Minimum-Time Quadrotor Waypoint Flight in Cluttered Environments

Robert Penicka and Davide Scaramuzza

Abstract—We tackle the problem of planning a minimum-time trajectory for a quadrotor over a sequence of specified waypoints in the presence of obstacles while exploiting the full quadrotor dynamics. This problem is crucial for autonomous search and rescue and drone racing scenarios but was, so far, unaddressed by the robotics community *in its entirety* due to the challenges of minimizing time in the presence of the non-convex constraints posed by collision avoidance. Early works relied on simplified dynamics or polynomial trajectory representations that did not exploit the full actuator potential of a quadrotor and, thus, did not aim at minimizing time. We address this challenging problem by using a hierarchical, sampling-based method with an incrementally more complex quadrotor model. Our method first finds paths in different topologies to guide subsequent trajectory search for a kinodynamic point-mass model. Then, it uses an asymptotically-optimal, kinodynamic sampling-based method based on a full quadrotor model on top of the point-mass solution to find a feasible trajectory with a time-optimal objective. The proposed method is shown to outperform all related baselines in cluttered environments and is further validated in real-world flights at over 60km/h in one of the world's largest motion capture systems. We release the code open source.

Index Terms—Aerial Systems: Applications, Motion and Path Planning

Supplementary Material

I. Introduction

QUADROTORS are among the most agile and maneuverable flying machines [1]. This renders them the ideal platform for first responders after disasters, such as earthquakes, forest fires, or floods, to search for survivors as quickly as possible. To push research in this field, autonomous drone racing has emerged as a research field, with international competitions being organized, such as the Autonomous Drone Racing series at the recent IROS and NeurIPS conferences [2], [3], [4] and the AlphaPilot challenge [5], [6]. A key requirement in autonomous drone racing is to plan a trajectory that minimizes the time to fly through a sequence of gates or doorways while avoiding collisions with the environment.

The authors are with the Robotics and Perception Group, Department of Informatics, University of Zurich, and Department of Neuroinformatics, University of Zurich and ETH Zurich, Switzerland (http://rpg.ifi.uzh.ch), (email: penicka@ifi.uzh.ch).

Fig. 1: Our quadrotor executes a near time-optimal multi-waypoint trajectory planned by the proposed method in a racing track with virtual obstacles, reaching speeds over 60km/h. The picture was taken in our large indoor drone-testing arena $(30 \times 30 \times 8m^3)$ equipped with a motion capture system. The photo was augmented with virtual obstacles simulating a two-floor cluttered environment.

This makes drone racing an ideal benchmark scenario for the humanitarian use of drones in search and rescue.

Both drone racing and search and rescue motivate the tackled problem of minimum-time trajectory planning through a sequence of waypoints in cluttered environments. This problem has not been previously solved in its entirety. Existing methods either do not consider obstacles in time-optimal planning [7], or cannot use full actuation due to simplification of the quadrotor dynamics [8]. Other methods do not support multi-waypoint scenarios and time-optimal objective [9], or rely on polynomials [10], that cannot represent time-optimal trajectories due to their inherent smoothness.

The proposed method uses a hierarchical, sampling-based approach that follows three high-level steps with an incrementally more complex quadrotor model. First, topological paths between the individual waypoints are found using a variant of the Probabilistic Roadmap (PRM) [11]. Paths with different homotopy classes (e.g., going from different sides around an obstacle) are found in the roadmap using a combination of iterative shortest path search and roadmap pruning. In the second step, we use the topological paths to guide trajectory planning for 3D kinodynamic point-mass model. A trajectory primitive between two waypoints of the point-mass with limited acceleration is found using a gradient descent method. A dynamic programming graph search algorithm is then employed to find a minimum-time collision-free trajectory for the point-mass by searching over the topological paths in different homotopy classes. In the final step, a trajectory exploiting the full quadro-

tor model is planned using a kinodynamic sampling-based Stable Sparse RRT (SST) [12] algorithm, extended for time-optimal objective and multi-waypoint planning. The sampling is guided using the point-mass trajectory to effectively explore the quadrotor input and state spaces. The proposed hierarchical method differs from existing approaches by considering not only the correct actuation limits and the full quadrotor model but also by planning in presence of obstacles with a minimum-time objective.

To the best of our knowledge, this is the first multi-waypoint quadrotor planner for cluttered environments that focuses solely on the minimum-time objective. The contributions of this paper are considered as follows. We propose a new topological path search to find distinct paths through multiple waypoints. We extend the point-mass trajectory search [5] to use acceleration norm constraints and account for gravity to better approximate the quadrotor translation dynamics. Finally, we extend the SST [12] for multi-waypoint scenarios with minimum-time objective, and for sampling guided by the point-mass trajectory. The proposed hierarchical method is shown to outperform all related baselines in cluttered environments, and is further validated in real-world flights at over 60km/h in one of the world's largest motion capture systems. The source code is released as an open-source package.

II. RELATED WORK

The existing approaches for quadrotor trajectory planning can be mostly categorized as polynomial approaches, discrete-time search-based methods, sampling-based approaches, and optimization methods.

The polynomial methods [10], [13], [6], [14] represent trajectories as continuous-time polynomials. They exploit quadrotors' differential flatness property [15] that allows planning for only four flat outputs, with their high order derivatives, to get the quadrotor states and control inputs. The polynomial trajectory planning is widely used for its computational efficiency, however, it can not minimize time by design due to the inherent smoothness of polynomials. Therefore, polynomial trajectories cannot represent the bang-bang input changes required for using full quadrotor actuation and can reach maximal motor forces only in a limited number of times by either scaling time of the found trajectory [10] or by sampling boundary conditions [14].

Other methods [16], [17], [18], [19] that also leverage the differential flatness property use B-spline representation for trajectory optimization. These methods jointly optimize smoothness, dynamic feasibility, collision cost, and also safety [18] or visual tracking objectives [19]. The computational efficiency of the methods is suitable for online replanning, however, they can not be used for minimum-time flight due to the different objectives. Furthermore, using full actuation of the quadrotor is limited as the dynamic feasibility is enforced as a soft constraint and mostly in a per-axis form. Similarly to our method, [18], [16] use topological path search to find distinct paths, however, using a different algorithm with guard nodes in PRM.

The search-based methods [20], [21] use discrete-time and discrete-state representations and convert the trajectory plan-

ning to a graph search problem. The search-based approaches optimize time up to discretization, however, they suffer from the curse of dimensionality. Furthermore, the employed per-axis acceleration limits have to be set pessimistically to plan feasible but suboptimal trajectories for the full quadrotor model. The existing search-based methods support planning only between two states, and extension to multi-waypoint scenarios would be challenging with additional search dimensions for already computationally demanding methods.

