Online State-to-State Time-Optimal Trajectory Planning for Quadrotors in Unknown Cluttered Environments

Binh Nguyen[1], Manzur Murshed[2], Tanveer Choudhury[1], Kathleen Keogh[1], Gayan Kahandawa Appuhamillage[1] and Linh Nguyen[1]

Abstract— **This paper introduces the first planner, called STAMINER (STAte-to-state tiMe-optImal memoryless planNER), which is able to real-time plan collision-free, local time-optimal trajectories in unknown and cluttered settings without the need for any maps or fused occupancy structures of the surrounding environment. Specifically, our method explores a library of state-to-state time-optimal trajectories in a memoryless collision-checking framework. It iteratively searches for the best trajectory with the longest projection on the direction to the goal. Results obtained from two different simulated cluttered scenarios demonstrate that our planner outperforms the state-of-the-art baselines concerning global finishing time. Furthermore, real-world flight trials were conducted to validate the effectiveness of our algorithm in an actual quadrotor. Finally, this paper also provides a mathematical proof of the solution characterization that is not available in the literature for the considered state-to-state time-optimal trajectory generation problem.**

I. INTRODUCTION

Today, while significant developments in aerial robotics have been enabling multi-sector applications, including inspection, search and rescue, surveillance, exploration, and more [1], [2], autonomous navigation in complex scenarios still presents challenges, especially in unknown obstacle-dense environments. The localization and mapping of aerial robots can be affected by state estimation uncertainty and sensor fusion error [3]–[5]. On the other hand, updating an online large-scale map requires substantial computing power [6]. Therefore, computationally efficient approaches, namely mapless [7]–[10] and memoryless planning [11]–[13], have been intensively studied to address those mentioned shortcomings by removing the need of *a priori* global maps and utilizing only the most recent sensor data. Applying either method, the robot has to rely completely on its onboard sensing for perception and local planning. While mapless methods still fuse one among several types of grid-based occupancy data structures for online local perception, memoryless algorithms plan directly on sensor data, reducing the required computational workload even further.

A typical approach among memoryless algorithms is to adopt a learning-based technique, using recent machine learning advancements. For example, using imitation learning, Loquercio *et al.* [14] planned high-speed trajectories directly from depth images and current states. The authors of [12] employed a deep prediction network to account for uncertainty in state estimation while predicting collision probabilities. However, because of the learning uncertainty, the robot may still collide with obstacles or get trapped in local minima, degrading overall success rates. To eliminate learning uncertainty, [13] proposed a hybrid planner that deterministically computes the best-direction path inside safe flight corridors generated by a learning-based object detector. However, their flight velocity is limited, and the flight time cannot be minimized.

Adopting classic memoryless methods does not suffer from learning uncertainty and often relies on sampling techniques. RAPPIDS planner [11] samples terminal position inside the field of view (FoV) space of the fixedly-mounted depth camera for trajectory generation using the minimum jerk library [15]. The trajectories are sampled, generated and checked for potential collisions in an iterative manner to identify the one that flies fastest to the destination. A set of rectangular pyramid safe corridors inflated on-demand directly in the latest depth image are used to check generated trajectories for collision. While RAPPIDS is able to navigate in cluttered environments as shown in their following work [16], it can easily get trapped in local minima, e.g., when facing an obstacle large enough to occupy the FoV. DESS [17] later improves RAPPIDS's efficiency and proposes a depth-based steering mechanism to further guarantee planning completeness for scenarios of convex obstacles. To do that, DESS searches for the best trajectory over a direction function that is independent of the trajectory duration. Thus, the total flight time cannot be minimized. Employing a state-to-state time-optimal (SSTO) trajectory generator may address that limitation thanks to its time optimality. Beul *et al.* [18] proposed to sample space using the SSTO trajectories presented in [19]. The sampled SSTO trajectories are then checked for collision with a single inflated axis-aligned bounding box (AABB) inside the lidar's point cloud. Since a single AABB cannot extract surrounding free space sufficiently, their method is not designed for planning in cluttered obstacle-dense settings, and it has only been tested with single obstacle scenarios. Moreover, their yaw control is explicitly provided by a global planner. Without that external yaw instruction, their planner may be unable to escape a local minima situation in a complex environment.

Therefore, the motivation behind this work is the desire to reduce the flight finishing time toward a destination for a quadrotor when navigating in cluttered environments. To

This research was supported by an Australian Government Research Training Program (RTP) Scholarship.

[1]Binh Nguyen, Tanveer Choudhury, Kathleen Keogh, Gayan Kahandawa Appuhamillage and Linh Nguyen are with the Institute of Innovation, Science and Sustainability, Federation University Australia.

[2]Manzur Murshed is with the School of Information Technology, Deakin University.

achieve that, this paper will consider SSTO trajectories in a memoryless collision-checking framework, which constitutes the first planner that does not need any maps or fused occupancy structures to generate collision-free, local time-optimal trajectories in unknown and cluttered settings.

State-of-the-art SSTO algorithms are deterministic, i.e., the solution is directly solved. Hehn *et al.* [20], [21] provided a solution for the problem of SSTO under jerk and acceleration constraints that can be solved in real time by the bisection algorithm. However, their method is still not efficient enough for online iterative sampling and collision checking. Furthermore, the absence of velocity constraints may lead to intractable non-linear aerodynamic effects on quadrotors [22], [23]. Beul *et al.* [19], [24] later intuitively derived an analytical solution for the SSTO problem of a triple integrator under jerk, acceleration, and velocity constraints. Further exploiting the same intuition, Berscheid *et al.* [25] contributed *Ruckig*, a real-time SSTO trajectory generator for the same problem. However, neither work has provided a mathematical proof for the formulation of their bang-zero-bang solution. This paper will apply classic optimal control theory to prove that the solution input derived in [19], [25] is, in fact, a bang-zero-bang control law.

This paper's contributions can be summarised as follows:

1) A mathematical proof for the characterization of the optimal solution found by the intuitive-based SSTO trajectory generation methods [19], [25].
2) A novel sampling-based planner, **STAMINER**, which employs the state-to-state time-optimal trajectory and memoryless collision checking. To the best of the authors' knowledge, this is the first navigational planner that can real-time generate local time-optimal and collision-free trajectories in unknown and cluttered environments without using any maps or fused occupancy structures.
3) Extensive simulation and real flights, where the obtained results verify the proposed method's outperformance regarding flight finishing time over state-of-the-art memoryless planners [11], [17].

II. STATE-TO-STATE TIME-OPTIMAL TRAJECTORY GENERATION

This section describes the quadrotor dynamic model and formulates a corresponding SSTO trajectory that guarantees traceability and input feasibility for control of that model. Then, the proof for the characterization of the SSTO problem's optimal solution will be provided.

