Time-Optimal Trajectory Generation with Input Boundaries and Dynamic Waypoints Constraints for A Swarm of Quadrotors

Yuyang Shen[1], Fangguo Zhao[1,2], Jiahao Mei[3], Jin Zhou[1], Jinming Xu[1] and Shuo Li[1] *
[1.] Zhejiang University, China
[2.] Northwestern Polytechnical University, China
[3.] Zhejiang University of Technology, China

ABSTRACT

Challenges in aggressive flights for a swarm of quadrotors lie in optimal trajctory generation technologies. However, using the quadrotors' physical input limits as the input boundaries in the optimization problem leaves no control margin to handle the deviation cause by disturbance, model inaccuracy and sensor imperfection, etc. Artificially setting input boundaries cannot guarantee any optimality. In this article, we describe a novel optimal trajectory generation method considering flying time and input boundaries for a swarm of quadrotors to fly through pre-defined waypoints including dynamic waypoints without collisions. We verify the method in the Gazebo simulations where a swarm of 5 quadrotors can fly through a complex 6-waypoint track including 1 dynamic waypoint in a $35m \times 35m$ space. Flight tests are performed on two quadrotors passing through 3 waypoints in a $4m \times 2m$ flight arena to demonstrate the feasibility of the proposed method in the real world.

1 INTRODUCTION

Autonomous quadrotors are widely applied and play an important role in multiple application scenarios and show their potential applications such as disaster response and air delivery scenarios. Due to their inherent agility, autonomous aggressive flight has become a very hot research area in the robotics community in recent years.

The most classic aggressive trajectory generation method is the differential-flatness based methods [1, 2, 3], which connect the waypoints using polynomials while keeping the quadrotor's inputs within pre-defined boundaries. These methods are very straightforward and can be easily deployed on real quadrotors. Thus, they are widely used in aggressive flight applications. However, although the traveling time between two waypoints can be minimized, the resulting polynomials make the trajectories far from the true optimal as the input curves cannot stay on the input boundaries for a long time and usually, the input curve just touches the boundary and then is bounced back immediately.

Another category of methods of generating aggressive trajectories for quadrotors are optimization-based methods. These methods treat the quadrotor's states and inputs as optimization variables and the quadrotor's dynamics, initial states, target states, waypoints, etc as constraints to formulate an optimization problem. By solving the optimization problem, the optimal inputs and the corresponding states can be calculated to minimize an optimization object such as flying time, total energy or a combination of both. For example, in [4], the time-optimal trajectories are generated that can guide one quadrotor to fly through pre-defined waypoints and the quadrotor's speed has surpassed the human pilots. To make the optimal guidance and control method onboard, machine learning techniques are used to learn the off-board generated optimal trajectory libraries [5, 6, 7]. However, these methods don't take the quadrotor's input boundaries into consideration in their time optimization problems. If the input boundaries are set exactly the same as the quadrotor's physical limits, the resulting trajectories should be very close to the 'theoretical' optimal trajectories. However, in real-world flights, the quadrotor will have no more ability to handle the deviation caused by disturbance, model inaccuracy and sensor imperfection, etc. A common way to handle this is to artificially lower the inputs' boundaries to leave some capacity for the feedback controllers to handle the deviations. However, artificially setting input boundaries cannot guarantee any optimality. Furthermore, most research about the quadrotor's aggressive flight focuses on single drone scenarios instead of a swarm of quadrotors.

In term of quadrotors' swarm, the majority of research use polynomials to navigate quadrotors but the optimality cannot be guaranteed [8, 9]. In [10], a swarm of micro quadrotors were used for the search and rescue tasks in an unknown environment. Carlos et al. presented a novel real-time and multi-vehicle motion planning framework, enabling complex transition tasks to be performed [11]. Duisterhof et al. used a swarm of quadrotors to seek gas-leaking in cluttered environments [12]. Zhou et al. realized a swarm of quadrotors flying through a bamboo forest [13]. However, the velocity of

*Email address(es): shuo.li@zju.edu.cn

the quadrotors in above research is still far from the quadrotors' limits. In our previous work [14], we proposed a novel optimization-based trajectory generation method for a swarm of racing drones that can generate the time-optimal trajectories for quadrotors to fly through the pre-defined waypoints while avoiding collisions with each other. In this paper, based on our previous work, we

1. jointly optimize the flight time and input boundaries in a newly defined optimization objective to leave control margins to handle disturbance while keeping high speed
2. add dynamic waypoint constraints to the optimization problems, which enables a swarm of quadrotors to fly through a more complex track
3. valid and analyze the proposed method in simulations where 5 quadrotors can fly through 6 waypoints with a maximum speed of $14m/s$ and also in the real-world flight tests to demonstrate the feasibility of the proposed method.