The sampling-based kinodynamic methods [12], [22] plan for a dynamic system like quadrotor by growing a tree of states where in each iteration a random state is selected and forward integrated using random inputs for a random time. Alternatively, RRT* algorithm [9] can be used for linear systems or linearized quadrotor model around hover conditions, however, this prohibits fast and aggressive flights. Other methods randomly sample polynomial primitives [23] or states of simplified quadrotor kinodynamic point-mass model [5]. All above sampling-based methods, with the exception of [5], focus on minimizing trajectory length. This requires only a slight deviation from the quadrotor hover inputs. However, the time-optimal inputs lie on the boundary of the input space, which is more challenging to sample using uniform random sampling. Furthermore, the typically much faster rotational dynamics of the quadrotor, compared to the translational dynamics, render the existing sampling-based planners with Voronoi bias [12], [22] unusable for the minimum-time objective. The Voronoi bias causes preferred expansion of states on the boundary of already explored space, which can block the planning algorithms when a state on the boundary is repeatedly selected and expanded to the unfeasible state, e.g., due to the quadrotor's rotation speed constraints. This motivated the proposed method to use the hierarchical approach to guide quadrotor input sampling using the previously found point-mass solution. Such point-mass solution has approximate information about the state and input of the full quadrotor dynamics, and can be used to effectively decrease the sampling complexity and mitigate the effects of the Voronoi bias.

Finally, a method that solves the most similar problem is the optimization-based planning for quadrotor waypoint flight [7]. The method finds time-optimal trajectories for the full quadrotor model through a sequence of waypoints, however, it does not consider environments cluttered with obstacles. This renders the method mostly unusable when additional non-convex constraints are introduced to prevent collisions. The proposed sampling-based method is able to plan minimum-time trajectories for the cluttered environments by checking collisions in all levels of hierarchical planning and by guiding the planning using already found collision-free paths from the previous level.

III. PROBLEM STATEMENT

We formulate the tackled minimum-time motion planning problem through a sequence of waypoints in a cluttered environment using the classical notion of the configuration space $\mathcal{C}$ [24]. Having the world $\mathcal{W} = \mathbb{R}^3$ with the obstacles $\mathcal{O} = \{\mathcal{O}_1, \ldots, \mathcal{O}_m\} \subset \mathcal{W}$, the motion planning problem is to

determine a collision free-path for a robot $\mathcal{A} \subset \mathcal{W}$ between two locations in $\mathcal{W}$ such that the path avoids $\mathcal{O}$. In this paper, multiple waypoints $P_w = \{\boldsymbol{p}_{wi}, i \in [1, \ldots, N]\}$ have to be visited with certain proximity r_{tol} in a given order, e.g. corresponding to the gate centers in the drone racing.

Let $\mathcal{A}(\boldsymbol{x}) \subset \mathcal{W}$ denote the geometry of the robot at a configuration $\boldsymbol{x}$. The robot can move in free space $\mathcal{C}_{free} = \mathcal{C} \setminus \mathcal{C}_{obs}$, where $\mathcal{C}_{obs} = \{\boldsymbol{x} \in \mathcal{C} | \delta(\mathcal{A}(\boldsymbol{x}), \mathcal{O}) \leq d_c\} \subseteq \mathcal{C}$ is a set of configurations where the robot is in collision, i.e. the shortest distance $\delta(\cdot, \cdot)$ between the robot and any obstacle is below a given threshold d_c. For the considered quadrotor, the robot's configuration $\boldsymbol{x} = \begin{bmatrix} \boldsymbol{p}, \boldsymbol{q}, \boldsymbol{v}, \boldsymbol{\omega} \end{bmatrix}^T$ consists of its position $\boldsymbol{p} \in \mathbb{R}^3$, velocity $\boldsymbol{v} \in \mathbb{R}^3$, unit quaternion rotation $\boldsymbol{q} \in \mathbb{SO}(3)$, and body rates $\boldsymbol{\omega} \in \mathbb{R}^3$. The dynamic equations with total collective thrust $\boldsymbol{f}_T$ and body torque $\boldsymbol{\tau}$ inputs are

$$
\begin{aligned}
\dot{\boldsymbol{p}} &= \boldsymbol{v} & \dot{\boldsymbol{q}} &= \frac{1}{2}\boldsymbol{q} \odot \begin{bmatrix} 0 \\ \boldsymbol{\omega} \end{bmatrix} \\
\dot{\boldsymbol{v}} &= \frac{1}{m}R(\boldsymbol{q})\boldsymbol{f}_T + \boldsymbol{g} & \dot{\boldsymbol{\omega}} &= \boldsymbol{J}^{-1}(\boldsymbol{\tau} - \boldsymbol{\omega} \times \boldsymbol{J}\boldsymbol{\omega}),
\end{aligned}
\tag{1}
$$

where $\odot$ denotes quaternion multiplication, $R(\boldsymbol{q})$ quaternion rotation, m quadrotor mass, $\boldsymbol{J}$ its inertia, and $\boldsymbol{g}$ is gravity.

However, the real quadrotor inputs are single rotor thrusts $\begin{bmatrix} f_1, f_2, f_3, f_4 \end{bmatrix}$ which are used to calculate $\boldsymbol{f}_T$ and $\boldsymbol{\tau}$ as

$$
\boldsymbol{f}_T = \begin{bmatrix} 0 \\ 0 \\ \sum f_i \end{bmatrix}, \ \boldsymbol{\tau}_b = \begin{bmatrix} l/\sqrt{2}(f_1 - f_2 - f_3 + f_4) \\ l/\sqrt{2}(-f_1 - f_2 + f_3 + f_4) \\ \kappa(f_1 - f_2 + f_3 - f_4) \end{bmatrix}, \tag{2}
$$

using torque constant κ and arm length l. The single rotor thrusts are further constrained (3) by minimal f_{min} and maximal f_{max} values. The body rates are limited (4) by a per-axis maximal allowed value ω_{max}.

$$
f_{min} \leq f_i \leq f_{max}, \text{ for } i \in \{1, \ldots, 4\} \tag{3}
$$
$$
-\omega_{max} \leq \omega_i \leq \omega_{max}, \text{ for } \boldsymbol{\omega} = \begin{bmatrix} \omega_1, \omega_2, \omega_3 \end{bmatrix}^T \tag{4}
$$

