Time-optimal planning for quadrotor waypoint flight

Foehn, Philipp ; Romero, Angel ; Scaramuzza, Davide

Abstract: Quadrotors are among the most agile flying robots. However, planning time-optimal trajectories at the actuation limit through multiple waypoints remains an open problem. This is crucial for applications such as inspection, delivery, search and rescue, and drone racing. Early works used polynomial trajectory formulations, which do not exploit the full actuator potential because of their inherent smoothness. Recent works resorted to numerical optimization but require waypoints to be allocated as costs or constraints at specific discrete times. However, this time allocation is a priori unknown and renders previous works incapable of producing truly time-optimal trajectories. To generate truly time-optimal trajectories, we propose a solution to the time allocation problem while exploiting the full quadrotor's actuator potential. We achieve this by introducing a formulation of progress along the trajectory, which enables the simultaneous optimization of the time allocation and the trajectory itself. We compare our method against related approaches and validate it in real-world flights in one of the world's largest motion-capture systems, where we outperform human expert drone pilots in a drone-racing task.

Time-Optimal Planning for Quadrotor Waypoint Flight

Philipp Foehn, Angel Romero, Davide Scaramuzza

Fig. 1: **A time-optimal trajectory.** This time-optimal flight path was computed using the proposed complementary progress constraints (CPC) and executed in a motion capture system, outperforming the best human expert.

Abstract—**Quadrotors are amongst the most agile flying robots. However, planning time-optimal trajectories at the actuation limit through multiple waypoints remains an open problem. This is crucial for applications such as inspection, delivery, search and rescue, and drone racing. Early works used polynomial trajectory formulations, which do not exploit the full actuator potential due to their inherent smoothness. Recent works resorted to numerical optimization, but require waypoints to be allocated as costs or constraints at specific discrete times. However, this time-allocation is a priori unknown and renders previous works incapable of producing truly time-optimal trajectories. To generate truly time-optimal trajectories, we propose a solution to the time allocation problem while exploiting the full quadrotor's actuator potential. We achieve this by introducing a formulation of progress along the trajectory, which enables the simultaneous optimization of the time-allocation and the trajectory itself. We compare our method against related approaches and validate it in real-world flights in one of the world's largest motion-capture systems, where we outperform human expert drone pilots in a drone-racing task.**

Multimedia Material

A video demonstrating the flight experiments can be found at https://youtu.be/ZPI8U1uSJUs.
Code: https://github.com/uzh-rpg/rpg_time_optimal

I. Introduction

Autonomous drones are nowadays used for inspection, delivery, cinematography, search-and-rescue, and entertainment such as drone racing [1]. The most prominent aerial system is the quadrotor, thanks to its simplicity and versatility, ranging from smooth maneuvers to extremely aggressive trajectories. This renders quadrotors amongst the most agile and maneuverable aerial robots[2, 3].

However, quadrotors have limited flight range, dictated by their battery capacity, which limits how much time can be spent on a specific task. If the task consists of visiting multiple waypoints (delivery, inspection, drone racing [4–6]), doing so in minimal time is often desired, and, in the context of search and rescue or drone racing (Fig. 1), even the ultimate goal. In fact, expert human drone racing pilots accomplish this with astonishing performance, guiding their quadrotors through race tracks at speeds so far unreached by any autonomous system. This begs the question of how close human pilots fly to the theoretical limit of a quadrotor, and whether planning algorithms could find and execute such theoretical optima.

For simple point-mass systems, time-optimal trajectories can be computed in closed-form, resulting in bang-bang acceleration trajectories [7], which can be sampled over multiple waypoints [8]. However, quadrotors are underactuated systems that need to rotate to adjust their actuated acceleration direction, which always lies in the body z-axis [9, 10]. 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.

A. Related Work

Two common approaches for planning quadrotor trajectories exist, continuous-time polynomials and discrete-time state space representations. The first option is the widely used polynomial formulation [10–12] exploiting the quadrotor's differentially-flat output states with high computational efficiency. However, these polynomials are inherently smooth and therefore cannot represent rapid state or input changes (e.g. bang-bang [7]) at reasonable order, and only reach the input limits for infinitesimal short durations, or constantly for the full trajectory time. This renders polynomials suboptimal since they cannot exploit the full actuator potential. Both problems are visualized and further explained in Section A, in the supplementary material.

The second option includes all approaches using time-discretized trajectories which can be found using search and sampling-based methods [13–16] or optimization-based methods [17–20]. However, sampling the 4-dimensional continuous input space over many discrete time steps with sufficient resolution quickly becomes computationally intractable, which is why prior work [13–16] restores to point-mass, polynomial, or differential-flatness approximations, and therefore does not handle single-rotor thrust constraints. Therefore, planning time-discretized trajectories with optimization-based methods is the only viable solution in the short-medium term. In such methods, 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. This time allocation is a priori undefined, since the time spent between any two waypoints is unknown, which renders tra-

ditional discretized state space formulations ineffective for time-optimal trajectory generation through multiple waypoints.

We investigate this problem and provide a solution that allows simultaneously optimizing the trajectory and waypoint allocation in a given sequence, exploiting the full actuator potential of a quadrotor. Our approach formulates a progress measure for each waypoint along the trajectory, indicating completion of a waypoint (see Fig. 2). We then introduce a "Complementary Progress Constraint" (CPC), that allows completion only in proximity to a waypoint. Intuitively, proximity and progress must complement each other, enforcing completion of all waypoints without specifying their time allocation.

There already exists a number of works towards time-optimal quadrotor flight [21–24], which, however, all suffer from severe limitations, such as limiting the collective thrust and bodyrates, rather than the actual constraint of limited single rotor thrusts.

The two earlier works [21, 22] are based on the aforementioned bang-bang approaches extended through numerical optimization of the switching times [21] and a trajectory representation using a convex combination of multiple analytical path functions [22]. However, both are restricted to 2-dimensional maneuvers, whereas our approach generalizes to arbitrary 3D waypoint sequences.

Another approach is taken in [23], where a change of variables along an analytic reference path is used to put the vehicle state space into a traverse-dynamics formulation. This allows using the arc length along the reference path as a progress measure and enables the formulation of costs and constraints independent of the time variable. However, as in the previous works, they simplify the platform limits to collective thrust and bodyrates, neglecting realistic actuator saturation. Furthermore, due to the use of Euler angles, their orientation space only covers a subset of the feasible attitudes and limits the solutions to a, possibly sub-optimal, subspace.

Finally, [24] uses a completely different approach, where the segment times of a polynomial trajectory are refined based on learning a Gaussian Classification model predicting feasibility. The classification is trained on analytic models, simulation, and real flight data. While emphasizing real-world applicability, this approach is still constrained to

polynomials and requires real-world data specifically collected for the given vehicle. Furthermore, it is an approximate method which only refines the execution speed of a predefined trajectory, rather than modifying the trajectory itself to a time-optimal solution, as opposed to our method.

B. Contribution

In contrast to existing methods, our approach resolves these problems by taking inspiration from optimization under contacts [25], proposing the formulation of *complementary progress constraints*, where we introduce a measure of progress and complement [26] it with waypoint proximity. More specifically, we formulate two factors that must complement each other, where, in our case, one factor is the completion of a waypoint (progress), while the other factor is the local proximity to a waypoint (Fig. 2). Intuitively, a waypoint can only be marked as completed when the quadrotor is within a certain tolerance of the waypoint (Fig. 3), allowing simultaneous optimization of the state and input trajectory, and the waypoint time allocation.

