2024 IEEE International Conference on Robotics and Automation (ICRA 2024)
May 13-17, 2024. Yokohama, Japan

Time-Optimal Gate-Traversing Planner for Autonomous Drone Racing

Chao Qin[1], Maxime S.J. Michet[2], Jingxiang Chen[2], and Hugh H.-T. Liu[1]

***Abstract*— In drone racing, the time-minimum trajectory is affected by the drone's capabilities, the layout of the race track, and the configurations of the gates (e.g., their shapes and sizes). However, previous studies neglect the configuration of the gates, simply rendering drone racing a waypoint-passing task. This formulation often leads to a conservative choice of paths through the gates, as the spatial potential of the gates is not fully utilized. To address this issue, we present a time-optimal planner that can faithfully model gate constraints with various configurations and thereby generate a more time-efficient trajectory while considering the single-rotor-thrust limits. Our approach excels in computational efficiency which only takes a few seconds to compute the full state and control trajectories of the drone through tracks with dozens of different gates. Extensive simulations and experiments confirm the effectiveness of the proposed methodology, showing that the lap time can be further reduced by taking into account the gate's configuration. We validate our planner in real-world flights and demonstrate super-extreme flight trajectory through race tracks.**

SUPPLEMENTARY MATERIAL

Code: https://github.com/FSC-Lab/TOGT-Planner

I. INTRODUCTION

First-person-view (FPV) drone racing is a rapidly expanding e-sport that has attracted significant attention from the public. Fig. 1a shows a typical racing scenario. In brief, the goal is to operate the quadrotor through gates in a predetermined sequence at a minimum lap time. We refer to the problem of generating such a time-minimum trajectory as the time-optimal gate-traversing (TOGT) problem. Solving the TOGT problem for general race tracks while considering the full quadrotor dynamics is very challenging. On the one hand, a single race track may contain many gate types as shown in Fig. 1b, and the solver is required to accommodate all of them in one optimization problem. On the other hand, a substantial amount of constraints must be considered, such as the single-rotor thrust and gate constraints.

To manage computational complexity, state-of-the-art approaches [1]–[6] simplify the TOGT problem to a time-optimal waypoint-passing (TOWP) problem, where each gate's center is associated with a waypoint to guarantee successful gate traversal. While this methodology has already led to exceptional racing performance, it can only yield an approximate solution to the TOGT problem unless the gate is truly a point. It can be seen from Fig. 1 that the TOGT trajectory (left bottom) can always produce a shorter path and thus a faster lap time by fully utilizing the available free space within the gates. Therefore, to achieve true time optimality in racing, it is necessary to take the gate's shape and size into account. Additionally, concerns about computational efficiency arise when tackling general race tracks typically including dozens of gates. Current research mainly focuses on problems with less than 20 gates, with computation times that have already exceeded minutes or even hours [1]. Hence, their scalability still requires further investigation. To sum up, it remains an open problem to efficiently generate TOGT trajectories on general race tracks.

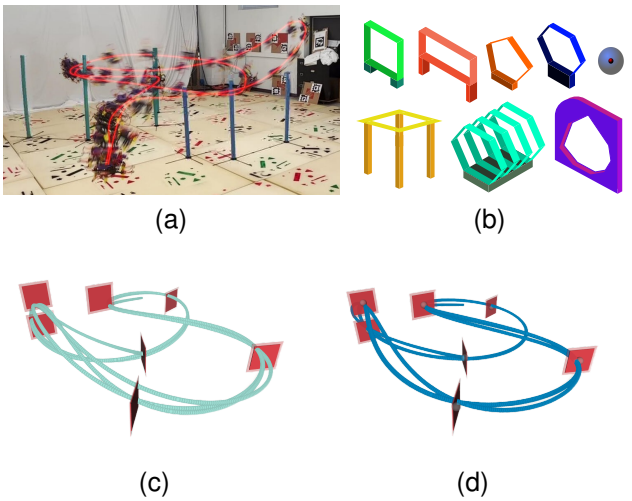

Fig. 1. (a) A time-optimal flight path generated by the proposed TOGT planner and executed in a motion capture room. (b) Our framework supports a wide range of gate shapes. (c-d) Comparison of trajectories obtained from our method with gate constraints (left figure) and with waypoint constraints (right figure). We can clearly see that the trajectory on the left is more time-efficient as it traverses the gates along their boundaries.

In this paper, we present a fast TOGT planner that can tackle dozens of gates with varying shapes in one single optimization problem within seconds. Our method supports all gate configurations listed in Fig. 1b. Moreover, to obtain high trajectory quality, we account for the full quadrotor dynamics and constraints such as the single-rotor-thrust limit. We apply polynomials as the trajectory representation and employ the differential-flat property of the quadrotor to avoid numerical integration of the system dynamics. Despite that polynomials are inherently smooth by definition, our approach yields a close or even shorter lap time than the state-of-the-art method [1] in most tested race tracks. This is because the spatial potential of gates is fully explored as

[1]Chao Qin and Hugh H.-T. Liu are with the University of Toronto Institute for Aerospace Studies (UTIAS), Toronto, Canada chao.qin@mail.utoronto.ca, hugh.liu@utoronto.ca

[2]Maxime S.J. Michet and Jingxiang Chen are with the University of Toronto Division of Engineering Science (EngSci), Toronto, Canada max.michet@mail.utoronto.ca, harryjx.chen@mail.utoronto.ca

shown in Fig. 1c. In addition, our planner can also address the TOWP problem by treating waypoints as special ball gates with a small radius (See Fig. 1d). Our contributions are summarized below:

- We propose a comprehensive formulation of the TOGT problem and provide a lightweight solver with a second-wise computation time.
- To our best knowledge, this is the first approach capable of handling race tracks with such a diverse array of gate types, and the first polynomial-based method that can enforce single-rotor-thrust constraints exactly.
- We validate the proposed approach through extensive simulation and real-world scenarios, demonstrating its great potential of generating racing benchmarks for general race tracks.

