2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

Sequential Convex Programming for Time-optimal Quadrotor Waypoint Flight

Zhipeng Shen, Guanzhong Zhou, and Hailong Huang

Abstract— **Agile flight is significant for target tracking, search and rescue, and delivery applications. To achieve agile flight, we can exploit the actuator's potential by utilizing the full dynamics of the quadrotor. However, the 6-degrees-of-freedom dynamics render the optimization problem non-convex, and thus computationally intractable. To tackle this issue, we convert the original non-convex optimal control problem (OCP) into a convex subproblem and use the sequential convex programming (SCP) algorithm to iteratively solve the subproblems. Moreover, the state-triggered constraints are proposed to simultaneously optimize the time allocation of the waypoint and the trajectory itself. The numerical and physical experiment results[1] show that the SCP algorithm can significantly reduce the computing time while ensuring a satisfactory solution.**

Fig. 1: The quadrotor is aggressively passing waypoints using the proposed method.

I. INTRODUCTION

Time-optimal trajectory planning can be challenging for quadrotors due to their complex dynamics. The 6-degrees-of-freedom (DoF) dynamics of quadrotors, which involve both translational and rotational motion, create a highly nonlinear and non-convex trajectory optimization problem. Additionally, quadrotors have limited actuators, which can further complicate time-optimal planning by imposing constraints on the control inputs. These factors make time-optimal planning for quadrotors a computationally demanding and challenging problem that requires sophisticated algorithms and control strategies. Despite these challenges, time-optimal planning can have significant benefits for quadrotors in terms of reducing flight time (see Fig. 1), increasing efficiency, and improving overall performance in a wide range of applications, such as high-dynamic photography, search and rescue missions, and package delivery [1]–[3].

Quadrotor trajectory planning for agile flight has received extensive attention in recent years. By utilizing the differential flatness of the quadrotor [4], polynomial trajectories are widely used for quadrotor planning due to their computational efficiency [5], [6]. However, a quadrotor cannot sustain the maximum thrust of its actuator for an extended duration using a polynomial trajectory, due to the inherent smoothness as highlighted by [3]. In [7], [8], polynomial interpolations were used for trajectory optimization to achieve high computational efficiency. However, the integration of multiple optimization objectives, such as dynamics feasibility and collision avoidance, within a single objective function can result in uncertainty regarding the strict satisfaction of constraints and the guaranteed optimality of a particular objective. The recent work [3] pushed the quadrotor to its physical limits by utilizing the full dynamics and single actuator constraints, which can even outperform human expert drone pilots in drone racing. However, the 6-DoF dynamics render the optimization problem non-convex and computationally intractable.

To address computational efficiency, the studies [9] and [10] have introduced point-mass model planning and model predictive contouring control (MPCC), enabling rapid drone flight by incorporating the nonlinear model directly into the controller, thereby mitigating computational challenges. Recent advancements in [11] and [12] have employed reinforcement learning techniques for drone racing. Despite these innovations, a comprehensive trajectory planning algorithm that effectively balances nonlinear model considerations with computational efficiency remains elusive.

As a powerful and computationally efficient method, the convex optimization-based approach can be used for reliable and efficient trajectory optimization [13]–[19]. The lossless convexification method was proposed in [13] to present a solution to optimal control problems (OCPs) with non-convex actuator constraints and then extended to problems with linear state constraints [14]. For OCPs with more complex nonconvexities, such as the 6-DoF dynamics, sequential convex programming (SCP) is an effective approach [15], [16]. The SCP method has been applied in fuel-optimal rocket landing, quadrotor obstacle avoidance, and 6-DoF free flyer [17]–[19]. By employing linearization, the SCP method approximates the solution to the original problem through the iterative resolution of convex subproblems. Recent work [19] provides theoretical analysis and convergence guarantees of SCP, which further improve its reliability. In this paper,

This work was supported by Research Centre for Unmanned Autonomous Systems, the Hong Kong Polytechnic University, via the project of P0049529.

The authors are with the Department of Aeronautical and Aviation Engineering, the Hong Kong Polytechnic University, Hong Kong, China. (*Corresponding author:* Hailong Huang.) hailong.huang@polyu.edu.hk

[1]Physical experiments video: https://youtu.be/kZ6nzU0uxVw

we reduce the computing burden by converting the original non-convex OCP into a convex subproblem, which the SCP algorithm can efficiently solve.

To achieve time-optimal waypoint flight, the quadrotor is required to not only fly as fast as possible but also pass through waypoints. Since the time allocation of the waypoint is a priori unknown, the time allocation can result in a suboptimal flight. In [3], a complementary progress constraint was proposed to realize simultaneous optimization of the waypoint allocation and the trajectory. However, the complementary constraint represents the strict *if-and-only-if* statements, which makes [3] introduce the slack variables to relax the constraint. Therefore, we introduce a continuous state-triggered constraint (STC), which represents more general *if* statements [17]. In this paper, we also show that using STC does not require additional slack variables for waypoint flight and thus renders the optimization problem less complex.

The primary contributions of this paper are as follows:

1) We formulate the time-optimal waypoint flight problem following the optimal control framework, which is in essence non-convex. Then, we convert the original problem into a convex subproblem, which can be efficiently solved by the SCP algorithm. The proposed method is much more computationally tractable than [3].
2) The STC is utilized to simultaneously optimize the waypoint allocation and the trajectory. Meanwhile, the complementary constraint in [3] requires additional slack variables to realize waypoint flight. Fewer optimization variables render the STC more computationally tractable. Besides, the STC in [17] uses the $\min(\cdot)$ function, which makes the constraint not continuously differentiable. Since the SCP algorithm is based on linearization, the $\min(\cdot)$ function can cause the linearization to be not well defined. In this paper, the STC in use can successfully bypass the $\min(\cdot)$ function.

The structure of this paper is delineated as follows: Section II delineates the formulation of the time-optimal waypoint flight problem. Section III shows the transformation of the initial non-convex OCP into a convex subproblem and presents the SCP algorithm. The experimental findings are disclosed in Section IV. Finally, Section V provides a summative closure to the discourse.

II. PROBLEM STATEMENT

In this section, the time-optimal quadrotor trajectory planning problem is formulated as a non-convex OCP, while considering the waypoint constraints.

A. Notation

In this paper, $\times$ denotes the vector cross product, while $\|\cdot\|_2$ represents the Euclidean norm. We define the initial time t_0 as the time at which the flight begins and the terminal time t_f at which the quadrotor reaches the terminal states. The subscripts $\mathcal{I}$ and $\mathcal{B}$ are employed to denote parameters articulated in the inertial frame $\mathcal{F}_\mathcal{I}$ and the body-fixed frame $\mathcal{F}_\mathcal{B}$, correspondingly.

