

Real-Time Spatial NMPC for Autonomous Drone Racing in Constrained Environments

Niwesh Sah* Jai Kumar Seth†

Technical University of Munich
Munich, Germany

Winter Semester 2025/26

Abstract

Autonomous drone racing pushes aerial vehicles to their performance limits by minimizing lap time while strictly satisfying safety and task constraints, such as collision-free flight and precise gate traversal. This work introduces a novel approach that combines spatial reformulation of the quadrotor dynamics with Nonlinear Model Predictive Control (NMPC) to efficiently handle non-convex obstacle avoidance constraints in real time. By reparameterizing the system dynamics with respect to progress along a reference racing line, the method transforms complex temporal requirements—such as accurate gate passing and obstacle avoidance—into simpler, geometry-based spatial constraints. This reformulation enables the NMPC framework to effectively manage challenging gate orientations and tight obstacle configurations at significantly lower computational cost than conventional time-indexed formulations, while preserving competitive racing performance and strong robustness to state and environmental uncertainty.

The source code is publicly available at: https://github.com/niweshsah/lisy_drone_racing

1 Introduction

Drone racing represents one of the most demanding applications for autonomous aerial systems. It combines extreme flight speeds, aggressive maneuvering, and navigation through highly constrained environments. Racing drones must minimize lap times while maintaining dynamic stability and strictly avoiding collisions with gates, track boundaries, and obstacles. These requirements impose stringent constraints on both trajectory generation and control strategies.

Autonomous drone racing has emerged as a benchmark problem in high-performance control due to its strongly nonlinear dynamics and strict real-time requirements. At racing velocities, even minor deviations from the planned trajectory or small computational delays in the control loop can lead to instability or catastrophic collisions. The primary objective of autonomous drone racing is to design a robust control system capable of guiding the vehicle through a sequence of gates and obstacles in the minimum possible time while ensuring collision-free flight under challenging aerodynamic and environmental conditions.

Consequently, autonomous racing systems need to produce collision-free, as well as dynamically feasible and time-optimal, trajectories. The classical approaches to planning include grid-based search, Rapidly-exploring Random Trees (RRT), and Probabilistic Roadmaps (PRM). While useful for creating collision-free paths, these methods are mainly concerned with geometric feasibility. Such methods do not explicitly consider time-optimality or complete vehicle dynamics and are therefore inappropriate in aggressive racing conditions where performance is a key factor.

To overcome these limitations, methods like Optimal Control Problems (OCPs) and Model Predictive Control (MPC) use system dynamics, actuator limits, and state constraints. Nevertheless, time-optimal OCPs formulated in the direct time domain have been found to be computationally expensive, in particular in the case of high-speed flight.

Spatial reformulation is an effective alternative because it presents system dynamics in terms of progress along a reference racing line instead of time. In this method, longitudinal motion is completely decoupled from transverse deviation, and racing gates and track boundaries are described as simple spatial constraints. Therefore, collision prevention and constraint management are greatly simplified.

This project targets the navigation and control level of autonomous drone racing, where a reference racing line is optimized into a path that is close to time-optimal, satis-

*niwesh.sah@tum.de

†go35yep@mytum.de

fies quadcopter dynamics, actuator constraints, and gate constraints, and is computationally efficient. The proposed framework adapts spatial reformulation and NMPC techniques to the drone racing field, where high robustness and fast computation are crucial.

2 Challenge Setup

The drone racing challenge considers a Crazyflie quadrotor [6] flying through a track with multiple gates and obstacles. At each discrete time step t , the controller receives the system state. The code runs on the lab pc and control input is passed onto the drone wirelessly. Before presenting both approaches, we briefly describe the common setup on which they are built. At each time step, the controller has access to the current system state

$$x = [p, \dot{p}, \xi, \dot{\xi}, G, O, i], \quad (1)$$

where $p = [x, y, z]$ denotes the drone position in the world frame and $\xi = [\phi, \theta, \psi]$ its orientation. The vectors G and O encode the poses of all racing gates and obstacles. These are initialized from nominal configurations and updated to their true physical values at runtime once the drone enters a local observation range. The discrete index selects the next gate that must be traversed.

