

Polynomial-based Online Planning for Autonomous Drone Racing in Dynamic Environments

Qianhao Wang [†], Dong Wang [†], Chao Xu, Alan Gao, and Fei Gao

Abstract—In recent years, there is a noteworthy advancement in autonomous drone racing. However, the primary focus is on attaining execution times, while scant attention is given to the challenges of dynamic environments. The high-speed nature of racing scenarios, coupled with the potential for unforeseeable environmental alterations, present stringent requirements for online replanning and its timeliness. For racing in dynamic environments, we propose an online replanning framework with an efficient polynomial trajectory representation. We trade off between aggressive speed and flexible obstacle avoidance based on an optimization approach. Additionally, to ensure safety and precision when crossing intermediate racing waypoints, we formulate the demand as hard constraints during planning. For dynamic obstacles, parallel multi-topology trajectory planning is designed based on engineering considerations to prevent racing time loss due to local optimums. The framework is integrated into a quadrotor system and successfully demonstrated at the DJI Robomaster Intelligent UAV Championship, where it successfully complete the racing track and placed first, finishing in less than half the time of the second-place¹.

I. INTRODUCTION

Quadrotors are gaining popularity in various industrial and commercial scenarios due to their versatility and exceptional performance. In recent years, autonomous drone racing, a research field focusing on planning trajectories for quadrotors to follow an aggressive reference routine while precisely crossing some intermediate landmarks, receives considerable attentions [1]–[4] and sparkes an international competition craze, such as the AlphaPilot Challenge [5, 6] and the Autonomous Drone Race [7, 8] in IEEE IROS. The ultimate pursuit of minimizing the execution time in drone racing increasingly ignites the fire for quadrotors to be applied in several emergencies, such as post-disaster communications and urgent transportation of essential supplies.

In such scenarios, dynamic obstacles or moving landmarks that require investigation and traversal will inevitably arise due to environmental changes. How to plan a minimum-time trajectory through a series of waypoints in dynamic environments remains a challenging problem that can not be completely solved by previous works. In detail, this problem requires that the planning method satisfies the conditions simultaneously: (1) highest possible speed of completing the track; (2) agile avoidance of the dynamic obstacles;

[†] **Equal contribution.**

All authors are with the College of Control Science and Engineering, Zhejiang University, Hangzhou, 310027, China, and also with the Huzhou Institute of Zhejiang University, Huzhou, 313000, China.

Email: {qhwangaa, fgaoaa}@zju.edu.cn

Corresponding Author: Fei Gao.

¹<https://pro-robomasters-hz-n5i3.oss-cn-hangzhou.aliyuncs.com/sass/event-list.html>

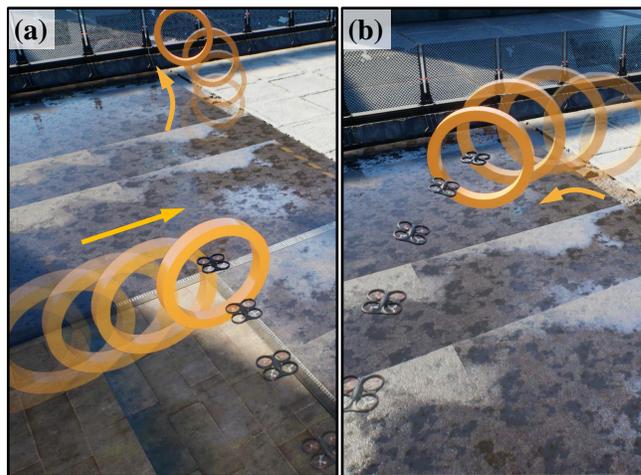


Fig. 1: The snapshot of our online planning method applied in a challenging dynamic environment, where the quadrotor is required to precisely pass through two dynamic gates in order. In (a), the quadrotor passes through the first gate and flies towards the second one. (b) depicts a close-up of crossing the second gate.

- (3) shortest possible computational time of trajectory replan;
- (4) precise traversal of the waypoints.

The first two points (1) and (2) are intuitive for safe racing in dynamic environments. As for (3), since not all variations in the dynamic surroundings can be preemptively observed and accurately predicted, these unforeseeable circumstances require online replanning. Particularly during high-speed drone racing, the efficiency of replanning is crucial for ensuring the timeliness of trajectories. Although existing works [1, 3] demonstrate the ability to compete with skilled human pilots in racing, minute-level computational demands make them prohibitive to respond to unforeseeable changes. For (4), as shown in Fig. 1, some waypoints may have low spatial tolerance even in motion. Inaccurate traversal can cause collision or mission failure. However, most racing works [2, 9] do not guarantee stable and precise traversal in planning, because of the use of soft constraint approach that relies on parameters.

In order to address the aforementioned requirements, we propose a strong polynomial-based online planning framework for racing in dynamic environments by incorporating careful engineering considerations with our previous works. We obtain the trajectory by an optimization method. For online replanning of (3), we adopt MINCO [10] as the trajectory representation and improve it to a time-uniform version. This implementation is more lightweight yet still maintains the capacity to allow for spatial-temporal deforma-

tions, improving computational efficiency. For rapid speed of (1), we boost the aggressiveness of quadrotors by minimizing execution time in the optimization based on the temporal freedom of trajectory. For precise waypoint traversal of (4), we formulate this waypoint-through requirement into a hard constraint to ensure safety and stability, regardless of whether the waypoint is static or in motion. For obstacle avoidance of (2), we build upon our previous work [11] by calculating multiple trajectories in parallel under different topologies segmented by dynamic obstacles and then selecting the optimal one.

Finally, we integrate the proposed planning framework into a customized quadrotor system, combining state estimation, control, and real-time vision-based detection modules. This system was deployed at the 2022 **DJI Robomaster Intelligent UAV Championship**², where quadrotors are tasked with navigating a track with dynamic obstacles, narrow gaps that require SE(3) planning, and gates. The gates must be traversed in a specific order as fast as possible, even though they are either in motion or have a random location within a range. In this competition, our system succeeded in completing the racing track and placed first, which proves our method’s comprehensive capabilities of performing high-speed flight in challenging dynamic environment.

To summarize, the contributions of this paper are:

- We implement time-uniform MINCO by improving our previous work, which boosts computational efficiency to enhance the timeliness of replan for high-speed flight.
- We achieve a hard constraint on the position of drone during crossing both static and moving waypoints, to ensure precision of crossing, by modeling the position of waypoints as trajectories about time and formulating them as boundary conditions for joint optimization.
- We propose a replanning framework, combined with evaluating different topologies segmented by dynamic obstacles and large attitude flight, to deal with challenging environments. Ablation experiments and competition prove our method’s effectiveness for racing in dynamic environments.

