0 \\ (\bar{t}_i, \bar{t}_{i+2s}], & \text{if } \nu_i + \alpha_i - 2s < 0 \end{cases}$$

(103)

holds for any $i = 1, \ldots, \tilde{D}$, where $\alpha_i$ is defined as

$$\alpha_i = \{\max_{j} : \bar{t}_i = \cdots = \bar{t}_{i+j-1}\}.$$ (104)

We show that (103) is always true in our case. It is obvious that $\alpha_i$ can be computed as

$$\alpha_i = \begin{cases} D_{2,0} - i + 1, & \text{if } 1 \leq i \leq D_{2,0} \\ D_{2,j} - i + 1, & \text{if } D_{2,j-1} < i \leq D_{2,j} \end{cases}.$$ (105)

Combining (101) and (105), we know that $\nu_i < s$ and $\alpha_i \leq s$ always hold for $i > s$, which means

$$\nu_i + \alpha_i - 2s = 0, \quad \text{if } 1 \leq i \leq s$$

(106)

$$\nu_i + \alpha_i - 2s < 0, \quad \text{if } s < i \leq \tilde{D}.$$ (107)

Thus, the interval $\delta_i$ is computed as

$$\delta_i = \begin{cases} [\bar{t}_i, \bar{t}_{i+2s}], & \text{if } 1 \leq i \leq s \\ (\bar{t}_i, \bar{t}_{i+2s}], & \text{if } s < i \leq \tilde{D}. \end{cases}$$

Consequently, we have

$$\tau_i = t_0 \subset [t_0, t_M] \subset [\bar{t}_i, \bar{t}_{i+2s}] = \delta_i, \quad 1 \leq i \leq s.$$ (108)

When $i > s$, we denote $\tilde{t}_i = t_k$, $\bar{t}_{i+2s} = t_l$ and $\tau_i = t_j$. As shown in (99) and (100), we have

$$D_{2,k-1} < i, \ (i + 2s) \leq D_{2,l}, \ D_{1,j-1} < i \leq D_{1,j}.$$ (109)

Owing to the fact that $d_i \leq s$ holds for any $1 \leq i < M$, the following two inequalities always hold:

$$D_{2,k-1} < i \leq D_{1,j} = (D_{2,j} - s) \leq D_{2,j-1}$$

(110)

$$D_{2,j} = (D_{1,j} + s) \leq (D_{1,j-1} + 2s) < (i + 2s) \leq D_{2, l}.$$ (111)

Inequalities (110) and (111) imply $k < j$ and $j < l$; thus

$$\tau_i = t_j \in (t_k, t_l) = (\tilde{t}_i, \tilde{t}_{i+2s}) = \delta_i, \quad s < i \leq \tilde{D}$$ (112)

always holds. Combining (108) and (112) gives (103). Therefore, the coefficient matrix $B$ on basis is always nonsingular for settings on the original problem, implying the existence and uniqueness of solution.

The optimality conditions guarantee one unique solution in each decoupled dimension, which gives its sufficiency.
B. Proof of Proposition 2

Proof: Denote by $J$ the Jacobian of $G$. For any $x \in \mathbb{D}_F$ and $y \in \mathbb{R}^N$, satisfying $x = G(y)$ or $y = G^{-1}(x)$, we have

$$\nabla H(y) = J(y)^T \nabla F(x). \quad (113)$$

Then, the nonsingularity of $J$ implies that the first statement always holds. Denote by $K_i$ the Hessian of the $i$th entry in $G$. If $x$ and $y$ are stationary points, the Hessian of $H$ is

$$\nabla^2 H(y) = J(y)^T \nabla^2 F(x)J(y) + \sum_{i=1}^{N} \frac{\partial F(x)}{\partial x_i} K_i(y),$$

Then, the nonsingular $J$ implies that $\nabla^2 F(x)$ and $\nabla^2 H(y)$ are congruent [45]. Thus, the second statement holds. ■

ACKNOWLEDGMENT

The authors would like to thank Shaohui Yang for his profound insight into the experiment design and applications of this framework, Hongkai Ye and Yuwei Wu for their help in the benchmark, and Yuman Gao, Tiankai Yang, and Neng Pan for the hardware platform for high-speed flights.

REFERENCES

[1] M. Ryll, J. Ware, J. Carter, and N. Roy, “Efficient trajectory planning for high speed flight in unknown environments,” in Proc. IEEE Int. Conf. Robot. Autom., 2019, pp. 732–738.

