Multi-stage Optimal Control Problem Formulation for Drone Racing Through Gates and Tunnels

Mathias Bos[1], Wilm Decré, Jan Swevers, Goele Pipeleers
MECO Research Team, Department of Mechanical Engineering, KU Leuven, Belgium
DMMS lab, *Flanders Make*, Leuven, Belgium
[1]mathias.bos@kuleuven.be, ORCID: 0000-0002-5471-6691

***Abstract*—Finding a time-optimal trajectory through gates and tunnels is a difficult challenge in autonomous drone racing, especially in a fully autonomous context where the computations are all performed onboard. This paper presents an optimal control problem formulation to represent and solve the motion planning problem for a multirotor drone racing through a series of gates or tunnels, without a priori knowledge about the drone pose right in front of or behind the gates nor about the exact time instance when to pass through the gates. The formulation is shown to produce feasible trajectories for a set of handcrafted configurations, and has the potential to be fast enough for online replanning purposes.**

***Index Terms*—Drones, Motion planning, Optimal control, Multi-stage, Gates**

I. Introduction

Drone racing is a spectacular sport for which the control research community has recently found an interest. One of the goals of autonomous drone racing is to defeat human pilots. The large interest expresses itself in a number of autonomous drone racing competitions such as those at the IROS and NeurIPS conferences and the AlphaPilot challenge [1]–[3]. In a drone race, the goal of each contender is to fly a race track in minimum time without crashing. A race track is composed of a sequence of gates, often rectangular, in a predefined order.

The literature on drone racing splits up into two main categories, each connected to an important challenge. Firstly, the localization of the drone and the perception of the environment, including the gates, is an important aspect of the aforementioned competitions, and is therefore under strong focus in the reports of contestants [4]–[6]. Secondly, the motion planning and control of drones consist of finding a trajectory that runs through the gates and is dynamically feasible given the platform's capabilities, and of generating control inputs that make the drone travel the trajectory as planned while rejecting disturbances and compensating for model imperfections. A fully autonomous drone performs all computations including state estimation, motion planning and control onboard, which adds the restriction of limited computational power. An example of a technique to speed up the computations in online optimal control is the use of real-time iteration (RTI) schemes [7]. The remainder of this paper considers the challenge of motion planning, while paying attention to computational efficiency and robustness. The drone pose, gate poses and gate geometries are assumed to be known.

Mellinger et al. [8], [9] presented strategies and successful experiments to race through hoops and narrow windows already in 2011. First in [8] they use polynomial representations of trajectories with "sufficiently smooth" derivatives and rely on differential flatness of the quadrotor dynamics to infer the control inputs from these polynomal trajectories. This approach does not allow for direct constraints on the control inputs (either thrust force and orientation or angular rates, or individual motor thrust forces). Later in [9] they represent trajectories as a sequence of segments, each with a controller parameterized by a goal state, tracking a desired velocity towards a window and tracking the desired orientation with constant thrust while passing through the window. This approach focuses on very accurate pose control through a narrow passage, with a recovery controller becoming active after the passage, rather than planning a full path through a gate taking into account what comes after the gate. Neunert et al. [10] consider trajectory generation and control through narrow windows with a simultaneous trajectory optimization and tracking control approach using fast Model Predictive Control (MPC). The full state of the drone right in front of and behind a gate, including the passing time, is considered given. A common difficulty that can render the planning problem quite complex, is that the times at which the drone will pass through the gates is a priori unknown. In Neunerts work the passing time of the waypoint is decided upon by a heuristic. A different motion planning approach that decouples motion planning and control instead of unifying them alleviates the computational burden of the motion planning by the use of motion primitives in a lattice, referred to as lattice-based motion planning [11].

Another approach to the gate passing problem is based on a more general obstacle avoidance technique: in [12] a gate is represented by surrounding walls or plates, which are avoided using a separating hyperplane formulation. The separating hyperplane approach to obstacle avoidance is elaborately employed in previous work of our research group on the OMG-tools motion planning toolbox [13]. However, for the very

This work has been carried out within the framework of projects Flanders Make SBO MULTISYSLECO: Multi-System Learning Control and Flanders Make SBO MULTIROB: Rigorous approach for programming and optimal control of multirobot systems. Flanders Make is the Flemish strategic research centre for the manufacturing industry. This work also benefits from project G0C4515N of the Research Foundation - Flanders (FWO - Flanders).

simplified quadrotor model with small angle approximation and only two edges of a gate, the order of magnitude of the computation time for a horizon of ten seconds using a cubic spline representation with ten polynomial intervals is already 200 ms. This computation time is sufficient for online replanning if additional reference tracking control is active, but it is expected to increase substantially when including full nonlinear dynamics and a full gate (or multiple gates) with four edges. In comparison, the replanning time in [10] is 25 ms, but there the gate is rather implicitly modeled via waypoints with a priori fully defined state. Moreover, the approach in [12] is not robust against varying initial and terminal (position) conditions: for a configuration where the optimal trajectory is strongly curved, the optimizer often fails to find a feasible solution [14].