B. Dynamics

The dynamics governing the quadrotor are shown as

$$\dot{\boldsymbol{p}}_\mathcal{I}(t) = \boldsymbol{v}_\mathcal{I}(t), \tag{1}$$

$$\dot{\boldsymbol{v}}_\mathcal{I}(t) = \frac{1}{m} R_{\mathcal{BI}}(t) \boldsymbol{T}_\mathcal{B}(t) + \boldsymbol{g}_\mathcal{I}, \tag{2}$$

$$\dot{\boldsymbol{q}}_{\mathcal{IB}}(t) = \frac{1}{2} \Omega\left(\boldsymbol{\omega}_\mathcal{B}(t)\right) \boldsymbol{q}_{\mathcal{IB}}(t), \tag{3}$$

$$J_\mathcal{B} \dot{\boldsymbol{\omega}}_\mathcal{B}(t) = \boldsymbol{M}_\mathcal{B}(t) - \boldsymbol{\omega}_\mathcal{B}(t) \times J_\mathcal{B} \boldsymbol{\omega}_\mathcal{B}(t), \tag{4}$$

where $\boldsymbol{p}_\mathcal{I}(t) \in \mathbb{R}^3$ is the position, $\boldsymbol{v}_\mathcal{I}(t) \in \mathbb{R}^3$ is the velocity, $m \in \mathbb{R}_{++}$ is the mass, $\boldsymbol{T}_\mathcal{B}(t)$ is the thrust, $\boldsymbol{g}_\mathcal{I} \in \mathbb{R}^3$ is the gravitational acceleration, and rotation matrix $R_{\mathcal{BI}}(t)$ corresponds to the quaternion $\boldsymbol{q}_{\mathcal{BI}}(t)$, which facilitates the transformation from the body-fixed frame $\mathcal{F}_\mathcal{B}$ to the inertial frame $\mathcal{F}_\mathcal{I}$. Conversely, the quaternion enabling the rotation from $\mathcal{F}_\mathcal{I}$ to $\mathcal{F}_\mathcal{B}$ is represented as $\boldsymbol{q}_{\mathcal{IB}}(t)$. $\Omega(\cdot)$ denotes a skew-symmetric matrix. The quadrotor's moment of inertia is denoted by $J_\mathcal{B} \in \mathbb{S}^3_+$, while $\boldsymbol{M}_\mathcal{B}(t) \in \mathbb{R}^3$ and $\boldsymbol{\omega}_\mathcal{B}(t) \in \mathbb{R}^3$ denote the torque and the angular rate, respectively. $\boldsymbol{T}_\mathcal{B}(t)$ and $\boldsymbol{M}_\mathcal{B}(t)$ can be presented in details as

$$\boldsymbol{T}_\mathcal{B}(t) = \begin{bmatrix} 0 & 0 & \sum_{i=1}^{4} T_i(t) \end{bmatrix}^{\mathrm{T}}, \tag{5}$$

$$\boldsymbol{M}_\mathcal{B}(t) = \begin{bmatrix} l_\tau & l_\tau & -l_\tau & -l_\tau \\ -l_\tau & l_\tau & l_\tau & -l_\tau \\ c_\tau & -c_\tau & c_\tau & -c_\tau \end{bmatrix} \boldsymbol{u}(t), \tag{6}$$

where $\boldsymbol{u}(t) \triangleq \begin{bmatrix} T_1(t) & T_2(t) & T_3(t) & T_4(t) \end{bmatrix}^{\mathrm{T}}$, $T_i(t)$ is the thrust generated by i-th rotor, $l_\tau = l/\sqrt{2}$, l and c_τ are the arm length and torque constant, respectively.

C. Optimal Control Problem

The quadrotor dynamics (1)-(4) can be represented as

$$\dot{\boldsymbol{x}}(t) = f(\boldsymbol{x}(t), \boldsymbol{u}(t)) \quad \forall t \in [t_0, t_f], \tag{7}$$

$$\boldsymbol{x}(t) \triangleq \begin{bmatrix} \boldsymbol{r}_\mathcal{I}^{\mathrm{T}}(t) & \boldsymbol{v}_\mathcal{I}^{\mathrm{T}}(t) & \boldsymbol{q}_{\mathcal{IB}}^{\mathrm{T}}(t) & \boldsymbol{\omega}_\mathcal{B}^{\mathrm{T}}(t) \end{bmatrix}^{\mathrm{T}}, \tag{8}$$

where $\boldsymbol{x}(t) \in \mathbb{R}^{13}$ denotes the state, and $f : \mathbb{R}^{13} \times \mathbb{R}^4 \to \mathbb{R}^{13}$ denotes the continuous-time non-convex dynamics.

The control constraints can be given by

$$T_{\min} \leq T_i(t) \leq T_{\max} \quad \forall i \in \{1, 2, 3, 4\}, \tag{9}$$

where $[T_{\min}, T_{\max}] \subset \mathbb{R}_+$ is the permitted thrust interval.

To force the quadrotor to pass through a waypoint $\boldsymbol{p}_w$, the trajectory should guarantee that there exists a time instant $t_k \in [t_0, t_f]$ such that the following constraint can be satisfied

$$\|\boldsymbol{p}_\mathcal{I}(t_k) - \boldsymbol{p}_w\|_2^2 \leq d_t^2, \tag{10}$$

where d_t is a tolerance distance.

The formulation of the OCP can be tailored to specific scenarios and mission objectives. In this study, our primary emphasis is on the time-optimal problem, aiming to minimize the terminal time. The OCP is summarized as Problem 1 for all $t \in [t_0, t_f]$.

Problem 1.

$$\min_{t_f, \boldsymbol{u}(t)} \; t_f$$
$$\text{s.t.} \begin{cases} \dot{\boldsymbol{x}}(t) = f(\boldsymbol{x}(t), \boldsymbol{u}(t)) \\ \boldsymbol{x}(t_0) = \boldsymbol{x}_i, \; \boldsymbol{x}(t_f) = \boldsymbol{x}_f, \\ T_{\min} \leq T_i(t) \leq T_{\max} \quad \forall i \in \{1,2,3,4\}, \\ \|\boldsymbol{p}_{\mathcal{I}}(t_k) - \boldsymbol{p}_w\|_2^2 \leq d_t^2 \quad \exists t_k \in [t_0, t_f]\,, \end{cases} \tag{11}$$

where $\boldsymbol{x}_i$ and $\boldsymbol{x}_f$ are the prescribed boundary conditions, respectively. Our objective is to find the control profile $\boldsymbol{u}(t)$ and the final time t_f to solve the non-convex OCP (11).

III. CONVEX FORMULATION

In this section, the SCP algorithm is presented to solve Problem 1. The algorithm introduced in this study iteratively addresses the original non-convex problem by solving a series of convex subproblems. Despite the necessity for multiple iterations to achieve convergence, the inherently lower computational demands of convex programming ensure that this approach remains significantly more efficient than tackling the original non-convex problem directly (see Section IV). In addition, the STCs are utilized to realize simultaneous optimization of the waypoint time allocation and the trajectory.