We demonstrate how our formulation can generate trajectories that are faster than human expert flights and evaluate it against two professional human drone racing pilots, outperforming them in terms of lap time and consistency on a 3D race track in a large-scale motion capture system (Fig. 1). Since our proposed optimization problem is highly non-convex, we also provoke non-convexity effects in simulation experiments in the supplementary material (Section B).

Our method can not only serve as a baseline for time-optimal quadrotor flight but might also find applications in other fields, such as (multi-) target interception, orbital maneuvers, avoiding mixed-integer formulations [27], and any problem where a sequence of task goals of unknown duration must be optimized under complex dynamic constraints.

II. METHODOLOGY

General Trajectory Optimization

The general optimization problem of finding the minimizer $\mathbf{x}^*$ for cost $L(\mathbf{x})$ in the state space $\mathbf{x} \in \mathbb{R}^n$ can be stated as

$$\mathbf{x}^* = \arg \min_{\mathbf{x}} L(\mathbf{x}) \qquad (1)$$
$$\text{subject to} \quad \mathbf{g}(\mathbf{x}) = 0 \quad \text{and} \quad \mathbf{h}(\mathbf{x}) \leq 0$$

where $\mathbf{g}(\mathbf{x})$ and $\mathbf{h}(\mathbf{x})$ contain all equality and inequality constraints respectively. The full state space $\mathbf{x}$ is used equivalently to the term *optimization variables*. The cost $L(\mathbf{x})$ 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 $\mathbf{x}_k$ is described at discrete times $t_k = dt \cdot k$ at $k \in [0, N]$, also called nodes, where its actuation inputs between two nodes are $\mathbf{u}_k$ at t_k with $k \in [0, N)$. The systems evolution is defined by the dynamics $\dot{\mathbf{x}} = \mathbf{f}_{dyn}(\mathbf{x}, \mathbf{u})$, anchored at $\mathbf{x}_0 = \mathbf{x}_{init}$, and implemented as an equality constraint of the 4th-order Runge-Kutta integration scheme (*RK4*):

$$\mathbf{x}_{k+1} - \mathbf{x}_k - dt \cdot \mathbf{f}_{RK4}(\mathbf{x}_k, \mathbf{u}_k) = 0 \qquad (2)$$

which is part of $\mathbf{g}(\mathbf{x}) = 0$ in the general formulation. Both $\mathbf{x}_k$, $\mathbf{u}_k$ are part of the state space and can be summarized as the vehicle's dynamic states $\mathbf{x}_{dyn,k}$ at node k. Note that this renders the problem formulation non-convex for non-linear system dynamics.

Time-Optimal Trajectory Optimization

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

A. Passing Waypoints through Optimization

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

$$L_{dist,j} = (\mathbf{p}_k - \mathbf{p}_{wj})^\top (\mathbf{p}_k - \mathbf{p}_{wj}) \qquad (3)$$

where $\mathbf{p}_k$, part of $\mathbf{x}$, 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
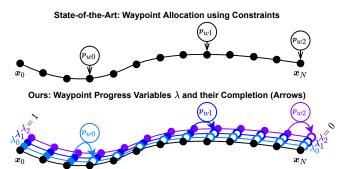
Fig. 2: **Progress variables.** Top: state-of-the-art fixed allocation of positional waypoints $\mathbf{p}_{wj}$ to specific nodes $\mathbf{x}_i$. Bottom: our method of defining one progress variable λ_j per waypoint. The progress variable can switch from 1 (incomplete) to 0 (completed) only when in the proximity of the relevant waypoint, implemented as a complementary constraint.
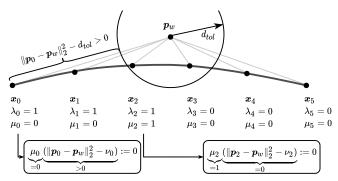

Fig. 3: **Complementary progress constraint.** The progress change μ can only be non-zero if the distance to the waypoint $\mathbf{p}_w$ is less than the tolerance d_{tol}. This is not the case for $\mathbf{x}_0$, but for $\mathbf{x}_1$, and allowing the progress variable to switch to 0 (complete).

guarantee to pass within a tolerance, constraint-based formulations can be used, such as

$$(\mathbf{p}_k - \mathbf{p}_{wj})^\top (\mathbf{p}_k - \mathbf{p}_{wj}) \le \tau_j^2 \qquad (4)$$

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

B. 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 $\boldsymbol{\lambda}_k \in \mathbb{R}^M$ 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

$$\boldsymbol{\lambda}_{k+1} = \boldsymbol{\lambda}_k - \boldsymbol{\mu}_k \qquad (5)$$

where the vector $\boldsymbol{\mu}_k \in \mathbb{R}^M$ indicates the progress change at every timestep. Note that the progress can only be positive, therefore $\mu_k^j \ge 0$. Both $\boldsymbol{\lambda}_k$ and $\boldsymbol{\mu}_k$ for every timestep are part of the optimization variables $\mathbf{x}$, which replicates the multiple shooting scheme for the progress variables. To define when and how the progress variables can change, we now imply a vector of constraints $\mathbf{f}_{prog}$ on $\boldsymbol{\mu}_k$, in its general form as

$$\boldsymbol{\epsilon}_k^- \le \mathbf{f}_{prog}(\mathbf{x}_k, \boldsymbol{\mu}_k) \le \boldsymbol{\epsilon}_k^+ \qquad (6)$$

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

$$\lambda_k^j \le \lambda_k^{j+1} \quad \forall \quad k \in [0, N], j \in [0, M]. \qquad (7)$$

Note that the last waypoint $\mathbf{p}_{wM}$ is always reached at the last node at t_N, and, therefore, could be implemented as a fixed positional constraint on $\mathbf{x}_N$, without loss of generality.

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, $\mathbf{f}_{prog}$ and $\boldsymbol{\epsilon}^- = \boldsymbol{\epsilon}^+ = 0$ are chosen to represent a *complementarity constraint* [25], as

$$f_{prog,j}(\mathbf{x}_k, \boldsymbol{\mu}_k) = \mu_k^j \cdot \|\mathbf{p}_k - \mathbf{p}_{wj}\|_2^2 := 0$$
$$\forall j \in [0, M] \qquad (8)$$

which can be interpreted as a mathematical *NAND (not and)* function, since either μ_k^j or $\|\mathbf{p}_k - \mathbf{p}_{wj}\|$ must be 0. Intuitively, the two elements *complement* each other.

C. Tolerance Relaxation

With (8) 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 (8) to

$$f_{prog,j}(\mathbf{x}_k, \boldsymbol{\mu}_k) = \mu_k^j \cdot \left(\|\mathbf{p}_k - \mathbf{p}_{wj}\|_2^2 - \nu_k^j \right) := 0$$
$$\text{subject to} \quad 0 \leq \nu_k^j \leq d_{tol}^2 \forall j \in [0, M] \quad (9)$$

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 maximum distance tolerance. This now enforces that the progress variables cannot change, except for the timesteps at which the system is within tolerance to the waypoint. Furthermore, please note that the spatial discretization δs depends on the number of nodes N and the speed profile. It should hold that $\delta s < d_{tol}$, to always allow at least one node to lie within the tolerance, as visualized in Fig. 3. This can be verified after the optimization and approximated beforehand by $\delta s \approx D/N$, where D is the cumulative distance between all waypoints.