Recent advances in (Deep) Reinforcement Learning (RL) have sparked new approaches to the drone racing challenge, most notably by Song et al. [15]. Rather than solving an optimization problem online to obtain a trajectory with corresponding control inputs, this approach applies a model-free policy search to learn a neural network policy by interaction with the environment in a large number of rollouts. Through a smart choice of rewards and techniques to allow for substantial data generation, it is possible to learn a policy that is applicable to unseen tracks. In practice, the neural network policy is trained in simulation using a high fidelity model of the drone dynamics. For experiments on the physical platform, a simulation rollout of the policy on the new track generates a reference trajectory, which is then tracked by the real drone with an MPC controller.

This paper proposes an optimal control problem formulation for the motion planning problem through a gate or a series of gates by representing the geometry of the gate by the free space rather than by the forbidden area. This formulation avoids having to know a priori specific waypoints or keyframes with a predefined state that guarantees safe passage through the gate. First, Section II describes the formulation of the optimal control problem as a multi-stage problem with convex box constraints. Next, Section III shows simulation results, which are discussed and related to the approaches described earlier.

II. Methodology

As the presented approach is model based, this section first discusses the used simplified dynamic quadrotor model. Next, it shows how the gates of a race track are parametrized. Finally, it discusses the multi-stage optimal control problem (OCP) formulation used to produce feasible, time optimal trajectories through the gates using the dynamic model.

A. Drone Dynamics

A derivation of the most commonly used basic quadrotor model is presented in [16]. In order to keep the optimization tractable to solve onboard, the continuous-time quadrotor dynamics $\boldsymbol{f}$ are expressed here by the following simplified quadrotor model, which relates the 3D position $\boldsymbol{p}$ and velocity $\boldsymbol{v}$ to the control inputs being the roll, pitch, yaw, and thrust acceleration references $\boldsymbol{u} = [\phi, \theta, \psi, a_t]^\top$:

$$\begin{aligned} \dot{\boldsymbol{x}} = \begin{bmatrix} \dot{\boldsymbol{p}} \\ \dot{\boldsymbol{v}} \end{bmatrix} &= \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}) \\ &= \begin{bmatrix} \boldsymbol{v} \\ \boldsymbol{a}_t - \boldsymbol{g} \end{bmatrix} \end{aligned} \tag{1}$$

Thrust acceleration a_t is defined as the mass-normalized thrust force from all propellers combined and $\boldsymbol{g}$ is the gravitational acceleration vector. For simplicity, aerodynamic drag is neglected. The thrust acceleration vector $\boldsymbol{a}_t$ is assumed to be oriented along the body z-axis, such that it is given in the world frame by $\boldsymbol{a}_t = \mathbf{R}\,[0, 0, a_t]^\top$. The rotation matrix $\mathbf{R}$ from the body to the world frame in terms of the roll-pitch-yaw Euler angles is, introducing $c_\gamma \triangleq cos(\gamma)$ and $s_\gamma \triangleq sin(\gamma)$,

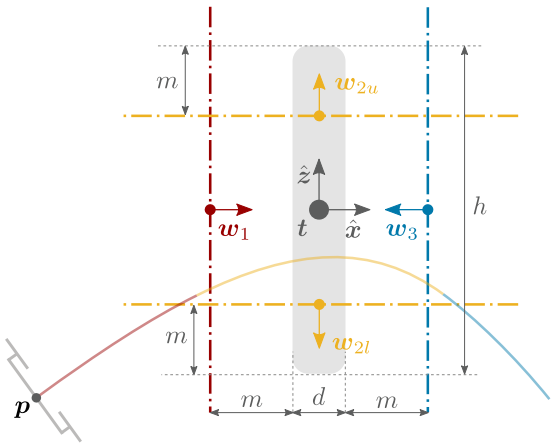
$$\begin{aligned} \mathbf{R} &= \mathbf{R}_\psi \mathbf{R}_\theta \mathbf{R}_\phi \\ &= \begin{bmatrix} c_\psi & -s_\psi & 0 \\ s_\psi & c_\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c_\theta & 0 & s_\theta \\ 0 & 1 & 0 \\ -s_\theta & 0 & c_\theta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & c_\phi & -s_\phi \\ 0 & s_\phi & c_\phi \end{bmatrix}. \end{aligned} \tag{2}$$

B. Gate Representation

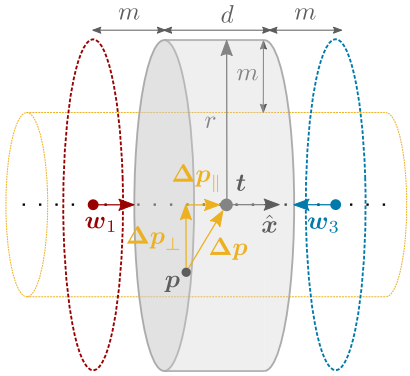
We present the mathematical description of two types of gate geometry: rectangular and circular. The multi-stage motion planning approach for both is fully equivalent, but their specific geometry requires a tailored description in order to formulate suitable path constraints.