A. Fixed Final Time

The Problem 1 is a free-final-time non-convex OCP. Utilization of a scaled time variable $\tau \in [0,1]$, enables the conversion of this free-final-time problem into one characterized by a predetermined, scaled final time. Utilizing the chain rule allows us to express the quadrotor dynamics in terms of this scaled time.

$$\boldsymbol{x}'(\tau) \triangleq \frac{d}{d\tau}\boldsymbol{x}(\tau) = \frac{dt}{d\tau} f(\boldsymbol{x}(\tau), \boldsymbol{u}(\tau)). \tag{12}$$

Let $t_0 = 0$, then $t = t_f \tau$. Thus, the dynamics can be transformed into

$$\boldsymbol{x}'(\tau) = t_f f(\boldsymbol{x}(\tau), \boldsymbol{u}(\tau)) \triangleq F(\boldsymbol{x}(\tau), \boldsymbol{u}(\tau), t_f). \tag{13}$$

The problem initially posed with a free final time, is reformulated into one with a predetermined final scaled time by normalizing the time interval to $\tau \in [0,1]$.

B. Linearization and Discretization

Leveraging the computational efficiency inherent in convex programming necessitates transforming the original Problem 1 into a convex form. Accordingly, the equality constraint needs to be rendered affine [20]. Consequently, the linearized dynamics (13) are given by

$$\boldsymbol{x}'(\tau) \approx A(\tau)\boldsymbol{x}(\tau) + B(\tau)\boldsymbol{u}(\tau) + \boldsymbol{s}(\tau)t_f + \boldsymbol{c}(\tau), \tag{14}$$

$$A(\tau) \triangleq \left.\frac{\partial F}{\partial \boldsymbol{x}}\right|_{\tilde{z}(\tau)}, B(\tau) \triangleq \left.\frac{\partial F}{\partial \boldsymbol{u}}\right|_{\tilde{z}(\tau)}, \tag{15}$$

$$\boldsymbol{s}(\tau) \triangleq \left.\frac{\partial F}{\partial t_f}\right|_{\tilde{z}(\tau)}, \boldsymbol{c}(\tau) \triangleq -A(\tau)\tilde{\boldsymbol{x}}(\tau) - B(\tau)\tilde{\boldsymbol{u}}(\tau), \tag{16}$$

where $\tilde{\boldsymbol{z}}(\tau) \triangleq \begin{bmatrix} \tilde{t}_f & \tilde{\boldsymbol{x}}^{\mathrm{T}}(\tau) & \tilde{\boldsymbol{u}}^{\mathrm{T}}(\tau) \end{bmatrix}^{\mathrm{T}}$ is the reference trajectory.

The continuous-time formulation of the original problem necessitates discretization for computational tractability. This is achieved by partitioning the time domain into $N-1$ subintervals using N uniformly distributed nodes. In every subinterval, the control command is obtained using first-order hold as

$$\boldsymbol{u}(\tau) = \boldsymbol{u}_k + \frac{\boldsymbol{u}_{k+1} - \boldsymbol{u}_k}{\tau_{k+1} - \tau_k}(\tau - \tau_k) \quad \forall \tau \in [\tau_k, \tau_{k+1}]\,, \tag{17}$$

where $k \in \bar{\mathbb{N}}$, $\bar{\mathbb{N}} \triangleq \{1, 2, \ldots, N-1\}$, $\tau_k = (k-1)/(N-1)$, and $\boldsymbol{u}_k \triangleq \boldsymbol{u}(\tau_k)$.

The dynamics, as linearized in equation (14), constitute a linear-time-varying (LTV) system. The discrete-time dynamics can be given for each $k \in \bar{\mathbb{N}}$ [21]:

$$\boldsymbol{x}_{k+1} = A_k \boldsymbol{x}_k + \hat{B}_k \boldsymbol{u}_k + B_k \boldsymbol{u}_{k+1} + \boldsymbol{s}_k t_f + \boldsymbol{c}_k, \tag{18}$$

$$\Phi(\tau, \tau_k) \triangleq I + \int_{\tau_k}^{\tau} A(\zeta)\Phi(\zeta, \tau_k)\,\mathrm{d}\zeta, \tag{19a}$$

$$A_k \triangleq \Phi(\tau_{k+1}, \tau_k)\,, \tag{19b}$$

$$\hat{B}_k \triangleq A_k \int_{\tau_k}^{\tau_{k+1}} \Phi^{-1}(\tau, \tau_i)\,\hat{\eta}(\tau)B(\tau)\mathrm{d}\tau, \tag{19c}$$

$$B_k \triangleq A_k \int_{\tau_k}^{\tau_{k+1}} \Phi^{-1}(\tau, \tau_k)\,\eta(\tau)B(\tau)\mathrm{d}\tau, \tag{19d}$$

$$\boldsymbol{s}_k \triangleq A_k \int_{\tau_k}^{\tau_{k+1}} \Phi^{-1}(\tau, \tau_k)\,\boldsymbol{s}(\tau)\mathrm{d}\tau, \tag{19e}$$

$$\boldsymbol{c}_k \triangleq A_k \int_{\tau_k}^{\tau_{k+1}} \Phi^{-1}(\tau, \tau_i)\,\boldsymbol{c}(\tau)\mathrm{d}\tau, \tag{19f}$$

where I is an identity matrix, and $\boldsymbol{x}_k \triangleq \boldsymbol{x}(\tau_k)$.

It is important to observe that the integration process delineated in equation (19) relies solely on the reference trajectories. Consequently, the (19) can be computed in parallel, significantly reducing the computational load [17].

Given the iterative nature of the SCP algorithm, necessitated by the process of linearization, we incorporate a trust region to maintain the effectiveness of linearization. To facilitate the selection of an appropriate trust region by the convex programming framework, the objective function is further refined by incorporating the following additional term:

$$J_{\mathrm{tr}}(\tilde{\boldsymbol{x}}, \boldsymbol{x}) \triangleq \sum_{k \in \mathbb{N}} \delta\boldsymbol{x}_k{}^{\mathrm{T}} W_{\mathrm{tr}} \delta\boldsymbol{x}_k, \tag{20}$$

where $k \in \mathbb{N}$, $\mathbb{N} \triangleq \{1, 2, \ldots, N\}$, $\delta\boldsymbol{x}_k \triangleq \boldsymbol{x}_k - \tilde{\boldsymbol{x}}_k$, and $W_{\mathrm{tr}} \in \mathbb{S}_+^{13}$ is the weighing matrix.