II. Related Work

Time-optimal planners for autonomous drone racing can be mainly categorized into discretization-based methods and polynomial-based methods.

Discretization-based methods employ time-discretized [7]–[9] or space-discretized trajectories [10]–[12] to represent the drone's state and control. Foehn et al. [1] addressed the time-allocation problem for discretized states by introducing a complementary progress constraint (CPC), leading to a tractable solution to the TOWP problem with the full quadrotor models. Penicka et al. [2] tackled the same problem using a sampling-based framework with hierarchical model refinement. However, these approaches may take minutes or even hours to obtain a solution. Zhou et al. [3] effectively reduced the computation burden by manually allocating waypoint constraints to specific states and refining the sampling period between consecutive waypoints. Romero et al. [5], [9] formulated the TOWP problem into a model predictive contouring control problem to enable real-time computation. Additionally, Spedicato et al. [10] showed that converting the quadrotor state space into a traverse dynamics formulation along a geometric reference path offers an alternative perspective to the drone racing problem. Following this technique, Arrizabalaga et al. [11] proposed a time-optimal planner for tunnel-like race tracks. While discretized controls can fit time-optimal control trajectories (e.g. bang-bang control [13]) very well, this representation suffers from the curse of dimensionality, especially when the required trajectory duration is considerable.

Polynomial-based methods are widely used in generating smooth collision-free trajectories for quadrotors [14]–[16], but they are less popular in time-minimum missions due to their limited capacity of representing trajectories with abrupt changes. Nonetheless, their ability to simulate trajectories using a minimal number of coefficients makes them well-suited for applications where computational efficiency outweighs time optimality, such as real-time path replanning. Han et al. [17] applied SE(3) constraints to ensure collision-free flight through narrow race courses and achieve a near-time-optimal solution by including time to the cost function. Wang et al. [18] proposed an online replanning framework to deal with dynamic gates during racing. However, these works do not aim for true time optimality as their objective functions trade-off between trajectory smoothness and time minimization. Ryou et al. [19] present one of the few works that pursue pure time minimization using a piece-wise polynomial representation, in which Bayesian optimization is utilized to learn the optimal time allocation over polynomial segments from multi-fidelity data obtained from analytic models. However, this approach does not realistically model the true constraint of limited single-rotor thrusts, and it is difficult to extend their scheme to racing scenarios. Our approach differs from these works in that we aim for pure time-minimum trajectory in racing while considering the authentic quadrotor model and actuation limits.

In both categories, there is a notable lack of research faithfully modeling geometric constraints of the gates; works in [1]–[3], [6], [9], [18] are indeed approximating gates by ball or waypoint constraints, while works in [10]–[12], [17] model the race track as a collision-free tunnel to avoid formulating gate constraints explicitly. Although Bos et al. [20] present a system to tackle circle and rectangle gates using a multi-stage optimal control framework, it is limited to two specific shapes, lacking the flexibility to handle other shapes commonly seen in real race tracks, such as pentagons. In contrast, our algorithm can readily deal with most gate types in racing and offer users a high degree of freedom to customize gates to meet practical needs.

III. Methodology

In this section, we detail the quadrotor model and gate constraints used, formulate the time-optimal gate-traversing problem, and introduce our solution.

A. Quadrotor Model

Let $\mathcal{F}^W$ and $\mathcal{F}^B$ denote the world frame and body frame, respectively. We define the state of the quadrotor as $\mathbf{x} = [\mathbf{p}^W, \mathbf{q}_{WB}, \mathbf{v}^W, \boldsymbol{\omega}^B]^T \in \mathbb{R}^n$ where $n=13$, corresponding to position expressed in $\mathcal{F}^W$, unit quaternion rotation from $\mathcal{F}^B$ to $\mathcal{F}^W$, velocity in $\mathcal{F}^W$, and angular rate in $\mathcal{F}^B$. Let $\mathbf{\Lambda}(\mathbf{q}) \in \mathbb{R}^{4\times4}$ represent the quaternion product matrix and $\mathbf{R}(\mathbf{q}) \in \mathbb{R}^{3\times3}$ the rotation matrix of the corresponding quaternion. We will omit the frame indices from here on as they remain consistent throughout the description. The control includes commanded thrusts of four motors, $\mathbf{u} = [f_1, f_2, f_3, f_4]^T \in \mathbb{R}^m$ where $m=4$. With a slight notation abuse, we also use m to represent the quadrotor's mass. In addition, we use $\mathbf{J}$, l, and c_τ to denote the inertia, arm length, and torque constant of the quadrotor, respectively. The body torque is expressed as $\boldsymbol{\tau} \in \mathbb{R}^3$ and the gravity vector in $\mathcal{F}^W$ as $\mathbf{g} \in \mathbb{R}^3$. Now the equation of motion can be written as:

$$\begin{aligned} &\dot{\mathbf{p}} = \mathbf{v}, && \dot{\mathbf{q}} = \tfrac{1}{2}\mathbf{\Lambda}(\mathbf{q}) \begin{bmatrix} 0 \\ \boldsymbol{\omega} \end{bmatrix}, \\ &\mathbf{v} = \mathbf{g} + \tfrac{1}{m}\mathbf{R}(\mathbf{q})\mathbf{F}_T, && \dot{\boldsymbol{\omega}} = \mathbf{J}^{-1}(\boldsymbol{\tau} - \boldsymbol{\omega} \times \mathbf{J}\boldsymbol{\omega}), \end{aligned} \tag{1}$$

where

$$\mathbf{F}_T = \begin{bmatrix} 0 \\ 0 \\ \sum f_i \end{bmatrix}, \ \boldsymbol{\tau} = \begin{bmatrix} l(f_1+f_2-f_3-f_4) \\ l(-f_1+f_2+f_3-f_4) \\ c_\tau(f_1-f_2+f_3-f_4) \end{bmatrix}. \tag{2}$$