1) Rectangular gates: In 2D, as illustrated in Fig. 1a, rectangular gates are defined through the following characteristics: their geometry, parametrized with the height and depth $\mathbf{\Gamma}_r = [h, d]^\top$, and their pose, comprising the position of the gate center $\boldsymbol{t}$ and the gate orientation. The latter is defined by the normal unit vector $\hat{\boldsymbol{x}}$, which points in the flight direction, and the unit vector $\hat{\boldsymbol{z}}$, which points parallel to the gate face. In 3D, the geometry is extended with the gate width w, such that $\mathbf{\Gamma}_r = [w, h, d]^\top$. w measures the gate opening along the unit vector $\hat{\boldsymbol{y}}$, which is perpendicular to $\hat{\boldsymbol{x}}$ and $\hat{\boldsymbol{z}}$ and therefore also points parallel to the gate face. The 3D orientation can be expressed in terms of the roll-pitch-yaw Euler angles $\mathbf{\Phi}_g = [\phi_g, \theta_g, \psi_g]^\top$ that represent the rotation from the world frame to the gate frame. The margin m determines how much a planned trajectory must at least be seperated from the gate edge. This should be more than the drone radius to avoid collision.

2) Circular Gates: Fig. 1b illustrates the representation of a circular gate. Gates of this type are defined through the following characteristics: their geometry, parametrized with the radius and depth $\mathbf{\Gamma}_c = [r, d]$, and their pose, comprising the position of the gate center $\boldsymbol{t}$ and the gate orientation defined by the normal vector $\hat{\boldsymbol{x}}$ which points in the flight direction. The margin m is defined in the same way as for a rectangular gate.



(a) 2D rectangular gate representation.



(b) Circular gate representation.

Fig. 1: Rectangular (in 2D) and circular gate representation. Gates are indicated in light gray.

C. Multi-stage Optimal Control Problem Formulation

With the drone dynamics and the representation of gates in place, the control problem can be formulated. For one gate, the time-optimal control problem is split up into three stages, similarly to the multi-frame approach by Mercy et al. for traveling through vast environments [17]: an *approach stage*, a *fly-through stage* and a *fly-away stage*. The drone position is subject to stage-specific box constraints, determined by the relative location with respect to the gate opening. In the approach stage, the drone position is restricted to the area in front of the gate. In the equivalent 2D case this is the area left of the red line in Fig. 1a. Similarly, in the fly-through stage, the position is restricted to the area between the yellow lines. For a circular gate this area is represented by the yellow cylinder in Fig. 1b. Finally, in the fly-away stage, the position is restricted to the area to the right of the blue line.

Mathematically, this is expressed as follows. Let $\boldsymbol{p}_h$ be the homogeneous coordinate representation of the position vector $\boldsymbol{p} = [p_x, p_y, p_z]^\top$, i.e. $\boldsymbol{p}_h = [p_x, p_y, p_z, 1]^\top$. Then the red, blue and yellow planes (in 2D equivalently lines) are formulated using the standard equation of a plane $w_x p_x + w_y p_y + w_z p_z + w_d = \boldsymbol{w}^\top \boldsymbol{p}_h = 0$. The plane parameter vectors $\boldsymbol{w}_j$ are derived from the gate representation by defining the normal vector $\boldsymbol{n}_j$ for each of the planes and a point $\boldsymbol{t}_j$ in each plane as follows:

$$
\begin{aligned}
&\text{Approach -} && \begin{cases} \boldsymbol{n}_1 &= \hat{\boldsymbol{x}} \\ \boldsymbol{t}_1 &= \boldsymbol{t} - (d/2 + m)\boldsymbol{n}_1 \\ \implies \quad \boldsymbol{w}_1 &= \begin{bmatrix} \boldsymbol{n}_1 \\ -\boldsymbol{n}_1^\top \boldsymbol{t}_1 \end{bmatrix} \end{cases} \\
&\text{Fly-through -} && \begin{cases} \boldsymbol{n}_{2u} &= \hat{\boldsymbol{z}} \\ \boldsymbol{t}_{2u} &= \boldsymbol{t} + (h/2 - m)\boldsymbol{n}_{2u} \\ \implies \quad \boldsymbol{w}_{2u} &= \begin{bmatrix} \boldsymbol{n}_{2u} \\ -\boldsymbol{n}_{2u}^\top \boldsymbol{t}_{2u} \end{bmatrix} \\ \boldsymbol{n}_{2l} &= -\hat{\boldsymbol{z}} \\ \boldsymbol{t}_{2l} &= \boldsymbol{t} + (h/2 - m)\boldsymbol{n}_{2l} \\ \implies \quad \boldsymbol{w}_{2l} &= \begin{bmatrix} \boldsymbol{n}_{2l} \\ -\boldsymbol{n}_{2l}^\top \boldsymbol{t}_{2l} \end{bmatrix} \end{cases} \\
&\text{Fly-away -} && \begin{cases} \boldsymbol{n}_3 &= -\hat{\boldsymbol{x}} \\ \boldsymbol{t}_3 &= \boldsymbol{t} - (d/2 + m)\boldsymbol{n}_3 \\ \implies \quad \boldsymbol{w}_3 &= \begin{bmatrix} \boldsymbol{n}_3 \\ -\boldsymbol{n}_3^\top \boldsymbol{t}_3 \end{bmatrix} \end{cases}
\end{aligned}
\tag{3}
$$

Each of the obtained $\boldsymbol{w}_j$ for stage j is now available to express a box constraint of the form $\boldsymbol{w}_j^\top \boldsymbol{p}_h \leq 0$. For brevity, (3) is given for a gate in 2D. Note that this formalism readily extends to 3D by adding two constraints for the sides of the gate, expressed in terms of the gate width w and the unit vector $\hat{\boldsymbol{y}}$.