A. Quadrotor Dynamic Model

The quadrotor dynamic model is considered a rigid body with six degrees of freedom (DoF), including translational movements described by 3-D positions (x, y, z) measured in the inertial reference system and the rotation of the body frame with respect to the inertial frame, specified by the rotation matrix $\mathbf{R} \in \mathbb{R}^{3\times3}$.

The control inputs of the actual quadrotor are taken as the mass-normalized collective thrust $\boldsymbol{f}\,[\mathrm{m}\cdot\mathrm{s}^{-2}]$, and angular rates $\boldsymbol{\omega} = (\omega_x, \omega_y, \omega_z)\,[\mathrm{rad}\cdot\mathrm{s}^{-1}]$ about the forward, left, up axis of the body-fixed frame, respectively. Because of the low rotational inertia of a typical quadrotor [26], it is assumed that $\boldsymbol{\omega}$ can be tracked efficiently by available high-bandwidth controllers, and the collective thrust $\boldsymbol{f}$ can be governed instantaneously.

Following [27], one has equations modelling the quadrotor dynamics:

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = \mathbf{R}\begin{bmatrix} 0 \\ 0 \\ \boldsymbol{f} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ -\mathrm{g} \end{bmatrix}, \tag{1}$$

$$\dot{\mathbf{R}} = \mathbf{R}\begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix}, \tag{2}$$

where $\dot{\square}$, from now on, denotes differentiation with respect to time. g is the gravitational acceleration expressed in the inertial frame. The permissible inputs are subjected to constraints:

$$0 \le f_{\min} \le \boldsymbol{f} \le f_{\max}, \tag{3}$$

$$\|\boldsymbol{\omega}\| \le \omega_{\max}. \tag{4}$$

To simplify the 3-D trajectory generation problem, the system described by Eq. (1), (2), (3) and (4) will be decoupled into independent DoFs, generating trajectories for each axis separately. It is shown in [15] that the constraint on jerk induces an upper bound on the product of the system inputs:

$$\boldsymbol{f}^2\left\|\omega_x^2 + \omega_y^2\right\| \le \left\|\dddot{x}^2 + \dddot{y}^2 + \dddot{z}^2\right\|, \tag{5}$$

combining with ω_z being explicitly governed, and thrust limits f_{max}, f_{min} are normally given, the constraints (4) can be interpreted to equivalent kinematic constraints on jerk for the decoupled system:

$$\|\dddot{x}\| \le \dddot{x}_{\max}, \quad \|\dddot{y}\| \le \dddot{y}_{\max}, \quad \|\dddot{z}\| \le \dddot{z}_{\max}, \tag{6}$$

and must satisfy

$$\boldsymbol{f}_{\max}^2\omega_{\max}^2 = \dddot{x}_{\max}^2 + \dddot{y}_{\max}^2 + \dddot{z}_{\max}^2. \tag{7}$$

Without loss of generality, the jerk boundary is chosen to be equal for the three DoFs:

$$\dddot{x}_{\max} = \dddot{y}_{\max} = \dddot{z}_{\max} = \frac{\boldsymbol{f}_{\max}\omega_{\max}}{\sqrt{3}}. \tag{8}$$

Per-axis acceleration constraints are

$$\|\ddot{x}\| \le \ddot{x}_{\max}, \quad \|\ddot{y}\| \le \ddot{y}_{\max}, \quad \ddot{z}_{\min} \le \ddot{z} \le \ddot{z}_{\max}. \tag{9}$$

These constraints can also be equivalently defined, satisfying

$$\begin{aligned} \ddot{x}_{\max} = \ddot{y}_{\max} &= f_{\max}\sin\left(\arccos\frac{\mathrm{g}}{f_{\max}}\right) \\ \ddot{z}_{\max} &= f_{\max} - \mathrm{g} \\ \ddot{z}_{\min} &= f_{\min} - \mathrm{g}, \end{aligned} \tag{10}$$

where $\arccos\frac{\mathrm{g}}{f_{\max}}$ is the maximum lean angle that the quadrotor can assert without decelerating vertically. Thrust limits also yield an upper bound $\ddot{z}_{\max}$ and lower bound $\ddot{z}_{\min}$ on the vertical acceleration. Thus, via (10), acceleration constraints (9) and thrust constraints (3) are interchangeable. Similarly, the body rate limit $\omega_{\max}$ can also be equivalently interpreted from jerk limits through (8), given thrust limits f_{max}, f_{min}. Conclusively, kinematic constraints on acceleration (9) and jerk (6) of the decoupled system induce deterministic constraints on the actual inputs of collective thrust (3) and body rate (4). In other words, the planning 3-D trajectory of the proposed decoupled system will ensure input feasibility for the original system dynamics (1) and (2). This proposed trajectory generation will be investigated in the next subsection.

B. Time-optimal Trajectory Generation

The 3-D trajectory of the decoupled system can be modelled by a time-dependent triple integrator accepting jerk $\boldsymbol{j}(t) = [\dddot{x}, \dddot{y}, \dddot{z}]^\top \in \mathbb{R}^3$ as the only control input. Let $\boldsymbol{p}(t) = [x, y, z]^\top$, $\boldsymbol{v}(t) = [\dot{x}, \dot{y}, \dot{z}]^\top$, $\boldsymbol{a}(t) = [\ddot{x}, \ddot{y}, \ddot{z}]^\top \in \mathbb{R}^3$ define the position, velocity, and acceleration state vectors of the robot's centre of mass, respectively, measured from a fixed point in the inertial frame. The following system dynamics hold true:

$$\dot{\boldsymbol{p}}(t) = \boldsymbol{v}(t), \quad \dot{\boldsymbol{v}}(t) = \boldsymbol{a}(t), \quad \dot{\boldsymbol{a}}(t) = \boldsymbol{j}(t), \tag{11}$$

the initial and final conditions are defined as follows:

$$\boldsymbol{p}(0) = p_i, \quad \boldsymbol{v}(0) = v_i, \quad \boldsymbol{a}(0) = a_i, \tag{12}$$
$$\boldsymbol{p}(T) = p_f, \quad \boldsymbol{v}(T) = v_f, \quad \boldsymbol{a}(T) = a_f, \tag{13}$$

the jerk input constraints (6), the state constraints on acceleration (9) and additional velocity constraints are defined by

$$\|\dot{x}\| \le \dot{x}_{\max}, \quad \|\dot{y}\| \le \dot{y}_{\max}, \quad \|\dot{z}\| \le \dot{z}_{\max}, \tag{14}$$

where $p_i = [x_i, y_i, z_i]^\top$, $v_i = [\dot{x}_i, \dot{y}_i, \dot{z}_i]^\top$, $a_i = [\ddot{x}_i, \ddot{y}_i, \ddot{z}_i]^\top$, $p_f = [x_f, y_f, z_f]^\top$, $v_f = [\dot{x}_f, \dot{y}_f, \dot{z}_f]^\top$, $a_f = [\ddot{x}_f, \ddot{y}_f, \ddot{z}_f]^\top \in \mathbb{R}^3$ are given constant vectors, and $T \in \mathbb{R}^+$ denotes the trajectory duration. The velocity constraints are applied herein to limit the unwanted speed-dependent effects on quadrotors, such as aerodynamic drag and rotor damping [22], [23]. Then, the SSTO trajectory generation problem can be stated as to find the control input $\boldsymbol{j}(t)$ minimizing the trajectory duration T

$$\boldsymbol{j}(t)^* = \arg\min_{\boldsymbol{j}(t)} \int_0^T dt. \tag{15}$$

Given the jerk input and acceleration and velocity state constraints, to solve the problem (15) it is proposed to exploit the algorithm in [19], [25], which adopts a bang-zero-bang jerk control to minimize the time and solve the optimal solution as a piecewise polynomial trajectory. Though the algorithm is intuitive, mathematical proof for its characterization of the optimal solution is not clear in the literature. Hence, we will provide the optimality analysis in the next subsection.

