CPC: Complementary Progress Constraints for Time-Optimal Quadrotor Trajectories

Philipp Foehn, Davide Scaramuzza

***Abstract*—In many mobile robotics scenarios, such as drone racing, the goal is to generate a trajectory that passes through multiple waypoints in minimal time. This problem is referred to as time-optimal planning. State-of-the-art approaches either use polynomial trajectory formulations, which are suboptimal due to their smoothness, or numerical optimization, which requires waypoints to be allocated as costs or constraints to specific discrete-time nodes. For time-optimal planning, this time-allocation is a priori unknown and renders traditional approaches incapable of producing truly time-optimal trajectories. We introduce a novel formulation of progress bound to waypoints by a complementarity constraint. While the progress variables indicate the completion of a waypoint, change of this progress is only allowed in local proximity to the waypoint via complementarity constraints. This enables the simultaneous optimization of the trajectory and the time-allocation of the waypoints. To the best of our knowledge, this is the first approach allowing for truly time-optimal trajectory planning for quadrotors and other systems. We perform and discuss evaluations on optimality and convexity, compare to other related approaches, and qualitatively to an expert-human baseline.**

I. Introduction

Autonomous aerial vehicles are nowadays being used for inspection, delivery, cinematography, search-and-rescue, and recently even drone racing. The most prominent aerial system is the quadrotor, thanks to its vertical take-off and hover capabilities, its simplicity, and its versatility ranging from smooth maneuvers to extremely aggressive trajectories. However, quadrotors have limited range dictated by their battery capacity, which in turn limits how much time can be spent on a specific task. If the tasks consists of visiting multiple waypoints (delivery, inspection, drone racing [1, 2]), doing so in minimal time is often viable, and in the context of search and rescue or drone racing even the ultimate goal.

Truly time-optimal quadrotor trajectories exploit the full actuator potential at every point in time, which excludes using continuous polynomial formulations due to their inherent smoothness. This leaves planning time-discretized trajectories as the only option, which is typically solved using numerical optimization. Unfortunately, such formulations require the allocation of waypoints as costs or constraints to specific discrete time nodes, which is a priori unknown. We investigate this problem and provide a solution that allows to simultaneously optimize the trajectory and waypoint allocation. Our approach formulates a progress measure for each waypoint along the trajectory, indicating completion of a waypoint (see Fig. 1). We then introduce a complementary progress constraint, that allows completion only in proximity to a waypoint.

For simple point-mass systems, time-optimal trajectories can be computed in closed-form resulting in bang-bang acceleration trajectories, which can easily be sampled over multiple waypoints. However, quadrotors are underactuated systems that need to rotate to adjust their acceleration direction, which always lies in the body z-axis. Both the linear and rotational acceleration are controlled through the rotor thrusts, which are physically limited by the actuators. This introduces a coupling in the achievable linear and rotational accelerations. Therefore, time-optimal planning becomes the search for the optimal tradeoff between maximizing these accelerations.

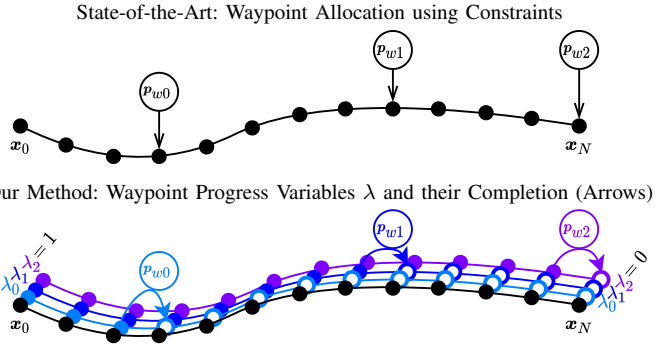

Fig. 1: Top: state-of-the-art fixed allocation of waypoints to specific nodes. Bottom: our method of defining one progress variable per waypoint. The progress variable can switch from 1 (incomplete) to 0 (completed) only when in proximity of the relevant waypoint, implemented as a complementary constraint (details in Fig. 3).

There exist two approaches in formulating the underlying state space for trajectory planning: (i) polynomial representations and (ii) discretized state space formulations.

The first approach exploits the quadrotor's differentially-flat output states represented by smooth polynomials. Their computation is extremely efficient and trajectories through multiple waypoints can be represented by concatenating polynomial segments, where sampling over segment times and boundary conditions provides minimal-time solutions. However, these polynomial, minimal-time solutions are still suboptimal for quadrotors, since they (i) are smooth and cannot represent rapidly-changing states, such as bang-bang trajectories, and (ii) can only touch the constant boundaries in infinitesimal points, but never stay at the limit for durations different than the total segment time. Both problems are visualized in Fig. 2 and further explained in Sec. II.

The second approach uses non-linear optimization to plan trajectories described by a time-discretized state space, where the system dynamics and input boundaries are enforced as constraints. In contrast to the polynomial formulation, this allows the optimization to pick any input within bounds for each discrete time step. For a time-optimal solution, the trajectory time t_N is part of the optimization variables and is the

sole term in the cost function. However, if multiple waypoints must be passed, these must be allocated as constraints to specific nodes on the trajectory. Since the fraction of overall time spent between any two waypoints is unknown, it is a priori undefined to which nodes on the trajectory such waypoint constraints should be allocated. This problem renders traditional discretized state space formulations ineffective for time-optimal trajectory generation through multiple waypoints.

In this work, we answer the following research questions:

1) How can we revise the waypoint constraints to optimize simultaneously time-allocation, while exploiting the system dynamics and actuation limits?
2) What are the resulting properties regarding initialization and convergence due to (non-) convex constraints?
3) What are the characteristics of time-optimal trajectories?

Contribution

Our approach optimizes a time-discretized state and input space for minimum execution time, with the quadrotor dynamics and input bounds implemented as equality and inequality constraints, respectively. Our contribution is a formulation of progress over the trajectory, where the completion of a waypoint is represented as a progress variable (see Fig. 1). We introduce a complementarity constraint, which allows the progress variable of a waypoint to switch from *incomplete* to *complete* only at the time nodes where the vehicle is within tolerance of the waypoint. This allows us to encapsulate the task of reaching multiple waypoints without specifying at which time a waypoint is reached, solving the time-allocation problem. We evaluate our approach experimentally in a multitude of scenarios, covering basic optimality and convergence tests, verifying time alignment of waypoints, many-waypoint scenarios, and finally comparing against human performance.

II. Preface: Time-Optimal Quadrotor Trajectory

A. Point-Mass Bang-Bang

Time-optimal trajectories encapsulate the best possible action to reach one or multiple targets in the lowest possible time. We first investigate a point mass in $\mathcal{R}^3$ controlled by bounded acceleration $\boldsymbol{a} \in \mathcal{R}^3 \mid \|\boldsymbol{a}\| \leq a_{max}$, starting at rest position $\boldsymbol{p}_0$ and translating to rest position $\boldsymbol{p}_1$. The time-optimal solution takes the form of a bang-bang trajectory over the time t_{opt}, accelerating with a_{max} for $t_{opt}/2$, followed by decelerating with a_{max} for $t_{opt}/2$. A trajectory through multiple waypoints can be generated similarly by optimizing over the switching times and intermittent waypoint velocities. For the general solution and applications we refer to [3].

B. Bang-Bang Relation for Quadrotors

A quadrotor would exploit the same maximal acceleration by generating the maximal possible thrust with its rotors. However, due to the quadrotor's underactuation, it can not instantaneously change the acceleration direction, but needs to rotate by applying differential thrust over its rotors. The linear and rotational acceleration are both controlled through the rotor thrusts, which in turn are limited, introducing a coupling

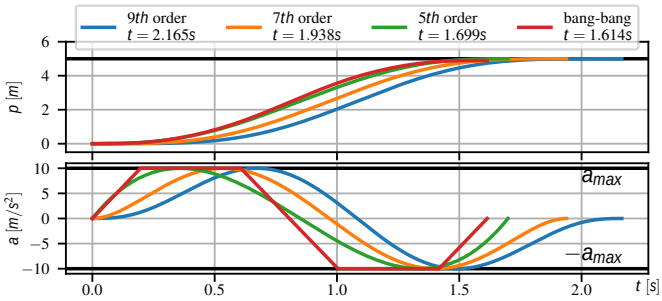

Fig. 2: Multiple orders of polynomial trajectories and one bang-bang trajectory with limited slope. The polynomial trajectories only touch the input extrema in two points, while the bang-bang spends more time at the limit and achieves a lower overall time.

of the achievable linear and rotational acceleration (visualized later in Fig. 4). Therefore, a time-optimal trajectory is the *optimal tradeoff between rotational and linear acceleration*, where the rotor thrusts only deviate from the maximum to adjust the rotational rates. Indeed, our experiments confirm exactly this behavior (see e.g. Fig, 5), as in Sec. VIII.