For a circular gate, the fly-through stage formulates a tubular box constraint on the drone position in terms of the gate's position vector, normal vector, radius and margin expressed as

$$\Delta \boldsymbol{p}_\perp^\top \Delta \boldsymbol{p}_\perp \leq (r - m)^2 \tag{4}$$

where

$$
\begin{aligned}
\Delta \boldsymbol{p}_\perp &= \Delta \boldsymbol{p} - \Delta \boldsymbol{p}_\parallel \\
&= \Delta \boldsymbol{p} - (\Delta \boldsymbol{p} \cdot \hat{\boldsymbol{x}})\, \hat{\boldsymbol{x}} \\
&= (\boldsymbol{t} - \boldsymbol{p}) - ((\boldsymbol{t} - \boldsymbol{p}) \cdot \hat{\boldsymbol{x}})\, \hat{\boldsymbol{x}},
\end{aligned}
$$

with $\Delta \boldsymbol{p}$ the displacement vector between the drone position $\boldsymbol{p}$ and gate center $\boldsymbol{t}$, and $\Delta \boldsymbol{p}_\perp$ and $\Delta \boldsymbol{p}_\parallel$ its components perpendicular and parallel to the gate normal respectively, as illustrated in Fig. 1b.

In order to combine the three stages into one OCP, the boundary conditions are linked together by stitching constraints. They impose equality of the full dynamic state at the end and start of two subsequent stages. The exact point on the time horizon of the stitching points is a priori unknown, as they are taken as degrees of freedom in the OCP. This approach directly addresses the problem of deciding at what time on the horizon to pass through the gate. The sum of the stage times T_j constitutes the minimization objective. The terminal condition is selected as the hover state at a given point behind the gate, possibly the entrance to a next gate. The planning problem through one gate is readily extended to a planning problem

through multiple subsequent gates by including more stages and corresponding stitching constraints.

To pass from the continuous-time dynamics to a finite-horizon OCP, we discretize the dynamics using a multiple shooting scheme with fourth order Runge Kutta integration. Starting from the continuous-time dynamics $\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t))$, this yields the discrete-time dynamics $\boldsymbol{x}_{k+1} = \boldsymbol{F}(\boldsymbol{x}_k, \boldsymbol{u}_k)$. Control inputs are parametrized as first-order hold equivalents, such that their derivatives can be bounded to account for finite tracking speed of low-level reference tracking control loops.

The complete OCP for a race track with a series of rectangular gates is formally expressed as

$$
\begin{aligned}
\underset{\boldsymbol{x}_{j,k}, \boldsymbol{u}_{j,k}, T_j}{\text{minimize}} \quad & T = \sum_{j=1}^{M} T_j \\
\text{subject to:} \quad & \boldsymbol{x}_{1,0} = \boldsymbol{x}_0 \\
& \boldsymbol{x}_{1,k+1} = \boldsymbol{F}(\boldsymbol{x}_{1,k}, \boldsymbol{u}_{1,k}) && \text{for } k{=}0,1,\ldots,N_1{-}1 \\
& \dot{\boldsymbol{u}}_{min} \leq \dot{\boldsymbol{u}}_{1,k} \leq \dot{\boldsymbol{u}}_{max} && \text{for } k{=}0,1,\ldots,N_1{-}1 \\
& \boldsymbol{w}_1^\top \boldsymbol{p}_{h,1,k} \leq 0 && \text{for } k{=}0,1,\ldots,N_1 \\
& \boldsymbol{x}_{1,N_1} = \boldsymbol{x}_{2,0} \\
& \vdots \\
& \boldsymbol{x}_{j,k+1} = \boldsymbol{F}(\boldsymbol{x}_{j,k}, \boldsymbol{u}_{j,k}) && \text{for } k{=}0,1,\ldots,N_j{-}1 \\
& \dot{\boldsymbol{u}}_{min} \leq \dot{\boldsymbol{u}}_{j,k} \leq \dot{\boldsymbol{u}}_{max} && \text{for } k{=}0,1,\ldots,N_j{-}1 \\
& \boldsymbol{w}_j^\top \boldsymbol{p}_{h,j,k} \leq 0 && \text{for } k{=}0,1,\ldots,N_j \\
& \boldsymbol{x}_{j,N_j} = \boldsymbol{x}_{j+1,0} \\
& \vdots \\
& \boldsymbol{x}_{M,k+1} = \boldsymbol{F}(\boldsymbol{x}_{M,k}, \boldsymbol{u}_{M,k}) && \text{for } k{=}0,1,\ldots,N_M{-}1 \\
& \dot{\boldsymbol{u}}_{min} \leq \dot{\boldsymbol{u}}_{M,k} \leq \dot{\boldsymbol{u}}_{max} && \text{for } k{=}0,1,\ldots,N_M{-}1 \\
& \boldsymbol{w}_M^\top \boldsymbol{p}_{h,M,k} \leq 0 && \text{for } k{=}0,1,\ldots,N_M \\
& \boldsymbol{x}_{M,N_M} = \boldsymbol{x}_f,
\end{aligned}
\tag{5}
$$