2 PRELIMINARY

For the benefit of the readers, we present the quadrotor's dynamics model below,

$$\dot{\mathbf{x}} = \mathbf{f}_{dyn}(\mathbf{x}, \mathbf{u}) = \begin{cases} \mathbf{v} \\ \mathbf{g} + \frac{1}{m}\mathbf{R}(\mathbf{q})\mathbf{T} \\ \frac{1}{2}\Lambda(\mathbf{q}) \begin{bmatrix} 0 \\ \boldsymbol{\omega} \end{bmatrix} \\ \mathbf{J}^{-1}(\boldsymbol{\tau} - \boldsymbol{\omega} \times \mathbf{J}\boldsymbol{\omega}) \end{cases} \tag{1}$$

where

$$\mathbf{T} = \begin{bmatrix} 0 \\ 0 \\ \sum T_s \end{bmatrix}$$

is the thrust vector of the quadrotor and

$$\boldsymbol{\tau} = \begin{bmatrix} l/\sqrt{2}(T_1 + T_2 - T_3 - T_4) \\ l/\sqrt{2}(-T_1 + T_2 + T_3 - T_4) \\ c_\tau(T_1 - T_2 + T_3 - T_4) \end{bmatrix}$$

is the torque vector of the quadrotor. In the equations above, $\mathbf{v}$, $\mathbf{q}$ and $\boldsymbol{\omega}$ are velocity, quaternions and angular velocity of the quadrotor, respectively. $\mathbf{R}(\mathbf{q})$ is the rotation matrix. l is the arm length. T_s is the thrust provided by the s^{th} rotor, which are the inputs of the dynamic system (1).

In the original CPC method [4], a progress measure variable matrix $\boldsymbol{\lambda}$ and a progress change matrix $\boldsymbol{\mu}$ are added to the optimization problem to control the assignment of the waypoints to ensure that one quadrotor can fly through the pre-defined waypoints. To be more specific, the trajectories are discretized to N nodes while matrice $\boldsymbol{\lambda}$ and $\boldsymbol{\mu}$, which are the optimization variables, are used to assign the nodes to the waypoints. The optimization goal is to adjust the flight time t_f (also the time interval between two nodes $\Delta t = t_f/N$) of the entire trajectory while satisfying the waypoint pass-through constraints and the quadrotors' dynamics constraints. However, the CPC method cannot be directly extended to multi-drone scenarios as each quadrotor has its own t_f so the nodes cannot be aligned. As a result, it is difficult to add collision constraints to the original optimization problem.

Thus, in our previous work [14], a novel method was proposed to generate time-optimal trajectories for a swarm of autonomous racing drones so that during the flight they can fly through the pre-defined waypoints while avoiding each other and arrive at the goals with minimum time. To address collision avoidance between the quadrotors, the nodes of the quadrotors are synchronized by using the fixed time interval Δt. Then, the optimization problem is transformed from adjusting the total flight time t_f while the quadrotor arrives at the goal at the last node to adjusting the arrival node with the fixed time interval Δt. The constraints are listed in equations below. Here, we use a left subscript i to indicate that the variables belong to the i^{th} quadrotor.

$${}_i\lambda_k^j \leq {}_i\lambda_k^{j+1} \tag{2}$$

$${}_i\lambda_{k+1}^j - {}_i\lambda_k^j + {}_i\mu_k^j = 0 \tag{3}$$

$${}_i\mu_k^j(\left\|{}_i\mathbf{P}_k - {}_i\mathbf{P}^{wj}\right\|_2^2 - {}_i\nu_k^j) := 0 \tag{4}$$

$$\|\mathbf{E}({}_i\mathbf{P}_k - {}_r\mathbf{P}_k)\|_2^2 - \delta_{col} \geq 0 \qquad i \neq r \tag{5}$$

where ${}_i\lambda_k^j$ is a bool variable indicating if the i^{th} quadrotor has flew through the j^{th} waypoint at time t_k. ${}_i\mu_k^j$ means if the i^{th} quadrotor is passing through the the j^{th} waypoint. ${}_i\nu_k^j$ is a tolerance slack. The operator ':=' means a NAND (not and) function. ${}_i\mathbf{P}_k$ is the position of the i^{th} quadrotor at time t_k. ${}_j\mathbf{P}^{wp}$ is the position of the j^{th} waypoint and $\delta_{col} > 0$ is the tolerance ensuring that two quadrotors do not collide with each other. $\mathbf{E}$ is the matrix used to relieve downwash risk, making quadrotors try to avoid collisions in x and y directions instead of z direction. The readers are referred to [14] for details of the proposed method.