The process of linearization may inadvertently introduce artificial infeasibility [17]. Specifically, a linearized version of the problem may be deemed infeasible, notwithstanding the feasibility of the original problem. This discrepancy often arises when linearized constraints clash with other constraints, rendering them mutually unsatisfiable due to the reliance on an unrealistic reference trajectory for linearization. To address this challenge, a virtual control term is added

$\boldsymbol{\nu}_k \in \mathbb{R}^{13}$ into the dynamics (18). Consequently, the modified dynamics are as follows:

$$\boldsymbol{x}_{k+1} = A_k \boldsymbol{x}_k + \hat{B}_k \boldsymbol{u}_k + B_k \boldsymbol{u}_{k+1} + \boldsymbol{s}_k t_f + \boldsymbol{c}_k + \boldsymbol{\nu}_k. \quad (21)$$

Note that $\boldsymbol{\nu}_k$ is invoked solely upon necessity. Thus, the following term is integrated into the objective function.

$$J_{\mathrm{vc}}(\boldsymbol{\nu}) \triangleq w_{\mathrm{vc}} \sum_{k \in \bar{\mathbb{N}}} \|\boldsymbol{\nu}_k\|_1, \quad (22)$$

where $\boldsymbol{\nu} \triangleq \begin{bmatrix} \boldsymbol{\nu}_1 & \boldsymbol{\nu}_2 & \dots & \boldsymbol{\nu}_{N-1} \end{bmatrix} \in \mathbb{R}^{13 \times (N-1)}$, $\|\cdot\|_1$ denotes one-norm, and $w_{\mathrm{vc}} \in \mathbb{R}_{++}$ is a large weighing term.

C. State-Triggered Constraints

In this paper, we use the STC to incorporate the waypoint constraint into an optimization problem such that the problem is well-defined (see Section III-D). An STC is a constraint that is conditionally enforced to obey the following logical statements:

$$y(\boldsymbol{z}) \leq 0 \Rightarrow h(\boldsymbol{z}) \geq 0, \quad (23)$$

$$y(\boldsymbol{z}) \geq 0 \Rightarrow h(\boldsymbol{z}) = 0, \quad (24)$$

where $\boldsymbol{z} \in \mathbb{R}^{n_z}$ is the optimization variable, $y(\cdot) : \mathbb{R}^{n_z} \to \mathbb{R}$ is the trigger function, and $h(\cdot) : \mathbb{R}^{n_z} \to \mathbb{R}$ is the constraint function.

The logical statements (23) and (24) imply that, if the conditions enforced on the trigger function $y(\cdot)$ are satisfied, then the constraints enforced on the constraint function $h(\cdot)$ should also be satisfied. To incorporate the STCs into a continuous optimization framework, the STCs can be equivalently converted into the following constraints:

$$g(\boldsymbol{z}) \leq 0, \quad (25)$$

$$-h(\boldsymbol{z}) \leq 0, \quad (26)$$

where $g(\boldsymbol{z}) \triangleq y(\boldsymbol{z}) \cdot h(\boldsymbol{z})$.

D. Waypoint Flight through STC

To achieve the simultaneous optimization of the trajectory and the waypoint time allocation, the progress measure variables are introduced to describe the progress along a trajectory [3]. For each $k \in \mathbb{N}$, the variable λ_k must satisfy the following constraints:

$$\lambda_{k+1} = \lambda_k - \mu_k, \quad (27a)$$

$$\lambda_0 = 1, \quad (27b)$$

$$\lambda_N = 0. \quad (27c)$$

The variable must start as 1 at t_0 and must become 0 at t_f, and the variable propagates according to (27b). We can enforce the trajectory passing through the waypoint by enforcing

$$\|\boldsymbol{p}_{\mathcal{I}}(\tau_k) - \boldsymbol{p}_w\|_2^2 - d_t^2 \leq 0 \Rightarrow \mu_k \geq 0, \quad (28)$$

$$\|\boldsymbol{p}_{\mathcal{I}}(\tau_k) - \boldsymbol{p}_w\|_2^2 - d_t^2 \geq 0 \Rightarrow \mu_k = 0, \quad (29)$$

for all $k \in \bar{\mathbb{N}}$. The constraints (28) and (29) can guarantee the progress variable becomes 0 only when the trajectory is close enough to the waypoint. According to Section III-C, the constraints can be rewritten as

$$g(\boldsymbol{y}_k) \triangleq (\|\boldsymbol{p}_{\mathcal{I}}(\tau_k) - \boldsymbol{p}_w\|_2^2 - d_t^2)\mu_k \leq 0, \quad (30)$$

$$-\mu_k \leq 0, \quad (31)$$

where $\boldsymbol{y}_k \triangleq \begin{bmatrix} \boldsymbol{x}_k{}^{\mathrm{T}} & \mu_k \end{bmatrix}^{\mathrm{T}}$. Note that the constraint (30) is not convex. The constraint (30) can be approximated as

$$g(\boldsymbol{y}_k) \approx g(\tilde{\boldsymbol{y}}_k) + \left.\frac{\partial g(\boldsymbol{y}_k)}{\partial \boldsymbol{y}_k}\right|_{\tilde{\boldsymbol{y}}_k} \delta \boldsymbol{y}_k, \quad (32)$$

where $\delta \boldsymbol{y}_k \triangleq \boldsymbol{y}_k - \tilde{\boldsymbol{y}}_k$, and $\tilde{\boldsymbol{y}}_k$ is the reference value obtained from the reference trajectory.

Remark 1. The complementary progress constraints introduced in [3] can also enforce the trajectory passing through the waypoint. However, the method used in [3] must introduce slack variables for tolerance relaxation. These additional optimization variables can further increase the computing burden. The STC introduced in this paper does not require additional variables, i.e., the STC is more computationally efficient than the complementary progress constraint.

E. Sequential Convex Programming

To ensure the optimal trajectory exists for every $k \in \mathbb{N}$, we must guarantee the trajectory is bounded (see, e.g., [19]). Thus, we additionally enforce the following constraints for every $k \in \mathbb{N}$:

$$-\boldsymbol{\omega}_{\max} \leq \boldsymbol{\omega}_{\mathcal{B}}(\tau_k) \leq \boldsymbol{\omega}_{\max}, \quad (33)$$

where $\boldsymbol{\omega}_{\max} \in \mathbb{R}^3_{++}$ is the maximum angular rate. The constraints (33) also ensure that the angular velocity lies in an allowable interval.

The original OCP is converted into Problem 2 for all $k \in \mathbb{N}$ and $\bar{k} \in \bar{\mathbb{N}}$, which is convex.