where N_j is the horizon length for the j'th stage and M is the number of stages, three for a problem with one gate. The estimated initial state and the desired final state are denoted by $\boldsymbol{x}_0$ and $\boldsymbol{x}_f$ respectively. For circular gates, the constraints in the fly-through stage of the form $\boldsymbol{w}_j^\top \boldsymbol{p}_h \leq 0$ are replaced by a constraint as in (4).

The Rockit toolbox for the rapid prototyping of optimal control problems in Python presented in [18] allows for the formulation of free end-time and multi-stage OCPs, making it a useful tool for the OCP presented in this paper. The OCP is solved in Rockit with Python3.6 using IPOPT with HSL's ma57 linear solver with default initialization for the decision variables [19], [20].

III. Results and Discussion

To show that the suggested approach is successful in solving the motion planning problem and retrieving time-optimal trajectories, this section presents the solution to three challenging problem configurations. The computation times of these cases are discussed, as well as a consideration on the limitations on possible gate configurations.

Table I reports the average computation times over 10 runs for each of three problem configurations[1] [2]. In all cases, the margin is $m = 0.4$ m, the maximal thrust acceleration $\boldsymbol{a}_{t,max} = 5g$, and the maximal angular rate is $\dot{\phi}_{max} = \dot{\theta}_{max} = 6$ rad/s. The yaw angle is kept constant. The corresponding optimal trajectories in 3D are displayed in Fig. 2, 3 and 4 for a configuration with one rectangular gate, a configuration with two rectangular gates of varying dimensions (a short stretched gate and a narrow long tunnel) and a configuration with a rectangular and a circular gate respectively. Fig. 2 also highlights the three stages in red, yellow and blue for the one rectangular gate problem.

The reported computation times are promising for online use of this motion planning approach, given that all simulations are performed in Python and there is quite some efficiency to be gained through techniques for speeding up computations. Examples of such techniques are: the use of SQP solvers dedicated to optimal control problems, rather than IPOPT, in an RTI scheme; introducing sensible initial guesses that will benefit the efficiency of the SQP solver, which were left out in this work because of the very limited added value for the current toolchain; formulating the problem in C-code, rather than the significantly slower Python using the CasADi virtual machine. The automatic C-code generation capabilities of CasADi, upon which Rockit is built, allow for a direct translation from the Python code used for this paper [21].

However, even with these techniques the potential computation efficiency gain should not be estimated overly optimistic, especially for implementation on an onboard computational unit. Therefore, for the practical application of the presented motion planning technique an additional lower level tracking control loop executed at a higher rate is expected to increase performance, as explained in [10]. This control loop is capable of rejecting disturbances at a frequency higher than that at which the optimization can be computed, and can compensate for modeling errors in the open loop prediction of the optimized trajectory. This approach reduces the need for large improvements in computation times, as recomputing the OCP in a receding horizon fashion must then only cope with low frequency disturbances.

Compared to the separating hyperplanes approach as in [12], the proposed multi-stage gate passing approach scales much better towards multiple gates. That method needed 200 ms on average for a problem that only took into account two sides of a gate, and moreover scaled badly when including more obstacles, as shown in [14]. Moreover, the suggested approach finds solutions even for challenging problem configurations with narrow passages, whereas the separating hyperplanes approach struggles to find a feasible solution for even moderately difficult configurations.

[1] All computations are performed on a system with an AMD®Ryzen 7 pro 3700u processor with eight cores at 2.3 GHz and 29.4 GiB of RAM.

[2] All numerical quantities are expressed in terms of SI-units: distances and positions in m, velocities in m/s, accelerations in m/s^2, angles in rad.

TABLE I: OCP configurations and corresponding average computation times.