D. Optimization Problem Summary

The full space of optimization variables $\mathbf{x}$ consists of the overall time and all variables assigned to nodes k as $\mathbf{x}_k$. All nodes k include the robot's dynamic state $\mathbf{x}_{dyn,k}$, its inputs $\mathbf{u}_k$, and all progress variables, $\mathbf{x} = [t_N, \mathbf{x}_0, \ldots, \mathbf{x}_N]$ where

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

Based on this representation, we write the full problem as

$$\mathbf{x}^* = \arg \min_{\mathbf{x}} t_N \quad (10)$$

subject to the system dynamics and initial constraint

$$\mathbf{x}_{k+1} - \mathbf{x}_k - dt \cdot \mathbf{f}_{RK4}(\mathbf{x}_k, \mathbf{u}_k) = 0 \quad \mathbf{x}_0 = \mathbf{x}_{init},$$

the input constraints

$$\mathbf{u}_{min} - \mathbf{u}_k \leq 0 \qquad \mathbf{u}_k - \mathbf{u}_{max} \leq 0, \quad (11)$$

the progress evolution, boundary, and sequence constraints

$$\boldsymbol{\lambda}_{k+1} - \boldsymbol{\lambda}_k + \boldsymbol{\mu}_k = \mathbf{0}$$
$$\boldsymbol{\lambda}_0 - 1 = \mathbf{0} \qquad \boldsymbol{\lambda}_N = \mathbf{0} \quad (12)$$
$$\boldsymbol{\mu}_k \geq 0 \quad \lambda_k^j - \lambda_k^{j+1} \leq 0 \quad \forall k \in [0, N), \quad j \in [0, M),$$

and the complementary progress constraint with tolerance

$$\mu_k^j \cdot \left(\|\mathbf{p}_k - \mathbf{p}_{wj}\|_2^2 - \nu_k^j \right) = 0 \quad (13)$$
$$-\nu_k^j \leq 0 \qquad \nu_k^j - d_{tol}^2 \leq 0. \quad (14)$$

Note that constraints (13, 14) are non-linear due to the norm on distance and the bilinearity of the progress change $\boldsymbol{\mu}$ and tolerance slack $\boldsymbol{\nu}$.

E. Quadrotor Dynamics

The quadrotor's state space is described between the inertial frame I and body frame B, as $\mathbf{x} = [\mathbf{p}_{IB}, \mathbf{q}_{IB}, \mathbf{v}_{IB}, \boldsymbol{\omega}_B]^\top$ corresponding to position $\mathbf{p}_{IB} \in \mathbb{R}^3$, unit quaternion rotation on the rotation group $\mathbf{q}_{IB} \in \mathbb{SO}(3)$ given $\|\mathbf{q}_{IB}\| = 1$, velocity $\mathbf{v}_{IB} \in \mathbb{R}^3$, and bodyrate $\boldsymbol{\omega}_B \in \mathbb{R}^3$. The input modality is on the level of collective thrust $\mathbf{T}_B = \begin{bmatrix} 0 & 0 & T_{Bz} \end{bmatrix}^\top$ 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

$$\dot{\mathbf{p}} = \mathbf{v} \qquad \dot{\mathbf{q}} = \frac{1}{2}\boldsymbol{\Lambda}(\mathbf{q})\begin{bmatrix} 0 \\ \boldsymbol{\omega} \end{bmatrix}$$
$$\dot{\mathbf{v}} = \mathbf{g} + \frac{1}{m}\mathbf{R}(\mathbf{q})\mathbf{T} \qquad \dot{\boldsymbol{\omega}} = \mathbf{J}^{-1}\left(\boldsymbol{\tau} - \boldsymbol{\omega} \times \mathbf{J}\boldsymbol{\omega} \right) \quad (15)$$

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

Quadrotor Inputs

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

$$\mathbf{T} = \begin{bmatrix} 0 \\ 0 \\ \sum T_i \end{bmatrix} \quad \text{and} \quad \boldsymbol{\tau} = \begin{bmatrix} l/\sqrt{2}(T_1 + T_2 - T_3 - T_4) \\ l/\sqrt{2}(-T_1 + T_2 + T_3 - T_4) \\ c_\tau(T_1 - T_2 + T_3 - T_4) \end{bmatrix} \quad (16)$$

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

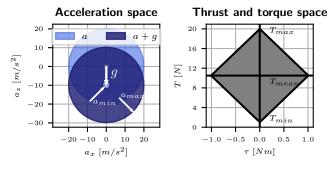$$0 \leq T_{min} \leq T_i \leq T_{max}. \quad (17)$$

Fig. 4: **Quadrotor input space.** Acceleration (left) and thrust/torque-space (right) of a standard quadrotor configuration. Note that the acceleration space is non-convex due to the minimum acceleration (the idle thrust at minimum motor speed) $a_{min} > 0$ being non-zero, and the thrust and torque limits are dependent on each other.

Fig. 2: **The quadrotor vehicle.** The autonomous platform used for the real-world experiments with a theoretical thrust-to-weight ration of ~ 4 at $0.8\,\mathrm{kg}$ weight, equipped with a Jetson TX2, a Laird communication module, off-the-shelf drone racing components, and infrared-reflective markers for motion capture.

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. 4 is non-convex due to $T_{min} > 0$ for the depicted model parameters from the standard configuration of Tab. I. The torque space is visualized in Fig. 4, where the coupling between the achievable thrust and torque is visible.

F. Approximative Linear Aerodynamic Drag

Finally, we extend the quadrotor's dynamics to include a linear drag model [28], to approximate the most dominant aerodynamic effects with diagonal matrix $\mathbf{D}$ by

$$\dot{\mathbf{v}} = \mathbf{g} + \frac{1}{m}\mathbf{R}(\mathbf{q})\mathbf{T} - \mathbf{R}(\mathbf{q})\mathbf{D}\mathbf{R}^\top(\mathbf{q}) \cdot \mathbf{v} \quad (18)$$

where we approximate $\mathbf{D} = \mathrm{diag}(d_x, d_y, d_z)$ in this work.

III. RESULTS

Video of the Results:
https://youtu.be/ZPI8U1uSJUs

We chose drone racing as a demonstrator for our method because in racing the ultimate goal is to fully exploit the actuator potential to accomplish a task in minimal time. In our experiment, we set up a human baseline on a 3D race track with 7 gates (Fig. 1 & 5) in a motion capture environment with two professional expert drone racing pilots. We plan a time-optimal trajectory through the same race track and use an in-house developed drone platform

and software stack to execute the trajectory in the same motion capture environment. We generate the trajectory at a slightly lower thrust bound than what the platform can deliver, to maintain controllability under disturbances as mentioned in Section A-D and discussed in Section IV-E. Our results show that we can outperform the humans and consistently beat their best lap time.

A. Experimental Drone Platform

The experiments are flown with an in-house developed drone platform based on off-the-shelf drone-racing components such as a carbon-fiber frame, BLDC motors, 5" propellers, and a BetaFlight flight controller. The quadrotor is equipped with an NVIDIA Jetson TX2 compute unit with WiFi and a Laird RM024 module for wireless low-latency communication. The static maximum thrust of a single rotor was measured using a load cell and verified in-flight, marking the platform's real maximum limit at a thrust to weight ratio of roughly ~ 4. Other specifications can be taken from Tab. I under the *Race Quad* configuration.