As stated before, if we set the input boundaries of the optimization problem to be the real quadrotor's physical limits, the quadrotor will have no ability to handle the deviation caused by disturbance, model inaccuracy, sensors' imperfection, etc, which in turn can significantly affect the flight

performance or even cause failures. A common way to handle this is to artificially lower the inputs' boundaries to leave some capacity for the feedback controllers to handle the deviations. However, artificially setting input boundaries cannot guarantee any optimality. Thus, in the following part, we model the flying time and input boundaries in our trajectory optimization problem to generate trajectories that can guide a swarm of quadrotors to fly through the pre-defined waypoints while achieving the minimal optimization objective.

3 METHODOLOGY

3.1 Dynamic Waypoint Constraints

Inspired by some aggressive flight scenarios such as autonomous drone racing where waypoints/racing gates are not only static but also moving with some specific patterns [15, 16], in this article, one of the waypoints moves with a pre-defined trajectory while others keep static during the flight. Thus, the generated trajectories may guide different quadrotors to fly through the dynamic waypoint at different locations. In our previous work [14], the corresponding constraints are expressed in equation (4).

However, in dynamic-waypoint scenarios, the position of the waypoints is no longer a constant and it becomes a function of time. Thus, constraints (4) becomes

$$ {}_i\mu_k^j(\left\|{}_i\mathbf{P}_k - {}_i\mathbf{P}(k\cdot\Delta t)^{wj}\right\|_2^2 - {}_i\nu_k^j) := 0 \;\; t\in[0,N], \quad (6)$$

where t is the k^{th} node.

3.2 Input Boundary Constraints

In the previous work, the optimization objective was to minimize the time for the quadrotors to fly through the whole tracks while keeping the inputs within pre-defined boundaries, which results in a bang-bang profile that the control inputs spend more time at the boundaries [4, 17]. As stated before, if the input boundaries are set exactly the same as the quadrotor's physical limits, the generated trajectories will leave no margin to handle the deviation caused by disturbance, model inaccuracy and sensor imperfections, etc and artificially setting lower boundaries may lose the optimality. Thus, to solve this problem we introduce an input scaling factor ${}_ie$ for each quadrotor i to adjust the inputs' boundaries.

$$0 \le {}_ie \le 1 \quad (7)$$

when ${}_ie = 1$, the optimizer will take advantage of the full throttle of the quadrotors and the generated trajectories are the most aggressive but the quadrotors are vulnerable to the disturbance as they have no ability to handle the deviation. Now the input constraints of the original problem can be written as

$${}_ie\, u_{min} \le {}_iu \le {}_ieu_{max}. \quad (8)$$

The scaling factors ${}_ie$ serve as the optimization variables in our optimization problem. Thus, the optimizer can limit the inputs' boundaries by adjusting ${}_ie$ and at last ${}_ie$ for each quadrotor will converge to constant values to make the optimization objective minimal, which will be explained later.

3.3 Thrust Change Rate Constraints

According to [17], the time-optimal trajectory of a quadrotor has a bang-bang structure which is very difficult to be tracked by a quadrotor. Because the actuators have their inertia, it is impossible for rotors to achieve the required RPMs immediately. To model this property, we add the thrust's changing rate to avoid the thrust's 'step' change which is impossible to realize in the real world.

$$-\dot{u}_{max} \le \frac{u_{k+1}-u_k}{\Delta t} \le \dot{u}_{max}\ \forall k\in[0,N-1], \quad (9)$$

where $\dot{u}_{max}$ is the maximum thrust changing rate and Δt is the time interval between to nodes.