C. Sub-Optimality of Polynomial Trajectories

The quadrotor is a differentially flat system [4] that can be described based on its four flat output states, position and yaw. This allows to represent the evolution of the flat output states as smooth differentiable polynomials of the time t. To generate such polynomials, one typically defines its boundary conditions at start and end time, and solves for a smooth polynomial, optionally minimizing one of the derivatives, commonly the snap (4th derivative of position, as in [5]). The intention behind such minimum-snap trajectories is to minimize and smooth the needed body torques and, therefore, single motor thrust differences, which are dependent on the snap. Since the polynomials are extremely efficient to compute, trajectories through many waypoints can be generated by concatenating segments of polynomials, and minimal-time solutions can be found by optimizing or sampling over the segment times and boundary conditions. However, these polynomials are smooth by definition, which stands in direct conflict with maximizing the acceleration at all times while simultaneously adapting the rotational rate, as explained in the previous Sec. II-B. In fact, due to the polynomial nature of the trajectories, the boundaries of the reachable input spaces can only be touched at one or multiple points, or constantly, but not at subsegments of the trajectory. This problem is visualized in Fig. 2, and not only applies to polynomial trajectories, but partially also to direct collocation methods for optimization [6].

D. Real-World Deployment

Note that the time optimal solutions cannot be tracked on a real system, because disturbances could not be corrected, since the system is at the boundary of the reachable input space [3].

III. Related Work

A. Basic Quadrotor Control

The quadrotor model, its underactuated nature, and controllability have been extensively study in [7, 8] even with extensions to aerodynamic modelling [9, 10]. Control algorithms for quadrotors are well established and reach from cascaded control [11] to model predictive control (MPC) [12–14]. Such MPC systems allow to track even aggressive trajectories with high reliability, thanks to their prediction and optimization over a sliding time window. However, even those sophisticate control approaches rely on references from high-level planning approaches, considering the whole time horizon of a task.

B. Polynomial Trajectory Planning

The first category of such planning algorithms is based on polynomial trajectories. The work by [5, 4] established the now widely used minimum-snap trajectories, minimizing the 4th derivative of the position to generate smooth trajectories.

The approach by [15] builds on [5], but extends the cost on an n-th order derivative by the trajectory time, effectively achieving a trade-off between lowest time and smoothness. Exploiting polynomial formulations allows for extremely fast computation of trajectory solutions, which [16] further exploits by limiting the approach to fifth order polynomials, which can then be solved in closed form.

In [17] a multilevel system is proposed in the context of obstacle avoidance, first approximating the system dynamics by assuming acceleration as the bounded control input. The simplified system is used to sample minimum-time bang-bang trajectories, which are then refined and optimized for quadrotors using polynomial representations.

By searching over boundary and intermittent conditions, or scaling the polynomials, many different problems can be solved with fast sampling based approaches. However, as elaborated in Sec. II-C, the smoothness does conflict with the desired maximization of the acceleration.

C. Optimization-based Trajectory Planning

Trajectory planning through numerical optimization on discretized state spaces has been widely used in the field of manipulators [18], legged [19], and aerial robots [20].

To formulate trajectory planning as an optimization problem, multiple approaches exist, such as direct multiple shooting [21] and direct collocation [6]. Direct collocation uses low-order splines between discrete time nodes, which can be beneficial for certain problems, but is inferior when dealing with a high number of inequality constraints. Direct multiple shooting methods discretize the time horizon of the trajectory in N steps and include both the state and inputs in the optimization variables. The system dynamics are enforced as equality constraints over a timestep, and input limits are represented as inequality constraints.

Many approaches on numerical optimal control [13, 14] and trajectory generation [22, 20] deploy these schemes successfully for quadrotors. They typically formulate goals and tasks as cost and/or constraints on the state and input space. Smooth trajectories i.e. be enabled by penalizing high input deviations, similar to minimum-snap trajectories.

A simplified approach is followed by [23] in the context of multi-vehicle planning, where the optimization variables only contain the vehicles acceleration, and no rotation is represented. Position, velocity and jerk are defined as affine functions of the acceleration, rendering this scheme a single-shooting approach with an approximative model but computationally very efficient. Unfortunately it does only impose box-constraints on acceleration and jerk to approximate the actuation limits, which does not represent the thrust limitations well, rendering it unfit for time-optimal planning.

While these approaches allow to solve for complex tasks and maneuvers, they still rely on the user to allocate cost and constraints to specific time steps. Normally this is not a problem, since the costs are allocated to the whole time horizon and e.g. waypoint constraints can be allocated to specific times. However, this is no longer possible for the problem of time-optimal waypoint flight, since the exact time or fraction at which a waypoint is passed is a priori unknown.

D. Time-Optimal Approaches

For simple systems, so called bang-bang or bang-singular approaches have been widely established and proven to second order sufficient conditions as in [24, 25, 3]. These approaches apply to systems with bounded input space, breaking down to a boundary value problem (e.g. switching between maximal and minimal inputs), capable of producing closed form solutions for a multitude of dynamic systems. However, if the input bounds are not independent, these approaches become infeasible. [26] proposes an approach to generate time-optimal trajectories between two known states based on the aforementioned bang-bang approaches using numerical optimization of the switching times. However, the approach is only developed for 2-dimensional maneuvers in a plane and with collective thrust and bodyrates as independent input modalities, which do not correctly represent the limited thrust inputs of a quadrotor.

[27] represents the trajectory of a quadrotor as a convex combination of multiple paths given by analytical functions, which can additionally fulfill spatial constraints, such as obstacles. The authors compare their approach to the one of [26] an verify their experimental results. However also [27] limit their results to 2-dimensional experiments with only start and end states, without intermediate waypoints.

[28] provides a very thorough literature review and problem analysis, also solving the problem of time allocation to waypoints. This approach uses a geometric reference path and a change of variables that puts the state space of the quadrotor into a traverse dynamics formulation along this path. This allows to define progress along the track as the arc length along the geometric reference path, as well as writing cost and constraints based on the arc length and therefore independent of time. While this approach is very elegant, it approximates the quadrotor's actuation constraints as bodyrates and thrust limits, not realistically representing the rotor thrust bounds. Furthermore, the vehicles orientation is restricted in magnitude of the individual rotation axis, to keep the dynamics well-defined, since their representation uses Euler angles. This

limits the space of solutions and therefore the optimality of the resulting trajectory.

Finally, [29] provides a method to generate close to time-optimal segment times for polynomial trajectories using Bayesian Optimization. The approach is based on learning Gaussian Classification models predicting feasibility of a trajectory based on multi-fidelity data from analytic models, simulation and real-world data. While this approach puts high emphasis on real-world applicability and can even account for aerodynamic effects, it is still constrained to polynomials and requires real-world data specifically collected for the given vehicle. Furthermore, it is an approximative method rather than the true time-optimal solution.

IV. Problem Formulation

A. General Trajectory Optimization

The general optimization problem of finding the minimizer $\boldsymbol{x}^*$ for cost $J(\boldsymbol{x})$ in the state space $\boldsymbol{x} \in \mathcal{X}^n$ can be stated as

$$\begin{aligned} \boldsymbol{x}^* &= \min_x L(\boldsymbol{x}) \\ \text{subject to} \quad \boldsymbol{g}(\boldsymbol{x}) &= 0 \text{ and } \boldsymbol{h}(\boldsymbol{x}) \quad \leq 0 \end{aligned} \tag{1}$$

where $\boldsymbol{g}(\boldsymbol{x})$ and $\boldsymbol{h}(\boldsymbol{x})$ contain all equality and inequality constraints respectively. The full state space $\boldsymbol{x}$ is used equivalent to the term *optimization variables*. The cost $L(\boldsymbol{x})$ typically contains one or multiple quadratic costs on the deviation from a reference, costs on the systems actuation inputs, or other costs describing any desired behaviors.

1) Multiple Shooting Method: To represent a dynamic system in the state space, the system state $\boldsymbol{x}_k$ is described at discrete times $t_k = dt * k$ at $k \in (0, N)$, also called nodes, where its actuation inputs between two nodes are $\boldsymbol{u}_k$ at t_k with $k \in (0, N]$. The systems evolution is defined by the dynamics $\boldsymbol{f}_{dyn}(\boldsymbol{x}, \boldsymbol{u}) = \dot{\boldsymbol{x}}$, anchored at $\boldsymbol{x}_0 = \boldsymbol{x}_{init}$, and implemented as an equality constraint of the first order forward integration:

$$\boldsymbol{x}_{k+1} - \boldsymbol{x}_k - dt \cdot \boldsymbol{f}_{dyn}(\boldsymbol{x}_k, \boldsymbol{u}_k) = 0 \tag{2}$$

which is part of $\boldsymbol{g}(\boldsymbol{x}) = 0$ in the general formulation. Both $\boldsymbol{x}_k$, $\boldsymbol{u}_k$ are part of the state space and can be summarized as the vehicles dynamic states $\boldsymbol{x}_{dyn,k}$ at node k.

2) Integration Scheme: Additionally we can change the formulation to use a higher order integration scheme to suppress linearization errors for highly non-linear systems. In this work, we deploy a 4th order Runge-Kutta scheme:

$$\boldsymbol{x}_{k+1} - \boldsymbol{x}_k - dt \cdot \boldsymbol{f}_{RK4}(\boldsymbol{x}_k, \boldsymbol{u}_k) = 0 \tag{3}$$

$$\begin{aligned} \boldsymbol{f}_{RK4}(\boldsymbol{x}_k, \boldsymbol{u}_k) &= 1/6 \cdot (\boldsymbol{k}_1 + 2\boldsymbol{k}_2 + 2\boldsymbol{k}_3 + \boldsymbol{k}_4) \\ \boldsymbol{k}_1 &= \boldsymbol{f}_{dyn}(\boldsymbol{x}_k, \boldsymbol{u}_k) \\ \boldsymbol{k}_2 &= \boldsymbol{f}_{dyn}(\boldsymbol{x}_k + dt/2 \cdot \boldsymbol{k}_1, \boldsymbol{u}_k) \\ \boldsymbol{k}_3 &= \boldsymbol{f}_{dyn}(\boldsymbol{x}_k + dt/2 \cdot \boldsymbol{k}_2, \boldsymbol{u}_k) \\ \boldsymbol{k}_4 &= \boldsymbol{f}_{dyn}(\boldsymbol{x}_k + dt \cdot \boldsymbol{k}_3, \boldsymbol{u}_k). \end{aligned} \tag{4}$$

B. Time-Optimal Trajectory Optimization

Optimizing for a time-optimal trajectory means that the only cost term is the overall trajectory time $L(\boldsymbol{x}) = t_N$. Therefore, t_N needs to be in the optimization variables $\boldsymbol{x} = [t_N, \dots]^\intercal$, and must be positive $t_N > 0$. The integration scheme can then be adapted to use $dt = t_N/N$.