C. Optimality Analysis

The necessary optimality conditions for each DoF of the problem (15) subject to the dynamics (11) with state constraints (14) (9), input constraints (6), and given initial (12) and terminal state (13) will now be derived independently by using Pontryagin's minimum principle [28]. Since the three DoFs are treated equally and separately, the solution for an arbitrary axis $\boldsymbol{p}(t)[r] \, \forall \, r \in \{0, 1, 2\}$ will now be presented for brevity. To simplify the notation, the time-dependence script (t) will be dropped where convenient. Let $\boldsymbol{s} = (\boldsymbol{p}[r], \boldsymbol{v}[r], \boldsymbol{a}[r])$ be the state vector, and $\boldsymbol{u} = \boldsymbol{j}[r]$ be the input control. The dynamics of $\boldsymbol{s}$ will be a single DoF of the third-order controlled system (11), expressed by

$$\frac{\mathrm{d}(\boldsymbol{p}[r])}{\mathrm{d}t} = \boldsymbol{v}[r], \quad \frac{\mathrm{d}(\boldsymbol{v}[r])}{\mathrm{d}t} = \boldsymbol{a}[r], \quad \frac{\mathrm{d}(\boldsymbol{a}[r])}{\mathrm{d}t} = \boldsymbol{j}[r]. \tag{16}$$

Problem (15), considered in only one axis, will become

$$\boldsymbol{j}[r](t)^* = \arg\min_{\boldsymbol{j}[r](t)} \int_0^T F(\boldsymbol{s}, \boldsymbol{u}, t) dt, \tag{17}$$

where $F(\boldsymbol{s}, \boldsymbol{u}, t) = 1$, subject to the initial and terminal state conditions

$$\boldsymbol{p}[r](0) = p_i[r], \quad \boldsymbol{v}[r](0) = v_i[r], \quad \boldsymbol{a}[r](0) = a_i[r], \tag{18}$$
$$\boldsymbol{p}[r](T) = p_f[r], \quad \boldsymbol{v}[r](T) = v_f[r], \quad \boldsymbol{a}[r](T) = a_f[r], \tag{19}$$

the state and input constraints

$$h_1(\boldsymbol{v}[r]) = \|\boldsymbol{v}[r]\| - v_{\max}[r] \le 0, \tag{20}$$
$$h_2(\boldsymbol{a}[r]) = \|\boldsymbol{a}[r]\| - a_{\max}[r] \le 0, \tag{21}$$
$$g(\boldsymbol{j}[r]) = \|\boldsymbol{j}[r]\| - j_{\max}[r] \le 0, \tag{22}$$

where $v_{\max} = [\dot{x}_{\max}, \dot{y}_{\max}, \dot{z}_{\max}]$, $a_{\max} = [\ddot{x}_{\max}, \ddot{y}_{\max}, \ddot{z}_{\max}]$, and $j_{\max} = [\dddot{x}_{\max}, \dddot{y}_{\max}, \dddot{z}_{\max}]$, $h_1(\boldsymbol{v}[r]), h_2(\boldsymbol{a}[r])$ are functions of pure state constraints, and $g(\boldsymbol{j}[r])$ is the function of pure input constraints. While asymmetrical lower and upper bounds of acceleration constraints are supported, a symmetrical case is presented herein, i.e., $\ddot{z}_{\min} = -\ddot{z}_{\max}$, for mathematical simplicity. The state and input constraints are considered effectively by the *direct adjoining approach* [29], which defines the Hamiltonian H

$$H(\boldsymbol{s}, \boldsymbol{u}, \lambda) = \lambda_1 \boldsymbol{v}[r] + \lambda_2 \boldsymbol{a}[r] + \lambda_3 \boldsymbol{j}[r] + F(\boldsymbol{s}, \boldsymbol{u}, t), \tag{23}$$

where $\lambda = (\lambda_1, \lambda_2, \lambda_3) \in \mathbb{R}^3$ is the adjoint variable vector. The optimal control $\boldsymbol{j}[r]^*(t)$ is the control input that minimizes the Hamiltonian function

$$\begin{aligned} \boldsymbol{j}[r]^* &= \arg\min_{\boldsymbol{j}[r]} H(\boldsymbol{p}[r], \boldsymbol{v}[r], \boldsymbol{a}[r], \boldsymbol{j}[r], \lambda) \\ &= \arg\min_{\boldsymbol{j}[r]} \lambda_3 \boldsymbol{j}[r]. \end{aligned} \tag{24}$$

Applying constraints $\|\boldsymbol{j}[r]\| \le j_{\max}[r]$ obtained from (22) to the problem (24), one obtains the optimal control input as the following switching function

$$\boldsymbol{j}[r]^* = \begin{cases} \bullet & \text{if } \lambda_3 = 0 \\ -j_{\max}[r] & \text{if } \lambda_3 > 0 \\ j_{\max}[r] & \text{if } \lambda_3 < 0 \end{cases}, \tag{25}$$

where the state $\bullet$ denotes the undetermined values. On the other hand, one also has the Lagrangian L defined by

$$L = H + \mu g(\boldsymbol{j}[r]) + \eta_1 h_1(\boldsymbol{v}[r]) + \eta_2 h_2(\boldsymbol{a}[r]), \tag{26}$$

where $\eta = (\eta_1, \eta_2) \in \mathbb{R}^2$ is the state constraint multiplier vector, $\mu \in \mathbb{R}$ is the input constraint multiplier. They must satisfy

$$\begin{aligned} &\eta, \mu \geq 0, \\ &\eta_1 = 0, \text{ if } h_1(\boldsymbol{v}[r]) < 0, \quad \eta_2 = 0, \text{ if } h_2(\boldsymbol{a}[r]) < 0, \end{aligned} \tag{27}$$

and λ is calculated by

$$\begin{aligned} \dot{\lambda} &= -\nabla_{\boldsymbol{s}} L(\boldsymbol{s}, \boldsymbol{u}, \lambda, \eta, \mu) \\ &\Rightarrow \begin{cases} \dot{\lambda}_1 = 0 \\ \dot{\lambda}_2 = \pm\eta_1 - \lambda_1 \\ \dot{\lambda}_3 = \pm\eta_2 - \lambda_2 \end{cases} . \end{aligned} \tag{28}$$