Ocp configuration	**Gate geometry & pose**	**Initital & terminal conditions**	**Horizon lengths**	**Avg. computation time (10 runs)**
One rectangular gate	$\boldsymbol{\Gamma}_r = [1, 1, 0.5]^\top$ $\boldsymbol{t} = [5, 5, 8]^\top$ $\boldsymbol{\Phi}_g = [0.2, -0.1, 0.5]^\top$	$\boldsymbol{x}_0 = [1, 1, 9, 0, 0, 0]^\top$ $\boldsymbol{x}_f = [9, 9, 1, 0, 0, 0]^\top$ $\boldsymbol{u}_0 = \boldsymbol{u}_f = [0, 0, 0, g]^\top$	$N_1 = 25$, $N_2 = 10$, $N_3 = 25$	246.39 ms
Two rectangular gates	$\boldsymbol{\Gamma}_{r,1} = [3, 0.9, 0.2]^\top$ $\boldsymbol{\Gamma}_{r,2} = [0.8, 1, 2]^\top$ $\boldsymbol{t}_1 = [3, 6, 8]^\top$ $\boldsymbol{t}_2 = [7, 5, 3]^\top$ $\boldsymbol{\Phi}_{g1} = [0.1, 0.2, 0.5]^\top$ $\boldsymbol{\Phi}_{g2} = [0, -0.1, 0.6]^\top$	$\boldsymbol{x}_0 = [1, 1, 9, 0, 0, 0]^\top$ $\boldsymbol{x}_f = [9, 2, 1, 0, 0, 0]^\top$ $\boldsymbol{u}_0 = \boldsymbol{u}_f = [0, 0, 0, g]^\top$	$N_1 = 20$, $N_2 = 10$, $N_3 = 20$, $N_4 = 10$, $N_5 = 20$	345.05 ms
One rectangular, one circular gate	$\boldsymbol{\Gamma}_{r,1} = [1, 0.7, 0.7]^\top$ $\boldsymbol{\Gamma}_{c,2} = [0.6, 0.5]^\top$ $\boldsymbol{t}_1 = [3, 2, 8]^\top$, $\boldsymbol{t}_2 = [6, 5, 5]^\top$, $\boldsymbol{\Phi}_{g1} = [0.1, 0.2, 0.2]^\top$, $\boldsymbol{\Phi}_{g2} = [0, 0.2, 1.5]^\top$	$\boldsymbol{x}_0 = [1, 1, 9, 0, 0, 0]^\top$, $\boldsymbol{x}_f = [9, 9, 5, 0, 0, 0]^\top$, $\boldsymbol{u}_0 = \boldsymbol{u}_f = [0, 0, 0, g]^\top$	$N_1 = 20$, $N_2 = 10$, $N_3 = 20$, $N_4 = 10$, $N_5 = 20$	384.17 ms

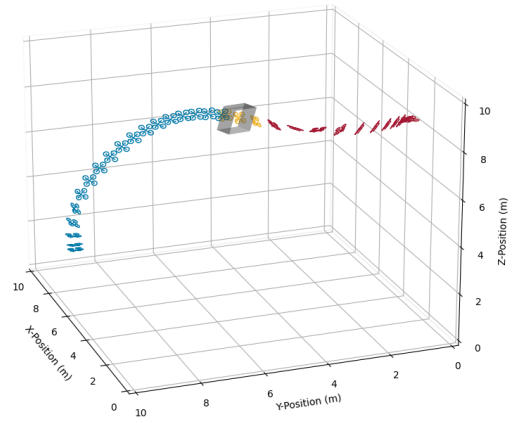


Fig. 2: Solution of the one rectangular gate problem, with the approach stage, fly-through stage and fly-away stage highlighted in red, yellow and blue respectively.

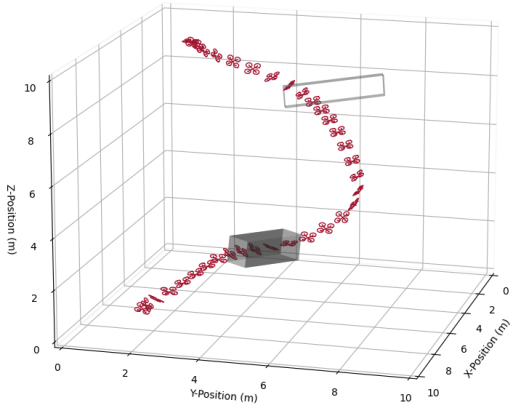


Fig. 3: Solution of the two rectangular gates problem with varying gate dimensions.

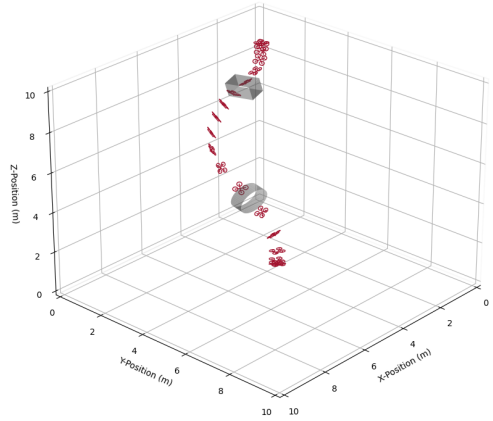


Fig. 4: Solution of the two gate problem with varying gate geometry.

Limitations to the Formulation

So far this discussion covered configurations that implicitly comply with the following limitations on the admissible problem configurations. Firstly, the initial drone state is such that its position is at the front side of the next gate. To clarify this, consider the situation depicted on the left in Fig. 5, where this assumption is violated. In that case, no feasible solution to the OCP exists. Secondly, it is assumed that the initial velocity is limited such that there exists a nonempty set of feasible control actions that satisfy the given constraints. Thirdly, it is assumed that the track layout is such that there are no gates or obstacles in between two successive gates, such that only those two gates must be taken into consideration for collision avoidance. Lastly, two subsequent gates are assumed to be separated by a distance no more than the order of magnitude of 10 m.