C. Passing Waypoints through Optimization

To generate trajectories passing through a sequence of waypoints $\boldsymbol{p}_{wj}$ with $j \in [0, \dots, M]$ called track, one would typically define a distance cost or constraint and allocate it to a specific state $\boldsymbol{x}_{dyn,k}$ at node k with time t_k. For cost-based formulations, quadratic distance costs are robust in terms of convergence and implemented as

$$L_{dist,j} = (\boldsymbol{p}_k - \boldsymbol{p}_{wj})^\intercal (\boldsymbol{p}_k - \boldsymbol{p}_{wj}) \tag{5}$$

where $\boldsymbol{p}_k$, part of $\boldsymbol{x}$, is the position state at a user defined time t_k. However, such a cost-based formulation is only a soft requirement and if summed with other cost terms does not imply that the waypoint is actually passed within a certain tolerance. To guarantee passing within a tolerance, constraint-based formulations can be used, such as

$$(\boldsymbol{p}_k - \boldsymbol{p}_{wj})^\intercal (\boldsymbol{p}_k - \boldsymbol{p}_{wj}) <= \tau_j^2 \tag{6}$$

which in the general problem is part of $\boldsymbol{h}(\boldsymbol{x}) \leq 0$, and requires the trajectory to pass by waypoint j at position $\boldsymbol{p}_{wj}$ within tolerance τ_j at time t_k.

D. *The Problem* of Time Allocation and Optimization

Adding waypoints to a trajectory means that their costs or constraints need to be allocated to specific nodes (at times t_k) on the trajectory. This fixes the time at which a waypoint is passed, or even if we optimize over the total time t_N, it still fixes the fractional time $t_k = t_N/N \cdot k$. However, it is not always possible to know a priori how much time or what fraction of the total time is spent between two waypoints. Therefore, it is not possible to allocate waypoint costs or constraints to nodes at specific times.

One approach is to spread the cost of reaching the waypoint over multiple nodes, which is done in [13] with an exponential weight spread. Even though this allows to shift the time of passing a waypoint, the width and mean of the weight spread are still user-defined, suboptimal in most scenarios, and does not allow to freely shift the time at which a waypoint is passed.

There are two possible ways to solve this: (i) change the time between fixed nodes, which implies changing dt for each timestep by adding all dt_k to the optimization variables; or (ii) change the node k to which a waypoint is allocated. The first option is to allow varying timeteps dt_k. This however negatively impacts the linearization quality and makes it inconsistent over the trajectory, allows trivial or suboptimal solutions, or even leads the optimization to exploit linearization errors. The second option requires a formulation that allows the optimization to change the node k to which a waypoint is allocated. Since this tackles the fundamental underlying allocation problem, we base our approach on this second option.

E. Problem Formulation Summary

In conclusion, the goal of this work is to find an optimization problem formulation which (i) satisfies the system dynamics and input limitations, (ii) minimizes total trajectory time, (iii) passes by multiple waypoints in sequence, and (iv) finds the optimal time at which to pass each waypoint.

V. Approach: Complementary Progress Constraints

Our approach consists of two main steps: (i) adding a measure of progress throughout the track, and (ii) adding a measure of how this progress changes. In the following sections we explain how these two steps can be formulated and added to an numerical optimization problem.

A. Progress Measure Variables

To describe the progress throughout a track we want a measure that fulfills the following requirements: (i) it starts at a defined value, (ii) it must reach a different value by the end of the trajectory, and (iii) it can only change when a waypoint is passed within a certain tolerance. To achieve this, let the vector $\boldsymbol{\lambda}_k \in \mathcal{R}^M$ define the progress variables λ_k^j at timestep t_k for all M waypoints indexed by j. All progress variables start at 1 as in $\boldsymbol{\lambda}_0 = \mathbf{1}$ and must reach 0 at the end of the trajectory as in $\boldsymbol{\lambda}_N = \mathbf{0}$. The progress variables $\boldsymbol{\lambda}$ are chained together and their evolution is defined by

$$\boldsymbol{\lambda}_{k+1} = \boldsymbol{\lambda}_k - \boldsymbol{\mu}_k \tag{7}$$

where the vector $\boldsymbol{\mu}_k \in \mathcal{R}^M$ indicates the progress step at every timestep. Note that the progress can only be positive, therefore $\mu_k^j \leq 0$. Both $\boldsymbol{\lambda}_k$ and $\boldsymbol{\mu}_k$ for every timestep are part of the optimization variables $\boldsymbol{x}$, which replicates the multiple shooting scheme for the progress variables.

To define when and how the progress variables can change, we now imply constraints on $\boldsymbol{\mu}_k$, in it's general form as

$$\boldsymbol{\epsilon}_k^- \leq \boldsymbol{f}_{prog}(\boldsymbol{x}_k, \boldsymbol{\mu}_k) \leq \boldsymbol{\epsilon}_k^+ \tag{8}$$

where $\boldsymbol{\epsilon}^-$, $\boldsymbol{\epsilon}^+$ can form equality or inequality constraints.

Finally, to ensure that the waypoints are passed in the given sequence, we enforce subsequent progress variables to be bigger than their prequel at each timestep by

$$\mu_k^j \leq \mu_k^{j+1} \forall k \in [0, N], j \in [0, M). \tag{9}$$

B. Complementary Progress Constraints

In the context of waypoint following, the goal is to allow $\boldsymbol{\mu}_k$ to only be non-zero at the time of passing a waypoint. Therefore, $\boldsymbol{f}_{prog}$ and $\boldsymbol{\epsilon}^- = \boldsymbol{\epsilon}^+ = 0$ are chosen to represent a *complementarity constraint*, as

$$\boldsymbol{f}_{prog}(\boldsymbol{x}_k, \boldsymbol{\mu}_k) = \left[\mu_k^j \cdot \|\boldsymbol{p}_k - \boldsymbol{p}_{wj}\|_2^2\right]_{j\in[0,M)} \stackrel{!}{=} 0 \tag{10}$$

which can be interpreted as a mathematical "*or*" function, since either μ_k^j or $\|\boldsymbol{p}_k - \boldsymbol{p}_{wj}\|$ must be 0. Intuitively, the two elements *complement* each other.

C. Tolerance Relaxation

With Eqn. 10 the trajectory is forced to pass *exactly* through a waypoint. Not only is this impractical, since often a certain tolerance is admitted or even wanted, but it also negatively impacts the convergence behavior and time-optimality, since the system dynamics are discretized and one of the discrete timesteps must coincide with the waypoint. Therefore, it is desirable to relax a waypoint constraint by a certain tolerance which is achieved by extending Eqn. 10 to

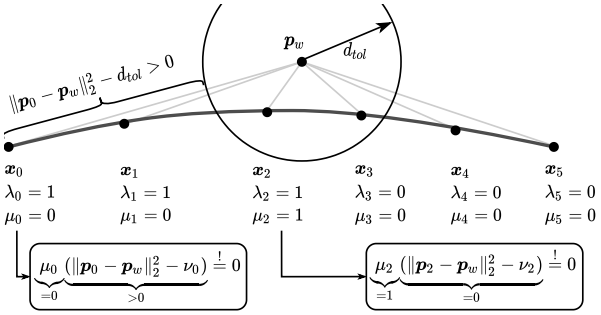

Fig. 3: Illustration of the Complementary Progress Constraint. μ can only be non-zero if the distance to the waypoint $\boldsymbol{p}_w$ is less than the tolerance d_{tol}. This is not the case for $\boldsymbol{x}_0$, but for $\boldsymbol{x}_1$, and allowing the progress variable to switch to 0 (complete).

$$\boldsymbol{f}_{prog}(\boldsymbol{x}_k, \boldsymbol{\mu}_k) = \left[\mu_k^j \cdot \left(\|\boldsymbol{p}_k - \boldsymbol{p}_{wj}\|_2^2 - \nu_k^j\right)\right]_{j\in[0,M)} \quad \text{subject to} \quad 0 \leq \nu_k^j \leq d_{tol}^2 \tag{11}$$

where ν_k^j is a slack variable to allow the distance to the waypoint to be relaxed to zero when it is smaller than d_{tol}, the maximal distance tolerance. This now enforces that the progress variables can not change, except for the timesteps at which the system is within tolerance to the waypoint.

