2024 European Control Conference (ECC)
June 25-28, 2024. Stockholm, Sweden

An alternating peak-optimization method for optimal trajectory generation of quadrotor drones

Wytze A.B. de Vries, Ming Li, Qirui Song and Zhiyong Sun

***Abstract*— In this paper, we propose an alternating optimization method to address a time-optimal trajectory generation problem. Different from the existing solutions, our approach introduces a new formulation that minimizes the overall trajectory running time while maintaining the polynomial smoothness constraints and incorporating hard limits on motion derivatives to ensure feasibility. To address this problem, an alternating peak-optimization method is developed, which splits the optimization process into two sub-optimizations: the first sub-optimization optimizes polynomial coefficients for smoothness, and the second sub-optimization adjusts the time allocated to each trajectory segment. These are alternated until a feasible minimum-time solution is found. We offer a comprehensive set of simulations and experiments to showcase the superior performance of our approach in comparison to existing methods.**
A collection of demonstration videos with real drone flying experiments can be accessed at `https://youtu.be/SgHXavVu7rU`.

I. Introduction

A quadrotor is a special type of unmanned aerial vehicle, which has received significant attention in recent years owing to its cost-effective design, ease of maintenance, and impressive maneuverability. Due to these advantages, quadrotors have found a wide range of applications, including environmental monitoring [1], agriculture [2], and search and rescue operations [3]. However, its operation with only four independent thrust forces leads to an underactuated system. The inherent nature of the quadrotor, compounded by its complex nonlinear dynamics, strong coupling, and multivariable actuation, presents a significant challenge when one seeks to optimize its motion for time efficiency [4].

Generally, time-optimal trajectory generation refers to the challenge of pushing a quadrotor to its theoretical limits, enabling the most aggressive motion possible. Within existing research, the continuous-time polynomials [5], [6], [7] serves as a prevalent approach for planning quadrotor trajectories. In particular, the trajectories are expressed as polynomial functions of the quadrotor's output variables, effectively leveraging the quadrotor's differential flatness property [5]. However, this approach has one major limitation, i.e., fixed running time, where the trajectory's overall running time is predetermined and not optimized for achieving the shortest possible duration, which means it is not time optimal. As an improvement, a novel time-optimal trajectory generation method is introduced in [8]. This method optimizes both the overall running time and trajectory polynomials, resulting in the generation of rapid and aggressive trajectories. However, this method fails to take control input limits into account, which is of great significance for many applications [9]. Furthermore, during the optimization process, it would stop when encountering a local minimum or reaching the limit of the maximum motor thrust. While this approach ensured that the drone utilized its maximum performance at a particular point in the trajectory, it did not guarantee the full utilization of available performance throughout the entire trajectory.

Fig. 1. Composite image of a single drone completing a trajectory generated by the proposed peak-optimization method.

In this paper, a novel time-optimal trajectory optimization is formulated, and a peak-optimization approach is proposed to address the time-optimal trajectory generation problem within the framework of continuous-time polynomials. A composite image of the drone flying in a physical world is exhibited in Fig. 1, which showcases that our method can be implemented on an actual flying platform. Different from the formulation in [8], our proposed optimization involves modifying the cost function to focus on minimizing the overall trajectory running time. While the polynomial coefficients are constrained to satisfy smoothness requirements, they are not optimized. Additionally, we explicitly set limits on the motion derivatives as hard constraints in the optimization. This guarantees that the generated time-optimal trajectories are feasible and can be executed by a given platform. To solve the optimization problem, an alternating peak-optimization method is developed. Specifically, the new optimization formulation is divided into two sub-optimization problems. For the first optimization problem, the overall running time is fixed and then the polynomial coefficients are deliberately selected to satisfy the smoothness constraints. For simplicity, we employ Mellinger's solution [5] directly to generate smooth polynomial trajectories in our implemen-

This work was supported in part by a starting grant from Eindhoven Artificial Intelligence Systems Institute (EAISI), Eindhoven, the Netherlands; in part by EU-Horizon2020 - Marie Skłodowska-Curie Actions (MSCA SE) grant No.101086228.