Based on the state information, the controller generates specific control actions. Rather than directly commanding individual motor thrusts, the control policy operates at a higher level of abstraction. It outputs a desired reference state vector:

$$\mathbf{u} = [\phi, \theta, \psi, T] \quad (2)$$

where ϕ , θ , and ψ represent the desired roll, pitch, and yaw, respectively, and T represents the collective thrust. An onboard tracking controller is responsible for converting this high-level reference into low-level motor inputs.

3 Proposed Method

3.1 Global Path Generation

The first step of our framework involves generating an offline global path traversing the sequence of gates prior to takeoff. This global reference is computed without initial obstacle constraints, as dynamic obstacle avoidance is handled online by the NMPC controller.

Common approaches for global path generation in the literature include:

- **Sampling-based methods**, such as PRM or RRT*. These are frequently used for planning trajectories, particularly for linearized quadrotor models.
- **Search-based planning methods** [8, 1], which rely on discretized state and time spaces to find optimal paths.

- **Polynomial and spline-based methods** that exploit the *differential flatness* property of quadrotors [10, 4]. These approaches represent the trajectory as a continuous-time polynomial or spline. By leveraging differential flatness, full-state trajectory planning reduces to planning only four flat outputs (typically 3D position and yaw). The higher-order derivatives of these flat outputs naturally yield dynamically feasible trajectories and corresponding control inputs.

In this work, we adopt a cubic spline interpolation strategy due to its computational efficiency and smoothness properties. We define the knot points of the spline using the gate centers. To ensure the drone passes through the gates with the correct orientation, we augment each gate center with two auxiliary waypoints: one positioned slightly before and one slightly after the gate frame, aligned along the gate’s normal vector. These points are then connected using cubic splines to form the continuous reference path Γ . Our experiments indicate that this approach offers the optimal trade-off between computational speed and tracking robustness.

We decided to use the well known strategy of cubic splines: we use the gate centers as waypoints, augmented with one additional point slightly ahead and one slightly behind along the gate normal direction. These points are then connected using cubic splines. This approach provides the best trade-off between speed and robustness in our experiments.

3.2 Spatial Reformulation

In autonomous racing applications, precise and aggressive motion must be achieved while following a predefined three-dimensional reference path. Conventional formulations describe motion as a function of time, which couples spatial evolution and velocity in a way that complicates path-following control. When operating in highly curved and constrained environments, the primary objective is not simply to reach a position at a given time but to maintain accuracy relative to a racing line while progressing as fast as possible. Spatial reformulation addresses this by reparameterizing the system dynamics with respect to the geometry of the path rather than time, allowing the motion to be expressed relative to the path itself.

The reference path is represented as a smooth curve

$$\Gamma = \{\gamma(s) \in \mathbb{R}^3 : s \in [0, L]\}, \quad (3)$$

where s denotes the arc-length parameter and L is the total path length. The arc-length parameterization enforces the condition $\|\gamma'(s)\| = 1$, ensuring that the parameter s directly corresponds to the physical distance traveled along the path. As a result, the variable s naturally represents progress along the racing line and provides a meaningful spatial coordinate for reformulating the motion equations.

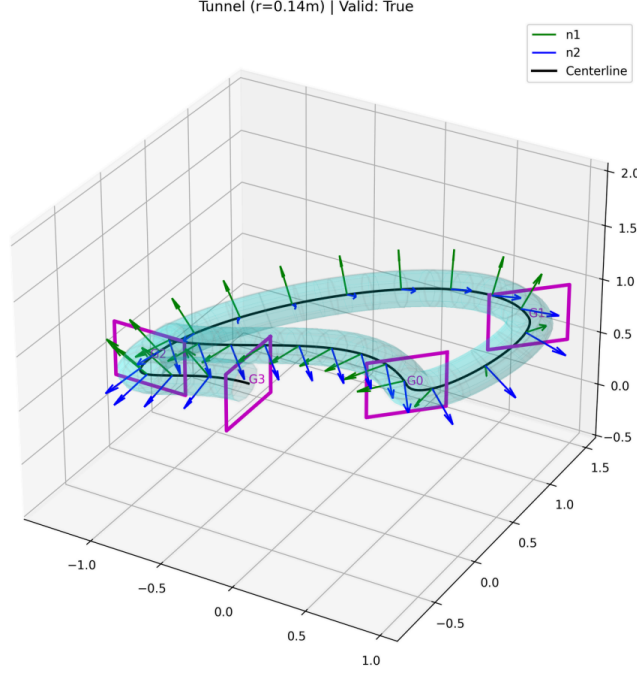


Figure 1: Example of parallel transport frame

3.3 Parallel Transport Frame

We use the Parallel Transport frame, which provides a smooth and singularity-free reference frame along the path, as seen in Figure 1. The Parallel Transport frame is defined by an orthonormal triad