VI. Full Problem Formulation

The full space of optimization variables $\boldsymbol{x}$ consists of the overall time and all variables assigned to nodes k as $\boldsymbol{x}_k$. All nodes k include the robots dynamic state $\boldsymbol{x}_{dyn,k}$, its inputs $\boldsymbol{u}_k$, and all progress variable, therefore:

$$\boldsymbol{x} = [t_N, \boldsymbol{x}_0, \ldots, \boldsymbol{x}_N] \tag{12}$$

where

$$\boldsymbol{x}_k = \begin{cases} [\boldsymbol{x}_{dyn,k}, \boldsymbol{u}_k, \boldsymbol{\lambda}_k, \boldsymbol{\mu}_k, \boldsymbol{\nu}_k] & \text{for } k \in [0, N) \\ [\boldsymbol{x}_{dyn,N}, \boldsymbol{\lambda}_N] & \text{for } k = N. \end{cases} \tag{13}$$

Based on this representation, we write the full problem as

$$\boldsymbol{x}^* = \min_{\boldsymbol{x}} t_N \tag{14}$$

subject to the system dynamics and initial constraint

$$\boldsymbol{x}_{k+1} - \boldsymbol{x}_k - dt \cdot \boldsymbol{f}_{RK4}(\boldsymbol{x}_k, \boldsymbol{u}_k) = 0 \tag{15}$$

$$\boldsymbol{x}_0 = \boldsymbol{x}_{init}, \tag{16}$$

the input constraints

$$\boldsymbol{u}_{min} - \boldsymbol{u}_k \leq 0 \qquad \boldsymbol{u}_k - \boldsymbol{u}_{max} \leq 0, \tag{17}$$

the progress evolution, boundary, and sequence constraints

$$\boldsymbol{\lambda}_{k+1} - \boldsymbol{\lambda}_k + \boldsymbol{\mu}_k = 0 \tag{18}$$

$$\boldsymbol{\lambda}_0 = 0 \quad \boldsymbol{\lambda}_N - 1 = 0 \tag{19}$$

$$\lambda_k^j - \lambda_k^{j+1} \leq 0 \quad \forall j \in [0, m), \tag{20}$$

and the complementarity constraint with tolerance

$$\mu_k^j \cdot \left(\|\boldsymbol{p}_k - \boldsymbol{p}_{wj}\|_2^2 - \nu_k^j\right) = 0 \tag{21}$$

$$-\nu_k^j \leq 0 \quad \nu_k^j - d_{tol}^2 \leq 0. \tag{22}$$

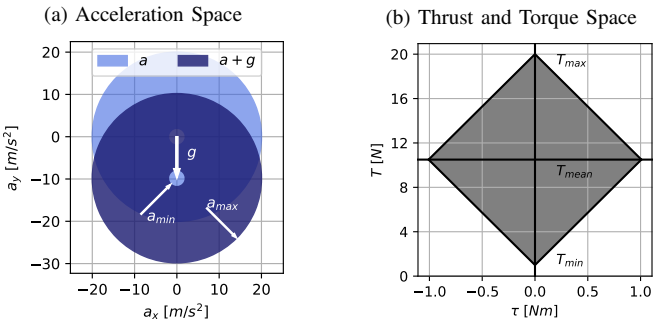

Fig. 4: Acceleration- (Fig.4a) and thrust/torque-space (Fig. 4b) of the STD quadrotor configuration (see Tab. I). Note that the acceleration space in Fig. 4a is non-convex due to $T_{min} > 0$ and the thrust and torque limits are dependent on each other in 4b.

VII. Application to Quadrotors

To apply our time-optimal planning method to a quadrotor, we first define its state space, input modality, and dynamics.

A. Quadrotor Dynamics

The quadrotor's state space is described between the inertial frame I and body frame B, as $\boldsymbol{x} = [\boldsymbol{p}_{IB}, \boldsymbol{q}_{IB}, \boldsymbol{v}_{IB}, \boldsymbol{w}_B]^\top$ corresponding to position $\boldsymbol{p}_{IB} \in \mathcal{R}^3$, unit quaternion rotation $\boldsymbol{q}_{IB} \in \mathcal{R}^4$ given $\|\boldsymbol{q}_{IB}\| = 1$, velocity $\boldsymbol{v}_{IB} \in \mathcal{R}^3$, and bodyrate $\boldsymbol{\omega}_B \in \mathcal{R}^3$. The input modality is on the level of collective thrust $\boldsymbol{T}_B = [00T_{Bz}]^\mathsf{I}$ and body torque $\boldsymbol{\tau}_B$. From here on we drop the frame indices since they are consistent throughout the description. The dynamic equations are given by

$$\dot{\boldsymbol{p}} = \boldsymbol{v} \qquad \dot{\boldsymbol{q}} = \frac{1}{2}\boldsymbol{\Lambda}(\boldsymbol{q}) \begin{bmatrix} 0 \\ \boldsymbol{\omega} \end{bmatrix} \tag{23}$$

$$\dot{\mathbf{v}} = \mathbf{g} + \frac{1}{m}\mathbf{R}(\mathbf{q})\mathbf{T} \qquad \dot{\boldsymbol{\omega}} = \mathbf{J}^{-1}\left(\boldsymbol{\tau} - \boldsymbol{\omega} \times \mathbf{J}\boldsymbol{\omega}\right) \tag{24}$$

where $\boldsymbol{\Lambda}$ represents a quaternion multiplication, $\boldsymbol{R}(\boldsymbol{q})$ the quaternion rotation, m the quadrotor's mass, and $\boldsymbol{J}$ its inertia.

B. Quadrotor Inputs

The input space given by $\boldsymbol{T}$ and $\boldsymbol{\tau}$ is further decomposed into the single rotor thrusts $\mathbf{u} = [T_1, T_2, T_3, T_4]$. where T is the thrust at rotor $i \in \{1, 2, 3, 4\}$.

$$\boldsymbol{T} = \begin{bmatrix} 0 \\ 0 \\ \sum T_i \end{bmatrix} \quad \text{and} \quad \boldsymbol{\tau} = \begin{bmatrix} l/\sqrt{2}(T_1 + T_2 - T_3 - T_4) \\ l/\sqrt{2}(-T_1 + T_2 + T_3 - T_4) \\ c_\tau(T_1 - T_2 + T_3 - T_4) \end{bmatrix} \tag{25}$$

with with the quadrotor's arm length l and the rotor's torque constant c_τ. The quadrotor's actuators limit the applicable thrust for each rotor, effectively constraining T_i as

$$0 \leq T_{min} \leq T_i \leq T_{max} \tag{26}$$

where T_{min}, T_{max} are positive for typical quadrotor setups. In Fig. 4 we visualize the acceleration space and the thrust torque space of a quadrotor in the xz-plane. Note that, the acceleration space in Fig. 4a is not convex due to $T_min > 0$ for the depicted model parameters from the STD configuration of Tab. I. The torque space is visualized in Fig. 4b, where the coupling between the achievable thrust and torque is visible.

TABLE I: Quadrotor Configurations

Property	RQ	MS	SIM	STD
m [kg]	0.76	1.0	3.2	1.0
l [m]	0.17	0.23	0.232	0.15
$diag(J)$ [gm^2]	$[3, 3, 5]$	$[10, 10, 20]$	$[50, 23, 67]$	$[5, 5, 10]$
T_{min} [N]	0.0	0.0	0.5	0.25
T_{max} [N]	16.0	4.179	12	5.0
c_τ [1]	0.01	0.0133	0.0133	0.01
ω_{max} [rad s^{-1}]	15	10	3	10
v_{max} [m s^{-1}]	42	19	20	–

C. Approximative Linear Aerodynamic Drag

Finally, we extend the quadrotor's dynamics to include a linear drag model, to approximate the most dominant aerodynamic effects with drag coefficient c_D by

$$\dot{\boldsymbol{v}} = \boldsymbol{g} + \frac{1}{m}\boldsymbol{R}(\boldsymbol{q})\boldsymbol{T} - c_D * \boldsymbol{v}. \tag{27}$$

We approximate $c_D = \sqrt{(4 * T_max/m)^2 - g^2}/v_{max}$ to cancle the full thrust in horizontal steady-state flight at v_{max}.

VIII. Experimental Evaluation

To demonstrate the capabilities and applicability of our method, we test it on a series of experiments. We first evaluate a simple point-to-point scenario and compare to [26, 27] in Sec. VIII-A. Next we investigate the time alignment for multiple waypoints in Sec. VIII-B, followed by an experimental investigation of the convergence characteristics on short tracks in terms of initialization in Sec. VIII-C and (non-) convexity in Sec VIII-D. Then we demonstrate applicability to longer tracks with ≥ 10 waypoints (Sec. VIII-E, VIII-F). Finally, we compare time-optimal trajectories to human trajectories flown in simulation (Sec. VIII-G and real-world (Sec. VIII-H).

All evaluations are performed using CASADI [30] with IPOPT [31] as solver backend. We use multiple different quadrotor configurations, as listed in Tab. I. The first configuration represents a typical race quadrotor, the second one is parameterized after the MicroSoft AirSim [32] SimpleFlight quadrotor, and the third one resembles the drone used in simulation flown by a human pilot. The last standard "STD" configuration resembles the one used in [26, 27].