A sensible solution to overcome the first assumption is to

add an extra stage, or for more complex cases multiple extra stages, before the approach stage that leads to the front of the gate, as illustrated with the initial yellow part on the left in Fig. 5. During this stage, one of the two constraints defined by $\boldsymbol{w}_{0u}$ or $\boldsymbol{w}_{0l}$ as shown in the figure must be imposed, in this case $\boldsymbol{w}_{0u}$. The selection of which one of the two boils down to checking at which side of the gate the initial drone position is situated, by looking at the scalar product between the displacement vector $\boldsymbol{\Delta p}$ between the drone position and the gate center, and the normal vector corresponding to either $\boldsymbol{w}_{0u}$ or $\boldsymbol{w}_{0l}$. Mathematically this means selecting $\boldsymbol{w}$ corresponding to the $\boldsymbol{n}$ that satisfies

$$\boldsymbol{n} = \underset{\boldsymbol{n}_{0i}}{\operatorname{argmax}} \ \boldsymbol{n}_{0i}^{\top} \boldsymbol{\Delta p}. \tag{6}$$

The second assumption is satisfied automatically if the initial condition results from a previous planning step that accounts for recursive feasibility. For instance, if the previous plan ends in the hover state in front of the gate, there is always a nonempty set of feasible control actions for the next planning update. Alternatively, if the previous planning step accounted for the passing through multiple gates ahead, then feasible control actions through the next gate are already given. To avoid actually braking and coming to standstill at the planned hover location at the end of the trajectory, which would drastically deter the performance in a racing context, the trajectory must be recomputed with a new, shifted terminal condition long enough in advance before reaching this hover location. The longer the horizon over which the motion is planned, the lower the effect on the first part of the trajectory the terminal constraint will have. This motivates including multiple gates rather than only the first following gate, as for more included gates a closer approximation of the time-optimal trajectory over the full racetrack is achieved in each planning step.

The third assumption ensures that the configuration on the right in Fig. 5 can be handled by including a stage similarly as for the configuration on the left in the same figure. In this case two constraints $\boldsymbol{w}_u^{\top} \boldsymbol{p}_h \leq 0$ and $\boldsymbol{w}_l^{\top} \boldsymbol{p}_h \leq 0$ are selected similarly as before, here as a function of the displacement vector between the two gate centers $\boldsymbol{\Delta p}_g$:

$$\begin{aligned} \boldsymbol{n}_u &= \underset{\boldsymbol{n}_{1i}}{\operatorname{argmin}} \, \boldsymbol{n}_{1i}^{\top} \boldsymbol{\Delta p_g} \\ \boldsymbol{n}_l &= \underset{\boldsymbol{n}_{2i}}{\operatorname{argmax}} \, \boldsymbol{n}_{2i}^{\top} \boldsymbol{\Delta p_g}. \end{aligned} \tag{7}$$

Note that more elaborate checks and solutions are necessary for more specific or complex cases, such as two gates placed right next to each other without enough space between them to pass through. In that case, the decision of passing the gate at one side or the other would first have to be tackled, for instance by a higher level planner.

Regarding the last assumption, more spatially extended tracks require an increased planning horizon to keep the time spacing and geometric spacing of control grid points in the same range, at the cost of increased computational load. Or, alternatively, the motion planning problem can be solved for flying through the gates as presented in this paper, and in between the gates resort to planning using motion primitives. This approach is close to optimality as the dynamics will reach a steady state when traversing between two distant gates.

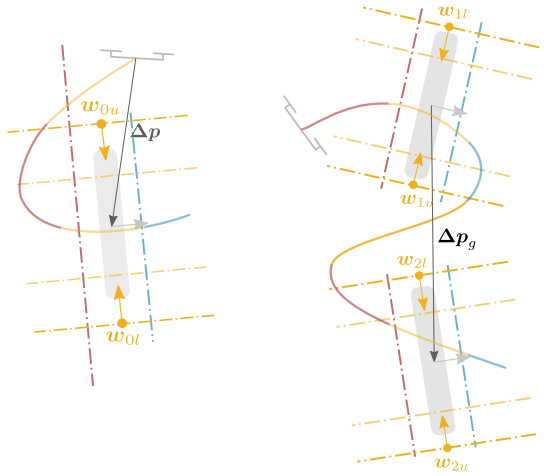


Fig. 5: Special cases of configurations.

IV. Conclusion

This paper presented an optimal control approach to the drone racing problem through gates in a known order. The proposed formulation tackles the problem of simultaneously generating control inputs and state trajectories with an approximate dynamic model. No a priori knowledge of the timing when to pass through the gates is required. The computation times are promising to enable onboard replanning.