Problem 2.

$$\min_{t_f, \boldsymbol{u}, \boldsymbol{x}, \boldsymbol{\nu}, \boldsymbol{\lambda}, \boldsymbol{\mu}} t_f + J_{\mathrm{tr}}(\tilde{\boldsymbol{x}}, \boldsymbol{x}) + J_{\mathrm{vc}}(\boldsymbol{\nu}) \quad (34)$$

subject to:

$$\boldsymbol{x}_{\bar{k}+1} = A_{\bar{k}} \boldsymbol{x}_{\bar{k}} + \hat{B}_{\bar{k}} \boldsymbol{u}_{\bar{k}} + B_{\bar{k}} \boldsymbol{u}_{\bar{k}+1} + \boldsymbol{s}_{\bar{k}} t_f + \boldsymbol{c}_{\bar{k}} + \boldsymbol{\nu}_{\bar{k}}, \quad (35)$$

$$\boldsymbol{x}_0 = \boldsymbol{x}_i, \ \boldsymbol{x}_N = \boldsymbol{x}_f, \quad (36)$$

$$T_{\min} \leq T_i(\tau_k) \leq T_{\max} \quad \forall i \in \{1, 2, 3, 4\}, \quad (37)$$

$$-\boldsymbol{\omega}_{\max} \leq \boldsymbol{\omega}_{\mathcal{B}}(\tau_k) \leq \boldsymbol{\omega}_{\max}, \quad (38)$$

$$\lambda_{\bar{k}+1} = \lambda_{\bar{k}} - \mu_{\bar{k}}, \ -\mu_{\bar{k}} \leq 0, \ \lambda_0 = 1, \ \lambda_N = 0, \quad (39)$$

$$g(\tilde{\boldsymbol{y}}_{\bar{k}}) + \left.\frac{\partial g(\boldsymbol{y}_{\bar{k}})}{\partial \boldsymbol{y}_{\bar{k}}}\right|_{\tilde{\boldsymbol{y}}_{\bar{k}}} \delta \boldsymbol{y}_{\bar{k}} \leq 0, \quad (40)$$

Since the SCP algorithm obtains the converged solution by iteratively solving the convex subproblems, we introduce how to initialize and terminate the algorithm. The initial reference trajectory is not required to be dynamically feasible for the OCP [19], though poor initial guess may lead to more

iterations to converge. We initialize the time allocation of the waypoint by

$$\tilde{\boldsymbol{p}}_{\mathcal{I}}(\tau_{k_w}) = \boldsymbol{p}_w, \tag{41}$$

$$k_w \triangleq N \cdot \mathrm{int}\left(\frac{\|\boldsymbol{p}_{\mathcal{I}}(\tau_0) - \boldsymbol{p}_w\|_2}{\|\boldsymbol{p}_{\mathcal{I}}(\tau_0) - \boldsymbol{p}_w\|_2 + \|\boldsymbol{p}_{\mathcal{I}}(\tau_N) - \boldsymbol{p}_w\|_2}\right), \tag{42}$$

$$\tilde{\mu}_k = \begin{cases} 1, & \text{if } k = k_w \\ 0, & \text{otherwise} \end{cases} \quad \forall k \in \bar{\mathbb{N}}, \tag{43}$$

where $\mathrm{int}(x)$ returns the largest integer not greater than x. Then, the initial positions can be obtained by linear interpolation of the initial position, waypoint, and final position. Other initial guesses can be given for all $k \in \mathbb{N}$ by

$$\tilde{\boldsymbol{v}}_{\mathcal{I}}(\tau_k) = \mathbf{0},\ \tilde{\boldsymbol{q}}_{\mathcal{IB}}(\tau_k) = \boldsymbol{q}_i,\ \tilde{\boldsymbol{\omega}}_{\mathcal{B}}(\tau_k) = \mathbf{0}, \tag{44}$$

$$\tilde{\boldsymbol{u}}(\tau_k) = \begin{bmatrix} T_{\max} & T_{\max} & T_{\max} & T_{\max} \end{bmatrix}^{\mathrm{T}}, \tag{45}$$

$$\tilde{t}_f = \frac{\|\boldsymbol{p}_{\mathcal{I}}(\tau_0) - \boldsymbol{p}_w\|_2 + \|\boldsymbol{p}_{\mathcal{I}}(\tau_N) - \boldsymbol{p}_w\|_2}{1.5}, \tag{46}$$

where $\boldsymbol{q}_i$ is the identity quaternion.

The termination of the algorithm is predicated upon the satisfaction of designated convergence criteria.

$$L_{\mathrm{tr}}(\tilde{\boldsymbol{x}}, \boldsymbol{x}) \triangleq \sum_{k \in \mathbb{N}} \delta \boldsymbol{x}_k{}^{\mathrm{T}} \delta \boldsymbol{x}_k \leq \epsilon_{\mathrm{tr}}, \tag{47}$$

$$L_{\mathrm{vc}}(\boldsymbol{\nu}) \triangleq \sum_{k \in \bar{\mathbb{N}}} \|\boldsymbol{\nu}_k\|_1 \leq \epsilon_{\mathrm{vc}}, \tag{48}$$

where $\epsilon_{\mathrm{tr}} \in \mathbb{R}_{++}$ and $\epsilon_{\mathrm{vc}} \in \mathbb{R}_{++}$ are the user-specified tolerances. The convergence criteria, denoted by equations (48) and (47), play pivotal roles in ensuring the dynamic feasibility and convergence of the solution, respectively. Specifically, the criterion (48) is mandatory for ascertaining that the solution adheres to the dynamics of the system, rendering it dynamically feasible. On the other hand, the criterion (47) is indicative of the solution's convergence. This approach is encapsulated within the SCP algorithm, delineated as Algorithm 1.

Algorithm 1 SCP Algorithm.

INITIALIZATION
  compute $\tilde{\boldsymbol{z}}$ and $\tilde{\boldsymbol{\mu}}$ using (41-46)
  return $(\tilde{\boldsymbol{z}}, \tilde{\boldsymbol{\mu}})$
SCP
  while *not converged* **do**
    use $(\tilde{\boldsymbol{z}}, \tilde{\boldsymbol{\mu}})$ to obtain Problem 2
    solve Problem 2 to get $(\boldsymbol{x}, \boldsymbol{u}, t_f, \boldsymbol{\nu}, \boldsymbol{\lambda}, \boldsymbol{\mu})$
    if conditions (47) and (48) are met
      converged
    end if
    $(\tilde{\boldsymbol{z}}, \tilde{\boldsymbol{\mu}}) \leftarrow (\boldsymbol{x}, \boldsymbol{u}, t_f, \boldsymbol{\mu})$
  end while
  return $(\boldsymbol{x}, \boldsymbol{u}, t_f)$

IV. RESULTS

This section evaluates the SCP algorithm through both numerical simulations and real-world experiments, comparing it against a baseline method from [3] that tackles non-convex optimization for trajectory generation. Simulations were conducted in the simplified environment of [22], employing nominal MPC for trajectory tracking without accounting for unmodeled disturbances.

Simulations utilized a desktop with an Intel Core i7-12700F processor, while real-world tests were conducted on a laptop with an Intel Core i7-13650HX, using WiFi for quadrotor command transmission. The convex subproblems were solved using CVXPY and the MOSEK solver.