$$\{\mathbf{t}(s), \mathbf{n}_1(s), \mathbf{n}_2(s)\}, \quad (4)$$

where $\mathbf{t}(s)$ is the tangent vector and $\mathbf{n}_1(s)$ and $\mathbf{n}_2(s)$ span the plane orthogonal to the path. [3] The evolution of this frame along the arc-length coordinate is governed by

$$\frac{d}{ds} \begin{bmatrix} \mathbf{t}(s) \\ \mathbf{n}_1(s) \\ \mathbf{n}_2(s) \end{bmatrix} = \begin{bmatrix} 0 & k_1(s) & k_2(s) \\ -k_1(s) & 0 & 0 \\ -k_2(s) & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{t}(s) \\ \mathbf{n}_1(s) \\ \mathbf{n}_2(s) \end{bmatrix}. \quad (5)$$

The scalar functions $k_1(s)$ and $k_2(s)$ describe the normal development of the path in the two transverse directions, and the curvature magnitude satisfies $\kappa(s)^2 = k_1(s)^2 + k_2(s)^2$.

At any given time, the vehicle position $\mathbf{p}(t)$ can be related to the reference path through orthogonal projection. The closest point on the path is defined as

$$s^*(t) = \arg \min_s \|\mathbf{p}(t) - \gamma(s)\|^2, \quad (6)$$

which yields a unique solution under standard smoothness assumptions. The displacement vector between the vehicle and the path is given by $\mathbf{w}(t) = \mathbf{p}(t) - \gamma(s^*)$. By construction, this displacement lies entirely in the plane orthogonal to the tangent vector and can therefore be expressed in the Parallel Transport frame as

$$\mathbf{w} = [0 \ w_1 \ w_2]^T, \quad (7)$$

where w_1 and w_2 represent the transverse deviations from the racing line.

3.4 Reformulation in transverse dynamics

Using the Parallel Transport frame, the vehicle position can be reconstructed as

$$\mathbf{p}(t) = \gamma(s^*) + R_{PT}(s^*)\mathbf{w}, \quad (8)$$

where $R_{PT}(s^*) = [\mathbf{t}(s^*) \ \mathbf{n}_1(s^*) \ \mathbf{n}_2(s^*)]$ is the rotation matrix associated with the frame. This expression clearly separates motion along the path from motion orthogonal to it, which forms the foundation of the spatial reformulation.

Differentiating the path-relative position with respect to time and accounting for the motion of the Parallel Transport frame yields the reformulated velocity dynamics. Projecting the velocity into the moving frame leads to scalar equations governing the evolution of the path progress and transverse deviations[3]. The progress rate along the path is given by

$$\dot{s} = \frac{\mathbf{t}^T(s)\mathbf{v}}{1 - k_1(s)w_1 - k_2(s)w_2}, \quad (9)$$

while the transverse dynamics are described by

$$\dot{w}_1 = \mathbf{n}_1^T(s)\mathbf{v}, \quad (10)$$

$$\dot{w}_2 = \mathbf{n}_2^T(s)\mathbf{v}. \quad (11)$$

These expressions show how forward motion contributes to progress along the racing line, whereas lateral motion causes deviation from it.

3.5 Online Obstacle Avoidance

To handle dynamic environments, we exploit the structure of the racing obstacles. Since the obstacles (and gate frames) are modeled as vertical cylinders, collision avoidance can be decoupled from the vertical dynamics and treated strictly as a constraint on the lateral deviation $w_1(s)$. This reduces the 3D collision avoidance problem to determining a feasible interval for w_1 at each discretization step s_k .