3.4 Multi-objective Optimization Target

The target of the optimization problem in this article is to generate time-optimal trajectories while leaving some control margin for disturbance rejection. However, minimum time and the inputs' lowest upper boundary are contradictory. Thus, the optimization object can be written as

$$\min_{\mathbf{X}} J = J_t + J_e \quad (10)$$

where

$$J_t = \sum_i^Q \mathcal{S}({}_i\boldsymbol{\lambda})$$

is an index measuring the total flying time of the quadrotors, which is the sum of ${}_i\boldsymbol{\lambda}$ matrices of all quadrotors and

$$J_e = \sum_{i=0}^Q {}_ie$$

is an index indicating the input boundaries of all quadrotors. However, in most cases $J_t \gg J_e$. In order to optimize two terms equally, here we introduce two functions $\mathcal{F}(\cdot)$ and $\mathcal{G}(\cdot)$ to normalize these two terms. Then the target in (10) can be written as

$$\min_{\mathbf{X}} J = \alpha\mathcal{F}(J_t) + (1-\alpha)\mathcal{G}(J_e) \quad (11)$$

where α is the weighting factor. The selection of normalized functionals $\mathcal{F}(\cdot)$ and $\mathcal{G}(\cdot)$ directly affects the optimization results and the iterations, which will be analyzed and compared in detail in Section 4.

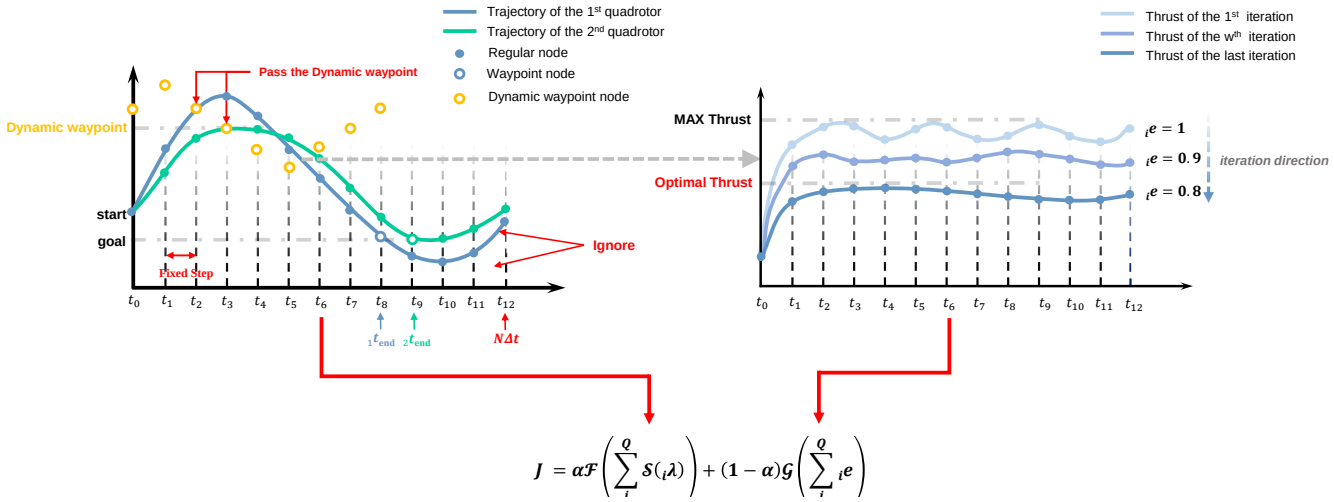


Figure 1: Sketch of the proposed method. The time step is fixed so that it is possible to add collision constraints for multiple quadrotors. The left figure shows the collision constraint and the dynamic waypoint constraint. The trajectory of dynamic waypoints is shown as yellow circles. The right figure shows that the maximum thrust decreases with the optimization. The optimization objective is to minimize the sum of the ${}_i\boldsymbol{\lambda}$ and ${}_ie$ while satisfying the dynamics constraints and collision constraints, etc.

3.5 *Optimization Problem Summary*

The optimization objective is to minimize the flying time and the input boundaries, which can be expressed as

$$\min_{\mathbf{X}} J = \alpha\mathcal{F}(\sum_{i}^{Q}\mathcal{S}({}_i\boldsymbol{\lambda})) + (1-\alpha)\mathcal{G}(\sum_{i}^{Q}{}_ie) \tag{12}$$

where Q is the number of quadrotors in the swarm. $\mathbf{X}$ is a set consisting of the optimization variables ${}_i\mathbf{x}$ of all the Q quadrotors, and ${}_i\mathbf{x} = [{}_ie, {}_i\mathbf{x}_0, {}_i\mathbf{x}_1, ..., {}_i\mathbf{x}_{N-1}]$ Note that all nodes of each drone share the same energy variable ${}_ie$ in the state variables ${}_i\mathbf{x}$, which is different from other state variables are redefined at each node ${}_i\mathbf{x}_k$.