The quadrotor's configuration and the initial state are specified in Tab. I. Tab. II shows the parameters of the SCP algorithm used in our experiments.

TABLE I: Configuration and Initial State

Parameter	Value	Parameter	Value
m [kg]	0.85	$J_{\mathcal{B}}$ [kg$\cdot$ m^2]	diag([4, 4, 1])
l [m]	0.21	c_τ	0.5
$\boldsymbol{\omega}_{\max}$ [rad/s]	1.5	$[T_{\min}, T_{\max}]$ [N]	[0.00, 6.88]
$\boldsymbol{p}_{\mathcal{I}}(t_0)$ [m]	$[1\ 1\ 1]^{\mathrm{T}}$	$\boldsymbol{g}_{\mathcal{I}}$ [m/s^2]	$[0\ 0\ -9.81]^{\mathrm{T}}$
$\boldsymbol{v}_{\mathcal{I}}(t_0)$ [m/s]	$[0\ 0\ 0]^{\mathrm{T}}$	$\boldsymbol{\omega}_{\mathcal{B}}(t_0)$ [rad/s]	$[0\ 0\ 0]^{\mathrm{T}}$
$\boldsymbol{q}_{\mathcal{IB}}(t_0)$	$\boldsymbol{q}_i$	d_t [m]	0.3

TABLE II: Parameters of SCP Algorithm

Parameter	Value
w_{vc}	1×10^4
W_{tr}	diag([0.1, 0.1, 0.1, ζ, ζ, ζ, ζ, ζ, ζ, ζ, ζ, ζ, ζ])
ϵ_{vc}	1×10^{-3}
ϵ_{tr}	8×10^{-3}

$\zeta = 1 \times 10^{-3}$

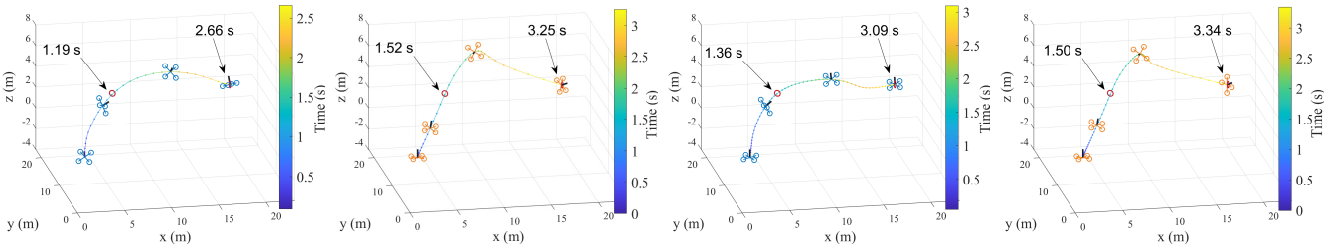
A. Scenario 1: Single Waypoint

In this scenario, the quadrotor is required to pass through a waypoint $\boldsymbol{p}_w = [6\ 12\ 4]^{\mathrm{T}}$ and finally reach a final position $\boldsymbol{p}_{\mathcal{I}}(t_f) = [20\ 20\ 2]^{\mathrm{T}}$. The performance results are reported in Tab. III. The generated trajectories are shown in Fig. 2.

TABLE III: Trajectory Performance Results of Scenario 1

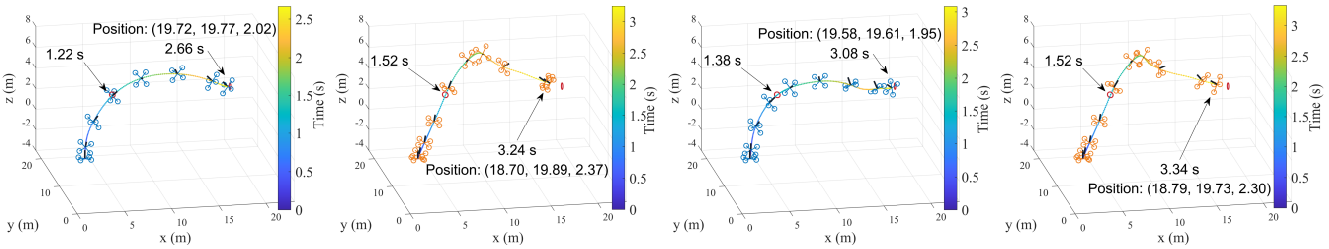
Method	N	Solve Time [s]	Waypoint Time [s]	t_f [s]
SCP	30	**0.98**	**1.19**	**2.66**
	60	2.93	1.36	3.09
[3]	30	10.86	1.52	3.25
	60	14.97	1.50	3.34

The SCP algorithm, utilizing 30 discrete nodes, demonstrates superior performance over the method in [3], as evidenced by Fig. 2 and the solve time comparisons in Tab. III. This efficiency stems from the tractability of convex optimization. Furthermore, the SCP method's solutions exhibit better optimality, a finding to be corroborated by subsequent MPC tracking tests in Fig. 3. Its ability to achieve high-quality solutions with fewer nodes is attributed to the effective discretization ensured by the propagation steps (18)



(a) (b) (c) (d)

Fig. 2: Trajectories generated for the single-waypoint flights. (a) The SCP algorithm with $N = 30$. (b) [3] with $N = 30$. (c) The SCP algorithm with $N = 60$. (d) [3] with $N = 60$.



(a) (b) (c) (d)

Fig. 3: MPC tracking results of the single-waypoint flights. (a) The SCP algorithm with $N = 30$. (b) [3] with $N = 30$. (c) The SCP algorithm with $N = 60$. (d) [3] with $N = 60$.

and (19), avoiding the convergence issues and suboptimal results associated with excessive discretization.

Trajectory feasibility and performance were further assessed using MPC tracking, with similar error levels observed across methods (see Tab. IV). The SCP algorithm achieved the shortest waypoint traversal time, aligning with reference trajectories (see Fig. 3). Note that the quadrotor will decelerate at the end of the trajectories, which is because the reference trajectories terminate at the terminal position, and then MPC will try to stop the quadrotor at the terminal position. Thus, the quadrotor does not reach the terminal position at t_f, the final position of the quadrotor is reported in Fig. 3. To better analyze the tracking performance, we will study the scenario of the quadrotor passing through multiple waypoints in the next subsection, where the quadrotor will fly longer such that the trajectory will not terminate prematurely.

TABLE IV: MPC Tracking Results of Scenario 1

Method	N	RMSE* [m]	Waypoint Time [s]
SCP	30	0.0410	**1.22**
	60	**0.0315**	1.38
[3]	30	0.0517	1.52
	60	0.0486	1.52

* RMSE: Root mean square error.