We define the static track boundaries as $[\underline{w}_{1,\text{track}}, \bar{w}_{1,\text{track}}]$. To incorporate obstacles, we compute the signed transverse distance $d(s)$ from the reference path to the obstacle center. Given a safety radius r_s , we identify two candidate navigable intervals within the track limits:

$$\mathcal{I}_{\text{neg}} = [\underline{w}_{1,\text{track}}, \min(\bar{w}_{1,\text{track}}, d(s) - r_s)], \quad (12)$$

$$\mathcal{I}_{\text{pos}} = [\max(\underline{w}_{1,\text{track}}, d(s) + r_s), \bar{w}_{1,\text{track}}]. \quad (13)$$

The first interval \mathcal{I}_{neg} represents passing the obstacle on the right (negative deviation), while \mathcal{I}_{pos} represents passing on the left.

To ensure numerical stability and avoid chattering when the obstacle lies directly on the reference line ($d(s) \approx 0$), we introduce a switching threshold $\delta = 0.05$ m. The active safety bounds $[\underline{w}_1, \bar{w}_1]$ are updated according to the following policy:

$$[\underline{w}_1, \bar{w}_1] = \begin{cases} \mathcal{I}_{\text{neg}} & \text{if } d(s) \geq \delta, \\ \mathcal{I}_{\text{pos}} & \text{if } d(s) \leq -\delta, \\ \arg \max_{\mathcal{I} \in \{\mathcal{I}_{\text{neg}}, \mathcal{I}_{\text{pos}}\}} |\mathcal{I}| & \text{if } |d(s)| < \delta. \end{cases} \quad (14)$$

In the ambiguous region $|d(s)| < \delta$, this policy explicitly selects the side offering the maximum corridor width $|\mathcal{I}|$, ensuring the optimization always operates within the largest available convex region.

3.6 Discrete Optimisation

In this section, we describe the formulation of the time-optimal trajectory generation problem in an NMPC-based optimal control problem (OCP) approach. Using the flight corridor approach described above, obstacle constraints can be represented as simple bounds on the state vector, especially transverse coordinates along the path.

The time-optimal OCP in continuous space can then be defined as:

$$\min_{x(t), u(t), T} \int_0^T 1 dt \quad (15a)$$

$$\text{s.t. } x(0) = x_0, \quad x(T) = x_T \quad (15b)$$

$$T \geq 0, \quad t \in [0, T] \quad (15c)$$

$$\dot{x}(t) = f(x(t), u(t)), \quad t \in [0, T] \quad (15d)$$

$$\underline{u} \leq u(t) \leq \bar{u}, \quad t \in [0, T] \quad (15e)$$

$$\underline{w}_1 \leq w_1(t) \leq \bar{w}_1, \quad t \in [0, T] \quad (15f)$$

$$\underline{w}_2 \leq w_2(t) \leq \bar{w}_2, \quad t \in [0, T] \quad (15g)$$

$$\underline{\phi} \leq \phi(t) \leq \bar{\phi}, \quad t \in [0, T] \quad (15h)$$

$$\underline{\theta} \leq \theta(t) \leq \bar{\theta}, \quad t \in [0, T] \quad (15i)$$

$$\underline{\psi} \leq \psi(t) \leq \bar{\psi}, \quad t \in [0, T] \quad (15j)$$

where the quadcopter state space model is given by $x = [s, w_1, w_2, \dot{s}, \dot{w}_1, \dot{w}_2, \phi, \psi, \theta, \dot{\phi}, \dot{\psi}, \dot{\theta}]^T$. The control input is defined as $u = [\phi_{\text{rho}}, \theta_{\text{pitch}}, \psi_{\text{yaw}}, F_{\text{thrust}}]^T$. $\dot{x} = f(x, u)$ is the state equation. x_0 and x_T refer to the initial and terminal state of the quadcopter. \underline{u} and \bar{u} in equation (15e) represent the lower and upper bounds of the control commands.

Problem (15) is a free-horizon problem that is complicated to solve. Inspired by [7, 11], given an initial trajectory, time-optimality behaviour can be incited by tracking a slightly infeasible reference point on the trajectory, provided that the prediction horizon is long enough. [11] provided a detailed explanation of the concept.