The authors are with the Department of Electrical Engineering, Eindhoven University of Technology, and also with the Eindhoven Artificial Intelligence Systems Institute, PO Box 513, Eindhoven 5600 MB, The Netherlands.
{w.a.b.d.vries, q.song}@student.tue.nl
{m.li3, z.sun}@tue.nl

tation, as they inherently satisfy the required smoothness constraint. In the second optimization step, the polynomial coefficients remain fixed, while the time allocated to each segment of the trajectory is adjusted via a peak optimization technique. This adjustment process changes the time intervals between segments, ensuring that the control input reaches its maximum limit, but does not exceed it to ensure feasibility. To demonstrate the advantages of our approach for generating time-optimal trajectories, our approach is compared with the existing trajectory generation methods through numerous simulation and experiment results via a platform called Crazyflie [10].

The structure of this paper is as follows. In Section II, we introduce the pertinent drone dynamics, explore the concept of differential flatness, and provide a framework for continuous-time polynomial trajectory optimization. The proposed new method to optimize the segment times is presented in Section III. Comparisons and performance of time-optimal trajectories generated by different methods are evaluated in Section IV. Experimental results of the proposed methods are presented in Section V and the paper is concluded in Section VI.

II. Preliminary on trajectory optimization

A. Polynomial trajectory optimization

When assessing the evaluation of a flat output variable denoted as $\boldsymbol{\sigma} = \left[\mathbf{r}, \psi\right]^\top$, with the position $\boldsymbol{r} = \left[x, y, z\right]^\top$, within a time interval $t = [0, \tau]$ defined by a polynomial $P(t)$ connecting two points in the flat output space, each flat output and segment trajectory are represented by a polynomial $P(t) = \sum_{i=0}^{N} p_i t^i$. The optimization of the coefficients for this polynomial can be achieved by minimizing a cost function J given by:

$$J = \int_{t=0}^{t=\tau} c_0 P(t)^2 + c_1 \dot{P}(t)^2 + c_2 \ddot{P}(t)^2 + ... + c_N P^{(N)}(t)^2 dt, \tag{1}$$

where $c_i, i = 0, \cdots, N$ denotes the weight of each derivative. This allows minimization of the total speed, acceleration, jerk, or snap present during the trajectory.

This cost function can be rewritten into a quadratic form as

$$J^k = \mathbf{p}_k^\top \mathbf{Q}_k \mathbf{p}_k, \tag{2}$$

where $\mathbf{p}_k$ is a vector that contains the coefficients of the polynomial and $\mathbf{Q}_k$ is the cost matrix. The computation of the cost matrix in the quadratic form was explained in [8]. Since trajectories often consist of multiple segments, a more comprehensive cost function is necessary to encompass all coefficients and cost matrices. This extended cost function can be succinctly expressed as:

$$J(\mathbf{p}) = \begin{bmatrix} \mathbf{p}_1 \\ \vdots \\ \mathbf{p}_K \end{bmatrix}^\top \begin{bmatrix} \mathbf{Q}_1 & & \\ & \ddots & \\ & & \mathbf{Q}_K \end{bmatrix} \begin{bmatrix} \mathbf{p}_1 \\ \vdots \\ \mathbf{p}_K \end{bmatrix}, \tag{3}$$

where $\mathbf{p} = [\mathbf{p}_1^\top, \cdots \mathbf{p}_K^\top]^\top$, and K denotes the number of the polynomial segments. By concatenating the desired constraints on the initial and final derivatives of the polynomial, we have a standard quadratic programming (QP) problem:

$$\min_{\mathbf{p}} \quad J(\mathbf{p}) \tag{4a}$$

$$\text{s.t.} \quad \mathbf{A}\mathbf{p} - \mathbf{b} = 0 \tag{4b}$$

where $\mathbf{A}$ and $\mathbf{b}$ are used to constrain the start, end or any intermediate positions, velocities or other derivatives and is also used to ensure continuity of any derivative.