B. Scenario 2: Multiple Waypoints

In this scenario, the quadrotor is required to fly through three waypoints: $[6\ 12\ 4]^{\mathrm{T}}$, $[20\ 20\ 2]^{\mathrm{T}}$, and $[14\ 8\ 1]^{\mathrm{T}}$. The quadrotor should return to the initial position after passing through all the waypoints. The performance results are summarized in Tab. V. The generated trajectories are shown in Fig. 4

TABLE V: Trajectory Performance Results of Scenario 2

Method	N	Solve Time [s]	Waypoint Time [s]
SCP	60	**3.03**	(**1.19**, **2.66**, 4.30)
	120	5.88	(1.36, 3.09, 5.25)
[3]	60	200.50	(2.29, 4.69, 5.73)
	120	134.98	(1.51, 2.90, **4.01**)

As shown in Tab. V and Fig. 4, the SCP algorithm can obtain a solution much faster than the method used in [3]. As the problem becomes more complex, the solve time of [3] grows rapidly, due to solving a non-convex optimization problem being computationally intractable. Meanwhile, the SCP algorithm can efficiently solve the convex subproblems to obtain a satisfactory solution.

Remark 2. One may notice that the SCP method can make the quadrotor pass through the first two waypoints faster but falls behind at the third waypoint. The multi-stage policy used to deal with multiple waypoints may lead to a suboptimal solution. However, this does not mean that the method in [3] can always get a better optimal solution. Fig. 4d shows that the quadrotor will fly a circle and then return to the initial position after passing through the last path point, while Fig. 4a shows that the SCP method will make the quadrotor directly return to the initial position. As also shown by the color bars in Figs. 4d and 4a, the total flight time of

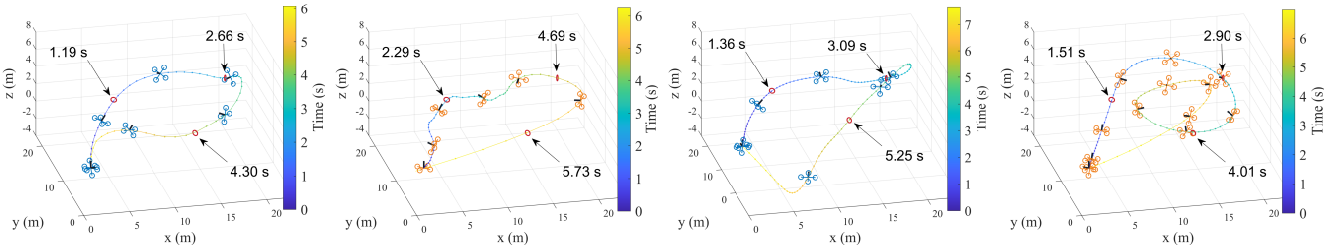


Fig. 4: Trajectories generated for the multiple-waypoint flights. (a) The SCP algorithm with $N = 60$. (b) [3] with $N = 60$. (c) The SCP algorithm with $N = 120$. (d) [3] with $N = 120$.

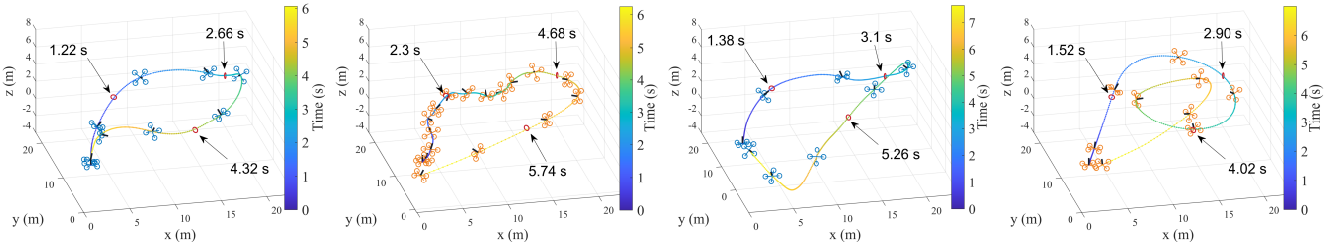


Fig. 5: MPC tracking results of the multiple-waypoint flights. (a) The SCP algorithm with $N = 60$. (b) [3] with $N = 60$. (c) The SCP algorithm with $N = 120$. (d) [3] with $N = 120$.

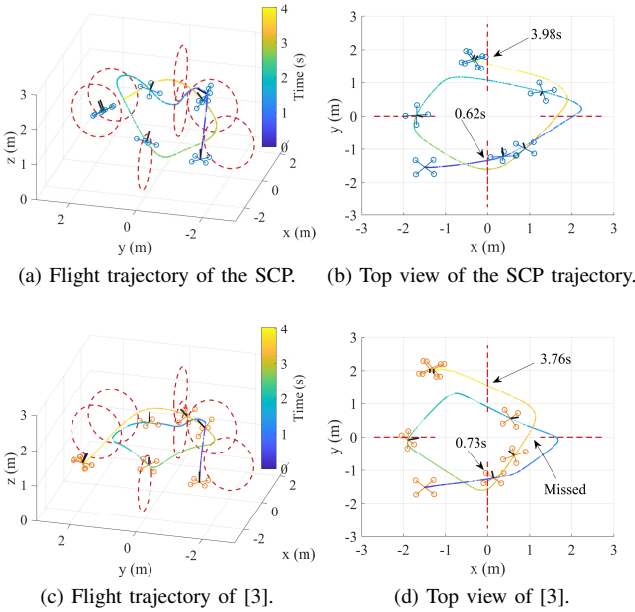
the SCP method is shorter than [3]. Therefore, both the SCP method and [3] can only ensure a locally optimal solution [3], [19], while the SCP method can further ensure higher computational efficiency.

The MPC tracking results are shown in Tab. VI and Fig. 5. The trajectory generated by the SCP method can achieve less tracking error, with the number of discrete nodes exerting minimal influence on this error metric. Conversely, a decrease in the number of discrete nodes increases the tracking error of the method used in [3] significantly. Similar to the previously mentioned, this is because the propagation steps (18) and (19) ensure better discretization quality. In addition, the time performance of MPC tracking is similar to the results in Tab. VI, which reflects the reliability of the SCP method.

TABLE VI: MPC Tracking Results of Scenario 2

Method	N	RMSE* [m]	Waypoint Time [s]
SCP	60	**0.0333**	(**1.22**, **2.66**, 4.32)
	120	0.0365	(1.38, 3.10, 5.26)
[3]	60	0.1896	(2.30, 4.68, 5.74)
	120	0.0456	(1.52, 2.90, **4.02**)

* The root mean square error.



(a) Flight trajectory of the SCP. (b) Top view of the SCP trajectory.

(c) Flight trajectory of [3]. (d) Top view of [3].

Fig. 6: Real-world experiment results.