4 NUMERICAL SOLUTION AND ANALYSIS

4.1 *Numerical simulation setup*

We utilize the nonlinear solver CasADi[18] to solve the optimization problem described in Section 3. A 6-waypoint track is designed to test the performance of the proposed method. The position of the waypoints is listed in Table 1. It should be noted that the second waypoint is a dynamic waypoint whose position changes in y direction by

$$\mathbf{P}_1^{wj}(t) = \begin{bmatrix} x_1(t) \\ y_1(t) \\ z_1(t) \end{bmatrix} = \begin{bmatrix} 25 \\ 5 + 6\sin(2\pi t/1.5) \\ 3 \end{bmatrix}$$

or

$$\mathbf{P}_2^{wj}(t) = \begin{bmatrix} x_2(t) \\ y_2(t) \\ z_2(t) \end{bmatrix} = \begin{bmatrix} 25 \\ 5 + 3\sin(2\pi t/6) \\ 3 \end{bmatrix}$$

to show the performance of the proposed method with different dynamic waypoints' speeds. 5 quadrotors ($Q = 5$) are used in the optimization problem. In the following part of this section, we will discuss the normalization functions and the weighting factors in detail.

Table 1: The position of the waypoints

waypoint NO.	$x[m]$	$y[m]$	$z[m]$
1	5	15	2
2	25	$5 + y_*(t)$	3
3	20	25	5
4	14	14	2
5	18	18	6
6	5	14	4

4.2 *normalization function*

We select the normalization function $\mathcal{G}(\cdot)$ for J_e as

$$\mathcal{G}(J_e) = \frac{1}{Q}J_e \tag{13}$$

since

$$J_e \in [0, Q] \tag{14}$$

so that

$$\mathcal{G}(J_e) \in [0, 1] \tag{15}$$

We also have two normalization functions $\mathcal{F}(\cdot)$ as the candidates to map the J_t term in equation (12) to $[0, 1]$ which are sigmoid function

$$\mathcal{F}_{sigmoid}(J_t) = \frac{\exp \mathcal{V}(J_t)}{1 + \exp \mathcal{V}(J_t)} \tag{16}$$

where

$$\mathcal{V}(J_t) = 0.005(J_t - V_m)$$

and linear mapping function

$$\mathcal{F}_{linear}(J_t) = \frac{1}{2(V_m - V_o)} \frac{1}{Q} J_t - \frac{V_o}{2(V_m - V_o)} \tag{17}$$

where

$$V_m = N \sum_{i=0}^{N_{wp}} \frac{i}{N_{wp}}$$

represents that the quadrotor passes through N_{wp} waypoints with the same time interval and

$$V_o = \frac{v_{min}}{v_{max}} V_m$$

is the value obtained by using the ratio of the maximum velocity v_{max} and minimum velocity v_{min} as the scaling factor.

To control the effects introduced by other variables, we set $\alpha = 0.8$ to check the optimization performance with different $\mathcal{F}(\cdot)$. The result is shown in Table 2.

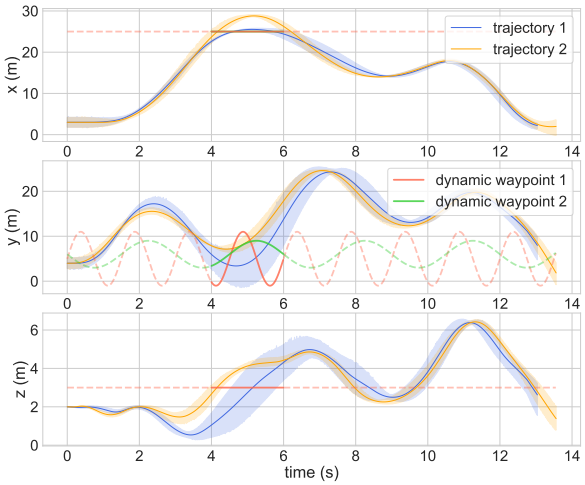


Figure 2: Trajectories of 5 quadrotors with two different dynamic waypoints. The trajectories of fast and slow dynamic waypoints are shown in red and green curves respectively.