B. Optimal trajectory with alternating optimization

The previously discussed method does not guarantee the minimal possible cost for the given problem since the segment times were initially fixed to construct the cost function. A solution with lower costs could probably be found by redistributing the time among the segments differently. Hence, it becomes essential to optimize both the time distribution and polynomial coefficients to achieve the most cost-effective solution. Mellinger and Kumar [5] (referred to as the "Mellinger method" hereafter) proposed an alternating optimization approach to address this challenge. This approach employs a gradient descent method, starting with an initial estimate of the segment times and iteratively optimizing the polynomial coefficients.

$$\min_{\mathbf{t}} \quad f(\mathbf{t}) \tag{5a}$$

$$\text{s.t.} \quad \sum_{\mathrm{k}=1}^{K} t_{\mathrm{k}} = T, \tag{5b}$$

$$t_{\mathrm{k}} \geq 0, \tag{5c}$$

where $f(\cdot)$ is the cost function with the optimised coefficients, $\mathbf{t} = [t_1, \cdots, t_{\mathrm{K}}]^\top$, and T is the total time.

C. Optimal trajectory with aggressive motion

An improved method to obtain aggressive motion was proposed by Bry et al. [8] (hereon referred to as "Bry method"). It removes the total time constrained in Eq. (5) so that the total time would decrease. Then the optimization function was reformulated so that both the polynomial coefficients and the segment times are optimization variables. The cost function was expanded and a cost was placed upon the total time, which was described by

$$J = \begin{bmatrix} \mathbf{p}_1 \\ \vdots \\ \mathbf{p}_K \end{bmatrix}^\top \begin{bmatrix} \mathbf{Q}_1(\tau_1) & & \\ & \ddots & \\ & & \mathbf{Q}_K(\tau_K) \end{bmatrix} \begin{bmatrix} \mathbf{p}_1 \\ \vdots \\ \mathbf{p}_K \end{bmatrix} + c_t \sum_{\mathrm{k}=1}^{K} \tau_{\mathrm{k}}, \tag{6}$$

where c_t is a weighting coefficient on the total time. This would lead to an increase in total jerk or snap, but a lower total time, which results in more aggressive and faster trajectories. The weighting coefficient could be increased if a more aggressive trajectory was desired and lowered if a trajectory with less total snap was desired.

It should be noted that the most aggressive and feasible trajectory is inherently non-smooth and would therefore not be achievable using continuous-time polynomial trajectories. Some other form of optimization such as time-optimal planning is required to achieve the most aggressive trajectory possible as was demonstrated by Foehn, Romero, and Scaramuzza in [4]. This approach does however require a very complex and accurate model of the drone, while the continuous-time polynomial trajectories require far less information and are therefore easier to implement.

III. A NOVEL TRAJECTORY OPTIMIZATION FORMULATION AND THE PEAK-OPTIMIZATION METHOD

In this section, we introduce a new time-optimal trajectory formulation that prioritizes minimizing the overall trajectory running time while preserving polynomial smoothness constraints. We also establish limits on the motion derivatives as hard constraints, which ensure the feasibility of generated time-optimal trajectories. To improve the efficiency of our optimization approach, we propose the "peak optimization" method, which splits the problem into two sub-optimization steps. First, a continuously smooth trajectory with a fixed time segment time is generated, followed by a an optimization that optimizes the segment times using a gradient descent approach. Then the two sub-optimizations are alternated until a feasible solution is found that achieves the minimal total running time.

A. A novel formulation of time-optimal trajectory optimization

We define the most aggressive trajectory possible as the trajectory that reaches all waypoints in the shortest time while staying within the limitations of the drone. There are multiple limitations that the feasible trajectories have to respect, such as maximum thrust, aerodynamic drag and the bandwidth of the low level controllers. These limitations can be mathematically expressed in motion derivatives by leveraging the property of differential flatness. Our goal is to obtain a trajectory that is both time-optimal, smooth and feasible. Therefore a new optimization formulation is presented as follows

$$\begin{aligned} \min_{\mathbf{t},\,\mathbf{p}} \quad & \sum_{\mathrm{k}=1}^{\mathrm{K}} t_{\mathrm{k}} \\ \text{s.t.} \quad & \rho_{\mathrm{s}} - \|\frac{\partial^{\mathrm{s}}\mathbf{r}_{\mathrm{k}}(t)}{\partial t^{\mathrm{s}}}\|_2 \geq 0 \ \forall \mathrm{s}, \\ & \mathbf{A}\mathbf{p} - \mathbf{b} = 0, \end{aligned} \tag{7}$$

where t_{k} is the time duration of segment $\mathrm{k} = [1, \cdots, K]$, s is the order of the motion derivative and ρ_{s} is the limit of the sth motion derivative. Here, the cost is placed only on the total time, but the constraints ensure the feasibility and smoothness of the trajectory.