To ensure feasible maneuvers, we impose force constraints [21] on each motor, $f_{min} \leq f_i \leq f_{max}$, as well as body rate constraints $|\boldsymbol{\omega}| < \boldsymbol{\omega}_{max}$. Lastly, we encapsulate all dynamic equations into $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u})$ and all state-input constraints into $\mathbf{h}(\mathbf{x}, \mathbf{u}) \leq \mathbf{0}$.

B. Definition of Gate

Throughout this paper, a gate is defined as a three-dimensional region enclosed by the corresponding geometrical shape. Let $\mathcal{G}^i \subset \mathbb{R}^3$ denote the space of the i-th gate. It is considered being traversed if there exists a point $\mathbf{p} \in \mathcal{G}^i$ in the state trajectory. We introduce inequalities $\mathbf{h}^i_{\mathcal{G}}(\mathbf{p}) \leq \mathbf{0}$ to indicate a successful traversal. Two base gate classes are considered below, using which all gates listed in Fig. 1b can be properly constructed.

The first base class is called the ball gate. It can be used to specify checkpoints that the drone must pass in racing. Let $\mathbf{p}_w \in \mathbb{R}^3$ denote the center of the ball and $\delta \geq 0$ its radius, which respectively correspond to the location and tolerance range of a waypoint. The enclosed space is given as:

$$\mathcal{G}_{\mathcal{B}} = \{\mathbf{p} \in \mathbb{R}^3 | \|\mathbf{p} - \mathbf{p}_w\|_2 \leq \delta\}, \tag{3}$$

The second base class is called the convex polygon/polyhedron gate. While the polygon can be used to describe gate shapes like rectangles, pentagons, and so on, the polyhedron can be utilized to represent collision-free space of some tunnels. They share the same expression as

$$\mathcal{G}_{\mathcal{P}} = \{\mathbf{p} \in \mathbb{R}^3 | \mathbf{A}\mathbf{p} \leq \mathbf{b}\}, \tag{4}$$

where $\mathbf{A}$ and $\mathbf{b}$ defines the half-spaces enclosing the gate. From here on, we will drop the subscripts, as the gate shape information is implied by their indices.

By concatenating multiple gates, we can create tunnels in the race track. For instance, a pentagonal tunnel can be constructed by two pentagons made by the polygon gate and one pentagonal prism made by the polyhedron gate. The order of these sub-gates can designate its entrance and exit.

C. Time-Optimal Gate-Traversing Problem

Consider an environment with L gates, denoted as $\mathcal{G}^1, \mathcal{G}^2, ..., \mathcal{G}^L$, and a quadrotor at the initial state $\bar{\mathbf{x}}_0$. The objective of TOGT can be described as finding time-minimum trajectories passing through these gates and finally reaching the terminal state at $\bar{\mathbf{x}}_f$. We use $\mathbf{x} : [0, t_f] \mapsto \mathbb{R}^n$ and $\mathbf{u} : [0, t_f] \mapsto \mathbb{R}^m$ to express the corresponding state and control trajectory, respectively, where t_f denotes the total time of the trajectory. The TOGT problem is given as:

$$\min_{\mathbf{x},\mathbf{u},t_f} \quad t_f \tag{5a}$$

$$\text{s.t.} \quad \mathbf{x}(0) = \bar{\mathbf{x}}_0,\ \mathbf{x}(t_f) = \bar{\mathbf{x}}_f, \tag{5b}$$

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}),\ \mathbf{h}(\mathbf{x}, \mathbf{u}) \leq \mathbf{0}, \tag{5c}$$

$$\exists\ 0 < t_1 < t_2 < ... < t_L < t_f, \tag{5d}$$

$$\mathbf{h}_{\mathcal{G}^i}(\mathbf{p}_{\mathbf{x}(t_i)}) \leq \mathbf{0},\ 1 \leq i \leq L, \tag{5e}$$

where t_i is the time passing the i-th gate and $\mathbf{p}_{\mathbf{x}(t_i)}$ is the position member of $\mathbf{x}(t_i)$.

D. Problem Transformation

We can see that the inequality (5e) is directly related to $\mathbf{x}(t_i)$ and consequently indirectly linked to t_i and t_f, making it a time-dependent constraint. To ease the computation, it is beneficial to perform certain transformations to decouple the time from the gate constraints.

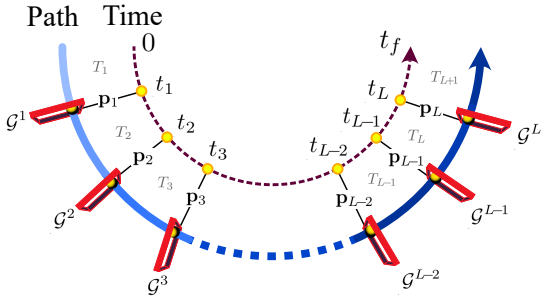

Fig. 2. Illustration of the segmented trajectories based on the gate order.