For state estimation (pose, linear and angular velocities), we use a VICON system with 36 cameras. For control, we deploy a Model Predictive Controller, similar to [29] but based on the quadrotor dynamics from (15–17), with the state space $\mathbf{x} = [\mathbf{p}_{IB}, \mathbf{q}_{IB}, \mathbf{v}_{IB}, \boldsymbol{\omega}_B]^\top$ and input space $\mathbf{u} = [T_1, T_2, T_3, T_4]$. The MPC operates over a horizon of $N_{MPC} = 20$ time steps of $\delta t = 0.05\,\mathrm{s}$, with a quadratic cost function, and also accounts for the single-rotor thrust constraints. The implementation is done using the ACADO[30] toolkit and

TABLE I: **Quadrotor Configurations**

Property	Race Quad	Airsim Quad	Standard Quad
m [kg]	0.8	1.0	1.0
l [m]	0.15	0.23	0.15
$diag(J)$ [gm^2]	$[1, 1, 1.7]$	$[10, 10, 20]$	$[5, 5, 10]$
$[T_{min}, T_{max}]$ [N]	$[0.0, 8.0]$	$[0.0, 4.179]$	$[0.25, 5.0]$
c_τ [1]	0.01	0.0133	0.01
ω_{max} [rad s^{-1}]	15	10	10
d_{drag} [s^{-1}]	0.4	0.6	–

qpOASES[31] as solver. We execute the MPC in a real-time iteration scheme [32] at a feedback rate of $100\,$Hz. The low-level BetaFlight controller has access to high-frequency IMU measurements, which allows precise tracking of bodyrate and collective thrust commands. These commands are extracted from the MPC controller and are guaranteed to stay within the platform capabilities due to the single-rotor thrust constraints.

B. Human Expert Pilot Baseline

Since human pilots so far outperformed autonomous vehicles, we establish a baseline by inviting two professional expert drone racing pilots, Michael Isler and Timothy Trowbridge, both of which compete in professional drone racing competitions. A list of their participation and rankings can be found in Tab. IV in the supplementary material. We created a 3D racing track in a VICON motion capture environment spanning roughly $25 \times 30 \times 8\mathrm{m}$ and let the humans train on this track for hours. We captured multiple races, each consisting of multiple laps, from which we evaluated the one with the overall best lap time, according to our timing strategy described in Section III-D. The quadrotor platform used for human flights has the same thrust-to-weight ratio as the autonomous platform. This provides a fair baseline since (I) both the humans and the autonomous drone have the same limitations, (II) the humans are given enough training time to adapt to the track, as is the case at a real drone racing event, and (III) we compare against the race including the absolute best lap time from all runs. Especially the latter point gives a substantial advantage to the human competitors, since they are typically not capable of reproducing the absolute best lap time reliably, and would therefore fall behind in any multi-lap evaluation. The evaluated best human runs each contain 7 laps.

The human platform is also tracked in the VICON system and resembles the autonomous drone described in Section III-A, but drops the Jetson and Laird modules for a remote control receiver and a first-person-view camera system, with no weight difference. To keep a fair baseline, the human platform is restricted to the same maximum thrust-to-weight limit as the autonomous drone. The track, time-optimal reference (thrust-to-weight ratio 3.3), and the human expert 1 trajectory are visualized in Fig. 5, with their speed and acceleration profile colorized.

C. Trajectory Generation

To generate the time-optimal trajectories that will be executed by our platform, we set the gate positions of the track shown in Fig. 5 as waypoint constraints. We generate 2.5 laps to ensure that our experiments are not affected by start and end transient effects or large drops in battery voltage, and to ensure at least two full laps at maximum speed. For optimal results, the laps are generated by concatenating the waypoints for a single lap multiple times and solving the full multi-lap problem at once. Therefore, the optimization passes through 18 waypoints and can be solved on a normal desktop computer in $\sim 40\,$min. We used $N = 720$ nodes with a tolerance of $d_{tol} = 0.3\,$m.

Because a time-optimal trajectory exploits the full actuator potential, it is extremely aggressive from start to end. To ensure safe execution on a real drone, we linearly ramp up the thrust limit for the trajectory generation from hover to the full thrust limit, guaranteeing a smooth start of the trajectory. Since we plan over 2.5 laps and exclude the start and end from the timing, this has no notable impact on the reported timing results. The same start and end exclusion are done for the human timings (Section III-D).

Additionally, we plan the trajectory for a multitude of thrust-to-weight ratios reaching from 2.5 with a lap time of $7.14\,$s to a maximum of 3.6 resulting in a lap time of $5.81\,$s. As expected, the time shrinks with higher thrust-to-weight capabilities. We evaluate two configurations with thrust-to-weight ratios of 3.15 ($6.27\,$s) and 3.3 ($6.10\,$s) in our real-world experiments, reported in the following section. Both of these configurations are consistently faster than the human trajectory while staying within a safe margin of the quadrotors absolute limit (TWR: ~ 4), which allows for robust control even under disturbances, noise, and model imperfections.
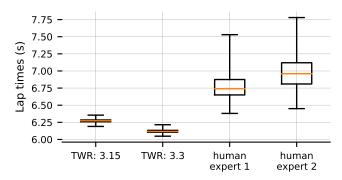
Fig. 3: **Timing analysis.** Lap time box plot of our time-optimal trajectory executed in real-world for two configurations (two laps), and two human races (seven laps). Note that even with a 5% reduced thrust-to-weight ratio of 3.15, the proposed time-optimal trajectories are faster than the best human lap time, with substantially lower variance.
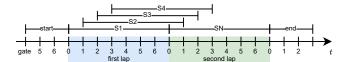

Fig. 4: **Visualization of the timing strategy.** A race includes two or more laps, over each of which we distribute 40 points at which we take timings of the resulting segments $S1, \ldots, SN$. This allows for statistically meaningful timing extraction.

TABLE II: **Timing Statistics**

Timing	human 1	human 2	ours TWR: 3.15	ours TWR: 3.3
mean [s]	6.794	6.987	3.15	6.120
median [s]	6.740	6.960	6.272	6.116
min [s]	6.389	6.450	6.190	6.048
max [s]	7.530	7.780	6.354	6.215
std [s]	0.2556	0.2859	0.0280	0.0278

D. Timing Analysis

First of all, our real-world experiments should provide a proof-of-concept that time-optimal trajectories planned below the drone's actual thrust limit are a feasible and viable solution. We refer the reader to the accompanying Movie at https://youtu.be/ZPI8U1uSJUs.

Second, we point out that it is possible to generate and execute trajectories that can outperform the human baseline. For this we provide a statistical lap-time analysis in Fig. 3, indicating the superior performance on both configurations.

To compute reliable lap times, we first define a full lap as each lap that is not affected by any start (take-off) or end (landing) segment. We can then pick a set of timing points $\mathcal{S}_{pt} = \{\mathbf{p}_{t0}, \mathbf{p}_{t1}, \ldots\}$ along the trajectory, and define the timing as the time needed to visit one such point twice, i.e. the time of a segment starting and ending at the same point. This measure allows us to extract a statistically valid timing of a single closed lap, which does not depend on the location of timing start and stop.

Effectively we time the best laps of the human flights, and two full laps of both our race trajectories, each flown twice. Fig. 3 shows the obtained timing results, where it is clearly visible that the autonomous drone outperforms the human pilots both in absolute time, but also consistency. The latter is expected since once the trajectory is generated, it can be repeated multiple times without

variation.