II. RELATED WORK

We divide racing works online and offline depending on whether the method can perform in real-time.

In general, offline methods are more comprehensive in problem construction and can serve as the baseline for online methods. Recently, there is a remarkable work [3] of Foehn et al. to generate time-optimal trajectories for drones to outperform professional pilots in racing. They use discrete state points to represent the trajectory and solve it using an optimization method while formulating the gate-through requirement as a complementary constraint. Additionally, they adopt a full-state quadrotor model with the thrust of single rotor as control input and impose constraint on each rotor, which saturates the actuator to achieve optimal time. Different from traditional racing, Han et al. [2] focus on

planning SE(3) trajectories to cross narrow gaps. They opt for polynomial trajectory and use the differential flatness of the quadrotor [12] to deliver spatial constraints to SE(3) state. Compared to solving a large-scale problem in Wang’s work [10], this method develops parallel computing for this planning problem, significantly enhancing efficiency. As for cluttered environments, Penicka et al. [13] extend on Li’s work [14] using a hierarchical sampling-based framework guided with an incrementally more complex quadrotor model. However, above methods do not have a real-time performance, making them inadequate to adapt to unpredictable changes, such as disturbances in the position of gate.

To enable the drone to handle environmental changes, online replanning is essential. Some search-based methods [15, 16] add the need to bring time to a minimum in the cost function to achieve real-time replan for rapid flight. But they can merely limit the acceleration of every axis while failing to explore the boundaries of the actuator, leading to conservative practices. For the 2019 AlphaPilot Challenge, Foehn [5] perform an onboard detection of the gates as a reference for replanning. They generate motion primitives based on maximum acceleration and obtain the time-optimal trajectory by sampling velocity states. Finally, they use a polynomial to fit the trajectory to track. Afterward, Romero [1] replace the polynomial parameterization of the above work [5] and use the Model Predictive Contouring Control (MPCC) [9] considering an accurate full-state model which is extended to include a linear drag model [17] to execute the trajectory obtained from sampling. Due to the penalty on progress item and constraining the dynamics of each individual rotor in MPCC, this approach can better exploit the drone’s performance limits. Nevertheless, both methods of leaving it to the controller to track the trajectory that ignores the dynamic feasibility constraints can only ensure the safety of the gate-through by adding a certain cost weight at the controller side. This is rarely applicable to dynamic environments.

As for the navigation works mentioned for collision-free flight, they concentrate on handling the information about the static or dynamic environment [11, 18] obtained from perception to serve subsequent trajectory generation and on constructing constraints to avoid obstacles [19, 20]. However, they have a shortage of adaptability for drone racing tasks, which require the quadrotor to traverse waypoints in order with a focus on execution time.

III. TRAJECTORY REPRESENTATION

In this section, for enhancing the computational efficiency for online replanning, we implement time-uniform MINCO based on our previous work [10] to conduct spatial-temporal deformation of the flat-output trajectory. We present the definitions and comparison in Tab. I. Essentially time-uniform MINCO and the normalized one are just two special kinds of MINCO. Therefore, to distinguish the special parts of the different trajectory representations, we use the symbols with hat $\hat{\cdot}$ and the symbols with horizontal lines $\bar{\cdot}$, such as Eq.(2)

²<https://www.robomaster.com/zh-CN/robo/drone>

TABLE I: Comparison of Different Trajectory Representations

	MINCO		Time-uniform MINCO		Normalized Time-uniform MINCO	
time allocation	$\mathbf{T} = (T_1, \dots, T_M)^T$,	(1)	$\hat{\mathbf{T}} = (T/M)\mathbf{1}$,	(2)	$\bar{\mathbf{T}} = \mathbf{1}$,	(3)
boundary conditions	$\mathbf{z}^o, \mathbf{z}^f$,	(4)	$\mathbf{z}^o, \mathbf{z}^f$,	(5)	$\bar{\mathbf{z}}^o = \mathbf{S}_s(T/M)\mathbf{z}^o$, $\bar{\mathbf{z}}^f = \mathbf{S}_s(T/M)\mathbf{z}^f$,	(6)
mapping equation	$\mathbf{M}(\mathbf{T})\mathbf{C} = \mathbf{b}(\mathbf{Q}, \mathbf{z}^o, \mathbf{z}^f)$,	(7)	$\mathbf{M}(\hat{\mathbf{T}})\hat{\mathbf{C}} = \mathbf{b}(\mathbf{Q}, \mathbf{z}^o, \mathbf{z}^f)$,	(8)	$\mathbf{M}(\mathbf{1})\bar{\mathbf{C}} = \mathbf{b}(\mathbf{Q}, \bar{\mathbf{z}}^o, \bar{\mathbf{z}}^f)$,	(9)
coefficients solving	by online PLU factorization and solving linear systems of equations		$\hat{\mathbf{c}}_i = \mathbf{S}_{2s}(M/T)\bar{\mathbf{c}}_i$,	(10)	$\bar{\mathbf{C}} = \mathbf{M}^{-1}(\mathbf{1})\mathbf{b}(\mathbf{Q}, \bar{\mathbf{z}}^o, \bar{\mathbf{z}}^f)$.	(11)

and Eq.(3), to denote the special parts in the time-uniform MINCO and normalized one, respectively. In the following, we go over the meaning of the variables that appear in Tab. I.

An s -order MINCO is defined as an m -dimensional M -piece polynomial trajectory. The i -th piece trajectory is defined by an $\mathcal{D} = 2s - 1$ degree polynomial as

$$p_i(t) = \mathbf{c}_i^T \beta(t), \quad \forall t \in [0, T_i], \quad (12)$$

where $\beta(x) := (1, x, \dots, x^{\mathcal{D}})^T$ is the natural basis. \mathbf{c}_i and T_i means the coefficients and duration of the i -th piece respectively. For the M -piece trajectory, the total duration is $T = \sum_{i=1}^M T_i$, and its coefficients \mathbf{C} , time allocation \mathbf{T} and intermediate points \mathbf{Q} can be written as

$$\begin{aligned} \mathbf{C} &= (\mathbf{c}_1^T, \dots, \mathbf{c}_1^T, \dots, \mathbf{c}_M^T)^T \in \mathbb{R}^{2Ms \times m}, \\ \mathbf{T} &= (T_1, \dots, T_i, \dots, T_M)^T \in \mathbb{R}_{>0}^M, \\ \mathbf{Q} &= (\mathbf{q}_1, \dots, \mathbf{q}_i, \dots, \mathbf{q}_{M-1}) \in \mathbb{R}^{m \times (M-1)}, \end{aligned} \quad (13)$$

where \mathbf{q}_i means the 0-order derivative of $p_i(t)$ at T_i . And in Tab. I, $\mathbf{z}^o, \mathbf{z}^f \in \mathbb{R}^{m \times s}$ is the boundary conditions containing high order derivative at $p_1(0)$ and $p_M(T_M)$. $\mathbf{M} \in \mathbb{R}^{2Ms \times 2Ms}$ and $\mathbf{b} \in \mathbb{R}^{2Ms \times m}$ are matrixes with \mathbf{T} and \mathbf{Q} , $\mathbf{z}^o, \mathbf{z}^f$ as variables, respectively, which refer to the Eq.(54, 55) in [10] for the detailed definition. In Eq.(2, 3), $\mathbf{1} = (1, 1, \dots, 1)^T \in \mathbb{R}^M$. In Eq.(6, 10), $\mathbf{S}_y(x) := \text{Diag}(1, x, \dots, x^{y-1})$.