The solution to the studied multi-waypoint motion planing between specified start $\boldsymbol{x}_s \in \mathcal{C}_{free}$ and end $\boldsymbol{x}_e \in \mathcal{C}_{free}$ configurations is a continuous sequence of trajectories $\tau_i : [0, 1] \to \mathcal{C}_{free}$ for $i \in \{0, \ldots, N\}$ that pass through P_w. The sequence has the given start $\tau_0(0) = \boldsymbol{x}_s$ and end $\tau_N(1) = \boldsymbol{x}_e$, the initial positions of the trajectories $\tau_i(0)_p$ have to be in waypoints' proximity $\|\tau_i(0)_p - \boldsymbol{p}_{wi}\| \leq r_{tol}$ for $i \in \{1, \ldots, N\}$, and the sequence has to be continuous $\tau_{i-1}(1) = \tau_i(0)$ for $i \in \{1, \ldots, N\}$. The objective of the problem is then to minimize the time $T = \sum_{i=0}^{N} t_i$ of reaching the end configuration, where t_i corresponds to the time duration of τ_i given (1)-(4). The whole multi-waypoint minimum-time planning problem can be summarized as

$$
\begin{aligned}
\underset{\tau_0 \ldots \tau_N}{\text{minimize}} \quad & T = \sum_{i=0}^{N} t_i \\
\text{s.t.} \quad & \tau_i \in \mathcal{C}_{free} \text{ for } i \in \{0, \ldots, N\}, \\
& \tau_0(0) = \boldsymbol{x}_s, \ \tau_N(1) = \boldsymbol{x}_e, \\
& \|\tau_i(0)_p - \boldsymbol{p}_{wi}\| \leq r_{tol} \text{ for } i \in \{1, \ldots, N\}, \\
& \tau_{i-1}(1) = \tau_i(0) \text{ for } i \in \{1, \ldots, N\}, \\
& (1), (2), (3), (4).
\end{aligned}
\tag{5}
$$

IV. METHOD

The proposed method uses a hierarchical approach that initially finds diverse topological paths with a variant of Probabilistic Roadmap (PRM) [11]. Then, the fastest trajectory for 3D point-mass model is found by searching over the topological paths. Finally, the SST method is used to find a minimum-time trajectory over multiple waypoints for the full model of the quadrotor using the point-mass solution to guide the exploration. The whole method uses Euclidean Signed Distance Field (ESDF) as a priory known map of the environment. The method is visualized in Figure 2.

A. Topological path search

The topological path search is used to find distinct paths with different homotopy classes, e.g. passing around an obstacle from a different side. This is necessary as the typically sought minimum-length path does not necessarily translate to the minimum-time trajectory for the dynamic point-mass model and further for the full quadrotor model.

The method described in Algorithm 1 plans sequentially for pair of adjacent goals in $G = [\boldsymbol{p}_s, \boldsymbol{w}_{wi} \in P_w, \boldsymbol{p}_e]$, where $\boldsymbol{p}_s$ and $\boldsymbol{p}_e$ are positions of the start and goal configurations, respectively. The PRM is used to create a roadmap (V, E) by sampling ellipsoid between two goals similarly to the Informed-RRT* [25]. The number of samples and ellipsoid major axis are iteratively increased by a constant factor if no path is found. Once at least one path can be found in the roadmap, method **FindDistinctPaths** is used to find distinct paths between the goals as described in Alg. 2.

The distinct paths are further shortened (similarly to [18]) in forward and backward pass. Afterward, paths with the same homotopy class are removed. The homotopy is approximated using *uniform visibility deformation* (UVD) [18], where trajectories $\tau_i^1 : [0, 1]$, $\tau_i^2 : [0, 1]$ belongs to same UVD class if the straight line connection $\tau_i^1(s) \to \tau_i^2(s)$ is collision free for all $s \in [0, 1]$. Finally, the paths are filtered by keeping only paths shorter than a fixed factor of the shortest one and by limiting the number of totally kept paths. Additionally, paths with the wrong angle of reaching a waypoint, i.e., the direction of flying through a gate, can be filtered out.

Algorithm 1: Topological PRM

Input: G – goals
Out : $\Pi_0, \ldots, \Pi_N$ – found topological paths

1 **foreach** $i \in 1, \ldots, \|G\| - 1$ **do**
2      $(V, E) \leftarrow \textbf{PRM}(G[i-1], G[i])$
3      $\Pi_{i-1}^d \leftarrow \textbf{findDistinctPaths}((V, P), \{G[i-1]\}, G[1])$
4      $\Pi_{i-1}^* \leftarrow \textbf{shortenPaths}(\Pi_{i-1}^d)$
5      $\Pi_{i-1}^u \leftarrow \textbf{removeEquivalentPaths}(\Pi_{i-1}^*)$
6      $\Pi_{i-1} \leftarrow \textbf{filterPaths}(\Pi_{i-1}^u)$

The distinct path search in Algorithm 2 iteratively searches for the shortest path using Dijkstra's algorithm between given start nodes S and end nodes F in the roadmap. For each path, a node v_n with smallest clearance d_n from obstacles is found. Additionally, nodes V_n that are connectable to v_n using collision-free path with length within d_n are found. Nodes V_n and v_n are then removed from the roadmap and saved among

(a) Distinct paths   (b) Shortened paths   (c) Filtered unique paths

(d) Point-mass solution   (e) Planning tree   (f) Final trajectory
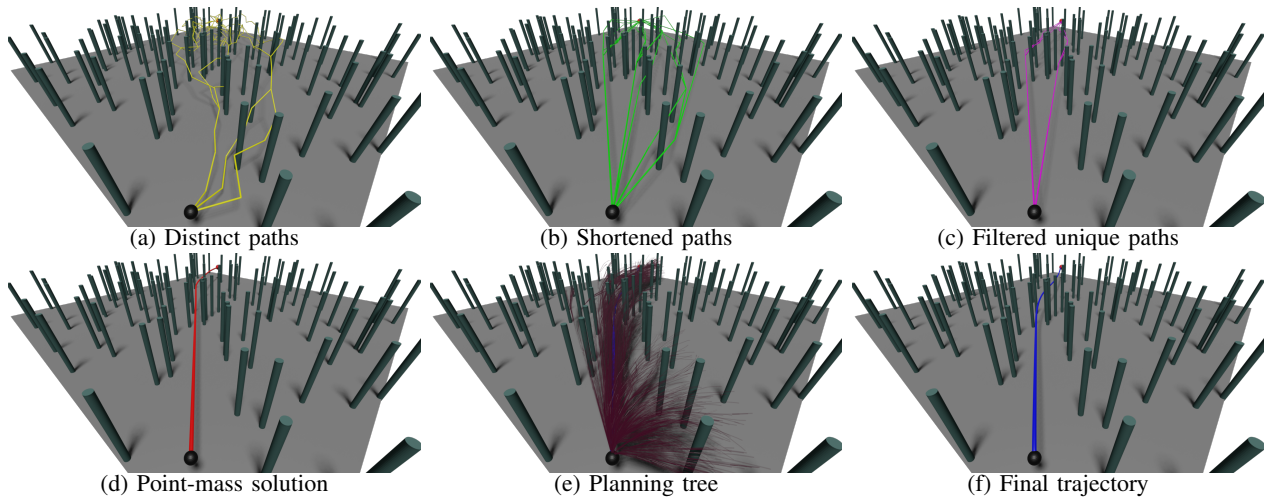
Fig. 2: Visualization of the individual stages of the hierarchical approach. Distinct topological paths are firstly found between goals (a) and are consequently shortened (b). Filtered paths (c) are then used to find the minimum-time trajectory for the point-mass model (d). Finally, the sampling-based method uses the point-mass solution to guide exploration of the solution space (e) to find the final trajectory (f). The start and goal are colored black and red, respectively.

deleted nodes V_d. The shortest path search is repeated until no new path can be found. This is either if no alternative path exists or vertices in some narrow passages were removed. To address the second case, the algorithm is recursively called from both S to V_d and from V_d to F. Finally, paths between original S and F nodes are connected using the found paths with corresponding nodes in V_d.