Initialization Setup

If not stated differently, the optimization is initialized with identity orientation, zero bodyrates, $1\,\mathrm{m\,s^{-1}}$ velocity, linearly interpolated position between the waypoints, and hover thrusts. The total time is set as the distance through all waypoints divided by the velocity guess. The node of passing a waypoint (respectively where the progress variables $\boldsymbol{\lambda}$ switch to zero) is initialized as equally distributed, i.e. for waypoint j the passing node is $k_j = N * j/M$. We define the total number of nodes N based on the number of nodes per waypoint N_w so that $N = MN_w$. We typically chose roughly $N_w \in (50, 100)$ nodes per waypoint, to get a good linearization depending on the overall length, complexity, and time of the trajectory. Note that high numbers of $N_w >> 100$ help with convergence and achieved stability of the underlying optimization algorithm, but on longer tracks can cause very long computation times of multiple hours on average laptop computers in 2020.

p_x	Time of [26]	Time of [27]	Time of ours
3 m	0.898 s	0.890 s	0.918 s
6 m	1.231 s	1.223 s	1.255 s
9 m	1.488 s	1.478 s	1.517 s
12 m	1.705 s	1.694 s	1.736 s
15 m	1.895 s	1.885 s	1.933 s

TABLE II: Comparison of the resulting timings between our approach and [26, 27]. Note that our approach is 2.00% slower than [26] and 2.68% slower than [27], because it accounts for rotation dynamics and true actuator limits.

A. Time-Optimal Hover-to-Hover Trajectories

We first evaluate trajectory generation between two known position states in hover, one at the origin, and one at $p_x = [3, 6, 9, 12, 15]\text{m}$, as in [26, 27]. Additionally to the problem setup explained in Sec. V, we add constraints to the end state to be in hover, i.e. $\boldsymbol{v}_N = \mathbf{0}$ and $\boldsymbol{q} = [1\mathbf{0}]$. We use $N = N_w = 50$ nodes and a tolerance of $d_{tol} = 10^{-3}$. Different from [26, 27], we compute the solution in full 3D space, which however does not matter for this experiment, since the optimal trajectory stays withing the y-plane. We defined the model properties so that it meets the maximal and minimal acceleration $[a_{min}, a_{max}] = [1, 20]\text{m s}^{-2}$ and maximal bodyrate $\omega_{max} = 10\,\text{rad s}^{-1}$ as in [26, 27], given by our STD configuration (see Tab. I).

The solutions are depicted in the xy-plane in Fig. 6 and the timings are stated and compared to [26, 27] in Tab. II. Note that our approach is 2.00% slower than [26] and 2.68% slower than [27] because, differently from those works, it also models the full rotational dynamics and accounts for realistic actuation limits.

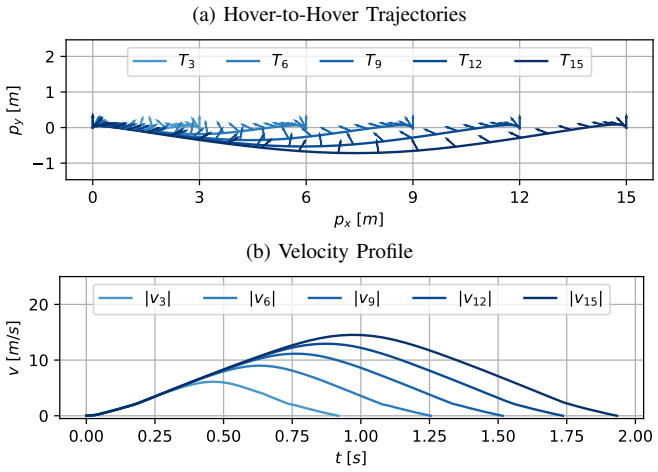

Fig. 6: Time-optimal hover-to-hover trajectories between states spaced $p_x = [3, 6, 9, 12, 15]\text{m}$ apart as in [26, 27]. The top Fig. 6a depicts the position on the xy-plane, while the lower Fig. 6b depicts the velocity profile.

B. Optimal Time Allocation on Multiple Waypoints

In this experiment, we define a straight track between origin and $p_x = 50\,\text{m}$ trough multiple waypoints. The goal is to show how that our method can choose the optimal time at which a waypoint is passed, and we expect to see both setups to converge to the same solution. Therefore, we test two different distributions of the waypoints $\boldsymbol{p}_{wj}$ over the straight track; specifically, we define a regular ($p_{x,reg} = [1, 20, 30, 40, 50]\text{m}$), and an irregular ($p_{x,ireg} = [10, 15, 20, 25, 50]\text{m}$) distribution. We chose $N = 125$ and a tolerance of $d_{tol} = 0.4\,\text{m}$, with the STD quadrotor (see Table I).

As expected, both setups converge to the same solution of $t_N = 2.430\,\text{s}$, depicted in Fig. 5, with an equal trajectory and inputs, despite the different waypoint distribution. Since the waypoints are located at different intervals, we can observe a different distribution of the progress variables in Fig. 5, while the trajectory time, dynamic states, and inputs stay the same.

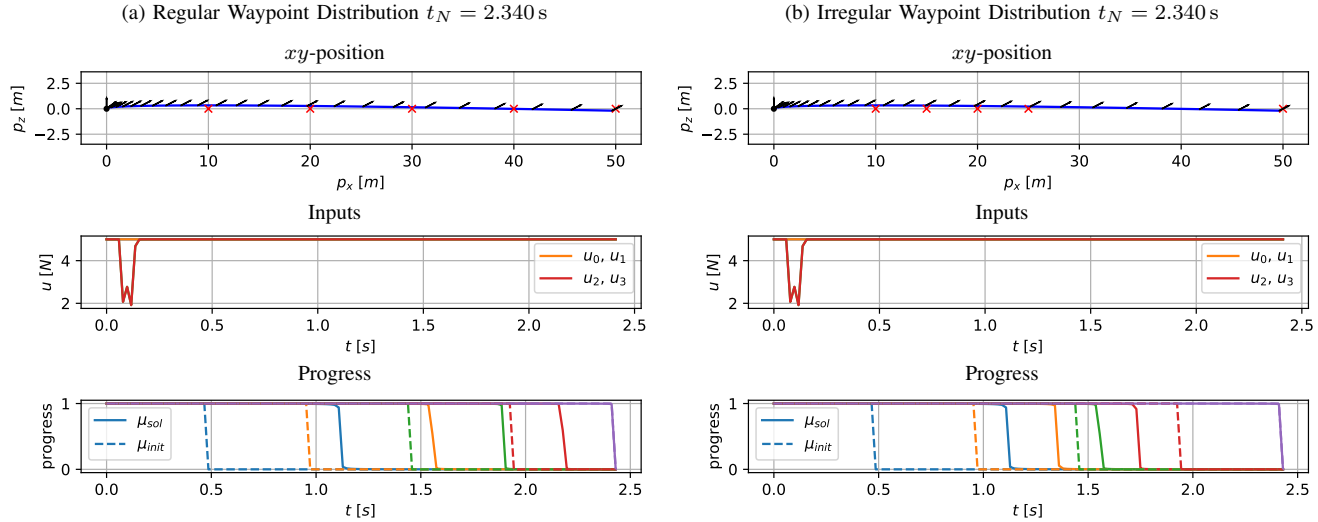

Fig. 5: A trajectory along waypoints distributed on a line over $50\,\text{m}$, flown in $< 2.430\,\text{s}$. In Fig. 5a the waypoints are equally distributed over the total distance, while in Fig. 5b the first 4 of 5 waypoints lie withing the first half of the total distance. The bottom plot depicts the progress variables as initialized (dashed $--$) and as in the final solution (solid $-$). Note that both settings converge to the exact same solution, while the time of passing the waypoints was significantly adjusted from the initialization, and is differs between the settings.

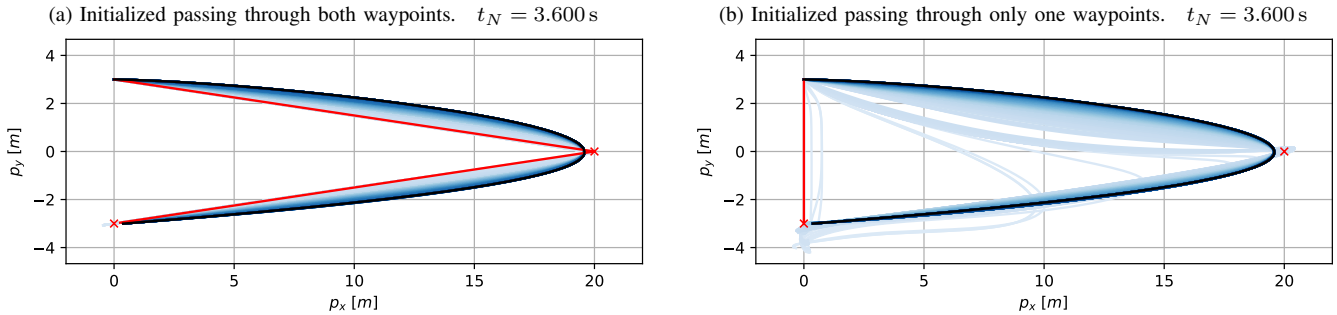

Fig. 7: Convergence in an open hairpin turn from two different initializations in red (—) over the iterations in light blue (—), to the final solution in dark blue(—). The trajectory starts at the top left and goes through the two waypoints (×). While the good initialization of Fig. 7a needs 216 iterations, the poor guess in Fig. 7b needs 303 iterations, but both converge to exactly the same solution with $t_N = 3.600\,\mathrm{s}$.