A. MINCO Trajectory

For an s -order MINCO, given boundary conditions Eq.(4), intermediate points \mathbf{Q} and time allocation Eq.(1), the coefficients \mathbf{C} can be uniquely determined by $\mathbf{z}^o, \mathbf{z}^f, \mathbf{Q}$ and \mathbf{T} based on the linear mapping equation Eq.(7), which is defined as Theorem 2 in [10]. Because \mathbf{M} is a nonsingular banded matrix, based on its banded PLU factorization, the coefficients can be computed by solving two linear systems of equations with linear time and space complexity [21], which prevents the need to explicitly calculate \mathbf{M}^{-1} .

For user-defined penalty function $F(\mathbf{C}, \mathbf{T})$ with available gradients, MINCO serves as a linear-complexity differentiable layer $H(\mathbf{z}^o, \mathbf{z}^f, \mathbf{Q}, \mathbf{T}) = F(\mathbf{C}, \mathbf{T})$. To accomplish the deformation of MINCO, we need to obtain the gradients of H w.r.t. the trajectory's variable $\mathbf{z}^o, \mathbf{z}^f, \mathbf{Q}$ and \mathbf{T} from the given gradients $\partial F / \partial \mathbf{C}$ and $\partial F / \partial \mathbf{T}$ as Eq.(60, 68) in [10]. During the process, results of right multiplying \mathbf{M}^{-1} by a vector are needed several times. Similar to coefficients solving, these results can be get by using the PLU factorization of \mathbf{M} .

B. Time-uniform MINCO Trajectory

It should be emphasized that, in Tab. I and this section, the role of normalized time-uniform MINCO is to serve as an intermediate state of the time-uniform MINCO to aid in the calculation of coefficients and gradients.

We refer to a special MINCO trajectory with uniform time allocation Eq.(2) as time-uniform MINCO, since each piece has the same duration. The time-uniform MINCO still allows for spatial-temporal deformation, which is accomplished through the freedom of the total time T .

As Eq.(8) states, even with a uniform time allocation, we still need to online deal with \mathbf{M}^{-1} whenever T changes. To avoid PLU factorization of \mathbf{M} every time, using temporal scaling, we define normalized time-uniform MINCO $\bar{p}(t)$ of a time-uniform MINCO $\hat{p}(t)$, and the i -th piece is

$$\bar{p}_i(t) = \hat{p}_i(T/M \cdot t), \quad \forall t \in [0, 1], \quad (14)$$

where $\bar{p}_i(t)$ takes 1 as piece duration. Since the original trajectory is time-uniform, As defined in Eq.(14), the intermediate waypoints \mathbf{Q} do not change because the spatial shape of the trajectory is kept constant. The high order derivatives of $\bar{p}_i(t)$ can be written as

$$\bar{p}_i^{(s)}(t) = (T/M)^s \hat{p}_i^{(s)}(T/M \cdot t), \quad \forall t \in [0, 1]. \quad (15)$$

Then the boundary condition is deflated by a factor of $(T/M)^s$ in the s -th order derivative due to the temporal scaling, as written in Eq.(6).

For the normalized $\bar{p}(t)$, the mapping is written by Eq.(9), where the banded matrix $\mathbf{M}(\mathbf{1})$ becomes constant because the quantity of the pieces M is fixed during trajectory optimization. Therefore, $\mathbf{M}(\mathbf{1})^{-1}$ can be computed explicitly or performed PLU factorization offline, instead of online factorization every time T changes in the optimization. Then the coefficients $\bar{\mathbf{C}}$ can be obtained from Eq.(11) and the coefficients $\hat{\mathbf{C}}$ of the original trajectory can be obtained by scaling the normalized trajectory, as written in Eq.(10). Moreover, the gradient calculation mentioned in Sec. III-A can be performed without online factorization either. These have an improvement in computational efficiency, as shown in Sec. V-A.1.

To summarize, to use time-uniform MINCO yet avoid online PLU factorization, we first normalize the trajectory in time. Then we use the property that $\mathbf{M}(\mathbf{1})$ of the normalized trajectory can be processed offline to quickly obtain the normalized parameters $\bar{\mathbf{C}}$ and gradients. Finally, we get the

parameters $\hat{\mathbf{C}}$ and gradients of the time-uniform MINCO which is truly desired by time deflating as shown in Eq.(10).

IV. POLYNOMIAL-BASED ONLINE PLANNING

In this section, we use time-uniform MINCO as the trajectory representation for online planning, requiring the shortest possible execution time while satisfying some environmental and actuator constraints. First in Sec. IV-A we construct an optimization problem considering above requirements. Then based on our dealing with its inequality and equation constraints respectively in Sec. IV-B and IV-C, the problem is reformulated into an unconstrained optimization problem in Sec. IV-D. Additionally, in Sec. IV-E, we state some engineering considerations that are effective in improving racing performance. Note that the symbol definitions of time-uniform MINCO in this section are inherited from Sec. III.