Algorithm 2: Distinct Path Search

Input: (V, E) – roadmap, S – start nodes, F – end nodes
Out : Π – found distinct topological paths

---

1 $V_d = \emptyset$ // set of deleted nodes
2 **do**
3      $\Pi \leftarrow \Pi \cup \leftarrow$ **findShortestPaths**$((V, E), S, E)$
4   **foreach** $\pi_s \in \Pi_s$ **do**
5         $v_n, d_n \leftarrow$ **nearestNodeToObstacle**(π_s)
6         $V_n \leftarrow \{v \in V \mid \|v_n - v\| \le d_n - d_c\}$
7         $V = V \setminus (\{v_n\} \cup V_n)$
8         $V_d \leftarrow V_d \cup \{v_n\} \cup V_n$
9 **while** $\Pi_s \neq \emptyset$
10 **if** $V_d \neq \emptyset$ **then**
11     $\Pi_b \leftarrow$ **FindDistinctPaths**$((V, P), S, V_d)$
12     $\Pi_a \leftarrow$ **FindDistinctPaths**$((V, P), V_d, E)$
13     $\Pi \leftarrow \Pi \cup$ **connectPaths**(Π_b, Π_a, V_d)

B. Point-mass trajectory planning

The found topological paths are further used to plan a trajectory for 3D point-mass model that approximates the translation part of the quadrotor dynamics. A minimum-time motion primitive is created between any two states of the point-mass model. A dynamic programming search, that utilizes the primitive, is then used to find a collision-free point-mass trajectory between multiple waypoints. The distinct topological paths are used during the search to constraint the position of the primitives to the already found collision-free topological paths.

1) Point-mass motion primitive: We use the 3D point-mass model with state consisting of position $\boldsymbol{p}$ and velocity $\boldsymbol{v}$, and with input on acceleration $\boldsymbol{u} = \boldsymbol{a}$. The motion primitive is a trajectory from given start position $\boldsymbol{p}^s = (p_1^s, p_2^s, p_3^s)^T$ and start

velocity $\boldsymbol{v}^s = (v_1^s, v_2^s, v_3^s)^T$ to end position $\boldsymbol{p}^e = (p_1^e, p_2^e, p_3^e)^T$ and end velocity $\boldsymbol{v}^e = (v_1^e, v_2^e, v_3^e)^T$. For a single axis, without loss of generality for the x-axis, it could be shown that for dynamics $\ddot{p}_1(t) = u_1(t)$ the time-optimal control input using Pontryagin's Maximum Principle (PMP) is a bang-bang using a given maximal per-axis acceleration a_1. The optimal control has form

$$a_1^* = \begin{cases} a_1 & 0 \le t \le t_1^*, \\ -a_1 & t_1^* \le t \le T_1^*, \end{cases} \tag{6}$$

or with switched inputs of $-a_1$ followed by a_1. The minimum final time $T_1^* = T_1^*(a_1, p_1^s, v_1^s, p_1^e, v_1^e)$ can be found in closed-form as a solution to a set of four classical kinematic equations that describe position and velocity for both bang-bang parts. The only unknowns are then the t_1^*, T_1^*, and the position and velocity in t_1^*.

A similar approach was proposed in [5] where the authors used per-axis acceleration limits. In this work, the point-mass primitive is extended by constraining the acceleration norm $a_{max} = 4f_{max}/m$ over joint axes and by accounting for gravity. This corresponds to point-mass thrust acceleration $\boldsymbol{a}_t = (a_{t,1}, a_{t,2}, a_{t,3})^T$, $\|\boldsymbol{a}_t\| = a_{max}$ with body acceleration used for the per axis closed-form solutions in (6) as $\boldsymbol{a} = \boldsymbol{a}_t + \boldsymbol{g}$. Finally, to minimize the time $T^*(\boldsymbol{a}, \boldsymbol{p}^s, \boldsymbol{v}^s, \boldsymbol{p}^e, \boldsymbol{v}^e) = \max_{i=\{1,2,3\}} T_i^*(a_i, \ldots)$ of the coupled 3D point-mass primitive, we optimize over the thrust acceleration $\boldsymbol{a}_t$ as

$$\begin{aligned} \underset{\boldsymbol{a}_t}{\text{minimize}} \quad & T^*(\boldsymbol{a}, \boldsymbol{p}^s, \boldsymbol{v}^s, \boldsymbol{p}^e, \boldsymbol{v}^e) \\ \text{s.t.} \quad & \boldsymbol{a} = \boldsymbol{a}_t + \boldsymbol{g}, \|\boldsymbol{a}_t\| = a_{max}. \end{aligned} \tag{7}$$

We use a Gradient Descent (GD) on a sphere with radius a_{max} and project the gradient onto the sphere to keep maximal acceleration norm. The gradient of individual axes can be found in closed-form similarly to the solution of (6). After the GD, the primitive is calculated by stretching the times of the axes T_i^* to match time T^*, which can be done in closed-form by lowering the per-axis accelerations, and is necessary to keep the desired $\boldsymbol{p}^e$ and $\boldsymbol{v}^e$.

2) Multi-waypoint point-mass trajectory search: The point-mass motion primitive is further used to plan a multi-waypoint trajectory utilizing the previously found topological paths Π. As the positions are known, the search converts to finding velocities for a given sequence of positions, initially $G = (\boldsymbol{p}_s, \boldsymbol{p}_{w1}, \ldots, \boldsymbol{p}_{wN}, \boldsymbol{p}_e)$, that minimize the time of reaching $\boldsymbol{p}_e$. In [5] this task is solved (without considering obstacles) by randomly sampling velocities in G with consequent shortest-time search in the computed graph of motion primitives. This, however, has quadratic complexity in the number of velocity samples per position. Therefore, we opted for deterministic sampling with a small number of velocity samples, i.e., having only 27 velocity samples in a cone with $3 \times 3 \times 3$ samples corresponding to yaw, pitch, and velocity norm dimensions. Dijkstras algorithm is used to find the shortest-time path in such a graph using the point-mass motion primitive between individual samples. Afterward, the cones are refocused around the found shortest-time path, and the search is repeated until the time converges. When refocused, the cones are adjusted individually for each dimension. The samples are either re-centered if a boundary sample is used, or the distance between samples is halved if the center sample is used. This way, the number of primitive evaluations is limited while iteratively focusing the search toward the minimum-time solution.