IV. DISCUSSION

A. Velocity and Acceleration Distribution

Furthermore, we evaluate the velocity and acceleration distribution over the different human and time-optimal flights. We depict the trajectories colored by their speed and acceleration profile in Fig. 5 with the time-optimal reference with TWR: 3.3. Inspecting the hotspots of darker colors in the velocity plots (Fig. 5, top row) indicating higher speeds, we can see that the velocity distribution is rather similar for the autonomous time-optimal and the human trajectory. However, comparing the acceleration of the human and the time-optimal trajectory (Fig. 5, bottom row), we observe that the acceleration of the human varies considerably more, and often dips to lower values than the time-optimal ones. This corresponds to sections where the human pilots do not use the full actuation spectrum of the platform and lose substantial performance over the time-optimal trajectory. This is especially visible in the right-most section of the track, where the flight path has relatively low curvature. The time-optimal trajectory exploits the full acceleration capabilities, whereas the human notably reduces the acceleration between the right-most gates, leading to lower speeds in the following bottom section of the track.

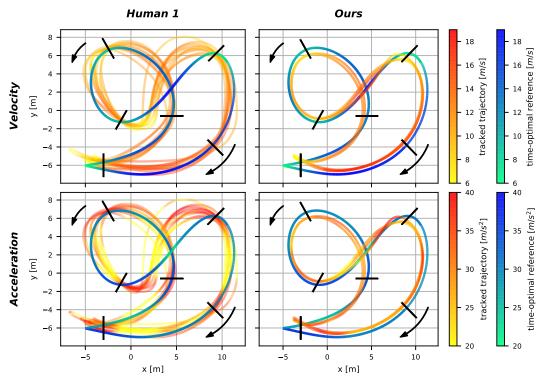Furthermore, we can identify the high-speed region in the sections of low curvature, where both

Fig. 5: **Comparison against a human pilot.** The race track with seven gates visualized with the yellow-red real-world trajectories of the humans (left, all seven laps of the race including the overall best lap time) and the autonomous drone (right, two laps visualized), and the green-blue time-optimal reference trajectory with a thrust-to-weight ratio (TWR) of 3.3. The trajectories are colored by their speed profile (top row) and acceleration profile (bottom row) to indicate the hotspots of highest velocity and acceleration along the track. The two black arrows indicate the direction of flight. The human pilots vary their acceleration substantially more than the autonomous drone and spend more time at sub-optimal acceleration (yellow coldspots on the lower left figure). Note that the gate in the lower-left corner consists of two gates stacked vertically.

human and autonomous platforms spend more acceleration in the velocity direction (accelerating and braking), than perpendicular to the velocity (direction change). Although the platforms have equal TWR, and our time-optimal trajectory is planned with a substantial margin to the platform's TWR limit, it exceeds the human speed profile.

B. Human Performance Comparison

From our findings in Section III-D and IV-A, we conclude that human pilots are not far away from our time-optimal trajectory, since they reach similar but slower velocity distribution and lap times. However, humans struggle to consistently exploit the full actuation spectrum of the vehicle, resulting in suboptimal performance compared to our approach. One possible reason for this can be found in [33], where we analyzed eye gaze fixations of human pilots during racing and found that they fixate their eyes on upcoming gates well before passing the next gate, indicating that humans use a receding planning horizon, whereas our approach optimizes the full trajectory at once.

C. Tracking Error Considerations

We want to point out that while we achieved successful deployment on a real quadrotor system, we experience substantial tracking errors in doing so, as visible in Fig. 5. While this work is not about improving the tracking performance, but should rather serve as a feasibility study given our method, we still feel responsible for pointing out the encountered difficulties. A number of effects lead to this tracking error of $\sim 0.7\,\mathrm{m}$ positional RMSE:

(I) We did not account for any latency correction of the whole pipeline, including motion capture and pose filtering, data transmission to the drone, MPC execution time, and communication to the flight controller.

(II) Our simple linear-drag aerodynamic model was verified in [28] at speeds up to $5\,\mathrm{m\,s^{-1}}$. However, in the vastly higher speed regimes we reach

during our real-world experiments, the model seems to be inaccurate, especially in terms of drag at higher speeds and in body z direction. The effect of this is visible in Fig. 5, where there is a disturbance towards the inside of the curvature for most of the time.

(III) The used BetaFlight controller is a reliable system for human drone pilots. However, as such, it includes many filtering, feed-forward, and control strategies that are tuned for a consistent human *flight-feeling*. Unfortunately, this does not translate to accurate closed-loop control or desirable control-loop shaping and causes more harm than good when used with closed-loop high-level controllers such as our MPC. Not only is it not possible to command the single rotor thrusts from the MPC to BetaFlight, but the provided bodyrate tracking also performed poorly. This, however, could be solved with a custom flight controller and might be part of further studies.

D. Convexity and Optimality

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 approach to generate feasible (in terms of model dynamics) trajectories. In fact, given a non-convex problem, solution schemes such as the used interior point method can only guarantee local optimality and global feasibility, but not global optimality. Within our approach, we can summarize the following non-convex properties:

- The quadrotor dynamics (15) are non-linear and therefore non-convex in the context of (2).
- The acceleration space of a quadrotor (17) is non-convex given non-zero minimal thrust $T_{min} > 0.0$.
- Eqn. (13) is non-convex due to the norm on distance and the bilinearity of the progress change $\boldsymbol{\mu}$ and tolerance slack $\boldsymbol{\nu}$.

In our experiments in Sections B-E and B-F, we provoke such non-convex properties, and explain how the optimization can be supported 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 $\mathbf{u} = \mathbf{a}$ where $\|\mathbf{a}\| \leq a_{max}$. The linear model removes the prominent non-convex dynamics and allows us 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 a non-continuous orientation guess based on the point-mass acceleration direction. While this initial guess is not yet dynamically feasible for quadrotors due to the absent rotational dynamics, it serves as a valid initial guess in the convex region of the global-optimal translation space. Finally, the rotational non-convexity can be resolved by solving multiple problems of different initializations, as demonstrated in Section B-E.

Last but not least, the reader should note that even in the case of a local (but not global) optimal solution, the vehicle dynamics are satisfied and the trajectory is dynamically feasible.

E. Real-World Deployment

There are three challenges when deploying our approach 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 damaging 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, ranging from a few minutes ($< 20\,\text{min}$) for scenarios as in Sections B-B and B-E towards an hour or more for larger scenarios such as Section III with $\sim 40\,\text{min}$ and Section B-G with $\sim 65\,\text{min}$ on a normal desktop computer. 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. Furthermore, our approach provides a method for finding the theoretical upper bound on performance, as a benchmark for other methods.

Third, we use a motion capture system to deploy our method. However, in real-world scenarios, such high-performance off-board localization systems are barely ever available. This necessitates the deployment using on-board state-estimation systems, such as visual-inertial odometry. Unfortunately, these systems can suffer from high motion blur in such fast flight scenarios, and therefore need substantial further research and development to be of sufficient robustness for the purpose of time-optimal flight [5, 4]. Despite those difficulties, we have demonstrated that our method *can* be deployed and *is* in fact substantially faster than human experts.

V. Acknowledgments

We thank the two professional human pilots, Michael Isler and Timothy Trowbridge, for their participation. Without their incredible skills, establishing a fair human baseline would not have been possible.

Ethics Approval: The study protocol has been approved by the local ethics committee of the University of Zurich. The participants gave their written informed consent before participating in the study and gave their written consent to be mentioned by name and listed with their competition participation record.