A. Problem Formulation

In this paper, we use segmented polynomials to represent the replan trajectory. Given the next N gates, we plan an N -segment trajectory. As shown in Fig. 2, each colored curve represents one segment. The n -th segment $\sigma_n(t)$ is an s -order m -dimensional M_n -piece time-uniform MINCO, whose coefficients and intermediate points are defined as $\hat{\mathbf{C}}_n$ and \mathbf{Q}_n respectively, detailed in Eq.(13). Its time allocation is defined as $\hat{\mathbf{T}}_n = (T_n/M_n)\mathbf{1}$, where T_n is the trajectory duration of n -th segment. The whole segmented trajectory $\sigma(t) : [t_0, t_N] \mapsto \mathbb{R}^m$ is formulated as

$$\begin{aligned} \sigma(t_{n-1} + t) &= \sigma_n(t), \\ \forall n \in \{1, 2, \dots, N\}, \forall t \in [0, T_n], \end{aligned} \quad (16)$$

where $t_n = t_0 + \sum_{j=1}^n T_j$ is the timestamp and t_0 is the start time of the trajectory. The coefficients, time allocation, and intermediate points of the whole trajectory can be written as

$$\begin{aligned} \mathbf{C} &= (\hat{\mathbf{C}}_1^T, \dots, \hat{\mathbf{C}}_n^T, \dots, \hat{\mathbf{C}}_N^T)^T \in \mathbb{R}^{(\sum_{n=1}^N 2M_n s) \times m}, \\ \mathcal{T} &= (\hat{\mathbf{T}}_1^T, \dots, \hat{\mathbf{T}}_n^T, \dots, \hat{\mathbf{T}}_N^T)^T \in \mathbb{R}_{>0}^{(\sum_{n=1}^N M_n)}, \\ \mathcal{Q} &= (\mathbf{Q}_1, \dots, \mathbf{Q}_n, \dots, \mathbf{Q}_N) \in \mathbb{R}^{m \times (\sum_{n=1}^N (M_n - 1))}. \end{aligned} \quad (17)$$

We require the trajectory to pass through a series of gates, both static and moving, as quickly as possible, with constraints of dynamical feasibility, narrow gap crossing, and dynamic obstacle avoidance. Taking all requirements into account, our problem takes the following form:

$$\min_{\mathcal{C}, \mathcal{T}} J_o = \int_{t_0}^{t_N} \|\sigma^{(s)}(t)\|^2 dt + \rho \cdot \|\mathcal{T}\|_1, \quad (18a)$$

$$s.t. \quad \sigma_1^{[0, s-1]}(0) = \mathbf{s}^o, \quad (18b)$$

$$\sigma_N^{[0, s-1]}(T_N) = \mathbf{s}^f, \quad (18c)$$

$$\sigma_n^{[0, s-1]}(T_n) = \sigma_{n+1}^{[0, s-1]}(0), \quad \forall n \in \{1, \dots, N-1\}, \quad (18d)$$

$$\sigma(t_n) = g_n(t_n), \quad \forall n \in \{1, \dots, N\}, \quad (18e)$$

$$\mathcal{G}_x(\sigma(t), \dots, \sigma^{(s)}(t), t) \preceq \mathbf{0}, \quad \forall x \in \mathcal{X}, \forall t \in [t_0, t_N], \quad (18f)$$

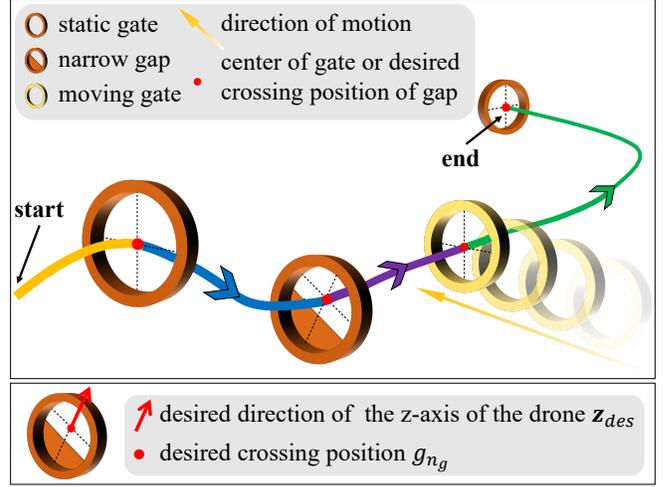


Fig. 2: Illustration of the segmented trajectory for replan to traversal the four gates in order.

where we define two costs in Eq.(18a) for smoothness and short execution time, which are weighed by parameter ρ . Eq.(18b–18c) and Eq.(18d) are the boundary conditions and the continuity constraint up to degree $s-1$. \mathbf{s}^o and \mathbf{s}^f are boundary states. We denote $\sigma^{[x,y]} \in \mathbb{R}^{m \times (y-x+1)}$ as

$$\sigma^{[x,y]} = (\sigma^{(x)}, \sigma^{(x+1)}, \dots, \sigma^{(y)}), \quad x < y. \quad (19)$$

Moreover, Eq.(18e) is the gate-through constraint, where $g_i(t)$ is the predicted trajectory for the i -th gate. Eq.(18f) is continuous-time constraints, the set $\mathcal{X} = \{t, b, g, d\}$ include actuator physical limits on thrust (t) and body rate (b), narrow gap crossing (g) and dynamic obstacle avoidance (d).

B. Inequality Constraints Transcription

For the inequality constraints Eq.(18f), which are required to be fulfilled along the whole trajectory. We use the thought behind penalty function method [22] to deal with these constraints becoming time integral of constraint violations, which is then evaluated by a finite sum of sample points. We define these points sampled on n -th segment by $\hat{\mathbf{p}}_{n,j} = \sigma_n((j/\kappa_n)T_n)$, $j \in \{0, 1, 2, \dots, \kappa_n\}$, where κ_n is the sample quantity and we name $\hat{\mathbf{p}}_{n,j}$ as **constraint points**. We denote the penalty function of the constraint points as

$$\mathcal{G}_x(n, j) = \mathcal{G}_x\left(\hat{\mathbf{p}}_{n,j}, \hat{\mathbf{p}}_{n,j}^{(1)}, \dots, \hat{\mathbf{p}}_{n,j}^{(s)}, t_{n-1} + \frac{j}{\kappa_n}T_n\right). \quad (20)$$

Then these inequality constraints Eq.(18f) can be transformed into a weighted sum of the sampled penalty as

$$\begin{aligned} \min_{\mathcal{C}, \mathcal{T}} \sum_x \lambda_x J_x, \\ J_x = \sum_{n=1}^N \frac{T_n}{\kappa_n} \sum_{j=0}^{\kappa_n} \bar{\omega}_j \max(\mathcal{G}_x(n, j), 0)^3, \end{aligned} \quad (21)$$

where λ_x is the weight for each cost J_x , we follow the trapezoidal rule [23] $(\bar{\omega}_0, \bar{\omega}_1, \dots, \bar{\omega}_{\kappa_i}) = (1/2, 1, \dots, 1, 1/2)$. Then for the inequality constraints Eq.(18f), with Eq.(12, 21), once the gradients of $\mathcal{G}_x(n, j)$ w.r.t. $\hat{\mathbf{p}}_{n,j}^{(k)}$, $k \in \{1, \dots, s\}$ and $\{T_1, \dots, T_n\}$ are given, we can derive the gradients $\partial J_x / \partial \mathbf{C}$ and $\partial J_x / \partial \mathcal{T}$ which are required in the optimization. Then we introduce each penalty function and its gradient.