C. Real-world Experiments

To further validate our proposed method, we conducted physical experiments using a Vicon motion capture system, with a quadrotor navigating through seven waypoints. As shown in Fig. 6, we compared the MPC tracking flight results obtained using the SCP method with those achieved by the method described in [3]. Initial observations reveal that the SCP method outperforms [3] at the first waypoint, taking only 0.62 seconds (see Fig. 6a) compared to 0.73 seconds (see Fig. 6c). However, by the final waypoint, SCP's performance dips, completing the task in 3.98 seconds (see

Fig. 6a), while the method in [3] finishes faster at 3.76 seconds (see Fig. 6c).

This initial advantage followed by a subsequent slowdown of the SCP method can be attributed to its reliance on linearization and initial guesses. While a good initial guess enables a strong start due to the drone's hovering state at the beginning, the simplicity of the initial guessing strategy becomes less effective as the flight progresses, resulting in a slower trajectory and ultimately falling behind the method in [3]. Moreover, the trajectory obtained by [3], as depicted in Fig. 6d, seeks the shortest possible path to expedite flight completion. Nevertheless, disturbances and model mismatches resulted in tracking errors, preventing the drone from closely approaching the penultimate waypoint, thus causing it to miss the designated penultimate waypoint (see Fig. 6d).

A critical advantage of the SCP method is its computational efficiency. It requires approximately 3.03 seconds to calculate the trajectory, significantly less than the 185.09 seconds needed by the method in [3]. Despite the SCP method's slower trajectory in later stages, its substantial lead in computational speed demonstrates a clear benefit, suggesting areas for further refinement in initial guess strategies and linearization dependency to enhance overall performance.

V. CONCLUSIONS

This paper presents the SCP algorithm for time-optimal quadrotor waypoint flight. The SCP algorithm is much more computationally efficient than directly solving the non-convex problem. Moreover, the proposed method simultaneously optimizes the waypoint allocation and the trajectory by using STCs. The proposed approach solves the multiple waypoint flight by using a multi-stage policy, which sequentially solves single waypoint flight to realize multiple waypoint flight. The multi-stage policy can still ensure a satisfactory solution while guaranteeing high computational efficiency. The proposed algorithm outperforms the existing method in terms of computational efficiency.

REFERENCES

[1] G. Loianno, D. Scaramuzza, and V. Kumar, "Special issue on high-speed vision-based autonomous navigation of UAVs," *Journal of Field Robotics*, vol. 35, no. 1, pp. 3–4, 2018.

[2] G. Loianno and D. Scaramuzza, "Special issue on future challenges and opportunities in vision-based drone navigation," *Journal of Field Robotics*, vol. 37, no. 4, pp. 495–496, 2020.

[3] P. Foehn, A. Romero, and D. Scaramuzza, "Time-optimal planning for quadrotor waypoint flight," *Science Robotics*, vol. 6, no. 56, p. eabh1221, 2021.

[4] D. Mellinger, N. Michael, and V. Kumar, "Trajectory generation and control for precise aggressive maneuvers with quadrotors," *The International Journal of Robotics Research*, vol. 31, no. 5, pp. 664–674, 2012.

[5] C. Richter, A. Bry, and N. Roy, "Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments," in *Robotics Research: The 16th International Symposium ISRR*, ser. Springer Tracts in Advanced Robotics, M. Inaba and P. Corke, Eds. Springer International Publishing, 2016, pp. 649–666.

[6] Z. Han, Z. Wang, N. Pan, Y. Lin, C. Xu, and F. Gao, "Fast-racing: An open-source strong baseline for SE(3) planning in autonomous drone racing," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 8631–8638, 2021.

[7] B. Zhou, F. Gao, L. Wang, C. Liu, and S. Shen, "Robust and efficient quadrotor trajectory generation for fast autonomous flight," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3529–3536, 2019.

[8] B. Zhou, J. Pan, F. Gao, and S. Shen, "Raptor: Robust and perception-aware trajectory replanning for quadrotor fast flight," *IEEE Transactions on Robotics*, vol. 37, no. 6, pp. 1992–2009, 2021.

[9] A. Romero, R. Penicka, and D. Scaramuzza, "Time-optimal online replanning for agile quadrotor flight," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 7730–7737, 2022.

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

[11] E. Kaufmann, L. Bauersfeld, A. Loquercio, M. Müller, V. Koltun, and D. Scaramuzza, "Champion-level drone racing using deep reinforcement learning," *Nature*, vol. 620, no. 7976, pp. 982–987, 2023.

[12] Y. Song, A. Romero, M. Müller, V. Koltun, and D. Scaramuzza, "Reaching the limit in autonomous racing: Optimal control versus reinforcement learning," *Science Robotics*, vol. 8, no. 82, p. eadg1462, 2023.

[13] B. Açıkmeşe and L. Blackmore, "Lossless convexification of a class of optimal control problems with non-convex control constraints," *Automatica*, vol. 47, no. 2, pp. 341–347, 2011.

[14] M. W. Harris and B. Açıkmeşe, "Lossless convexification of non-convex optimal control problems for state constrained linear systems," *Automatica*, vol. 50, no. 9, pp. 2304–2311, 2014.

[15] Y. Mao, M. Szmuk, and B. Açıkmeşe, "Successive convexification of non-convex optimal control problems and its convergence properties," in *2016 IEEE 55th Conference on Decision and Control (CDC)*, 2016, pp. 3636–3641.

[16] Y. Mao, D. Dueri, M. Szmuk, and B. Açıkmeşe, "Successive convexification of non-convex optimal control problems with state constraints," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 4063–4069, 2017.

[17] M. Szmuk, T. P. Reynolds, and B. Açıkmeşe, "Successive convexification for real-time six-degree-of-freedom powered descent guidance with state-triggered constraints," *Journal of Guidance, Control, and Dynamics*, vol. 43, no. 8, pp. 1399–1413, 2020.

[18] D. Malyuta, T. P. Reynolds, M. Szmuk, T. Lew, R. Bonalli, M. Pavone, and B. Açıkmeşe, "Convex optimization for trajectory generation: A tutorial on generating dynamically feasible trajectories reliably and efficiently," *IEEE Control Systems Magazine*, vol. 42, no. 5, pp. 40–113, 2022.

[19] R. Bonalli, T. Lew, and M. Pavone, "Analysis of theoretical and numerical properties of sequential convex programming for continuous-time optimal control," *IEEE Transactions on Automatic Control*, pp. 1–16, 2022.

[20] S. Boyd, S. P. Boyd, and L. Vandenberghe, *Convex optimization*. Cambridge University Press, 2004.

[21] C.-T. Chen, *Linear System Theory and Design*, 3rd ed., ser. The Oxford Series in Electrical and Computer Engineering. Oxford University Press, 1999.

[22] G. Torrente, E. Kaufmann, P. Fohn, and D. Scaramuzza, "Data-driven mpc for quadrotors," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3769–3776, 2021.