Concerning state constraints, the solution trajectory $\boldsymbol{\mathcal{T}}$ can be completely decomposed into *interior sections* $\mathcal{S}_i$ and *boundary sections* $\mathcal{S}_b$. A subinterval $(\tau_1, \tau_2) \subset [0, T]$ with $\tau_1 < \tau_2$ is an $\mathcal{S}_i$ if $\eta_1 = \eta_2 = 0 \,\forall t \in [\tau_1, \tau_2]$. An interval $[\tau_1, \tau_2]$ with $\tau_1 < \tau_2$ is an $\mathcal{S}_b$ if $h_1(\boldsymbol{v}[r]) = 0$ or $h_2(\boldsymbol{a}[r]) = 0 \,\forall t \in [\tau_1, \tau_2]$. An *entry point* is an instant τ_1 if at which ends an $\mathcal{S}_i$ and starts an $\mathcal{S}_b$. Correspondingly, an *exit point* is an instant τ_2 if at which ends an $\mathcal{S}_b$ and starts an $\mathcal{S}_i$. If the trajectory just touches the boundary at time τ, i.e., $h_1(\boldsymbol{v}[r](\tau), \tau) = 0$ or $h_2(\boldsymbol{a}[r](\tau), \tau) = 0$ and if an $\mathcal{S}_i$ ends right before and another $\mathcal{S}_i$ starts right after τ, then τ is called a *contact point*. Collectively, entry, exit, and contact points are referred to as *junction points*.

It is shown for the above-formulated problem that the switch function λ_3 is continuous and $\lambda_3 = 0$ during boundary sections from [29, Proposition 4.5], i.e.,

$$\mathcal{S}_b \Rightarrow \lambda_3 = 0. \tag{29}$$

During an $\mathcal{S}_i$, $\eta_1 = \eta_2 = 0$, thus, (28) gives $\dddot{\lambda}_3 = \dot{\lambda}_1 = 0$ which means λ_3 is a second-order polynomial in time and $\lambda_3(t) = 0$ has maximum two roots, as a result. Because λ_3 is continuous and is constant at zero during an $\mathcal{S}_b$, these roots must be junction points and lie at either end of an $\mathcal{S}_i$. Being an subinterval, $\mathcal{S}_i$ does not include those ends and therefore

$$\lambda_3(t) \neq 0 \Rightarrow \mathcal{S}_i. \tag{30}$$

Since some $\mathcal{S}_i$ and $\mathcal{S}_b$ fully constitute $\boldsymbol{\mathcal{T}}$ by definition, from (30) one can derive

$$\lambda_3 = 0 \Rightarrow \mathcal{S}_b. \tag{31}$$

Combining (29) and (31) one has

$$\lambda_3 = 0 \Leftrightarrow \mathcal{S}_b. \tag{32}$$

Then, (32) and (30) result in

$$\lambda_3 \neq 0 \Leftrightarrow \mathcal{S}_i. \tag{33}$$

Plugging conditions (32) and (33) to (25) yields

$$\boldsymbol{j}[r]^* = \begin{cases} \bullet & \text{during } \mathcal{S}_b \\ \pm j_{\max}[r] & \text{during } \mathcal{S}_i \end{cases} . \tag{34}$$

Now the optimal control law during boundary sections will be solved.

Lemma 1. $\boldsymbol{j}[r]^* = 0$ *during* $\mathcal{S}_b$.

Proof. Throughout a boundary section $\mathcal{S}_b$,

1) if $h_2(\boldsymbol{a}[r]) = 0$, i.e., $\boldsymbol{a}[r]$ is constant at either $a_{\max}[r]$ or $-a_{\max}[r]$, the control law as the time derivative of $\boldsymbol{a}[r]$ will be zero $\boldsymbol{j}[r]^* = 0$.
2) if $h_1(\boldsymbol{v}[r]) = 0$, i.e., $\boldsymbol{v}[r]$ is constant at either $v_{\max}[r]$ or $-v_{\max}[r]$, one has $\boldsymbol{a}[r] = 0$ and the optimal control law $\boldsymbol{j}[r]^* = 0$. Intuitively speaking, when velocity $\boldsymbol{v}[r]$ is constant at any value, acceleration $\boldsymbol{a}[r]$ must be zero and jerk $\boldsymbol{j}[r]$ must also be zero as a consequence.

It can also be seen from the above inference that constraints (20) and (21) cannot be active concurrently, i.e., $\{h_1(\boldsymbol{v}[r]) = 0\} \cap \{h_2(\boldsymbol{a}[r]) = 0\} = \emptyset$. □

Theorem 1. *The optimal control law is bang-zero-bang jerk input, i.e.,* $\boldsymbol{j}[r]^* \in \{-j_{\max}[r], 0, j_{\max}[r]\}$.

Proof. Combining **Lemma 1** and (34), one obtains the complete time optimal control law $\boldsymbol{j}[r]^*$ as

$$\boldsymbol{j}[r]^* = \begin{cases} 0 & \text{during } \mathcal{S}_b \\ \pm j_{\max}[r] & \text{during } \mathcal{S}_i \end{cases} ,$$

which yields a bang-zero-bang control policy. □

Assuming the result of **Theorem 1** is given, it is proved in [25] that the time-optimal trajectory would follow a template consisting of up to seven constant-jerk segments: a maximum of two acceleration limits and one velocity limit and up to four optional interior sections before, after and in between the three boundary sections. Since the control input jerk is piecewise-constant, each trajectory's segment has time-parameterized linear acceleration, quadratic velocity, and position as a cubic polynomial as follows $\boldsymbol{\mathcal{T}}(t) = [\boldsymbol{p}(t), \boldsymbol{v}(t), \boldsymbol{a}(t), \boldsymbol{j}(t)]$.

III. COLLISION-FREE TIME-OPTIMAL TRAJECTORY PLANNING

This section provides settings and assumptions given for the planning problem, paving a platform for a sampling-based planner to be constructed afterwards.