The solution to this problem would require that the time and polynomial coefficients be optimized simultaneously. This is however a very difficult problem to solve analytically as it is non-linear and non-convex. Therefore we adopt the alternating approach that was introduced by Mellinger as discussed in Section II, splitting the optimization into two sub-optimization problems involving the iterative optimization of polynomial coefficients $\mathbf{p}_{\mathrm{k}}$ and segment times t_{k}. Specifically, we introduce the alternating peak-optimization method in the next subsection.

B. Alternating peak-optimization approach

Generally, the alternating peak-optimization approach involves splitting the optimization problem in Eq. (7) into two distinct optimization processes.

1) First sub-optimization problem: For the first optimization, the segment times are fixed and then the polynomial coefficients are selected to satisfy the second constraint of Eq. (7). Mellinger's optimization as described in Eq. (4) is used to generate a new smooth polynomial trajectory, as it inherently satisfies the smoothness constraints.

2) Second sub-optimization problem: For the second optimization, the polynomial coefficients are fixed and the segment times are then updated by solving the following problem using gradient descent

$$\begin{aligned} \min_{\mathbf{t}} \quad & \sum_{\mathrm{k}=1}^{\mathrm{K}} t_{\mathrm{k}} \\ \text{s.t.} \quad & \rho_{\mathrm{s}} - \|\frac{\partial^{\mathrm{s}}\mathbf{r}_{\mathrm{k}}(t)}{\partial t^{\mathrm{s}}}\|_2 \geq 0 \ \forall \mathrm{s}, \end{aligned} \tag{8}$$

This is the same problem as Eq. (7), but with the second constraint removed that was already satisfied by the first sub-optimization problem. The segment times can either be decreased to minimise the cost function of Eq. (7), or be increased to ensure feasibility by satisfying the first constraint of Eq. (7).

The complexity of checking the constraint of Eq. (8) can be significantly decreased by reducing the time dependent norm of the motion derivative to a single scalar per segment by first computing the peak magnitude of each motion derivative during each segment. This peak κ_{k} is normalised with the limit ρ_{s} and calculated for each segment k by

$$\kappa_{\mathrm{k}}(\mathrm{s}) = \frac{\max(\|\frac{\partial^{\mathrm{s}}\mathbf{r}_{\mathrm{k}}(t)}{\partial t^{\mathrm{s}}}\|_2)}{\rho_{\mathrm{s}}}, \tag{9}$$

where $s \in \{1, 2, 3, 4\}$ as the motion derivatives up to the fourth order are considered. Therefore $\kappa_{\mathrm{k}} > 1$ when its limit is exceeded and $\kappa_{\mathrm{k}} < 1$ when it is below its limit. These peak values are then used to update the current segment times $t_{\mathrm{k,old}}$ to the new segment times $t_{\mathrm{k,new}}$ ensuring that the trajectory is as close to its most restrictive limit as possible using

$$t_{\mathrm{k,new}} = t_{\mathrm{k,old}} \cdot (1 - \delta_1 \cdot (1 - \max(\boldsymbol{\kappa}_{\mathrm{k}}))), \tag{10}$$

where $\boldsymbol{\kappa}_{\mathrm{k}} = [\kappa_{\mathrm{k}}(1), \cdots, \kappa_{\mathrm{k}}(4)]^\top$ and δ_1 is the first step size. The step size was added as the shape of the trajectory (and with it the shape its derivatives) will change in non-linearly when the segment times are adjusted. Therefore the segment times are adjusted in smaller steps during each iteration. The step size can be increased to improve computational

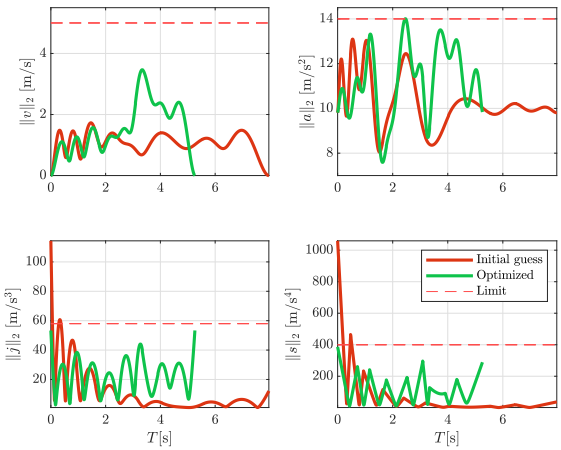

Fig. 2. Motion derivatives of an example trajectory.

efficiency, and reduced to ensure convergence. The two sub-optimization problem are alternated until the segment times can not be further reduced without violating the constraints.