Both functions can be guaranteed to be differentiable within the range of values. As shown in Table 2, the smoother normalization function, sigmoid, achieves better results in fewer iterations than the lower objective function value. However, its total time is hundreds of times of the linear function. Thus, we select the linear function as the normalization function. The generated optimal trajectories with $\mathcal{F}(J_t) = \frac{1}{2(V_m - V_o)} \frac{1}{Q} J_t - \frac{V_o}{2(V_m - V_o)}$ and $\alpha = 0.8$ is shown in Figure 2 and Figure 3.

Table 2: Two different normalization functions

Function type	sigmoid function	linear function
Iterations	7165	13162
Objective value	$4.02e{-2}$	$9.05e{-1}$
Single iteration time (s)	71.94	0.73
Total time (s)	515450	9608

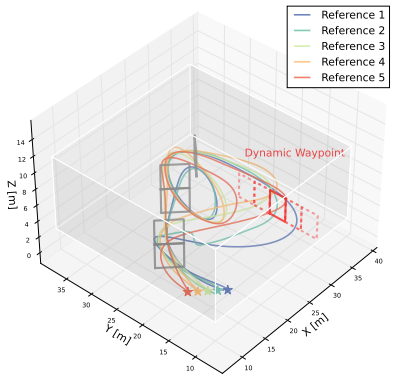


Figure 3: The generated optimal trajctories for 5 quadrotors

4.3 *Weighting factor*

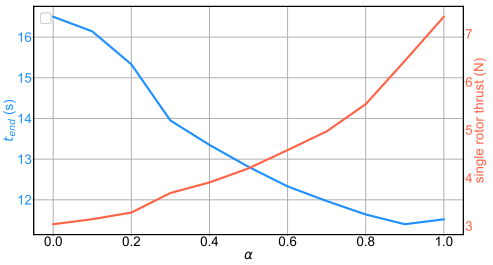
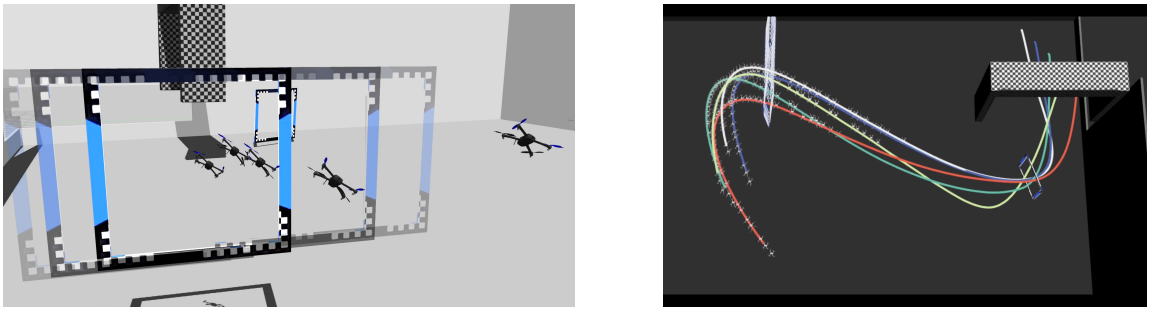


Figure 4: The flying time and maximum thrust with different α

In this section, we do a parameter study of α to show how this weighting factor influences the optimization result. We solve the optimization problem (12) with varying α and the track in Table 1 for 5 quadrotors and the result is shown in Figure 4

As shown in Figure 4, as α gradually increases, the flying time of the quadrotors decreases while the maximum thrust



(a) (b)

Figure 5: Gazebo simulation environment. 5 quadrotors are flying through the dynamic gate.

increases as discussed in Section 3. Figure 4 gives us a rough idea of how to tune α for some specific application scenarios.

5 SIMULATION AND REAL-WORLD FLIGHT EXPERIMENT

We first verify the proposed method in Gazebo simulation environments. The racing gates (waypoints) are deployed according to Table 1. The optimization result is used as the reference trajectories to be tracked by the quadrotors. A nonlinear model predictive controller (NMPC) is used for trajectory tracking and the control structure is shown in Figure 6. The model used in the NMPC is a 9-state quadrotor model whose states are the quadrotor's position, velocity, quaternions while the inputs are the angular velocities of the quadrotor. The angular velocity controller is a default PX4 PID controller for tracking the commands provided by the NMPC. The ACADO software is used to implement the NMPC algorithm. The readers are referred to [19] for the implementation details.