[2] H. Oleynikova et al., “An open-source system for vision-based micro-aerial vehicle mapping, planning, and flight in cluttered environments,” J. Field Robot., vol. 37, no. 4, pp. 462–466, 2020.

[3] J. Zhang, C. Hu, R. G. Chadha, and S. Singh, “Falco: Fast likelihood-based collision avoidance with extension to human-guided navigation,” J. Field Robot., vol. 37, no. 8, pp. 1300–1313, 2020.

[4] L. Campos-Macías, R. Aldana-López, R. de la Guardia, L. J. Parra-Vilchis, and D. Gómez-Gutiérrez, “Autonomous navigation of MAVs in unknown cluttered environments,” J. Field Robot., vol. 38, no. 2, pp. 307–326, 2021.

[5] X. Zhou, Z. Wang, H. Ye, C. Xu, and F. Gao, “EGO-Planner: An ESDF-free gradient-based local planner for quadrotors,” IEEE Robot. Autom. Lett., vol. 6, no. 2, pp. 478–485, Apr. 2021.

[6] P. Foehn et al., “AlphaPilot: Autonomous drone racing,” Auton. Robots, vol. 46, pp. 307–320, 2022.

[7] M. Fliesen, J. Lévine, P. Martin, and P. Rouchon, “Flatness and defect of non-linear systems: Introductory theory and examples,” Int. J. Control, vol. 61, no. 6, pp. 1327–1361, 1995.

[8] M. J. Van Nieuwstadt and R. M. Murray, “Real-time trajectory generation for differentially flat systems,” Int. J. Robust Nonlinear Control, vol. 8, no. 11, pp. 995–1020, 1998.

[9] P. Martin, R. M. Murray, and P. Rouchon, “Flat systems, equivalence and trajectory generation,” California Inst. Technol., Pasadena, CA, USA, Tech. Rep. CDS 2003-008, 2003.

[10] J.-C. Ryu and S. K. Agrawal, “Differential flatness-based robust control of mobile robots in the presence of slip,” Int. J. Robot. Res., vol. 30, no. 4, pp. 463–475, 2011.

[11] D. Mellinger and V. Kumar, “Minimum snap trajectory generation and control for quadrotors,” in Proc. IEEE Int. Conf. Robot. Autom., 2011, pp. 2520–2525.

[12] M. Waterson and V. Kumar, “Control of quadrotors using the Hopf fibration on SO(3),” in Proc. Int. Symp. Robot. Res., 2019, pp. 199–215.

[13] J. Ferrin, R. Leishman, R. Beard, and T. McLain, “Differential flatness based control of a rotorcraft for aggressive maneuvers,” in Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst., 2011, pp. 2688–2693.

[14] M. Faessler, A. Franchi, and D. Scaramuzza, “Differential flatness of quadrotor dynamics subject to rotor drag for accurate tracking of high-speed trajectories,” IEEE Robot. Autom. Lett., vol. 3, no. 2, pp. 620–626, Apr. 2018.
Zhepei Wang received the B.Eng. degree in control science and engineering in 2017 from Zhejiang University, Hangzhou, China, where he is currently working toward the Ph.D. degree in control science and engineering.

His research interests include motion planning, discrete and computational geometry, numerical optimization, and autonomous navigation of unmanned vehicles.

Xin Zhou received the B.Eng. degree in electrical engineering and automation from the China University of Mining and Technology, Xuzhou, China, in 2019. He is currently working toward the Ph.D. degree in control engineering with Zhejiang University, Hangzhou, China.

His research interests include motion planning and mapping for aerial swarm robotics.

Chao Xu received the Ph.D. degree in mechanical engineering from Lehigh University, Bethlehem, PA, USA, in 2010.

He is currently a Professor of Cyber-Systems and Robotics, the Associate Dean of the College of Control Science and Engineering, and the Founding Dean of the Huazhao Institute of Zhejiang University, Hangzhou, China. He founded the Field Autonomous System and Computing Lab, Zhejiang University, Hangzhou, China. His research interests include robot mechanics and control.

Dr. Xu is the Managing Editor for *Journal of Industrial and Management Optimization* and the Founding Managing Editor for *IET Cyber-Systems and Robotics*.

Fei Gao received the Ph.D. degree in electronic and computer engineering from the Hong Kong University of Science and Technology, Hong Kong, in 2019. He is currently an Assistant Professor with the College of Control Science and Engineering, Zhejiang University, Hangzhou, China, where he codirects the Field Autonomous System and Computing Lab and leads the Flying Autonomous Robotics Group. His research interests include aerial robots, swarms, autonomous navigation, motion planning, and localization and mapping.