1) *Actuator Limits with Drag Effects \mathcal{G}_t and \mathcal{G}_b* : To ensure that the trajectory is physically feasible, we constrain the thrust f and body rate ω through the differential flatness. Meanwhile, due to the high speed of racing, aerodynamic effects cannot be ignored, then we extended the quadrotor's dynamics with a drag model of our previous work [24]. As shown in the Eq.(17-21) of [24], given the trajectory, thrust, body rate and rotation $R \in SO(3)$ in the world frame of the constraint point $\hat{\mathbf{p}}_{n,j}$ can be denoted as

$$f, \omega, R = \mathcal{F} \left(\hat{\mathbf{p}}_{n,j}^{(1)}, \hat{\mathbf{p}}_{n,j}^{(2)}, \hat{\mathbf{p}}_{n,j}^{(3)} \right), \quad (22)$$

We define the penalty of actuator physical limits as

$$\begin{aligned} \mathcal{G}_t(n, j) &= (f - f_m)^2 - f_r^2, \\ \mathcal{G}_b(n, j) &= \|\omega\|^2 - \omega_{max}^2, \end{aligned} \quad (23)$$

where $f_m = (f_{max} + f_{min})/2$ and $f_r = (f_{max} - f_{min})/2$, f_{max} and f_{min} are the maximum and minimum values of thrust. ω_{max} is the maximum body rate. With Eq.(22, 23), the gradients $\partial \mathcal{G}_t(n, j) / \partial \hat{\mathbf{p}}_{n,j}^{(k)}$, $\partial \mathcal{G}_b(n, j) / \partial \hat{\mathbf{p}}_{n,j}^{(k)}$, $k = 1, 2, 3$ can be calculated easily using the chain rule.

2) *Narrow Gap Crossing \mathcal{G}_g* : In this framework, we assume that we can obtain the optimal attitude and position for crossing a narrow gap from other modules such as online detection. As shown in Fig. 2, we set the desired direction of the z-axis of the quadrotor when traversing the gate as a normalized vector $\mathbf{z}_{des} \in \mathbb{R}^m$. To avoid loss of generality, we set the optimal crossing position as a gate, denoted as the n_g -th gate to be passed through. The position constraint will be detailed in Sec. IV-C.2, and here we only describe the direction constraint during gate crossing. To make it safer to traverse the narrow gate with a large attitude, we impose this constraint on the trajectory at a certain sample range n_{ran} in front of and behind the n_g -th gate:

$$\mathcal{G}_g(n, j) = \|\mathbf{R}\mathbf{e}_3 - \mathbf{z}_{des}\|^2 - \theta_{tol}, \quad (24)$$

$$\forall j \in \begin{cases} \{0, 1, \dots, n_{ran}\}, & n = n_g \\ \{\kappa_n - n_{ran}, \dots, \kappa_n\}, & n = n_g - 1 \end{cases}$$

where $\mathbf{e}_3 = (0, 0, 1)^T$, $\theta_{tol} \in (0, 1)$ is the tolerance, \mathbf{R} is the rotation of $\hat{\mathbf{p}}_{n,j}$ obtained from Eq.(22). Then the gradient $\partial \mathcal{G}_g(n, j) / \partial \hat{\mathbf{p}}_{n,j}^{(k)}$ can be computed with Eq.(22, 24).

3) *Dynamic Obstacle Avoidance \mathcal{G}_d* : Based on our previous work [11], we model a dynamic obstacle as ellipsoid:

$$\mathbb{E} = \{\mathbf{x} | E(\mathbf{x}, \mathbf{y}) < 1\}, \quad E(\mathbf{x}, \mathbf{y}) = (\mathbf{x} - \mathbf{y})^T \mathbf{H}(\mathbf{x} - \mathbf{y}), \quad (25)$$

where $\mathbf{H} = R_E^T \text{diag}(1/a^2, 1/b^2, 1/c^2) R_E$, a, b, c is the axis-length, R_E is its rotation and \mathbf{y} is the center of the ellipsoid. Then we define the obstacle avoidance penalty as

$$\mathcal{G}_d(n, j) = d_{thr} - E \left(\hat{\mathbf{p}}_{n,j}, p_d(t_{n-1} + \frac{j}{\kappa_n} T_n) \right), \quad (26)$$

where $p_d(t)$ is the predicted trajectory of the dynamic obstacle, d_{thr} is security threshold. Similar to Sec. IV-B.1 and IV-B.2, by the chain rule, we can utilize Eq.(26) to obtain gradient of $\mathcal{G}_d(n, j)$ w.r.t. $\hat{\mathbf{p}}_{n,j}$ and $\{T_1, \dots, T_n\}$, which are used to derive $\partial J_d / \partial \mathcal{C}$ and $\partial J_d / \partial \mathcal{T}$ for optimization, as stated in Sec. IV-B.

C. Equality constraints Elimination

In Sec. IV-B we transform the inequality constraints by the penalty function method, in this section we eliminate the equation constraints by replacing the decision variables.

1) *Boundary Conditions and Continuity Constraint*: We define \mathbf{z}_0 as the start state of the first segment and \mathbf{z}_n as the end state of the n -th segment:

$$\begin{aligned} \mathbf{z}_0 &= \sigma_1^{[0, s-1]}(0), \\ \mathbf{z}_n &= \sigma_n^{[0, s-1]}(T_n), \quad \forall n \in \{1, \dots, N\}. \end{aligned} \quad (27)$$

For the n -th segment, given the time allocation $\hat{\mathbf{T}}_n$, intermediate points $\hat{\mathbf{Q}}_n$ and the boundary conditions $\mathbf{z}_{n-1}, \mathbf{z}_n$, its coefficients $\hat{\mathbf{C}}_n$ can be uniquely determined by a serie of calculations Eq.(8–11). We denote this calculation process as

$$\mathbf{C}_n = \mathcal{M}(\hat{\mathbf{Q}}_n, \hat{\mathbf{T}}_n, \mathbf{z}_n, \mathbf{z}_{n-1}). \quad (28)$$

We set $\mathcal{Z} = (\mathbf{z}_1, \dots, \mathbf{z}_{N-1})$, and fix $\mathbf{z}_0 = \mathbf{s}^o$ and $\mathbf{z}_N = \mathbf{s}^f$. Based on Eq.(28), \mathcal{C} can be determined with variables \mathcal{Z}, \mathcal{T} and \mathcal{Q} . Therefore, for our original problem Eq.(18), we replace the decision variables from $(\mathcal{C}, \mathcal{T})$ to $(\mathcal{Z}, \mathcal{T}, \mathcal{Q})$, eliminating the constraints of Eq.(18b–18c).