C. Discussions on the alternating optimization and implementation

The effect of the proposed peak optimization will be illustrated by comparing trajectories generated by an initial guess of the segment times and the Peak optimization. Both trajectories will have their polynomial coefficients optimized for minimal jerk and their total time will be 8 seconds. The set of waypoints selected for this comparison, as shown in Fig. 4 was chosen for this example, because the quadcopter has to manoeuvre a tight section at the start that will show large values for jerk and snap, while the rest of the trajectory requires a lot of acceleration and speed, but not a lot of jerk and snap due to the long straight sections and gentle curves. Fig. 2 shows motion derivatives of a trajectory generated using an initial guess for the segment times. The initial guess for the segment times is proportional to the euclidean distance between the waypoints. The limits on jerk and snap are exceeded during the first second, which will make the trajectory infeasible. It is however far below any limit during the latter 4 seconds of this trajectory. The figure shows that the trajectory that is generated using the Peak optimization method is feasible and over 2.6 seconds faster than the infeasible initial guess. Fig. 3 show that it is done by allocating more time to the tight and twisty first four segments while reducing the segment times for the latter segments. Therefore the large peaks in jerk and snap are reduced for the first segments, and the acceleration and speeds are significantly increased for the latter segments.

Remark 1. *The acceleration is limited by the thrust the drone is able to produce. Due to gravity, the drone is able to accelerate slower in the positive z direction and faster in the negative z direction. Therefore $\kappa_{\mathrm{k}}(2)$ from Eq. (9) is adjusted to make it represent the mass normalised force that the drone has to produce to execute the trajectory. This is therefore represented by*

$$\kappa_{\mathrm{k}}(2) = \frac{max(\sqrt{(\frac{\partial^2 x_{\mathrm{k}}(t)}{\partial t^2})^2 + (\frac{\partial^2 y_{\mathrm{k}}(t)}{\partial t^2})^2 + (\frac{\partial^2 z_{\mathrm{k}}(t)}{\partial t^2} + g)^2})}{\rho_{\mathrm{s}}}, \tag{11}$$

where g is the gravitational constant.

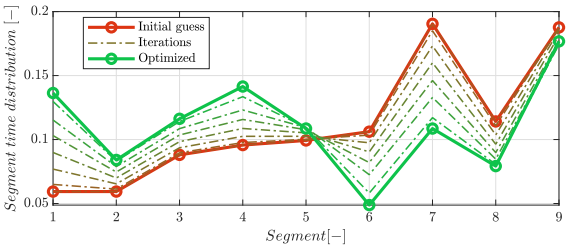

Fig. 3. Iterative optimized segment times of an example trajectory.

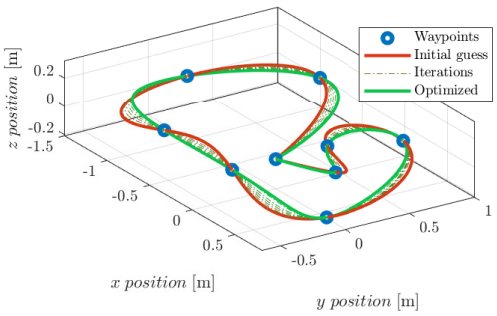

Fig. 4. Iterative optimized example trajectory.

Remark 2. *The continuity constraints make the solution of a segment depend on its neighbouring segments. Therefore it would be beneficial to not only ajust the segment time during the current segment, but also its neighbouring segments. This is done according to*

$$t_{\mathrm{k}\pm1,\mathrm{new}} = t_{\mathrm{k}\pm1,\mathrm{old}} \cdot (1 - \delta_2 \cdot (1 - max(\boldsymbol{\kappa}_{\mathrm{k}}))), \tag{12}$$

where δ_2 is the second step size.