Data and Materials: All materials are described within this paper and completed by the dataset and code necessary to reproduce our results can be found in [34] at

https://doi.org/10.5061/dryad.9kd51c5h7. A video of the flight performance can be found at https://youtu.be/ZPI8U1uSJUs.

References

[1] G. Loianno and D. Scaramuzza. Special issue on future challenges and opportunities in vision-based drone navigation. *J. Field Robot.*, 2020. doi: 10.1002/rob.21962.

[2] Evan Ackermann. Ai-powered drone learns extreme acrobatics. https://spectrum.ieee.org/automaton/robotics/drones/ai-powered-drone-extreme-acrobatics, 2020. [Online; accessed 21.6.2021].

[3] Jon Verbeke and Joris De Schutter. Experimental maneuverability and agility quantification for rotary unmanned aerial vehicle. *International Journal of Micro Air Vehicles*, 2018. doi: 10.1177/1756829317736204.

[4] 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. *J. Intell. Service Robot.*, 2019. doi: 10.1007/s11370-018-00271-6.

[5] 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.

[6] Antonio Loquercio, Elia Kaufmann, René Ranftl, Alexey Dosovitskiy, Vladlen Koltun, and Davide Scaramuzza. Deep drone racing: From simulation to reality with domain randomization. *IEEE Transactions on Robotics*, 2019. doi: 10.1109/TRO.2019.2942989.

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

[8] Lev Semenovich Pontryagin, EF Mishchenko, VG Boltyanskii, and RV Gamkrelidze. *The mathematical theory of optimal processes*. Wiley, 1962.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] D. J. Webb and J. van den Berg. Kinodynamic RRT*: Asymptotically optimal motion planning for robots with linear dynamics. In *IEEE Int. Conf. Robot. Autom. (ICRA)*, 2013. doi: 10.1109/ICRA.2013.6631299.

[14] 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.

[15] S. Liu, K. Mohta, N. Atanasov, and V. Kumar. Search-based motion planning for aggressive flight in se(3). *IEEE Robot. Autom. Lett.*, 2018. doi: 10.1109/LRA.2018.2795654.

[16] B. Zhou, F. Gao, L. Wang, C. Liu, and S. Shen. Robust and efficient quadrotor trajectory generation for fast autonomous flight. *IEEE Robot. Autom. Lett.*, 2019. doi: 10.1109/LRA.2019.2927938.

[17] 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.

[18] 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.

[19] 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.

[20] B. Penin, R. Spica, P. Robuffo Giordano, and F. Chaumette. Vision-based minimum-time trajectory generation for a quadrotor uav. In *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*, 2017. doi: 10.1109/IROS.2017.8206522.

[21] 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.

[22] 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.

[23] 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.

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

[25] Michael Posa, Cecilia Cantu, and Russ Tedrake. A direct method for trajectory optimization of rigid bodies through contact. *Int. J. Robot. Research*, 2014. doi: 10.1177/0278364913506757.

[26] Richard W. Cottle and George B. Dantzig. Complementary pivot theory of mathematical programming. *Linear Algebra and its Applications*, 1968. doi: 10.1016/0024-3795(68)90052-9.

[27] A. Richards and J. P. How. Aircraft trajectory planning with collision avoidance using mixed integer linear programming. In *IEEE Am. Control Conf. (ACC)*, 2002. doi: 10.1109/ACC.2002.1023918.

[28] 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.

[29] 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.

[30] B. Houska, H.J. Ferreau, and M. Diehl. ACADO toolkit – an open source framework for automatic control and dynamic optimization. *Optimal Control Applications and Methods*, 2011. doi: 10.1002/oca.939.

[31] Joachim Ferreau, Christian Kirches, Andreas Potschka, Hans Bock, and Moritz Diehl. qpoases: A parametric active-set algorithm for quadratic programming. *Mathematical Programming Computation*, 12 2014. doi: 10.1007/s12532-014-0071-1.

[32] 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.

[33] Christian Pfeiffer and Davide Scaramuzza. Human-piloted drone racing: Visual processing and control. *IEEE Robot. Autom. Lett.*, 2021. doi: 10.1109/LRA.2021.3064282.

[34] Philipp Foehn, Angel Romero, and Davide Scaramuzza. Time-optimal planning for quadrotor waypoint flight: Dataset. https://doi.org/10.5061/dryad.9kd51c5h7, 2021.

[35] 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.

[36] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl. CasADi – A software framework for non-linear optimization and optimal control. *Mathematical Programming Computation*, 2018.

[37] 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.

[38] 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.

[39] 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.

APPENDIX A
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 $\mathbb{R}^3$ controlled by bounded acceleration $\mathbf{a} \in \mathbb{R}^3 \mid \|\mathbf{a}\| \leq a_{max}$, starting at rest position $\mathbf{p}_0$ and translating to rest position $\mathbf{p}_1$. 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 or sampling over the switching times and intermittent waypoint velocities. For the general solution and applications, we refer to [8] and [7].

B. Bang-Bang Relation for Quadrotors

A quadrotor would exploit the same maximal acceleration by generating the maximum thrust. However, due to the quadrotor's underactuation, it cannot instantaneously change the acceleration direction but needs to rotate by applying differential thrust over its rotors. As a result, the linear and rotational acceleration, both controlled through the limited rotor thrusts, are coupled (visualized later in Fig. 4). Therefore, a time-optimal trajectory is the *optimal tradeoff between rotational and linear acceleration*, where the thrusts only deviate from the maximum to adjust the rotational rates. Indeed, our experiments confirm exactly this behavior (see e.g. Fig. 8).

C. Sub-Optimality of Polynomial Trajectories

The quadrotor is a differentially flat system [11] that can be described based on its four flat output states, position, and yaw. This allows representing 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 the start and end time, and minimizes for one of the derivatives, commonly the jerk or snap (3rd/4th derivative of position, as in [12, 35]). The intention behind such 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 very efficient to compute (especially [12]), 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 Section A-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, as visualized in Fig. 6.

D. Real-World Restrictions

While the time-optimal solutions are at the boundary of the reachable input space and cannot be tracked robustly [7], they represent an upper bound on the performance for a given track, intended as a baseline for other algorithms. However, we can lower the input bounds (w.r.t the real quadrotor actuator limit) for the purpose of trajectory generation, allowing for control authority and therefore restoring a margin for robustness, rendering the planned trajectories trackable in the real world. As an application example, we present a demonstration of our method, where we track a time-optimal trajectory with a real quadrotor in a motion capture system and consistently outperform the human expert baseline.
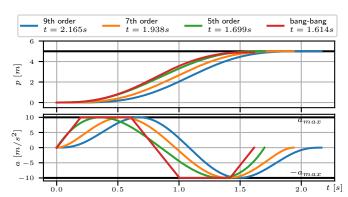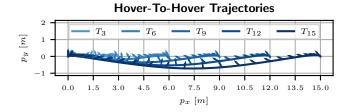

Fig. 6: **Bang-bang and polynomial trajectories.** Multiple orders of polynomial trajectories and one bang-bang trajectory with limited slope. The polynomial trajectories only touch the input extrema in two points, while the bang-bang spends more time at the limit and achieves a lower overall time.
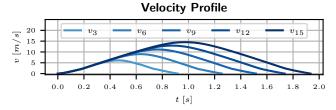
Fig. 7: **Baseline comparison.** Time-optimal hover-to-hover trajectories between states spaced $p_x = [3, 6, 9, 12, 15]$m apart as in [21, 22]. The top figure depicts the position on the xz-plane, while the bottom figure depicts the velocity profile.

p_x vs Time	Ref. [21]	Ref. [22]	CPC-RT	CPC (ours)
3 m	0.898 s	0.890 s	0.891 s	0.918 s
6 m	1.231 s	1.223 s	1.227 s	1.255 s
9 m	1.488 s	1.478 s	1.484 s	1.517 s
12 m	1.705 s	1.694 s	1.702 s	1.736 s
15 m	1.895 s	1.885 s	1.894 s	1.933 s

TABLE III: Comparison of the resulting timings between our approach and [21, 22]. Note that our approach is 2.00% slower than [21] and 2.68% slower than [22], because it accounts for rotation dynamics and single rotor limits. Our approach applied to the dynamics and limits of [21, 22], closely reproduces their results (column *CPC-RT*).