The above described **velocitySearch** method computes point-mass trajectory over the positions in G without considering the obstacles. An additional graph search over the topological paths, described in Algorithm 3, is used to find a collision-free point-mass trajectory. It uses dynamic programming search with priority queue to always test collisions in the currently shortest-time trajectory. In case of a collision, a new position is added to G between nodes where the first collision occurred. A new sequence with an additional position is created for each possible topological path between such nodes. This way, the algorithm keeps attaching the point-mass trajectory to the collision-free topological paths until the shortest feasible path is found.

Algorithm 3: Point-mass trajectory search

Input: Π – topological paths, G – goals
Out : τ_{pm} – found point-mass trajectory

---
1   $\pi_i \leftarrow$ **velocitySearch**(G) // initial path
2   $\Pi_h \leftarrow (\pi_i, G)$ // initialize heap
3   **while** $\Pi \neq \emptyset$ **do**
4      $(\pi_b, G_b) \leftarrow$ **findAndRemoveBestTrajectory**(Π_h)
5   **if** π_b *is collision-free* **then**
6          $\tau_{pm} \leftarrow \pi_b$; break   // trajectory found
7   **else**
8          $g_b, g_a \leftarrow$ **getNodesOfFirstCollision**(π_b)
9   **for** π *between* g_b, g_a *found in* Π **do**
10             $G_{new} \leftarrow$ add goal between g_b, g_a in π to G_b
11             $\pi_{new} \leftarrow$ **velocitySearch**(G_{new})
12             $\Pi_h \leftarrow \Pi_h \cup (\pi_{new}, G_{new})$

C. Full quadrotor model sampling-based planning

This part extends the original kinodynamic sampling-based method called SST [12] to the multi-waypoint scenario guided by the point-mass trajectory. The minimum-time cost is employed instead of the easier-to-sample shortest-path objective

used in the original method. The SST is implemented as the final planning step that uses the full quadrotor dynamics and creates a feasible solution compared to the point-mass approximation. The guiding with the point-mass solution is, however, used not only to reduce the volume of sampled configuration space but is needed to avoid blocking the SST by Voronoi bias when sampling on the edge of feasible input space for the minimum-time objective.

The SST works by protecting the configuration space with a set of witnesses S_i, each containing its center configuration and a representative configuration that contains the node with the best cost within δ_s radius of the center. A new node is added to the SST tree and to active nodes $V_{a,i}$ only if it reaches unexplored space and forms a new witness, or its cost is better than the cost of its witness representative. The old representative is replaced in the latter case by the better one and moved to inactive nodes $V_{u,i}$. This way, the SST keeps only minimum-time trajectories and explores the configuration space until the desired goal is reached. We extend the original SST to multi-waypoint scenarios by considering S_i, $V_{a,i}$ and $V_{u,i}$ for all $i = 0, \ldots, \|G\| - 2$, i.e. goals in the sequence G. The planning tree starts in x_s, and in each iteration, a random goal is selected among the already reached goals. The tree is allowed to grow to the next goal if the previous goal is reached with tolerance r_{tol}. The multi-waypoint trajectory is thus planned using one tree that keeps expanding towards unreached goals until the stopping conditions are met.

The proposed guided multi-goal SST (Algorithm 4) uses the point-mass trajectory τ_{pm} to create a non-continuous trajectory primitive that has the translational dynamics part of τ_{pm}. The rotational parts are added to match the acceleration changes of the point-mass using PMP bang-bang or bang-singular-bang rotational inputs. The inputs are found by calculating quaternion errors in the body x-y plane in the changes of the point-mass accelerations. The error is used to find rotational axis and maximal angular acceleration producible given the motor constraints (3). Finally, the rotational parts along the found axis are calculated similarly to single-axis motion (6) using the maximal angular acceleration, zero initial and final angular velocities, and angular position from the error quaternions. The bang-singular-bang solution is used to satisfy the angular velocity constraints (4) and is calculated similarly to (6) in closed-form. This trajectory primitive assumes decoupled rotational and translational dynamics, however, it serves as a guiding reference for the SST that uses the full quadrotor dynamics and couples the dynamics while randomizing quadrotor inputs around the primitive.

In each iteration of the Alg. 4, a random goal index g_i is selected from the already reached goals. The **bestNearSelection** method first generates a state q_{rand} using a random state from the guiding reference between g_i and g_{i+1} with additional uniform random noise with variance $(\sigma_p^2, \sigma_q^2, \sigma_v^2, \sigma_w^2)$ for each part of the state. Selection of a node that lies within r_{tol} distance of goal g_{i-1} is randomly biased with probability p_g to focus exploration around goals. Then a node q_{sel} with smallest cost from the start within δ_{bn} radius from q_{rand} is selected for expansion. If none is found, the nearest node from V_{a,g_i} to q_{rand} is used instead.
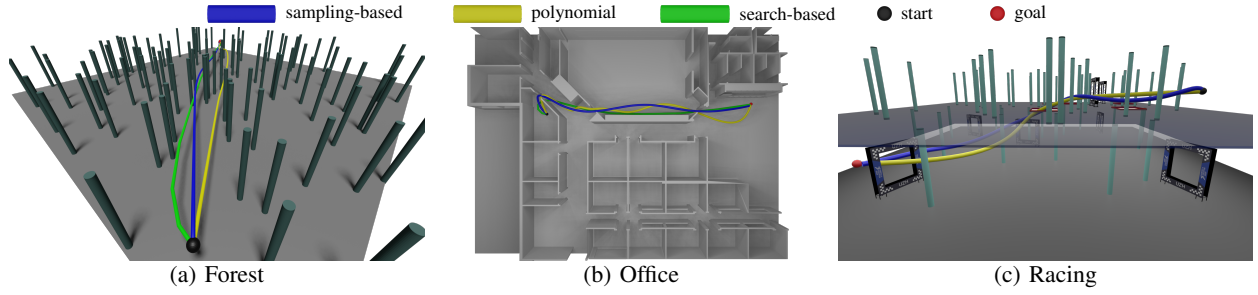
(a) Forest   (b) Office   (c) Racing

Fig. 3: Maps of the environments used for evaluating the proposed method and comparison with related algorithms. Example trajectories are shown for the proposed sampling-based planner, the polynomial method [10], and the search-based method [20].