A. Planning Architecture

The general concern of this project considers a quadrotor flying through a completely unknown and static obstacle-dense setting from a starting point to a goal position without access to the map of the environment. This work does not particularly consider dynamic obstacles as a tradeoff to explore the maximized safe flight speed. Integrated sensing comprises only a limited-range and confined FoV depth camera. It is assumed that the vehicle has access to its real-time localization by using external positioning systems such as Global Navigation Satellite System (GNSS) or other indoor positioning systems, and the goal position is given *a priori* to the vehicle. The proposed planner aims to navigate

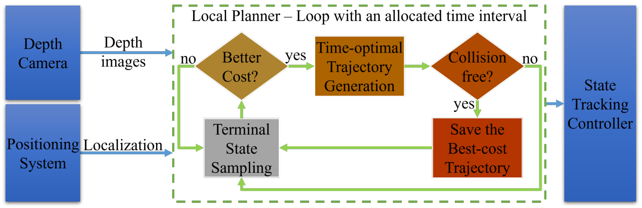
Depth Camera
Depth images
Positioning System
Localization
Local Planner – Loop with an allocated time interval
Better Cost?
no
yes
Time-optimal Trajectory Generation
Collision free?
no
yes
Terminal State Sampling
Save the Best-cost Trajectory
State Tracking Controller

Fig. 1: Planning system architecture.

the robot to travel through obstacles to reach the destination as quickly as possible.

Since perception and following trajectory planning can only be done in real-time because of unknown environments, the planning system is designed to work in a receding horizon manner, as shown in the architecture outlined in Fig 1. Specifically, the local planner processes only the latest depth image $\mathcal{D}_t$ and the vehicle's current state (p_i, v_i, a_i) estimated from external positioning systems' localization to periodically plan a trajectory $\boldsymbol{\mathcal{T}}(t)$ in a limited time interval T_a.

In order to complete the planning-control system, the real-time actual inputs, including the collective thrust $\boldsymbol{f}(t)$ and the body angular rate $\boldsymbol{\omega}(t)$ need to be recovered, given the planned trajectory $\boldsymbol{\mathcal{T}}(t)$. Differential flatness property of a quadrotor [30] allows its full state and input can be expressed as algebraic functions of the four flat outputs $[x, y, z, \psi]$ and their finite-order time derivatives, where ψ is the heading angle. That is, $\boldsymbol{f}(t)$ and $\boldsymbol{\omega}(t)$ can be retrieved from the group of the time-parameterized polynomial state $\boldsymbol{\mathcal{T}}(t) = [\boldsymbol{p}(t), \boldsymbol{v}(t), \boldsymbol{a}(t), \boldsymbol{j}(t)]$ and the time-dependent reference heading $\boldsymbol{\psi}(t)$. For this paper's purpose, $\boldsymbol{\psi}(t)$ will be computed explicitly. Therefore, to establish the connection between the chosen trajectory representation and the true control inputs, our proposed planning system employs the differential flatness-based method described in [31] as the state tracking controller. To ensure the continuity of the state tracking, it will be run concurrently with the trajectory planning, and the latest planned trajectory will be stored as a buffer of the reference state. The next planning sequence will start from the current reference state of the buffer. Because the maximum velocity plateau is likely in the middle of an SSTO trajectory [25], the next planning loop will be triggered once t exceeds half of the current trajectory duration, creating a continuously high-speed sequence of trajectories.

The 3-D trajectory representation $\boldsymbol{\mathcal{T}}(t)$ given in section II allows the online generation of many motion primitives that can be searched for to select the best one $\boldsymbol{\mathcal{T}}^*(t) = [\boldsymbol{p}^*(t), \boldsymbol{v}^*(t), \boldsymbol{a}^*(t), \boldsymbol{j}^*(t)]$ based on some criteria. Then, the formal problem becomes finding a locally optimized collision-free trajectory and reference heading $(\boldsymbol{\mathcal{T}}^*(t), \boldsymbol{\psi}(t))$ given $(\mathcal{D}_t, p_i, v_i, a_i)$ for each planning cycle that enables the robot to fly safely and reach the goal as quickly as possible. The following subsections will discuss the remaining details of the local planner's design and algorithms.

B. Sampling-based Planning Algorithm

As sketched inside the dashed box of Fig. 1, the local planner is designed to run as a loop of periodical steps, including terminal state sampling, cost evaluation, time-optimal trajectory generation and collision checking.

The first step is terminal state sampling for trajectory generation. A trajectory $\boldsymbol{\mathcal{T}}(t)$ is characterized by the initial state (12), the terminal state (13), input constraints (6) and kinematic state constraints (9), (14). The constraints (6), (9), and (14) are defined *a priori* depending on the application's dynamics. While the initial state (p_i, v_i, a_i) is determined at the beginning of the planning cycle, the terminal state (p_f, v_f, a_f) can be sampled freely in the state space. Because of the unknown environment, trajectories are sampled as stopping maneuvers, i.e., $v_f = a_f = [0, 0, 0]$ to ensure safety. Thus, the terminal position $p_f = [x_f, y_f, z_f]$ is the only remaining quantity to be sampled, constituting only 3-D sample space. This sampling problem is one dimension simpler than other time-dependent state-to-state motion primitive generators such as [15], where a 3-D terminal position plus a duration must be specified before generation.

The cost function of a trajectory $\mathcal{J}(\boldsymbol{\mathcal{T}}(t))$ is proposed to prioritize movement toward the goal, specifically minimizing the negative length of the trajectory projected in the goal direction

$$\mathcal{J}(\boldsymbol{\mathcal{T}}(t)) = \frac{\mathbf{G} - p_i}{\|\mathbf{G} - p_i\|} \cdot (p_i - p_f) = \mathcal{J}(p_f) \in \mathbb{R}, \tag{35}$$

where $\mathbf{G} \in \mathbb{R}^3$ is the goal position expressed in the camera-fixed frame. Computing this cost function and evaluating its value for each sampled trajectory is the second step. Eq. (35) shows that the cost value is independent of the corresponding trajectory generation. Therefore, the full trajectory representation $\mathcal{T}(t)$ needed for collision checking will only be generated if the sampled p_f has a better cost than the last saved collision-free trajectory. Then, the generated trajectory will be checked for collision using the memoryless algorithm presented in [17], which utilizes only the latest depth image. The checked trajectory will then be stored for the next loop evaluation if it is collision-free. Otherwise, it will be rejected, and the next loop starts immediately. The loop will be repeated until a predefined planning time interval is exceeded. The local planner then outputs the best cost trajectory $\boldsymbol{\mathcal{T}}^*(t)$ or reports failure if no collision-free trajectory is found. The reference yaw $\boldsymbol{\psi}(t)$ is set to the direction of the current path vector specified by $\overrightarrow{\boldsymbol{p}^*(t)}$ to prioritise the camera view in the current position. Furthermore, the depth-based steering mechanism from our previous work [17] will also be integrated into $\boldsymbol{\psi}(t)$ calculation to guarantee planning completeness over large convex obstacles.

IV. IMPLEMENTATION, RESULTS AND DISCUSSIONS

In this section, we demonstrate the effectiveness of our proposed method, **STAMINER** which was evaluated by comparison against the state-of-the-art methods in various simulation experiments and furthermore validated using an actual quadrotor in real-world flight experiments.