In this paper, the NMPC technique is adopted for tracking a reference [3]. We utilise Nonlinear Programming (NLP) to approximate the OCP using the multiple-shooting approach [2]. The OCP is approximated as an NLP with the following linear-quadratic regulator (LQR) structure:

$$\min_{x_0:N, u_0:N-1} \sum_{k=0}^{N-1} \left((x_k - x_{k,\text{ref}})^T Q (x_k - x_{k,\text{ref}}) + u_k^T R u_k \right) + (x_N - x_{N,\text{ref}})^T Q_N (x_N - x_{N,\text{ref}}) \quad (16a)$$

$$\text{s.t. } x_0 = x_c, \quad (16b)$$

$$x_{k+1} = F(x_k, u_k, \Delta t), \quad (16c)$$

$$\underline{u} \leq u_k \leq \bar{u}, \quad (16d)$$

$$\underline{w}_1 \leq w_{1,k} \leq \bar{w}_1, \quad (16e)$$

$$\underline{w}_2 \leq w_{2,k} \leq \bar{w}_2, \quad (16f)$$

$$\underline{\phi} \leq \phi_k \leq \bar{\phi}, \quad (16g)$$

$$\underline{\theta} \leq \theta_k \leq \bar{\theta}, \quad (16h)$$

$$\underline{\psi} \leq \psi_k \leq \bar{\psi}, \quad (16i)$$

$$\forall k = 0, \dots, N-1 \quad (16j)$$

where x_c is the quadrotor's current state. F is the fourth-order Runge-Kutta numerical integrator for the spatial

quadrotor dynamics. Q , R , and Q_N are positive definite weighting matrices. We simulate the time-optimal behaviour by choosing a slightly infeasible standard reference $s_{k,\text{ref}} = s_k + \dot{s}_{k,\text{ref}}k$ for $k = 0, \dots, N$. $\dot{s}_{k,\text{ref}}$ is set to the maximum drone speed. N is the total number of steps in the prediction horizon. Thus, the reference state at the k -th interval is given by:

$$\underline{x}_{k,\text{ref}} = [s_{k,\text{ref}}, 0, 0, \dot{s}_{k,\text{ref}}, 0, 0, 0, 0, 0, 0]^T \quad (17)$$

where ϕ , ψ_{ref} , and θ_{ref} are dependent on the reference curve (typically zero for unconstrained flight paths) and the transverse coordinates w_1, w_2 are set to zero.

Upper and lower bounds of Euler angles are denoted by $\underline{\phi}, \bar{\phi}, \underline{\theta}, \bar{\theta}, \underline{\psi}$ and $\bar{\psi}$ in equations (16). Equations (16e) and (16f) represent the constraints on transverse coordinates, which are enforced using the flight corridor discussed in Section 2.4 as s -dependent state bounds, i.e., $\underline{w}_1(s), \bar{w}_1(s), \underline{w}_2(s)$ and $\bar{w}_2(s)$. Given that existing building information is already known, this process can be performed offline. Additionally, the corridor is meant to be updated online for real-time obstacle avoidance, where any incoming obstacles can be modeled using the same method.

4 Results

4.1 Experimental Setup

The baseline trajectory is defined by a cubic spline generated through three key points: the gate center, and two auxiliary points located 0.3 m before and 0.3 m after the gate center along the nominal waypoint direction. For this baseline, the drone maintains a constant commanded speed of 1.3 m s^{-1} .

In environments containing obstacles, the baseline method is augmented with an additional waypoint placed adjacent to each obstacle to ensure collision-free flight. To evaluate performance and robustness, we define two difficulty levels:

- **Level 1 (Deterministic):** No randomness is introduced; gate and obstacle poses are fixed at their nominal specifications.
- **Level 2 (Perturbed):** Realistic stochastic perturbations are applied to the ground-truth poses:
 - **Position:** $\pm 0.15 \text{ m}$ in x, y , and $\pm 0.1 \text{ m}$ in z .
 - **Orientation:** Roll $\pm 2.9^\circ$, Pitch $\pm 5.7^\circ$, and Yaw $\pm 11.5^\circ$.