To begin with, we segment the trajectory into $(L+1)$ pieces according to the gate order and assign each gate a waypoint, as shown in Fig. 2. Let $\mathbf{P} = [\mathbf{p}_1^T, \mathbf{p}_2^T, ..., \mathbf{p}_L^T]^T \in \mathbb{R}^{3L}$ denote the vector storing all resulting waypoints. The times allocated for all segments are given as $\mathbf{T} = [T_1, T_2, ..., T_L, T_{L+1}]^T \in \mathbb{R}^{L+1}_{>}$. We can derive the new expression for the total trajectory duration as $t_f \triangleq T_\Sigma = \sum_{k=1}^{L+1} T_k$ and the traversal time of the i-th gate as $t_i \triangleq t_{\Sigma_i} = \sum_{k=1}^{i} T_k$.

After the waypoints are given, we put all feasible time allocation vectors for this specific waypoint flight into a set:

$$\begin{aligned} \mathcal{T}(\mathbf{P}) = \{\mathbf{T} \in \mathbb{R}^{L+1}_{>} \mid \exists \mathbf{x}, \mathbf{u} : [0, T_\Sigma] \to \mathbb{R}^n, \mathbb{R}^m \\ \text{s.t. } \mathbf{x}(0) = \bar{\mathbf{x}}_0,\ \mathbf{x}(T_\Sigma) = \bar{\mathbf{x}}_f \\ \dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}), \mathbf{h}(\mathbf{x}, \mathbf{u}) \leq \mathbf{0} \\ \mathbf{p}_{\mathbf{x}(t_{\Sigma_i})} = \mathbf{p}_i,\ 1 \leq i \leq L\}. \end{aligned} \tag{6}$$

Note that $\mathcal{T}(\mathbf{P})$ encompasses all possible functionals to represent $\mathbf{x}$ and $\mathbf{u}$. According to this definition, if $\mathbf{T} \in \mathcal{T}(\mathbf{P})$, there must exist a dynamically feasible trajectory passing all specified waypoints at the specified times.

Then, we can rewrite (5) as

$$\min_{\mathbf{P},\mathbf{T}} \quad T_\Sigma + I_{\mathcal{T}(\mathbf{P})}(\mathbf{T}) \tag{7a}$$

$$\text{s.t.} \quad \mathbf{h}_{\mathcal{G}^i}(\mathbf{p}_i) \leq \mathbf{0},\ 1 \leq i \leq L, \tag{7b}$$

where

$$I_{\mathcal{T}(\mathbf{P})}(\mathbf{T}) = \begin{cases} 0 & \text{if } \mathbf{T} \in \mathcal{T}(\mathbf{P}), \\ \infty & \text{if } \mathbf{T} \notin \mathcal{T}(\mathbf{P}). \end{cases} \tag{8}$$

Checking the equivalence between (5) and (7) is pretty straightforward and we omit the proof here.

It can be seen that the inequality (7b) is no longer time-dependent, which is key for us to eliminate it in Section III-F. However, solving $\mathcal{T}(\mathbf{P})$ and subsequently $I_{\mathcal{T}(\mathbf{P})}(\mathbf{T})$ is still intractable, because we cannot travel the entire functional space due to the curse of dimentionality [22]. Our strategy is to focus on one specific functional and aim for an approximate solution, $I_{\hat{\mathcal{T}}(\mathbf{P})}(\mathbf{T}) \approx I_{\mathcal{T}(\mathbf{P})}(\mathbf{T})$. In the next section, we will discuss how to select a suitable functional.

E. Functional Selection and Calculation of $I_{\hat{\mathcal{T}}(\mathbf{P})}(\mathbf{T})$

The most suitable functional varies with the design objective. If the goal is true time optimality like in [1], the piece-wise constant control becomes a viable candidate, albeit with the drawback of large computation costs. In our case, to enable rapid trajectory generation, we prioritize computational efficiency over the utmost time optimality. Therefore, we prefer a functional that (i) can eliminate as many constraints in (6) as possible and (ii) has a minimal number of coefficients, using which the problem complexity can be greatly alleviated.

The minimum-control (MINCO) trajectory functional proposed in [23] meets the above criteria very well. This functional in essence is a set of piece-wise polynomials that minimize energies (e.g., jerk). However, we apply it primarily as a lightweight tool to compute polynomial coefficients, rather than taking energy minimization as a design objective like in [16]–[18]. By parameterizing the quadrotor's flat output trajectory using piece-wise polynomials obtained from this method, we not only can eliminate constraints on system dynamics, initial state, and terminal state but also ensure precise waypoint traversal. These properties enable it to remove a large number of constraints from (6), leaving only the state-input constraints. Another advantage of this functional is that solving the polynomial coefficients is as straightforward as solving a matrix equation, making the trajectory generation and evaluation very efficient.

Let $\mathbf{y}:[0,t_f]\rightarrow\mathbb{R}^m$ denote the flat output trajectory which is implemented as piece-wise polynomials with an order of $s=5$. We have $\mathbf{y}(t)=[\mathbf{p}^T,\psi]^T$ where ψ is the yaw angle [24]. The state and control trajectories, along with their associated constraints, can now be expressed as functions of $\mathbf{y}$ and its derivatives up to the s-th order:

$$\mathbf{x}=\Psi_{\mathbf{x}}(\mathbf{y},\dot{\mathbf{y}},...,\mathbf{y}^{(s-1)})\triangleq\Psi_{\mathbf{x}}(\mathbf{y}^{[s-1]}), \tag{9}$$

$$\mathbf{u}=\Psi_{\mathbf{u}}(\mathbf{y},\dot{\mathbf{y}},...,\mathbf{y}^{(s)})\triangleq\Psi_{\mathbf{u}}(\mathbf{y}^{[s]}), \tag{10}$$

$$\mathbf{h}(\mathbf{x},\mathbf{u})=\mathbf{h}_{\Psi}(\mathbf{y}^{[s]})\leq\mathbf{0}, \tag{11}$$