Remark 3. *Increasing the segment time does not guarantee a decrease in all of the motion derivatives. This can be solved by scaling all the segment times uniformly when any of the limits are exceeded. This preserves the shape of the trajectory, while uniformly lowering all the derivatives. This is done according to*

$$\boldsymbol{t}_{\mathrm{new}} = \boldsymbol{t}_{\mathrm{old}} \cdot (1 - \delta_1 \cdot (1 - max(\boldsymbol{\kappa}))), \tag{13}$$

where $\boldsymbol{t}_{\mathrm{new}}$ and $\boldsymbol{t}_{\mathrm{old}}$ are vectors of all the new and old segment times respectively, and $\boldsymbol{\kappa} = [\boldsymbol{\kappa}_1, \cdots, \boldsymbol{\kappa}_{\mathrm{n_{segments}}}]$ is a matrix of all the limit normalised peak values for all motion derivatives and all segments.

IV. Comparison in Simulation

The peak optimization method was implemented and compared to Mellingers method and the initial guess. The waypoints and initial total times are the same as described in the previous sections. The trajectories shown were generated

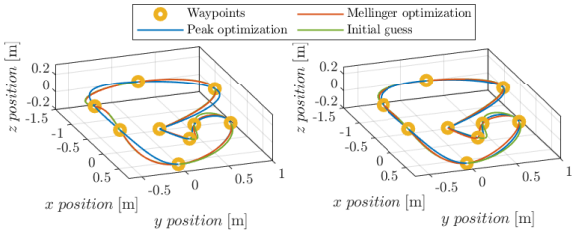

Fig. 5. Comparison of minimum jerk (left) and minimum snap (right) trajectories.

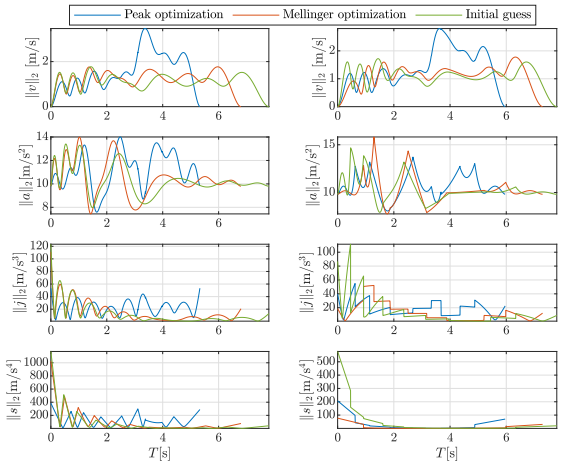

Fig. 6. Motion derivatives of minimum jerk (left) and minimum snap (right) trajectories.

to achieve the lowest total running time and were validated to be feasible during experiments which will be discussed in the next section. The limits for the peak optimization method are difficult to obtain analytically due to the large complexity of the system. Therefore these limits were determined experimentally. All trajectories used fifth order polynomials. The method proposed by Bry et. al. was not taken into account as the code available through the UAV toolbox from MATLAB [11] could not be used as it returns a 10th order polynomial while the Crazyswarm system uses 8th order polynomials.

A. Minimum jerk trajectories

The peak optimization was generated using the limits of 5 m/s, 14 m/s^2, 58 m/s^3 and 400 m/s^4 and resulted in a total time of 5.3 seconds. The Mellinger optimization used a total time of 6.8 seconds, and the initial guess used a total time of 7.8 seconds. All the resulting trajectories are shown in Fig. 5. The peak optimization methods shows lower magnitudes of speed, acceleration, jerk, and snap during the first seconds compared to the other methods as shown in Fig. 6. However these magnitudes decrease significantly after that for the initial guess and the Mellinger optimization, while the magnitudes for the peak optimization are significantly higher. The reason for this can explained with the segment time distribution as shown in Fig. 7. The peak optimization allocates significantly less time to the latter segments and more time to the first four segments.

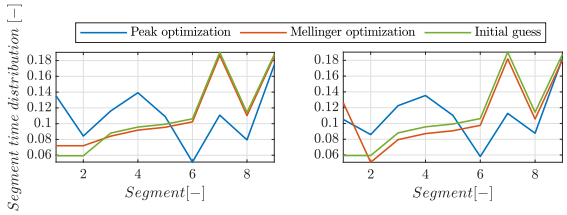

Fig. 7. Segment time distribution of minimum jerk (left) and minimum snap (right) trajectories.