APPENDIX B
SIMULATION EXPERIMENTS

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 [21, 22] in Section B-B. Next, we investigate the time alignment for multiple waypoints in Section B-C, followed by an experimental investigation of the convergence characteristics on short tracks in terms of initialization in Section B-D and (non-) convexity in Section B-E. Finally, we demonstrate applicability to longer tracks with ≥ 10 waypoints (Section B-G). Note that this section will focus on the trajectory generation, while Section III will demonstrate real-world deployment.

All evaluations are performed using CasADI [36] with IPOPT [37] 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 [38] SimpleFlight quadrotor, and

the third standard configuration resembles the one used in [21, 22].

A. 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 $k_j = N \cdot j/M$. 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 $N_w \gg 100$ help with convergence and achieved stability of the underlying optimization algorithm.

B. 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 $p_x = [3, 6, 9, 12, 15]$m, as in [21, 22]. Additionally to the problem setup explained in Section II, we add constraints to the end state to be in hover, i.e. $\mathbf{v}_N = \mathbf{0}$ and $\mathbf{q} = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix}$. We use $N = N_w = 300$ nodes and a tolerance of $d_{tol} = 10^{-3}$. Different from [21, 22], we compute the solution in full 3D space, which however does not matter for this experiment, since the optimal trajectory stays within the y-plane. We defined the model properties so that it meets the maximal and minimal acceleration $[a_{min}, a_{max}] = [1, 20]\mathrm{m\,s^{-2}}$ and maximal bodyrate $\omega_{max} = 10\,\mathrm{rad\,s^{-1}}$ as in [21, 22], given by our standard quadrotor configuration (see Tab. I).

The solutions are depicted in the xy-plane in Fig. 7 and the timings are stated and compared to [21, 22] in Tab. III. In addition to our proposed single-rotor-thrust model 15 (reported as *ours* in tab. III), we also compute the timings for a simplified model with only collective thrust and bodyrate constraints, but no single-rotor-thrust modeling, as used in [21, 22] (reported as *CPC-RT* in Tab/ III). Note that our approach with the simplified model
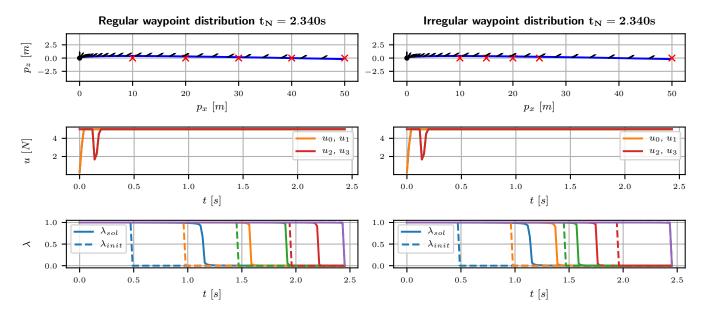
Fig. 8: **Time allocation.** A trajectory along waypoints distributed on a line over $50\,\text{m}$, flown in $2.430\,\text{s}$. On the left side, the waypoints are equally distributed over the total distance, while on the right side the first 4 out of 5 waypoints are located within the first half of 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 substantially adjusted from the initialization, and is different between the settings.

produces results very similar to [21, 22], but with the increased model-fidelity, our approach is 2.00% slower than [21] and 2.68% slower than [22], due to the added rotational dynamics. However, [21, 22] does not allow to compute trajectories through multiple waypoints and [23] employs the same unrealistic bodyrate and thrust limits. In contrast, our method accounts for both, realistic single rotor thrust limits and multiple waypoints, while simultaneously solving the time-allocation, as evaluated in the next section.

C. 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 our method can choose the optimal time at which a waypoint is passed. Therefore, we test two different distributions of the waypoints $\mathbf{p}_{wj}$ over the straight track; specifically, we define a regular ($p_{x,reg} = [1, 20, 30, 40, 50]$m), and an irregular ($p_{x,ireg} = [10, 15, 20, 25, 50]$m) distribution. We chose $N = 125$ and a tolerance of $d_{tol} = 0.4\,\text{m}$, with the standard quadrotor (see Table I).

As expected, both waypoint distributions converge to the same solution of $t_N = 2.430\,\text{s}$, depicted in Fig. 8, with equal state and input trajectories,

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. 8, while the trajectory time, dynamic states, and inputs stay the same.

D. Initialization & Convergence

As a next step, the method is tested for convergence properties given different initializations. For this, we again use the standard quadrotor configuration and discretize the problem into $N = 160$ nodes with a tolerance of $d_{tol} = 0.4\,\text{m}$. We use a track consisting of a so-called open hairpin, consisting of two waypoints as seen on the xy-plane in Fig. 9, starting on the top left and passing the waypoint to the far right and bottom left, where the endpoint is not in hover. Multiple setups are tested, where the first one is initialized with the position interpolated between the waypoints (Fig. 9, left), the second one is initialized with a poor guess interpolated only from start to endpoint (Fig. 9, middle), and the third one includes 100 random initializations (Fig. 9, right).

The expected outcome is that all initialization setups should converge to the same solution. Indeed we can observe this behavior in Fig. 9, where we
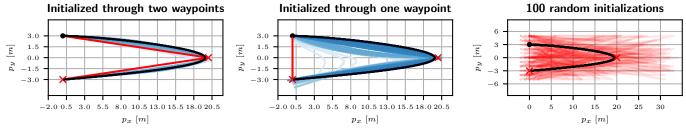
Fig. 9: **Convergence quality.** Convergence in an open hairpin turn from different initializations in red (—) over the iterations in light blue (—), to the final solution in dark blue(—). The trajectory starts at the top left and passes through the two waypoints ($\times$). While the good initialization in the left figure needs 216 iterations, the poor guess in the middle figure needs 303 iterations, but both converge to exactly the same solution with $t_N = 3.617$ s. The rightmost figure additionally shows 100 random initializations, all converging to the same solution.

depict the initial position guess in red and the convergence from light to dark blue. The good initial guess in the leftmost Fig. 9 results in 216 iterations until convergence, while the poor guess in the middle Fig. 9 needs 303 iterations. In the rightmost Fig. 9 we perform 100 uniform random initializations on a 6×6m xy-plane around the start and end point, and a 30×12meter xy-plane around the mid point. All initializations converge to the same solution. Overall, this indicates that our method is not sensitive to initialization variations in the translation space, but profits from good guesses. However, since the acceleration and orientation space of a quadrotor is not convex, the next two experiments elaborate on how to provoke and circumvent possible non-convexity issues in the rotation dynamics.