C. Initialization & Convergence

As a next step, the method is tested for convergence properties given different initializations. For this, we again use the STD quadrotor configuration and discretize the problem into $N = 160$ nodes with a tolerance of $d_{tol} = 0.4\,\mathrm{m}$. We use a track consisting of a so-called open hairpin, consisting of two waypoints as seen on the xy-plane in Fig. 7, starting on the top left and passing the waypoint to the far right and bottom left, where the end point is not in hover. Two setups are tested, where the first one is initialized with the position interpolated between the waypoints as in Fig. 7a, and the second one is initialized with a poor guess interpolated only from start to end point, as in Fig. 7b.

The expected outcome is that both initialization setups should converge to the same solution. Indeed we can observe this behavior in Fig. 7, where we depict the initial position guess in red and the convergence from light to dark blue. The good initial guess in Fig. 7a results in 216 iterations until convergence, while the poor guess in Fig. 7b needs 303 iterations, all plotted in Fig. 7. This indicates that our method is not sensitive to correct initializations, but profits from good guesses. However, since the acceleration space of a quadrotor is not necessarily convex and the resulting problem is highly non-convex, the next experiment elaborates on how to provoke and circumvent this issue.

D. Provoking Non-Convexity Issues

Since quadrotors can only produce thrust in body z-axis and the motors of most real-world systems cannot turn off, and therefore always produce a positive minimal thrust $T_{min} > 0$, the resulting acceleration space is non-convex. We evaluate this on a vertical turn where we fly from hover at the origin through two waypoints directly above each other, back to the origin but not in hover. The Track can be seen in Fig. 8. First, the STD quadrotor configuration is used, with $N = 150$ nodes and a tolerance of $d_{tol} = 0.1\,\mathrm{m}$, with the general initialization setup where the orientation is kept at identity. A second setup uses a different initialization, where we interpolate the orientation around the y-axis between $\alpha_{init} = 0$, $\alpha_0 = \pi$ for the second waypoint and $\alpha_1 = \alpha_2 = 2\pi$ for the remaining waypoints. The solutions and associated initializations are depicted in Fig. 8a, in which it is obvious that they do not converge to the same solution and the second setup actually performs a flip which is slightly faster. This is expected due to the non-convex properties of the problem.

However, a second set of experiments is performed with the same initialization setups but the RQ quadrotor configuration. This configuration has a minimum thrust of $T_{min} = 0\,\mathrm{N}$, which renders the achievable acceleration space convex. Both setups for the RQ configuration are depicted in Fig.8b. Indeed, both initializations now converge to the same solution, which overlay each other.

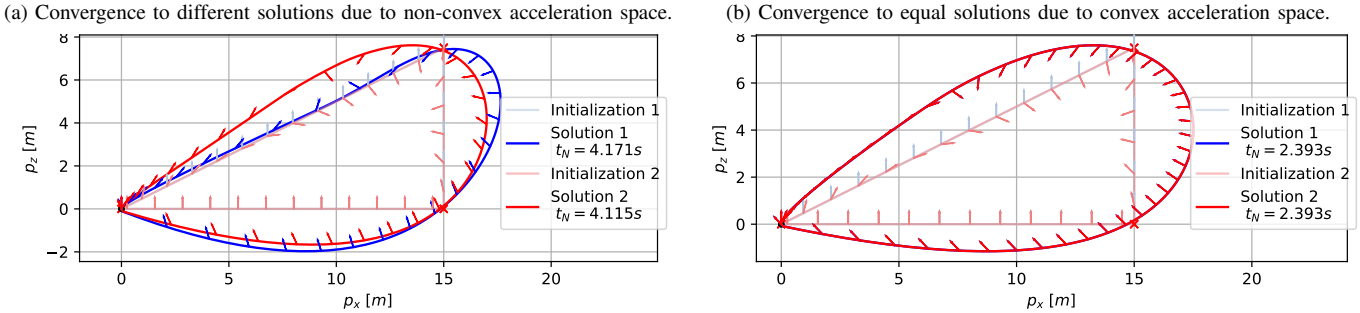

Fig. 8: A vertical turn flown starting at the origin in hover and passing through both waypoints from top to bottom, and back to the origin. Two quadrotor configurations are used (STD in Fig. 8a and RQ in Fig. 8b), with two different initialization setups each. The first setup is as described in VIII with identity orientation, while the second setup uses a linearly interpolated orientation guess. The arrows indicate the thrust direction of the quadrotor. The STD quadrotor configuration converges to two different solutions in Fig. 8a depending on the initialization due to its non-convex acceleration space with $T_{min} > 0\,\mathrm{N}$, while the RQ quadrotor configuration converges to equal (and overlaying) solutions in Fig. 8b, due to its convex acceleration space with $T_{min} = 0\,\mathrm{N}$.

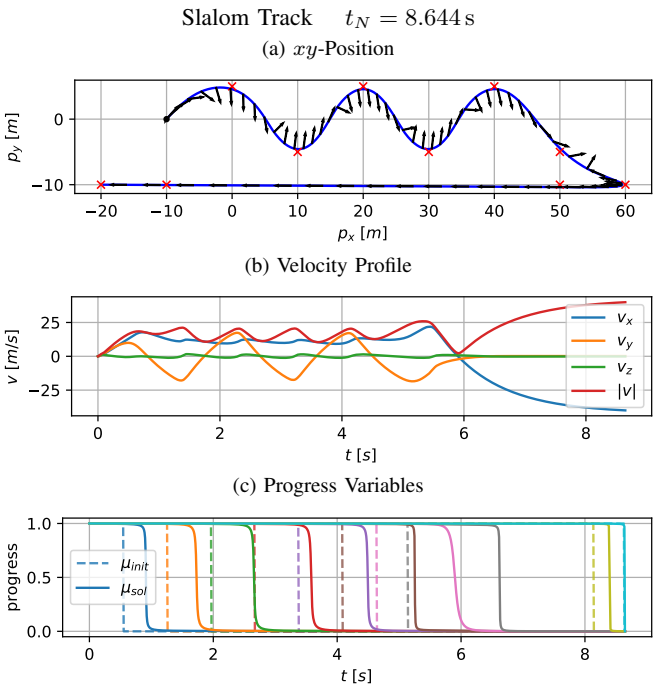

Fig. 9: A track consisting of a slalom, an open hairpin and a long stretch back, flown with the RQ quadrotor configuration in $t_N = 8.644\,\mathrm{s}$. In Fig. 9a the black arrows denote the acceleration direction. Note how the velocity approaches a steady state due to the linear aerodynamic drag on this quadrotor, and how the progress variables are being significantly shifted from their initial guess.

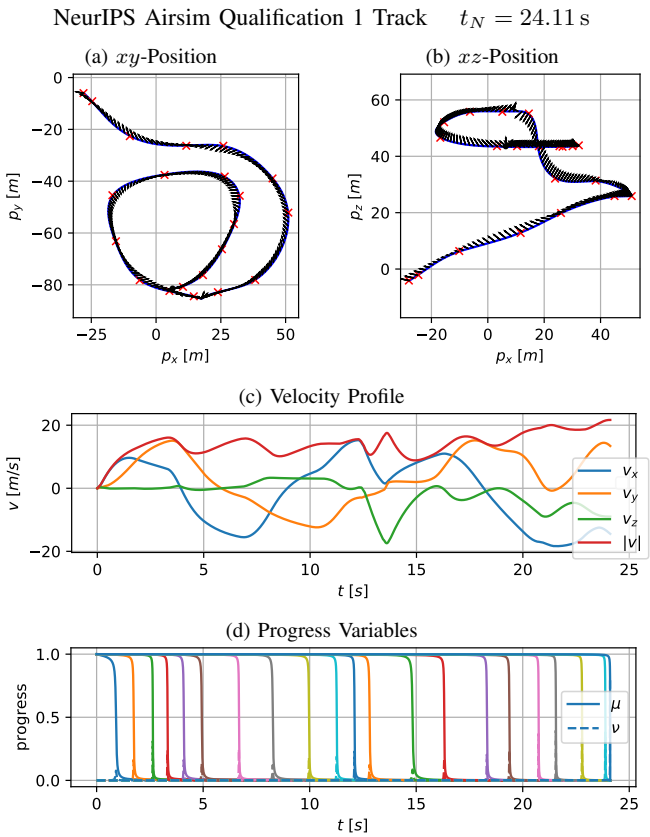

Fig. 10: The NeurIPS Airsim Qualification 1 track, covered in $t_N = 24.11\,\mathrm{s}$, as opposed to the best team's $30.11\,\mathrm{s}$. The top row depicts the trajectory in xy- and xy-plane, while the second and thrid row depict the velocity and progress respectively, plotted over time.

A simple solution would be to first solve the same problem using a linear and therefore convex point-mass model and using this to initialize the problem with the full quadrotor model. We further elaborate on the convexity property in the discussion Sec. IX-A.

E. Slalom

In a next step, a longer track is created, including a slalom and a long straight part, with 10 waypoints in total, as in Fig. 9. The trajectory is discretized into $N = 800$ nodes with a tolerance of $d_{tol} = 0.4\,\mathrm{m}$. The RC quadrotor configuration is deployed in this experiment. We expect the trajectory to smoothly slalom through the first 6 waypoints, take an sharp turn and accelerate back through the long straight segment. The straight segment is on purpose spanned by 4 unequally spaced waypoints, which together with the 6 previous waypoint provide a poor initial guess for the progress variables.

The resulting time-optimal trajectory is plotted in xy-plane in Fig. 9a, with the velocity in Fig. 9b and the progress variables in Fig. 9c. Note the dashed initial guess for the progress variables in Fig. 9c, which are significantly shifted to enable a time-optimal waypoint allocation. Furthermore, the velocity approaches its limit at the end of the straight segment, due to linear aerodynamic drag in this quadrotor configuration.