Here it is also shown that the time segment distribution from the Mellinger optimization does not differ greatly from the initial guess. It should be noted that the Mellinger optimization is reliant on a good initial guess as it does not give guarantee of global optimality due to its gradient descent based approach. The maximum speed reached with peak optimization is 95% and 96% higher than the initial guess and the Mellinger optimization respectively. This results in a theoretical reduction of the total time by 32% and 21% compared to the initial guess and the Mellinger optimization respectively.

B. Minimum snap trajectories

The results are similar for when the polynomial coefficients are optimised for minimum snap. Here, the limits of 5 m/s, 13.7 m/s^2, 58 m/s^3 and 400 m/s^4 were found to be feasible and resulted in a total time of 5.9 seconds. The Mellinger optimization used a total time of 7.3 seconds, and the initial guess used a total time of 7.8 seconds, these trajectories are all shown in Fig. 5. The peak optimized method is again able to generate a faster feasible trajectory by allocating significantly more time to the first four segments and less to the latter segments as shown in Fig. 7, therefore operating significantly closer to its limit for the second half of the trajectory as shown in Fig. 6. The peak optimization methods generates a shorter total time than the initial guess and the Mellinger method by 24% and 19% respectively.

V. Implementation and Results

The trajectories shown in the previous section were flown using the crazyswarm [12] package combined with the VICON motion capture system and the Crazyflie 2.1 drones. The position controller used was the Mellinger controller [5] and a Kalman filter was used for state estimation. The motion derivatives are calculated by differentiating the position recorded by the VICON system, which was sampled at 100Hz, and smoothed using a 5 point averaging filter for each differentiation step. More results were shown and discussed in [13].

A. Minimum jerk trajectories

The peak optimised trajectory, where the polynomial coefficients were optimized for minimum jerk was 15% faster than the Mellinger optimization and 25% faster than the initial guess. The total times are summarised in Table I and the motion derivatives are shown in Fig. 8. The

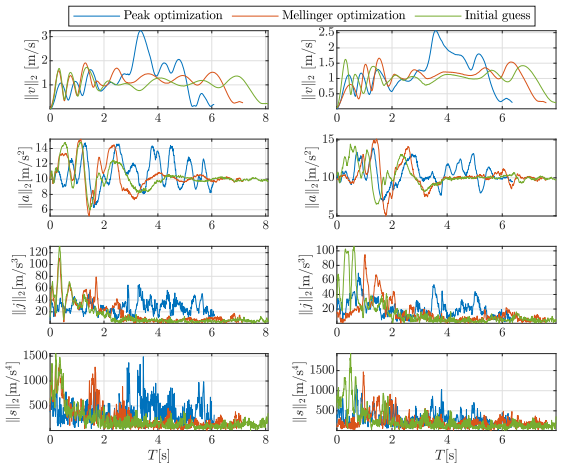

Fig. 8. Experimental results, motion derivatives of minimum jerk (left) and minimum snap (right) trajectories.

generated trajectories are used as a time dependent reference point in space and the drone uses a cascaded PID position controller, combined with a feed forward term to track this reference. The drone starts to lag behind over time when very aggressive trajectories are used, which results in non-negligible position error. The RMS position errors were 13.1 cm, 13.0 cm and 10.2 cm for the peak optimization, Mellinger optimization and the initial guess, respectively.

B. Minimum snap trajectories

As indicated by Table I, the peak optimised trajectory is 16% faster than the Mellinger optimization and 20% faster than the initial guess when the polynomial coefficients are optimized for minimum snap. Fig. 8 shows that the differences between the time optimization methods are similar as when the polynomial coefficients were optimised for minimum jerk. The RMS errors are 11.2 cm, 8.4 cm and 9.2 cm for the peak optimization, Mellinger optimization and the initial guess respectively.

TABLE I

TIME TAKEN TO COMPLETE THE FULL TRAJECTORY.