2) *Gate-through Constraint*: The requirement of crossing gate is usually handled with soft constraints, for instance, penalty function $\|\sigma(t_n) - g_n(t_n)\|^2$ used in [1, 9]. However, As shown in Fig. 2, when the gate is small, the space tolerance at the moment of crossing is very low. The use of soft constraints does not guarantee that trajectory passes through the center of the gate, which leads to more pressure on safety being placed on other modules, such as the controller. In this paper, we construct hard constraints for gate-through to ensure that the trajectory can traverse the center of the gate.

We split the state matrix \mathbf{z}_n as $((\mathbf{z}_n)_0, (\mathbf{z}_n)_*)$, where $(\mathbf{z}_n)_0 \in \mathbb{R}^{m \times 1}$ is the 0-order derivative state, and $(\mathbf{z}_n)_* \in \mathbb{R}^{m \times (s-1)}$ means the derivatives whose order from 1 to s . Then the gate-through constraints Eq.(18e) can be written as

$$\begin{aligned} (\mathbf{z}_n)_0 &= g_n(t_n) \\ &= g_n(t_0 + \sum_{j=1}^n T_j), \end{aligned} \quad (29)$$

which demonstrates that $(\mathbf{z}_n)_0$ can be determined by \mathcal{T} . Therefore, the mapping function Eq.(28) can be formulated as

$$\begin{aligned} \mathbf{C}_n &= \mathcal{M} \left(\hat{\mathbf{Q}}_n, \hat{\mathbf{T}}_n, \left(g_n(t_0 + \sum_{j=1}^n T_j), (\mathbf{z}_n)_* \right), \right. \\ &\quad \left. \left(g_{n-1}(t_0 + \sum_{j=1}^{n-1} T_j), (\mathbf{z}_{n-1})_* \right) \right). \end{aligned} \quad (30)$$

To eliminate the constraints of Eq.(18e), We replace the variables in Sec. IV-C.1 from $(\mathcal{Z}, \mathcal{T}, \mathcal{Q})$ to $(\mathcal{Z}_*, \mathcal{T}, \mathcal{Q})$, where $\mathcal{Z}_* = ((\mathbf{z}_1)_*, \dots, (\mathbf{z}_{N-1})_*)$.

D. Problem Reformulation

Combining the transcription of inequality constraints in Sec. IV-B and the elimination of equality constraints in

Sec. IV-C, the whole original problem Eq.(18) is finally reformulated as an unconstrained optimization problem:

$$\min_{\mathcal{Z}, \mathcal{T}, \mathcal{Q}} J = (J_o + \sum_x \lambda_x J_x). \quad (31)$$

Sec. IV-B gives the gradients of J w.r.t. $(\mathcal{C}, \mathcal{T})$, but we change the decision variables in Eq.(31). Based on Eq.(28), we can obtain the gradients of \mathcal{C} w.r.t. $(\mathcal{Z}, \mathcal{T}, \mathcal{Q})$, which are detailed in the Eq.(60, 68) in [10]. However, in the new mapping function Eq.(30), we set \mathbf{z}_n a matrix with $(\mathbf{z}_n)_0$ and $(\mathbf{z}_n)_*$ as variables, and $(\mathbf{z}_n)_0$ is a vector with \mathcal{T} as decision variable. Therefore, to obtain the gradient of J w.r.t. the new decision variables $(\mathcal{Z}_*, \mathcal{T}, \mathcal{Q})$, which are required when solving the final problem Eq.(31), we derive the gradient

$$\frac{\partial (\mathbf{z}_n)_0}{\partial T_k} = \frac{\partial g_n(t_n)}{\partial t_n} \frac{\partial (t_0 + \sum_{j=1}^n T_j)}{\partial T_k} = \begin{cases} 0, & n < k \\ \dot{g}_n(t_n), & n \geq k \end{cases}. \quad (32)$$

E. Implementation Details

1) *Parallel Optimization for Different Topologies:* Although dynamic obstacles are addressed in many works [11, 18], they only care about obstacle avoidance, ignoring the spatial topology segmentation brought by dynamic obstacles. The choice of different topologies has a significant impact on the execution time of racing, as demonstrated in Sec. V-A.3. Therefore, in this paper, as shown in Fig. 6, we generate several trajectories with different topologies split by the dynamic obstacle. Then we use them in parallel as initial values for trajectory optimization and finally choose the one with the shortest flight time for the drone to execute.

2) *Numerical Optimization:* We adopt L-BFGS³ [25] to solve the unconstrained optimization problem Eq.(31).

3) *Global Planning:* For some large-scale scenarios, such as Sec. V-B, we first generate a global reference trajectory using a one-segment MINCO trajectory, and then use part of the global trajectory as the initial value for the optimization of the local online replanning during flight.

V. EVALUATIONS

In this section, to verify the effectiveness of our contributions summarized in Sec. I, we design three ablation experiments. Then to validate the practicality of our planning method, we integrate the method into a complete quadrotor system, which is applied to 2022 DJI Robomaster Intelligent UAV Championship². All the simulation experiments are run on a desktop equipped with an Intel Core i7-10700 CPU.

A. Ablation Experiments

1) *Evaluation for Efficiency Improvement of Time-uniform MINCO:* We compare the computational efficiency of MINCO and time-uniform MINCO with the same segment and piece number. For both of them, we test the number of segments N varied from 2 to 5, while the number of pieces $M = \{2, 4, \dots, 10\}$. Both methods are given the same initial trajectory for each comparison. The results are

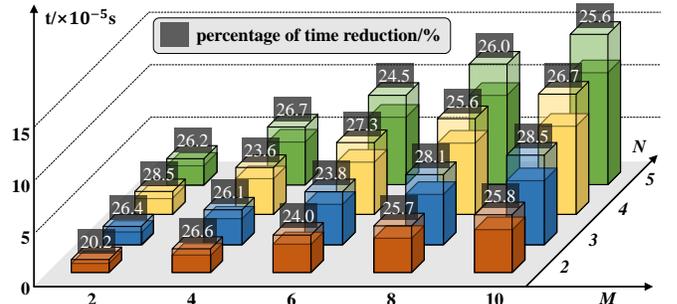


Fig. 3: Illustration of the time of a single iteration in the trajectory optimization. The transparent area represents the more computation time that MINCO takes than the time-uniform MINCO.

illustrated in Fig. 3, where the time of a single iteration of the trajectory optimization is visualized. We represent the time spent by time-uniform MINCO with different colored bars. The transparent part indicates the additional time required for MINCO compared to time-uniform MINCO. Additionally, we show the percentage of time reduction of time-uniform MINCO over MINCO on each bar.

As shown in Fig. 3, using time-uniform MINCO effectively improves the calculation efficiency, even if the quantity of segments or pieces varies.