Future work involves speeding up the computations by using for instance SQP solvers, rather than IPOPT, in an RTI scheme. Together with adding an approach to recede the planning horizon, this would enable online replanning in an MPC fashion. SQP solvers strongly benefit from initial guesses for the optimization variables, which were left out in this paper. Moreover, all computations in this paper were performed using the Python coding language and the CasADi virtual machine. Automatic C-code generation which is available in CasADi (upon which Rockit is built) is also expected to drastically decrease function evaluation times, which will contribute to a decrease in total computation time. The practical implementation on a physical experimental platform will demonstrate the viability of the presented approach.

References

[1] IROS 2018. Iros 2018 competitions. Accessed on: Aug. 26, 2021. [Online]. Available: https://www.iros2018.org/competitions

[2] H. J. Escalante and R. Hadsell, "NeurIPS 2019 competition and demonstration track: Revised selected papers," in *Proceedings of the NeurIPS 2019 Competition and Demonstration Track*, ser. Proceedings of Machine Learning Research, H. J. Escalante and R. Hadsell, Eds., vol. 123. PMLR, 08–14 Dec 2020, pp. 1–12.

[3] Lockheed Martin Corporation. Alphapilot AI drone innovation challenge. Accessed on: Aug. 26, 2021. [Online]. Available: https://www.lockheedmartin.com/en-us/news/events/ai-innovation-challenge.html

[4] P. Foehn, D. Brescianini, E. Kaufmann, T. Cieslewski, M. Gehrig, M. Muglikar, and D. Scaramuzza. (2020) Alphapilot: Autonomous drone racing. [Online]. Available: https://arxiv.org/abs/2005.12813

[5] E. Kaufmann, M. Gehrig, P. Foehn, R. Ranftl, A. Dosovitskiy, V. Koltun, and D. Scaramuzza, "Beauty and the beast: Optimal methods meet learning for drone racing," *2019 International Conference on Robotics and Automation (ICRA)*, pp. 690–696, 2019.

[6] F. Ölsner and S. Milz, "Catch me, if you can! A mediated perception approach towards fully autonomous drone racing," in *Proceedings of the NeurIPS 2019 Competition and Demonstration Track*, ser. Proceedings of Machine Learning Research, H. J. Escalante and R. Hadsell, Eds., vol. 123. PMLR, 08–14 Dec 2020, pp. 90–99.

[7] M. Diehl, H. Bock, H. Diedam, and P.-B. Wieber, *Fast Direct Multiple Shooting Algorithms for Optimal Robot Control*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 65–93.

[8] 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.

[9] 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.

[10] M. Neunert, C. de Crousaz, F. Furrer, M. Kamel, F. Farshidian, R. Siegwart, and J. Buchli, "Fast nonlinear model predictive control for unified trajectory optimization and tracking," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 2016, pp. 1398–1404.

[11] O. Andersson, O. Ljungqvist, M. Tiger, D. Axehill, and F. Heintz, "Receding-horizon lattice-based motion planning with dynamic obstacle avoidance," in *2018 IEEE Conference on Decision and Control (CDC)*, 2018, pp. 4467–4474.

[12] F. Giulietti, G. Pipeleers, G. Rossetti, and R. Van Parys, "Optimal autonomous quadrotor navigation in an obstructed space," in *2017 Workshop on Research, Education and Development of Unmanned Aerial Systems (RED-UAS)*, 2017, pp. 19–24.

[13] T. Mercy, R. Van Parys, and G. Pipeleers, "Spline-based motion planning for autonomous guided vehicles in a dynamic environment," *IEEE Transactions on Control Systems Technology*, vol. 26, no. 6, pp. 2182–2189, 2018.

[14] M. Bos, R. Beck, J. Swevers, and G. Pipeleers, "Interactive demo on the indoor localization, control and navigation of drones," Master's thesis, KU Leuven, Leuven, Belgium, 2019.

[15] Y. Song, M. Steinweg, E. Kaufmann, and D. Scaramuzza, "Autonomous drone racing with deep reinforcement learning," *CoRR*, vol. abs/2103.08624, 2021.

[16] X. Zhang, X. Li, K. Wang, and Y. Lu, "A survey of modelling and identification of quadrotor robot," *Abstract and Applied Analysis*, vol. 2014, p. 320526, Oct 2014.

[17] T. Mercy, E. Hostens, and G. Pipeleers, "Online motion planning for autonomous vehicles in vast environments," *2018 IEEE 15TH International workshop on advanced motion control (AMC)*, pp. 114–119, 2018.

[18] J. Gillis, B. Vandewal, G. Pipeleers, and J. Swevers, "Effortless modeling of optimal control problems with rockit," in *39th Benelux Meeting on Systems and Control*, Elspeet, The Netherlands, 2020.

[19] A. Wächter and L. T. Biegler, "On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming," Mathematical Programming, 106(1):2557, 2006, preprint at http://www.optimization-online.org/DB_HTML/2004/03/836.html.

[20] HSL. A collection of fortran codes for large scale scientific computation. Accessed on: Aug. 31, 2021. [Online]. Available: https://www.hsl.rl.ac.uk/

[21] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi – A software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation*, vol. 11, no. 1, pp. 1–36, 2019.