Algorithm 4: Guided multi-goal SST

Input: τ_{pm} – point-mass trajectory, G – goals
Out : τ – found quadrotor trajectory

1   $V_{a,i} \leftarrow \emptyset, V_{u,i} \leftarrow \emptyset, S_i \leftarrow \emptyset, \forall i = 0, \ldots, \|G\| - 2$
2   $V_{a,0} \leftarrow \{x_s\}, E \leftarrow \emptyset, s_0 \leftarrow x_s, s_0.rep \leftarrow x_s, S_0 \leftarrow s_1$
3   **while** *stopping conditions not met* **do**
4      $g_i \leftarrow$ **randomGoalIndex**(G)
5      $q_{sel} \leftarrow$ **bestNearSelection**(V_{a,g_i}, δ_{bn})
6      $q_{new} \leftarrow$ **propagate**(q_{sel}, τ_{pm})
7   **if** *isPassing*(q_{new}) **then**
8   **if** r_{tol} *radius of* g_i *reached by* q_{sel}, q_{new} **then** $g_i \mathrel{+}= 1$
9   **if** *isLocalBest*$(q_{new}, S_{g_i}, \delta_s)$ **then**
10             $V_{a,g_i} \leftarrow V_{a,g_i} \cup \{q_{new}\}$
11             $E \leftarrow E \cup \{q_{sel} \rightarrow q_{new,j}\}$
12   **pruneNodes**$(q_{new}, V_{a,g_i}, V_{u,g_i})$

The state q_{sel} is further propagated to q_{new} using a 4th-order Runge-Kutta integration of the quadrotor dynamics (1) for a random time in range $[t_{min}, t_{max}]$. The motor inputs are calculated based on the guiding reference, and the nominal time of applying individual inputs is randomly scaled in the range $[s_{rmin}, s_{rmax}]$. Furthermore, the rotational parts have an additionally randomized rotation axis with variance σ^2_{qrot}. The randomized time of propagation is a standard way of exploring the solution space by SST, while the randomized scale of reference inputs and the rotation axis is used to account for the artificially added non-continuous rotational parts in the point-mass reference. The new state q_{new} is checked in the **isPassing** method to have 1) distance from the reference below a given threshold δ_{ref}, 2) time-cost from the start below a certain ratio r_{pmm} of the closest state in the reference, and 3) collision-free trajectory between q_{sel} and q_{new}. The method thus removes both colliding trajectories and expansions that are far from the point-mass reference in time or in position. If the goal g_i is reached by the propagated trajectory, the next goal is selected.

Afterwards, method **isLocalBest** finds nearest witness $s_{new} \in S_{g_i}$ to q_{new}. If the distance $\|q_{new} - s_{new}\|$ is above δ_s a new witness s_{new} centered in q_{new} is added to S_{g_i}. The method allows the algorithm to continue if the representative of s_{new} has not been set or the representative's time-cost is lower than the one of the q_{new}. In both cases, q_{new} is added among active nodes V_{a,g_i} and trajectory $q_{sel} \rightarrow q_{new}$ is added among the edges E. Finally, method **pruneNodes** sets the representative of s_{new} to q_{new} and moves the old representative from V_{a,g_i} to V_{in,g_i}. Furthermore, if the old representative is an inactive leaf node, it is removed from the tree, and the removal is repeated for its parent node recursively

until a non-leaf or active node is found. The algorithm keeps expanding the tree until the stopping conditions are met, consisting of a maximal number of iterations and a limited number of iterations without improvement.

V. RESULTS

The proposed planning method is evaluated on three environments shown in Figure 3. It is compared with related algorithms for single and multi-waypoint scenarios. Furthermore, the method is validated in real-world racing track at over 60km/h in one of the worlds largest motion capture systems ($30 \times 30 \times 8m^3$). The flight is captured in Figure 1.

All trajectories are found using C++ implementation of the method and a single-core Intel Xeon Gold 6252 CPU with $2.10\,\mathrm{GHz}$. The parameters of the considered quadrotor and of the algorithms are summarized in Table I. The stopping conditions used for the method are the maximal number of $2 \cdot 10^6$ iterations and $2 \cdot 10^5$ of iterations without improvement.

TABLE I: Quadrotor and algorithm parameters

	Variable	Value	Variable	Value
Quadrotor	m [kg]	0.85	l [m]	0.15
	f_{min} [N]	0	f_{max} [N]	7
	$\mathrm{diag}(J)$ [g m^2]	$[1, 1, 1.7]$	κ []	0.05
	w_{max} [rad s^{-1}]	15		
[10]	k_T	105000	N_{poly}	10
Method	d_{tol} [m]	0.3	d_c [m]	0.2
	δ_s []	0.5	δ_{bn} []	1.3
	(σ^2_p, σ^2_q)[m^2,rad^2]	$(1.3, 0.08)$	σ^2_{qrot}[rad^2]	0.013
	$(\sigma^2_v, \sigma^2_w)[\frac{m^2}{s^2}, \frac{rad^2}{s^2}]$	$(8.3, 8.3)$	$[s_{rmin}, s_{rmax}]$[]	[0.6,1.4]
	$[t_{min}, t_{max}]$[s]	$[0.004, 1.2]$	r_{pmm}[]	1.05
	δ_{ref}[m]	2	p_g[]	0.05

Computational time of the proposed method is mostly influenced by δ_{bn} and δ_s, that increase the complexity of nearest-neighbor search for larger δ_{bn} and increase number of created witnesses with smaller δ_s. Values of $(\sigma^2_p, \sigma^2_q, \sigma^2_v, \sigma^2_w)$ were scaled to have approximately same scale in encountered states during planning. The solution quality is mainly influenced by δ_{ref} and r_{pmm} that bind the quality of the final solution to the quality of the point-mass trajectory. Other parameters of the method were empirically tuned for the best solution quality.