F. Microsoft AirSim, NIPS 2019 Qualification 1

As an additional demonstration, we apply our algorithm on a track from the 2019 NeurIPS AirSim Drone racing Challenge [33], specifically on the Qualifier Tier 1 setup. We chose a quadrotor with roughly the same properties, described as the MS configuration in Tab. I. The track is set up with the initial pose and 21 waypoints as defined in the environment provided in [33][1]. We use a discretization of $N = 3360$ nodes and a tolerance of $d_{tol} = 0.1\,\mathrm{m}$.

The original work by [33] also provides a simple and conservative baseline performance of $t_{total} \approx 110\,\mathrm{s}$ under maximal velocity and acceleration of $v_{max} = 30\,\mathrm{m\,s^{-1}}$ and $a_{max} = 15\,\mathrm{m\,s^{-1}}$, respectively. However, the best team achieved a time of $t_{total} = 30.11\,\mathrm{s}$ according to the evaluation page[2]. Our method generates a trajectory that passes all waypoints at a mere $t_N = 24.11\,\mathrm{s}$. Please note that this trajectory should only serve as a theoretical lower bound on the possibly achievable time given the model parameters. However, the model parameters were chosen slightly more conservative than the actual simulated drone for the qualification, to provide a reasonable optimal-time solution.

[1] https://github.com/microsoft/AirSim-NeurIPS2019-Drone-Racing, as of May 2020

[2] https://microsoft.github.io/AirSim-NeurIPS2019-Drone-Racing/leaderboard.html, as of May 2020

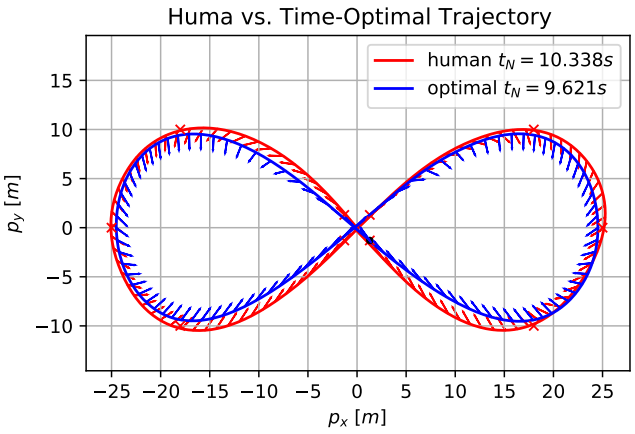

Fig. 11: Comparison of a human trajectory of $t_N = 10.338\,\mathrm{s}$ flown in simulation by an expert pilot and a time optimal trajectory of $t_N = 9.621\,\mathrm{s}$ generated by our proposed method. Note that the human pilot is commanding collective thrust and bodyrates, which are tracked by controller. This means that the human can not fully exploit the single rotor thrust and therefore the true actuation limit.

G. Qualitative Human-Comparison in Simulation

In this experiment, a comparison to human flown trajectories is established. As a short foreword, we want to point out that the human trajectories were collected in a experimental third-party simulation, focused on good flight characteristics for human pilots, but enabling logging of the drone state.

We setup the experiment as a figure-8-shaped track as depicted in Fig. 11. The used quadrotor model is the SIM configuration listed in Tab. I. While the mass, inertia, size, and drag constant is set as in the nominal model, the maximal thrust and bodyrates were fixed to the maximal values the human pilot was able to achieve with this platform in the simulation for a more fair comparison. (corresponding to the listed values in Tab. I). Therefore, the maximal bodyrate was set to $\omega_{max} = 3\,\mathrm{rad\,s^{-1}}$, and the maximal thrust to $T_{max} = 16\,\mathrm{N}$. We use $N = 1000$ discretization nodes and a tolerance of $d_{tol} = 0.4\,\mathrm{m}$.

The human had multiple tries for preparation and also at testing time > 10 consecutive rounds through the figure-8 track were collected, where one round starts and ends at the trajectory point passing closest to the origin, corresponding to the center intersection. The best lap with the lowest time was used and is depicted in Fig. 11, with a time of $10.338\,\mathrm{s}$. The generated time-optimal trajectory takes only $9.621\,\mathrm{s}$, exploiting the maximal available thrust at all times, except for when introducing and stopping quick rotations to realign the thrust through the gate passes.

Note that the human commands bodyrate and collective thrust, where the single rotor thrust control is left to a simulated low-level controller, as usual for human drone flight. This however imposes limits on how much of the input space the human can exploit, which is especially significant given the low thrust=to-weight ration of this drone configuration. Due to these limitations we also perform a real-world comparison of a high-speed maneuver in the next section.

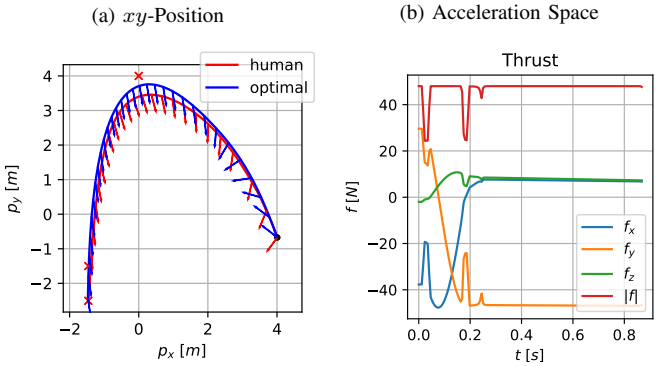

Fig. 12: Comparison of a human-flown real-world trajectory to a time-optimal trajectory, starting in the lower right corner (see Fig. 12a). The human-flown maneuver was recorded in a motion capture system and timed at $t_{human} = 0.984\,\mathrm{s}$ while the optimal trajectory reached a timing of $t_N = 0.874\,\mathrm{s}$. The optimal trajectory exploits the full acceleration of the quadrotor as long as possible (see Fig. 12b), and rotates into deceleration later than the human (see Fig. 12a).

H. Qualitative Human-Comparison in Real-World

As a last experiment, we provide a qualitative comparison of a time-optimal trajectory to a real human-flown trajectory recorded in a motion capture system. The trajectory represents a hairpin turn around a pole (see Fig. 12), which was picked because of its simplicity and repeatability for the human, and because it is a common element in drone racing where often the maximal available acceleration is exploited. the optimization is initialized at the point where the human pilot entered the trackable region and initial position and velocity of the vehicle are set equal to the human's, while the orientation and bodyrate is left open for optimization. We then add 3 waypoints forcing the drone to pass around the pole further away than the human, and leading the trajectory to exit at the same point and with roughly the same exit direction as the human. The problem is solved with $N = 150$ nodes and $d_{tol} = 0.4\,\mathrm{m}$ using the RC quadrotor configuration, modelled after the human's real quadrotor used for this experiment. As in the previous experiment, we restrict the maximal thrust to the same limit the human was able to exploit over a significant period of time as measured by the onboard IMU.

The maneuver is extremely quick and even the human trajectory only lasts for $0.984\,\mathrm{s}$, while the optimal solution reduces this to $0.874\,\mathrm{s}$. We can see that the time-optimal solution has a different acceleration direction at the beginning (see Fig. 12a), and starts to break later than the human trajectory, achieving a slightly better time. Additionally, we plot the the thrust (absolute and directional) in Fig. 12b and see the common behavior of the optimization exploiting the maximal thrust, only lowering for adjusting the rotational rates at the beginning.

Please note that, as before, the optimization uses the nominal model and is not aware of complex aerodynamic effects, noise or disturbances. It should serve as a theoretical ceiling or guidance for what could be achieved and as a demonstrator that our method does indeed work.

IX. Discussion

A. Convexity

While the problem of trajectory optimization quickly becomes non-convex when using complex and/or non-linear dynamic models, constraints, or even cost formulations (e.g. obstacle avoidance), it is often a valid approache to generate feasible (in terms of model dynamics) and near optimal trajectories. In our experiments, we have provoked and demonstrated one such non-convex property, and also quickly elaborate how one could support the optimization with an advanced initialization scheme to start close to the global optimum.

This can be achieved by reducing the non-linear quadrotor model into a linear point-mass model with bounded 3D acceleration input $\boldsymbol{u} = \boldsymbol{a}\ \forall \|\boldsymbol{a}\| \leq a_{max}$. This linear model renders the problem convex and allows to find a solution from which the original problem with the quadrotor model can be initialized, both in terms of translational trajectory, and also with an orientation guess based on the point-mass acceleration direction. While this initial guess is not yet a dynamically feasible trajectory due to the absent rotational dynamics, it serves as a good initial guess in close proximity to the optimal solution.

B. Optimality

Our approach does not provide optimality guarantees, since the problem by definition is non-convex and the optimal solution, while always guaranteed to be locally optimal, does not necessarily need to be globally optimal. Additionally, the used implementation framework [30] and solver backend [31] might influence the convergence behavior and the applicable guarantees. The previously described initialization scheme allows to mitigate these problems and, by running multiple setups (e.g. with different orientation guesses), one could cross check different solutions to find the optimal one with certainty (comparable to sampling methods).

C. Real-World Applicability

There are two problems hindering our approach from being applied in real world scenarios.