Ground-truth poses of gates and obstacles are assumed to be known only when the drone is within a sensing radius of 0.7 m. To isolate the impact of obstacle-handling logic, we also evaluate both levels in obstacle-free environments (denoted as `level1.noObstacles` and

`level2.noObstacles`). These control environments are identical to their counterparts except for the removal of all obstacles. All reported metrics are averaged over 40 independent flights per condition to ensure statistical significance.

4.2 Quantitative Comparison

Table 1 summarizes the performance of the baseline and proposed methods for autonomous drone racing, evaluated across two difficulty levels (Level 1 and Level 2), both with and without obstacles.

The baseline method uses a fixed cubic spline trajectory passing through gate-centered waypoints, with pre-defined detours around known obstacles when present. It exhibits perfect temporal consistency (zero standard deviation) under nominal conditions. In Level 1 (deterministic gate and obstacle poses), it requires 16.52 s with obstacles and 14.16 s without, revealing a consistent time penalty of approximately 2.4 s due to obstacle avoidance maneuvers.

In the more realistic Level 2 setting (gate position perturbations of $\pm 0.15 \text{ m}$ and orientation perturbations of $\pm 0.1 \text{ m}$), success rate collapses to 25% (with obstacles) and 35% (without), demonstrating high sensitivity to pose uncertainty and limited online adaptation capability.

By contrast, the proposed method, which takes shortcuts by cutting the corners and dynamically slowing or speeding up along the path, achieves substantially better performance in both speed and robustness. In Level 1, it completes the course in 8.87 s (with obstacles) and 7.96 s (without) — corresponding to a speedup of 46–48% over the baseline — while attaining success rates of 90% and 100%, respectively. Even in the challenging Level 2 condition, it maintains mean times of 9.11 s (with obstacles) and 8.92 s (without) — still $\sim 45\%$ faster than the baseline’s nominal times — with success rates of 75% and 95%.

5 Ablation Study

5.1 Evaluation of Reference Path Generation

We evaluated several strategies for generating efficient global paths to guide our proposed method. Initial experiments focused on the *Complementary Progress Constraint* (CPC) formulation. This approach allows progress toward completion only when the drone is in proximity to a waypoint, explicitly incorporating the vehicle’s dynamics. However, this formulation introduces non-convexity for nonlinear system dynamics, leading to significant computational overhead. In our simulations, the solver frequently failed to converge within the real-time iteration limits, making it unsuitable for high-speed racing.[5]

Alternatively, we implemented a *Time-Optimal Gate-Traversing Planner*. Unlike fixed waypoints, this planner

Table 1: Performance comparison of baseline and proposed methods. Results are averaged over 40 independent runs per condition.

Method	Condition	Mean Time (s)	Std. Time (s)	Success Rate (%)
Baseline	Level 1	16.52	0.00	100
	Level 1 (No Obstacle)	14.16	0.00	100
	Level 2	16.52	0.00	25
	Level 2 (No Obstacle)	14.16	0.00	35

Proposed	Level 1	8.87	0.36	90
	Level 1 (No Obstacle)	7.96	0.00	100
	Level 2	9.11	0.45	75
	Level 2 (No Obstacle)	8.92	0.10	95

models racing gates as traversable geometric volumes, allowing the trajectory to pass through any point within the gate to minimize lap time. While this method reduced lap times by approximately 0.5 s to 1.0 s, the success rate dropped to roughly 60 % in simulation. Furthermore, the method exhibited poor generalization in real-world environments due to the extreme corner cutting which caused collisions with gate frame.[9]

Consequently, we adopted a more robust and standard approach of utilizing a *cubic spline* passing through gate centers and offset points along the normal through gate centres, which provided the optimal trade-off between aggressive speed and tracking reliability.

5.2 Reference Speed on Curves

To maintain feasibility during high-speed maneuvers, we investigated the impact of curvature-dependent velocity limits. Drones experience significant lateral acceleration when traversing curves; a constant high-speed reference often exceeds the physical limits of the motors or the tracking capabilities of the controller, leading to solver divergence or crashes.

To mitigate this, we constrained the reference velocity based on the local curvature κ and a maximum allowable lateral acceleration a_{\max} . The reference velocity v_{ref} is calculated as:

$$v_{\text{ref}} = \min \left(v_{\text{nominal}}, \sqrt{\frac{a_{\max}}{\kappa + \epsilon}} \right) \quad (18)$$

where ϵ is a small regularization constant to prevent division by zero in straight segments.