A. Single goal planning

The first set of experiments is for single target scenarios where a trajectory is planned between start and goal states only. This allows comparison between all considered baseline algorithms. The first baseline algorithm is the polynomial

TABLE II: Comparison of the baseline methods and the proposed method in single target scenarios

Env.	Test case	Polynomial [10]			Search-based [20]				CPC [7]		**Ours**		
		c. time[s]	T_a[s]	T_b[s]	c. time[s]^p	T[s]^p	c. time[s]^o	T[s]^o	c. time[s]	T[s]	c. time[s]	T_a[s]	T_b[s]
Forest	0	0.41	4.67±0.63	3.86	5.82	1.80	10.39	1.60	70.23	**0.95**	14.92	1.10±0.13	0.96
	1	0.30	3.43±0.00	3.43	0.73	1.60	5.67	1.40	67.40	**0.96**	2.85	0.97±0.00	**0.96**
	2	0.30	3.74±0.93	3.22	0.73	1.40	1.65	1.40	65.49	**0.95**	8.03	0.98±0.01	0.96
	3	1.47	7.20±1.17	5.25	3.85	2.00	24.84	1.80	-	-	135.83	1.50±0.17	**1.30**
Office	0	1.36	8.64±1.03	7.34	3.15	2.80	22.41	2.60	-	-	139.19	2.38±0.28	**1.93**
	1	0.65	7.50±0.44	6.49	13.18	2.60	77.16	2.20	-	-	103.64	1.74±0.06	**1.69**
	2	0.89	9.01±0.77	6.38	11.37	2.60	39.70	2.20	-	-	155.23	2.20±0.13	**1.93**
	3	0.47	5.26±0.35	5.14	5.40	2.40	35.29	2.00	-	-	223.64	1.81±0.11	**1.58**
Racing	0	1.98	6.56±0.66	5.79	-	-	-	-	-	-	365.91	1.61±0.29	**1.34**
	1	2.06	6.21±0.88	5.13	-	-	-	-	-	-	428.02	1.63±0.15	**1.36**
	2	1.87	6.26±0.42	4.94	-	-	-	-	-	-	138.17	1.45±0.12	**1.37**
	3	1.91	5.27±0.69	4.73	-	-	-	-	-	-	604.55	2.14±0.74	**1.57**

method [10] which jointly minimizes snap and final time of reaching the goal (using time penalty k_T) with polynomials of order N_{poly}. Contrary to the original method, we employ the PRM* algorithm to find paths between individual waypoints. The second baseline is the search-based minimum-time method [20], where we limit the point-mass state to include position derivatives only up to acceleration due to computational complexity. Method [20] is evaluated for pessimistic $22.2\,\mathrm{m\,s^{-2}}$ and for optimistic (though unfeasible) $31.4\,\mathrm{m\,s^{-2}}$ maximal acceleration per axis, denoted with 'p' and 'o', respectively. The first is is derived from the maximal motor thrust in diagonal lateral motion, while the second as maximal acceleration in lateral motion along single axis. Finally, the time-optimal CPC optimization method [7] is extended to a variant for cluttered environments using position constrained by the ESDF map of the environment.

The results for the single goal planning are summarized in Table II for all three environments with four test cases (i.e., different start-goal positions) for each environment. We report the computational time of each algorithm and the time duration (i.e., solution quality) T of found trajectories. For the polynomial method and the proposed method, we show the average duration with standard deviation T_a and the best duration T_b found within 30 conducted runs per test case. The results show that the proposed method is able to find the highest quality trajectories for the majority of the test cases. The computational times of our methods, however, render it only as an offline planning method similarly to [7], [20]. The polynomial method, although fast in finding solutions, is by far the one with the highest trajectory durations. This demonstrates the inherent inability of the polynomial trajectory planning to represent the minimum-time trajectories. The search-based method is 14%-66% worse in solution quality for even the optimistic ('o') acceleration limit than the proposed sampling-based method and has comparable computational times. The comparably lower solution quality of the search-based method is caused jointly by the time discretization and the per-axis acceleration limit that restricts exploitation of the full quadrotor actuation. The search-based method is able to find solutions only for the Forest and Office scenarios with 2D planning problems. In the Racing environment, where planning in 3D is required, the method is not able to find a solution before reaching the memory limit of $32\,\mathrm{GB}$. This is caused by the curse of dimensionality of the state discretization which requires exponential growth of memory for additional searched dimensions. On the other hand, the time-optimal CPC

method is able to find a solution only for the three easiest scenarios in the Forest environment. In other environments, the CPC is not able to find a feasible solution due to the introduced collision constraints that add non-convexities that the optimization-based method can not handle. However, the comparison with the solutions found by CPC shows the very small gap in duration between time-optimal solutions and the best trajectories found by the proposed method. Furthermore, our method is faster approximately by a factor of ten compared to the CPC.

B. Multi-waypoint planning

The second set of evaluations focuses on the multi-waypoint scenarios. The scalability of the method is tested for a variable number of targets and density of the obstacles in the environment. Furthermore, the related algorithms are compared for the multi-waypoint drone-racing task in both environments with and without obstacles.

Computation time and solution quality are shown in Table III to demonstrate the scalability of the method in multi-waypoint planning. It is tested in an environment similar to the Forest 3a with an increasingly larger number of randomly placed columns and targets (NT). The number of tested columns [50, 100, 150, 200] corresponds, respectively, to [3.5%, 6.6%, 10%, 13%] portion of occupied environment.

TABLE III: Scalability evaluation with increasing number of waypoints and obstacles

NT	N. Col. 50		N. Col. 100		N. Col. 150		N. Col. 200	
	c.t.[s]	T[s]	c.t.[s]	T[s]	c.t.[s]	T[s]	c.t.[s]	T[s]
2	26.5	0.74	88.8	0.74	158.2	0.79	220.6	0.76
3	155.9	1.90	389.2	1.98	524.9	2.54	1044.2	2.79
4	248.0	2.90	421.1	3.40	491.3	4.38	-	-
5	1095.8	4.77	958.1	4.88	1125.5	6.38	-	-

The presented data shows the expected trend of higher computational times and larger trajectory durations for both higher number of targets and obstacles. For the highest number of columns and targets, the method was unable to find a solution within the limited number of iterations. This is mainly due to a high number of narrow passages that have a low probability of being sampled, especially for high-speed multi-waypoint planning.

The multi-waypoint planning is also tested in a racing environment (Fig. 3c) that features seven gates. Our method is compared with the polynomial trajectory planning and the CPC. The search-based method is not considered as it allows

only single target planning. Both trajectory durations and computational times are presented in Table IV for environments with and without obstacles.

TABLE IV: Comparison for the drone racing task

O.	Poly. [10]			CPC [7]		Ours		
	c.t.[s]	T_a[s]	T_b[s]	c.t.[s]	T[s]	c.t.[s]	T_a[s]	T_b[s]
w/o	4.34	15.33±0.00	15.33	1009.34	**6.53**	919.84	6.99±0.06	6.88
w	8.12	27.54±0.62	26.98	-	-	734.65	7.10±0.06	**7.01**

Similar to the single target scenarios, the solution quality of the polynomial trajectories is significantly lower. Comparison of the CPC and our method shows similar computational time. Our method is able to find solutions with only 5.3% higher trajectory duration than the optimal CPC. Most importantly, unlike the CPC, our method can find the trajectory for the cluttered environment while the solution quality is comparable to the one without obstacles.