A. *Simulation Flights*

Simulation flight performance was evaluated regarding global finishing time, travelled distance, and success rate of trials in various simulated obstacle-dense settings. Two scenarios of convex obstacles were built on top of a flight simulation platform, including a spherical-obstacle environment and a forest, as captured in Fig. 2a and 2b. The mission

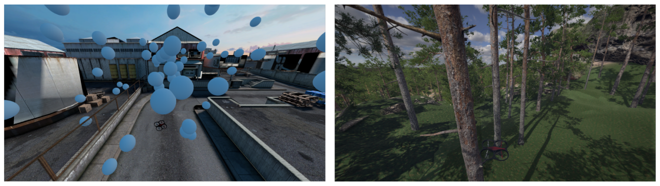
(a) Spherical obstacles (b) Forest

Fig. 2: The spherical obstacles and the forest scenarios.

is inspired by DodgeDrone Challenge 2022 [32], in which a quadrotor attempts to fly through an obstacle-dense corridor to reach a destination positioned at the end of the route. Specifically, the corridor is confined to the shape of a rectangular box coordinated at $\mathrm{xyzxyz} = (0, -5, 0, 15, 5, 10)\,\mathrm{m}$ in an inertial frame where the quadrotor starts at $(0, 0, 0)\,\mathrm{m}$ and the goal is placed at $(17, 0, 5)\,\mathrm{m}$. The first scenario features stationary spherical obstacles sizing from 0.1 to $1.2\,\mathrm{m}$ in diameter, scattered randomly inside the confined box. The second scenario is a forest area featuring random trees obstructing the path from the start position toward the goal. Figures 2a and 2b capture the actual scenes of the two settings. A flight trial is considered successful if the quadrotor reaches the destination before the cutoff time of 35 seconds and does not collide with any tree, obstacle, or the flight box's boundary.

The global planning performance considered in this flight evaluation is characterized by three aspects, including the finishing time, distance travelled and success rate. A comparison presented in [17] demonstrates that our previous memoryless planner DESS [17] outperforms the RAPPIDS's implementation [16] considering planning completeness over convex obstacles. Therefore, this evaluation considered the DESS planner a sufficient baseline to benchmark the flight performance of STAMINER. Each policy was run 1000 times in each scenario of spherical obstacles and forest on an i7-11800H laptop, resulting in 4000 flight trials. It is noted that the environment for each flight trial was randomly generated and a priori unknown to the robot.

Table I summarised success rates of flight trials by policies. Both policies had high success rates because of their planning completeness against convex obstacle settings. Specifically, STAMINER succeeded in $97.65\,\%$ trials in the spherical and $97.74\,\%$ trials in the forest, while the figures for DESS were $95.18\,\%$ and $97.33\,\%$, respectively. The small number of failed trials was mainly due to trajectory tracking errors. DESS had a few less successful trials than STAMINER because it planned slower trajectories and, thus, failed to reach the goal before the cutoff time. It should be noted that these scenarios herein were made more cluttered (the obstacle volume density is higher) compared with those described in [17], and the cutoff time is slightly shortened, from $40\,\mathrm{s}$ to $35\,\mathrm{s}$. Since there may be more trials that had not been finished before the timeout, success rates for DESS were also marginally decreased.

TABLE I: Success rate of flight trials by policies

	DESS [17]	**STAMINER (proposed)**
Spherical	95.18%	**97.65%**
Forest	97.33%	**97.74%**

It is clear in Fig. 3 that, STAMINER has a significantly shorter and steadier mean finishing time for both scenarios. While STAMINER completed the spherical in an average of $11.14\,\mathrm{s}$ and the forest in an average of $10.46\,\mathrm{s}$, DESS completed the two scenarios in averages of $26.89\,\mathrm{s}$ and $23.48\,\mathrm{s}$, respectively. This outperformance of STAMINER came from the state-to-state time optimality of its planned trajectories. On the other hand, while the finishing time standard deviations for DESS vary across scenarios, those for STAMINER remain steady. Specifically, the figures for DESS are at $2.81\,\mathrm{s}$ and $1.20\,\mathrm{s}$ in the spherical obstacle scenario and the forest one, respectively, compared with $2.03\,\mathrm{s}$ and $2.00\,\mathrm{s}$ for STAMINER in corresponding scenarios. It should be noted that the destination is at the same distance for both scenarios. That is, STAMINER can robustly complete same-length courses in roughly similar periods despite the variation of cluttered levels. This result proves the robustness and stability of our proposed method in different environmental settings.

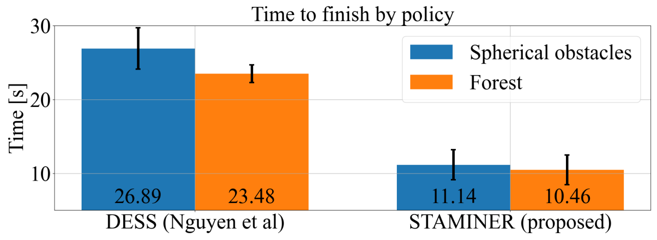

Fig. 3: Traveling time to the goal, summarising only successful trials. The comparison between the proposed planner and DESS [17]. The numbers in the figure are the mean values.

Both planners clearly cannot improve the global travelled distance result because of the completely unknown environment. It is notable in Fig. 4 that, the distance travelled results are roughly comparable across policies, being $26.89\,\mathrm{m}$ for DESS and $26.86\,\mathrm{m}$ for STAMINER in the spherical; $27.27\,\mathrm{m}$, and $25.64\,\mathrm{m}$ for the two planners in the forest, respectively. However, the result standard deviations for STAMINER are considerably larger than those of DESS regarding travelled distance. Specifically, standard deviations of STAMINER average travelled distances are $6.00\,\mathrm{m}$ in the spherical and $5.46\,\mathrm{m}$ in the forest scenario, while the

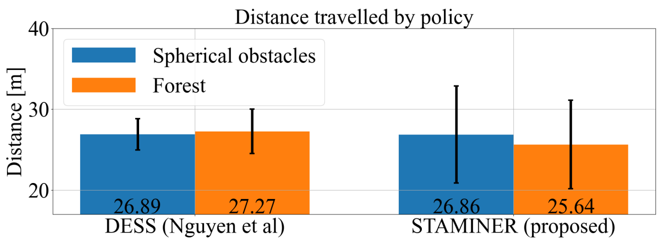

Fig. 4: Traveling distances to the goal, summarising only successful trials. The comparison between the proposed planner and DESS [17]. The numbers in the figure are the mean values.

figures for DESS are $1.93\,\mathrm{m}$ and $2.73\,\mathrm{m}$, respectively. These facts can be explained by the dynamic level of policies' trajectories. The SSTO trajectory used in STAMINER may result in a curvier path when diverging from the current direction, because of their higher dynamics compared with the minimum jerk used in DESS. Hence, planned trajectories in STAMINER may occasionally form a longer path to ensure safety. On the other hand, STAMINER generally produces a slightly shorter path in most trials compared with a typical path planned by DESS, thanks to its local state-to-state optimality. As a result, the average travelled distances per trial for the two planners are still comparable. Although travelling distance is quite similar in both policies, STAMINER is much faster, and its success rate is also higher than DESS, as depicted in Fig. 3 and Table I.