where $\Psi_{\mathbf{x}}:\mathbb{R}^{m\times(s-1)}\rightarrow\mathbb{R}^n$ and $\Psi_{\mathbf{u}}:\mathbb{R}^{m\times s}\rightarrow\mathbb{R}^n$ are defined in [24]–[26]. Since (11) is the only remaining constraint in (6), any violation at an arbitrary time would yield $\mathbf{T}\notin\hat{\mathcal{T}}(\mathbf{P})$. Now we define $I_{\hat{\mathcal{T}}(\mathbf{P})}(\mathbf{T})$ as

$$\begin{aligned}I_{\hat{\mathcal{T}}(\mathbf{P})}(\mathbf{T})&=\int_0^{T_\Sigma}\max[\mathbf{h}_\Psi(\mathbf{y}^{[s]}(t)),\mathbf{0}]^3dt,\\&\approx\sum_{i=1}^{L+1}\sum_{j=0}^{\kappa_i}\max[\mathbf{h}_\Psi(\mathbf{y}^{[s]}(t_{i-1}+j\Delta t_i)),\mathbf{0}]^3\Delta t_i,\end{aligned} \tag{12}$$

where κ_i the sample resolution, $t_0=0$, and $\Delta t_i=T_i/\kappa_i$. It can be observed that $I_{\hat{\mathcal{T}}(\mathbf{P})}(\mathbf{T})$ returns zero if all constraints are met and nonzero otherwise, functioning just like $I_{\mathcal{T}(\mathbf{P})}(\mathbf{T})$. Therefore, when $I_{\hat{\mathcal{T}}(\mathbf{P})}(\mathbf{T})$ is small, we consider the state-input constraints to be satisfied.

F. Gate and Time Constraints Elimination

The gate constraints in (7b) can become high-dimensional when dealing with intricate gate shapes or a substantial number of gates. The inherent positivity constraints in $\mathbf{T}$ also bring an additional $L+1$ constraints to the system. These constraints not only introduce a large number of Lagrange multipliers into the solver but also contribute to slow convergence [27]. Therefore, it is worthwhile to seek ways to eliminate these constraints. We employ the change-of-variable technique proposed in [23] for this purpose.

We know from [23] that for an arbitrary ball gate, there exist a smooth surjection $\mathbf{g}_B(\cdot):\mathbb{R}^4\rightarrow\mathcal{G}_{\mathcal{B}}$:

$$\mathbf{g}_B(\mathbf{d})=\mathbf{p}_w+\left[\frac{2\delta\mathbf{d}}{\mathbf{d}^T\mathbf{d}+1}\right]_3, \tag{13}$$

such that optimization over the new variables $\mathbf{d}$ implicitly satisfies the constraint $\mathcal{G}_{\mathcal{B}}$. Note that $[\cdot]_v$ returns the first v entries of the input vector and here $v=3$. Similarly, for a convex polygon/polyhedron gate with v corners, there also exist a smooth surjection: $\mathbf{g}_{\mathcal{P}}(\cdot):\mathbb{R}^v\rightarrow\mathcal{G}_{\mathcal{P}}$

$$\mathbf{g}_{\mathcal{P}}(\mathbf{d})=\mathbf{o}+\mathbf{V}\left[\frac{[\mathbf{d}]^2}{(\mathbf{d}^T\mathbf{d})^2}\right]_v, \tag{14}$$

that can serve the same purpose, where $\mathbf{o}\in\mathbb{R}^3$ is the origin of the barycentric coordinate [28] of the gate and $\mathbf{V}\in\mathbb{R}^{3\times v}$ its basis-vector matrix. By employing a set of new variables, $\mathbf{D}=[\mathbf{d}_1^T,\mathbf{d}_2^T,...,\mathbf{d}_L^T]^T$, as the underlying implementation of the waypoints, we ensure that these waypoints can always stay within their corresponding gates. We adopt a similar process to eliminate constraints on $\mathbf{T}$, which yields a set of new variables, $\mathbf{K}\in\mathbb{R}^{L+1}$. Now the waypoint and time allocation are obtained from $\mathbf{P}(\mathbf{D})$ and $\mathbf{T}(\mathbf{K})$, respectively.

G. Unconstrained Optimization

Utilizing new decision variables, we arrive at an unconstrained optimization problem below:

$$\min_{\mathbf{D},\mathbf{K}}\quad T_\Sigma(\mathbf{K})+I_{\hat{\mathcal{T}}(\mathbf{P}(\mathbf{D}))}(\mathbf{T}(\mathbf{K})). \tag{15}$$

We derive the analytical gradient of (15) and solve it by using the L-BFGS algorithm [29]. After obtaining $\mathbf{P}$ from $\mathbf{D}$ and $\mathbf{T}$ from $\mathbf{K}$, we first compute the coefficients of the MINCO functional, followed by calculating the state and control trajectories via (9) and (10).

TABLE I
QUADROTOR PARAMETERS

	m [kg]	l [m]	$\mathbf{J}_{diag}$ [gm^2]	f_{max} [N]	c_τ [1]	$\boldsymbol{\omega}_{max}$ [rad s $^{-1}$]
QuadA	0.85	0.15	[1,1,1.7]	6.88	0.05	[15, 15, 3]
QuadB	1.05	0.125	[2.5,2.1,4.3]	6.375	0.022	[8, 8, 3]

IV. RESULTS

In this section, we evaluate the performance of the proposed planner in two separate modes, gate-traversing mode (referred to as TOGT) and waypoint-passing mode (abbreviated as TOGT-WP). We compare them to the state-of-the-art approach, CPC [1], which employs the same system model and constraint as our approach. The computational unit is an Intel Core i7-12650H CPU, and an optimization attempt is considered unsuccessful if it fails to converge within 8 hours.