Implementing this adaptive speed profile resulted in a marginal increase in average lap time (approximately 0.2 s) but yielded a 10 % improvement in success rate. Given this significant boost in reliability with negligible performance loss, this curvature-aware reference was utilized for our final code.

6 Sim-to-Real Performance

The complete pipeline was successfully transferred and executed on the real drone hardware, as described in section 2. In the level-2 real-world environment, the average task completion time was approximately 0.3 seconds longer than in simulation, while achieving a success rate of 70%. This was primarily due to dynamic mismatch between simulation and real world.

Notably, the observed sim-to-real performance degradation remains significantly smaller compared to most state-of-the-art methods reported in the literature. We mainly attribute this small sim2real gap to:

- the deliberately conservative global path planner, and
- the robust reactive obstacle avoidance behavior employed during online execution.

7 Conclusion

In this work, we presented an autonomous drone racing framework that leverages spatial reformulation and Non-linear Model Predictive Control (NMPC) to achieve high-speed, collision-free flight. By reparameterizing quadrotor dynamics relative to a reference path using a Parallel Transport frame, we transformed complex 3D obstacle avoidance into a computationally efficient flight corridor problem.

Our experimental results demonstrate that the proposed method significantly outperforms traditional time-indexed baseline trajectories, achieving a reduction in lap time of approximately 45% – 48% while maintaining a high success rate under state uncertainty. The ablation study further highlights that while time-optimal gate-traversing planners offer theoretical speed gains, the combination of a conservative cubic spline global path with a curvature-aware NMPC provides the most reliable performance for real-world deployment.

References

- [1] Ross Allen and Marco Pavone. “A real-time framework for kinodynamic planning with application to quadrotor obstacle avoidance”. In: *AIAA Guidance, Navigation, and Control Conference*. 2016, p. 1374.
- [2] Hans Georg Bock and Karl-Jürgen Plitt. “A multiple shooting algorithm for direct solution of optimal control problems”. In: *Proceedings of the 9th IFAC World Congress*. Vol. 17. 2. 1984, pp. 242–247.
- [3] Y.Y. Chan et al. “Near time-optimal trajectory optimisation for drones in last-mile delivery using spatial reformulation approach”. In: *Transportation Research Part C: Emerging Technologies* 171 (2025), p. 104986. ISSN: 0968-090X. DOI: <https://doi.org/10.1016/j.trc.2024.104986>. URL: <https://www.sciencedirect.com/science/article/pii/S0968090X24005072>.
- [4] Matthias Faessler, Antonio Franchi, and Davide Scaramuzza. “Differential flatness of quadrotor dynamics subject to rotor drag for accurate tracking of high-speed trajectories”. In: *IEEE Robotics and Automation Letters* 3.2 (2017), pp. 620–626.
- [5] Philipp Foehn and Davide Scaramuzza. *CPC: Complementary Progress Constraints for Time-Optimal Quadrotor Trajectories*. 2020. arXiv: 2007.06255 [cs.R0]. URL: <https://arxiv.org/abs/2007.06255>.
- [6] Wojciech Giernacki et al. “Crazyflie 2.0 quadrotor as a platform for research and education in robotics and control engineering”. In: Aug. 2017, pp. 37–42. DOI: 10.1109/MMAR.2017.8046794.
- [7] Daniel Kloeser, Jesus Tordesillas, and Jonathan P How. “Time-optimal model predictive control for path following of a quadcopter”. In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2020, pp. 6454–6461.
- [8] Sikang Liu et al. “Search-based motion planning for quadrotors using linear quadratic minimum time control”. In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2017, pp. 3141–3148.
- [9] Chao Qin et al. *Time-Optimal Gate-Traversing Planner for Autonomous Drone Racing*. 2023. arXiv: 2309.06837 [cs.R0]. URL: <https://arxiv.org/abs/2309.06837>.
- [10] Charles Richter, Adam Bry, and Nicholas 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.
- [11] Robin Verschueren et al. “Towards time-optimal race car driving using nonlinear MPC in real-time”. In: *53rd IEEE Conference on Decision and Control*. IEEE. 2014, pp. 2505–2510.