B. Real-world Flights

Real flight tests will be conducted indoors with a $250\,\mathrm{mm}$ wheelbase experimental quadrotor. It is equipped with a depth camera Intel RealSense D435i for environmental perception. Because the GNSS signal is absent inside a room, a radio-based positioning system called Nooploop LinkTrack is deployed to provide the vehicle with real-time localization data. Moreover, an Intel RealSense T265 tracking camera is equipped onboard to estimate heading attitude. The planner will be implemented in a companion computer, NVIDIA Jetson Orin NX, which feeds reference trajectories to PixRacer Pro, a flight controller running low-level state tracking controllers and 6DoF estimation. The flight controller in our system runs Ardupilot software, aided by above-ground-level altitude measurement provided by a TOFSense laser ranging sensor. Because the radio-based positioning system has the 3D accuracy of $30\,\mathrm{cm}$, the planning radius vehicle is set to $36\,\mathrm{cm}$ to ensure safety. The quadrotor system design is illustrated in Fig. 5. Our scenarios are assembled from foamed column obstacles. Admissible flight space for scenarios is confined inside a net cage sized $7\,\mathrm{m}$ by $3\,\mathrm{m}$ by $2\,\mathrm{m}$. The goal position is $1.0\,\mathrm{m}$ in height and placed $5.6\,\mathrm{m}$ horizontally in front of the starting point. The planned local trajectories, along with the current point cloud, can be monitored online via the ROS node running in the host machine, which is captured in Fig. 6. According to that, the illustrated trajectory correlated well with actual safe

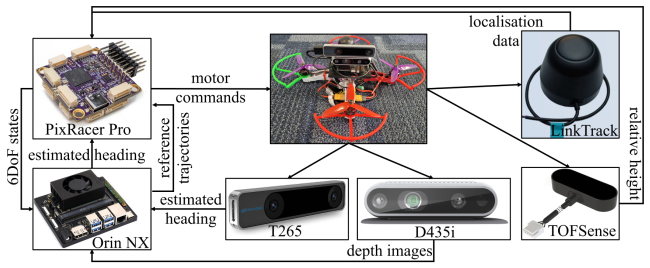

Fig. 5: The experimental quadrotor.

free space in the point cloud. The trajectories were planned

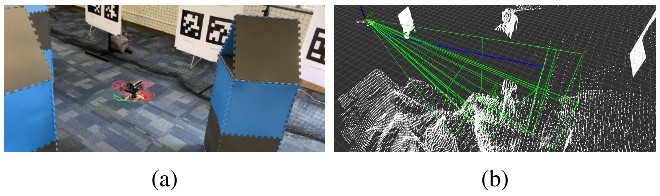

Fig. 6: (a) A real-world experiment setup. (b) A planned local trajectory in real-world scenarios, visualized by the blue curve inside the capture point cloud. Safe flight corridors are illustrated in green rectangular pyramids.

once a new depth image was received, which is at about $60\,\mathrm{Hz}$. 40 real flight trials have been conducted, and the success rate is 90%. It is noted that some failed attempts were caused by the estimation uncertainty in the positioning system. And, the vehicle never got trapped in a dead end. Peak velocity reached $1.5\,\mathrm{m/s}$ in the $5.6\,\mathrm{m}$ slalom flight course. Executed paths were logged in the flight controller's flash memory; thus, one can be visualized post-flight in Fig. 7. As shown, the vehicle navigates itself safely around obstacles to land at the goal. Those obtained results show that planned SSTO trajectories can still be smooth and safely tractable in such a cluttered and confined space. A video demonstrating how the quadrotor flew through the scenarios in the simulations and real-world experiments can be found at `https://github.com/thethaibinh/STAMINER`.

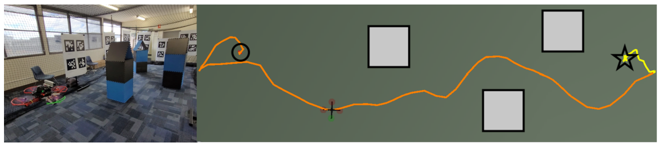

Fig. 7: A total path executed by the vehicle in a real flight trial viewed from the top. The hollow circle and star mark the start position and the goal, respectively. Solid grey squares visualize obstacles. The orange and yellow lines illustrate the navigational flying and the landing, respectively.

V. CONCLUSIONS

This paper contributes proof for the characterization of the optimal solution assumed by intuitive-based SSTO trajectory generation methods and STAMINER, an obstacle avoidance planner, on top of that. To the best of the authors' knowledge, this is the first navigational planner that can real-time plan local time-optimal and collision-free trajectories in unknown, cluttered environments without using any maps or fused occupancy structures. Results obtained from a multitude of simulation flight trials demonstrate that the proposed outperforms state-of-the-art memoryless planners in flight time. Real flight-testing results verify that STAMINER works effectively in actual hardware and real-world cluttered settings.

One limitation of STAMINER is that it does not explicitly address depth and state estimation uncertainty. Currently, uncertainty is overcome only by tuning the planning safety margin between safe flight corridors and obstacles. In future research, we will explore solutions for this limitation and bring it to more challenging settings, such as avoiding dynamic obstacles.

References

[1] T. H. Chung, V. Orekhov, and A. Maio, "Into the robotic depths: Analysis and insights from the darpa subterranean challenge," *https://doi.org/10.1146/annurev-control-062722-100728*, vol. 6, pp. 477–502, 5 2023.

[2] O. Maghazei, M. A. Lewis, and T. H. Netland, "Emerging technologies and the use case: A multi-year study of drone adoption," *Journal of Operations Management*, vol. 68, pp. 560–591, 9 2022.

[3] S. Khattak, H. Nguyen, F. Mascarich, T. Dang, and K. Alexis, "Complementary multi-modal sensor fusion for resilient robot pose estimation in subterranean environments," *2020 International Conference on Unmanned Aircraft Systems, ICUAS 2020*, pp. 1024–1029, 9 2020.

[4] S. Zhao, P. Wang, H. Zhang, Z. Fang, and S. Scherer, "Tp-tio: A robust thermal-inertial odometry with deep thermalpoint," *IEEE International Conference on Intelligent Robots and Systems*, pp. 4505–4512, 10 2020.

[5] M. Palieri, B. Morrell, A. Thakur, K. Ebadi, J. Nash, A. Chatterjee, C. Kanellakis, L. Carlone, C. Guaragnella, and A. A. Agha-Mohammadi, "Locus: A multi-sensor lidar-centric solution for high-precision odometry and 3d mapping in real-time," *IEEE Robotics and Automation Letters*, vol. 6, p. 3760, 4 2021.