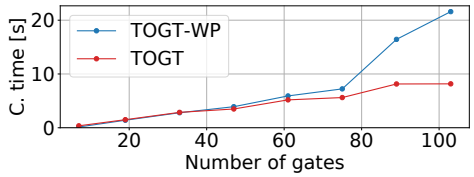

Fig. 3. Our planner presents a nearly linear increase in computation time as the number of gates increases.

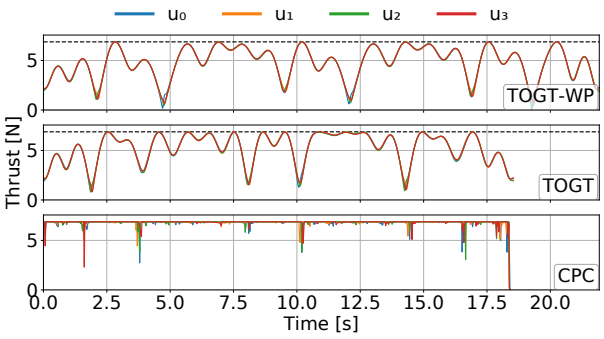

Fig. 4. Comparison of single-rotor thrust trajectories from our method (the TOGT and TOGT-WP) and CPC. Although the TOGT suffers from inherent smoothness, leveraging the spatial potential of gates allows it to produce almost the same flight time as in the CPC.

A. Solution Quality, Computation Time, and Scalability

We simulate a racing environment described in [30] which included seven square gates with a side length of 2.4 m. In the TOGT setup, a safe margin of 0.3 m is considered to avoid the collision, leading to an actual side length of 2.1 m. To ensure that the underlying problems for the CPC and TOGT-WP are identical, we utilize the same waypoints, which are set as the centers of the gates, and employ the same tolerance range of 0.3 m. The quadrotor parameters are provided in the QuadA configuration in Tab. I. Both the initial and terminal states are set as hovering with zero velocity. We generate trajectories with varying laps by concatenating the gates for a single lap multiple times and solving the full multi-lap problem in a single optimization.

TABLE II

COMPARISON OF COMPUTATION TIMES AND TRAJECTORY DURATIONS

G. Num.	CPC [1]		**TOGT-WP**		**TOGT**	
	c. time [s]	t_f [s]	c. time [s]	t_f [s]	c. time [s]	t_f [s]
7	466	6.85	**0.14**	8.45	**0.36**	7.53
19	2718	18.41	**1.39**	21.93	**1.50**	18.54
33	9428	31.22	**2.79**	36.71	**2.86**	**30.87**
47	28405	44.01	**3.91**	51.51	**3.49**	**43.25**
61	–	–	**5.91**	66.30	**5.18**	**55.61**
75	–	–	**7.22**	81.08	**5.61**	**68.00**

Tab. II compares the timings and computation times of different methods with laps from 1 (7 gates) to 10.5 (75 gates). Notably, our planner is significantly faster than the CPC. While the CPC takes 7.9 hours in the task of 43 gates, our method completes the computation within 4 s. As the number of gates grows, the limitations of the CPC become increasingly evident, eventually causing its first failure in the task of 55 gates. In contrast, our method exhibits high efficiency and numerical stability across all tasks. From Fig. 3, we observe that the computation times increase nearly linearly as the number of gates increases.

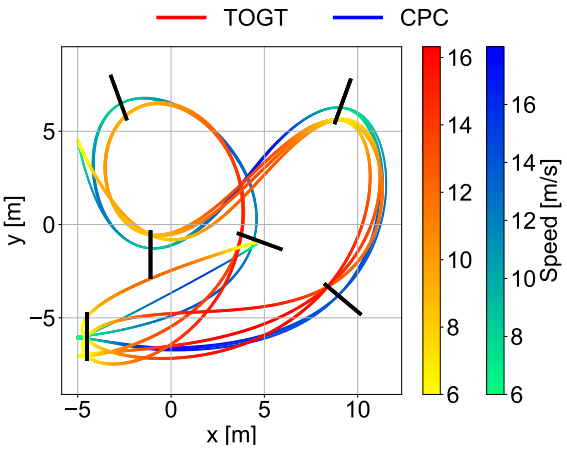

Fig. 5. Comparison of trajectories obtained from the TOGT and CPC in the task of traversing 19 square gates located at 7 separate locations. Note that the gate in the lower-left corner comprises two vertically stacked gates. The trajectories are colored based on their speed profiles.

We found that the CPC consistently outperforms the TOGT-WP in terms of solution quality, with lap times 14.6% to 18.9% shorter. The reason is revealed in Fig. 4, where the CPC always maintains a maximum thrust output, whereas the TOGT-WP only occasionally reaches these limits. However, this disadvantage is mitigated or even reversed when we turn to the TOGT. From Tab. II, we observe a notable reduction in lap times compared to the TOGT-WP. In tasks with more than 31 gates, the TOGT even begins to outperform the CPC, with lap times approximately 1% shorter. Fig. 5 demonstrates that the TOGT can keep pushing the flight aggressiveness until it reaches either the gate boundary or the drone's actuation limit. This capability results in a path that is 16.1 m shorter and a flight time that is 0.35 s shorter compared to the CPC in the task with 19 gates, despite its limited ability to maintain maximum thrust outputs.