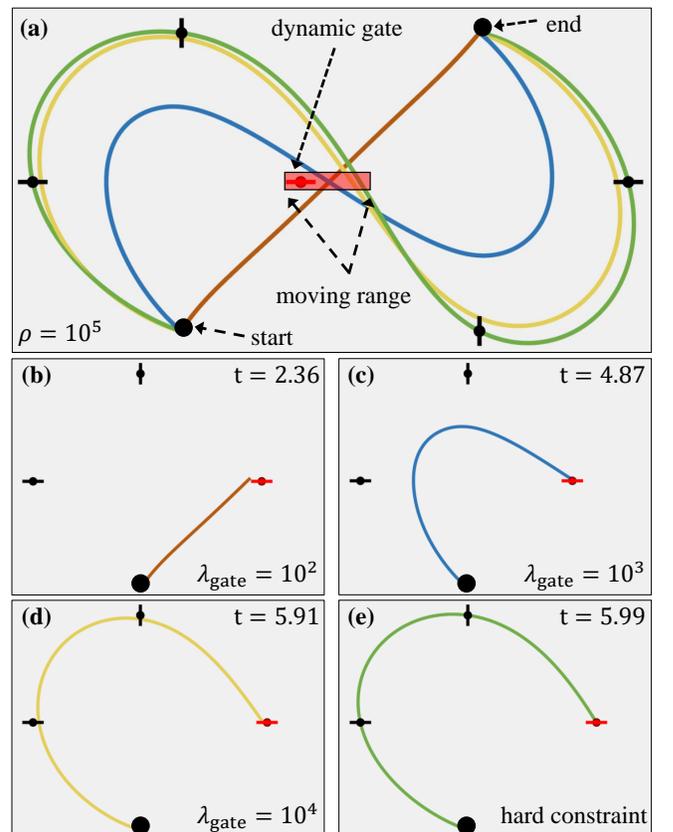


Fig. 4: Illustration of the top view of the results in Sec. V-A.2. The short thick black lines represent the static gates and the short thick red line represents the dynamic gate. The circle on the line represents the center of the gate. The red bar on the red line indicates the movement range of the dynamic gate. (a) demonstrates the trajectories of the quadrotor completing the track under different soft constraint weights and the same time weight $\rho = 10^5$. (b)-(e) represent the moments for the quadrotor to pass through the dynamic gate with different parameters.

³<https://github.com/ZJU-FAST-Lab/LBFGS-Lite>

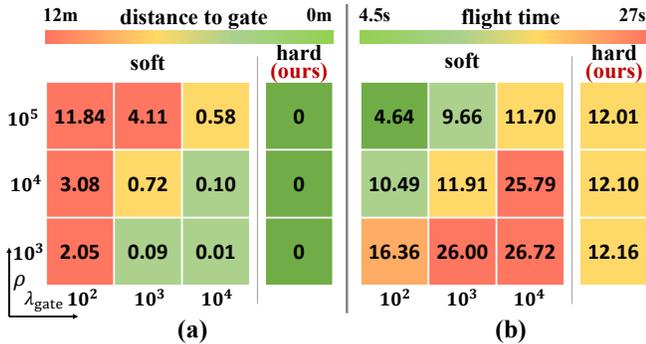


Fig. 5: Results of experiments in Sec. V-A.2, where time weight ρ and soft constraint weight λ_{gate} vary from 10^3 to 10^5 and 10^2 to 10^4 , respectively. (a) shows the average distance from the trajectory to the corresponding n -th gate at the moment t_n , from far to near, the color changes from red to green. (b) displays the flight time with different parameters, from less to more, and the color varies from green to red.

2) Evaluation for constraints for Precise Gate Crossing:

We compare our method proposed in Sec. IV-C.2 with a commonly used soft constraint method which uses penalty function $\lambda_{gate} \|\sigma(t_n) - g_n(t_n)\|^2$, where λ_{gate} is the weight. The experiment scenario is set up as shown in Fig. 4.(a), where the quadrotor is required to pass through 5 gates in order. The red dynamic gate moves at 2 m/s horizontally back and forth from side to side within a certain range, which is indicated by the red bar in Fig. 4.(a). To compare fairly, each trajectory optimization is given the same initial value and we set other parameters that are not about this experiment the same. We set different time weight ρ and soft constraint weights λ_{gate} for the experiments.

The results are illustrated in Fig. 5, soft constraint method struggles to find suitable parameters to trade off the shortest possible execution time with precise gate crossing. To elaborate, when choosing the parameters that achieve a small distance to gate as shown in the green area of Fig. 5.(a), the soft constraint method causes poor results in terms of the time of flight as shown in the red area of Fig. 5.(b). Conversely, when choosing the parameters that aim for the shortest possible flight time in the green or yellow area of Fig. 5.(b), the soft constraint method is difficult to precisely crossing the gate, which corresponds to the red area of Fig. 5.(a). In contrast, our hard constraint approach only needs to focus on adjusting the time weight ρ to shorten the execution time, while maintaining precise traversal.

To more visually demonstrate the importance of accurate traversal, in Fig. 4, we visualize the trajectories of our method and the soft constraint method when the time weight $\rho = 10^5$. The results show that, compared to the soft constraint method, our method guarantees accurate traversal for both dynamic and static gates.

3) *Evaluation for Choosing Different Topologies Segmented by Dynamic Obstacles:* We use a typical scenario to verify the usefulness of this engineering consideration. As shown in Fig. 6.(a), the quadrotor is required to take off from the start point, pass through two gates in order, and then reach the end point. Between the two gates, there is a dynamic obstacle which we model as an ellipsoid. The movement of its central position with respect to time we

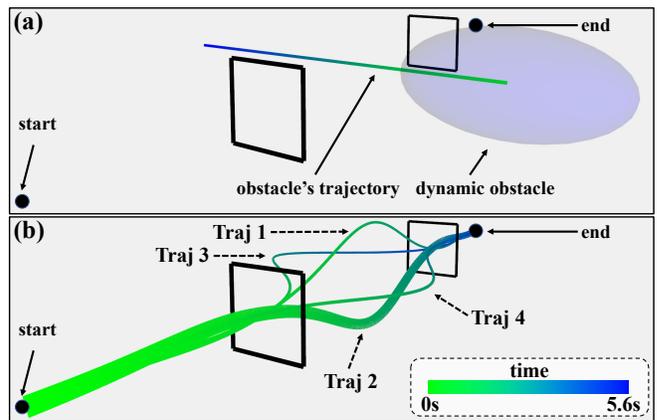


Fig. 6: Illustration of the results in Sec. V-A.3. (a) demonstrates the experiment setup. (b) shows the trajectories optimized based on initial values from different topological.