[6] L. Han, F. Gao, B. Zhou, and S. Shen, "Fiesta: Fast incremental euclidean distance fields for online motion planning of aerial robots," *IEEE International Conference on Intelligent Robots and Systems*, pp. 4423–4430, 11 2019.

[7] V. Usenko, L. von Stumberg, A. Pangercic, and D. Cremers, "Real-time trajectory replanning for mavs using uniform b-splines and a 3d circular buffer," *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 215–222, 9 2017.

[8] P. Florence, J. Carter, and R. Tedrake, "Integrated perception and control at high speed: Evaluating collision avoidance maneuvers without maps," *Springer Proceedings in Advanced Robotics*, vol. 13, pp. 304–319, 2020.

[9] J. Ji, Z. Wang, Y. Wang, C. Xu, and F. Gao, "Mapless-planner: A robust and fast planning framework for aggressive autonomous flight without map fusion," *2021 IEEE International Conference on Robotics and Automation (ICRA)*, vol. 2021-May, pp. 6315–6321, 5 2021.

[10] G. Chen, D. Sun, W. Dong, X. Sheng, X. Zhu, and H. Ding, "Computationally efficient trajectory planning for high speed obstacle avoidance of a quadrotor with active sensing," *IEEE Robotics and Automation Letters*, vol. 6, pp. 3365–3372, 4 2021.

[11] N. Bucki, J. Lee, and M. W. Mueller, "Rectangular pyramid partitioning using integrated depth sensors (rappids): A fast planner for multicopter navigation," *IEEE Robotics and Automation Letters*, vol. 5, pp. 4626–4633, 2020.

[12] H. Nguyen, S. H. Fyhn, P. D. Petris, and K. Alexis, "Motion primitives-based navigation planning using deep collision prediction," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 9660–9667, 2022.

[13] T. B. Nguyen, M. Murshed, T. Choudhury, K. Keogh, G. Kahandawa Appuhamillage, and L. Nguyen, "A depth-based hybrid approach for safe flight corridor generation in memoryless planning," *Sensors*, vol. 23, no. 16, 2023.

[14] A. Loquercio, E. Kaufmann, R. Ranftl, M. Müller, V. Koltun, and D. Scaramuzza, "Learning high-speed flight in the wild," *Science Robotics*, vol. 6, 10 2021.

[15] M. W. Mueller, M. Hehn, and R. D'Andrea, "A computationally efficient motion primitive for quadrocopter trajectory generation," *IEEE Transactions on Robotics*, vol. 31, pp. 1294–1310, 12 2015.

[16] J. Lee, X. Wu, S. J. Lee, and M. W. Mueller, "Autonomous flight through cluttered outdoor environments using a memoryless planner," *2021 International Conference on Unmanned Aircraft Systems, ICUAS 2021*, pp. 1131–1138, 2021.

[17] T. B. Nguyen, L. Nguyen, T. Choudhury, K. Keogh, and M. Murshed, "Depth-based sampling and steering constraints for memoryless local planners," *Journal of Intelligent and Robotic Systems*, 10 2023. [Online]. Available: https://link.springer.com/article/10.1007/s10846-023-01971-7

[18] M. Beul and S. Behnke, "Trajectory generation with fast lidar-based 3d collision avoidance for agile mavs," *2020 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, pp. 42–48, 11 2020.

[19] ——, "Fast full state trajectory generation for multirotors," *2017 International Conference on Unmanned Aircraft Systems, ICUAS 2017*, pp. 408–416, 7 2017.

[20] M. Hehn and R. D'Andrea, "Quadrocopter trajectory generation and control," *IFAC Proceedings Volumes*, vol. 44, pp. 1485–1491, 1 2011. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S147466701643819X

[21] ——, "Real-time trajectory generation for quadrocopters," *IEEE Transactions on Robotics*, vol. 31, pp. 877–892, 8 2015. [Online]. Available: http://ieeexplore.ieee.org/document/7128399/

[22] P. Pounds, R. Mahony, and P. Corke, "Modelling and control of a quad-rotor robot," *Proceedings of Australasian Conference on Robotics and Automation*, 2006. [Online]. Available: http://www.araa.asn.au/acra/acra2006/papers/paper_5_26.pdf

[23] M. Bangura and R. Mahony, "Nonlinear dynamic modeling for high performance control of a quadrotor," *Proceedings of Australasian Conference on Robotics and Automation*, 2012.

[24] M. Beul and S. Behnke, "Analytical time-optimal trajectory generation and control for multirotors," *2016 International Conference on Unmanned Aircraft Systems (ICUAS)*, pp. 87–96, 6 2016.

[25] L. Berscheid and T. Kroeger, "Jerk-limited real-time trajectory generation with arbitrary target states," *Robotics: Science and Systems XVII*, 7 2021. [Online]. Available: http://www.roboticsproceedings.org/rss17/p015.pdf

[26] Q. Jiang, D. Mellinger, C. Kappeyne, and V. Kumar, "Analysis and synthesis of multi-rotor aerial vehicles," *Volume 6: 35th Mechanisms and Robotics Conference, Parts A and B*, vol. 6, pp. 711–720, 1 2011. [Online]. Available: https://asmedigitalcollection.asme.org/IDETC-CIE/proceedings/IDETC-CIE2011/54839/711/354319

[27] P. H. Zipfel, "Modeling and simulation of aerospace vehicle dynamics, second edition," *Modeling and Simulation of Aerospace Vehicle Dynamics, Second Edition*, 4 2007.

[28] H. Maurer, "On optimal control problems with bounded state variables and control appearing linearly," *Lecture Notes in Computer Science*, vol. 41 LNCS, pp. 555–559, 1976. [Online]. Available: https://link.springer.com/chapter/10.1007/3-540-07623-9_310

[29] R. F. Hartl, S. P. Sethi, and R. G. Vickson, "A survey of the maximum principles for optimal control problems with state constraints," *SIAM Review*, vol. 37, pp. 181–218, 6 1995. [Online]. Available: http://epubs.siam.org/doi/10.1137/1037043

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

[31] M. Faessler, A. Franchi, and D. Scaramuzza, "Differential flatness of quadrotor dynamics subject to rotor drag for accurate tracking of high-speed trajectories," *IEEE Robotics and Automation Letters*, vol. 3, pp. 620–626, 4 2018. [Online]. Available: http://ieeexplore.ieee.org/document/8118153/

[32] Y. Song, E. Kaufmann, L. Bauersfeld, A. Loquercio, and D. Scaramuzza. Icra 2022 dodgedrone challenge: Vision-based agile drone flight. [Online]. Available: https://uzh-rpg.github.io/icra2022-dodgedrone/