The first problem is posed by the nature of time-optimal trajectories themselves, as the true solution for a given platform is nearly always at the actuator constraints, and leaves no control authority. This means that even the smallest disturbance could potentially have fatal consequences for the drone and render the remainder of the trajectory unreachable. One has to define a margin lowering the actuator constraints used for the trajectory generation to add control authority and therefore robustness against disturbances. However, this also leads to a slower solution, which is no longer the platform-specific time-optimal one. In the context of a competition, this effectively becomes a risk-management problem with interesting connections to game theory.

Second, our method is computationally demanding, on a modern laptop taking many minutes ($\approx\ 1 - 40\mathrm{min}$) for scenarios as in Sec. VIII-A-VIII-D or sometimes even hours for larger scenarios such as VIII-E-VIII-G. However, this is highly implementation-dependent and could be vastly broken down to usable times, or precomputed for static race tracks and other non-dynamic environments.

X. Conclusion

In this work, we proposed a novel complementary progress constraint that allows to dynamically allocate waypoints to unknown nodes on an trajectory, and therefore find a time-optimal waypoint-flight solution without having to sample or allocate waypoint-completion timings. We validated our method with many experiments in a bottom-up fashion, checking against existing work, verifying the dynamic time allocation, provoking and elaborating on unwanted non-convex properties and initialization dependence. Finally, we demonstrated our method on longer tracks with more than $M \geq 10$ waypoints, and in a qualitative comparison against a human expert. We conclude that our method can be used to generate theoretical time-optimal trajectories at the actuation limit of the platform, which serve as an upper-bound of the achievable performance and guidance to develop fast and agile quadrotor flight.

XI. Acknowledgements

The authors want to thank Elia Kaufmann, Thomas Laengle, Christian Pfeiffer, and our professional pilot and collaborator Gabriel Kocher for their support and contribution throughout this work. This work was supported by the National Centre of Competence in Research Robotics (NCCR) through the Swiss National Science Foundation, the SNSF-ERC Starting Grant, and the EU H2020 Research and Innovation Program through the AERIAL-CORE project (H2020-2019-871479).

References

[1] H. Moon, J. Martinez-Carranza, T. Cieslewski, M. Faessler, D. Falanga, A. Simovic, D. Scaramuzza, S. Li, M. Ozo, C. De Wagter, G. de Croon, S. Hwang, S. Jung, H. Shim, H. Kim, M. Park, T.-C. Au, and S. J. Kim. Challenges and implemented technologies used in autonomous drone racing. 2019. URL https://link.springer.com/article/10.1007/s11370-018-00271-6.

[2] P. Foehn, D. Brescianini, E. Kaufmann, T. Cieslewski, M. Gehrig, M. Muglikar, and D. Scaramuzza. Alphapilot: Autonomous drone racing. *Robotics: Science and Systems (RSS)*, 2020. URL https://link.springer.com/article/10.1007/s11370-018-00271-6.

[3] S. M. LaValle. *Planning algorithms*. Cambridge university press, 2006. URL http://planning.cs.uiuc.edu.

[4] D. Mellinger, N. Michael, and V. Kumar. Trajectory generation and control for precise aggressive maneuvers with quadrotors. *Int. J. Robot. Research*, 2012. doi: 10.1177/0278364911434236.

[5] D. Mellinger and V. Kumar. Minimum snap trajectory generation and control for quadrotors. In *IEEE Int. Conf. Robot. Autom. (ICRA)*, 2011. doi: 10.1109/ICRA.2011.5980409.

[6] C. R. Hargraves and S. W. Paris. Direct trajectory optimization using nonlinear programming and collocation. *J. Guidance, Control, and Dynamics*, 1987. doi: 10.2514/3.20223.

[7] S. Bouabdallah and R. Siegwart. Full control of a quadrotor. In *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*, 2007. doi: 10.1109/IROS.2007.4399042.

[8] R. Mahony, V. Kumar, and P. Corke. Multirotor aerial vehicles: Modeling, estimation, and control of quadrotor. *IEEE Robot. Autom. Mag.*, 2012. doi: 10.1109/MRA.2012.2206474.

[9] M. Faessler, A. Franchi, and D. Scaramuzza. Differential flatness of quadrotor dynamics subject to rotor drag for accurate tracking of high-speed trajectories. *IEEE Robot. Autom. Lett.*, April 2018. doi: 10.1109/LRA.2017.2776353.

[10] T. Tomić, P. Lutz, K. Schmid, A. Mathers, and S. Haddadin. Simultaneous contact and aerodynamic force estimation (s-cafe) for aerial robots. *Int. J. Robot. Research*, 2020. doi: 10.1177/0278364920904788.

[11] M. Faessler, D. Falanga, and D. Scaramuzza. Thrust mixing, saturation, and body-rate control for accurate aggressive quadrotor flight. *IEEE Robot. Autom. Lett.*, April 2017. doi: 10.1109/LRA.2016.2640362.

[12] M. Bangura and R. Mahony. Real-time model predictive control for quadrotors. *IFAC World Congress*, 2014. doi: 10.3182/20140824-6-za-1003.00203.

[13] 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 *IEEE Int. Conf. Robot. Autom. (ICRA)*, 2016. doi: 10.1109/icra.2016.7487274.

[14] D. Falanga, P. Foehn, P. Lu, and D. Scaramuzza. PAMPC: Perception-aware model predictive control for quadrotors. In *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*, 2018.

[15] C. Richter, A. Bry, and N. Roy. *Polynomial Trajectory Planning for Aggressive Quadrotor Flight in Dense Indoor Environments*. 2016. doi: 10.1007/978-3-319-28872-7-37.

[16] M. W. Mueller, M. Hehn, and R. D'Andrea. A computationally efficient algorithm for state-to-state quadrocopter trajectory generation and feasibility verification. In *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*, 2013. doi: 10.1109/iros.2013.6696852.

[17] R. Allen and M. Pavone. A real-time framework for kinodynamic planning with application to quadrotor obstacle avoidance. In *AIAA Guidance, Navigation, and Control Conference*, 2016. doi: 10.2514/6.2016-1374.

[18] N. Ratliff, M. Zucker, J A. Bagnell, and S. Srinivasa. CHOMP: Gradient optimization techniques for efficient motion planning. In *IEEE Int. Conf. Robot. Autom. (ICRA)*, 2009. doi: 10.1109/ROBOT.2009.5152817.

[19] M. Posa, S. Kuindersma, and R. Tedrake. Optimization and stabilization of trajectories for constrained dynamical systems. In *IEEE Int. Conf. Robot. Autom. (ICRA)*, 2016. doi: 10.1109/ICRA.2016.7487270.

[20] P. Foehn, D. Falanga, N. Kuppuswamy, R. Tedrake, and D. Scaramuzza. Fast trajectory optimization for agile quadrotor maneuvers with a cable-suspended payload. In *Robotics: Science and Systems (RSS)*, 2017.

[21] M. Diehl, H. G. Bock, H. Diedam, and P. B. Wieber. Fast direct multiple shooting algorithms for optimal robot control. In *Fast motions in biomechanics and robotics*. Springer, 2006.

[22] M. Geisert and N. Mansard. Trajectory generation for quadrotor based systems using numerical optimal control. In *IEEE Int. Conf. Robot. Autom. (ICRA)*, 2016. doi: 10.1109/ICRA.2016.7487460.

[23] F. Augugliaro, A. P. Schoellig, and R. D'Andrea. Generation of collision-free trajectories for a quadrocopter fleet: A sequential convex programming approach. In *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*, 2012. doi: 10.1109/IROS.2012.6385823.

[24] S. Vakhrameev. A bang-bang theorem with a finite number of switchings for nonlinear smooth control systems. *J. Math. Sciences*, June 1997. doi: 10.1007/BF02355111.

[25] L. Poggiolini and G. Stefani. Minimum time optimality for a bang-singular arc: Second order sufficient conditions. In *IEEE Eur. Control Conf. (ECC)*, January 2006. doi: 10.1109/CDC.2005.1582360.

[26] M. Hehn, R. Ritz, and R. D'Andrea. Performance benchmarking of quadrotor systems using time-optimal control. *Auton. Robots*, March 2012. doi: 10.1007/s10514-012-9282-3.

[27] W. Van Loock, G. Pipeleers, and J. Swevers. Time-optimal quadrotor flight. In *IEEE Eur. Control Conf. (ECC)*, 2013. doi: 10.23919/ECC.2013.6669253.

[28] S. Spedicato and G. Notarstefano. Minimum-time trajectory generation for quadrotors in constrained environments. *IEEE Transactions on Control Systems Technology*, 2018. doi: 10.1109/TCST.2017.2709268.

[29] G. Ryou, E. Tal, and S. Karaman. Multi-fidelity black-box optimization for time-optimal quadrotor maneuvers. *arXiv prepring: arXiv:2006.02513*, June 2020. URL https://arxiv.org/abs/2006.02513.

[30] 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*, 2018.

[31] A. Wächter and L. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical programming*, 2006. doi: 10.1007/s10107-004-0559-y.

[32] S. Shah, D. Dey, C. Lovett, and A. Kapoor. AirSim: High-fidelity visual and physical simulation for autonomous vehicles. In *Field and Service Robot.*, pages 621–635, 2017.

[33] R. Madaan, N. Gyde, S. Vemprala, M. Brown, K. Nagami, T. Taubner, E. Cristofalo, D. Scaramuzza, M. Schwager, and A. Kapoor. Airsim drone racing lab. *PMLR post-proceedings of the NeurIPS 2019's Competition Track*, 2020.