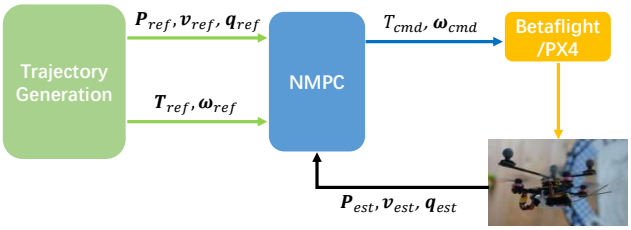


Figure 6: The control structure for the trajectory tracking. The nonlinear model predictive controller is used to track the reference trajectory. The PX4 is used as a low-level controller in simulations while the Betaflight is used in the real-world experiments.

The simulation results show that the proposed method can generate aggressive trajectories for a swarm of racing drones to fly through a pre-defined track including a dynamic waypoint. Also, as we use α to leave control margins for disturbance and model inaccuracy.

For the real-world experiments, we develop a new micro quadrotor that only weights $100g$ as shown in Figure 7. This

Figure 7: The $100g$ flying platform used in the experiment.

quadrotor runs the Betaflight onboard to provide an angular rate loop controller as it has a very accurate and stable angular velocity tracking performance. It receives angular velocity commands and throttle commands from the high-level controller. The high-level controller NMPC runs on a laptop. The onboard autopilot sends the IMU measurements and the attitude estimation to the laptop using the MAVLink protocol via the onboard WiFi module and the Optitrack motion capture system also sends the quadrotors' position measurements to the laptop. The NMPC optimizes the control inputs, the angular rates and the throttle in our case, and sends them to the onboard autopilot via the WiFi module to control the quadrotors to track the trajectories.

Due to the very limited size of the flying arena $4m \times 2m \times 2m$, we use two quadrotors to give a flying example to demonstrate the performance of the proposed method in the real world. The virtual dynamic waypoint moves in y direction by $\mathcal{P}_1^{wj}(t)$. The virtual dynamic waypoint and the trajectories of the quadrotors are shown in Figure 8.

From Figure 8, it can be seen that in the real world, the proposed method can generate the optimal trajectories for a swarm of quadrotors to fly through dynamic waypoints and also minimize the input boundaries to leave margin for the disturbance and model inaccuracy.

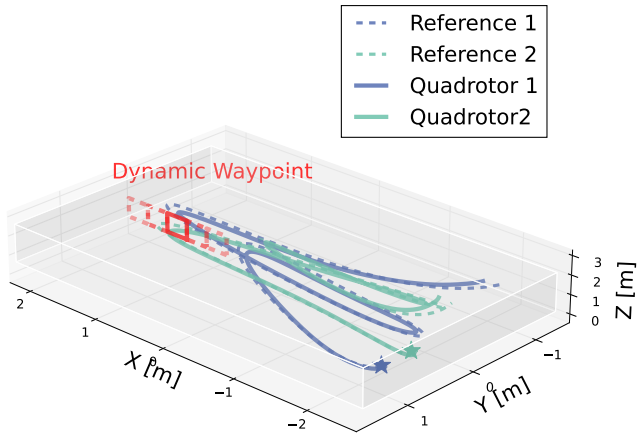


Figure 8: The collision-free trajectories of two quadrotors in the real-world experiment. The trajectory of the dynamic waypoint as shown in dashed red gates

6 CONCLUSION

In this article, we proposed a novel trajectory generation method for a swarm of quadrotors that considers the inputs' boundaries in the optimization objective. So that the generated optimal trajectories leave control margins for rejecting the deviation caused by the disturbance, model inaccuracy and sensors' imperfection. Furthermore, we also take dynamic waypoints into consideration in our optimization problem so that the quadrotors can fly through them with the optimal trajectories. Furthermore, the simulation and real-world flight results valid that the proposed trajectory generation method can guide a swarm of quadrotors flying through the waypoints (including dynamic waypoints) with the optimal index which is not only time optimal but also leaves control margins for the deviation correction.

REFERENCES

[1] Daniel Mellinger and Vijay Kumar. Minimum snap trajectory generation and control for quadrotors. In *2011 IEEE International Conference on Robotics and Automation*, pages 2520–2525, 2011.