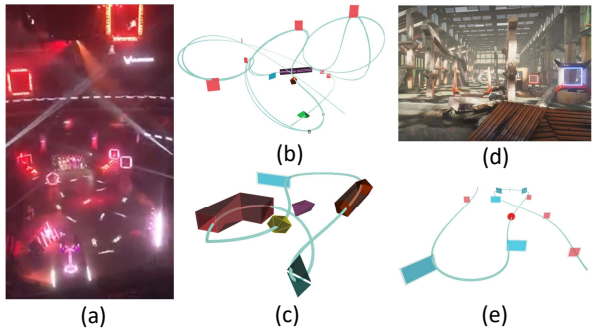

Fig. 6. Trajectories generated by our method for general race tracks: (a-b) the FedExForum race track; (c) a complex race track with 6 different gates; (d-e) the FlightGoggle race track.

B. Application to General Race Tracks

To demonstrate its generalizability, we apply the TOGT to various race tracks, including the FedExForum track from the 2021-22 DRL World Championship, the FlightGoggle race track from the 2019 AlphaPilot Challenge [4], and an imaginary race track showing how flexible our method can be. The generated paths are given in Fig. 6, Note that the setup remains the same as in the last section.

C. Experiments

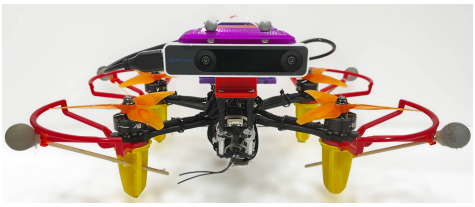

Fig. 7. Platform used for the real-world experiments equipped with a Jetson Xavier computer, an Intel Realsense T265 tracking camera, a PixRacer flight controller, and infrared-reflective markers for motion capture.

The labortory experimentation is performed in a motion capture room with a $4 \times 4 \times 2$ m^3 tracking volume. Our hardware configuration is shown in Fig. 7. The onboard computer runs the *Agilicious* autopilot system [31], which includes an extended Kalman filter that fuses VICON data at 100 Hz and visual-inertial odometry at 200 Hz to obtain full-state estimates, along with a model predictive controller (MPC) operating at 100 Hz. Detailed controller setups are provided in [30]. It is worth noting that the PixRacer controller can only accept body rates and collective thrust as the lowest-level control inputs. Therefore, following [1], [9], we extract the body rates from the MPC states and compute the collective thrust by using the control commands. To maintain controllability under disturbances, we generate the trajectory with a slightly lower thrust bound than what the platform can deliver, as depicted in the QuadB configuration.

We first compare the time optimality between the TOGT and TOGT-WP. The race track is illustrated in Fig. 8a which contains a total of 10 gates, including 7 square gates with a side length of 1.0 m and 3 ball gates with a radius of 0.1 m. A margin of 0.6 m is set for each square gate to account for the actual drone size (≈0.34 m) and the frame width (≈0.26 m). In the TOGT-WP setup, a tolerance range of 0.1 m is set for each waypoint. The trajectories from both methods are provided in Fig. 8d. The results suggest that the TOGT can generate more aggressive racing behavior in terms of space utilization. The required lap time is also shorten from 6.14 s to 5.96 s without causing any collision to gates.

In the second experiment, we evaluate the racing performance on a narrow race track made by 4 identical square tunnels and 1 ball gate. The size and margin parameters for the square and ball are the same as in the first experiment, and the depth for each tunnel is 2 m. As shown in Fig. 8b, the drone successfully completes the race within 5.56 s, with an average tracking RMSE of 0.15 m. The maximum

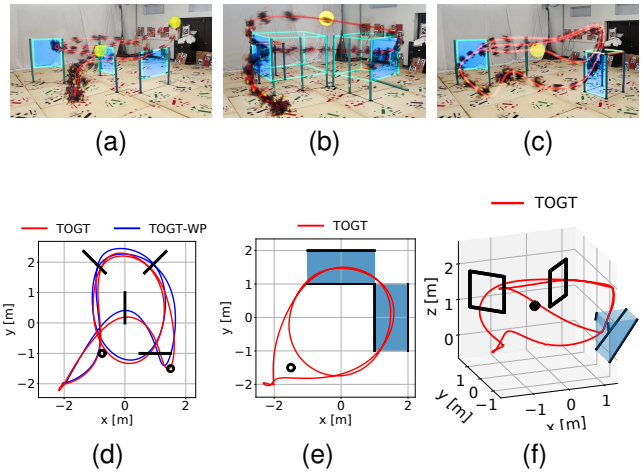

Fig. 8. Real-world flight trajectories in: (a,d) a course with 4 square gates; (b,e) a course with 2 tunnels; and (c,f) a course with a dive gate.

speed reached 6.83 m/s, which is a considerable value in such a small environment. Fig. 8e plots the trajectory from a top view. It can be seen that the drone is navigating through the most time-efficient corners within the tunnel, showcasing exceptional spatial utilization in such a narrow race track.

In the final experiment, we demonstrate the applicability of our planner in challenging race tracks. As illustrated in Figure 8c, the drone is required to dive through the rightmost gate from above and exit through the left face. Accurate modeling of gate constraints is crucial in such scenarios, as collisions can easily happen once the drone dives into the gate. We model the dive gate as a tunnel with two faces. Besides a upper square gate and a left rectangular gate, there are two polyhedron gates to specify the collision-free region inside the dive gate, as shown in Fig. 8f. The results show that our method can consistently accomplish diving flight with a speed of 2.90 m/s during the diving, which validates the remarkable versatility of our method.

V. CONCLUSIONS

We present an efficient time-optimal trajectory planner that can faithfully account for the racing gate's shape and size. Compared to existing works, our framework can better utilize the free space of the gates to find the most time-efficient path and thereby yield a comparable time optimality to discretization-based methods. We validate our method via real-world experiments and demonstrate superior performance. In the future work, we will incorporate gates with irregular structures such that all types of gates or corridors appearing in drone racing can be well-modeled.