The planned trajectory has been finally validated in real-world flights in multi-waypoint racing track depicted in Figure 3c. The used quadrotor platform is based on the open-hardware and open-software Agilicious quadrotor framework [26] with the same parameters as listed in Table I. A Model Predictive Control with Incremental Nonlinear Dynamic Inversion low-level controller [27] were used to follow the planned trajectory. Figure 1 and the attached video show the flight of successfully executed trajectory reaching velocities over 60km/h. The root-mean-square error of tracking the trajectory is 0.55m, which is similar to the ones measured in [7], [26] for the same platform. This shows that our method is able to plan feasible high-speed and near time-optimal trajectories in cluttered environments.

VI. CONCLUSIONS

This paper introduced a novel hierarchical, sampling-based method that plans with an incrementally more complex quadrotor model. The method is used to find minimum-time trajectories over a given sequence of waypoints in cluttered environments, which is a problem that had not been previously solved in its entirety. We showed that the proposed method outperforms all existing baselines in finding minimum-time trajectories for cluttered environments and achieves near time-optimal solutions for scenarios without obstacles. The planned trajectories were validated in a real-world racing track at speeds over 60km/h. For future work, we plan to include perception-awareness to allow onboard gate detection and visual-inertial odometry in drone racing. We want to investigate optimality guarantees of our method and further study methods suitable for minimum-time online planning. Finally, we believe the method could be extended for different vehicles as a general framework for hierarchical planning with increasing model fidelity.

REFERENCES

[1] E. Ackermann, "Ai-powered drone learns extreme acrobatics," accessed 2.9.2021. [Online]. Available: https://spectrum.ieee.org/ai-powered-drone-extreme-acrobatics

[2] H. Moon, J. Martinez-Carranza, T. Cieslewski, M. Faessler, D. Falanga, A. Simovic, D. Scaramuzza, S. Li, M. Ozo, C. De Wagter et al., "Challenges and implemented technologies used in autonomous drone racing," Intelligent Service Robotics, vol. 12, no. 2, pp. 137–148, 2019.

[3] W. Guerra, E. Tal, V. Murali, G. Ryou, and S. Karaman, "FlightGoggles: Photorealistic sensor simulation for perception-driven robotics using photogrammetry and virtual reality," in IROS. IEEE, 2019.

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

[5] P. Foehn, D. Brescianini, E. Kaufmann, T. Cieslewski, M. Gehrig, M. Muglikar, and D. Scaramuzza, "Alphapilot: Autonomous drone racing," Robotics: Science and Systems, 2020.

[6] Z. Han, Z. Wang, N. Pan, Y. Lin, C. Xu, and F. Gao, "Fast-racing: An open-source strong baseline for SE(3) planning in autonomous drone racing," IEEE Robotics and Automation Letters, vol. 6, no. 4, pp. 8631–8638, 2021.

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

[8] S. Liu, K. Mohta, N. Atanasov, and V. Kumar, "Search-based motion planning for aggressive flight in se (3)," IEEE Robotics and Automation Letters, vol. 3, no. 3, pp. 2439–2446, 2018.

[9] D. J. Webb and J. van den Berg, "Kinodynamic rrt*: Asymptotically optimal motion planning for robots with linear dynamics," in ICRA, 2013, pp. 5054–5061.

[10] C. Richter, A. Bry, and N. Roy, "Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments," in Robotics Research. Springer, 2016, pp. 649–666.

[11] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," IEEE Transactions on Robotics and Automation, vol. 12, pp. 566–580, 1996.

[12] Y. Li, Z. Littlefield, and K. E. Bekris, "Asymptotically optimal sampling-based kinodynamic planning," The International Journal of Robotics Research, vol. 35, no. 5, pp. 528–564, 2016.

[13] M. Burri, H. Oleynikova, , M. W. Achtelik, and R. Siegwart, "Real-time visual-inertial mapping, re-localization and planning onboard mavs in unknown environments," in IROS, Sept 2015.

[14] M. W. Mueller, M. Hehn, and R. D'Andrea, "A computationally efficient motion primitive for quadrocopter trajectory generation," IEEE Transactions on Robotics, vol. 31, no. 6, pp. 1294–1310, 2015.

[15] D. Mellinger, N. Michael, and V. Kumar, "Trajectory generation and control for precise aggressive maneuvers with quadrotors," The International Journal of Robotics Research, vol. 31, no. 5, pp. 664–674, 2012.

[16] 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.

[17] B. Zhou, F. Gao, J. Pan, and S. Shen, "Robust real-time uav replanning using guided gradient-based optimization and topological paths," in ICRA, 2020, pp. 1208–1214.

[18] B. Zhou, J. Pan, F. Gao, and S. Shen, "Raptor: Robust and perception-aware trajectory replanning for quadrotor fast flight," IEEE Transactions on Robotics, vol. 37, no. 6, pp. 1992–2009, 2021.

[19] B. Penin, P. R. Giordano, and F. Chaumette, "Vision-based reactive planning for aggressive target tracking while avoiding collisions and occlusions," IEEE Robotics and Automation Letters, vol. 3, no. 4, pp. 3725–3732, 2018.

[20] S. Liu, N. Atanasov, K. Mohta, and V. Kumar, "Search-based motion planning for quadrotors using linear quadratic minimum time control," in IROS, 2017, pp. 2872–2879.

[21] S. Liu, K. Mohta, N. Atanasov, and V. Kumar, "Search-based motion planning for aggressive flight in se(3)," IEEE Robotics and Automation Letters, vol. 3, no. 3, pp. 2439–2446, 2018.

[22] K. Hauser and Y. Zhou, "Asymptotically optimal planning by feasible kinodynamic planning in a statecost space," IEEE Transactions on Robotics, vol. 32, no. 6, pp. 1431–1443, 2016.

[23] T. Zhiling, B. Chen, R. Lan, and S. Li, "Vector field guided rrt* based on motion primitives for quadrotor kinodynamic planning," Journal of Intelligent & Robotic Systems, vol. 100, 12 2020.

[24] S. M. LaValle, Planning algorithms. Cambridge university press, 2006.

[25] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, "Informed rrt*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic," in IROS, 2014, pp. 2997–3004.

[26] P. Foehn, E. Kaufmann, A. Romero, R. Penicka, S. Sun, L. Bauersfeld, T. Laengle, G. Cioffi, Y. Song, A. Loquercio, and D. Scaramuzza, "Agilicious: Open-source and open-hardware agile quadrotor for vision-based flight," 2021, under review. [Online]. Available: agilicious.dev

[27] S. Sun, A. Romero, P. Foehn, E. Kaufmann, and D. Scaramuzza, "A comparative study of nonlinear mpc and differential-flatness-based control for quadrotor agile flight," arXiv:2109.01365, 2021.