[2] Daniel Mellinger, Nathan Michael, and Vijay Kumar. Trajectory generation and control for precise aggressive maneuvers with quadrotors. *The International Journal of Robotics Research*, 31(5):664–674, 2012.

[3] Adam Bry, Charles Richter, Abraham Bachrach, and Nicholas Roy. Aggressive flight of fixed-wing and quadrotor aircraft in dense indoor environments. *The International Journal of Robotics Research*, 34(7):969–1002, 2015.

[4] Philipp Foehn, Angel Romero, and Davide Scaramuzza. Time-optimal planning for quadrotor waypoint flight. *Science Robotics*, 6(56):eabh1221, 2021.

[5] Gao Tang, Weidong Sun, and Kris Hauser. Learning trajectories for real-time optimal control of quadrotors. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3620–3625. IEEE, 2018.

[6] Shuo Li, Ekin Öztürk, Christophe De Wagter, Guido CHE De Croon, and Dario Izzo. Aggressive online control of a quadrotor via deep network representations of optimality principles. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6282–6287. IEEE, 2020.

[7] Sebastien Origer, Christophe De Wagter, Robin Ferede, Guido CHE de Croon, and Dario Izzo. Guidance & control networks for time-optimal quadcopter flight. *arXiv preprint arXiv:2305.02705*, 2023.

[8] Alex Kushleyev, Daniel Mellinger, Caitlin Powers, and Vijay Kumar. Towards a swarm of agile micro quadrotors. *Autonomous Robots*, 35(4):287–300, 2013.

[9] Wolfgang Hönig, James A Preiss, TK Satish Kumar, Gaurav S Sukhatme, and Nora Ayanian. Trajectory planning for quadrotor swarms. *IEEE Transactions on Robotics*, 34(4):856–869, 2018.

[10] KN McGuire, Christophe De Wagter, Karl Tuyls, HJ Kappen, and Guido CHE de Croon. Minimal navigation solution for a swarm of tiny flying robots to explore an unknown environment. *Science Robotics*, 4(35):eaaw9710, 2019.

[11] Carlos E Luis, Marijan Vukosavljev, and Angela P Schoellig. Online trajectory generation with distributed model predictive control for multi-robot motion planning. *IEEE Robotics and Automation Letters*, 5(2):604–611, 2020.

[12] Bardienus P Duisterhof, Shushuai Li, Javier Burgués, Vijay Janapa Reddi, and Guido CHE de Croon. Sniffy bug: A fully autonomous swarm of gas-seeking nano quadcopters in cluttered environments. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 9099–9106. IEEE, 2021.

[13] Xin Zhou, Xiangyong Wen, Zhepei Wang, Yuman Gao, Haojia Li, Qianhao Wang, Tiankai Yang, Haojian Lu, Yanjun Cao, Chao Xu, et al. Swarm of micro flying robots in the wild. *Science Robotics*, 7(66):eabm5954, 2022.

[14] Yuyang Shen, Jinming Xu, Jin Zhou, Danzhe Xu, Fangguo Zhao, Jiming Chen, and Shuo Li. Aggressive trajectory generation for a swarm of autonomous racing drones. *arXiv preprint arXiv:2303.00851*, 2023.

[15] Hyungpil Moon, Jose Martinez-Carranza, Titus Cieslewski, Matthias Faessler, Davide Falanga, Alessandro Simovic, Davide Scaramuzza, Shuo Li, Michael Ozo, Christophe De Wagter, et al. Challenges and implemented technologies used in autonomous drone racing. *Intelligent Service Robotics*, 12:137–148, 2019.

[16] Ziyu Zhou, Gang Wang, Jian Sun, Jikai Wang, and Jie Chen. Efficient and robust time-optimal trajectory planning and control for agile quadrotor flight. *arXiv preprint arXiv:2305.02772*, 2023.

[17] Dharmesh Tailor and Dario Izzo. Learning the optimal state-feedback via supervised imitation learning. *Astrodynamics*, 3:361–374, 2019.

[18] Joel AE Andersson, Joris Gillis, Greg Horn, James B Rawlings, and Moritz Diehl. Casadi: a software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, 11:1–36, 2019.

[19] Davide Falanga, Philipp Foehn, Peng Lu, and Davide Scaramuzza. Pampc: Perception-aware model predictive control for quadrotors. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–8. IEEE, 2018.