TABLE II: Trajectories of Different Topologies

Traj number	1	2	3	4
execution time (s)	3.82	3.79	5.60	4.38
trajectory length (m)	20.41	20.05	22.97	20.64

represent by a colored trajectory in Fig. 6.(a). We set the trajectories through different topologies past the obstacle as initial values. The final trajectories after optimization are illustrated in Fig. 6.(b), the bolded trajectory is significantly faster than the trajectories of other topologies. Meanwhile, we present the execution time of each trajectory in Tab. II, where the choice of different topologies has a considerable impact on the execution time, which confirms the validity of our evaluating and choosing different topologies segmented by the dynamic obstacle.

B. 2022 DJI Robomaster Intelligent UAV Championship

We take part in the second event Autonomous Racing of this competition, where the quadrotor is required to fly in a dynamic and challenging environment as fast as possible. The racing track is about 160 m long, which contains a series of static gates (as shown in Fig. 7.(a)), moving obstacles, dynamic gates with different movement patterns (as illustrated in Fig. 1), and a shaped gate which demands to plan SE(3) trajectory (as shown in Fig. 7.(b)). The radius of the gates is 1 m. The positions of the gates are provided ahead of each race up to approximately 2 m uncertainty. This requires the drone to be able to detect gates and make trajectory adjustments in real time, such as online replanning based on the detection results.

TABLE III: Parameters of Planning

N	M	m	λ_t	λ_b	λ_g	λ_d
2	4	3	100	100	10000	10000

In the competition, we integrate the proposed framework into a customized quadrotor system, combining localization, detection, and control modules. For the planning module, the parameters defined in Sec. IV of trajectory optimization are



Fig. 7: Illustration of the flight of our system in the 2022 DJI Robomaster Intelligent UAV Championship.

shown in Tab. III. The average time overhead of replan is 16.6 ms when considering SE(3) for narrow gaps and 7.0 ms when not considering. We opt for VINS-MONO [26] as the localization module. Then we use the point cloud from the depth image for the detection of gates and dynamic obstacles. Finally, we adopt MPC [17] as the control method to track the planned aggressive trajectory.

We visualize the flight of our system in the competition in Fig. 1 and 7. Readers can get a better understanding of the experiment from the attached video. Additionally, we show the final results and rankings in Tab. IV, where our system is significantly faster than the other teams, demonstrating the great performance of our method for racing in dynamic environments.

TABLE IV: Ranking of Competition Results¹

rankings	our team	2nd place	3rd place	...
completion time (s)	22.0	50.3	76.9	...

VI. CONCLUSION

In this paper, we propose a polynomial-based trajectory planning method to address the 4 requirements for racing in dynamic environments presented in Sec. I. Efforts in trajectory representation, hard constraint designed for crossing waypoints, and parallel evaluation of trajectory under different topologies, effectively improve replan efficiency, the accuracy of waypoint traversal, and flight time when facing dynamic obstacles. Finally, the method is applied to the DJI competition, and the outstanding result proves the good performance of our method.

REFERENCES

- [1] 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.
- [2] 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.
- [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. Hanover, A. Loquercio, L. Bauersfeld, A. Romero, R. Penicka, Y. Song, G. Cioffi, E. Kaufmann, and D. Scaramuzza, "Autonomous drone racing: A survey," *arXiv e-prints*, pp. arXiv-2301, 2023.

- [5] 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.
- [6] W. Guerra, E. Tal, V. Murali, G. Ryou, and S. Karaman, "Flightgoggles: Photorealistic sensor simulation for perception-driven robotics using photogrammetry and virtual reality," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2019, pp. 6941–6948.
- [7] H. Moon, J. Martinez-Carranza, T. Cieslewski, M. Faessler, D. Falanga, A. Simovic, D. Scaramuzza, S. Li, M. Ozo, C. De Wagter, et al., "Challenges and implemented technologies used in autonomous drone racing," *Intelligent Service Robotics*, vol. 12, pp. 137–148, 2019.
- [8] J. A. Cocomo-Ortega and J. Martínez-Carranza, "Towards high-speed localisation for autonomous drone racing," in *Advances in Soft Computing: 18th Mexican International Conference on Artificial Intelligence, MICAI 2019, Xalapa, Mexico, October 27–November 2, 2019, Proceedings 18*. Springer, 2019, pp. 740–751.
- [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] 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.
- [11] Y. Wang, J. Ji, Q. Wang, C. Xu, and F. Gao, "Autonomous flights in dynamic environments with onboard vision," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2021, pp. 1966–1973.
- [12] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *2011 IEEE international conference on robotics and automation*, 2011, pp. 2520–2525.
- [13] 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.
- [14] Y. Li, Z. Littlefield, and K. E. Bekris, "Asymptotically optimal sampling-based kinodynamic planning," *The International Journal of Robotics Research*, vol. 35, no. 5, pp. 528–564, 2016.
- [15] S. Liu, N. Atanasov, K. Mohta, and V. Kumar, "Search-based motion planning for quadrotors using linear quadratic minimum time control," in *2017 IEEE/RSJ international conference on intelligent robots and systems*, 2017, pp. 2872–2879.
- [16] S. Liu, K. Mohta, N. Atanasov, and V. Kumar, "Search-based motion planning for aggressive flight in se (3)," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 2439–2446, 2018.
- [17] 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.
- [18] G. Chen, S. Wu, M. Shi, W. Dong, H. Zhu, and J. Alonso-Mora, "Rast: Risk-aware spatio-temporal safety corridors for mav navigation in dynamic uncertain environments," *IEEE Robotics and Automation Letters*, 2022.
- [19] X. Zhou, Z. Wang, H. Ye, C. Xu, and F. Gao, "Ego-planner: An esdf-free gradient-based local planner for quadrotors," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 478–485, 2020.
- [20] 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.
- [21] G. H. Golub and C. F. Van Loan, *Matrix computations*. JHU press, 2013.
- [22] L. S. Jennings and K. L. Teo, "A computational algorithm for functional inequality constrained optimization problems," *Automatica*, vol. 26, no. 2, pp. 371–375, 1990.
- [23] W. H. Press, H. William, S. A. Teukolsky, A. Saul, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes 3rd Edition: The Art of Scientific Computing*. Cambridge University Press, 2007.
- [24] Z. Wang, C. Xu, and F. Gao, "Robust trajectory planning for spatial-temporal multi-drone coordination in large scenes," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2022, pp. 12 182–12 188.
- [25] D. C. Liu and J. Nocedal, "On the limited memory bfgs method for large scale optimization," *Mathematical programming*, vol. 45, no. 1, pp. 503–528, 1989.
- [26] T. Qin, P. Li, and S. Shen, "Vins-mono: A robust and versatile monocular visual-inertial state estimator," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 1004–1020, 2018.