Minimised derivative / segment times optimization method	Jerk [s]	Snap [s]
Initial guess	8.09	7.95
Mellinger optimization	7.16	7.6
peak optimization	6.09	6.37

C. Discussions

It can be concluded from the previous section that the most aggressive trajectories possible with the discussed methods can be generated using the peak optimization method where the polynomial coefficients are optimised for minimum jerk. The peak optimization does not guarantee feasibility while providing the most aggressive trajectories possible as different limits for acceleration and jerk had to be used for jerk optimized and snap optimized trajectories. It is not possible to give a guarantee of feasibility when flying on the limit due to the thrust being coupled to both the acceleration and the snap. The limits also change over time due their dependency on the state of charge of the battery and external factors such as wind. However, when it is not required to obtain the most aggressive trajectory possible, the limits could be lowered and generate a guaranteed feasible trajectory, this would also lower the RMS position error.

VI. CONCLUSION

A new optimization method for time-optimal trajectory generation under the framework of continuous-time polynomials approach was presented. This new method focuses generating minimal time feasible trajectories. This is achieved by optimising the segment times and polynomials in an alternating fashion. It generates trajectories that are consistently close to the limits of the drone for each motion derivative up to the fourth order, which are directly linked to the control inputs by differential flatness. The new and existing methods were implemented in simulation and proven by real world experiments using the Crazyflie platform. The peak optimization can achieve the lowest total running time for all the compared continuous time polynomial optimization methods.

REFERENCES

[1] S. S. Mansouri, C. Kanellakis, E. Fresk, D. Kominiak, and G. Nikolakopoulos, "Cooperative coverage path planning for visual inspection," *Control Engineering Practice*, vol. 74, pp. 118–131, 2018.

[2] C. Zhang and J. M. Kovacs, "The application of small unmanned aerial systems for precision agriculture: a review," *Precision agriculture*, vol. 13, pp. 693–712, 2012.

[3] T. Tomic, K. Schmid, P. Lutz, A. Domel, M. Kassecker, E. Mair, I. L. Grixa, F. Ruess, M. Suppa, and D. Burschka, "Toward a fully autonomous UAV: Research platform for indoor and outdoor urban search and rescue," *IEEE Robotics & Automation Magazine*, vol. 19, no. 3, pp. 46–56, 2012.

[4] P. Foehn, A. Romero, and D. Scaramuzza, "Time-optimal planning for quadrotor waypoint flight," *Science Robotics*, vol. 6, no. 56, p. eabh1221, 2021.

[5] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *2011 IEEE International Conference on Robotics and Automation*, 2011, pp. 2520–2525.

[6] B. Zhou, F. Gao, L. Wang, C. Liu, and S. Shen, "Robust and efficient quadrotor trajectory generation for fast autonomous flight," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3529–3536, 2019.

[7] G. Ryou, E. Tal, and S. Karaman, "Multi-fidelity black-box optimization for time-optimal quadrotor maneuvers," *The International Journal of Robotics Research*, vol. 40, no. 12-14, pp. 1352–1369, 2021.

[8] A. Bry, C. Richter, A. Bachrach, and N. Roy, "Aggressive flight of fixed-wing and quadrotor aircraft in dense indoor environments," *The International Journal of Robotics Research*, vol. 34, no. 7, pp. 969–1002, 2015.

[9] Z. Wang, X. Zhou, C. Xu, and F. Gao, "Geometrically constrained trajectory optimization for multicopters," *IEEE Transactions on Robotics*, vol. 38, no. 5, pp. 3259–3278, 2022.

[10] G. Silano, E. Aucone, and L. Iannelli, "CrazyS: a software-in-the-loop platform for the crazyflie 2.0 nano-quadcopter," in *2018 26th Mediterranean Conference on Control and Automation (MED)*. IEEE, 2018, pp. 1–6.

[11] Mathworks, "UAV toolbox," 2024, https://nl.mathworks.com/help/uav/index.html?s_tid=CRUX_lftnav [Accessed: 15-3-2024].

[12] J. A. Preiss, W. Honig, G. S. Sukhatme, and N. Ayanian, "Crazyswarm: A large nano-quadcopter swarm," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 3299–3304.

[13] W. A. B. de Vries, M. Li, Q. Song, and Z. Sun, "An alternating peak-optimization method for optimal trajectory generation of quadrotor drones," 2023, https://arxiv.org/abs/2312.02944.