E. 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. 10.

First, the standard quadrotor configuration is used, with $N = 150$ nodes and a tolerance of $d_{tol} = 0.1$ 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 the top Fig. 10, in which it is obvious that they do not converge to the same solution. The second setup actually performs a flip which is slightly faster at 4.115 s compared to the first setup at 4.171 s (1.3% faster). This is expected due to the non-convex properties of the problem.
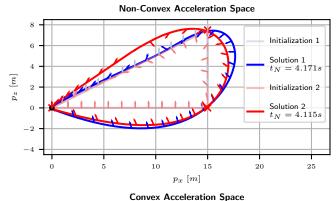
However, a second set of experiments is performed with the same initialization setups but the race quadrotor configuration. This configuration has a minimum thrust of $T_{min} = 0$ N, which renders the achievable acceleration space convex. Note that the full problem formulation still is non-convex, due to the non-linear dynamics and constraints. Both setups for the race configuration are depicted in the bottom Fig. 10, and indeed, both initializations now converge to the same solution, which overlay each other and achieve the same timing at 2.885 s.

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 Section IV-D.

F. Local-vs-Global Optimum

We follow up the previous non-convex example with another edge case where the non-convex dynamics can be used to provoke a local optimum, and we show how to resolve this special case. The problem addressed is the pure-vertical translation from hover to hover, e.g. descending from an initial hover point at 5 m height, to the origin at 0 m, parameterized with $N = 100$ nodes, a tolerance of $d = 0.1$ m, and using the race quad configuration.

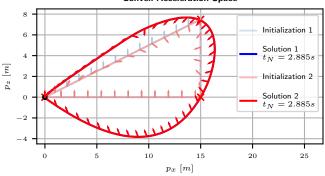When initialized with upright orientation and linear translation, the solution converges to a free-fall

Fig. 10: **Influence of convexity.** 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 (standard in the top figure and race configuration in the bottom figure), with two different initialization setups each. The first setup is as described in Section B with identity orientation, while the second setup uses a linearly interpolated orientation guess. The arrows indicate the thrust direction of the quadrotor. The standard quadrotor configuration converges to two different solutions in the top figure depending on the initialization due to its non-convex acceleration space with $T_{min} > 0\,\text{N}$, while the race quadrotor configuration converges to equal (and overlaying) solutions in the bottom figure, due to its convex acceleration space with $T_{min} = 0\,\text{N}$.

trajectory, which is a feasible local optimal solution, depicted in Fig. 11. However, the optimal solution would exploit the available inputs in addition to the gravity acceleration, effectively performing a flip to accelerate downwards, before turning upright to

decelerate into a hover state. We can resolve this problem of local optimality by initializing the problem from a bang-bang point-mass guess. This initial bang-bang guess exploits the full actuation space to accelerate for $t_{acc} = t_N/2$ towards the end state, followed by symmetric deceleration phase of equal $t_{dec} = t_N/2$, under the assumption of no gravity. Essentially, this provides us with an initial guess for our orientation, which resolves the local optimum and converges to the global solution of performing a flip, visible in Fig. 11. This initialization scheme can be applied for arbitrary trajectories to resolve most local optima.
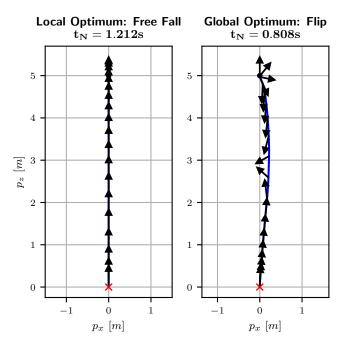

Fig. 11: **Local and global optimum.** Two solutions to travel from a starting point at $5\,\text{m}$ height to the origin at $0\,\text{m}$. The left solution is initialized from an upright linear interpolated guess converging to a locally optimal free-fall descent. In contrast, the right solution is initialized from a bang-bang acceleration guess, reaching the global optimal solution of performing a flip. Note that the global optimal flip is notably faster ($0.808\,\text{s}$), than the locally optimal free fall ($1.212\,\text{s}$).
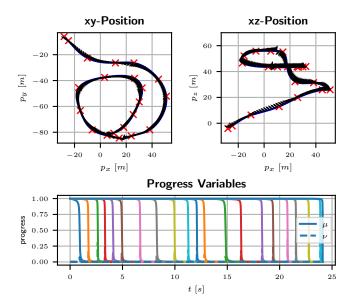
Fig. 12: **Airsim qualification.** The NeurIPS Airsim Qualification 1 track, covered in $t_N = 24.11$ s, as opposed to the best team's 30.11 s. The top row depicts the trajectory in xy- and xy-plane, while the second row depicts the velocity, respectively, plotted over time.

G. Microsoft AirSim, NeurIPS 2019 Qualification 1

As an additional demonstration, we apply our algorithm on a track from the 2019 NeurIPS AirSim Drone racing Challenge [39], specifically on the Qualifier Tier 1 setup. We choose 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 [39]. We use a discretization of $N = 3360$ nodes and a tolerance of $d_{tol} = 0.1$ m. The optimization of such a large state space took ~ 65 min on a normal desktop computer.

The original work by [39] provides a simple and conservative baseline performance of $t_{total} \approx 110$ s under maximal velocity and acceleration of $v_{max} = 30$ m s^{-1} and $a_{max} = 15$ m/s^2, respectively. However, the best team achieved a time of $t_{total} = 30.11$ s according to the evaluation page from [39]. Our method generates a trajectory that passes all waypoints at a mere $t_N = 24.11$ s, visualized in Fig. 12. Please note that this trajectory should only serve as a theoretical lower bound on the possibly achievable time given the model parameters.

APPENDIX C
HUMAN PILOT RANKINGS

TABLE IV: Rankings

Michael Isler

Year	Event	Ranking
2020	Swiss Drone League Race Luzern	2nd
	Highest Drone Race in the World - St. Moritz	1st
2019	Drone Champions League Laax	3rd
	Swiss National Team FAI World Championship Shanghai	team
2018	Southern Germany Race	3rd
	Drag Race Southern Germany	3rd
	Swiss Online Freestyle Challenge	2nd
	Kamikaze Race	1st
	FAI Lausanne	2nd
	Swiss Drone League Race St. Gallen	1st
	Drone Champions League Rapperswil	3rd
	Swiss National Team FAI World Championship Shenzen	team
	Drag Race at FAI World Championship Shenzen	3rd
2017	Swiss Nationals	2nd
	Swiss Indoor Masters	2nd
	Drone Night	1st

Timothy Trowbridge

Year	Event	Ranking
2020	Drone Racing League	par
2019	Drone Champions League Season	2nd
	Drone Champions League Romania	1st
	Drone Champions League Vaduz	2nd
	Swiss Drone League Bern	1st
	Swiss Drone League Luzern	2nd
	Swiss Drone League Basel	1st
	Drone Champions League Turin	1st
2018	Drag Race at FAI World Championship Shenzen	1st
	Drone Champions League Season	1st
	Drone Champions League Zurich	1st
	Drone Champions League Brussels	1st
	Drone Champions League China	1st
	XFly Great Wall of China	1st
	Inter Copter Racing Cup	2nd
	Thüringen Saison Opening	2nd
2017	Moto Drone St. Gallen	2nd
	Drone Champions League Brussels	2nd
	Morph X Masters	2nd
	Athens Drone GP	1st
	Drone Champions League Paris	2nd
	UpGreat Drone Race	1st
2016	Danish Drone Nationals	2nd
	Drone Night	1st

Entries where the participants competed as part of a team are marked as "team",
while participation with undisclosed results are marked as "par".