ACKNOWLEDGMENT

The authors would like to acknowledge the sponsorship by the Natural Sciences and Engineering Research Council of Canada (NSERC) under grant RGPIN-2023-05148. Additionally, we express our gratitude for the experimental support provided by the Learning Systems & Robotics Lab (formerly Dynamic Systems Lab) at the UTIAS.

REFERENCES

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

[2] R. Penicka and D. Scaramuzza, "Minimum-time quadrotor waypoint flight in cluttered environments," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 5719–5726, 2022.

[3] Z. Zhou, G. Wang, J. Sun, J. Wang, and J. Chen, "Efficient and robust time-optimal trajectory planning and control for agile quadrotor flight," *arXiv preprint arXiv:2305.02772*, 2023.

[4] P. Foehn, D. Brescianini, E. Kaufmann, T. Cieslewski, M. Gehrig, M. Muglikar, and D. Scaramuzza, "Alphapilot: Autonomous drone racing," *Autonomous Robots*, vol. 46, no. 1, pp. 307–320, 2022.

[5] 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.

[6] O. E. Ramos, "Minimum-time optimal control for a quadcopter trajectory through gates with arbitrary pose," in *2021 IEEE XXVIII International Conference on Electronics, Electrical Engineering and Computing (INTERCON)*. IEEE, 2021, pp. 1–4.

[7] L.-C. Lai, C.-C. Yang, and C.-J. Wu, "Time-optimal control of a hovering quad-rotor helicopter," *Journal of Intelligent and Robotic Systems*, vol. 45, pp. 115–135, 2006.

[8] Y. Shen, J. Xu, J. Zhou, D. Xu, F. Zhao, J. Chen, and S. Li, "Aggressive trajectory generation for a swarm of autonomous racing drones," *arXiv preprint arXiv:2303.00851*, 2023.

[9] 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.

[10] S. Spedicato and G. Notarstefano, "Minimum-time trajectory generation for quadrotors in constrained environments," *IEEE Transactions on Control Systems Technology*, vol. 26, no. 4, pp. 1335–1344, 2017.

[11] J. Arrizabalaga and M. Ryll, "Towards time-optimal tunnel-following for quadrotors," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 4044–4050.

[12] W. Van Loock, G. Pipeleers, and J. Swevers, "Time-optimal quadrotor flight," in *2013 European Control Conference (ECC)*. IEEE, 2013, pp. 1788–1792.

[13] M. Hehn, R. Ritz, and R. D'Andrea, "Performance benchmarking of quadrotor systems using time-optimal control," *Autonomous Robots*, vol. 33, pp. 69–88, 2012.

[14] 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.

[15] 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*. Springer, 2016, pp. 649–666.

[16] J. Ji, N. Pan, C. Xu, and F. Gao, "Elastic tracker: A spatio-temporal trajectory planner for flexible aerial tracking," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 47–53.

[17] 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.

[18] Q. Wang, D. Wang, C. Xu, A. Gao, and F. Gao, "Polynomial-based online planning for autonomous drone racing in dynamic environments," *arXiv preprint arXiv:2306.14461*, 2023.

[19] G. Ryou, E. Tal, and S. Karaman, "Multi-fidelity black-box optimization for time-optimal quadrotor maneuvers," *The International Journal of Robotics Research*, vol. 40, no. 12-14, pp. 1352–1369, 2021.

[20] M. Bos, W. Decré, J. Swevers, and G. Pipeleers, "Multi-stage optimal control problem formulation for drone racing through gates and tunnels," in *2022 IEEE 17th International Conference on Advanced Motion Control (AMC)*. IEEE, 2022, pp. 376–382.

[21] T. Schneider, G. Ducard, K. Rudin, and P. Strupler, "Fault-tolerant control allocation for multirotor helicopters using parametric programming," *rN*, vol. 1, p. r2, 2012.

[22] M. Diehl and S. Gros, "Numerical optimal control," *Optimization in Engineering Center (OPTEC)*, 2011.

[23] Z. Wang, X. Zhou, C. Xu, and F. Gao, "Geometrically constrained trajectory optimization for multicopters," *IEEE Transactions on Robotics*, vol. 38, no. 5, pp. 3259–3278, 2022.

[24] M. Faessler, A. Franchi, and D. Scaramuzza, "Differential flatness of quadrotor dynamics subject to rotor drag for accurate tracking of high-speed trajectories," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 620–626, 2017.

[25] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *2011 IEEE international conference on robotics and automation*. IEEE, 2011, pp. 2520–2525.

[26] J. Ferrin, R. Leishman, R. Beard, and T. McLain, "Differential flatness based control of a rotorcraft for aggressive maneuvers," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. Ieee, 2011, pp. 2688–2693.

[27] S. Wright, J. Nocedal *et al.*, "Numerical optimization," *Springer Science*, vol. 35, no. 67-68, p. 7, 1999.

[28] J. Warren, S. Schaefer, A. N. Hirani, and M. Desbrun, "Barycentric coordinates for convex sets," *Advances in computational mathematics*, vol. 27, pp. 319–338, 2007.

[29] D. C. Liu and J. Nocedal, "On the limited memory bfgs method for large scale optimization," *Mathematical programming*, vol. 45, no. 1-3, pp. 503–528, 1989.

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

[31] P. Foehn, E. Kaufmann, A. Romero, R. Penicka, S. Sun, L. Bauersfeld, T. Laengle, G. Cioffi, Y. Song, A. Loquercio *et al.*, "Agilicious: Open-source and open-hardware agile quadrotor for vision-based flight," *Science Robotics*, vol. 7, no. 67, p. eabl6259, 2022.